PhreePlot

Creating graphical output with PHREEQC

David Kinniburgh
David Cooper
Creating graphical output with Phreeqc

David G Kinniburgh
http://www.phreeplot.org

David M Cooper
Centre for Ecology and Hydrology, Deiniol Road, Bangor, Gwynedd, LL57 2UW, UK

June 2011
# Table of contents

1 **INTRODUCTION** ............................................................................................. 1  
   1.1 What does PhreePlot do?............................................................................ 1  
   1.2 What PhreePlot does not do....................................................................... 2  
   1.3 What you need to know before using PhreePlot......................................... 2  
   1.4 Navigating this document .......................................................................... 3  

2 **INSTALLATION**................................................................................................ 5  
   2.1 Installing PhreePlot.................................................................................... 5  
      2.1.1 Windows .............................................................................................. 5  
      2.1.2 Mac OS X machines ............................................................................. 6  
   2.2 Checking for updates ................................................................................. 6  
   2.3 Changing the console appearance............................................................. 6  
   2.4 Installing GSview ....................................................................................... 6  
      2.4.1 Launching PhreePlot from Windows Explorer or similar ...................... 7  
      2.4.2 Specifying the input file name on the command line ......................... 7  
      2.4.3 Specifying input and output filenames in PhreePlot input files .......... 7  
      2.4.4 Setting the PhreePlot environment variable ..................................... 8  
      2.4.5 Adding the path to PhreePlot to the Windows PATH setting ............. 8  
      2.4.6 Search path for files ........................................................................... 8  
      2.4.7 Ensuring that the correct databases are found .................................... 8  
   2.5 Other useful software ................................................................................. 9  
   2.6 Trouble-shooting ....................................................................................... 9  
      File conversions ................................................................................... 9  
      Problem and bug reporting.................................................................. 9  

3 **GETTING STARTED** ....................................................................................... 11  
   3.1 The command line interface and batch processing ................................... 11  
   3.2 Running the demo examples .................................................................... 11  
   3.3 The ‘pp.log’ file........................................................................................ 12  
   3.4 Examining the results of the run............................................................... 13  
   3.5 Making a working directory ..................................................................... 13  
   3.6 Getting familiar with the options ............................................................. 14  
   3.7 Using batch files to run a set of runs......................................................... 14  
   3.8 Stopping a run ......................................................................................... 14  

4 **PHREEQC BASICS** .......................................................................................... 15  
   4.1 Online Phreeqc documentation............................................................... 15  
   4.2 How Phreeqc interacts with PhreePlot ..................................................... 15  
   4.3 Thermodynamic databases ...................................................................... 16  
   4.4 Types of output produced by Phreeqc...................................................... 16  
      4.4.1 The selected_output and the user_punch data blocks....................... 17  
      4.5.1 The selected_output filename and forcing the file to be written ......... 18
4.5.2 Scope of Phreeqc keywords ................................................................. 18
4.5.3 What is sent to the SELECTED_OUTPUT file? .............................. 19

4.6 Setting up the SELECTED_OUTPUT file for input to PhreePlot .......... 20
4.6.1 Possibilities for looping of Phreeqc input files ................................. 20
- Predominance plots ........................................................................... 21
- Data-led calculations ........................................................................ 22
- Custom plots .................................................................................... 23
- Knowing which minerals might form using the given database ......... 25
4.6.2 Setting up a loop file ....................................................................... 25

4.7 Running Phreeqc without any plotting ............................................ 26

4.8 Include files ..................................................................................... 27
4.8.1 Use of ‘include’ files ....................................................................... 27
4.8.2 Supplied include files ..................................................................... 27

4.9 Using Phreeqc’s _RAW and _MODIFY keywords .......................... 27

5 PHREEPLOT INPUT AND OUTPUT FILES .............................................. 29
5.1 Input/output files .............................................................................. 29
5.1.1 Use ............................................................................................... 29
5.1.2 Difference in execution of input files between PhreePlot and Phreeqc 29

5.2 Input files .......................................................................................... 30
5.2.1 Different types of input file ............................................................ 30
5.2.2 Structure of the main input files .................................................... 31
5.2.3 The input file pre-processor ......................................................... 32
5.2.4 Exceptions to the ‘latest keyword definition overrides earlier ones’ rule ... 32
5.2.5 The colour dictionaries and other files .......................................... 32
5.2.6 Format of all input files .................................................................. 32
- Physical and logical lines ................................................................ 32
- Specifying keyword-value pairs and keyword-lists .......................... 33
- Format of keywords and their associated values .............................. 34
5.2.7 Data separators and parsing input files ......................................... 34
5.2.8 Case sensitivity of input ............................................................... 35
5.2.9 Reporting of errors in input files ................................................ 35

5.3 Tags .................................................................................................. 35
5.3.1 What are tags used for? ............................................................... 35
5.3.2 Rules for choosing tag names ....................................................... 36
5.3.3 Tag expressions .......................................................................... 36
5.3.4 Numeric tag expressions and available functions ....................... 36
5.3.5 Tags for character variables ......................................................... 37
5.3.6 System tags ................................................................................. 37
5.3.7 User-defined tags ........................................................................ 38
5.3.8 The scope of tags, their initial values and their order of evaluation ... 39
5.3.9 Examples of the use of tags ......................................................... 40

5.4 Output files ....................................................................................... 42
5.4.1 Output files produced ................................................................. 42
5.4.2 The logical switches ................................................................. 43
5.4.3 ‘log’ file (log) ............................................................................. 44
5.4.4 ‘out’ file (out) ........................................................................... 44
5.4.5 ‘track’ file (trak) ......................................................................... 45
5.4.6 ‘points’ file (pts) ........................................................................ 45
- Predominance plot calculations (ht1 only) ...................................... 45
Other calculations ................................................................. 46
5.4.7 ‘vectors’ file (vec) ............................................................ 46
5.4.8 ‘polygon’ file (pol) ........................................................... 46
5.4.9 ‘labels’ file (labelFile) ..................................................... 47
5.4.10 Other output files .......................................................... 48
5.5 Inserting plot files into Microsoft Word, Powerpoint and other software 48
5.6 Speed of computations and plotting .................................... 49

6 Running PHREEPlot .............................................................. 51
6.1 Conventions for data input ................................................... 51
6.1.1 Types of variables ......................................................... 51
6.1.2 Phreeqc notation for chemical formulae .............................. 51
6.2 PhreePlot Looping .............................................................. 52
6.2.1 Loop variables and their use ............................................. 52
6.2.2 Looping over a list of character variables ............................ 53
6.2.3 An example of the use of various looping mechanisms ............ 53
   Using the <x_axis> tag ......................................................... 53
   Using the <loop> tag ......................................................... 55
   Using a loop file ............................................................. 55
   Using the ‘simulate’ calculationMethod ................................. 55
   Looping over two variables ................................................. 56
6.2.4 Looping in multi-simulation input files - pre-loop simulations and the
   main loop ........................................................................... 57
   Introduction .......................................................................... 57
   Basic structure of a multi-simulation Phreeqc input file ............... 58
6.2.5 Dynamic switching between Phreeqc models (code) .............. 60
6.2.6 Defining the expected output in the selected output file ........ 60
6.2.7 Timing execution .......................................................... 60
6.2.8 Speeding-up calculations ................................................. 61
6.3 Possible types of calculations and plots .................................. 61
6.4 Preparing the SELECTED OUTPUT file ................................. 62
6.4.1 Normal behaviour .......................................................... 62
6.4.2 The use of the ‘headings’ identifier ................................... 62
   Controlling the plotting of individual columns ......................... 64
   Use of labels in a custom plot and the minimum text size .......... 64
6.5 Debugging ............................................................................ 65
6.5.1 Types of problem .......................................................... 65
6.5.2 Checking the return status of a Phreeqc run ....................... 66
6.5.3 General approach to debugging ...................................... 66
6.5.4 Using the debug keyword .............................................. 67
6.5.5 The most common reason for a failure to converge ............... 67
6.5.6 Changing Phreeqc’s convergence parameters ..................... 69
6.6 Interrupting execution and changing keyword values .............. 70
6.7 Running the standard Phreeqc examples ............................... 71
   6.7.1 Going round for just one iteration ................................. 71

7 Plotting Basics ..................................................................... 73
7.1 Introduction ......................................................................... 73
7.2 Types of plot ........................................................................ 73
7.3 Summary of basic plotting .................................................. 73
7.3.1 Introduction ..................................................................... 73
7.3.2 Setting up the plotting area ................................................................. 73
7.3.3 Setting up the axes scales, tick marks and grid lines ......................... 74
7.3.4 Axis numbering and annotation .......................................................... 75
7.3.5 Adding fills, lines and points .............................................................. 75
7.4 Controlling the style of lines and points (symbols) in Custom plots......... 76
7.4.1 The default style and colour for lines and points (symbols) ............... 76
7.4.2 Lines ................................................................................................. 77
  Line styles .............................................................................................. 77
  Automatic colouring of lines and points .................................................. 78
7.4.3 Points (symbols) ............................................................................... 78
7.4.4 Order of plotting of lines and points (symbols) .................................. 79
7.5 Labelling plots ...................................................................................... 79
7.5.1 Label names and label position .......................................................... 79
7.5.2 Overriding calculated label positions and angles ............................... 80
7.5.3 Legend ............................................................................................. 81
7.6 Inputting text strings ........................................................................... 81
7.6.1 Available fonts and character sizes ................................................. 81
7.6.2 Available characters and inserting ‘special’ characters ...................... 82
7.6.3 Text enhancements (bold, italic, subscript, superscript) and Greek characters 82
7.6.4 Accented and other ‘foreign’ characters - the Latin-1 encoding ........... 84
7.6.5 Non-printing characters .................................................................... 85
7.6.6 Justification ...................................................................................... 85
7.6.7 Angle ............................................................................................... 85
7.6.8 Tags in text strings .......................................................................... 85
7.7 Special PhreePlot variables or tags ....................................................... 85
7.7.1 Available tags .................................................................................. 85
7.8 Axis scaling ......................................................................................... 86
7.8.1 Auto or user-defined axis scaling ..................................................... 86
7.8.2 Secondary y axis (the 2y axis) ........................................................... 86
7.9 Controlling the properties of text, symbols, polygon fills and lines ......... 87
7.9.1 Principles ......................................................................................... 87
7.9.2 The colour palette .......................................................................... 88
7.9.3 Automatic or explicit ....................................................................... 89
7.9.4 The colour dictionaries .................................................................... 89
  Fill colours ............................................................................................ 89
  Line colours and auto line colouring ...................................................... 89
  Point colours ....................................................................................... 91
  Filled symbols with a different rim colour .......................................... 91
7.9.5 Directories for the colour dictionaries .......................................... 92
7.10 Labelling .......................................................................................... 92
7.10.1 Predominance plots ....................................................................... 93
7.10.2 Contour plots ................................................................................ 93
7.10.3 Custom plots ................................................................................. 93
7.11 Replotting without recalculating ....................................................... 94
7.11.1 The ‘replot’ option ...................................................................... 94
7.11.2 The ‘reprocess and replot’ option for predominance plots ............. 95
7.12 Adding extra lines, symbols and text ............................................... 95
7.12.1 Lines and symbols ....................................................................... 95
7.12.2 Text ............................................................................................... 96
<input> .................................................................................................. 97
7.12.3 Formatting numbers in plots – varying the number of significant figures displayed

7.12.4 Making fancy plots

8 PREDOMINANCE PLOTS

8.1 setting up a file to calculate a predominance diagram

8.1.1 The ‘grid’ and ‘ht1’ approaches

Grid approach

‘Hunt and track’ approach

8.1.2 Using the ht1.inc code to return the dominant species

8.1.3 Problems with the ‘hunt and track’ approach – ‘unclosed polygons’

8.1.4 Optimising the calculation efficiency

8.1.5 Predominance vs mineral stability diagrams

8.1.6 Using ht1minerals.inc to determine the minerals present

8.2 The ‘grid’ approach

8.3 The ‘Hunt and track’ approach

8.3.1 Strategy

8.3.2 Details of the ‘hunt and track’ algorithm

Applying the concepts of hunting and tracking at the practical level

Definitions of ‘predominance’ and mineral stability

8.3.3 Failure of the ‘hunt and track’ approach

8.4 Feasible domains and the preparation of Eh (pe) -pH diagrams

8.4.1 General principles

8.4.2 Domain tags - avoiding speciation calculations

8.4.3 Speciation failure when there is not enough reactant present

8.5 Choice of the resolution of the plot

8.6 Monitoring the progress of a ‘hunt and track’ run

8.7 Plotting and Replotting

8.8 Modifying the appearance of predominance plots

8.9 Adding lines and points to a predominance plot

8.10 Controlling the labelling of plots and the plotting of fields

8.11 Why do I not see methane gas when using llnl.dat?

8.12 Failure to complete a predominance diagram

9 CONTOUR PLOTS

9.1 What are contour plots?

9.2 Implementation

9.2.1 Generating the contour data

9.2.2 Choosing the contour levels

9.2.3 Types of contour plot

9.2.4 Changing the appearance of the contour lines

9.2.5 Colouring the plot

9.2.6 Labels and Legend

9.2.7 Flow of data

9.2.8 What if Phreeqc fails?

9.3 A simple example

9.4 Some details of the data processing

9.4.1 Algorithm
12.6.1 Simple example ................................................................................ 157
12.6.2 Automatic updating of parameter values in an input file .......... 158
12.7 Interrupting or stopping the fitting .................................................. 158
12.8 Forcing relations between parameter values .................................. 158
12.9 Output files.......................................................................................... 159
12.10 Response in the event of a failure of Phreeqc to converge .............. 159
12.11 Plotting the results of the fitting....................................................... 159
12.11.1 Plots of observed and calculated results ....................................... 159
12.11.2 Contour plot of the residual sum of squares ............................... 160
12.12 Simulations ....................................................................................... 161
12.13 Root finding – solving equations, adding constraints.................... 162

13 The input file pre-processor.................................................................. 165
13.1 Use of the pre-processor ................................................................. 165

14 Keywords ............................................................................................. 167
14.1 Summary of available keywords....................................................... 167
14.2 Conventions....................................................................................... 170
14.3 Keyword description ........................................................................ 171
   all.................................................................................................................171
   axisLineColor............................................................................................171
   axisLineWidth ..........................................................................................172
   axisNumberColor.......................................................................................172
   axisNumberSize ........................................................................................172
   axisTitleColor ..........................................................................................172
   axisTitleSize ...........................................................................................173
   backgroundColor.......................................................................................173
   beep .............................................................................................................173
   blockRangeColumn .................................................................................174
   calculationMethod ..................................................................................174
   calculationType .......................................................................................175
   changeColor .............................................................................................176
   characterTags..........................................................................................177
   checkForUpdate.......................................................................................177
   colorModel ..............................................................................................178
   contourDashesPerInch ............................................................................179
   contourFillColor .....................................................................................179
   contourLabelColor ..................................................................................179
   contourLabelFigs ....................................................................................180
   contourLabelFont ....................................................................................180
   contourLabelSize ....................................................................................181
   contourLineColor ....................................................................................181
   contourLineType ......................................................................................181
   contourLineWidth ....................................................................................182
   contourOptions .......................................................................................182
   contours .................................................................................................183
   contourShiftLabel ..................................................................................184
   contourZvariable ....................................................................................186
   convertLabels ..........................................................................................186
   customLoopManyPlots ............................................................................187
   customXcolumn ......................................................................................187
dashesPerInch, dashesPerInch2y ................................................................. 188
database ................................................................................................. 188
databaseVersion ..................................................................................... 189
dataFile .................................................................................................... 189
dataSeparators ........................................................................................ 190
dateDatabase .......................................................................................... 192
debug ....................................................................................................... 192
dependentVariableColumnObs ............................................................... 193
dependentVariableColumnCalc ............................................................... 194
domain ..................................................................................................... 194
dominant ................................................................................................. 195
eps ........................................................................................................... 195
epsi .......................................................................................................... 196
extradat ................................................................................................... 197
extraSymbolsLines .................................................................................. 198
extraText .................................................................................................. 200
fillColorDictionary .................................................................................. 201
FIT .......................................................................................................... 202
fitAdjustableParameters ......................................................................... 202
fitConvergenceCriterion ........................................................................ 202
fitFiniteDiffStepSize .............................................................................. 203
fitLogParameters .................................................................................... 203
fitLowerParameterValues ....................................................................... 204
fitMaxIterations ..................................................................................... 204
fitMethod ............................................................................................... 205
fitnpt ...................................................................................................... 205
fitParameterNames .................................................................................. 206
fitParameterValues .................................................................................. 206
fitStepSize ............................................................................................. 206
fitUpperParameterValues ....................................................................... 207
fitWeightingMethod ............................................................................... 207
font ......................................................................................................... 208
gridColor ............................................................................................... 210
gridDashesPerInch .................................................................................. 211
gridLines ............................................................................................... 211
gridLineType .......................................................................................... 211
info ........................................................................................................ 212
initialValue ............................................................................................. 213
jobTitle .................................................................................................... 213
jpg .......................................................................................................... 213
labelColor ............................................................................................... 214
labelEffort .............................................................................................. 214
labelFile ................................................................................................. 215
labels ...................................................................................................... 215
labelSize ............................................................................................... 216
legendBox ............................................................................................. 217
legendTitle ............................................................................................. 217
legendTextSize ...................................................................................... 218
lineColor, lineColor2y ............................................................................ 218
lineColorDictionary ................................................................................. 219
lines, lines2y ......................................................................................... 219
printScreenFrequency................................................................. 248
ps ............................................................................................. 248
pts .............................................................................................. 249
pxdec .............................................................. 249
pxmajor ................................................................................. 249
pxmax .................................................................................... 250
pxmin .................................................................................... 250
pxminor ................................................................................. 250
pydec, p2ydec ................................................................. 251
pymajor, p2ymajor ........................................................... 251
pymax, p2ymax ................................................................. 251
pymin, p2ymin ................................................................. 252
pyminor, p2ymajor ........................................................... 252
resolution .............................................................................. 252
restartColorSequence ....................................................... 254
rimColor .............................................................................. 254
rimFactor .............................................................................. 254
screen .................................................................................... 255
selectedOutputFile ............................................................ 255
selectedOutputLines .......................................................... 256
simplify .................................................................................. 257
skip ......................................................................................... 258
SPECIATION ................................................................. 258
speciationProgram ............................................................ 258
speciationProgramVersion .................................................. 259
startTemperature .............................................................. 259
stopOnFail ................................................................. 259
tickColor .................................................................................. 259
tickSize .................................................................................... 260
trackSymbolColor .................................................................. 261
trackSymbolSize .................................................................. 262
trk ......................................................................................... 262
units ....................................................................................... 263
unrecognisedKeywordIsFatal ............................................... 263
updateFitParameters .......................................................... 264
useLabelsFile ................................................................. 264
useLineColorDictionary ....................................................... 264
vec ......................................................................................... 265
weightColumn ................................................................. 265
writeAllInputFiles .............................................................. 266
writePlaceholder .............................................................. 266
xaxisLength ......................................................................... 266
xmax ..................................................................................... 267
xmin ..................................................................................... 267
xoffset ................................................................................... 267
xtitle ..................................................................................... 267
yaxisLength ......................................................................... 268
ymax ..................................................................................... 268
ymin ..................................................................................... 269
yoffset ................................................................................... 269
yscale ................................................................................... 269
15 EXAMPLES ........................................................................................................ 271
Predominance plots (grid approach) ....................................................................... 272
1 Fe-H$_2$O (grid approach) .................................................................................. 274
2 Cu-S-C ('island' found with 'grid') .................................................................... 278
Predominance plots (ht1 approach) ......................................................................... 281
3 Fe-H$_2$O ('Hunt and track' approach) ................................................................. 282
4 Fe-H$_2$O: close-up (predominance criterion) ..................................................... 284
5 Fe-H$_2$O: close-up (stability criterion) ............................................................... 286
6 Fe-H$_2$O (pe scale) ........................................................................................... 288
7 Fe-H$_2$O (Eh scale) ........................................................................................... 290
8 As-O$_2$(g)-H$_2$O .............................................................................................. 292
9 As-O$_2$(g)-H$_2$O (sub-dominant) ...................................................................... 294
10 Fe-As-C-S ....................................................................................................... 296
11 Fe-As ................................................................................................................ 300
12 Fe-As-Hfo (ht1.inc) ......................................................................................... 302
13 Fe-As-Hfo (ht1c.inc) ....................................................................................... 304
14 Fe-S-As (without sorption) .............................................................................. 306
15 Fe-S-As (low Fe, without surface speciation) .................................................. 310
16 Fe-S-As (goethite, CD-MUSIC) ....................................................................... 314
17 Fe-As-S (high Fe) ........................................................................................... 318
18 Fe-CO$_2$-SO$_4$-H$_2$O with sample points ....................................................... 322
19 Fe-Ni-S ............................................................................................................ 326
20 Fe-Zn-C-H$_2$O (HFO) ..................................................................................... 330
21 Fe-Zn-C-H$_2$O (HFO) ..................................................................................... 334
22 Ca-Fe-Na-X-HFO (adsorption and ion exchange) ......................................... 338
23 Acid mine drainage ......................................................................................... 342
24 AMD (carbon) ................................................................................................. 346
25 Ca-Mg-Zn-CO$_3$ ............................................................................................ 350
26 U-CO$_3$ ........................................................................................................... 354
27 Ca-Mg-CO$_3$ at high T .................................................................................... 356
28 P-Ca-Mg-CO$_3$(aq) ......................................................................................... 358
29 P-Ca-Mg-CO$_3$ (with solids) .......................................................................... 360
30 Mn-CO$_2$-H$_2$O (no minerals) ....................................................................... 364
31 Mn-H$_2$O (with minerals) ............................................................................... 366
32 Mn-CO$_2$-H$_2$O ............................................................................................. 368
33 Pu-F-H$_2$O ....................................................................................................... 370
34 U-C-H$_2$O (wateq4f.dat) ................................................................................ 372
35 U-C-H$_2$O (NAPSI) ....................................................................................... 374
36 U-C-H$_2$O (llnl.dat) ....................................................................................... 376
37 U-Fe-C-H$_2$O ................................................................................................. 380
38 U-Fe-C (risk colours) ..................................................................................... 384
39 U-H$_2$O (redox state) .................................................................................... 388
40 U-F-P (U) ........................................................................................................ 390
41 U-F-P (F) ........................................................................................................ 394
42 U-F-P (P) ........................................................................................................ 396
43 Cu-S-C ('island' not found with 'ht1') ............................................................... 398
44 Cu-S-C (simplification factor = 1) ................................................................... 400
45 Cu-S-C (simplification factor = 3) ................................................................... 404
46 Cu-EDTA-H$_2$O ............................................................................................. 408
47 Ca-F-P-C-H$_2$O ............................................................................................. 412
48 Ni-S-C-H$_2$O .................................................................................................. 414
49 Zn-S-C-H$_2$O ................................................................................................. 418
## References

Contour plots.................................................................................................................... 543
Appendix 4. The Standard and Latin-1 character sets........................................................570
Appendix 3. Symbol numbers and names..........................................................................566
Appendix 2. Thermodynamic databases ............................................................................553
Appendix 1. Glossary of terms...........................................................................................551
Acknowledgements............................................................................................................549
Fitting models to data .......................................................................................................520
Species plots.................................................................................................................. ....502
Custom plots.....................................................................................................................439
Gibbsite solubility vs pH.................................................................................................440
Acid titration of groundwater (using ‘REACTION’) ......................................................444
Acid titration of groundwater (using PhreePlot looping)............................................446
SPECIATION....................................................................................................................448
Kd’s for trace metals as a function of pH........................................................................454
Cd speciation vs pH.........................................................................................................458
Zn-HFO: % sorption vs pH curves..................................................................................462
Zn-HFO: Surface speciation............................................................................................466
As-HFO: reduction in surface area..................................................................................470
CD-MUSIC: As(III) adsorption on goethite.................................................................474
CD-MUSIC: As(V) adsorption on goethite .....................................................................476
Kinetics of pyrite oxidation............................................................................................480
Kinetics of quartz dissolution.........................................................................................484
Symbols and lines ...........................................................................................................488
Varying symbols and lines in plots................................................................................490
Text..................................................................................................................................494
Simple Phreeqc looping...................................................................................................496
Phreeqc mineral species.................................................................................................500
Species plots.................................................................................................................. ....502
Cd speciation vs pH (species plot)................................................................................404
Cd speciation vs pH (log species plot)............................................................................506
U species plot (oxidizing)..............................................................................................508
U species plot (reducing)..............................................................................................510
Carbon speciation vs pH...............................................................................................512
CD-MUSIC: As(V) surface speciation on goethite......................................................514
Test plot output formats.................................................................................................518
Fitting models to data .......................................................................................................520
Langmuir isotherm (1)....................................................................................................522
Langmuir isotherm (n)....................................................................................................526
Langmuir isotherm (pre-processor).............................................................................532
Ni sorption by goethite.................................................................................................536
As(V) sorption on hydrous ferric oxide ..........................................................................540
Contour plots....................................................................................................................543
Contour two metals at three resolutions.........................................................................544
References ......................................................................................................................547
Acknowledgements........................................................................................................549
Appendix 1. Glossary of terms........................................................................................551
Appendix 2. Thermodynamic databases .........................................................................553
Appendix 3. Symbol numbers and names.......................................................................566
Appendix 4. The Standard and Latin-1 character sets....................................................570

## 16 The WATEQ4F.DAT DATABASE

50  Cd-S-C-H$_2$O...........................................................................................................422
51  Pb-S-C-P-H$_2$O.......................................................................................................426
52  Sb-S..........................................................................................................................430
53  Se-S ..........................................................................................................................434
54  Clay mineral stability diagram...............................................................................436
55  Gibbsite solubility vs pH........................................................................................440
56  Acid titration of groundwater (using ‘REACTION’) ..............................................444
57  Acid titration of groundwater (using PhreePlot looping).......................................446
58  SPECIATION ............................................................................................................448
59  Kd’s for trace metals as a function of pH................................................................454
60  Cd speciation vs pH..................................................................................................458
61  Zn-HFO: % sorption vs pH curves............................................................................462
62  Zn-HFO: Surface speciation......................................................................................466
63  As-HFO: reduction in surface area...........................................................................470
64  CD-MUSIC: As(III) adsorption on goethite..........................................................474
65  CD-MUSIC: As(V) adsorption on goethite.............................................................476
66  Kinetics of pyrite oxidation......................................................................................480
67  Kinetics of quartz dissolution....................................................................................484
68  Symbols and lines ....................................................................................................488
69  Varying symbols and lines in plots..........................................................................490
70  Text............................................................................................................................494
71  Simple Phreeqc looping...........................................................................................496
72  Phreeqc mineral species...........................................................................................500
73  Cd speciation vs pH (species plot)..........................................................................504
74  Cd speciation vs pH (log species plot).....................................................................506
75  U species plot (oxidizing)........................................................................................508
76  U species plot (reducing)........................................................................................510
77  Carbon speciation vs pH........................................................................................512
78  CD-MUSIC: As(V) surface speciation on goethite................................................514
79  Test plot output formats...........................................................................................518
80  Langmuir isotherm (1)............................................................................................522
81  Langmuir isotherm (n)............................................................................................526
82  Langmuir isotherm (pre-processor).......................................................................532
83  Ni sorption by goethite............................................................................................536
84  As(V) sorption on hydrous ferric oxide ....................................................................540
85  Contour two metals at three resolutions..................................................................544
References ......................................................................................................................547
Acknowledgements........................................................................................................549
Appendix 1. Glossary of terms........................................................................................551
Appendix 2. Thermodynamic databases .........................................................................553
Appendix 3. Symbol numbers and names.......................................................................566
Appendix 4. The Standard and Latin-1 character sets....................................................570
1 Introduction

1.1 What does PhreePlot do?

PhreePlot makes it possible to produce certain types of high quality geochemical plots using Phreeqc (Parkhurst and Appelo, 2013). Phreeqc is a popular and freely-available program for calculating geochemical speciation and mass transport. It has a very flexible input structure that makes it easy to customise the type of calculations performed. This includes the ability to modify the thermodynamic database used. It also has a very flexible mechanism for outputting the results of these calculations which makes it straightforward for programs such as PhreePlot to interface with it. In short, it is an excellent geochemical calculator.

Phreeqc originally did not include any charting options or any mechanism for repeating blocks of calculations with a DO ... ENDDO-type structure. It was these two missing features that led to the development of PhreePlot.

An early fork of Phreeqc, Phreeqc for Windows (Post, 2011) incorporated some simple charting and this has been extended and is now fully incorporated in the latest Windows version of Phreeqc (Version 3). Phreeqc is also now available as a modular version as iPhreeqc (Charlton and Parkhurst, 2011). This includes most of the functionality of the batch version and means that Phreeqc can be incorporated into programs written in C++, Fortran as well as in interpreted languages such as Python and R (there is a ‘phreeqc’ package). It is also available as a Windows COM server and so can be linked to spreadsheets and databases. These options allow fine-grained programming giving the developer a large degree of control over the calculations made. These programming languages also usually have access to libraries of other useful functions to facilitate programming.

So what does PhreePlot do? PhreePlot sits on top of Phreeqc and makes it relatively straightforward to do certain kinds of repetitive Phreeqc calculations, the type of calculations that are often needed to make a plot. It does this while maintaining the basic structure of Phreeqc input files. PhreePlot uses tags placed within the Phreeqc input files to identify places where substitutions are to be made and has several mechanisms to control the values substituted and the looping done. It also contains software for generating Postscript plot files. If Ghostscript is installed then automatic conversion to pdf, png, jpg, ai, eps and epsi formats is possible. With this basic functionality, PhreePlot can be used to generate predominance diagrams, contour plots, and to fit observations to Phreeqc models.

One of the motivations for developing PhreePlot was to develop the ability to calculate predominance and mineral stability diagrams, often known as Eh-pH or pe-pH diagrams, based on a full chemical speciation. This numerical approach is different from the ‘equation-based’ approach typically found in textbooks and employed in software such as the Geochemist’s Workbench (Bethke, 2005). One of the advantages of the numerical approach is that reactions that do not obey the classical speciation model can also be included in the diagrams. This includes adsorption, ion exchange and co-precipitation reactions. Since a full speciation is undertaken, the impact of all interactions are automatically taken into account and the results are fully consistent with those of other speciation and reaction path calculations undertaken using the same database.

The intellectual ‘penalty’ with this approach is that realistic and solvable reaction paths have to be devised to map the whole of the activity space of interest. The practical penalty is that the large number of computations required means that the numerical approach is significantly slower than the analytical approach.
We have developed two ways of constructing predominance diagrams: the ‘grid’ approach simply calculates the speciation at all points on a grid and while the ‘hunt and track’ attempts to track the boundaries between predominant fields. This approach usually requires considerably fewer speciation calculations than the ‘grid’ approach to produce a diagram of equivalent quality but it makes the assumption that all fields are interconnected and can be reached in some way by tracking along their boundaries. This is not necessarily the case and so fields can be missed\(^1\). This appears to be relatively uncommon in practice but is nevertheless an important limitation of the ‘hunt and track’ approach. The ‘grid’ approach makes no such assumption and so should always be the final arbiter.

A somewhat different way of looking at predominance diagrams is to ‘contour’ the data for some diagnostic variable such as the total dissolved amount or concentration of some element. Phreeqc is very flexible in the way that it can define its output and this is translated into a great flexibility in the variables that can be contoured.

PhreePlot also makes use of its looping capability to fit chemical models to data. This can be used, for example, to derive log K values from experimental data.

PhreePlot makes use of the PSPL0T Postscript library (Kohler, 2005) to produce high quality Postscript plot files. These can be edited, printed or converted to other graphical formats using various software packages, e.g. GPL Ghostscript/GSview, Adobe Illustrator, Inkscape, CorelDRAW, GIMP, etc. As of February 2018, Ghostscript is optionally included in the PhreePlot distribution and if selected to be installed, does not need to be separately installed.

1.2 WHAT PHREEPLOT DOES NOT DO

The iteration scheme available in PhreePlot is quite limited and follows a fixed format: once through zero or more ‘initialization’ or pre-loop simulations and then a four-loop iteration scheme (x, y, z and character). If something else is wanted, then this will have to be programmed specifically using the iPhreeqc module as mentioned above (Müller et al., 2011).

PhreePlot leaves getting the geochemistry right and the necessary output produced up to the user. You will have to be reasonably proficient with Phreeqc before attempting to use PhreePlot. We provide many demos to help get you started.

The plotting is hopefully of a high quality but the types of plots produced are rather limited. If more sophisticated plots are wanted, then some other plotting program must be used. There are many. Some of these such as R, Python, Matlab and Mathematica allow external programs to be run and so could still use PhreePlot to generate the data with all the communications being done via files.

The interface to PhreePlot is via a console or command prompt. There is no GUI. You will need to use a text editor to edit the input files.

1.3 WHAT YOU NEED TO KNOW BEFORE USING PHREEPLOT

PhreePlot currently runs on the Windows operating system. It contains iPhreeqc, an embedded version of Phreeqc, and so does not need another copy of Phreeqc. However, it will be necessary to have an up-to-date version of Phreeqc available for the documentation, release notes, licence conditions and other information.

The plotting part of PhreePlot uses as input the output from Phreeqc, as communicated through its main ‘selected output’ file (number 0). Phreeqc provides very versatile facilities for writing these files. Therefore it is necessary to be fairly competent at running Phreeqc in the normal way and of manipulating the selected output file(s). If you are not, follow the documentation provided with Phreeqc and study the examples included in that manual carefully, initially choosing the one closest to your needs as a template. The demos included with Phree-
Plot also provide examples and templates for many types of plot.

Since Phreeqc includes a BASIC interpreter for customising output to the selected output file(s), some knowledge of BASIC programming is useful. A careful study of the demo examples provided here should give an introduction to this and will provide example of the link to PhreePlot.

1.4 NAVIGATING THIS DOCUMENT

This Guide is primarily intended for online browsing not for printing. There are several aids to help navigate around the document using Adobe Reader. Some tips for navigating the document are given below though these may vary slightly depending on the version of Adobe Reader used.

A roadmap to the documentation can be seen by enabling the bookmarks for this document in Adobe Reader. If these bookmarks cannot be seen, toggle them on by clicking the bookmark icon or on the left-hand side of the Reader window or go through the menu system and tick the View|Navigation Panels|bookmarks item.

Various hyperlinks are available within the Guide including links to all of the keywords used in the text. These link to the corresponding description in the Keywords section (Section 14). All links appear in blue text and are underlined. Hyperlinks are indicated by the cursor changing to a pointed finger when hovering over the link.

You can navigate over your navigation history in one of three ways: (i) use the toolbar icons; (ii) use the Alt+Left arrow to go backwards; Alt+Right arrow navigates forward again, or to go backwards (iii) use the Previous View item from the right click (context) menu. If the toolbar icons are not already visible in Adobe Reader, activate them with the Tools|Customize Toolbars|Page Navigation Toolbar dialog or similar.
2 Installation

2.1 Installing PhreePlot

The latest version of PhreePlot can be downloaded from http://www.phreeplot.org. The filename for the Windows installer should have the format, setup-pp-win-yymmdd-zzzz.exe where yymmdd is the date of the PhreePlot build and zzzz is the Phreeqc build number.

2.1.1 Windows

PhreePlot is available in both 32-bit (win32) and 64-bit Windows (x64) versions. The Win32 version will run in both 32- and 64-bit versions of Windows whereas the 64-bit version will only run in native 64-bit versions of Windows. The 32-bit and 64-bit executables both have the same file name, pp.exe, so that the batch files will work for both versions in a straightforward manner. The banner sent to the screen and log file will indicate the version of PhreePlot actually being used.

The installer should be executed and PhreePlot installed to your preferred directory (called a ‘folder’ in Windows parlance). The default directory is a PhreePlot sub-directory in your application data directory. This can be changed during installation. The installation will create a series of sub-directories in which the PhreePlot files will be installed. However, the program executable (pp.exe) will always be installed in the Program Files directory, e.g. C:\Program Files\PhreePlot\ (or in C:\Program Files (x86)\PhreePlot\ when installing the 32-bit version of PhreePlot on a 64-bit system). You have no control over this.

The following files and directories will be created:

```
\system
| \demo
| \doc
where
```

```
\system
| pp.set User-defined initial settings and preferences
| override.set any override settings.
| htl.inc Phreeqc USER_PUNCH code to calculate predominance diagrams.
| htlc.inc As above but combines all adsorbed species for a given sorbent-element combination.
| More inc and other ‘system’ and database files
\demo A directory containing examples (ppi and associated files), one or more ppi files per subdirectory (see the Examples Section)
\doc PhreePlot.pdf This user guide
| changes.pdf List of changes made.
```

Each of the demo sub-directories contains a specific example, or collection of related examples. These include a PhreePlot input file and any other input files required. Input filenames generally have the extension .ppi though this is not necessary. However, if ppi is associated with the PhreePlot executable during installation, as recommended, then double clicking a ppi file in Windows Explorer or similar will automatically execute it with PhreePlot. This is the easiest way to run PhreePlot.

Spaces in input filenames should work but if in doubt, enclose the filename in quotes (or preferably, avoid!). The search path for the input file follows the normal operating system conventions although as with most PhreePlot searches, PhreePlot will also search the system directory. In batch files, the current working directory is the directory from which the batch
file originated. Use the ‘change directory’ (cd ...) command in a batch file to change to a
new working directory if required.

If PhreePlot is having trouble finding the input file, use the full path name including the
drive.

Output files are automatically put in the same directory as the input file using the input file
name minus the extension as the root.

PhreePlot also needs to know where to find certain files such as Ghostscript files. It does not
use the Windows registry for this and so some file paths need to be set explicitly. The steps
outlined below should be taken to ensure that PhreePlot knows where to find the necessary
files.

Installing PhreePlot under Windows requires Administrator rights. You will be asked to
accept the installation.

The installer should be executed and PhreePlot installed to your preferred directory.

The PhreePlot executable, pp.exe, does not require Administrator privileges to run. If set,
these should be turned off by opening the Properties dialog for the pp.exe file (right-click the
file), opening the Compatibility tab and unticking the ‘Run this program as an administrator’
tick box. The same should be done for all users by clicking the ‘Show settings for all users’ but-
tton. This should prevent UAC prompts when running PhreePlot.

Batch files such as demo.bat also do not need Administrator privileges to run but will need
permission to ‘Read & execute’ (set under the Security tab of the file’s properties).

2.1.2 Mac OS X machines

PhreePlot has been reported to work on Apple Mac’s with Windows emulators such as Wine
and Parallels.

2.2 CHECKING FOR UPDATES

The latest version will always be available for download from the PhreePlot website,
www.phreeplot.org. If you would like to be emailed about updates, send a message to sub-
scribe@phreeplot.org with the word ’subscribe’ in the Subject line. To unsubscribe, put the
word ‘unsubscribe’ in the Subject line (not case sensitive).

If the keyword, checkForUpdate is set to true, then PhreePlot will automatically check the
server to see if a more recent version is available. This uses the wget program. The second
parameter for the checkForUpdate keyword sets the minimum time gap (in days) between
checking. Setting this to 1 means that the server will be checked once every day whereas set-
ing it to 0 will mean that the server will be checked every time PhreePlot is executed.

2.3 CHANGING THE CONSOLE APPEARANCE

The default appearance of the console is white text on a black background. If you want to
change this, right click on an existing console and select Defaults. This gives the option of
changing the default window size and position, the text font and the foreground and back-
ground colours, e.g. to black text on a white background.

These settings should become the default for all new console windows. If only a temporary
change is required, use the console Properties dialogs instead.

2.4 INSTALLING GSVIEW

As of 2018, PhreePlot optionally comes with Ghostscript installed but it is possible to install
it yourself in the normal way. If you want to use your installed version, you must point the
pdfMaker setting to the appropriate path (see pdfMaker). It is also useful to install a native ps
viewer such as **GSview**.

**GSview** 5.0 is available from Ghostgum Software Pty Ltd at [http://pages.cs.wisc.edu/~ghost/gsview/](http://pages.cs.wisc.edu/~ghost/gsview/). After downloading, run **GSview** and under Options|Advanced Configure|Ghostscript DLL enter `<%PHREEPLOT_PATH%>/gsdll<nn>.c.dll` where `<%PHREEPLOT_PATH%>` is the long-hand description of the folder containing pp.exe (**GSview** does not accept environment variables here), `<nn>` is either 32 or 64 depending on which version of **PhreePlot** you have installed. This should point to the folder that contains your pp.exe executable. If this does not work, install **Ghostscript** in the usual way and use these settings for **GSview**.

If **PhreePlot** has been installed correctly, typing

```
pp -v
```

from a console should show some information about the current version of **PhreePlot** including its release date, whether the 32- or 64-bit versions is running and whether **Ghostscript** is installed, and if so, its source and version.

Running the `demo\test\test.ppi` file will indicate whether your **Ghostscript** setup has been successful. If successful, this should produce `ps`, `pdf`, `png`, `eps`, `epsi` and `jpg` plot files.

### 2.4.1 Launching PhreePlot from Windows Explorer or similar

If the `.ppi` extension has been associated with the **PhreePlot** executable, then double clicking a `.ppi` input file in Widows Explorer or similar should launch **PhreePlot**. This is probably the easiest way to run **PhreePlot**.

### 2.4.2 Specifying the input file name on the command line

**PhreePlot** expects an input file to be given on the command line following `pp`. The usual file naming conventions apply in terms of the use of quotes, .. (parent directory). Windows filenames are not case sensitive and respect both forward and backward slashes as separators. It can get complicated when batch files and changes of directory are used and **PhreePlot** may not be able to find the required input file. In such cases, launch from a console window and use the full pathname. In Windows, the environment variable `%PHREEPLOT_PATH%` should point to the folder containing the executables so `%PHREEPLOT_PATH%\pp.exe` should always find the executable and provides a safe way of specifying it in a console and in batch files. Note that the quotes are necessary here because of the space in the Program Files directory and the possibility of command line arguments.

### 2.4.3 Specifying input and output filenames in PhreePlot input files

Various file paths can be specified in **PhreePlot** input files but somewhat stricter requirements than above apply when specifying these file paths.

File paths should not contain a `+` sign even though this is legal in Windows. **PhreePlot** uses a system shell command to copy various files and your system may interpret an unquoted `+` sign as the beginning of another file to copy.

File paths can in principle contain any characters that are compatible with normal operating system rules (Windows disallows `/ ? < > \ : * | `). In order to avoid having to escape characters, it is wise to also ignore parentheses, brackets and the ampersand. Other characters including the `+` sign, comma, semi-colon, percent sign and space are best avoided. In other words, keep it simple for an easy life!

File paths are not case sensitive in Windows so any mixture of cases will do. However, whatever is entered is preserved in **PhreePlot**. **PhreePlot** itself tends to use lowercase filenames with the exception that chemical elements follow their normal notation.

### 2.4.4 Setting the PhreePlot environment variable

The environment variable, `PHREEPLOT`, must be set before **PhreePlot** will work. The installer
should set this to the PhreePlot directory that you specify during installation, i.e. the root
directory containing the system, demo and doc directories. In Windows, this could be your
AppData directory (default) or a location that you chose during installation such as C:\Phree-
Plot. Note that there is no trailing backslash.

The environment variable, PHREEPLOT_PATH, is also set to the location of the PhreePlot exec-
cutable, pp.exe, e.g. C:\Program Files\PhreePlot.

You can check that this has been done correctly by typing ‘set PhreePlot’ in a console win-
dow. This will return the two directories currently set.

Once set, the ‘demo directory’ for example could be referred to as “%PHREEPLOT%\demo” in
batch files.

2.4.5 Adding the path to PhreePlot to the Windows PATH setting

The installer should automatically add the path to the PhreePlot executable (pp.exe) to your
PATH during installation.

2.4.6 Search path for files

The search path for all input and data files is, in order of checking: (i) the specified filepath;
(ii) the current directory; (iii) the PhreePlot ‘system’ directory and its sub-directories, and (iv)
the path, if any, defined by a <file> tag.

If in doubt, include the full path to be sure. Put in quotes if there is a space in the name.

2.4.7 Ensuring that the correct databases are found

The database keyword points to the location of the thermodynamic database file to use. This
should be a standard Phreeqc-format database file. Several of these are included in the normal
Phreeqc distribution and have been copied to the PhreePlot system directory for conven-
ience. Check the Phreeqc website (http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/) for
the latest files. Several other public-domain databases are also provided here (see Appendix 2).

Providing that the database files are kept in the system directory, they should be able to be
located by PhreePlot from their filenames alone, e.g. wateq4f.dat, since the system directory
is automatically included in the search path. Some demo examples specify other freely-availa-
ble databases that are not included in the PhreePlot distribution. This is usually because
obtaining them requires some form of registration.

The correctness of the results of geochemical calculations is directly related to the quality of
the associated thermodynamic databases. It is entirely your responsibility to make sure that the
databases used are adequate for the purposes for which you are using them - caveat emptor.

Keeping a critical eye on the quality of the databases used is an important part of geochemical
modelling. Other caveats, notably that thermodynamic equilibrium is not always, even rarely,
achieved should also be borne in mind. This is particularly true of dissolution and precipita-
tion reactions.

2.5 OTHER USEFUL SOFTWARE

Each to their own, but we have found the following software to be useful when working with
PhreePlot:

Notepad++ a free and highly capable text editor that includes syntax highlighting for a
large number of file types. The normal Phreeqc installations now come
with a file to colour Phreeqc keywords in pqi and ppi files (http://note-
pad-plus-plus.org/).

7-zip free file compression utility that is efficient and easy to use (http://www.7-
zip.org/).
xplorer2  dual pane Windows Explorer that makes a great way for launching PhreePlot files and for viewing the graphical and text files produced (plus many of the other things you have to do for file management) (http://zabkat.com/index.htm).

Able Batch Converter  batch conversion of Postscript files to other image formats including autocropping and resizing (http://www.graphicregion.com).

CoPlot  Flexible and powerful scientific plotting package (http://www.cohort.com/coplot.html).

R  Powerful and well-supported open-source working environment for data processing including flexible, high quality graphics (http://www.r-project.org/).

Inkscape  Open-source vector graphics editor capable of manipulating Postscript files and exporting SVG-format files (http://www.inkscape.org/).

It is useful to have access to software that can edit native ps files so that other features can be added and label positions etc changed. Inkscape mentioned above is one such editor.

Although PhreePlot does contain some plotting functionality, it is quite limited in what it can do and is not intended to replace a proper scientific plotting package. The ASCII-format output files are designed to be read by more powerful plotting and data analysis packages including those mentioned above.

2.6 TROUBLE-SHOOTING

File conversions

File conversions from ps to other formats can be automatically carried out by Ghostscript under the control of PhreePlot. If this is not working, make the following checks. If all else fails, read in the ps file into GSview and make the required conversions in the normal GSview way.

Once installed correctly, the demo examples should run (see Section 3.2).

Problem and bug reporting

Contact David Kinniburgh (david@phreeplot.org).
3 Getting started

3.1 THE COMMAND LINE INTERFACE AND BATCH PROCESSING

Like the batch version of Phreeqc, PhreePlot can be run from a console or executed via a shortcut providing the following format is given:

```
pp input_filename [otherpp.set]
```

where `input_filename` is the name of a valid input file (see Section 5.2). If only a partially qualified filename is given, care must be taken to ensure that it is sufficient for the file to be found (Section 2.4.2). We have adopted the convention of using the ppi extension for input filenames. The optional `otherpp.set` is the name of a settings file to use instead of the default `pp.set` from the system directory. Normally this second parameter is left blank and the default used.

Output will be sent to the screen and to various output files. If `input_filename` contains blanks, embed it in quotes.

If the `ppi` extension is associated with PhreePlot, as recommended, then the easiest way of running a PhreePlot input file is to double click it in an Explorer window.

Collections of the above-type statements may be collected together in a batch file and run as one job. The `demo.bat` file included in the distribution is one such example. This is the mechanism for plotting multiple curves from different runs in a single custom plot – the output files are created in the initial runs and then the last run does all the plotting using the `extradat` keyword to load the output from the earlier runs.

Using the `override.set` file with `calculationMethod` 2 can make global changes to the output from a set of already-calculated files without changing the individual `ppi` files.

Input files should be prepared with a standard text editor. Notepad will do but many better editors exist. It is useful to have an editor that automatically checks for and loads updated files. PhreePlot is not interactive (no GUI) but with a little effort in setup, it can be made to work quite efficiently.

Ordinary Phreeqc input files can be run by just adding the line `CHEMISTRY` to the beginning of the input file (otherwise PhreePlot will interpret this input as PhreePlot keywords). Adding `all T` or `debug 2` just before this will cause the `*.all` file to be created which will contain a copy of all the Phreeqc output.

3.2 RUNNING THE DEMO EXAMPLES

Providing the paths have been set correctly as described above, launching the `demo1.bat` file from the `\demo` directory should begin the calculations. This can be done either by double clicking on the `demo1.bat` file in a Windows Explorer-type window, or by opening a console and executing it from there (as below).

This demo is an example of using the ‘hunt and track’ algorithm for producing a predominance diagram for an Fe-Cl system. It also creates `pdf`, `ai`, `eps`, `epsi` and `jpg` files if Ghostscript is installed and so can be used to test that installation.

The output looks something like:

```plaintext
*** PhreePlot 1 *** (10:39:45 17 May 2011)
Incorporating the Phreeqc library by DL Parkhurst, SR Charlton (USGS),
```
The screen output provides feedback on progress. The columns are: (i) iteration number; (ii) x-axis variable (automatically generated); (iii) y-axis variable (automatically generated); (iv) the type of step being taken; (v) truncated name of the predominant species (most abundant); (vi) truncated name of the sub-dominant (second most abundant) species; log concentrations of the dominant and sub-dominant species (mol/kgw) when solution species or log partial pressures when gases. Where one or more constraints are operating, these are elevated to the top-most position(s) and the values given are determined by the type of species as outlined above.

The above example makes use of the `ht1.inc` include file. This determines exactly what values are returned to PhreePlot.

The code returned for the type of step taken is determined as follows:

- the first digit is 1 while hunting for boundaries along an edge or 2 while tracking an internal boundary;
- the second digit is either the side number or the cell corner (1-4, counted clockwise from bottom left. 1 is the left-hand y axis, 2 is the top x axis...);
- a negative sign indicates that a constraint is operating;
- 00 is a special code for a non-tracking move (as used by a ‘grid’ plot).

The above example indicates that PhreePlot starts by hunting for boundaries along the left-hand y axis. It then starts tracking along an internal boundary at iteration 14. It will finish by tracking along the remaining boundaries to check that there are no more intersections to start tracking from. This example takes 1226 iterations to complete.

The demo.bat file included contains many more examples including many ‘custom’ plots which use the selected output from Phreeqc to generate a plot. This includes the standard set of examples distributed with Phreeqc. Each example will take from a few seconds up to several minutes or more to calculate. Most of the time in these examples is spent running Phreeqc.

Note that the demo examples are based on the pp.set file provided. If changes to this file are made, it may be necessary to change the input files. For example, pp.set sets the commonly used tag `<log_H>` to `<-x_axis>` so that the x-axis limits can be specified in terms of pH directly rather than as the log (H+) activity.

### 3.3 The ‘pp.log’ File

Providing, pplog is set to TRUE, a log of every PhreePlot run is written to a file called pp.log which is created in the PhreePlot system directory. This can be checked at the end of the run
to make sure that all has run as expected and is especially useful for checking the results of multiple runs from a batch file.

Each run normally gives rise to two lines on the pp.log file. The first line indicates the time started and the second line gives the completion status. The absence of a second line indicates a crash. Normally, an ‘OK’ status should be returned for each input file if all has run well.

<table>
<thead>
<tr>
<th>Time</th>
<th>Date</th>
<th>Input_file</th>
<th>Type</th>
<th>Method</th>
<th>n</th>
<th>Time(min)</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:24:38</td>
<td>15_June_2010</td>
<td>C:\PhreePlot\demo\test\test.ppi</td>
<td>ht1</td>
<td>calculate</td>
<td>0</td>
<td>0.000</td>
<td>Started</td>
</tr>
<tr>
<td>9:24:44</td>
<td>15_June_2010</td>
<td>C:\PhreePlot\demo\test\test.ppi</td>
<td>ht1</td>
<td>calculate</td>
<td>1539</td>
<td>0.092</td>
<td>OK</td>
</tr>
<tr>
<td>9:24:44</td>
<td>15_June_2010</td>
<td>C:\PhreePlot\demo\Phreeqcexamples\ex1\ex1.ppi</td>
<td>custom</td>
<td>calculate</td>
<td>0</td>
<td>0.000</td>
<td>Started</td>
</tr>
<tr>
<td>9:24:44</td>
<td>15_June_2010</td>
<td>C:\PhreePlot\demo\Phreeqcexamples\ex1\ex1.ppi</td>
<td>custom</td>
<td>calculate</td>
<td>1</td>
<td>0.004</td>
<td>OK</td>
</tr>
<tr>
<td>9:24:45</td>
<td>15_June_2010</td>
<td>C:\PhreePlot\demo\Phreeqcexamples\ex2\ex2.ppi</td>
<td>custom</td>
<td>calculate</td>
<td>0</td>
<td>0.000</td>
<td>Started</td>
</tr>
<tr>
<td>9:24:51</td>
<td>15_June_2010</td>
<td>C:\PhreePlot\demo\Phreeqcexamples\ex2\ex2.ppi</td>
<td>custom</td>
<td>calculate</td>
<td>1</td>
<td>0.018</td>
<td>OK</td>
</tr>
</tbody>
</table>

This log file will accumulate output from every run and so should be periodically emptied or erased. It will be automatically recreated or appended to as necessary.

If there has been a failure of Phreeqc such that no selected output was produced, then a ‘?’ is appended to the right of the number of speciation calculations, n. Details of the offending output will be written to the log file if that was active.

If debug is set to 1 or stopOnFail is set to 1, then PhreePlot will stop at the first failure. If PhreePlot has had to adjust the resolution of a ‘hunt and track’-generated predominance plot for various reasons then a ‘*’ is printed next to the number of speciation calculations.

If there has been an error reading one of the data input files, e.g. while reading an extraSymbolsLines file, then an exclamation mark (‘!’) is appended to the status. Check the log file for details. The error may stop PhreePlot from running or may continue by skipping the erroneous data.

Other possible variations of the logged status on termination are:

- **Error**: an error occurred somewhere in the calculations
- **Input_error**: an error occurred reading input
- **Plotting_error**: an error occurred during plotting
- **GS_error**: an error occurred while Ghostscript was converting the ps file
- **No_plot**: no plot was specified
- **Interrupted**: the <Esc> interrupt was used to halt execution
- **Started**: still running (or crashed while running)

Setting pplog to FALSE will prevent anything being written to the pp.log file.

### 3.4 Examining the Results of the Run

Various output files will be written to the input file directory. The formats of these files are described more fully elsewhere (Section 5). The file plot.ps is always a copy of the last Postscript plot file produced. The log file if written should give a more detailed summary of the calculations undertaken.

### 3.5 Making a Working Directory

It is best to keep all your working files in a directory that is quite separate from the setup directories. Since each run can produce a large number of files, it is best to make a new directory for each ‘problem’. It is usually best to copy an existing similar working input file to this directory and edit that as needed.

Running the file should be straightforward providing the PhreePlot environment variable has been set properly (see Section 2.4.1), e.g.

```
C:\projects\PhreePlot>md FeS2
C:\projects\PhreePlot>cd FeS2
```
3.6 GETTING FAMILIAR WITH THE OPTIONS

The calculations and plotting are controlled by the various keyword-value pairs and lists which are read from various input files. There is also usually some Phreeqc-format appended to the end of the main input file. There are many options, some of which are more important than others, and it is difficult in the beginning to know where to start.

The best way to learn is to run the demo examples. Pick an example that is closest to what you are interested in and run it. If one of the keywords in the input file looks interesting, look it up to see what it does and experiment by changing it.

3.7 USING BATCH FILES TO RUN A SET OF RUNS

PhreePlot is designed to produce a series of plots by varying one of the looping variables (main species, loop parameter or the x- and y parameters). However, it is not designed to do more than one independent type of plot in one run. To do this, it is necessary to run PhreePlot multiple times and then combine the results. The overlay feature enables plots from earlier runs to be combined with the current plot to produce complex page layouts containing multiple plots.

Multiple independent plots can be executed in batch mode by preparing a batch file. This is surprisingly powerful. The results from PhreePlot can even be intercepted, modified by another program and returned to PhreePlot for plotting. Many programs exist to convert images to other image formats, for example. Even a rudimentary understanding of Windows batch scripting can be useful when processing large numbers of files.

The demo.bat file illustrates how a set of runs can be run in batch mode. This has obvious advantages for repeatedly running a set of examples. On multi-processor machines, it may be advantageous in terms of speed to split the batch files into two or more to take full advantage of the separate processors.

It is possible to intersperse other batch commands in a batch file of PhreePlot runs in order to rename, copy or delete files etc between runs.

It may be necessary to change the current working directory to that of the input file if a shortened file name is given.

The start command can be used to launch individual batch files simultaneously from within this file (see demo2.bat). Alternatively, use call to run a batch file from within a batch file. This will run the batch files sequentially.

The override.set file can be a useful place to add settings that will apply to all files run from a batch file. For example, to generate png files for all the plots add

```
calculationMethod 2
png t
```

to the override.set file and re-run.

3.8 STOPPING A RUN

You can usually stop a run using the Esc key followed by 's' for stop.

Stopping a batch file or script completely is a bit different. In Windows, Ctrl-c will always abort the current run and when it is a Windows batch file, you will be given the opportunity to exit the entire batch file.
4 Phreeqc basics

4.1 Online Phreeqc documentation

Since the chemistry section of PhreePlot input files is itself essentially Phreeqc code, it is necessary to be familiar with the way that Phreeqc input files are set up. This is described in detail in the Phreeqc manual (pdf). This manual is also available online in browser format. Changes and corrections added since the initial release are given in the 'Release notes' on the USGS website.

4.2 How Phreeqc interacts with PhreePlot

PhreePlot uses Phreeqc for all geochemical calculations and runs only slightly modified Phreeqc input files. Phreeqc calculations are controlled by an input file, a database file and the program itself. The input can include one or more simulations. These need not be related but they usually are. In many cases, only a single simulation is all that is needed to generate the output required but sometimes more than one simulation is necessary, or it may be desirable to split a simulation into two or more for the sake of efficiency (see Example 61).

A Phreeqc input file consists of a series of keyword data blocks separated into 'simulations' by the END keyword. This file is read sequentially. When an END is found or the end of file is reached, the statements accumulated since the last END are executed. We call this a 'run'.

This execution triggers the specified calculations and the writing of results to the normal output and selected output 'files' (if active). A number of data structures including the composition of various SOLUTIONS, EQUILIBRIUM_PHASES etc are also created or updated.

Many of these data structures persist across simulations but some of them can be explicitly saved and re-used with the SAVE and USE keywords. The PUT and GET Basic statements also enable user-defined numeric variables to be stored in, and retrieved, from global storage.

Phreeqc does not provide any explicit means of looping around specific lines of the input file although some of the keywords such as REACTION and TRANSPORT implicitly involve a user-defined set of iterations. This lack of general looping capability means that the input files required for some calculations, including those often required for plotting, can become large and repetitive.

PhreePlot attempts to overcome this by providing a framework for iterating across sections of the Phreeqc input file while requiring minimal changes to the Phreeqc input file itself. It does this by defining a set of four nested 'DO' loops which iterate over certain sections of the Phreeqc code.

These loops, from the outside (least rapidly changing) in, are known as: (i) the 'main species' loop; (ii) the 'z'- or 'main' loop; (iii) the y-axis loop, and (iv) the x-axis loop. The main species loop iterates over a list of character variables while the remaining loops all iterate on numeric variables. Not all loops need to be used all of the time. Indeed, you do not need to use any of the loops.

Setting the iteration parameters for these various loops and providing instructions describing which parts of the input file to loop over, plus many other PhreePlot settings, are either inherited from the default settings (a file) or specified at the top of the PhreePlot input file. Phreeqc input is at the bottom. A line containing the word chemistry separates these two sections.
Special tags – character strings between angled brackets – are used within the Phreeqc input to act as placeholders which are substituted at run time by values generated by the various PhreePlot looping mechanisms, and by other means. Phreeqc never sees these tags, just the substituted values. Tags can also be used in the upper (PhreePlot) section of the input file. These tags serve as global variables that enable transfer between different Phreeqc simulations (somewhat like the Basic put/get mechanism) but they also enable communication between Phreeqc and the plot, and can be used to dynamically control such things as labelling, scaling or sizing the plot.

Tags can be defined in a PhreePlot input file but can also be automatically generated from the output of earlier simulations, or from reading an external data file. PhreePlot maintains a table with the current values of all these tag variables ready for substitution at the appropriate time. However, note that dynamic tags generated during execution will not be available during replots (calculationMethod 2 or 3).

The sections of Phreeqc code iterated over are always based on contiguous blocks of one or more simulations. The default is that the main species and z- loops iterate over all the simulations while the x- and y-axis loops only iterate over the last n simulations where n is one by default.

Communication of results between Phreeqc and PhreePlot is via the selected output. Phreeqc’s in-built Basic interpreter gives you a great deal of flexibility in controlling what is sent to the selected output.

Each simulation normally produces one or more lines of selected output although this can be turned on or off at will. Where no output has been requested, a blank line is produced. This output typically consists of the results of one or more initial solution etc calculations followed by one or more lines giving the results of a reaction.

It is often the results on this last line that are wanted. PhreePlot only reads the last k lines of the selected output where k by default is again normally one (in the cases where Phreeqc itself does iterations, a whole block of results may need to be read and so k can be set to be greater than one). This output is accumulated in a special file, called the ‘out’ file, which has a tabular format ready for plotting.

PhreePlot has limited plotting capabilities though the output that is available is normally of high quality (the native format is Postscript). The aim is to be able to get a reasonably quick visual feel of the output, and once satisfied, to be able to generate plot files later, if necessary in an automated (batch) fashion. All of the data files used to generate plots are well-structured text files so can be readily imported into other plotting programs.

The ability to use tag variables in Phreeqc input files means that it is straightforward to keep re-running a set of simulations with a different set of values. This is the basis of the model fitting that is built into PhreePlot.

4.3 THERMODYNAMIC DATABASES

The standard databases distributed with Phreeqc and PhreePlot include a varied range of elements and ligands. The scope of these databases in terms of the elements defined are given in Appendix 2. Check the appropriate web sites for updates.

It is straightforward in PhreePlot to change the database used using the database keyword. Bear in mind that the same minerals and gases may have different names in the different databases. This must be reflected in the use of such names in the Chemistry section of a PhreePlot input file.

4.4 TYPES OF OUTPUT PRODUCED BY PHREEQC

Phreeqc can create two types of output files:

(i) normal output file: the PRINT and USER_PRINT data blocks control the output to the main
output file. This consists of a well-structured but verbose log of the speciation calculations split into various blocks corresponding to each stage of the calculations. It also includes any user-defined output defined by PRINT statements in the USER_PRINT or USER_PUNCH data blocks. In PhreePlot, this output is directed to the Phreeqc.0.out and *.all files. The Phreeqc.0.out file is only written if the Phreeqc.0.out keyword is 'T', or 'auto' and debug >0. The *.all file is only written if the all keyword is 'T', or 'auto' and debug >1. The name of the *.all file can be changed by adding the new filename as the second parameter on the all keyword line.

(ii) selected output file: the selected output n and user punch n data blocks control tabular output to the selected output file(s). The main selected output file (n = 1) is the file normally used by PhreePlot for generating plots and it is normally this file that has to be manipulated to give the required output. Certain rows of data from this file are accumulated in the 'out' file which is often used to generate plots. Therefore familiarity with the ways of controlling output to the selected output file is a prerequisite for running PhreePlot. This is described in detail in the Phreeqc manual (Parkhurst and Appelo, 1999).

In general, a single line of selected output is produced for each Phreeqc calculation — “after each initial solution, initial exchange-composition, initial surface-composition, or initial gas-phase-composition calculation and after each step in batch-reaction or each shift in transport calculations”. If no USER_PUNCH variables have been defined, a blank line is output or if the selected output has been turned off with the PRINT statement or -active FALSE setting, a header line but no output is produced.

The Phreeqc library used by PhreePlot has switches to control whether the selected output is written to a physical file or to memory. In PhreePlot, this is controlled by the value of the debug setting with debug = 0 normally writing only to memory and greater values writing increasing amounts to ‘disc’ (this could be a solid-state drive).

4.5 THE SELECTED_OUTPUT AND THE USER_PUNCH DATA BLOCKS

All output communications between Phreeqc and PhreePlot are sent via the selected output. Therefore it is necessary to ensure that the correct output is sent to this ‘file’ (it may be just a piece of memory or a ‘virtual’ file) and to tell PhreePlot what the format of the selected output file is in relation to what PhreePlot has to do. This is done with a combination of the selectedOutputFile Basic statements in Phreeqc and selectedOutputLines keywords in PhreePlot.

Phreeqc now supports multiple selected_output/user_punch blocks. These are numbered with a user number, n, e.g. selected_output n/user_punch n where n is an integer which if not specified is given the value 1. Within a given simulation, the selected_output n/ user_punch n numbers should be the same. Unlike with the batch and interactive versions of Phreeqc, the default for the iPheeqc library as used in PhreePlot is for the selected output not to be written to a file. Rather, the selected output values are expected to be read directly from memory and there are facilities to do this. This is in principle faster as it involves less input/output. So in order to get arbitrary selected output files actually written, it is necessary to switch them on first. The selectedOutputFile keyword has a switch for doing this. This switch combined with the selected_output/user_punch blocks determines if a selected output file will be created or not.

There is one more important factor: the selected_output block must be executed in a simulation before the simulation triggering the selected output, i.e. the selected output switch must be set ‘on’ before the simulation generating the output is executed. In PhreePlot’s modus operandi, this means that the selected output block(s) should either be placed in a pre-loop simulation or for main loop simulations, the ‘one simulation at a time’ approach should be adopted (see mainLoop) with the selected output block placed in a simulation preceding the simulation that triggers the wanted selected output.

PhreePlot only transfers data from one selected output block to the ‘out’ file, the structured-
output file that is used for plotting. The block chosen is always the one with the highest user number. This is not necessarily in the last simulation to be executed. Therefore judicious numbering of the blocks provides a straightforward way of selecting the simulation which will provide the output data.

**selectedOutputLines** is a number giving the number of lines from the chosen **SELECTED_OUTPUT**/**USER_PUNCH** block to transfer to the 'out' file, counting from the bottom of the output upwards. This setting does not affect the data that will be sent to other files specified with the **SELECTED_OUTPUT** -file identifier.

Selected output will only be generated if both the **SELECTED_OUTPUT** and **USER_PUNCH** data blocks are both present somewhere in the **Phreeqc** part of the input file. By default they are active from the point of definition downwards. Selected output will be triggered from all simulations with a defined **SELECTED_OUTPUT** n/**USER_PUNCH** n pair each time an initialization, reaction or timestep occurs. These can be selectively turned off/on with the **selected_output** identifier in the **PRINT** block and the **-active** and **-user_punch** identifiers in the **SELECTED_OUTPUT** block. There will have to be a reason to emit some output so usually at least one **SOLUTION**/**REACTION** block is needed (it can be empty).

It may also be useful to include the **-reset FALSE** and **-high_precision TRUE** identifiers to suppress unnecessary headers and to retain maximum precision in the output numbers. The system variables like **SIM_NO** are only produced without being explicitly defined when n = 1 in **SELECTED_OUTPUT** n/**USER_PUNCH** n.

While the 'out' file is the principal data file used for plotting, other files can be used to supply data for plotting by specifying them with the **extradat** keyword.

The built-in BASIC interpreter in **Phreeqc** provides a very flexible approach for defining the selected output. The interpreter provides access to most of the fundamental system variables such as species concentrations and activities. It also includes various summary functions such as **TOT()**, **SURF()** and **SYS()**. The data sent to the selected output from various stages of **Phreeqc** calculations can be controlled in the **USER_PUNCH** n data block(s) by checking the **STEP_NO** and jumping over any **PUNCH** statement(s) for which the output is not wanted.

4.5.1 The **SELECTED_OUTPUT** filename and forcing the file to be written

The default name of the **SELECTED_OUTPUT** file in **PhreePlot** is 'selected_1.0.out' but this can be changed using the **SELECTED_OUTPUT** -file identifier, as normal in **Phreeqc**. As mentioned above, the selected output in **PhreePlot** is normally written to a 'virtual' file (a block of memory) and is not necessarily written to a 'physical' or disc file. The writing of the physical file is controlled by a switch set by the **selectedOutputFile** keyword. This setting applies to all selected output files created during the run.

For **debug > 1** a physical file will always be produced with the given file name so that the output can be inspected. There will be small performance penalty because of the file writing.

It can be useful to force a physical file to be written with multi-simulation input files. Data from each simulation could be sent to a different file and plotted accordingly using the **extradat** keyword to define the data files to be searched for plot data.

If the **-selected_out** identifier of the **USER_PUNCH** data block is set to **FALSE**, no lines are written to the selected output file. However, a selected output file will still be produced but it will be blank. This will be translated to a set of variable values all given zero values, i.e. all output variables will be reported as **0.000000000000E+00** in the log file.

If a change in the structure of the selected output is wanted, make sure the simulations involved are executed as separate blocks (see **mainLoop**).

4.5.2 Scope of **Phreeqc** keywords

Each **Phreeqc** simulation consists of a series of keyword data blocks which define the calculations for that simulation. The order of these keywords within a simulation is normally not
important other than if a keyword is replicated, the last instance overrides earlier ones. An exception is that the position of the -reset in USER_PUNCH keyword blocks can be important. Also the position of the units and numberOfFitParameters keywords can be important in relation to the related settings that follow.

The simulations are separated from one another by an END keyword. Each END can therefore be interpreted as 'Calculate'.

Other keywords such as SELECTED_OUTPUT and USER_PUNCH have a broader scope and operate from their point of insertion forward.

For example, the following input defines four Cd solutions and does an 'initial solution' calculation (speciation) for each one. The four simulations are essentially unrelated.

```
SOLUTION 1 # Simulation 1
  Cd 1.0
END
SELECTED_OUTPUT #Simulation 2
  high_precision true
  reset false
USER_PUNCH
  headings Cd+2
  10 punch mol("Cd+2")
SOLUTION 2
  Cd 0.1
END
SOLUTION 3 #Simulation 3
  Cd 0.35
END
SOLUTION 4 #Simulation 4
  Cd 0.6
END
```

This produces the following output in the selected_1.0.out file when the wateq4f.dat database is used:

```
Cd+2
9.92072733798e-005
3.497329579806e-004
5.995528842616e-004
```

Note that no output has been produced for the first initial solution calculation since it is in a simulation that precedes the definition of the SELECTED_OUTPUT data block. The SELECTED_OUTPUT file is 'turned on' in simulation 2 and the output appears from this point forward, hence the three lines of output representing output from simulations 2 to 4. Additional PUNCH statements within a simulation result in more output columns. The 'headings' line in the USER_PUNCH data block controls the header used for the column in the selected output file.

Therefore, for as long as the selected output file is turned on, at least one line of output is produced by each simulation providing that a USER_PUNCH block has been defined. The output for the whole job accumulates in the selected output file. PhreePlot accumulates 'selected data from the main selected output file' into a single output file called the 'out' file or outfile. The default is to accumulate only the last line from the last simulation here the 5.995528842616e-004.

The scope of many other Phreeqc 'structures' is global in the sense that once created in a simulation they persist for the remainder of the run unless overwritten. For example, solutions defined by the SOLUTION keyword are automatically preserved across simulations. These solutions can be used in subsequent simulations providing that the solution number is not reused or redefined by a reaction. The same is true of PHASES, SOLUTION_SPECIES etc.

4.5.3 What is sent to the SELECTED_OUTPUT file?

Both numeric variables and text strings can be sent to the SELECTED_OUTPUT file by defining them in a USER_PUNCH data block. The column headings should reflect each entry on a one to
one basis. If the list of headings is shorter than the list of variables output, the missing headings are given the value 'no_heading'.

The names of the column headings take on especial importance in PhreePlot since they are used to automatically generate the names of new tags (see Section 6.4.2) and can ultimately be used to label plots.

The number of significant figures sent to the selected_output file is controlled by the -high_precision identifier in Phreeqc. It is normally safest to set this to TRUE, i.e. output at high precision (12 decimal places, 13 significant figures). Normal precision is 4 decimal places (5 significant figures). The default is FALSE so the high_precision identifier needs to be set explicitly as above if high precision output is wanted. The high precision option is definitely preferable when fitting data to models and when calculating predominance diagrams.

The sequence of columns sent to the selected_output file is set by the following rules:

(i) one column for each of the selected_output data item switches (simulation, state, solution...) that is set to TRUE. The column headers for these switches, and their order, is given by: sim, state, soln, dist_x, time, step, pH, pe, reaction, temp, Alk, mu, mass_H2O, charge and pct_err. The default value for the first eight of these is TRUE and for the remainder is FALSE. It is normally advisable to use the -reset false option at the top of the selected_output data block to turn all of these off. Then the ones that are wanted can be turned on by explicitly defining them as TRUE.

(ii) one column for each variable defined in the list data items such as -totals, -activities etc. output in the sequence specified.

(iii) one column for each item punched within the user_punch data block in the cumulative order in which they are specified by the BASIC statements. There can be one or more items per punch statement.

An example is:

```
SELECTED_OUTPUT
  high_precision true
  reset false
USER_PUNCH
  headings  pH Ca Mg
  10 punch -la("H+"), tot("Ca"), tot("Mg")
```

4.6 Setting up the selected_output file for input to PhreePlot

4.6.1 Possibilities for looping of Phreeqc input files

The structure of Phreeqc input files is very flexible in terms of the number of simulations within a file and the relation between the various simulations. These are executed sequentially until the end of file is found. Phreeqc does not contain any mechanism to enable looping of the various simulations. This is what PhreePlot attempts to do without impinging unduly on the overall structure of the Phreeqc input. PhreePlot expects a certain Phreeqc structure in order to control this looping. This structure depends on the calculationType and certain other keyword settings.

The general philosophy in preparing PhreePlot/Phreeqc input files should be to (i) keep the input file as simple as possible; (ii) put any preliminary calculations that only need to be executed once in one or more 'pre-loop' simulations at the beginning of the file; (iii) finish with the simulation, or range of simulations, that need to be repeated many times with minor changes (the 'main loop').

PhreePlot also recognises two types of looping: (i) a 'continuous' type of looping which focuses on the 'resolution' of the calculations, (ii) a 'discontinuous' type of looping which generates a list of discrete values to be used. The x- and y-axis loops belong to (i), and the main species and z-loop belong to (ii). These differences are reflected in the way that the iterations are specified: (i) is specified in terms of a minimum value, a maximum value and a 'resolution'
while for (ii) the main species loop uses a list of character variables and the z-loop uses a minimum value, a maximum value and an increment value. An irregular list of z-loop values can also be supplied.

Typically, the x- and y-axis loops are used to control the smoothness of generated curves for plotting while main species repeats calculations over a range of chemical elements and the z-loop controls the spacing between curves based on a range of discrete values of some important variable.

It is only the 'main loop' simulations that are repeated under the x- and y-axis looping mechanisms. The pre-loop simulations should be used for 'one-off' calculations such as initial solution calculations or database definitions that do not need to be varied during the main loop but which might need to be used recalculated for each of the main species and z-loops. More details about PhreePlot looping and the structure of multi-simulation input files is given in Section 6.2.

**Predominance plots**

![Diagram](image_url)

**Figure 4.1.** Flow during the execution of a multi-simulation file generating a predominance plot (calculationType 'ht1' or 'grid'). Simulations 1–4 are 'pre-loop' simulations used for initial solution etc calculations. The <x_axis> and <y_axis> tags are only present in the 5th or 'main loop' simulation. It is this one which is repeatedly called while tracking or traversing the specified domain. It is always the last line of the selected output generated by this main loop simulation that returns the predominant species for PhreePlot to process. The selected output file has a special structure and is normally generated by including the ht1.inc file or some variant of it in the input file. Note that this flow diagram refers to a single value of the main species and z-loop variables.

The structure of the input file to generate a predominance diagram typically consists of two simulations (Figure 4.1). It could all be done with one simulation but it executes more rapidly if the initialization parts (the 'pre-loop' calculations which only need to be executed once) are separated from those calculations that vary and that need to be calculated repeatedly (the 'main loop' calculations). The number of the first main loop simulation is identified with mainLoop.

The first simulation usually pulls in the ht1.inc file which defines the Fix_H+ phase and sets up the selected output 'file' and the required USER_PUNCH definitions that transmit the predominant species to PhreePlot. It also includes a SOLUTION data block which defines the total quantities of all elements in the system of interest.

The second simulation uses the chemical system defined above and subjects it to control by the x axis and y-axis variables.

A simple example for generating an Fe predominance diagram is:

```bash
# the first simulation defines the total quantities involved
include 'ht1.inc'
SOLUTION 1
pH 1.8
```

The structure of the input file to generate a predominance diagram typically consists of two simulations (Figure 4.1). It could all be done with one simulation but it executes more rapidly if the initialization parts (the ‘pre-loop’ calculations which only need to be executed once) are separated from those calculations that vary and that need to be calculated repeatedly (the ‘main loop’ calculations). The number of the first main loop simulation is identified with mainLoop.

The first simulation usually pulls in the ht1.inc file which defines the Fix_H+ phase and sets up the selected output ‘file’ and the required USER_PUNCH definitions that transmit the predominant species to PhreePlot. It also includes a SOLUTION data block which defines the total quantities of all elements in the system of interest.

The second simulation uses the chemical system defined above and subjects it to control by the x axis and y-axis variables.

A simple example for generating an Fe predominance diagram is:

```bash
# the first simulation defines the total quantities involved
include 'ht1.inc'
SOLUTION 1
pH 1.8
```
Note that solution 1 is titrated with NaOH and O$_2$(g) to achieve the required endpoints. The initial pH of solution 1 should be less than the minimum pH of interest so that adding NaOH can be guaranteed to achieve the full range of pH's required.

If the `<mainspecies>` tag has more than one variable associated with it or if the `<loop>` variable has been set up to perform more than one z-loop, then the entire input file is run each time one of these loop variables changes value. This can be used to prepare a set of predominance plots for several elements each with the its total concentration, for example, varying by some amount. The ...\demo\loop.html examples produce Fe predominance diagrams for a range of total Fe concentrations. If an irregular sequence of z-loop values is required use the `loopFile` keyword to read the values from a file.

If the main loop contains more than one simulation, then by default all of these simulations are executed in a single run of Phreeqc. This means that tags will not be updated between simulations. If this is needed, it is necessary to run the main loop simulations one at a time. This is done by setting the `selectedOutputFile` switch to TRUE.

### Data-led calculations

![Diagram of pre-loop/main loop flow during data-led calculations](image)

The simulation(s) used for each data point depend on the value or range of values specified in the column defined by the `blockRangeColumn` of the data file. The pre-loop/main loop division is determined by the value in the `mainLoopColumn`.

Figure 4.2. Flow during the execution of data-led calculations ('simulate' or 'fit' calculation types) in which the simulation used is specified in the fit data file (or is simulation 1 by default).

The 'fit' and 'simulate' calculation types both read in certain parameters from a fit data file. In order that the global optimization can include data calculated by different chemical models, each data point can point to a different chemical model (Figure 4.2). Each chemical model is defined by one or more simulations in the Phreeqc input code. These are specified by a data column in the fit data file - the column used for this is defined by the `blockRangeColumn`. The default value for the simulation is 1 which is the value assumed if no `blockRangeColumn`
is present in the fit data file. In this case, all values are calculated by the same chemical model. If more than one simulation is needed, then a contiguous range can be entered, e.g., "1–2" (or equivalently "1_2") to indicate that simulations 1 and 2 will be used. There should be no spaces in the string.

**Custom plots**

The 'custom' calculation type can be used to generate data for a variety of Phreeqc-type calculations especially where repetition is required that is not covered under the normal Phreeqc options (Figure 4.3).

A custom calculation generally consists of zero or more pre-loop simulations which calculate various initializations and then one (or more) simulations which are iterated using PhreePlot's x- and y-looping mechanisms. Normally it is the last line from the selected output generated from the last simulation that is accumulated in the "out" file and used in any subsequent plotting.

If a z-loop variable is included, the whole input file is re-run for each z-value including any pre-loop simulations.

The following input first defines a 1 mmol/kgw solution of CdCl₂ and then equilibrates this

```
mainLoop = 'last' (= 5)
```

**Figure 4.3.** Normal flow during the execution of a multi-simulation input file for custom calculations. The output depends on various settings including whether each simulation is executed in turn with just the final simulation contributing to the 'out' file (upper figure) or whether the has been set to point to an earlier simulation (lower figure). Only simulations from that number forward are repeated during any 'looping' and are by default used to populate the 'out' file.
with carbon dioxide at a $P_{\text{CO}_2}$ partial pressure of $10^{-3.5}$ atm. Solution 1 is carried forward to the second simulation. This simulation fixes the pH at 8.0 by titrating with NaOH and allows amorphous cadmium hydroxide to precipitate if its solubility product is exceeded (which it is). Note that a maximum of 1 mol NaOH is allowed to be used to prevent very high ionic strengths from being created (the Pitzer option would have to be used for very high ionic strength solutions).

```
SELECTED_OUTPUT #Simulation 1
  high_precision true
  reset false
USER_PUNCH
  headings Cd+2 SI_Otavite
  10 punch mol("Cd+2"), SI("Otavite")
SOLUTION 1
  Cd  1.0
  Cl  2.0 charge
END

USE SOLUTION 1 #Simulation 2
PHASES
  Fix_H+  
  H+ = H+  
  log_k 0.0
EQUILIBRIUM_PHASES
  Otavite  0 0   #Otavite is CdCO3   
  CO2(g) -3.5 10
  Fix_H+ -8 NaOH 1
END
```

The selected output for this looks like this:

```
Cd+2               SI_Otavite
  4.409910346467e-007     0.000000000000e+000
```

This output is from the second (final) simulation. It gives the Cd$^{2+}$ concentration after otavite has precipitated most of the Cd. These data are also transferred to the 'out' file.

Defining a pure phase to consist of a single species and then using the `EQUILIBRIUM_PHASES` keyword to define its saturation index (SI), as here to fix the pH, is the Phreeqc way of fixing a species activity. This simply forces the log activity to be numerically equal to the SI since $SI = \log(IAP/SP) = \log(aH+/log_k) = \log(aH+)$ where $IAP$ is the ion activity product and $SP$ is the solubility product.

In this example, the first simulation sets up the initial Cd solution and the second simulation performs the reaction. The same effect could have been achieved by reducing the whole file to a single simulation by removing the END and USE keywords. The selected output then looks like:

```
Cd+2               SI_Otavite
  8.738892163591e-004    -9.999000000000e+001
  4.409910346467e-007     0.000000000000e+000
```

with the first line of output being derived from the initial solution calculation and the second line having been derived from the second (reaction) simulation.

Another way of running both simulations together would be to set `mainLoop` to 1 so that both simulations are run together as 'main loop' simulations. By default, the 'out' file only picks up the last line of the selected output but if all three lines are wanted, `selectedOutput-Lines` for the simulation should be set to 3 or 'auto'. 'auto' will always transfer all the data lines to the 'out' file.

If only the final concentration is wanted and the two simulations are run separately, then it is also possible to omit the output from the first simulation by turning the selected output off then on again in the second simulation using the `-selected_output` identifier of the `PRINT` data block, e.g.

```
SELECTED_OUTPUT #Simulation 1
  high_precision true
  reset false
USER_PUNCH
  headings Cd+2 SI_Otavite
  10 punch mol("Cd+2"), SI("Otavite")
SOLUTION 1
  Cd  1.0
  Cl  2.0 charge
END
```

SELECTED_OUTPUT #Simulation 2
```
PHASES
  Fix_H+
  H+ = H+
  log_k 0.0
EQUILIBRIUM_PHASES
  Otavite  0 0   #Otavite is CdCO3
  CO2(g) -3.5 10
  Fix_H+ -8 NaOH 1
END
```

The selected output for this looks like this:

```
Cd+2               SI_Otavite
  4.409910346467e-007     0.000000000000e+000
```

This output is from the second (final) simulation. It gives the Cd$^{2+}$ concentration after otavite has precipitated most of the Cd. These data are also transferred to the 'out' file.
Phreeqc basics

- high_precision true
- reset false

PRINT
- selected_output false

SOLUTION 1
Cd 1.0
Cl 2.0 charge

EQUILIBRIUM_PHASES
Otavite 0 0 #Otavite is CdCO3
CO2(g) -3.5 10
Fix H+ -8 NaOH 10

SAVE Solution 2
END

USE SOLUTION 2 #Simulation 2
PRINT
- selected_output true

EQUILIBRIUM_PHASES
Otavite 0 0 #Otavite is CdCO3
CO2(g) -1.5 10
Fix H+ -8 NaOH 10

END

gives the selected output as:

Cd+2 SI_Otavite
4.40991036467e-007 0.000000000000e+000

Knowing which minerals might form using the given database

Phreeqc has no simple way of automatically inserting a valid set of minerals into an EQUILIBRIUM_PHASES keyword block such that any mineral that is predicted to form does form. This normally has to be done manually. The mineral names will depend on the database used and the solution composition. The printphases.inc include file extracts a list of all possible minerals by using the SYS() function. This include file can be added to an input file to get the mineral names printed to the file Phreeqc.out. These can then be pasted back into the input file as needed.

An alternative and slightly simpler approach for h5 and grid plots is to just set the resolution to 1. This automatically inserts the printphases.inc code directly into the Phreeqc input stream just ahead of the first (and hopefully only) SOLUTION keyword block. It also ensures that the Phreeqc.out file is written and that all the PRINT settings are reset to TRUE. This will only work properly for single simulation input files and providing that there are no USER_PRINT blocks following the SOLUTION keyword block (these would override the inserted code).

With these provisions, a single iteration is performed with all loop variables set at their initial values and the names of all possible mineral species are written to Phreeqc.out.

It is possible to use Phreeplot to automatically generate a list of all possible mineral species in one simulation, write them to a tag, and then to retrieve this tag in the EQUILIBRIUM_PHASES data block of a subsequent simulation. This approach is used in demo\minstab\allminerals.ppi to generate a predominance diagram that automatically adds all of the minerals in the database to the list of potentially precipitating minerals. This must be used with caution since many minerals, while thermodynamically stable, do not form in a reasonable timescale.

4.6.2 Setting up a loop file

The z-loop or loop variable is used for discontinuous variables and will result in a separate calculation and associated curve (or plot) for each value of the loop variable. This contrasts with the x- and y-variables which are designed for ‘continuous’ variables in which the resolution defines the number of calculations per curve.

<loopmin>, <loopmax> etc can be used to define a regular sequence of values for the loop variable but if an irregular sequence is required or if more than one variable has to be carried in parallel for each iteration, then a loop file must be created.
The loop file is an ASCII file which is read in free format. This file is primarily intended to contain numeric data but it can also include character data. It can optionally contain a header row with column names and an initial column with loop names. The format is deduced from the first two columns and first two rows of the file. Columns are either numeric or character. The first two columns of the first row determine if it is a header row (if both are character variables). The first column of the first two rows determine if loop names are present (if both are character variables). The remaining columns can be either numeric or character – this is determined by the type of data in the first non-header row. It may be necessary to introduce a dummy numeric column as column 1 or 2 to force the correct interpretation of the file.

The four possible formats are shown in Figure 4.4.

(a) no header
   no loop names

(b) no header
   loop names

(c) header
   no loop names

(d) header
   loop names

\[ \begin{array}{c c}
  \text{num} & \text{num} \\
  \text{char} & \text{num} \\
  \text{char} & \text{num} \\
  \text{char} & \text{char} \\
  \text{num} & \text{numeric value} \\
  \text{char} & \text{character value}
\end{array} \]

Figure 4.4. The format of fit data files is determined by the type of data in the first two rows and first two columns of data.

The column headers if present are used to make the tag names, e.g., Na will make the tag \(<Na>\). Make sure that the column headers, if present, give rise to unique tag names.

If the header line is absent, then the tag names will be automatically set to \(<\text{loop1}>, <\text{loop2}>\) for numeric column 1, 2 etc. These tags can be used in the input file. \(<\text{loop1}>\) is also known simply as \(<\text{loop}>\).

The loop names, if present, are used in exactly the same way as the loop names read in with the labels keyword. Names in the loop file takes precedence.

A blank line in the loop file forces a blank line to be written to the ‘out’ file in the corresponding position. This is useful for creating line breaks in plots.

A loop file is used to generate a set of discrete tag values that can be used in the Phreeqc code. Each row of values is picked off in turn during an iteration of the z-loop, i.e., the number of rows determines the number of iterations.

4.7 Running Phreeqc without any plotting

The looping facilities in PhreePlot make it useful for some types of repetitive Phreeqc calculations which do not require a plot. Setting calculationMethod \(<0\) will suppress any plotting, as will setting plotFactor \(= 0\). If data are to be read from a data file, as in fitting, then the calculationType = "simulate" setting should be used to avoid calling the fitting routine. The "simulate" setting can also be used to make a set of simulations after fitting, e.g., to plot a simulated curve.

The input data file, which is probably most conveniently prepared in a spreadsheet or database and exported in csv or tab format, contains the data to be used. Tags are created from the headers.

The SIs.ppi file gives an example of the use of "simulate" for calculating saturation indices. It contains a translation table that assists in converting non-standard headings in the text data file to standard Phreeqc format. PhreePlot is used to loop one-by-one through a data file containing analyses of groundwater chemistry. It runs a small Phreeqc include file which contains
the USER_PUNCH code necessary to calculate various saturation indices and other parameters. This can be readily modified. The results are accumulated in the ‘out’ file.

The use of a data file for passing on information is somewhat similar to the use of a loop file (Section 6.2.1).

4.8 INCLUDE FILES

4.8.1 Use of ‘include’ files

The input files can contain INCLUDE statements to pull in other files, e.g.

INCLUDE ht1.inc

The text following the INCLUDE statement, here ht1.inc, is the name of a file. The filename can be optionally embedded in quotes. The normal rules apply for the search path when looking for include files (Section 2.4.6).

All of the statements in this file will be inserted line by line at the insertion point. This substitution occurs when the input file is initially read, before any code execution. This makes it possible to have a library of commonly-used pieces of code. The include statement is recursive – an include file can itself contain references to other include files.

The BASIC program runs strictly in the order of the BASIC line numbers not necessarily the sequence of lines in the file. If a line number is repeated, the last one read is used. This means that it is possible to add edits to an include file by including an ‘edit file’ after the main file (see e.g. ht1s.inc).

Using include files can reduce repetition of commonly-used code and make it easier to manage such code. It also can increase the readability of input files.

Phreeqc3 contains its own version of ‘include’ in the form of the INCLUDE$ keyword. This is a more powerful form of include than PhreePlot’s since it is ‘dynamic’ (dollar for dynamic!): the include file is read anew each time the directive is encountered. Therefore an earlier piece of Phreeqc code within the same run may write or modify the contents of the include file using PUNCH statements for example. In contrast, PhreePlot only reads the include file once – at the beginning, before the Phreeqc code has been executed. Conclusion – use INCLUDE$ if you want to read a file that is generated during a Phreeqc run. And if in doubt, use INCLUDE$. Remember that if the file is not found in the current directory, PhreePlot’s INCLUDE automatically checks the system directory whereas INCLUDE$ does not.

4.8.2 Supplied include files

Several include files are provided for commonly-used functions. These will be found in the system sub-directory. The uses of some of them are summarised in Table 4.1.

ht1.inc can be used to calculate a predominance diagram. If adsorbed species are present, then their concentration is considered on a species by species basis just like solution species. ht1combined.inc is similar except that all adsorbed species of one element and one surface are combined into a single species (a ‘superspecies’) for the purposes of the predominance calculations (ranking) and for plotting. Other include files are variations on these. See the examples in the \demo directory for their use.

4.9 USING PHREEQC’S _RAW AND _MODIFY KEYWORDS

Phreeqc (Version 3) introduced new keywords to retrieve and modify various existing data structures. These are based on existing keywords with the suffixes _RAW and _MODIFY. They are intended to provide more flexibility in the ways that the chemical system can be defined and modified, and provide ways of reading in data structures sent to a file by DUMP. These new keywords are not expected to be widely used.
They enable the updating of concentrations to be simplified and maybe speeded up. For example, the SOLUTION keyword always does an initial solution calculation whereas SOLUTION_MODIFY does not. It may also be possible to avoid initial exchange and initial surface calculations in an analogous way.

<table>
<thead>
<tr>
<th>file</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>htl1.inc</td>
<td>for calculating predominance plots</td>
</tr>
<tr>
<td>htl1combined.inc</td>
<td>as above but combines all adsorbed fields for a common surface</td>
</tr>
<tr>
<td>htl1cCO3.inc</td>
<td>into a single field; also gives an option of using the mineral stability criterion for identifying boundaries</td>
</tr>
<tr>
<td>htl1cStability.inc</td>
<td>as above but includes an additional total CO3 constraint</td>
</tr>
<tr>
<td>htl1s.inc</td>
<td>as above but includes the stability criterion</td>
</tr>
<tr>
<td>htl1minerals.inc</td>
<td>as above but also adds ‘(s)’ to the labels for mineral names</td>
</tr>
<tr>
<td>htl1_phase_formula.inc</td>
<td>as htl1.inc but also adds the mineral formula below the mineral name when labelling the plot</td>
</tr>
<tr>
<td>minstab1.inc</td>
<td>used for calculating traditional mineral (only) stability diagrams</td>
</tr>
<tr>
<td>htl1allminerals.inc</td>
<td>as htl1.inc but automatically adds all possible minerals as potentially precipitating mineral phases</td>
</tr>
<tr>
<td>printphases.inc</td>
<td>used to print the possible mineral phases to the Phreeqc.out file</td>
</tr>
<tr>
<td>speciesvspH.inc</td>
<td>used for making species-pH plots</td>
</tr>
<tr>
<td>logspeciesvspH.inc</td>
<td>used for making species plots with log y-scale</td>
</tr>
</tbody>
</table>

Table 4.1. Some of the supplied include files and their functions
5 PhreePlot input and output files

5.1 INPUT/OUTPUT FILES

5.1.1 Use

PhreePlot uses a number of files for input and output. The default 'settings' file, pp.set, is used to read in default values for all keywords. These are modified, and the Phreeqc chemistry part is added in the 'normal' input file (usually with a ppi extension), and finally keyword values can be overridden with the override.set file.

All of these input files are in ASCII text format and so can be read and written with a normal text editor. The input files determine the calculations that will be carried out. The extension is stripped from the input filename and this is used as the 'root' for automatically naming the output files. Many of the output files are optional and their production is set by a series of logical switches which can be set to TRUE or FALSE.

In 'Safe' mode (the way PhreePlot has currently been set), all the necessary files needed to produce the specified plots, and to be able to replot them, will be created even if their logical switches have been set to FALSE. Where file switches are specified to be FALSE, the corresponding files will be deleted at the end of the run if present, even if they were created from an earlier run.

Any existing files with the same name as the files to be created/deleted will be overwritten or deleted without warning.

5.1.2 Difference in execution of input files between PhreePlot and Phreeqc

Aside from the substitution of tags with values in PhreePlot input files, the CHEMISTRY part of a PhreePlot input files looks very like a Phreeqc input file, and in fact, it is often easier to test small pieces of code using Phreeqc or Phreeqc Interactive. When there is only one simulation in a file, there is essentially no difference in terms of execution.

However, where there is more than one simulation, PhreePlot has greater flexibility in the way that individual simulations are run. There are two key features here: (i) a separation between 'pre-loop' simulations and 'main loop' simulations, and (ii) the way that the main loop simulations are executed.

The modus operandi of PhreePlot is that some simulations may be required to set up the database, define other fixed things, prepare initial solutions etc and these need only be done once. These are called 'pre-loop' simulations. Following this in terms of layout and execution, there may be one or more 'main loop' simulations which are iterated or 'looped'. Normally there will be one or more tags in the 'main loop' part which will be altered during each iteration, thus varying the output, and ultimately preparing a set of data for fitting or plotting.

Phreeqc necessarily runs all simulations consecutively and without user intervention. Data structures are carried from simulation to simulation and some between-simulation user data can be transmitted via the Basic PUT(), GET() functions, but there are few other opportunities for dynamically altering values given in the input file.

PhreePlot has greater opportunities since it has the option to control the way that a multi-simulation input file is executed. As indicated above, there is the basic division between pre-loop and main loop simulations. There is another important option. PhreePlot feeds the input file that it has read into the Phreeqc calculator, line by line. Calculations are only begun
when a whole simulation has been read in (defined by an `END` statement) but **PhreePlot** decides when to look at the results of each simulation by exiting **Phreeqc** and looking at the output, updating the tag dictionary, making new substitutions etc.

These two options: (i) 'one simulation at a time’ mode or (ii) 'all at once’ mode. (i) cedes control to **PhreePlot** after every simulation has been executed which gives some opportunity to alter values for subsequent execution. (ii) is faster in execution but **Phreeqc** only returns to **PhreePlot** after all simulations have been executed. This means that there is no opportunity to intervene.

**PhreePlot** has somewhat arbitrarily made the decision that:

(i) all pre-loop simulations will be run 'one at a time' (being executed just once, speed is not such an issue while the added flexibility can be useful).

(ii) there is the option or running main loop either 'one at a time’ or 'all at once’.

These two features are controlled by the `mainLoop` keyword. e.g.

```
mainLoop 3 false
```

means that the main loop simulations start at simulation 3 (simulations 1 and 2 are therefore 'pre-loop') and that 'one simulation at a time' is false, i.e. all main loop simulations will be run together in one block.

The default in `pp.set` is

```
mainLoop auto false
```

where `auto` normally refers to the last simulation, i.e. the looping will only occur over the last simulation. For calculationType’s `fit` and `simulate`, `auto` is set to 1.

Using `debug` equal to 2 or greater will log the details of how the simulations are executed to the log file.

### 5.2 INPUT FILES

#### 5.2.1 Different types of input file

There are three main types of input files: (i) those that define certain keyword values or settings plus the chemical definition of the problem and dictionary files ('main input files'); (ii) those auxiliary files that provide additional data such as data for fitting and additional data or text for plotting ('data input files'), and (iii) those that contain chunks of **Phreeqc** code to be included in one of the main input files ('**Phreeqc** input files').

This section describes the first of these while the separators used for parsing input files is described in Section 5.2.7.

**PhreePlot** runs in response to the settings of various keyword-value pairs and lists. The values associated with these keywords can be defined in various ways (Figure 5.1). In running order, these are:
1. **PhreePlot** program defaults: set by **PhreePlot** internally; usually provide minimal functionality;

2. the *pp.set* file: user-defined default values read from a file; contains general preferences;

3. the main input file, the one given on the command line. It defines values for the particular problem of interest and normally contains the line ‘CHEMISTRY’ somewhere in it. This must be on a line by itself. It divides the input files into two with **PhreePlot** keywords in the upper section and **Phreeqc**-format chemistry, if any, in the lower section. It is best if this main input file is given the ppi file extension so that it can be associated with the **PhreePlot** program;

4. the *override.set* file: useful for overriding one or more settings without having to edit the main input file(s);

5. input made during interrupts during execution of **PhreePlot**: emergency redefinitions, e.g. changing the debug level (see Section 6.6).

The last defined keyword value or list of values is always used from its point of definition forwards.

The full list of keywords is normally given in the default *pp.set* file in the *system* subdirectory. The *pp.set* and *override.set* files should be in the *system* sub-directory if present. The *pp.set* file should be modified to set commonly-used attributes that remain constant between runs, including system-specific features such as the filepath for **Ghostscript** as well as a wide range of plotting parameters including the preferred units of length.

The override file (*override.set*), if present, is read after the input file and can be used to override any previously-defined values. It is especially useful for temporarily changing attributes for a whole series of files called via a batch file, e.g. changing the plot method, a font, a colour or turning the beep off.

### 5.2.2 Structure of the main input files

These files are the problem file (*.ppi), the *pp.set* file and the *override.set* file.

Although **PhreePlot** input files are rather unstructured, they logically divide into the following four sections:

- **SPECIATION**: Details of the speciation calculations
- **FIT**: Details of any fitting
- **PLOT**: Controls the plotting parameters
- **CHEMISTRY**: Contains the **Phreeqc** code.

The first three of these four sections headings may be included anywhere in the input files. These section headings are only included for improving the legibility of the files and are not used by **PhreePlot**. If present, the **CHEMISTRY** keyword signals the beginning of **Phreeqc**-type input and, must appear as the last entry in the **PhreePlot** section, i.e. the main input file must always end with the **CHEMISTRY** section if it is expected to do any chemical calculations.

The structure of a typical input file is therefore:

```
...<PhreePlot section heads and keywords that define various keyword settings and tag values. This section also defines the looping parameters and what type of plot, if any, will be produced>
...
CHEMISTRY
```
There is essentially no limit to the number of lines in the PhreePlot or Phreeqc parts. The chemistry line, which should be on a line of its own, defines the divide between the two sections and instructs PhreePlot to interpret the input accordingly.

The chemistry section includes the Phreeqc code. This can only be included in the main input file and any 'include' files called by the main input file. This determines what is calculated and has almost the same format as a normal Phreeqc input file. The principal difference is that it can contain special tags ('<...>') that are substituted with appropriate values before running Phreeqc.

Results from Phreeqc calculations are communicated to PhreePlot via the selected_output 'file' which itself is generated in response to the Phreeqc user_punch and selected_output blocks.

Therefore the chemistry section is essentially a Phreeqc input file with tags. The tags provide placeholders for substituting variable values generated by PhreePlot and give PhreePlot the ability to loop, fit data to models etc. The PhreePlot section defines how the tag values are generated and other aspects of the calculations including the plotting of results. The keyword values can be floating point, integer, character or logical.

### 5.2.3 The input file pre-processor

Where a simulation is repeated many times with the only change being an increment in one or more numbers, the simulations can be replaced with a single 'template' simulation which has tags indicating the start and end of the sequence to be repeated, the increment and where to substitute the generated number. This part of the code is executed before the input file is processed. The input file pre-processor is described in detail in Section 13.

### 5.2.4 Exceptions to the 'latest keyword definition overrides earlier ones' rule

Normally when a keyword and its settings are read, these settings will override all previous ones for this keyword. The exceptions are for numericTags, characterTags and overlay where multiple instances will add to the list of tags or overlay files to be processed.

### 5.2.5 The colour dictionaries and other files

There are two colour dictionaries which store the colours used for the lines, points and fills: (i) line colour dictionary for plots based on a custom plot, and (ii) the fill colour dictionary for predominance diagrams. Contour plots do not use the dictionaries.

These files can be edited to change any colours. For fill colours, this is all that need be done. For line and point colours, it is necessary to set useLineColorDictionary to 1 or 2 to force the use of the dictionary.

There are also data files for loop variables and for data to be used in simulations or fits.

### 5.2.6 Format of all input files

**Physical and logical lines**

All input files including data files have a similar structure. The input format conventions are similar to those used for standard Phreeqc input files. The maximum length of input lines is only limited by memory but most strings and character expressions are limited to 10000 characters.
A physical line is a text string ending with a normal line ending which for the Windows operating system is <CR><LF>. Each physical line appears as a distinct line of text in a text editor. It can consist of less than one, one or more than one logical lines.

A logical line is a string which is interpreted as a single block of data by PhreePlot. The keyword-value(s) combination of PhreePlot input files must always be present on a single logical line.

Strings containing spaces (or more specifically, separators) should be enclosed in quotes, paired single or double. One or more separators must precede and follow these quotes for it to be recognised.

Any input following a comment character (#) is ignored for the rest of that logical line. This includes ; and \ (see below). Blank lines and lines which are entirely made up of a comment are not counted as logical lines and are ignored. So

```plaintext
pdf TRUE \\
<BLANK LINE HERE>
png TRUE
```

will fail because the continuation will ignore the blank line and join the other two lines to give

```plaintext
pdf TRUE png TRUE
```

Logical lines are terminated by a normal line ending, or by a semi-colon (;). A comment character takes precedence over ‘;’. For example, the comment in

```plaintext
# PHASES; Fixed_H+; H+ = H+; log_k 0.0
```

will comment out all four logical lines.

The above rules for # and ; even apply when embedded in quotes. Their special behaviour takes precedence over quotes. Therefore it is not possible to use these characters even in quoted text strings, i.e. “Sample #76” and “bloomington; Rochester” would produce errors. This is not true in Phreeqc where the quotes take precedence.

A logical line may be split across two or more physical lines by using a continuation character. A continuation character is a backslash (\) providing it is present as the last non-whitespace character on a physical line after comments have been removed. So unlike Phreeqc, this is true even when the \ is followed by spaces and a comment character, i.e.

```plaintext
numericTags <logH> = -<x_axis> \ # comment
<ph> = -<logH>  # -pH
```

is valid as is

```plaintext
#pdf \\
T
```

is not.

Tabs, being whitespace, are ignored by Phreeqc but can be significant in data files. Consecutive tabs signify a blank fields. However, a line consisting only of tabs is treated as a blank line. Data files prepared by pasting from a spreadsheet into a text editor can contain tabs, including trailing tabs. Such files should be read using the tab as a separator. This is done by adding "\t" after the filename. An alternative to this way of reading null values is to explicitly use the ‘missing data’ code, namely -99999.

### Specifying keyword-value pairs and keyword-lists

Keywords and their values are separated by any number of separators on a logical line.

Quotation marks should always be used when there is a space or tab embedded within a character variable. A null character variable is entered as a pair of quotation marks with or without
one or more blanks, e.g. ‘ ‘, ‘ ” or “ “. It is necessary to use this format when entering a blank value for a character variable.

The following are some examples of valid input lines:

```plaintext
jobTitle  Iron
jobTitle  "Iron hydrolysis"
jobTitle  ‘Iron hydrolysis’
calculationType htl;
calculationType htl
calculationType htl #This is a comment
pxmin  0; pxmax 10; pymin -10; pymax 20
pxmin 0pxmax 10 pymin 10 pymax 20
```

**Format of keywords and their associated values**

Most keywords are followed by a single value of a specified type, either an integer, a floating point number, a character string or a logical. Case is not significant except within character strings and tags. Some keywords are followed by a list of variable length, e.g. `mainspecies`.

An integer is any set of digits with or without a sign. A number is any set of digits with or without a valid exponent (in E format), decimal point or sign and includes all integers. A character string is any set of valid characters (see below), and is optionally placed within a pair of delimiters (a pair of single or double quotes). A logical value can be entered as `TRUE` or `FALSE` or `T` or `F`, irrespective of case. Examples are given below, each value being separated by a comma:

- **Integers:** 0, 12345
- **Numbers:** 1, 2., 3.1, 4e0, 5E0, 6d0, 7D0
- **Character expressions:** PhreePlot, "PhreePlot", "PhreePlot program", "", " " "This is "PhreePlot", "This is ‘PhreePlot’"
- **Logical:** t, T, true, TRUE, True, f, F, false, FALSE

5.2.7 Data separators and parsing input files

All of the input files consist of a set of logical lines with a collection of zero or more ‘words’ on a line. The difference between physical and logical lines is described above. In many cases, the first logical line of a file is used as a ‘header’ to describe the data that follows. In some cases, these header names are converted to variable (tag) names for the columns. All of the input files are read in ‘free format’, i.e. the column position of the entry on the line is not important.

The words on a line are separated by ‘data separators’, sometimes called delimiters. The parsing of input files (separating the words) depends on the structure of the input file and the data separator(s) specified. You have to ensure that the two match so that the file can be parsed correctly.

Commonly-used separators are spaces, (horizontal) tabs and commas.

The main input files are read in ‘very free format’ in which case all of the three main separators – a blank, tab or comma – are treated as valid separators and consecutive separators of any sort are treated as a single separator.

Quotes should be used to specify character strings containing these separators (Section 5.2.6). A quoted string should always be followed by a separator or an end-of-line marker. If not, the text after the closing quote is added to the quoted part of the string and an additional quote added to the end of the string.

Spaces and other special characters (other than ; and #) enclosed within quotes (single or double) are treated as part of a character string and will not be split.

Since data files such as those used for fitting (datafile) or plotting (e.g. extradat) can come from many sources and can include blank fields, a somewhat more rigid type of ‘free format’ is required for this type of file. The default data separator is always taken from the first entry in `dataSeparators` but this can be overridden by appending a format string after the filename.
Valid entries for this format string are:

- \w\ signify whitespace (one or more blanks or tabs)
- \b\ signify one or more blanks
- \t\ signify a single tab (often found in files derived from spreadsheets)
- ",\ signify a single comma (for csv files)
- \"\ or " (null string) signify whitespace or one or more commas (if at the end of a line, make sure that \ is embedded in quotes otherwise it will be interpreted as the line continuation character). This is the ‘very free format’ option that is used to read the input files and will read many of the most-common types of formatted files.
- char\ where char\ signify any valid single character.

Note that when a single tab, comma or character are used as separators, consecutive separators will define a blank field. This means that blank fields can be preserved when reading files based on single character separators such as those produced by Microsoft Excel and OpenOffice Calc.

5.2.8 Case sensitivity of input

Most of the text in the main input files is case insensitive. This includes all keywords. The only exceptions are the names of tags (anything between angle brackets) including text within tags (e.g. <input...>) and the names of column headings used to define columns in a data file – these are case sensitive. File names under Windows are not case sensitive.

In general terms, things that have been defined by PhreePlot are not case-sensitive whereas things that you have defined are case sensitive.

5.2.9 Reporting of errors in input files

PhreePlot stops if it detects errors in one of the keyword input files. These errors include syntax errors as well as any errors based on ‘compile time’ inconsistencies in the settings. These are signalled with an error message to the screen and to the log file if open, e.g.

File: test.ppi:6
Keyword: xmax
Input: xmax 1x2
Error: Expecting a number.

The message indicates the file involved and the position in the file where the error was detected in terms of the physical line (here line 6). The keyword involved (if known), the line of text where the error was detected, and an error message are also given. These may not identify the location or prime source of error exactly but should be sufficient to help to identify it.

Where an error occurs within a section where continuations (\) are being used, then the physical line indicated will be the end of the position of the offending section and the word will be a negative number indicating the number of words to count back to the offending input (roughly).

The whole input file is checked before reporting the error and exiting. Where several errors occur on the same logical line, only the first error will be reported.

Errors in other input files are signalled in a similar way.
5.3 TAGS

5.3.1 What are tags used for?

Tags are special symbols which have values associated with them (i.e. variables). Each tag has a name which is a text string embedded within angle brackets, e.g. `<x_axis>` is the tag that holds the current value of the x-axis variable.

When tags are used within a file, they are essentially place markers which indicate where various substitutions are to be made at a later time. Tags can be used to define ‘global’ variables that retain their values between simulations. Some tags are automatically created by PhreePlot in order to make their values available for subsequent use.

Tags come in two flavours. Tags can be either numeric or character depending on the type of value that they represent.

Each tag has a tag expression associated with it. These are evaluated before running Phreeqc and the derived tag values substituted in the appropriate places within the script. The tag expressions for numeric tags can include simple arithmetic expressions as well as more complex expressions using other previously-defined tags.

If the substitutions are not made correctly within the Phreeqc code section, e.g. because a tag is not recognised or not defined, Phreeqc will normally fail because of the illegal characters found at run time.

Tags can be placed anywhere in an input file and in the optional extraText and extraSymbolsLines files. There is no limit to the number of tags used.

Tags, combined with the looping facilities within PhreePlot, are used to vary the calculations made by Phreeqc in a dynamic way. Some tags are defined by PhreePlot, others are defined by the user. They provide a simple way of defining variables and of giving Phreeqc some basic looping capabilities without changing the format of the Phreeqc input file greatly.

5.3.2 Rules for choosing tag names

Tag names should always start and finish with open and closed angle brackets, e.g. `<tagname>`. Tag names should preferably be restricted to upper and lower case letters, numbers and the underscore. Other characters can be included although they will be temporarily replaced by a full stop (‘.’) and so tag names with multiple characters such as +, - etc. can become degenerate. This also applies to tag names automatically created from the selected output and fit data files and from the fit parameter names so the parent names should also follow this advice.

System tag names are ‘reserved’ names and should not be redefined. These are: `<x_axis>`, `<y_axis>`, `<loop>`, `<logloop>`, `<mainspecies>`, `<timedate>`, `<nexecute>`, `<systime>`, `<Phreeqc_status_0>`, `<pxmin>`, `<pxmax>`, `<pymin>`, `<pymax>`, `<p2ymin>` and `<p2ymax>`.

Case is significant in tag names. There is no length limitation to tag names.

5.3.3 Tag expressions

Tag expressions are the text on the right-hand side of a tag equation. These are stored as character strings and can contain any valid combination of numbers, character strings and other tag names (see below). The tag expression can be any length. Long expressions can be subdivided and saved as separate sub-expressions.

Numeric tags can represent variable values; character tags always refer to constant string expressions.

5.3.4 Numeric tag expressions and available functions

Numeric tag expressions can be simple numbers such as 1, 1.2, 1.2e2, 1.2d-4 or arithmetic expressions such as 2*4, log10(3.5), 2+(3*4) etc. The arithmetic operators available are: +, -, *, / and ^ (exponentiation).
The following functions are also allowed: abs, exp, log10, log, sqrt, sinh, cosh, tanh, sin, cos, tan, asin, acos, atan, rand and nrand.

The random number generators, rand and nrand, return a single pseudo random value generated from the uniform (range = (0,1)) or normal (mean = 0, standard deviation = 1) distributions, respectively. Both functions take a single integer argument which acts as a seed. If the seed is positive this value is used to start the distribution. Using the same seed on different runs means that the same pseudo random sequence will be generated. Using 0 or a negative integer value for the seed means that the system date and time are used to generate the start of the random sequence. This will ensure that a different sequence is started for each run.

Parentheses are used for specifying precedence in the normal way. All numeric tags are stored and evaluated with high precision (double precision).

Tag expressions can also include other tag variables providing that they have already been defined. The sequence in which the tags are evaluated is controlled by the order in which they are defined.

Undefined numeric tags have the value UNDEFINED which is stored internally as -99999.

Valid tag expressions are:

\[
\log_{10}(\text{<loop>}) \\
2\times\text{<x_axis>}
\]

When tag values are substituted, they are rounded and then trimmed of leading and trailing spaces. Therefore

\[-\text{<x_axis>}\]

should work providing that the value of \(<x_axis>\) is not negative.

5.3.5 Tags for character variables

These tags are substituted at the indicated positions. If the tag expression contains spaces, enclose in quotes. Quotes are always removed before substitution and so they may need to be used to surround the tag expression, e.g. "<mainspecies>" so that the substituted expression is correctly parsed.

5.3.6 System tags

Certain tags are automatically created and updated by PhreePlot. These are:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;x_axis&gt;</td>
<td>the value of the x-axis variable (numeric)</td>
</tr>
<tr>
<td>&lt;y_axis&gt;</td>
<td>the value of the y-axis variable (numeric)</td>
</tr>
<tr>
<td>&lt;loop&gt;</td>
<td>the value of the z-loop variable after being exponentiated (10^x) if necessary (numeric)</td>
</tr>
<tr>
<td>&lt;logloop&gt;</td>
<td>log_{10} of the value of the z-loop variable (numeric)</td>
</tr>
<tr>
<td>&lt;mainspecies&gt;</td>
<td>the name of the main species (character)</td>
</tr>
<tr>
<td>&lt;nexecute&gt;</td>
<td>the number of times that Phreeqc has been called</td>
</tr>
<tr>
<td>&lt;timedate&gt;</td>
<td>a time (hr:min:sec.millisecond)/date string with the format 23:30:45.123 23 September 2004 (character)</td>
</tr>
<tr>
<td>&lt;systime&gt;</td>
<td>uses the system clock to get the elapsed time (in sec) since the beginning of PhreePlot execution (this reflects 'wall time' not cpu time). Resolution about 1 msec.</td>
</tr>
<tr>
<td>&lt;Phreeqc_status_0&gt;</td>
<td>returns the exit status from the last Phreeqc run: 0 = OK, &gt;0 = error</td>
</tr>
</tbody>
</table>
These tags are known as 'system' tags and are updated throughout a run. They have reserved names, effectively making them 'read-only'. They should not appear on the left-hand side of a tag expression.

There are also some additional tags which are created on start-up or after fitting:

- `<command_linen>` the values of the n command line arguments (n=0 gives the name of the pp.exe file, n= 1 gives the name of the ppi file, n=2, 3, 4, ... give any additional arguments) (character)
- `<R2>` the value of R² after a successful fit, see Section 5.3.7 (numeric)
- `<RMSE>` the RMSE after a successful fit (numeric)
- `<nFit>` the number of 'observations' in a fit (numeric)
- `<fitMethod>` the fitMethod (char)
- `<objectiveFunction>` the objective function, “L1”, “L2” etc (char).

If a loop file is read, then a separate tag value is automatically created for each column. These are named `<loop1>`, `<loop2>`, ... for column 1, 2, ... . They will also be named after their column header names, if present.

Other tag names are reserved for formatting text (Section 7.6.3). These are:

- `<sub>` and `</sub>`
- `<sup>` and `</sup>`
- `<i>` and `</i>`
- `<b>` and `</b>`
- `<g>` and `</g>`
- `<subsup>` and `</subsup>`
- `<br>`

Other tags are automatically defined by the user or automatically created during the course of calculations (Section 12.12) and can be used in the extraText file.

### 5.3.7 User-defined tags

User-defined tags are defined in one of the input or data files using the numericTags or characterTags keywords. The order of definition is important since once defined, these tags can themselves be used to define new tag values in subsequent tag expressions. Ultimately every numeric tag will need to be evaluated to provide a numeric value ready for substitution in an input file.

The order of evaluation of numeric tags is given in Section 5.3.8.

Examples of numeric tag expressions are:

```plaintext
<log_H> = 7.5
```
Substitution takes place on each iteration just before the Phreeqc calculations are performed.

Tag types cannot be mixed in an expression. Numeric tag expressions can include other numeric tags but character tags must not be used in numeric tag expressions neither must numeric tags be embedded in character expressions. For example

\[ <x> = "4 + <x_axis>" \]
\[ <y> = "6 + <y_axis>" \]
\[ <z> = <x> + <y> \]

where \(<x>, <y>\) and \(<z>\) are all valid numeric tags. Note the use of quotes where the tag expressions contain spaces. The tag expression appear as a single text string.

\[ <c1> = "Title" + <x> \]
\[ <c2> = <mainspecies> + 6.3 \]
\[ <c3> = <x> + <c1> \]

where \(<c1>, <c2>\) and \(<c3>\) are character tags, are invalid tag definitions because the addition mixes character and numeric tags.

Numeric and character tags can be defined in extradat files in much the same way that they are defined by selected output.

5.3.8 The scope of tags, their initial values and their order of evaluation

Each tag is defined by a tag expression. Since these expressions can themselves contain tags, it is important to understand the order in which they are evaluated in order to avoid a tag referring to an as-yet undefined tag or getting an out-of-date value for a tag.

Tags can be defined in a number of ways: by PhreePlot itself (system tags), from user-defined tags in an input file, from reading a loop file or fit data file, from the selected output file, or as a result of fitting.

The tags, their expressions and their current values are stored in a tag dictionary. This is used to substitute the values of any tags found in the input files before carrying out the next simulation. The whole tag dictionary is available for all simulations.

Tags can also be used in any text strings that are used in plotting: plotTitle, xtitle, points and lines and their 2y equivalents, customXcolumn and in extraText files. Tags used in the 'lines' and 'points' lists can themselves be lists.

The initial value of all numeric tags is set to UNDEFINED (-99999) but can be set to another value with the initialValue keyword. This same value is applied to all undefined numeric tags.

The initial value of character tags is set to the null or empty string.

It is also possible to set the initial value of an individual tag by using a pre-loop simulation, e.g.

```
SOLUTION
SELECTED_OUTPUT
   -reset FALSE
USER_PUNCH
   -heading z
10 PUNCH 0.0
END
```

will set the initial value of the \(<z>\) tag to 0.0. SOLUTION ensures that the selected output is actually written. A pre-loop simulation is only executed once.

After a block of one or more simulations has been computed by Phreeqc, new tags are formed if appropriate and all tag values are updated. This updating only occurs after all the simulations within the block have been completed. Phreeqc has complete control during this execu-
tion phase. Therefore tag values cannot be passed from one simulation to another when they are part of the same execution block since execution does not leave the Phreeqc module and so the tags cannot get updated by PhreePlot. This has implications on how the input file is set up.

Tags can be used in the ‘upper’ part (the part before chemistry) of an input file as well as the in Phreeqc section (the part that follows chemistry). Tags can also be in extraText and extra-SymbolsLines files which, if present, are updated during the plotting.

Once set, tag values retain their values until reset. Tags provide a simple mechanism for passing a numeric value from one Phreeqc simulation to another as well as for providing the values of certain system variables which can be used for looping and other tasks. Tags can also be used during the plotting phase to control text input and its positioning.

The order of evaluation of numeric tags (first to last) is:

- system tags defined internally by PhreePlot (e.g. <x_axis>, <y_axis>, <loop>, ...);
- independent variables from the column headers in a loop or data file;
- extradat file tags tags defined by a two-line (header and data) extradat file;
- USER_PUNCH tags the names of the tags defined in the header line of the SELECTED_OUTPUT file created in a Phreeqc USER_PUNCH data block within the Chemistry Section. The values are those that were ‘punched’;
- fit parameters from fitParameterNames and fitParameterValues as defined in one of the input files, or after fitting (e.g. <R2> etc);
- user-defined tags from numericTags in one of the input files defined in the order the files and tags are read.

The input files are read in the order:

1. the user-defaults file (pp.set)
2. the main input file (*.ppi)
3. the override file (override.set).

The tags should not be used until after they have been defined, e.g. if a tag is defined in the second simulation, it should not be used until the third or later simulations). Substitution of all tags takes place before execution of a simulation and so the values of tags created during an execution cannot be used to update other tags that depend on the values created during that execution. For example, USER_PUNCH tag definitions cannot refer to other USER_PUNCH tags defined in the same simulation.

The input files can contain tag expressions which may themselves refer to other tags whereas the system, user punch, and fit tags are generated automatically by PhreePlot and simply have numeric values associated with them. In that sense, only the order of tags defined in the various input files is of significance.

Tags can be used to pass numeric values from one simulation to a later one (this is also possible using Phreeqc’s PUT/GET mechanism). For example, it may be wanted to subtract the initial value of a calculated variable such as the pH from all subsequently generated pH values to find the change in pH. This can be done by first generating the initial pH in a simulation. Send this pH to the selected output using a USER_PUNCH data block and a column name such as ‘pHorig’. This will automatically generate a tag <pHorig> with the desired value which will be stored for the duration of the run. In subsequent simulations, either write the new pH to the selected output using a new column name such as ‘pH’ or generate the change in pH directly in the USER_PUNCH data block (e.g. la("H+") - <pHorig>) and output this difference directly to the selected output. In the first case, this will create a <pH> tag with a new value each time the simulation is run. Use numericTags to subtract the two.
In order to see which tags have been defined and their values, set `debug=1, 2 or 3` and enable the log file (`log TRUE`). A list of the tags defined at each iteration and the values of all tags used will then be written to the log file. These tables can be used to identify undefined tags.

### 5.3.9 Examples of the use of tags

The following example shows how user-defined tags can be used to manipulate the Phreeqc input file when there are multiple simulations (`ENDs`) and when the early simulations prepare for looping on the final simulation.

Adsorption is defined with the `SURFACE` data block and in one of its forms requires values for the number of adsorption sites (in moles), the specific surface area (in m² per gram) and the mass of adsorbent (in grams).

```
SURFACE
surface binding-site name, sites, specific_area_per_gram, mass
```

In order to see how the amount of metal adsorption might vary in a system as the specific surface area changes, it is reasonable in the first instance to assume a constant surface site density (i.e. a constant number of sites per m²). For the weak sites of HFO, we have

\[
isite = \text{sdm}/\text{gfa}
\]

and

\[
\text{sdm} = \text{isite}/\text{isa}
\]

where

- `isite` = initial number of sites (mol)
- `sdm` = density of sites per mol HFO (mol/mol),
- `gfw` = gram formula weight of HFO,
- `isa` = initial specific surface area (m²/g),
- `sd` = site density of sites (mol/m²)

and

\[
\text{m} = \text{mass (g)}.
\]

So for any other specific surface area, `sa`, the number of sites (mol) is

\[
isite = sa*sdm*m.
\]

This can be implemented by defining a series of numeric tags as follows:

```
<initial_site_density_w_per_mol> = 0.2 mol/mol Fe \\
<gfw> = 89 g/mol \\
<initial_site_density_w_per_g> = <initial_site_density_w_per_mol>/<molecular_wt> \\
<initial_specific_area_per_g> = 600 m²/g
```

Note the structure is

```
<tag> = <tag_expression>
```

where the tag has a unique name (case significant), followed by an equal sign (spaces optional), followed by an expression. The expression can contain other tag values provided they have already been assigned a value.

Therefore

```bash
<site_density_w_per_m2> = <initial_site_density_w_per_g>/
```
and for any surface area and mass, we have

\[
\langle \text{surface area} \rangle = 300 \text{ m}^2/\text{g} \\
\langle \text{mass} \rangle = 1 \\
\langle \text{sites} \rangle = \langle \text{surface area} \rangle \times \langle \text{site density w/ m}^2 \rangle \times \langle \text{mass} \rangle
\]

The final tag defines the number of sites as required by Phreeqc. The density of strong sites can be defined similarly.

The Phreeqc input is therefore given as:

SURFACE 1
  - equilibrate with solution 1
  Hfo_w <sites> <surface_area> <mass>

In order to create a plot of the amount adsorbed vs surface area, \langle surface_area \rangle must be replaced by \langle x_axis \rangle or redefined as such, and the amount adsorbed must be written to the selected output file. An example that implements this procedure is given in Example 63.

5.4 OUTPUT FILES

The various output files are used internally for storing intermediate data as well as the data actually used for plotting (and later replotting). The output files can be used to examine in detail the Phreeqc output, the intermediate results generated by PhreePlot, or to export data to other packages for further analysis or plotting. If the structure of the Phreeqc input file is relatively straightforward, PhreePlot provides a quick way of looping through Phreeqc calculations that would otherwise be rather tedious to set up (see Section 6.2 and Example 71). Phreeqc-type calculations can be made without generating any plot files by setting plotFactor to 0.

5.4.1 Output files produced

The output is sent to a variety of files, most of which derive their names from the root of the input filename with an added extension. For example, if the main input file for a htl or grid calculation is C:\PhreePlot\demos\amd\amd.ppi, then the root is taken as C:\phreeplot\demos\amd\ which will then produce a series of files with the general format

root__[mainspecies][loopIndex].ext

where root is the root, mainspecies is the name of the main species and is only included when the number of main species is greater than one, loopIndex is the index value (1...nz) of the loop variable and is only included when number of loop values is greater than one. ext is the output file extension. Possible extensions are:

<table>
<thead>
<tr>
<th>Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>log</td>
<td>log file (filename is simply &lt;root&gt;.log)</td>
</tr>
<tr>
<td>pts</td>
<td>points file (boundary points, species distribution)</td>
</tr>
<tr>
<td>trk</td>
<td>tracking file with output from each Phreeqc iteration</td>
</tr>
<tr>
<td>vec</td>
<td>vector file for predominance and contour plots including the boundaries</td>
</tr>
<tr>
<td>pol</td>
<td>polygon file for predominance and contour plots defining each field</td>
</tr>
<tr>
<td>ps</td>
<td>Postscript graphics file, potentially multi-page (not eps)</td>
</tr>
<tr>
<td>eps</td>
<td>encapsulated Postscript (eps) graphics file with bounding box suitable for embedding in documents (single page only)</td>
</tr>
<tr>
<td>epsi</td>
<td>encapsulated Postscript (epsi) graphics file with bounding box and a platform device independent preview suitable for embedding in documents (single page only)</td>
</tr>
</tbody>
</table>
pdf  pdf (portable document format) file
png  png (portable network graphics) graphics file
log  log file containing details of run
lab  editable labels file giving the position and orientation of labels in a predominance diagram. Edit and replot with calculationMethod 2 or 3.
out  tabular output file with accumulated results from the selected_output file used as input by some of the plotting routines.
all  cumulative output from Phreeqc

All files are output as space or tab-delimited ASCII files. The first five of these files are mainly for PhreePlot's internal use and for debugging while the latter ones provide graphical and text output for further analysis. The output from each file can be turned on or off using one of the logical keywords assigned to the file type (Section 5.4.2).

Looping of the main species variable always produces a separate 'out' file for each value of the main species variable.

With custom and fit calculations, looping of the z-loop variable produces a single 'out' file with, by default, a blank line separating each value of the loop variable. This blank line can be suppressed by setting the fourth entry in dataSeparators to the null string, "".

Whether the corresponding plots are in separate files or not depends on the multipageFile setting.

For example, assuming the multipageFile setting is set to FALSE and ps is set to TRUE, if there are two main species, Fe and Zn, and two iterations of the z-loop variable, then the following four files will be produced:

C:\phreeplot\demos\amd_Fe1.ps
C:\phreeplot\demos\amd_Fe2.ps
C:\phreeplot\demos\amd_Zn1.ps
C:\phreeplot\demos\amd_Zn2.ps

Other extensions are generated as appropriate. The log file in the example above will be called C:\phreeplot\demos\amd.log.

If multipageFile is set to TRUE, a single file will be produced:

C:\phreeplot\demos\amd.ps

and it will contain all four plots in the order given above.

With custom and fit plots, only one plot file is produced even when the <loop> variable is used. The results from the various loops are all plotted on the same plot with the labelling (from the selected output file column headings or the labels keyword) appended with an underscore and the loop number. This file has the same name as the multipage file above.

The file plot.ps is a copy of the last Postscript file generated and is always produced if a plot is generated. It is also the name of the file generated as the temporary tracking plot file when the 'p' key is pressed during predominance calculations (Section 6.6), or when the calculations are interrupted ('Esc') and then terminated ('s'). Derivatives of the ps file under such circumstances will be given the corresponding names such as plot.pdf, plot.eps etc.

5.4.2 The logical switches

Each output file has a logical switch associated with it (true or false). These are designed to give the user some control over the number of files produced. In general, the output data (text) files that are needed for plotting will be created whatever the value of their logical switch and they will not be deleted by PhreePlot at the end of a run. This ensures that plots can be edited and replotted without recalculating the underlying data. The most important switches for the user to control are for the log and track files which can both be large and are informa-
tive rather than being essential for the plotting (except that the track file is used to replot a grid plot).

Providing \texttt{plotFactor} is greater than zero, the following files will definitely be produced and retained during the following types of plots: \texttt{ht1} – points, vectors, polygons and labels; \texttt{grid} – track; \texttt{custom} and \texttt{species} – output; \texttt{fit} and \texttt{speciate} – points and output. If any of these files are deleted manually, then it will not be possible to carry out a ‘replot’; it will be necessary to recreate them. Sometimes the files are created but nothing is written to them. This will result in zero byte files.

The primary output data file produced by the \texttt{ht1} method is the points file. The vector, polygon and labels files are generated from this. These four files are all used to create the ps plot file during plotting and replotting. If the ‘reprocess and replot’ method \texttt{(calculationMethod 3)} is requested, then the calculations start with a pre-existing points file and recreate the other files anew. This will regenerate all the label positions and if appropriate, rewrite the labels file. This setting should be used when the \texttt{yscale} is changed. Replotting alone \texttt{(calculationMethod 2)} starts with all the existing data files including the labels file and then regenerates the plot files. With this setting, editing the labels file can be used to move the labels.

The grid plot uses the track file as the primary data file. The custom and fit plots use the output file. The fit plot also uses the points file.

Unlike the ‘data’ files, the plot files are all optional and the logical switch determines whether they are created or not, or more specifically, whether they are retained at the end of a run. There are some interactions amongst the various plot file type because the ps file is the primary plot file and is required in order to produce all the other graphic files using \texttt{Ghostscript}, e.g. if the settings are \texttt{ps \texttt{FALSE}} and \texttt{pdf \texttt{TRUE}} then the ps file will be created because it is necessary in order to produce the pdf file but it will be deleted at the end of the run because the ps switch was set to \texttt{FALSE}.

5.4.3 ‘log’ file \texttt{(log)}

The log file provides a log of the calculations performed and for monitoring progress. It is most useful for debugging. The amount of information sent to the log file can be very large. It increases as the \texttt{debug} parameter increases from 0 (small) to 3 (large). A copy of the main input file (i.e. the file specified on the command line) is written to the log file. This includes any comments. If the \texttt{writeInputFiles} switch is set to \texttt{TRUE}, then all input files are written to the log file. This means all ‘include’ files plus the \texttt{override.set} file.

5.4.4 ‘out’ file \texttt{(out)}

The ‘out’ file accumulates the selected lines of output from the main selected output file \texttt{(n = 1)} and is used by custom plots for plotting. The format of the ‘out’ file depends on the task being undertaken. In general, the output file contains the accumulated information sent by \texttt{Phreeqc to PhreePlot} via the \texttt{SELECTED_OUTPUT [1]} file. This does not include all the lines in the selected output but only those chosen by \texttt{.} By default, this is just the last line of the last simulation (assuming that this contains the chosen selected output block). It is assumed that earlier lines are from unwanted intermediate calculations such as initial solution calculations.

No output file is created during a replot (or resimplify).

In fit mode, the output file is an accumulation of the selected output from each call to \texttt{Phreeqc}, one line per data point. The order of the headings is dictated by the way in which the \texttt{SELECTED_OUTPUT} data block was written but should definitely contain the dependent variable (fitted value) and probably the independent variables (see the \texttt{-headings} line in the \texttt{USER_PUNCH} block).

In ‘grid’ and ‘ht1’ calculations, the ‘out’ file contains a cumulative version of the data sent to the selected output using a \texttt{PUNCH} statement. After a header line, the ‘out’ file has the following format:
an ordered list of species name, species value pairs for each of the five species types given below. The lists run consecutively one after the other. If there are no entries for a species type, then nothing is written. The line ends with five integers which serve as counters for each of the five lists. The five counters tell PhreePlot how many name-value pairs of each type have been PUNCH’ed. They are used by PhreePlot to read the data and construct a predominance diagram. The name-value pairs can be seen being set in the ht1.inc and htic.inc include files.

In summary, the five counters are:

nout1 the number of candidate predominant solution/adsorbed/exchanged/mineral species and their ‘concentrations’ PUNCH’ed. This can be any number but, after PhreePlot sorts them by concentration, only the largest three at most are passed on for further consideration. At present, only the largest two – the dominant and sub-dominant species – are actually used by PhreePlot.

nout2 the number of mineral species and their concentrations PUNCH’ed – if one or more minerals is actually stable and present, then when the stability criterion is used. these will be sorted in terms of ‘concentration’ and the one with the largest concentration will take precedence over all solution species. This can be any number but, after PhreePlot sorts them by concentration, only the largest three at most are passed on for further consideration.

nout3 the number of constraints PUNCH’ed as name-value pairs – these may override the nout1 species (e.g. they can be used to impose the ‘water limits’). This can be any number.

nout4 the number of ‘carry’ variables PUNCH’ed as name-value pairs – these are numeric values that are not used in predominance calculations but which you might want to examine for some other reason. These ‘species’-value pairs are sent to the ‘out’ and ‘trk’ output files for viewing and are summarised in the log file. The values are also added to the tag dictionary. This can be any number.

nout5 the number of ‘system’ variables returned. This must be 5! Always in the order: pH, pe, PO2(g), PH2(g), temperature (°C).

More details about the expected format of the selected output are given in Section 4.5.

With predominance and contour plots, a blank line is written to the ‘out’ file when no selected output is produced when it should have been – e.g. when the speciation has failed or when a calculation has been skipped because it is not in the calculation domain.

5.4.5 ‘track’ file (trk)

This records the results of each speciation calculation as performed for a predominance plot (both ht1 and grid). It is generated from the selected output after species coding. The header line and the first data line look like this:

```
n x y pe T step species1 isp1 species2 isp2 c1 c2
1 -12.0000 -85.0000 -12.4518 25.0000 -11 "H2(g) > 1 atm" 4 "Fe(OH)3-" 1 0.90370 -2.0100
```

n is the sequential number of the speciation calculation; x and y are the x- and y-values, pe is the calculated pe, T is the temperature, step is the type of step being taken (see Section 3.2), then the dominant and sub-dominant species (after any overrides have been applied) with their names (species.), species code number and concentration (mol/kgw). These data are normally echoed to the screen during execution.

The track file can also be turned on during fitting. It will contain a copy of the convergence monitoring with the number of the function evaluation, value of the objective function (RSS)
and its logarithm, and a list of the current parameter values. If multiple fitting methods are chosen, then the headers to the track file are subscripted with the names of the method, e.g. \texttt{RSS\_nlls}, \texttt{log10\{RSS\}\_bobyqa}, \texttt{logK\_nlls} etc. This differentiates the results from different algorithms for the purpose of plotting.

5.4.6 ‘points’ file (\texttt{pts})

The format of the points file depends on what task created it.

\textit{Predominance plot calculations (ht1 only)}

This records the results of calculating boundary positions (see Section 8.3.2). These points are the raw data for calculating predominance diagrams. The number of points can be very large (even while tracking a straight line) and so these data are subjected to simplification to reduce the size of the file and to improve the appearance of the diagram. It is these simplified lines that are written to the vectors file and then to the polygon file.

\textit{Other calculations}

This is an ASCII file in space-delimited format. It is created by a ‘species’ plot and contains a table of species suitable for plotting. This variables in this file are automatically added to the search list of variables and so can be used explicitly for plotting.

5.4.7 ‘vectors’ file (\texttt{vec})

This file is generated from the points file and it stores the boundaries of the fields in a ‘ht1’ predominance diagram as a series of vectors. The order of the vectors is determined by the order of tracking. The individual polygons are assembled from this file, and coloured accordingly. These vectors divide two fields. They are not drawn for the domain boundary. The file looks like this:

\begin{verbatim}
250 1190 3562 16 25.000000000000 -0.9995420904108 0.2499810326136 20.79331349261 -12.00000000000 -83.30000000000 4 1 1 -12.04377661614 -12.00000000000 -83.12734375000 4 1 1 -12.00061311406 -10.74000000000 -83.13000000000 4 1 1 -10.74144424834 -10.74000000000 -81.77000000000 3 1 2 -10.74144424834 -10.74000000000 -81.77000000000 3 1 2 -10.40145620289
\end{verbatim}

The first line contains the values of various system parameters that were in force when the file was created: the resolution, the number of points in the points file, the number of times that the speciation program was executed, the number of species recorded as dominant or sub-dominant, the temperature of the last speciation and three numbers defining the relationship between pe and the x- and y-axis variables.

The subsequent lines have the following columns:

\begin{verbatim}
x-value  y-value  species1  species2  vector_sequence_number  pe
\end{verbatim}

where x- and y-value are the x and y values, species1 and species2 are the integer codes for the two species at the boundary, vector_sequence_number is the sequential number assigned to the vector and pe is the calculated pe returned by \texttt{Phreeqc}. In a contour plot, the pe is replaced by a distance parameter – this is a scaled distance reflecting the relative distance of the point from the drawn line. It is a measure of the influence or importance of the point – large distance means large influence.

A species code of 99 indicates a domain boundary.

This file is used in ‘ht1’ plots to draw the outlines of the predominance fields. Commenting out or deleting lines from this file combined with \texttt{calculationMethod} = 2 can be used to omit specific lines from the plot. This can be useful for removing domain boundaries (those that include a 99 code) in a pe-pH plot.
A ‘vec’ file is also produced by a contour plot. It has a different header line but again the file contains all the vectors needed to draw the contours and to assemble the polygons used for colouring. It contains information about the type of boundary, normally the fields (class 1 and class 2) differ by one but if they are the same, then this indicates a non-intersecting contour (with the domain boundary) or ‘island’.

With contour plots, the order of the vertices is always written with ‘the high side to the right’ when reading down the file.

5.4.8 ‘polygon’ file (pol)

This file is generated from the vectors (‘ht1’ and ‘contour’) or track (‘grid’) files and is used to colour-fill the polygons. It is also used to determine a default labelling position near the polygon centre in predominance plots. It is assembled from the vectors file (‘ht1’ and ‘contour’) by selecting the appropriate set of vectors for each field based on the ‘species’ involved and matching the ends until polygon closure has been achieved. In grid plots, ‘pixel aggregation’ is used to determine the polygon boundaries.

The header line includes the x- and y- resolution, the species code to which the polygon corresponds (see the labels file for the full species name in predominance plots), the number of points represented by each polygon segment, and the pe at each point. A species code of 99 stands for the domain boundary.

The pol keyword can be used to omit specified fields from a predominance plot.

Commenting out all the lines for a polygon can also be used to omit a particular polygon from the plot providing calculationMethod = 2 is used. In ‘ht1’ and ‘contour’ plots, the lines outlining each field are drawn using the vectors file but in a ‘grid’ plot these outlines are drawn using the polygon file.

If the value of sp (the species number) for all the points making up a particular polygon in the ‘pol’ file are less than or equal to zero, the field will not be drawn. It is necessary to edit the ‘pol’ file and replot (calculationMethod = 2) for this to work.

5.4.9 ‘labels’ file (labelFile)

This file is generated by predominance plots. It provides the dictionary to connect the species number in the vector and polygon files with a species name. It also includes the angle of the text. Editing this file provides the means of rotating a label.

This file is used to tell the plot where to centre the labels as well as providing a list of the species names. It also includes other attributes of the label such as rotation (theta in degrees from the horizontal measured clockwise). The pe is carried so that the labels can be placed properly if the y axis is changed to pe or a derivative of that. This file can be edited manually to reposition or rotate the labels. A labels file looks something like this:

<table>
<thead>
<tr>
<th>sp</th>
<th>x</th>
<th>y</th>
<th>pe</th>
<th>angle</th>
<th>area</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-8.300</td>
<td>-32.338</td>
<td>4.406</td>
<td>61.62</td>
<td>Pe(OH)3(a)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-11.541</td>
<td>-81.218</td>
<td>-11.054</td>
<td>0.41</td>
<td>Fe(OH)3-</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-4.789</td>
<td>-61.095</td>
<td>0.735</td>
<td>32.10</td>
<td>Fe+2</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>-2.371</td>
<td>-12.279</td>
<td>15.333</td>
<td>2.10</td>
<td>Fe+3</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>-2.854</td>
<td>-11.174</td>
<td>15.15</td>
<td>0.49</td>
<td>Fe2(OH)2+4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-10.189</td>
<td>-81.442</td>
<td>-9.75</td>
<td>0.25</td>
<td>FeOH+</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>-2.783</td>
<td>-22.243</td>
<td>12.454</td>
<td>0.00</td>
<td>FeOH+2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-6.990</td>
<td>-83.939</td>
<td>-7.177</td>
<td>2.49</td>
<td>H2(g) &gt; 1 atm</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>-7.000</td>
<td>-0.212</td>
<td>13.745</td>
<td>0.50</td>
<td>O2(g) &gt; 0.21 atm</td>
<td></td>
</tr>
</tbody>
</table>

Label size and colour are determined by the labelSize and labelColor keywords. Individual labels and the colouring of their associated polygons can be removed by commenting out the appropriate lines in the labels file. Alternatively, making the species number (sp) negative means the label will not be plotted. Setting the species number to zero will plot the label and field but will not colour it. In order to not plot the field or its label at all, use pol to exclude the polygon (see above) or edit the polygon file to give negative species numbers. The mini-
mumAreaForLabelling setting and editing of the area field in the labels file can be used to omit a label, e.g. make the area negative and set minimumAreaForLabelling to 0.1, say.

If labelColor is set to ‘nd’ or labelSize is set to 0.0, no labels will be plotted.

A species can have more than one label if it occupies more than one distinct field. The order of the labels is important and corresponds with the order that the polygons are read from the polygon file. This should not be changed.

Species names may be changed by editing this file and replotting with calculationMethod 2. Species names are assumed to be in Phreeqc formula format and numbers will be sub- and super-scripted accordingly. The species names can be appended with an identifier which will not be sub- or super-scripted. This should follow the species name and be preceded by an underscore. In order to allow surface species to still be interpreted correctly, the following rules apply: the identifier will not be subscripted if it is 3 or less characters long, or if at least one other underscore precedes it, e.g. the numbers in the identifiers for Cd+2_L or Cd+2_A2014 (two underscores) will not be subscripted whereas the charge in Hfo_Cd+2 will be super-scripted.

Possibly the easiest way of moving labels is to use a ‘nudge file’. This can be used for all plots except contour plots which have their own mechanism for moving labels.

5.4.10 Other output files

Various other files are produced, some of which are temporary files and normally deleted, others are left in the file system for perusal.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot.ps</td>
<td>this is always a copy of the last ps file produced (and the name of the tracking file optionally produced during calculations)</td>
</tr>
<tr>
<td>selected_1.0.out</td>
<td>this is the default name of the file containing the last SELECTED_OUTPUT returned by Phreeqc when \text{ABS(debug)} &gt; 0 (it is continually being overwritten with new data). Other file names can be set using the SELECTED_OUTPUT -file identifier in the CHEMISTRY section.</td>
</tr>
<tr>
<td>Phreeqc.out</td>
<td>this is the normal PRINT output from Phreeqc for the last iteration when \text{ABS(debug)} &gt; 0</td>
</tr>
<tr>
<td>*.all</td>
<td>this is an accumulation of the normal PRINT output from Phreeqc for all iterations and is only produced when \text{ABS(debug)} &gt; 1 or when the all keyword is set to TRUE. It contains end-of-file (Control-z) characters at the end of each iteration's output. This file can be very large which may prove problematic when opening with some editors.</td>
</tr>
</tbody>
</table>

Whether the latter two files are produced or not is controlled by the debug keyword and possibly the value of resolution.

If the two colour dictionary files are undefined and needed, then the files lineColor.dat and fillColor.dat will be automatically created in the working directory, i.e. the same directory as the main input file. Note that since the label positions are recorded in the line colour dictionary, it may be important that each problem run from the same directory is given a different dictionary name.

Some other temporary files may be produced during a run. These are normally deleted at the end of the run.
5.5 Inserting plot files into Microsoft Word, PowerPoint and other software

The information given below is likely to vary with the version of the software being used, and will in any case age quickly, so treat this as a starter and experiment yourself. Amazingly, the situation in 2017 has not improved much and you really will have to see what works for you. Unfortunately, the eps format seems a moving feast and generally does not transfer well. If you can afford Adobe Illustrator, then this is a good way of editing ps/eps files, and saving to a format which will be able to be imported into most documents. At the moment, probably the simplest approach to import a diagram into a document without resorting to other software is to export from PhreePlot in the png format, and then clip it.

Graphic files in various formats can in principle be imported into office software such as Microsoft Word and Open Office/LibreOffice. The filters available depend on the versions used. With Word 2002 and later versions of Word, Postscript (ps) and Encapsulated Postscript (eps and epsi) files can be imported with Insert|Picture|From file. You will have to change the ps and epsi files to have the eps extension for this to work or prepare an eps file directly (this will be clipped tightly to the image’s bounding box). They must be single page ps files of course. These files all have a preview when viewed in Word. png and jpg files can also be imported directly into most Office-type documents in the same way but they will not normally be of such good quality as the Postscript files since they are bit maps rather than vector-based and so will not scale so well.

The eps and epsi files will be cropped, the other formats will not be. These other format files can either be cropped with image editing software such as IrfanView, or the cropping done directly on the imported file in Word by specifying the crop parameters (Format|Picture).

If all else fails, the ps or epsi files can be converted to high resolution jpg files using GSview – you will have more control over the resolution this way than is available when these files are generated in PhreePlot.

Rather than importing the files into Word on a one-off basis, the ‘pictures’ can be linked to a particular file using the ‘Link to file’ option available on the Insert drop-down menu. This inserts an INCLUDEPICTURE field code into the document and reduces the document size since the actual code for the image is no longer included in the file. The file is automatically updated in the Word document when the file is reopened or when the fields are updated (Cntrl-F9). The link file name can be viewed by viewing the field codes (Alt-F9).

Using the ‘Insert and Link’ option when importing will insert the code for the picture and link it to the file for updating. This results in a larger document size than linking alone.

Probably the best format for importing into Word in terms of quality, size and convenience is the eps format but you will have to experiment.

Powerpoint is not able to render Postscript files. Probably the best format for inserting into Powerpoint is png. If all else fails use high resolution jpg. If the default resolution of the png file produced by PhreePlot (300 dpi) is not what is wanted, change the resolution using the second png parameter.

Many of the graphic formats exported by PhreePlot can also be imported into Open Office/LibreOffice documents. For example, Insert|Graphics|From File will import eps, jpg files. These files can also be either imported into the document or linked to it, as in Word. Cropping is also possible. png files are often the easiest to import and of reasonable quality.

Open Office/LibreOffice does not provide a preview for eps files and the output needs to be printed using a printer with a Postscript interpreter otherwise only the placeholder will be printed. If you do not have a Postscript printer available and do not already have a generic Postscript printer driver installed, install such a printer driver (e.g. in Windows 7, Control Panel | Hardware and Sound | Add a printer | Add a local printer | Use an existing port: FILE | Lexmark | C935 PS (MS)) (or any other good coloured Postscript printer driver). Then select this ‘printer’ when printing, and name the file as a ps file. This will pro-
duce a ps file that can then be read, converted and printed to paper with GSview etc.

If the graphic files need to be edited, Adobe Illustrator or Inkscape or other suitable vector-based software can be used. Inkscape is vector-based (based on the SVG format) and is versatile, modern and free. Many software suites (e.g. Microsoft Office and Open Office/LibreOffice) can now import pdf files. GIMP is free and powerful but is raster-based. There are also many ways of merging pdf files including use of Adobe Acrobat and Cloud-based methods (e.g. http://www.mergepdf.net/). Slideshows can also be prepared using Adobe Acrobat which means that pdf files can be used directly for presentation.

The vector-based quality of Postscript files should be retained for as long as possible and text should be kept as text rather than bit-mapped. Software that allows insertion of eps/pdf files directly will normally retain the highest quality.

5.6 SPEED OF COMPUTATIONS AND PLOTTING

The speed with which calculations and plotting take place clearly depends on the processing power available and the number of computations undertaken. It also depends on certain settings that are under user control. Normally most of the time is taken up within Phreeqc so particular attention should be taken to the Phreeqc setup.

The following tips might help speed up computations:

- reduce the number of species considered: the speed of Phreeqc computations depends strongly on the number of pure phases present. These are defined in the EQUILIBRIUM_PHASES keyword block. Reducing the number of phases being considered will reduce the computation time required. For example, phases that are obviously not going to feature in the calculations can be removed from consideration. The same is probably true to a lesser extent for the number of solution species. Reducing these will involve either changing the database used, or not including them as species in the EQUILIBRIUM_PHASES keyword block. With predominance diagrams, the ht1minerals.inc file (Section 8.1.6) can be used to monitor the saturation indices (SI’s) of all the possible minerals using the ‘carry variables’ approach. The summary statistics for these SI’s are written to the log file. If the maximum value is zero or very close to it, then this indicates that the mineral has precipitated somewhere in the calculations.

- reduce the resolution of the plot: this will reduce the number of calculations undertaken at the expense of the smoothness and maybe the accuracy of the plot. Once the plot is what is wanted, the resolution can be increased for a final, production plot.

- alter the KNOBS settings: these do not normally need to be adjusted but they can affect the speed with which Phreeqc converges (and even if it does converge) and so may be important in certain cases (see Section 6.5.5 and Section 8.12). Their impact may vary with the different versions of Phreeqc due to changes in the solution finding strategy.

The structure of the Phreeqc library used by PhreePlot has not be optimised for carrying out the types of calculations often done with PhreePlot. For example, successive calculations are often similar to each other with just a minor difference. However, Phreeqc starts from essentially the same starting point each time. Sometimes ‘setup’-type calculations can be taken ‘outside the loop’ by making use of the END keyword to separate setup code from the looping code.
6 Running PhreePlot

6.1 CONVENTIONS FOR DATA INPUT

6.1.1 Types of variables

Each keyword has an associated value or list of values associated with it. These can be of four types as given in Table 6.1

Table 6.1. Examples of how a Phreeqc column heading will be interpreted during plotting

<table>
<thead>
<tr>
<th>Input</th>
<th>Graphical output</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer (I)</td>
<td>Integer</td>
<td>-1, 0, 1, 123</td>
</tr>
<tr>
<td>Floating point number (F)</td>
<td>Non-integer numbers. Range allowed depends on storage type: usually stored as IEEE 'doubles' (=xxx to +*<strong>) but plotting parameters are stored as 'singles', *** to +</strong></td>
<td>-1.0, 1.23, 1e-5</td>
</tr>
<tr>
<td>Logical (L)</td>
<td>Logical value</td>
<td>TRUE, FALSE, true, false, t, f, T, F</td>
</tr>
<tr>
<td>Alphanumeric (A)</td>
<td>Strings consisting of the following characters: (0–9, a–z, A–Z, (space),!%'$%^&amp;*()_+[]:;&lt;&gt;?/ Enclose strings in single or double quotes if they include a space. If the string is enclosed in square brackets, these are removed and the rest of the string is taken literally, i.e. not interpreted. The following are not allowed: £$¬!</td>
<td>@</td>
</tr>
</tbody>
</table>

6.1.2 Phreeqc notation for chemical formulae

Phreeqc has a super/subscript-free way of specifying chemical formulae (Table 6.2) and this is used by PhreePlot to interpret formulae strings when labelling plots.

Table 6.2. Examples of various chemical formulae in Phreeqc format

<table>
<thead>
<tr>
<th>Conventional formula or name</th>
<th>Phreeqc format</th>
<th>PhreePlot</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>b</td>
<td>B</td>
</tr>
<tr>
<td>Br-</td>
<td>Br-</td>
<td>Br-</td>
</tr>
<tr>
<td>SO4^2-</td>
<td>SO4-2</td>
<td>SO4^2-</td>
</tr>
<tr>
<td>Ca^2+</td>
<td>Ca+2</td>
<td>Ca^2+</td>
</tr>
<tr>
<td>CaSO4.2H2O</td>
<td>CaSO4:2H2O</td>
<td>CaSO4.2H2O</td>
</tr>
<tr>
<td>FeS_{1.7}</td>
<td>FeS1.7</td>
<td>FeS_{1.7}</td>
</tr>
<tr>
<td>(CH3)2COOH</td>
<td>(CH3)2COOH</td>
<td>(CH3)2COOH</td>
</tr>
<tr>
<td>Hg0</td>
<td>Hg0</td>
<td>Hg0</td>
</tr>
<tr>
<td>Phenanthroline</td>
<td>Phenanthroline</td>
<td>Phenanthroline</td>
</tr>
<tr>
<td>HfO_s</td>
<td>Hfo_s</td>
<td>Hfo_s</td>
</tr>
<tr>
<td>Fe(3)</td>
<td>Fe(3)</td>
<td>Fe(3)</td>
</tr>
<tr>
<td>^18O</td>
<td>[180]</td>
<td>180</td>
</tr>
<tr>
<td>As(V)</td>
<td>[As5]</td>
<td>As5</td>
</tr>
</tbody>
</table>
PhreePlot assumes that some of the character strings (e.g. species and labels) used in PhreePlot may represent formulae in Phreeqc format and attempts to translate them accordingly. The places where this occurs are listed elsewhere (Section 10.1).

This interpretation of strings as formulae can be turned off by setting convertLabels to FALSE. This setting applies to all strings. A second, optional logical switch to convertLabels controls the conversion of colons when the Phreeqc conversion is true. This controls whether ZnCO3:H2O is converted to ZnCO3.H2O (TRUE) or to ZnCO3:H2O (FALSE).

6.2 PhreePlot Looping

6.2.1 Loop variables and their use

Looping, or iteration, is at the centre of PhreePlot. The other important feature is the separation of Phreeqc simulations into pre-loop and main loop simulations (see Section 4.6.1).

There are four loop variables available in PhreePlot and these operate in a nested fashion with the outer loop going round most slowly. These loops and their tag names are as follows:

Outer loop: main species loop (<mainspecies>)

z-loop (<loop>)

y-axis loop (<y_axis>)

Inner loop: x-axis loop (<x_axis>) (most rapidly changing)

The 'main species loop' loops on a list of character strings (each up to 30 characters), typically element names, but it can be any character string that requires substitution.

The y-axis loop and x-axis loops are set internally to loop over the specified y- and x-axis ranges, ymin to ymax and xmin to xmax, respectively, with the intervals of both controlled by the same resolution keyword. The resolution gives the number of Phreeqc iterations used. So if calculations are wanted over an x-range from 0 to 10 inclusive in steps of 1 then xmin = 0, xmax = 10 and resolution = 11. If xmax = xmin then no looping is undertaken. resolution = 1 signals some special behaviour. The y-variables are defined in exactly the same way. The pre-loop simulations are executed once and only the main loop simulations are looped.

The z-loop is controlled in several ways, most simply by the keywords loopMin, loopMax, loopInt and loopLogVar. These generate a regular sequence of numeric values for the <loop> variable. Alternatively the <loop> variable can be set to any discrete set of values, including an irregular series, using loopFile. The loop file approach also provides a mechanism for carrying varying values of other variables in parallel for each z-loop iteration including character variables. The z-loop iterates on both the pre-loop and main loop simulations.

Note that the x- and y-axis tag names use underscores not hyphens. The x- and y-axis loops are used in predominance plots where movement along both axes is required. The resolution in both dimensions is always the same in these types of plot. This arrangement is rarely useful in custom plots. Here it is more common to use the x-axis loop to drive the principal independent variable, like pH, and the z-loop to alter some other factor or 'level' in a systematic way.

The main species loop produces a new plot for each species. These may or may not be in the same file depending on the multipageFile setting.

The z-loop either produces a new plot for each value or introduces a blank line in the 'out' file every time its value changes. This gives a break in the plotted line at each of these changes. Whether or not a new file is produced or a blank line is introduced depends on the type of plot and the customLoopManyPlots setting. A new plot file will always be produced for species plots.

In plots other than predominance plots, the y-loop should normally be left undefined and unused.

In summary, by default, a blank line is inserted in the output files for each iteration of the
<loop> variable. This results in a break in the plotted curve. This does not happen with the <x_axis> or <y_axis> variables. The <x_axis> and <y_axis> variables are primarily designed for continuous variables whereas the <loop> variable is primarily designed to loop over discrete values of a variable. This difference is not rigid but there are some differences in the output that reflect this motivation.

For example, in predominance plots, the z-loop can be used to loop over a discrete range of concentrations of the main species. The z-loop variable can be specified with the loopMin, loopMax, loopInt and loopLogVar keywords. The loopLogVar keyword is used to signal whether to loop over a linear (0) or log (1) range of z-values.

z-looping can be used to repeat Phreeqc calculations in which just one or two parameters are changed between runs, such as the pH. The custom plot option is used for this.

These built-in looping mechanisms always generate a regular sequence of values. If an irregular sequence of loop values is needed, or if more than one loop variable needs to be changed on each iteration, use loopFile with calculationType = 'custom' or a data file with calculationType = 'simulate'. If a non-blank loop file name is set, the values in this file take precedence over the loopMin etc keyword settings. An important difference between these two approaches is that the loopfile approach will repeat the pre-loop iterations whereas the simulate approach does not.

Phreeqc also has its own internal looping mechanisms, for example with the REACTION and KINETICS blocks, and also to some extent with SOLUTION_SPREAD.

An example of the simulate approach is given below and in the calculation of saturation indices for a batch of water samples (see the SIs example in the \demo\SIs directory).

6.2.2 Looping over a list of character variables

The 3-parameter z-looping mechanism discussed above is designed to loop over a range of numeric values. If you want to loop over a list of character variables, either use a loopFile with character columns or in simple one-variable cases, use the mainspecies loop (even if the variables involved are not actually 'species').

6.2.3 An example of the use of various looping mechanisms

A series of examples described below shows how the various looping mechanisms can be used to calculate the solubility of iron oxide as a function of pH in a fluoride-rich medium. These examples highlight the different setups that can be used to implement looping in PhreePlot. These examples can all be found in the demo\FeSolubility directory.

Using the <x_axis> tag

The first example uses the <x_axis> variable. The input file (FeSolubilityXaxis.ppi) is:

```plaintext
SPECIATION
  calculationType                      "custom"
  calculationMethod                    1
  xmin                                 2
  xmax                                 12
  resolution                           101
  numericTags                          <log_H> = -<x_axis>
PLOT
  plotTitle                            "Fe(OH)3(a) solubility vs pH"
  xtitle                               "pH"
  ytitle                               "log<sub>10</sub> Fe<sub>T</sub> (mol/kgw)"
  pymax                                0
  customXcolumn                        1
  lineColor                            blue
  useLineColorDictionary               0
  legendTextSize                       0
  label                                0
CHEMISTRY
PHASES
  Fix_H+; H+ = H+; log_k 0
```
The range of the \texttt{x_axis} variable is given by \texttt{xmin} and \texttt{xmax}. The span in pH is 10 units and the resolution is 101 so calculations will be made at \texttt{x_axis} values of 2.0, 2.1, 2.2 ...12.0. The \texttt{y_axis} variable has been left undefined (here and in the other input files) and so is unused.

The \texttt{log\_H} tag is defined as the negative of the \texttt{x_axis} variable so that it can be substituted on the \texttt{Fix\_H} line. This is the only tag used within the CHEMISTRY section.

The plot produced from this file is shown in Figure 6.1. Exactly the same plot can be produced by using the \texttt{y_axis} variable in place of \texttt{x_axis}.

The useLineColorDictionary setting of 0 means that the line colour dictionary is ignored as a source of line and point colours or label coordinates, even if they are present in the dictionary. Rather auto colouring is used starting with the colours given in the lineColor setting (here set as blue). The colour dictionary is always created if absent or updated if present with the latest colours and coordinates.

Figure 6.1. Amorphous iron oxide solubility as a function of pH calculated using the \texttt{x_axis} loop variable.
Using the `<loop>` tag

Exactly the same plot can be produced using the `<loop>` variable. This requires several changes to the SPECIATION section but none to the sections following that. The modified input is shown below:

```plaintext
SPECIATION
  calculationType                      "custom"
  calculationMethod                    1
  loopMin                              2
  loopMax                              12
  loopInt                              0.1
  numericTags                          <log_H> = -<loop>
  dataSeparators                       "{"  \\
                                      "\t"  \\
                                      "\p"  "泸"  "泸"  "泸"
```

The two main differences are that `loopMin`, `loopMax` and `loopInt` replace `xmin`, `xmax` and `resolution`. The fourth data separator must also be redefined to remove the blank line that would normally be placed after each iteration of the `<loop>` variable. It is set to the null separator which means that no new line (paragraph) is put between each iteration. Since blank lines in the 'out' file are interpreted as line breaks when custom plotting, without this change to the data separator, 101 separate lines would be plotted rather than a single curve.

This is why it is normally preferable to use the `<x_axis>` approach for continuous variables and to leave the `<loop>` variable for defining discrete intervals or levels of a variable.

Using a loop file

It is also possible to read the 101 values from a loop file. The name of a loop file needs to be given and the `dataSeparators` may need to be redefined. The input is:

```plaintext
SPECIATION
  calculationType                      "custom"
  calculationMethod                    1
  loopFile                             FeLoop.txt
  numericTags                          <log_H> = -<loop>
  dataSeparators                       "}"  \\
                                      "\t"  \\
                                      "{"  "泸"  "泸"  "泸"  "泸"  "泸"  "泸"
```

The `FeLoop.txt` file looks like this:

```
#pH
2.0
2.1
2.2
2.3
...
12.0
```

It has 101 rows of data with a comment at the top which is ignored. This gives the same plot as in Figure 6.1.

Using the 'simulate' calculationMethod

This method also reads the data from a file, this time the file specified by the `dataFile` keyword. The relevant part of the input file looks like this:

```plaintext
SPECIATION
  calculationType                      "simulate"
  calculationMethod                    1
  dataFile                             FeSimulate.txt
  logVariableIn                        0
  dependentVariableColumnCalc          2
  numericTags                          <log_H> = -<pHIn>
  customXColumn                        3
```

The `FeSimulate.txt` file looks like this:
Note that as for the loop file, this file has a header row which define a series of tags, one per column. While the header row is optional for the loop file, it is compulsory for this data file. Here the `<pHin>` tag is defined which is used in the definition of `<log_H>`. The logVariableIn keyword (a list of 0, 1 or -1’s) must also be included to indicate the number of columns in the data file and whether any data transformations are required. Here, a single 0 indicates one column without any transformation on input.

It is also necessary to define where the dependent variable (the calculated value) is to be found in the main selected output file - this is done with the dependentVariableColumnCalc keyword. The customXcolumn also has to be set. This is the only plot type that uses the ‘pts’ file rather than the ‘out’ file, for plotting and this file has a slightly different format from the ‘out’ file consisting of the line number from the input file, calculated values and all of the variables read in from the data file - whether they were used or not.

**Looping over two variables**

It is also possible to plot several curves on the same plot by using two looping variables. a line break is normally The following file (FeSolubilityXaxisLoop.ppi) does this:

```plaintext
SPECIFICATION
  calculationType                      "custom"
  calculationMethod                    1
  xmin                                 2
  xmax                                 12
  loopmin                              -4
  loopmax                              -1
  loopint                              1
  looplogvar                           1
  resolution                           101
  debug                                0
  numericTags         <log_H> = -<x_axis>

PLOT
  plotTitle                            "Fe(OH)3(a) solubility vs pH"
  xtitle                               "pH"
  ytitle                               "log<sub>10</sub> Fe<sub>T</sub> (mol/kgw)"
  labels                               "10<sup>-4</sup>M F" "10<sup>-3</sup>M F" "10<sup>-2</sup>M F" "10<sup>-1</sup>M F"
  pymax                                0
  customXColumn                        1
  linecolor                            blue
  useLineColorDictionary               0
  legendTextSize                       2
  label                                1.5

CHEMISTRY

PHASES
  Fix_H+; H+ = H+; log_k 0

SELECTED_OUTPUT
  reset false

USER_PUNCH
  headings pH FeT
  10 PUNCH -la("H+*", log10(TOT("Fe")))

SOLUTION 1
  pH        1.8
  units     mol/kgw
  Fe(3)     1e-1
```
This is similar to the other files but has the `<x_axis>` tag to drive the x axis (pH) and the `<loop>` tag to drive the variable Na and F concentrations. The `<loop>` or z loop adds a blank line at the end of each iteration. These are interpreted as line breaks by the custom plotting routines which then plots them separately with different line colours (Figure 6.2). The labels keyword provides a list of names to use for labelling the curves and producing the legend.

If a separate plot is wanted for each value of the z-loop variable in a multiple-z value custom plot, set `customLoopManyPlots` to `TRUE`.

### 6.2.4 Looping in multi-simulation input files - pre-loop simulations and the main loop

**Introduction**

Decisions had to be made about how PhreePlot would deal with multi-simulation input files, i.e. those with more than one `END` keyword (assuming the file finishes with an `END`). In principle these files can be of any degree of complexity and can include completely unrelated simulations.

Two aspects had to be considered.

Firstly, which simulations are repeated and which are not. There is obviously a time and clutter penalty in repeating invariant operations that do not have to be repeated such as reading in a database. This led to the concept of pre-loop (once only) simulations and main loop (repeated) simulations.

Further, when several simulations need to be run, should they be fed to Phreeqc one at a time...
or all together. **PhreePlot** can only update tags and make substitutions to the input file when control is returned to itself between runs.

Secondly, which results of the various simulations are picked up and sent to the various **PhreePlot** files ready for fitting, plotting etc. Some simulations produce no useful output while others produce one or more lines of useful output.

The format of the output produced by a given block of code can depend on whether the simulations are executed one-at-a-time or in a single run.

Controlling these aspects of the input and output is the key to running **PhreePlot** successfully.

**Basic structure of a multi-simulation Phreeqc input file**

Each **Phreeqc** simulation in an input file is numbered consecutively from the top down, 1, 2, 3, ... A multi-simulation input file can be viewed as a series of one or more contiguous simulations or 'blocks' of simulations. A block of simulations is therefore defined by a range of simulations.

In many cases, **PhreePlot** considers the input file to be a single block but it can also be treated as a series of unrelated blocks. This is possible with the 'fit' and 'simulate' types of calculations where each line of data (each 'observation') in the associated data file can point to a different block of **Phreeqc** code.

Each block is itself treated as having two parts: (i) the top part consists of zero or more 'pre-loop' simulations, and (ii) the lower part consists of one or more 'main loop' simulations.

The innermost two **PhreePlot** loops (the y- and x-axis loops) act only on the main loop simulations. This division between the two is designated by the `mainLoop` setting which is the number of the simulation in the block starting counting from the top of the block, i.e. the starting point of the looping.

The default is 'last' which automatically sets `mainLoop` to the number of the last simulation in the block. Setting `mainLoop` to 1 would mean that all the simulations in that block are treated as main loop simulations.

Much of this behaviour is hidden when running **PhreePlot** in normal mode but setting `debug` to 1, 2 or 3 will cause more or less of the calculation trail to be written explicitly to the log file.

The following decisions were made based on the `calculationType`.

For custom, species and predominance-type calculations, it is assumed that the **Phreeqc** input file is written to perform a single set of calculations and consists of zero or more initial simulations to set up the database, do initial solution calculations, and to define other static settings. This is then followed by one or more simulations which will be subject to the y- and x-axis **PhreePlot** looping mechanisms. Any simulation which contains a tag which is expected to change on each iteration must be in this latter block. This division into 'pre-loop' and 'main loop' simulations is specified with the `mainLoop` keyword which specifies the number of the **Phreeqc** simulation at which the main loop starts.

In a multi-simulation input file, **PhreePlot** looping only applies to the simulations numbered from `mainLoop` onwards. The simulations before this are assumed to be 'pre-loop' simulations. Each of these pre-loop simulations is run individually and the tag dictionary updated between runs (Figure 6.3). This means that tag variables can be passed from one simulation to the next in these pre-loop simulations.

The 'main loop' iterates as rapidly as possible over the y-axis and x-axis loops. This loop runs with minimum overheads and is intended for the most repetitive calculations. All simulations in this main loop are run as a single **Phreeqc** run so there is no possibility of using tags created in one simulation in the next. The tags are updated just once then the whole block of simulations is executed.

The earlier ('pre-loop') simulations are re-run for each value of the main species and z-loop.

The default value for `mainLoop` is 'auto' which sets `mainLoop` to the number of simulations
Running PhreePlot 59

Care needs to be taken with the selected output when looping over more than one simulation. A USER_PUNCH data block entered in simulation 1 say will remain active in subsequent simulations unless specifically turned off with the -selected_output identifier in the PRINT data block. If the USER_PUNCH data block is redefined in later simulations then the output may become confusing since a new header line is not written to the 'out' file.

With multi-simulation input files, selectedOutputLines controls the amount of selected output actually sent to the 'out' file. The number specified refers to the number of lines sent from the SELECTED_OUTPUT in the chosen simulation (the SELECTED_OUTPUT block with the highest user number).

The 'out' file is often used directly for plotting and the challenge is to end up with a well-formed file suitable for plotting. Normally this should be a rectangular table with a header row and columns of data possibly with blank rows to signify breaks in the data.

The exception to the above behaviour is during fitting and simulations. The 'fit' and 'simulate' calculationTypes do not allow looping since they use multi-simulation files to allow for the possibility of doing different calculations for each data point (observation) or set of data points. Each data point can in principle choose its own range of simulations and its own mainLoop parameter. This makes it possible to optimise over a number of different types of calculations within the same fit (‘global optimization’). The simulation or range of simulations to be used for each point is specified in a special column in the fit data file, the mainLoopColumn. Weighting of the residuals becomes a particularly important consideration when mixing ‘apples and oranges’ in this way.

![Figure 6.3](image)

**Figure 6.3.** Two examples to illustrate how the mainLoop defines the division between 'pre-loop' simulations and the 'main loop' simulations in which the x- and y-axis variables operate. In this example, the difference controls the number of times simulation 3 is run and consequently when selected output is normally written to the 'out' file and the tags updated. The default (‘last’) is for the mainLoop to only loop over the last simulation, as illustrated on the right. The 'pre-loop' simulations are each run separately and sequentially. In contrast, the 'main loop' simulations are run as a single block with no updating of tags between simulations.
6.2.5 Dynamic switching between Phreeqc models (code)

It is sometimes necessary during looping to switch between different pieces of Phreeqc code to obtain the required output. This switch may reflect the results of an earlier simulation or the value of some tag variable.

The most flexible approach is to engineer for a tag to be the ‘switch’ tag and to then use this in the USER_PUNCH code to calculate what is required and to send the required output to the selected output. This can often be made possible using the ‘one simulation at a time’ mode (see mainLoop) based on the fact that in this mode USER_PUNCH’ed output from intermediate simulations is automatically turned into tags which can then be used in subsequent simulations. Furthermore, this intermediate output does not get copied to the ‘out’ file and so will not interfere with the output used for plotting or fitting. So a simulation can calculate something, punch a switch value to the selected output and then the next simulation can make the switch accordingly, perhaps even repeating the previous simulation with slightly different input or output.

6.2.6 Defining the expected output in the selected output file

Use selectedOutputLines to define the number of lines (rows) to be read from the output defined by the targeted SELECTED_OUTPUT (n)/USER_PUNCH (n) block, counting lines from the bottom upwards. Set to 1 to pick just the last line, 2 to pick the last two line etc., or ‘auto’ to pick all lines. During fitting, the number of lines picked will automatically be set to 1 when onePass is FALSE and to that given by the number of observations in the fit data file when onePass is TRUE.

If the selectedOutputLines value is 0, no data transfer will take place. This option can be used when PhreePlot is used to produce some other output file such as a print file.

In order to accumulate output from all the simulations in one input file into a single selected output file (and ‘out’ file), set mainLoop to 1 and put the SELECTED_OUTPUT (n)/USER_PUNCH (n) block in the first simulation. This ensures that all the simulations will be run in a single run and so forces the accumulation of all output data into a single output file. Don’t use any other looping mechanism, i.e. just loop through the whole set of simulations once. However, note that tags will only get updated and substituted once at the beginning of the run.

It may be necessary to define a ‘break variable’ to force the plotted curves to break between simulations. This can be done with the sixth parameter of the dataSeparators keyword. Whenever the break variable is encountered, a blank line is sent to the outfile.

6.2.7 Timing execution

It can be useful to see the difference between execution times for different iterations. This can be done by making use of the <systime> system tag. This is initialised to zero at the beginning of a PhreePlot run and returns the cumulative time in seconds for each iteration thereafter. It can be accessed using this tag in the USER_PUNCH section. The time taken for each iteration can be calculated by using the PUT/GET mechanism to save the time of the previous iteration and subtracting it.

For example, in a custom plot add code something like

```plaintext
- headings nexecute time dt
  100 IF (EXISTS(1)=0) THEN PUT(0,1)
  200 dt = <systime> - GET(1)
  300 PUNCH <nexecute>, <systime>, dt
  400 PUT(<systime>,1)

and monitor in the ‘out’ file. For a predominance plot, add something like the following as ‘carry’ variables in the htl.inc (or similar) file.

```plaintext
295 IF (EXISTS(1)=0) THEN PUT(0,1)
296 dt = <systime> - GET(1)
```
297 PUNCH "nexecute", <nexecute>, "time", <systime>, "dt", dt
298 PUT(<systime>,1)
299 nout4 = 3

and monitor in the 'track' file (trk). If a plot of this timing is wanted, set up another ppi file
something like

PLOT
extradat                             hfo.trk
customxColumn                        nexecute
lines                                dt
labelsize                            0
legendtextsize                       0
xaxislength                          150

and execute the two together with a batch file.

6.2.8 Speeding-up calculations

Sometimes calculations are noticeably slow because the initial solution calculation is a poor
estimate of the final solution. In these cases, it may be possible to speed-up calculations signif-
icantly by saving the result of the last calculation and using it as a starting point for the next
calculation. This makes use of one or more of the '_MODIFY' Phreeqc keywords.

A typical scenario for preparing a predominance diagram would be:

# simulation 1 - preloop, once only
SOLUTION 1
....
EQUILIBRIUM_PHASES 1
Fix_H+     -7 NaOH
   -force_equality true
O2(g)      -45
   -force_equality true
....
SAVE SOLUTION 2
SAVE EQUILIBRIUM_PHASES 2
END

# simulation 2 - main loop, iterate here
USE solution 2
USE equilibrium_phases 2
USE surface 2
EQUILIBRIUM_PHASES_MODIFY 2
   -component Fix_H+
   -si -<x_axis>
   -component O2(g)
   -si <y_axis>
SAVE solution 2
SAVE equilibrium_phases 2
SAVE surface 2
END

If other reactions such as surface reactions are involved, then these must also be SAVE'd and
USE'd in the same way. This approach makes most sense for grid-type calculations but can also
be used to speed-up 'hunt-and-track' style calculations. It makes little or no difference when
the chemistry is 'simple' and calculations are relatively fast, e.g. demo\Fe\hfo.ppi.

An example to demonstrate this approach, FeAsS(cd-music)-mod.ppi, is given in the
C:\PhreePlot\demo\FeAsS-cd-music\ folder.

A similar approach using RUN_CELLS is also possible.

6.3 POSSIBLE TYPES OF CALCULATIONS AND PLOTS

The types of calculations undertaken and the type of plot produced is controlled by the calculationType keyword (Table 6.3).
The other critical keyword that should be included in each input file is that of calculation-Method which has the following values:

- 0 = do calculations but do not plot anything
- 1 = do calculations and plot everything from scratch
- 2 = don’t re-speciate or reprocess data but replot using existing plot data files
- 3 = as for 2 but goes back one stage further and reprocesses the output from speciation.

### 6.4 PREPARING THE SELECTED OUTPUT FILE

#### 6.4.1 Normal behaviour

PhreePlot receives output from Phreeqc via its selected output ‘file’ mechanism. Originally Phreeqc had just one combination of SELECTED_OUTPUT and USER_PUNCH keyword blocks but this has now been extended to any number of SELECTED_OUTPUT n and USER_PUNCH n blocks where the integer n defines a specific combination. The main (default) selected output is n = 1, and it is this file that provides the data that are extracted to the ‘out’ file.

Although the selected output file was originally a permanent file that could be accessed via the operating system in the usual ways, one of the options with the Phreeqc library used in PhreePlot is to make this transfer entirely in memory. Storing the results in memory reduces I/O and speeds up execution. However, it does not leave a file to be inspected after execution has finished. PhreePlot uses this memory approach when debug=0 but uses the permanent file approach when \( \text{ABS(debug)}>0 \).

It is also often nearer to turn off the default selected output using the -reset false option in the SELECTED_OUTPUT keyword block. This minimises the transfer of unwanted data. If this option is not used, default selected output variables, like ‘state’, ‘simulation’ etc., will be included in the output files.

The selected output ‘file’ (physical or actual) is constantly being overwritten with the results of the last iteration. The ‘out’ file (see Section 5.4.4) accumulates the relevant lines (see selected-OutputLines) that are produced each time the selected output is ‘punched’ (once per iteration of the USER_PUNCH block providing that it is not blank and that the output has not been turned off with PRINT; -selected_output FALSE or SELECTED_OUTPUT 1; -active FALSE).

#### 6.4.2 The use of the ‘headings’ identifier

The -headings line in the USER_PUNCH block defines the column labels that are output to the first line of the selected output file. These column labels in turn are used to define the tag names for the variables that are automatically produced from the selected output.

Because of the way that headers are used to define variable names in PhreePlot and because of
the limitations of the **PhreePlot** function parser, the choice of header names is somewhat more restrictive in **PhreePlot** than in **Phreeqc**. In particular, the following special characters `+-/*()<>^\` can cause problems. Quotes are not treated specially. Therefore quoted strings should not be used for headings and spaces are not allowed within an individual heading name – spaces are used as separators.

If one or more of the special characters is found, **PhreePlot** will replace each of them with a period (\`). This means that “\`a+b\`” and “\`a-b\`” will both be translated to “\`a.b\`”, leading to an error if both are present. Providing that such degeneracy is avoided, the use of special characters in headers should cause no problem. The reporting of tag values in the log file and the use of column headers in defining plot variables is based on the original names.

The list of column headings are associated in turn with each of the punched variables:

```plaintext
SELECTED_OUTPUT
  -high_precision       true
  -reset                false
USER_PUNCH
  -headings  pH Zn Cd ZnOam
  -start
  10 sorbedZn=SURF("Zn","Hfo")
  20 totZn=SYS("Zn")
  30 sorbed1=100*sorbedZn/totZn
  40 mineral=100*equi("ZnO(a)")/totZn
  50 punch -la("H\+") sorbed1
  60 sorbedCd=SURF("Cd","Hfo")
  70 totCd=tot("Zn")*tot("water")+sorbedCd
  80 sorbed2=100*sorbedCd/totCd
  99 punch sorbed2 mineral
```

- **pH** will head the column containing values of `-la("H\+")`.
- **Zn** will head the column containing values of `sorbed1`.
- **Cd** will head the column containing values of `sorbed2`.
- **ZnOam** will head the column containing the percentage of Zn in the mineral ZnO(a).

The column headings are used by **PhreePlot** in four specific ways:

(i) to generate tag names like `<pH>`, `<Zn>` etc
(ii) to label the curves plotted in the main plot area (suppress with `labelSize<=0`)
(iii) to label the legend to the plot (suppress with `legendTextSize=0`)
(iv) to control whether the data column is to be plotted as points, lines, both or not at all.

If `convertLabels` is `TRUE`, then column headings are checked by **PhreePlot** to see if they are consistent with **Phreeqc** chemical formulae format and if so are interpreted accordingly when used as in-plot labels for the curves and in the legend. This means that numbers within the text string will normally be interpreted as a stoichiometry and subscripted. A check is made to see if the numbers could be interpreted as valences rather than stoichiometries. This checking is not particularly thorough and some strings may be interpreted as formula when in fact they are not.

If certain characters are found in the heading which indicate that the string is not a formula then this prevents the translation to **Phreeqc** formula format. These characters are: `¬~@\[\{\$%&\^\_\`\-
. Some of these (`¬\£\€\$\%\&\^\_\`\-
`\) are non-plotting characters with ASCII encoding and so they can be used to force interpretation as a non-**Phreeqc** formula on a one-off basis without affecting the appearance in the plot, e.g. `¬Zn8` will plot as Zn8 rather than as Zn8. However, these non-plotting characters should be used with caution as their behaviour is non-standard and may be unpredictable. They are plotted using the Latin-1 character set.

Starting a heading with a lowercase character will force it to **not** be interpreted as a formula.

The backslash has special behaviour in a heading since on printing **Phreeqc** strips out the \

and the following character from the name. It is therefore necessary to use `\` if a backslash is wanted. In **PhreePlot**, a backslash is also used to indicate that the next character is to be interpreted as a Greek character so `\b` in a heading would print the Greek character beta. Some examples which demonstrate these rules are given in Table 6.4.

### Table 6.4. Examples of how a **Phreeqc** column heading (label name) will be interpreted during plotting

<table>
<thead>
<tr>
<th>Input</th>
<th>Graphical output</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH4</td>
<td>CH₄</td>
<td>normal Phreeqc formula (first character is uppercase)</td>
</tr>
<tr>
<td>cH4</td>
<td>cH₄</td>
<td>not a formula (first character is not uppercase)</td>
</tr>
<tr>
<td>C(-4)H₄</td>
<td>C(-4)H₄</td>
<td>Phreeqc formula but (-4) is a valence not a charge</td>
</tr>
<tr>
<td>C(-4)</td>
<td>C(-4)</td>
<td>Phreeqc formula as above</td>
</tr>
<tr>
<td>C-4</td>
<td>C⁻</td>
<td>Phreeqc formula but -4 is treated as a charge</td>
</tr>
<tr>
<td>¬CH4</td>
<td>CH₄</td>
<td>Non-printing character at beginning prevents interpretation as a Phreeqc formula (ASCII encoding)</td>
</tr>
<tr>
<td>\C(-4)</td>
<td>(-4)</td>
<td>Phreeqc removes backslash and first character after it and the remaining is interpreted literally (it is not a formula)</td>
</tr>
<tr>
<td>\C(-4)</td>
<td>C(-4)</td>
<td>the first \ removes the second \ and the remainder is interpreted as in the first example above</td>
</tr>
<tr>
<td>\C(-4)</td>
<td>X(−4)</td>
<td>the first two \ are removed leaving a single \ which indicates that the next character should be treated as a Greek character (here chi). This indicates that it is not a formula.</td>
</tr>
</tbody>
</table>

Remember that `#` and `;` have special meanings in **Phreeqc** input files and should be avoided in headings. Single and double quotes may also be interpreted in a special way in strings including headings. Tags such as `<sub>...</sub>` will always be interpreted as indicated and so can be used to force certain behaviour, e.g. `%Zn<sup>2+</sup>` will appear as `%Zn²⁺`.

Unpaired quotes should not be used in column names. Strings enclosed in square brackets will be stripped of the brackets and then interpreted literally.

### Controlling the plotting of individual columns

Data from the selected output are accumulated in one or more output files. These files may then be used for plotting the data.

The **points** and **lines** keywords (and their 2y counterparts) control whether points and/or lines are selected for plotting. Properties of these points and lines such as colour, size or width are controlled by a series of keyword lists such as **lineWidth**, **pointSize** (and **lineWidth2y** and **pointSize2y**). Each dataset selected to be plotted with points and lines has a corresponding entry in a property list, or if the list is short, is either generated automatically (**pointColor** or **lineColor**) or recycled from the existing list.

For example, if **pointSize** is set to 0.0 or the **pointColor** is set to ‘nd’ then no symbol will be drawn for any of the selected point datasets. If **pointSize** is set to 5.0 3.0 and **pointColor** is set to **red** **blue** then the symbols for the selected datasets will alternate red-blue-red-blue ... as needed with sizes alternating 5, 3, 5, 3 ....

The six standard filled symbols can have a coloured rim with their colours and widths set by the corresponding **rimColor** and **rimFactor** keyword lists.

### Use of labels in a custom plot and the minimum text size

The labels are used for labelling the lines/points in a custom plot and its legend. The size of the label text is given by **labelSize**. The minimum height of text is 0.01 inch and is reset to this value if it is entered as a smaller positive value than this. 0.0 suppresses plotting of text altogether.

The headings can contain super- or subscript tags, e.g. `H<sub>2</sub>` or `Fe<sup>2+</sup>`,
and Greek characters as per the normal rule (only to the first level – no superscripted Greek characters are allowed).

Label headings must not be duplicated.

6.5 DEBUGGING

6.5.1 Types of problem

If a file has not run as expected or has crashed then some debugging is required. Providing that the run has not returned an immediate error report giving the file:line:word location of a syntax error, the input file must be syntactically correct and the problem must lie deeper. An audible low beep signifies a recognised calculation problem of some sort.

First check the screen and log files if present to see if there are any messages which might give a clue to the problem. Then it is necessary to work out whether it is Phreeqc/chemistry problem or a Phreeplot problem.

If the problem has failed on its first iteration of Phreeqc, there is probably something wrong with the input file, either the PhreePlot part or the Phreeqc part.

It may be helpful to get the Phreeqc code working first by pasting the chemistry code into a standalone version of Phreeqc, Phreeqci or Phreeqc for Windows. It will be necessary to substitute values for all the tags before running the code.

PhreePlot problems should be reported to the address given on www.phreeplot.com.

The flags for ancillary output files such as the track file in the case of a predominance diagram should be set to TRUE. Also make sure that all Phreeqc output is sent to the various log files by ensuring that

\[
\text{PRINT; -reset TRUE}
\]

has been set in the Phreeqc input (CHEMISTRY) section. If there are several PRINT data blocks in the CHEMISTRY section, it is the value of the last one that is used. This is useful since several of the include files use -reset false which means none of the normal Phreeqc output will be printed. Therefore during debugging, providing the line above is included after the include file, reset will be set to TRUE and the Phreeqc output will be sent to the Phreeqc output files without needing to edit the include file.

One this has been set, you can use debug = 2 or debug = 3 to get a listing of all the Phreeqc output copied to the file *.all (see below). Examine this carefully making sure that all the expected substitutions have been made properly. The values of all the tags just before executing Phreeqc can be found in the log file so it should be able to tell if it is a PhreePlot problem or Phreeqc problem.

You may just need a few iterations run. You can interrupt a run by pressing Esc and entering ‘s’ for Stop.

When the run is crashing later on and the early runs look good, it is more likely to be a failure to converge in Phreeqc. This is most likely to be due to a user error in setting up the Phreeqc input file leading to extreme chemistry of some kind. Failure of Phreeqc on well-defined and well setup problems is rare enough to be dismissed as the most likely explanation. However, failure of Phreeqc to converge can be quite a common problem and is usually due to the calculations straying into some region of ‘unrealistic’ territory. Occasionally the chemistry may be too complex and Phreeqc struggles to find a solution – this can be tested by simplifying the problem including reducing the number of PHASES.

The problem is usually solved by tweaking the convergence parameters. The FeAsS.ppi example is an example in which Phreeqc fails to converge in one place with the default convergence tolerance. Relaxing the criterion from 1e-12 to 1e-10 solves the problem.
6.5.2 Checking the return status of a Phreeqc run

It is possible to check the status of a Phreeqc run by outputting the status return using the \texttt{<phreeqc\_status\_0>} tag in a \texttt{USER\_PRINT} or \texttt{USER\_PUNCH} block. This system tag is automatically created and updated after each run. The 0 reflects the thread number of the run. A status return of zero indicates successful completion while a positive value gives the number of errors detected.

In principle, it is possible to check the status of a run, take avoiding action if it has failed and then rerun the simulation, all in one script. For example, it is possible to automatically switch the chemical used to achieve the target value in an \texttt{EQUILIBRIUM\_PHASES} block, see the \texttt{\demo\switch} examples.

6.5.3 General approach to debugging

Debugging is an acquired skill and some general principles apply whatever the language. David Agans has nine general rules for debugging. In a review of Agans’ book, David A. Wheeler listed them as:

- \textit{Understand the system}: Read the manual, read everything in depth, know the fundamentals, know the road map, understand your tools, and look up the details.
- \textit{Make it fail}: Do it again, start at the beginning, stimulate the failure, don’t simulate the failure, find the uncontrolled condition that makes it intermittent, record everything and find the signature of intermittent bugs, don’t trust statistics too much, know that “that” can happen, and never throw away a debugging tool.
- \textit{Quit thinking and look} (get data first, don’t just do complicated repairs based on guessing): See the failure, see the details, build instrumentation in, add instrumentation on, don’t be afraid to dive in, watch out for Heisenberg, and guess only to focus the search.
- \textit{Divide and conquer}: Narrow the search with successive approximation, get the range, determine which side of the bug you’re on, use easy-to-spot test patterns, start with the bad, fix the bugs you know about, and fix the noise first.
- \textit{Change one thing at a time}: Isolate the key factor, grab the brass bar with both hands (understand what’s wrong before fixing), change one test at a time, compare it with a good one, and determine what you changed since the last time it worked.
- \textit{Keep an audit trail}: Write down what you did in what order and what happened as a result, understand that any detail could be the important one, correlate events, understand that audit trails for design are also good for testing, and write it down!
- \textit{Check the plug}: Question your assumptions, start at the beginning, and test the tool.
- \textit{Get a fresh view}: Ask for fresh insights (just explaining the problem to a mannequin may help!), tap expertise, listen to the voice of experience, know that help is all around you, don’t be proud, report symptoms (not theories), and realize that you don’t have to be sure.
- \textit{If you didn’t fix it, it ain’t fixed}: Check that it’s really fixed, check that it’s really your fix that fixed it, know that it never just goes away by itself, fix the cause, and fix the process.

The ‘divide and conquer’ rule applies well for solving PhreePlot-type problems. Simplify the failing example until it works and then work forward, narrowing the range of possibilities until the source of the error is found.

Sometimes it is the data that is giving the problem and the ‘divide and conquer’ approach can work well for that too. Split the data into two and see if both halves cause a failure. If only one half does, keep dividing the failing half into two until the data giving the problem can be iden-
tified. Then work out what is special about those data. **Phreeqc** rarely fails from programming errors or bugs but calculating predominance plots can explore a large range of chemistries in terms of redox, acid-base, mineral stability, adsorption etc. and only the best speciation programs are robust enough to complete such a challenging set of calculations reliably.

### 6.5.4 Using the **debug** keyword

You often need to know exactly what is being computed especially when there appears to be a problem. The higher the **debug** setting (0-3), the more information is returned to the screen and log file. If there has been a failure in **Phreeqc**, the relevant **Phreeqc** output is normally sent to the log file and echoed to the screen and so examining this output should be the first thing to do. The exception to this is during the calculation of predominance plots with **debug**=0 when **PhreePlot** will record a na species and attempt to battle on. In this case, setting **debug**=1 will induce **PhreePlot** to stop immediately it has detected an error.

In general, with the default ('auto') settings for the **Phreeqc.0.out** and all keywords, **debug** works in the following way:

- **debug**=0: minimal diagnostic output
- **debug**=1: selected_1.0.out and Phreeqc.out produced
- **debug**=2: selected_1.0.out, Phreeqc.out and *.all produced
- **debug**=3: as for **debug**=2 except that the input is echoed to the screen on each iteration and the **Phreeqc** output is also inserted into the log file.

**selected_1.0.out** is the SELECTED_OUTPUT file from the last iteration and if produced, will be found in the working directory. **Phreeqc.out** contains the normal **Phreeqc** output from the last iteration and is controlled by the **Phreeqc** PRINT keyword. A copy of this is also sent to the log file and the screen.

**Debug** also controls the degree to which the input files are echoed to the log file. With **debug** = 0, only the main input file is written. With higher debug levels, all include files and the **override.set** file are also copied.

The file *.all file is generated with **debug**=2 or greater, or when the **all** keyword switch is set to **TRUE**. This accumulates **Phreeqc.out** from all of the iterations. This can produce a very large file but it is the definitive record of all that **Phreeqc** has done. It may be necessary to change the **-reset** option in the **PRINT** keyword block to **-reset TRUE** to ensure that all of the **Phreeqc** output is written to the output file.

The log file will also give the value of many of the **PhreePlot** settings, including all the tag values and loop variables that have been generated. It will also have a copy of input to **Phreeqc** after substitution. In the case of a failure, this should be checked for errors to make sure that **Phreeqc** is receiving valid code. This input can also be seen on the screen by setting **debug**=3.

If **Phreeqc** fails (usually because of syntax or setup errors), then the **Phreeqc** output is usually written to the log file and echoed to the screen, and calculations stop. The exception to this is that with **debug**=0 and when calculating a predominance or contour plot, or when **stopOnFail** is set to 0, computations continue unless stopped manually with the **Esc** key.

The **selected_1.0.out** file will indicate whether the expected output is being sent from **Phreeqc** to **PhreePlot**. If **Phreeqc** has failed to converge, this file may not be formed properly.

### 6.5.5 The most common reason for a failure to converge

As mentioned above, **Phreeqc** is a very reliable and well-tested program and rarely fails to converge given ‘reasonable’ chemistry. We have run it through millions of iterations without problems. However, it can be easily be made to fail if it is forced to make calculations under conditions for which it was not designed, namely very high ionic strengths (e.g. above 5 mol/l/
kgw). Such conditions can arise from rather benign starting conditions if the system is subjected to extreme constraints. For example, at the extremes of pe, water decomposes leading to small volumes of water remaining. This concentrates the initial solutes and Phreeqc may, not unreasonably, then fail to converge. This is exacerbated at high temperatures. Therefore in non-obvious cases of failure always check for high ionic strengths or diminishing volumes of water.

A second common failure is when a phase is designated to adjust the activity of some species but in reality cannot. In the present context, this most often arises when trying to fix the pH by adding an acid or base but choosing the wrong one. For example, trying to change the pH of a solution initially at pH 3 to pH 9 by adding HCl is impossible. Phreeqc attempts to do this by adding negative concentrations of HCl (removing HCl) but if there is not enough Cl in the system to do this, Phreeqc will fail. For example,

```
PHASES
  Fix_H
  H+=H+
  log_k 0
SOLUTION 1
  units mol/kgw
  pH  3
  Na  1e-3
  Cl  1e-3
EQUILIBRIUM_PHASES
  Fix_H   -9 HCl
END
```

will fail. If the initial pH was 4, there is sufficient Cl available to be removed and Phreeqc converges without difficulty. Of course, if NaOH is specified as the reactant, Phreeqc does not fail even starting at pH 3.

Our problem is that it is neither always obvious what reactant should be used nor is it necessarily possible to specify a single reactant to cover the entire range of conditions desired (this can be very wide when constructing pe-pH diagrams). For example, redox reactions can produce or consume large amounts of acidity. So changing a sulphate-rich oxidising solution to a reducing one at a higher pH may actually require acid not base to be added because of the large amount of OH⁻ released as a result of sulphate reduction.

One way around this problem is to try and arrange for it not to happen by starting at an extreme pH such that addition of the specified acid/base will always succeed.

Alternatively, add a ‘large’ amount of the co-solute (here Cl⁻) such that the above balancing reaction can always be used to withdraw negative quantities of HCl. More generally when ‘large’ is not known, add a relatively benign equilibrium phase such as NaCl such that it always maintains a finite (but probably small) concentration of the co-solute to allow the required removal to be achieved, e.g.

```
SOLUTION_MASTER_SPECIES
  [Na] [Na]⁺ 0 23 23
  [Cl] [Cl]⁻ 0 35 35

SOLUTION_SPECIES
  [Na]⁺ = [Na]⁺
  log_k 0

  [Cl]⁻ = [Cl]⁻
  log_k 0

PHASES
  Salt
  [Na][Cl] = [Na]⁺ + [Cl]⁻
  log_k 0

  Fix_H+
  H⁺=H⁺
  log_k 0
```
SOLUTION 1
units mol/kgw
pH 3
Na 1e-3
Cl 1e-3

EQUILIBRIUM_PHASES
  Fix_H+ -9  H[Cl]
  Salt -12 [Na][Cl] dissolve
END...

By defining [Na] and [Cl] as new 'elements', there will be no additional side-effects arising from reactions in which Na and Cl are involved. These new notional ions will be included in the calculation of the ionic strength though this can be avoided by making their atomic masses 0.0. Adding 'dissolve' means that this action is only taken when needed, i.e. when Na needs to be added – this becomes more important when simple Na and Cl are used as it prevents the disappearance or 'precipitation' of Cl. Of course this approach does not actually reflect a plausible reaction path.

An alternative approach is to run a simulation, then in the following simulation check whether Phreeqc has run properly by testing the <Phreeqc_status_0> tag, then using this information either re-run the original simulation or change it in some way and then rerun, e.g. change the chemical used in EQUILIBRIUM_PHASES.

Using the -force_equality TRUE option in the EQUILIBRIUM_PHASES keyword block may be necessary to ensure that the target value for one of the axis variables, such as Fix_H+, is reached exactly. This setting should only be used for phases that are definitely present, not for conditional phases. It may be helpful to use it for CO2(g) at high pH but this can cause problems at low pH.

There can be occasional lack of convergence to a reasonable solution brought about by excessive mass transfers from one of the PHASES, e.g. of O2(g) or CO2(g). For example, the mass transfers of O2(g) required to fix the pe in most systems is rather small, usually much less than 0.1 mol/kgw and so unless there are compelling reasons otherwise, include a limited reservoir size as the second parameter in the definition of a the activity of a phase (the default is large, 10 mol), e.g. use

EQUILIBRIUM_PHASES
O2(g)  <y_axis>  0.1

rather than

EQUILIBRIUM_PHASES
O2(g)  <y_axis>

or

EQUILIBRIUM_PHASES
O2(g)  <y_axis>  10.

Similar comments apply to CO2(g). High concentrations of carbonate (> 0.1 mol/kgw) can be created when solutions above pH 10 are equilibrated with CO2(g) at atmospheric partial pressures or above. Even at lower pH’s, it may be necessary to limit the size of the CO2(g) reservoir to a value less than the default value of 10 moles. This can prevent rare problems in Phreeqc convergence.

Limiting the reservoir in this way is part of the normal Phreeqc setup. No error is triggered by Phreeqc when the target phase activity is not achieved because of insufficient reservoir size. Therefore care is necessary with the setup to ensure that what is being calculated is what is required.

### 6.5.6 Changing Phreeqc’s convergence parameters

It is occasionally necessary to alter the default KNOBS settings in Phreeqc to get convergence.
We have found the four most critical options to adjust are \texttt{-iterations}, \texttt{-convergence\_tolerance}, \texttt{step} and \texttt{-pe\_step\_size}. The ‘nuclear’ option for testing would be to set \texttt{KNOBS} as:

\texttt{KNOBS}
\begin{itemize}
  \item \texttt{-iterations 1000}
  \item \texttt{-convergence\_tolerance 1e-10}
  \item \texttt{-step 3}
  \item \texttt{-pe\_step\_size 1.5}
\end{itemize}

The \texttt{-high\_precision} setting should be changed from \texttt{TRUE} to \texttt{FALSE} either in the \texttt{ht1.inc} file or by redefining it later in the input file. Another \texttt{KNOBS} setting that can help convergence is:

\texttt{KNOBS}
\begin{itemize}
  \item \texttt{-diag T}
\end{itemize}

Note that \texttt{KNOBS} only applies to the simulation in which it is placed so if two simulations are used, as is common in calculating predominance diagrams, \texttt{KNOBS} should be placed in the simulation which does the mass transfer calculations.

Another approach to solve the lack of non-convergence is to add the following fictitious species:

\texttt{SOLUTION\_SPECIES}
\begin{itemize}
  \item \texttt{H2O + 0.01\_e- = H2O\_0.01; log\_k -9.0}
\end{itemize}

This can help convergence (see the \texttt{Phreeqe\_3\_manual}, \texttt{KNOBS} p 117) as demonstrated in the \texttt{demo\\_\_Cuedta} example.

6.6 INTERRUPTING EXECUTION AND CHANGING KEYWORD VALUES

\texttt{PhreePlot} can often be stopped or interrupted using the \texttt{Esc} key. This returns the following prompt:

\texttt{Press "s" to stop, "i" for input or <CR> to continue}

\texttt{<CR>} (the Enter key) resumes execution, “s” stops the execution and “i” gives the following prompt:

\texttt{Enter keyword-value pairs/lists, "s" to stop or <CR> to continue:}

at which keyword-value pairs or lists can be entered in the same way that they are entered for input files. These new settings will take effect immediately on resumption of the calculations. No checking on the reasonableness of the new settings is made and since it is clearly possible to cause great confusion, this option should be used with care. A blank \texttt{<CR>} ends the input.

If the letter ‘s’ is entered, execution will be stopped as soon as possible. If a fit is being processed, then execution will not stop until the calculations have been completed for a whole set of data points. If a fit plot has been requested, a plot will be produced that reflects the best fit up to that point. For other calculations, the stopping will be almost immediate.

If a run is stopped during the execution of a batch file, the next line in the batch file will be executed. \texttt{Control-break} will enable execution of the whole batch job to be halted.

Pressing the ‘p’ key (case insensitive) during the execution of a \texttt{ht1} or grid plot will write a plot file, \texttt{plot.ps}, showing the progress of the calculations (this does not work for ‘\texttt{grids}’ plots). This plotting can be automatically done on a regular basis by setting the \texttt{plotFrequency} keyword to the frequency of the speciation calculations for which the automatic plotting is to be done. The plot file will then be renewed every \texttt{n}th iteration. A message, ‘File “plot.ps” written’, is sent to the screen whenever this file is written. This option can only be used when the \texttt{multipageFile} option is set to \texttt{FALSE}, i.e. a separate file is being created for each plot within
a run. This is because only one instance of the plotting routine can be in operation at any time (a single thread) and multipage plots leave the plot file open between plots.

6.7 Running the Standard Phreeqc Examples

It is possible to run most of the Phreeqc example input files distributed with Phreeqc from within PhreePlot.

These examples can usually be run using a minimal template such as:

```
calculationMethod       -1         # calculate but don't attempt to plot 
all                      T          # accumulate output in *.all file
CHEMISTRY
include ".\examples\ex1"
```

The output will be sent to the directory from which the example is run and will include the normal Phreeqc output in the *.all file.

The main challenge with these examples is to isolate the data that need plotting or tabulating from that which does not. In most cases, the aim is to get a well-formed 'out' file. This can usually be achieved with judicious use of PUNCH and the -selected_output switches in the Phreeqc code, and the mainLoop, selectedOutputLines and dataSeparators in the PhreePlot section.

6.7.1 Going round for just one iteration

In predominance plot calculations, it is useful to know the set of minerals that could theoretically form given the input chemistry and database used. Setting the resolution to 1, all to T and ensuring that PRINT is TRUE in the Chemistry section automatically includes a commented block of USER_PRINT statements in the cumulative Phreeqc output (*.all) that give a commented list of all possible mineral species in the system being considered. This text can be pasted back into an EQUILIBRIUM_PHASES data block and those which can realistically be expected to form activated by un-commenting them.

6.8 Return Status and Exit Codes

PhreePlot will normally give a return status of 0 if it has run without errors. If it returns with one or more errors, the return status is normally 1. This can be useful when constructing sequences of runs in a batch file or script.

For example, the following Windows batch file script will only run the second input file if the first one exited successfully:

```
pp file1.ppi
IF %ERRORLEVEL% EQU 0 ( 
  pp file2.ppi
  )
```
7 Plotting basics

7.1 INTRODUCTION

PhreePlot will normally attempt to produce a plot after a run. The type of plot produced is determined by the plot type, currently grid, htl, custom, species, fit or simulate. See the appropriate Sections below for details.

The Chemistry section of the input file largely governs the type of data generated. What is plotted and its appearance is governed by a series of keyword settings. A full list of these can be found in the pp.set file and are described in more detail in Section 14.

7.2 TYPES OF PLOT

There are six options for generating potentially plottable output. These are controlled by the calculationType setting:

- grid: ‘brute force’ method for making predominance or ‘pe-pH diagrams
- htl: hunt and track method for making predominance or ‘pe-pH diagrams
- contour: 2D contour plots
- custom: direct plotting of output from the selected_output file (the default)
- species: plotting species distribution plots, e.g. % species vs pH
- fit: calculating and plotting the fit of observations to a Phreeqc chemical model. The observations and associated independent variables are read from an external file.
- simulate: similar to fit but without the observations and so no fitting.

There are only two basic types of plots: (i) an x-y filled-area plot used for displaying predominance diagrams and contour plots, and (ii) an x-y plot with lines and points for displaying other data. The two types of predominance diagrams use (i) while all the other types of calculations use (ii).

It is possible to overlay lines and points on predominance plots but not vice versa.

In order to suppress the generation of any plot file(s), set plotFactor to 0.0 or use a negative calculationMethod.

7.3 SUMMARY OF BASIC PLOTTING

7.3.1 Introduction

A plot can be produced directly after the generation of the data or by replotting an existing plot without generating the chemical data a second time. This is governed by the calculationMethod keyword.

7.3.2 Setting up the plotting area

A plot is positioned on a notional piece of paper. The size of the paper is set with the paperSize keyword. It can be either one of the ISO sizes (A0-A5, B4-B5), US sizes (Ledger, Letter or Legal) or Note. This should be set to your preferred units in the pp.set file.
The pageOrientation can be set to 0 (portrait) or 1 (landscape).

It is simplest if all dimensions use the same units as defined by the units keyword. The units can be ‘mm’, ‘inch’ or ‘pt’. The units used during plotting are determined by the last set value of the units keyword as read from the various input files. This applies to any features defined in the extraText or extraSymbolsLines files. This is best set in the pp.set file so that it is read in first and does not need to be reset. All of the other default settings that use these units should also be changed in the pp.set file.

The font can be set for the document as a whole but the font used cannot be changed for individual text strings except through the extraText mechanism. It is best to set your default font in the pp.set file.

The plot area is set on this piece of paper. The bottom left-hand corner (‘origin’) of the plot area is offset by xoffset from the left and yoffset from the bottom of the paper. The length of the x axis is given by xaxisLength and the y axis by yaxisLength.

The colorModel used is either ‘rgb’ for red-green-blue, ‘gray’ for grayscale, or ‘b&w’ for black and white. Colours for text, lines, symbols and fills are specified on the rgb scale by colour codes such as red4 and, if necessary, transformed to the other colour models.

The background colour of the plot area (within the axes) is given by backgroundColor(1). This setting is less useful with predominance plots as they are normally completely filled with colour (including ‘white’) anyway. The background colour of the whole page is given by backgroundColor(2).

plotFactor can be used to rescale everything plotted by a given factor.

7.3.3 Setting up the axes scales, tick marks and grid lines

The axis scales are set up by minimum and maximum values of the axes on user coordinate x- (pxmin, pxmax) and y- scales (pymin, pymax) with the ‘p’ standing for ‘plot’. The first tick mark, a major tick, for both axes is always placed at the plot origin (bottom left-hand corner). Major tick marks are then added at intervals of pxmajor and pymajor and minor ticks at pxminor and pyminor. By default, minor axis ticks are plotted at the centre of each major tick interval. The minor ticks can be turned off by setting pxmajor or pymajor to 0.

If any 2y lines or points have been specified, a second y axis (‘2y axis’) will be used on the right-hand side of the plot with settings p2ymin, p2ymax, p2ymajor, and p2yminor. It shares a common x-axis scale with the main y axis. It is used by specifying the lines2y or points2y variables.

All or any of these settings can be set to ‘auto’ for automatic scaling based on the range of data being plotted.

The colour of the axes is set with axisLineColor and the width with axisLineWidth. Four axes will always be drawn, each with major and minor tick marks. Tick lengths are set with tickSize and colour with tickColor. Separate tick sizes can be chosen for each of the major and minor x axes, major and minor y axes, and major and minor 2y axes and up to six tick colours can be similarly specified. The line width of the major tick marks is the same as the axis line width while the width of the minor tick marks is half the axis line width. By default, the minor ticks are half the length of the major ticks.

tickSize can also be used to specify the position of the ticks in relation to the axis – ‘inside’ or ‘outside’. Just add the ‘inside’ or ‘outside’ qualifier to the end of the list of ticksizes.

Grid lines can be plotted by setting the appropriate gridLines setting to TRUE or choosing a very large tick size (more than half the length of the ‘other’ axis). These can be dashed if the corresponding gridLineType style is greater than 1. Alternatively, a negative tick size can be used. The appearance of the dashed line is also controlled by the gridColor and gridDashesPerInch settings.
7.3.4 Axis numbering and annotation

Axis numbers will be drawn at the major tick marks. The numbers of digits after the decimal point are determined by $pxdec$, $pydec$ and $p2ydec$. -1 means integer. These can be 'auto'.

Very large and very small numbers automatically invoke scaling of the numbers using the 'divide/multiply by a power of ten' approach.

The size of the axis numbers is given by $axisNumberSize$ and their colour by $axisNumberColor$.

Axis titles are given by $xtitle$ and $ytitle$ along with $axisTitleSize$ and $axisTitleColor$. An optional second string for the axis title is put after any 'divide by a power of ten' scaling so that the units can be correctly placed to give dimensionless scales.

An overall plot title is given by $plotTitle$, $plotTitleSize$ and $plotTitleColor$. This is centered above the plot.

7.3.5 Adding fills, lines and points

Fills can only be added in predominance and contour plots. Lines and points (or symbols) can be added to these plots or more commonly used to generate their own 'custom' plots. All this plotting is controlled by reading data from files, either from files generated during the run or from existing files. These files are all ASCII files and so can be viewed and edited with a text editor. These files should normally be in a spreadsheet-type format in rows and columns. They should have a header row which is used to name each column. The separators can be specified as spaces, tabs or commas.

Predominance plots maintain a $fillColorDictionary$ which holds the colour to use for each species. If the dictionary is not initially present or a species not found, a sequence of pale fill colours will be automatically generated and used. The fill colour dictionary will always be updated with the colours used at the end. A labels file is also generated with the positions of the labels used. This can be edited to move the labels.

Contour plots normally derive their properties from a series of lists, one value for each contour, specified by settings such as $contourFillColor$. These lists are recycled if they are shorter than required.

$points$, $points2y$, $lines$ and $lines2y$ are used to specify the lists of custom data series to plot. The names used are the column names normally from the outfile though data series from extra files can be added with $extradat$ providing that they share a common x-axis name given by $customXcolumn$ name. Column numbers can also be used for specifying a column starting with the outfile and then the $extradat$ files in the order specified.

Points are plotted with symbols, some of which are 'filled' symbols which have separate fill and rim colours. The filled colour can be white and so when combined with a coloured rim, can give the appearance of open circles. Point size(s) are determined by $pointSize$ and colour(s) by $pointColor$. The colour(s) for points from data series for which the colour has not been defined by $pointColor$ are either automatically selected based on a rotating 15-colour pallette or are taken from the line colour dictionary depending on the $useLineColorDictionary$ setting.

All lines are drawn with a thickness given by $lineWidth$. Negative values will give dashed lines. The dash density is given by $dashesPerInch$. As with the points, the starting line colour(s) are either determined by the $lineColor$ setting and then automatically follow the default sequence, or are taken from the line colour dictionary.

The line colour dictionary also contains information about the in-plot position of labels for the lines and can be edited and replotted accordingly. $labelSize$ and $labelColor$ can be used to adjust the appearance of the labels or turn them off completely. Label names for lines are automatically generated from the column names but can be overridden with the $labels$ keyword.

It is possible to synchronise the point and line color for the same data series with $pointsSynchronise$.
A simple legend with an optional `legendTitle` and `legendBox` is automatically drawn to the right of the plot. It can be turned off by setting `legendTextSize` to 0. The position of the legend can be moved with a line of `extraText`.

Additional lines, points, symbols and text can be added to a plot with `extraLinesSymbols` and `extraText` as described in more detail below. Points or lines from additional files can be added with `extradat`.

### 7.4 Controlling the Style of Lines and Points (Symbols) in Custom Plots

#### 7.4.1 The default style and colour for lines and points (symbols)

Each `lines` variable has a `lineType` associated with it. This defines the line style and is specified by a number from 1 to 20:

- 1 solid line
- 2-10 dashed line with an increasing proportion of space
- 11 dotted line
- 12-20 dot-dashed line with an increasing proportion of dash

The appearance of a line can also be changed with `lineColor` and `lineWidth` as described below. There are 2y equivalents of these keywords.

Similarly, the symbol styles of `points` are defined by their respective `pointType`. See below for a description of the symbols.

Each dataset (lines, points, lines2y, points2y) has its own list of 15 colours. These are picked off one by one as needed. Specifying specific colours in an input file promotes that colour or colours to the top of the list.

Each list starts with the same default list of colours. This is:

```
red
blue
green
orange
cyan
magenta
brown
sky
purple
gray
yellow
maroon
lawn
spring
black
```

So by default, the first colour to be used will be `red`, the second `blue`, etc. The default colour density is 4, i.e. `red4`, `blue4`, ...

The various colour keywords such as `lineColor` provide a mechanism for promoting a colour to the top of the list.

```
lineColor purple green
```

means that the line colour list becomes:

```
purple
green
red
blue
orange
cyan
magenta
```
Once the list is exhausted, it is recycled but the colour density may also be cycled 4, 6, 8, 2, 4... depending on the `changeColor` setting. Points are plotted after lines and may inherit the same colour as the corresponding line by using the `pointSameColor` setting.

### 7.4.2 Lines

#### Line styles

The line style pattern is defined by a number in the range 1-20. These are shown in Figure 7.1 using a dash density of 5 dashes per inch. The demo file, `\demo\linestyle\linestyle.ppi`, can be used to generate a figure similar to that below for a range of dash densities.

![Figure 7.1](image-url)

Figure 7.1. Examples of the 20 line style patterns available. These are specified with the `lineType`, `lineType2y`, `contourLineType` or `gridLineType` settings. The appearance can also be altered by varying the dash density (`dashesPerInch`), line width (`lineWidth`) and line color (`lineColor`) settings, and their 2y and grid relatives.
Automatic colouring of lines and points

The colour of lines and points are automatically selected based on the amended colour sequence in effect (see above). The colour picked is determined by the position of the variable being plotted in the input list (lines, lines2y, points, points2y) and the position of any colours specified explicitly in the corresponding colour setting (colorLines, colorLines2y, colorPoints, colorPoints2y) if necessary taking into account the recycling rules, e.g.

```plaintext
lines pH Ca Mg Na K
lineColor purple green
```

will give the following colours: pH = purple4, Ca = green4, Mg = purple4, Na = green4, K = purple4.

Colours can be set multiple times to force a particular colour. Rims for open-filled symbols also have their own colours that follow the same colouring rules.

Sometimes there are subsets of data with the same heading. This occurs where a line break (blank line) in the data file has been found. The colour for all lines from this column will follow the colour selected above but the colour density will change if the colour was specified without a colour density (no number at the end, e.g. red) and if changeColor is TRUE. If the colour is specified explicitly as red4 then this will always be used with no change in density.

‘auto’ can also be used as a colour. This will cause different colours to be used following the colour sequence in effect. ‘nd’ (not drawn) is also a colour that is distinct from the background colour.

When there are multiple plots/datasets per run, restartColorSequence controls whether the line color sequence for auto-generated colours is restarted from the beginning of the sequence for each plot/dataset.

Colours can also be controlled explicitly with the line colour dictionary (Section 7.9.4). All points and line segments within a given sequence will have the same colour. If different attributes are wanted for each point or line segment, use an extraSymbolsLines file.

The selection strategy outlined above is followed for the colour of all lines and points, y and 2y.

### 7.4.3 Points (symbols)

Symbols can be plotted to represent the data points in a curve. The symbol used is specified with the pointType keyword and can be specified by a number or a name. The available symbols and their symbol codes are shown in Appendix 3 and Figure 7.5.

The six standard filled symbols and their code numbers are:

1. filled circle
2. filled square
3. filled triangle
4. filled upside down triangle
5. filled diamond
6. filled octagon

The list of symbols is recycled as needs be. The default symbol type is 1.

If pointsSameColor is FALSE, pointColor controls the colour of the symbols using the point colour list as for the lines described above.

If pointsSameColor is TRUE and lines are plotted, the symbols will always have the same colour as their corresponding lines irrespective of the pointColor setting.
There is no guarantee that all symbols will be centered exactly though the first six should be.
The size of symbols is specified by the `pointSize` keyword list.
Filled symbols can have a separate rim colour (`rimColor`) and rim width specified as a fraction
of the corresponding symbol size (`rimFactor`). This enables open circle symbols to be specified.
If these lists are short, the values are recycled.
The minimum size of positively-sized symbols is set to 0.01 inch. A size of 0.0 suppresses plotting of symbols.

7.4.4 Order of plotting of lines and points (symbols)
The general order of plotting of lines and points (symbols) is controlled by the `plotOrder`
keyword. The default is lines then points. Within a given class, e.g. ‘lines’, the order of plotting of individual lines is controlled by their order of definition in the keyword in the input file(s).
The order of plotting affects any overprinting - the last plotted will appear on top.
Although `plotOrder` does control the order of plotting of separate sets of lines and points,
when the same variable is plotted as both a line and a set of points, the line is always plotted first. This means that the points will always overprint the line.

7.5 Labelling plots

Fields and lines in plots will normally be labelled, although the precise way this is done
depends on the type of plot and the various settings available.
Labelling of plots is important but it can be difficult to automate effectively. It is therefore
always possible to edit a ‘labels’ file to rename labels or to move their positions. Some details
on the choice of label names and their positioning is given below.

7.5.1 Label names and label position

Label names (up to 30 characters long) are automatically assigned a value depending on the
type of plot.
In ‘ht1’ and ‘grid’ plots, label names are derived directly from the species names and x- and y-
coordinates of the label positions are approximately centered in the field. The species name is
in turn derived from the database and any changes subsequently made by the `ht1.inc` or similar file. The labels file can be edited to change the label positions and/or the label attributes and the plot replotted (calculationMethod = 2). The plot should not be recalculated (calculationMethod = 3) as this will recalculate the label positions and return them to their original positions.
The label position for predominance plots is, by default, placed near the centre of each field.
In custom and fit plots the positioning is more complex and an attempt is made to place the
labels in a legible place (Section 7.10.3). This can take a lot of effort to calculate – a brute force (simulated annealing) approach is taken. The time taken is broadly controlled by the `labelEffort` setting (0-3). The second parameter, if present, provides an upper limit to the time taken (in sec).
In custom plots, the centre of the label position is stored in the line colour dictionary which
can be edited and the plot remade. The label name can be edited in the same way. Automatic
label positioning is affected by the `labelEffort` setting.
Label names for each of the ‘curves’ (including a set of points) can be overridden with a list of
names given by the `labels` keyword. ‘label’ names are simply picked from this list one by one as
required. Multiple curves generated in the same iteration because of the presence of line breaks
will be picked first, then multiple values of the loop or z-variable, then any more ‘outer’ iterations. The list is recycled if necessary. Label names for custom plots can also be read from the first column of the `loopFile` if present though the `labels` setting takes priority if not null.
When labels are derived from column headings, these may be set in the input file or in the case of a simulate or fit plot, from the column headings of the fit data file.

Label names are automatically tested to see if they are plausible **Phreeqc** chemical formulae and if so, subscripted and superscripted appropriately. Whether a label name is interpreted as a formula or not depends on its format (Section 6.4.2). This conversion can be bypassed by setting `convertLabels` to `FALSE` or by including a non-printing character in the label name (Section 6.4.2).

Label names stored in the line colour dictionary and used as labels in plots and legends are automatically appended with a special character when referring to the secondary (2y) axis. By default this is a `*` but this can be changed (see `ytitle`). This character should not be used as the last character in undecorated label names.

### 7.5.2 Overriding calculated label positions and angles

Sometimes it is necessary to omit a particular label from a plot or move its position or change its angle.

For predominance plots, removing a label can be done by changing the relevant species number in the labels file to a negative number, e.g. change 4 to -4.

This can also be achieved in other plots by editing the line colour dictionary as described above.

The following approach applies to all types of plots except contour plots which have their own way of shifting label positions (the labels in contour plots are in-line so cannot be randomly shifted).

The label positions and angles can be changed by making a label overrides file which redefines or ‘nudges’ the position of a label. These files are named with the `nudgeFile` keyword and can either supply the incremental shift (`diff`) or a new value (`abs`) for x, y and angle. Normal input file searching rules apply for finding these files. So by placing the file in the system directory, it can apply to all future plots.

These files should have a single header line (e.g. to remind yourself of the column headings) followed by 6 or 7 columns of data in free format as follows:

```
plot species_name type xmm ymm angle pos
```

<table>
<thead>
<tr>
<th>plot</th>
<th>species_name</th>
<th>type</th>
<th>xmm</th>
<th>ymm</th>
<th>angle</th>
<th>pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>&quot;H2(g) &gt; 1 atm&quot;</td>
<td>diff</td>
<td>1.2</td>
<td>-6</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>auto</td>
<td>&quot;O2(g) &gt; 0.21 atm&quot;</td>
<td>diff</td>
<td>1.2</td>
<td>5.5</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>&quot;Fe+3&quot;</td>
<td>diff</td>
<td>4</td>
<td>6</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

If the type keyword parameter is `abs`, this sets a new value for x,y coordinates and angle. The position set by the coordinates is for the centre of the label in the units used for plotting (mm, inch, etc). This contrasts with the labels file where the x,y coordinates are specified in terms of graph units and so a nudge labels file can be useful for automating shifts in batch processing where different coordinate systems are used. x and y are for the anchor position of the label; y is located at the vertical centre of the label (when horizontal), and the angle is in degrees from the horizontal, rotated clockwise. The seventh column, `pos`, is only used for `abs` shifts and specifies whether the anchor position given is for the left end of the label (0), middle (1) or right end (2).

If the type parameter is `diff`, then this does the same but specifies the shift in the position and angle of the label from its calculated position. The `pos` column is not used.

`plot` refers to the sequential plot number (as given by the `info` block) of the file for which this override applies. ‘auto’ means that it applies to all of the plots.

The overrides only apply if the plot number and species name match that in a plot.

It is possible to delete a label from plotting by making the x-y coordinates off plot. A nudge file can also be useful for rotating the labels for the water limits in predominance plots. The angle for this is given in the log file.
7.5.3 Legend

A legend is automatically drawn for custom plots.

Normally a simple legend will be drawn to the right of the plot. This shows the colour of the plotted lines and symbols against their label names. The size of the legend text is controlled by `legendTextSize` and the line thickness is the same as in the plot. The `colour` of the legend text is controlled by `legendTextColor`.

The order of items in the legend is controlled by the `plotOrder` setting and the orders specified in the `lines` and `points` settings in the input file(s).

The legend will not be drawn if `legendTextSize` is 0.0 or if all of the lines or symbols have the same colour.

The legend and all labelling can therefore be suppressed by setting both `legendTextSize` and `labelSize` to zero. Individual labels can be suppressed by setting their colour to `nd` in the line colour dictionary and forcing the dictionary to be used by setting `useLineColorDictionary > 0`.

A legend box can be drawn around the legend using `legendBox`.

The placement of the legend can be changed by specifying its position in the `extraText` file using the special `<legend>` option. `<legend>` is not like a typical tag but rather acts as a placeholder for the legend contents.

This approach can also be used to suppress the legend by specifying its coordinates to be outside of the page area.

A legend title can be added by preceding the `<legend>` tag with text, e.g.

```plaintext
auto 9 -20 "<b>Concentration</b><br> mg/L<legend>" 1.5 blue
```

`auto` in the line above means that the text will be applied to every plot. Any text after `<legend>` but within the double quotes is ignored. The position of the legend can be automatically controlled using the `<pxmin>` etc tags (see Section 7.12).

### 7.6 INPUTTING TEXT STRINGS

#### 7.6.1 Available fonts and character sizes

Fonts from the Helvetica, Helvetica-Narrow, Bookman, Avantgarde, Times, Palatino, NewCenturySchoolbook and Courier font families are available (see font). These are the eight font families included in the 35 standard Postscript fonts. The regular, italic and bold faces of these fonts can be specified with PhreePlot and should be able to be displayed and printed by Postscript-conforming devices. The italicized ZapfChancery font is also available.

The `symbol` and `dingbats` fonts are also used for plotting Greek characters, symbols and various icons.

All of these fonts are available with the standard Ghostscript distribution.

However, the fonts cannot be mixed. Only one of the font families can be specified for the main text (in titles etc) although text enhancements such as bold and italic can be used. Text in other fonts can only be written using the `extraText` mechanism.

The set of characters available within a font depends on the character encoding used. The default is ‘Latin-1’ which includes many of the accented characters of Western European languages plus a few other symbols. The popular Windows-1252 character set adds a few extra accented characters to this (plus the euro sign) but is not available here. The alternative, ‘Standard’ encoding, includes all 7-bit ASCII characters plus a smaller range of extended characters which depend on the setup of your PC. This standard set was the default character set in PhreePlot until January 2014. The ‘ASCII’ encoding only includes the 7-bit ASCII codes (decimal 0-127) which includes all the numbers, lower and upper case Latin alphabet,
punctuation as well as some special characters (as found on most keyboards). This was used in earlier (pre-2014) versions of **PhreePlot**.

The size of characters can usually be specified by a character size parameter. These are nominal character sizes and do not usually match the actual size of the characters as plotted. For example, an uppercase 'O' in Helvetica font with a specified size (height) of 10 mm will actually be about 12.8 mm high. Other fonts will differ somewhat from this.

### 7.6.2 Available characters and inserting ‘special’ characters

Text is required as input for various options such as the plot title, axes titles and extra text. General features of text input are:

All standard ASCII characters (32-126) are available including numbers, alphabetic characters and some special characters. These include: `\|!"#$%&*()_+-=\{\}\;:\;;@~#<,>.?/`. A space can be included but when this is done, the entire text string must be included in single or double quotes, e.g.

"Zinc concentration" or 'Zinc concentration'

The two types of quote should not be mixed. If a single quote character itself needs to be included as well as a space, embed the text string in double quotes; *vice versa* for including a double quote character. A warning will be given if an unpaired quote is found without being embedded in quotes. In this case, the string will not be plotted. Note that the quotes should be of the simple vertical type, not the angled open/closing quote marks beloved of word processors and their fancy fonts (as here, so be careful not to copy/paste these!)

"It’s easy"

and

'A double quote (") can also be used.'

are acceptable.

It’s easy

and

(')

are not. If in doubt, include the text string in paired quote delimiters.

Ultimately the fonts available to the printer and display device will determine which characters are available. In principle, most devices are able to print the standard ASCII set of characters without a problem. **Example 68** provides a view of many of the available characters. The full range of characters is determined by the encoding and can include accented characters.

Text strings can contain system and user-defined tags. These should be assigned values using the `numericTags` or `characterTags` keywords or have their values assigned by **PhreePlot**.

If the character is not readily available on your keyboard or input device, then any valid character can be added to a text string by using a backslash followed by the 3-digit octal code for the character from the character set in force (\000-\377). For example, using the Latin-1 encoding, 20¢ (twenty cents) can be entered as 20\242 since 242 is the octal code for cent (decimal 162). In order to avoid backslashes being misinterpreted when wanted as such rather than being interpreted as part of the octal code, enter them as double backslashes. They will print as single backslashes.

### 7.6.3 Text enhancements (bold, italic, subscript, superscript) and Greek characters

A certain degree of text enhancement is possible although the possible combinations are rather
limited. The following tags are available for modifying the appearance of text. Case is significant. All text enhancement tags should be in lowercase.

Most of the tags are paired and should be turned ‘on’ and ‘off’ in their nested order otherwise unpredictable output may result.

**Bold:** `<b>This is bold text</b>`

**Italic:** `<i>This is italic text</i>`

**Bold:** `<b><i>This is text</i></b>` # N.B. tags must be properly nested

**Superscript:** `e = mc<sup>2</sup>`

**Subscript:** `Ca(NO<sub>3</sub>)<sub>2</sub>`

**Sub and superscript:** `Sum <subsup>under over</subsup>`

Greek strings: `<g>abcdef</g>` gives αβχδεφ.

The mapping of lower and uppercase Greek characters is:

- lower case: αβχδε ϕηιμ φλνπ ωξψζ
  - alpha: a
  - micro: μg/L

An alternative way of getting a single Greek character is to precede the corresponding letter with a backslash:

- \a\ gives α
- \mg/L\ gives μg/L

If a backslash itself is wanted and is immediately succeeded by a character, use two consecutive backslashes or insert a blank enhanced string immediately after the backslash, e.g. `</i></p>` will print \p.

Characters embedded in a Greek string for which there is no translation to a Greek character will be replaced by a space.

**Break:** `<br>` produces a line break. Each line is treated as a separate text string. Therefore any other tags such as `<b>` /`/ must be properly paired tags before and after any `<br>` and may need to be repeated, e.g. `<b>bold1</b><br><b>bold2</b>`.

All but `<br>` are paired tags. If one of the pair is missing or has been mistyped or the tag has not been recognised, that part of the string will be printed as is.

In most cases, text enhancement tags (i.e. Greek, bold, italics, subscript, superscript, subsuperscript) cannot be embedded within one another, e.g.

```
<sup><i>this gives superscript but not italics</i></sup>
```

and

```
<b>this will not be bold</b><br>this will not be bold</b>
```

will not work and may produce incorrect output but

```
<b>bold</b><i>italics</i>
```
and

```html
<b>bold</b><br><b>bold again</b>
```

will work. The exceptions are that bold can be used with other tags: `<b><i>...</i></b>` and `<i><b>...</b></i>` will both produce the bold-italic font and `<b>Cu<sub>T</sub>` will work as hoped. Note that the order of the ‘off’ tags is important to maintain the correct nesting; otherwise unpredictable results may occur. Illegally nested tags will be ignored or incorrectly translated.

It is **not** possible to define subscripted superscripts such as \( a_{Fe} \) by

```html
a<sub>Fe</sub><sup>3+</sup> (illegal)
```

or superscripted superscripts

```html
a<sup>Fe</sup><sup>3+</sup> (illegal)
```

or superscripted Greek characters.

It is not possible to enhance any Greek characters, e.g. Greek italics or Greek bold are not supported.

It is possible to embed many PhreePlot tags such as `<loop>` and `<mainspecies>` (but not `<input>`) between text enhancement tags since the text substitution has already taken place before being interpreted for plotting:

"As = 10<sup><loop></sup>M".

Ensure embedded spaces are enclosed using single or double quotes.

### 7.6.4 Accented and other ‘foreign’ characters - the Latin-1 encoding

The default or ‘Standard’ encoding is based on the ASCII 7-bit characters plus a variable number of characters from an ‘extended’ character set as found on many keyboards. Many Western European languages have accented characters which are not in this set. These are found in the ISO-8859-1 character encoding which is similar to the Windows 1252 (Western European) character set minus a few characters.

The Latin-1 character encoding is supported in Postscript and in the Postscript fonts supplied by Ghostscript and many other Postscript interpreters. The full character sets are shown in Appendix 4.

There are two ways of entering accented and foreign characters not available directly from your keyboard:

(i) most Windows text editors such as Notepad and Notepad++ support the Latin-1 encoding (and others) and accented characters can be entered by using Alt-num key codes. This is done by holding the Alt key and then typing the decimal code for the character with the numeric keypad including the leading zero, e.g. Alt-0200 for È. The editor must be configured to view and export the text with the correct encoding (i.e. ISO-8859-1 or ANSI) which may not be the default.

(ii) Postscript interprets 3-digit numeric strings preceded by a backslash as octal codes and associates these with the appropriate characters according to the encoding in force, e.g. \310 for È and \350 for ë with Latin-1. So these octal codes can be included directly in text strings, e.g. caract\350res europ\351ens for caractères européens.

The first method is probably better if the Latin-1 encoding is being used since this follows closely the Windows encoding and so the text can be read and checked visually in the monitor. The second method is less easily interpreted on-screen but avoids any problems with encoding of the text from editor/monitor to Postscript. Internally, the extended characters are
always replaced by their octal codes in the Postscript output.

The usual limit on the length of plotted text strings of 200 characters remains. This includes the necessary replacement of the extended characters using their octal representation, i.e. each extended character takes four ASCII characters. Long lines will either be truncated or any translation aborted.

The default is to interpret text strings with the Latin-1 encoding. In order to enable the Standard encoding, use the `font` keyword

```
font Helvetica Standard
```
or to retain the current font

```
font Standard
```

The encodings for both 'Standard' and 'Latin-1' encodings are given in Appendix 4. Alternatively visit a website such as http://en.wikipedia.org/wiki/ISO/IEC_8859-1 for Latin-1.

As well as the accented characters, umlauts and inverted marks, Latin-1 includes some other useful characters such as the degree (°), plus-minus (±) and some common fractions. On the other hand, the standard encoding has some characters that Latin-1 does not, such as the permil symbol and the oe ligature.

### 7.6.5 Non-printing characters

It is sometimes useful to add a non-printing character to the plotted output. Postscript interpreters usually just ignore these characters.

Whether a character (decimal code) will print or not depends on the font and its encoding. With the ASCII encoding, the ¬ (‘Not sign’) character does not print and is found on many keyboards. This does print with Latin-1 and Standard encodings so other characters are required. Characters not available on keyboards can be entered with either of the two methods discussed above, namely Alt-num key codes and octal codes. Characters with decimal codes from 129 to 159 (octal codes 201 to 237) may not be defined and are not normally printable but avoid those before decimal 138 (octal 212) as these are used internally by PhreePlot.

### 7.6.6 Justification

This can usually be specified as either left, centre or right justified horizontally. The text is also aligned on the text baseline (bottom of the letter a) though the exact position can depend on the particular characters and font. Where a `<br>` is included, the x and y coordinates refer to the bottom of the first (top) line.

For accurately centered symbols, use the `extraSymbolsLines` symbols rather than the `extraText` symbol font.

### 7.6.7 Angle

The text can be rotated. The angle of rotation is given in degrees from the horizontal, rotating clockwise.

### 7.6.8 Tags in text strings

Tags can be added to text strings and their values, if known, substituted at plot time. Tags generated during a run are not stored and so will be `UNDEFINED` on replotting.

### 7.7 SPECIAL PHREEPLOT VARIABLES OR TAGS

#### 7.7.1 Available tags

A number of special variables, or tags, can be included in a PhreePlot input file or extra text file. These are substituted by values or specific operations at run time. These are:
PhreePlot Guide

<x_axis> The current value of the x-axis variable
<y_axis> The current value of the y-axis variable
<loop> The current value of the loop (z) variable
<logloop> The current log10 value of the loop (z) variable
<mainspecies> The name of the main species
<legend> The entire legend key as used in the current plot
<input...> Part of the input file (only in extraText files)
<pxmin> etc The value of pxmin etc at plot time
<command_line0> etc The values of the command line arguments

In addition, several special tags are automatically produced during a fit which contain information about the fit.

7.7.2 System pH, system pe and system temperature

Internally, PhreePlot needs to know certain system variables to convert from one yscale to another. Sometimes this information is passed through standard PUNCH output formats, e.g. for predominance plots (ht1.inc etc), but sometimes it is not, e.g. for contour plots. This information can be passed by PUNCHing the appropriate columns and naming the columns in a particular way. These three variables, pH, pe and temperature (Celsius) are recognised by the column headings, “pH”, “pe” and “TC” (case sensitive), respectively.

7.8 AXIS SCALING

7.8.1 Auto or user-defined axis scaling

Axis scaling is set with keywords such as pxmin, pxmax etc. Axis scaling can either be automatic or manual. A keyword value of ‘auto’ means that PhreePlot attempts to choose the axis scaling so that all valid data in the data file are included in the plot and the tick intervals are at ‘pretty’ intervals.

Manual scaling by setting pxmin etc gives more control over the minimum and maximum range, the numbering of the axes and the positioning of tick marks. It also enables ‘zooming in’ on particular parts of the plot without recalculation though this is often better done by recalculating with the new domain settings. Automatic label placement for lines is only carried out after a new calculation.

The axis labelling always starts at the bottom left hand corner usually at xmin, ymin. The x and y axes are then labelled every pxmajor (or pymajor) graph units until xmax (or ymax) is reached. There is a major tick mark at each label. There is not necessarily an axis label at the maximum value. Additional tick marks are calculated according to the value of the pxminor and pyminor.

When a plot has a max-min range of 0–100, say and a lot of data are at or close to 0., then this can create a lot of untidy plotting and labelling close to the lower x axis. This can be avoided either by removing the offending columns completely from the plot or by shifting the y-axis scale by a small amount, e.g. to 0.001 and 100.001, respectively. This will remove the columns where all the data are below 0.001 from both the plot and from the key. Of course, some information is lost in the process. It is also possible to eliminate lines by setting the minimumYValueForPlotting keyword at an appropriate value.

If the scope of a predominance plot calculation (set by xmin, xmax etc) is changed and the plot is replotted rather than recalculated then the scale and positioning of the polygon labelling and axis scaling will reflect the selected data from the original files and may appear somewhat odd - not much of the original data may be selected if the plot area is a small proportion part of the original area or if not many points are selected from the polygon file. The corre-
sponding polygon and label files should therefore be regenerated using 'reprocess (labels) and repplot' (calculationMethod = 3) or 'Calculate and plot' (calculationMethod =1) to ensure the correct display of labels within the specified domain.

7.8.2 Secondary y axis (the 2y axis)

The left-hand vertical axis is the main y axis. The ticks on this axis are normally mirrored on the right-hand y axis. It is also possible to define different scaling for the right-hand y axis (the secondary or '2y' axis) using keywords such as p2ymin, p2ymax, 2ytitle etc. Variables for this axis are specified for this scale using points2y and lines2y keywords as for the main y-axis variables. An example in which the 2y axis is used as an expanded y-scale is shown in Figure 7.2.

![Image of Al solubility vs pH (with 2y axis)](C:\PhreePlot\demo\AlsolPH\Alvsph2y.ps)

Figure 7.2. Use of the secondary (2y) axis to provide an expanded y-scale for displaying the saturation

The 2y title is plotted if any variables have been defined with points2y or lines2y. If 2ytitle is set to 'auto', the first variable name from points2y or lines2y is used. If labels or a legend are drawn, the names of any labels referring to the 2y axis have, by default, an asterisk ('*') appended to their label name. This modified name is also stored in the line colour dictionary. The asterisk can be replaced with any single character using the third parameter of the 2ytitle setting.

There is no corresponding minimumYValueForPlotting for the 2y axis.

7.9 CONTROLLING THE PROPERTIES OF TEXT, SYMBOLS, POLYGON FILLS AND LINES

7.9.1 Principles

Text, symbols, polygon fills and lines have various properties associated with them such as type, size and colour.

Some of these are fixed by PhreePlot but many can be set in PhreePlot. However, the way
that this is done can depend on the type of plot involved – predominance plot, custom plot or contour plot. These properties often take on default values specified in the `pp.set` file but many of the colours, such as those for lines and symbols, can either be auto-generated or set more explicitly.

Some properties such as the colour of the plot title only have a single value and these are usually set by a separate keyword, here `plotTitleColor`. Other properties, e.g. those of lines and points (symbols) in custom and contour plots, have an array of values, one for each separate line or set of symbols.

In custom plots, these array properties are set in two ways: line and point colours are automatically picked from a sequence of 15 colours including black while other properties such as line width, point size and where appropriate, rim colour and size, are recycled from the list associated with the various keywords such as `lineWidth`, `pointSize`, `rimColor` and `rimFactor`. The principle here is that some properties such as `lineWidth` are often constant within a given plot and so it would be tedious to specify them for each line. Hence properties such as these only need to be specified once and are then recycled.

However, it is often desirable to colour each line separately (within reason) and so here a longer list is recycled. There is always a default and ordered list of colours associated with the plotted lines but specific colours can be promoted to the top of the list to ensure that they are used first. Four of these colour lists or sequences are stored, one for points (symbols), one for lines, and one each for their 2y counterparts.

Lines of a particular dataset are always plotted before the points. This ensures that the points will overwrite the lines.

The ways that these colour sequences are used and their interactions are controlled by three keywords: `pointsSameColor` (ensures points have the same colours as any associated lines), `changeColor` (individual datasets are plotted with different colours as far as possible), and `restartColorSequence` – if true, restarts the auto-generated color sequence for each new plot and for each new plot type (lines, points).

Colours from custom plots are stored in the line colour dictionary. This can be edited to change the line and symbol properties (see `useLineColorDictionary`).

With contour plots, the properties of each contour are derived from a corresponding list, e.g. `contourLineColor`.

Fill colours used in predominance diagrams are also automatically selected from a colour sequence but unlike line and point colours cannot be preset. They can however be changed by editing the fill colour dictionary and replotting.

While the colours used are always stored with their Cohort colour names, the rendered colours may be changed if the grayscale or black & white colour models are used.

Details about how to add additional text to a plot and how to format numbers is given in Section 7.12.

### 7.9.2 The colour palette

Colours are defined using the Cohort rgb colour palette ([Cohort Software, 2004](#)). This has 14 base colours, each one with 10 shades of increasing colour density or darkness (Figure 7.3) plus ‘black’, ‘white’ and ‘nd’ (for ‘not drawn’). The base colours are centered on number 4, e.g. ‘red4’ is pure red. Numbers from 3 to 0 have increasing amounts of white in them while numbers from 5-9 have increasing amounts of black in them. Therefore ‘red0’ is the palest red (more like a pink) and ‘red9’ the darkest red. Colour names are not case sensitive.

A null colour string, ‘’, is interpreted as ‘take the next auto colour’ and so is different from ‘nd’.

If a colour name is not recognised, then black or white are used depending on the context (black for text and lines; white for fills).
7.9.3 Automatic or explicit

The colour of text, symbols and lines can be explicitly specified in the input files. If this is not done, PhreePlot will choose its own colours according to a set of rules. These are described below. The colours used in the plots can be changed without recalculating the original plot. This is done either by specifying or changing the relevant attribute in the input file or editing one of the colour dictionaries and then re-running the problem with calculationMethod set to 2 (just replot) or 3 (reprocess and replot).

7.9.4 The colour dictionaries

There are two colour dictionaries which control the placement of labels and the colour of fills and lines. The location of these is specified by the fillColorDictionary and lineColorDictionary settings in the input files. Both files are automatically created and maintained by PhreePlot but can be edited to change their settings, e.g. the colours, or in the case of the line colour dictionary, the placement of the labels in a custom plot (repositioning labels in a predominance plot is achieved by editing the labels file).

The dictionaries are read in free format in the same way as the input files. The species name can contain blank characters if embedded in single or double quotes. If the labels appear to be chemical formula in Phreeqc format, sub- and superscripts are substituted as appropriate.

If the dictionaries are not present, then they will be automatically created with the name specified. Default names are 'fillColor.dat' and 'lineColor.dat'.

**Fill colours**

The fill colours are used for the area fills of predominance plots and the line colours are used for the lines in custom plots including fit plots. By default and in the absence of a fill colour dictionary, or when the species is absent from the dictionary, the colour fill is chosen from a sequence of pale colours (starting at sky1...). These default colours are overruled by settings in the fill colour dictionary. The fill colour dictionary contains a list of the 'species' names (as returned by Phreeqc) and their corresponding colours.

**Line colours and auto line colouring**

The line colour dictionary is used for custom and fit plots and contains a list of the label
name, x and y location (in graph coordinates) and colour of the various plotted lines. The colours are only used if the line or symbol is plotted, and by themselves do not dictate whether this is the case. For example, if a line has zero width or a symbol has zero size, it will not be plotted.

The line colour dictionary is used or created whenever a custom plot is made. If the file already exists, then it may be used to determine the line colour associated with a particular 'species' if it is present.

The line colour dictionary consists of seven columns of data containing: the label name, x- and y- plotting positions in graph coordinates and three colours, the first applies to the line colour, the second to the points colour and the third to the rim colour of filled symbols, for each label. The last column is the code number for the type of symbol used (pointType), 0 for no symbol. If a points colour has not been defined, it will be written as a blank field ("""). The x-position refers to the horizontal centre of the label and the y-position refers to the baseline.

UNDEFINED refers to an undefined ('not set') coordinate, e.g. when labels are not plotted. An empty string ("") for a colour will force automatic selection of the colour, if necessary. This is the default when the colour dictionary is rewritten and the when the point or line colour has not been set.

An example of a line colour dictionary is:

```
# label                   x            y       lines     points    rims symbol
"Cd+2"                  9.7065       48.624   green4       red4      ""       0
"Cd2OH+3"               9.3670      -3.5947  orange4      blue4      ""       1
"CdCl+"                 8.1980       2.2705    cyan4      black      ""       1
"CdCl2"                 8.5480      -3.5722 magenta4         nd      ""       0
```

Whether the settings in these dictionaries are used or overridden in a custom plot is determined by the useLineColorDictionary and changeColor settings. If useLineColorDictionary is 0, then the labels and line colours are either taken from the lineColor setting in one of the input files or are automatically generated. The line colour dictionary is ignored. If useLineColorDictionary is 1 or 2 then the line colour dictionary will be searched for the species being plotted and if found will use the colour (= 1 or 2) and label position (= 2) from the line colour dictionary. changeColor determines whether all the curves have the same base colour (= FALSE) or not (= TRUE). If changeColor is set to FALSE and useLineColorDictionary is 1, then the line colour dictionary will take precedence.

If the species colour is not found or if useLineColorDictionary is 0, then a line colour will be automatically selected according to the line colour sequence starting at the top of the list of colours in effect at the time. The default sequence is:

- red
- blue
- green
- orange
- cyan
- magenta
- brown
- sky
- purple
- gray
- yellow
- maroon
- lawn
- spring
- black
- maroon
- green
- blue
- red

Yellow is not plotted by default as a line or point colour as it is often hard to see (it will be included if explicitly given a colour density, e.g. yellow4). The sequence specified by this list is modified by the lineColor settings which promotes the given colour(s) to the top of the 15-long list. If the number of lineColor's is greater than 15, the list is extended to accommodate all of these. The default lineColor setting in pp.set is 'auto' which means the colour sequence will be red, blue, green, ... as above. If lineColor setting is set to blue, then the
sequence would be: blue, red, green, ... . If the number of colours required is greater than the length of the list, the list is recycled. If this is not wanted, use the colour dictionary to specify colours.

Autocolours only consider the basic colors (colors without a number). If the lineColor setting is red2, red6, black, the autocolour selected for the next (fourth) color would be blue4, the next unused colour in the sequence.

If a dataset is not plotted because there is no valid data in the plotting domain, then the corresponding auto-generated colour for that dataset is skipped.

If changeColor is set to false and useLineColorDictionary is 0 then the colour(s) in the lineColor list will be used first (unless this colour is 'auto'). If there are several subsets of data for the same variable the actual colour will rotate the colour density, 4, 6, 8, 2, 4..., e.g. red4, red6, red8, red2, red4,... providing the colour was defined without a density, i.e. as 'red' rather than 'red4'. If a density is given, this colour will always be used.

With calculationMethod 2 or 3 (replot), PhreePlot uses the dictionary coordinates if present otherwise it omits the label. Use this to change the colour or position of labels, lines and markers. Use calculationMethod 1 to regenerate a fully populated dictionary.

When there are several plots produced per run, for example due to use of the loop variable, there is an option of whether to restart the auto-generated colour sequence at the beginning for each plot, or whether to continue where the colour sequence ended on the previous plot. This is controlled by the restartColorSequence keyword. TRUE will restart it, FALSE will not.

If the trackSymbolSize is greater than zero and trackSymbolColor is not 'nd' then a coloured track symbol or anchor point is drawn at the point on the line to which any automatically positioned labels have been associated. This symbol is not drawn on replots.

The x,y position of the label placement is calculated by PhreePlot when calculationMethod 1 (calculate) although whether this position is actually used is determined by the various settings described above. The position is always read from the file when calculationMethod 2 or 3 (replot) is used.

The line colour dictionary is always updated with the latest species and colours at the end of each plot and the results written to the dictionary file at the end of each run.

If a record from a colour dictionary cannot be read properly, it is ignored.

Line widths are set by the lineWidth setting for each particular line (recycled as necessary). Negative line widths indicate dashed lines. The appearance of dashed lines is controlled by their respective ldashesPerInch and lineType settings.

Point colours

Points are coloured in the same way as lines except that the initial colour is defined with pointColor. If pointsSameColor is TRUE, then the points will always have the same colour as the lines if defined. If the line colour dictionary is present, the second colour setting can be used to override the automatic selections. Delete or rename the dictionary or set useLineColorDictionary to 0 if this is not wanted.

The interactions between the changeColor and pointsSameColor settings for the automatic selection of colours (useLineColorDictionary set to 0) are illustrated in Figure 7.4. These examples can be found in the autocolorn.ppi files found in the demo\Phreeqcloooping directory.

Filled symbols with a different rim colour

Normally points are plotted as simple symbols. However, there are six filled symbols which can have a separate rim colour. The filled colour is controlled by the points, pointSize, pointColor and changeColor settings.

Points will be plotted providing their size is greater than zero and their colour is not 'nd'. Rims
will be added if the rim colour defined by rimColor is not ‘nd’ and the line width of the rim defined by rimFactor is greater than zero. If changeColor is FALSE so that all the lines are the same colour (as set by lineColor), then pointColor is used for all the lines.

7.9.5 Directories for the colour dictionaries

If only the filename is given, then the specified file is sought following the usual search path rules (Section 2.4.6). If in doubt, give a full path name.

7.10 LABELLING

By default, all filled contour plots are automatically labelled and a legend produced to the right of the plot. An attempt is made to do this ‘nicely’ but this can be time-consuming especially when there are a large number of labels and when there is the potential for overlap. It is difficult to find a universal set of criteria that define good placement and it may be necessary to manually move labels using the appropriate label file or colour dictionary. In the case of contour plots, the labels are moved along the contour using the contourShiftLabel setting.

The algorithms employed at present are ‘experimental’ and rather inefficient. The time taken for labelling goes up roughly as the square of the number of labels and so can quickly become excessive. The effort taken and the number of possible label positions can be controlled using...
the \texttt{labelEffort} keyword (see below).

With lines only contour plots, the labelling is only done via the legend which takes the attributes of the contour lines.

7.10.1 Predominance plots

Labels are automatically centered at the centre of each field. There is some effort to try and resolve overlapping labels should any be identified and to ensure that labels are centred within their respective polygons. The present approach cannot deal with polygons with holes or islands in them. \texttt{calculationMethod} 2 uses the label coordinates from the labels file if present whereas \texttt{calculationMethod} 3 re-calculates the label coordinates. The latter is necessary to re-centre labels if the plotting domain has been changed.

Normally with \texttt{calculationMethod} 1, the label positions are automatically recalculated but it is possible to force \texttt{PhreePlot} to take the properties from the labels file (if present) by setting \texttt{useLabelsFile} \texttt{TRUE}. This enables the position or angle of a label to be preset.

7.10.2 Contour plots

Labels are automatically placed at the centre of the ‘longest, straightest’ part of a contour segment. If this is not satisfactory, the alternative option is to place all the labels at the centre of their contour – this is done by setting \texttt{contourOptions labelPosition=centre}.

It is also possible to shift contour labels individually. See the \texttt{contourShiftLabel} setting for details.

7.10.3 Custom plots

The method of simulated annealing is used to determine the position of label placement in custom plots. This is why ‘temperature’ appears in the output as the notional ‘temperature’ is gradually reduced allowing less freedom to roam randomly in search of a better set of label positions. This approach is experimental and can be very slow. The \texttt{labelEffort} keyword controls the amount of effort put into searching for good label placement. It only applies with \texttt{calculationMethod} 1 or 3.

The main objectives are to make the labels legible and their attribution unambiguous. The size of the labels is an important parameter (\texttt{labelSize}) since small labels obviously tend to overlap less than large labels. Set against this is the overall readability of the labels.

The \texttt{labelEffort} parameter is used as follows:

\begin{itemize}
  \item \texttt{labelEffort = 0} \quad \text{Labels are plotted roughly half-way along x axis; avoids the issue of label overlap etc altogether but rarely satisfactory}
  \item \texttt{labelEffort = 1} \quad \text{Minimal effort; this should take no more than a few seconds}
  \item \texttt{labelEffort = 2} \quad \text{Medium effort; slower}
  \item \texttt{labelEffort = 3} \quad \text{Much effort; much slower, can take many minutes – designed for batch processing where time is not such an important factor.}
\end{itemize}

The following objectives are sought when deciding where to place labels:

\begin{itemize}
  \item avoid labels overlapping with other labels
  \item avoid labels overlapping lines
  \item avoid labels being placed outside the plotting area
  \item place labels as close to the centre of the plot as possible
\end{itemize}

For custom plots, a second parameter, if present, provides an upper limit to the time taken (in
sec) for optimizing label placement.

These objectives will conflict to some extent and PhreePlot tries to find a reasonable compromise. This is computationally demanding and the optimum solution may not be found in the time available.

Interrupting this process using the Esc key will exit gracefully while retaining the best label positions found up to that point.

In custom plots, each label is anchored to one of the calculated points. This anchor is shown by a small filled circle (by default red) which is controlled by the track symbol (trackSymbol-Color and trackSymbolSize).

7.11 REPLOTTING WITHOUT RECALCULATING

7.11.1 The ‘replot’ option

It is often necessary to ‘fine tune’ the appearance of existing plots. The calculationMethod parameter controls the extent of recalculating: 1 = calculate from scratch and plot; 2 = just replot; 3 = reprocess data, relabel and replot but do not re-speciate. This enables the appearance of a plot to be changed without repeating the underlying calculations.

1 does speciation calculations; 2 and 3 do not. The replot options make use of the output files produced during an earlier calculation. These files must be present.

It is possible to make the following changes during a replot:

- change the base font
- change the size and colour of all text, symbols and lines
- add, remove or change any extra text, symbols and lines that are specified by the ‘extra-Text’ and ‘extraSymbolsLines’ files
- implement any changes made to the label file or colour dictionaries including deletion or repositioning of labels
- change from a native y-axis scale to some other scale in predominance plots
- change the sign on the x- or y axis
- change which graphic output files are produced (pdf, png etc).
- change from a series of single page graphic files to a multipage file and vice versa.

Unpredictable results will occur if you:

- change anything to do with the scope of the calculations involved, e.g. xmin, xmax, ymin, ymax, looping, number and type of main species, fitting parameters, extent of simplification
- change the resolution
- change the thermodynamic database used.

All the required files must be present and in the correct format for replotting to work. If they are not, regenerate them from scratch (calculationMethod = 1). The following output files are required for replotting:

- htl vec, pol, lab (calculationMethod = 2)
- pts vec, lab (calculationMethod = 3)
- grid pol, lab (calculationMethod = 2)
- trk (calculationMethod = 3)
- contour vec, pol (calculationMethod = 2)
out, vec, pol (calculationMethod = 3)

fit or simulate out

custom out

species out

Adobe Reader locks open files so a new pdf file cannot be recreated while a file with the same name is already open. Close it first. This limitation does not apply to files opened by GSview.

It is possible to use the replot option to enable PhreePlot to make a plot of any data in a user-derived text file providing the file format and name are correct, i.e. data in regular columns with the first line containing the labels. The filename should be the base filename with the extension 'out'. The calculationType should be 'custom' and mainSpecies = '', say. Alternatively, a data file for making a custom plot can be added with the extradat keyword. This can be used for adding data to predominance plots too.

The calculationType should be 'custom' and mainSpecies = ‘’.

7.11.2 The 'reprocess and replot' option for predominance plots

The reprocess and replot option (calculationMethod = 3) goes back one stage further than simple replotting and starts with the stored output from the speciation calculations. For the h1 calculation type this is the pts file rather than the vec file. With 'grid' plots, this is the 'trk' file and with 'grids' plot this is the 'out' file. In all cases, the polygons are re-assembled and the label positions recalculated before replotting. In the case of the h1 calculation type, this enables the degree of simplification to be changed without recalculating the chemistry.

In the case of the 'grid' and 'grids' plots, if the speciation calculations have been terminated early for some reason, the track and out files, respectively, will not be fully populated and it may not be possible to complete the plot. This can happen when Esc has been used to terminate calculations early or if PhreePlot has crashed part way through the speciation calculations, for example if the operating system has run out of virtual memory.

Restarting 'grids' plots with calculationMethod = 3 will attempt to fill incomplete 'out' files thus preserving the effort of earlier calculations.

7.12 ADDING EXTRA LINES, SYMBOLS AND TEXT

Lines and symbols can be added to predominance plots using the 'lines' and 'points' method used by 'custom' plots but if this is not appropriate, or if extra text, symbols or lines need to be added to any plot, these can be specified in separate files. Tags can also be used anywhere in these extraSymbolsLines or extraText files.

7.12.1 Lines and symbols

Points on plots are plotted with symbols. Extra symbols can be added to a custom plot or predominance plot with an extraSymbolsLines file. This file has the format:

\[
\text{plotnumber}, x, y, [lw, [linecol, [isymb, [sizesymb, [symcol, [rimcol, [rimfactor, [linetype, [dashesperinch]]]]]]]]]]
\]

See extraSymbolsLines for more details. A full list of the symbol numbers and their names is given in Appendix A3 and Figure 7.5. The default is to draw a black line but most combinations of line and symbol size, colour and type are possible by explicitly specifying them. Once set, these properties stay in force until changed.

The first ten symbols are centered symbols:

1 = filled circle
2 = filled square
3 = filled triangle
Symbols, dingbats and lines

Figure 7.5. The symbols available in the native, Symbols and ZapfDingbats fonts and their corresponding `isymb` codes. This diagram can be generated from the `demo\symbols\symbols.ppi` file.

4 = filled upside down triangle  
5 = filled diamond  
6 = filled octagon  
7 = open circle  
8 = plus  
9 = multiply  
10 = star

The six 'filled' symbols above can each have a separate rim colour and associated rim thickness. It is therefore possible to plot an open circle using the filled circle code but specifying that the point or symbol colour is 'nd'. Specifying the symbol colour as 'white' will have a similar effect on white 'paper' except that it will remove any underlying features.

7.12.2 Text

Additional text can be placed inside or outside the plot area (as defined by the axes) using an extra file, the These are specified in the 'extra text' file and defined in detail under the extra-Text keyword. This file has the format:

```
plotnumber,x,y,text,[height,[colour,[angle,[justify,[digits,[font]]]]]]
```

`plotnumber` is the plot number (auto for 'all'), `x` and `y` are the x- and y-coordinates of the anchor position, and `text` is the single text string (usually embedded in quotes). This text string can contain tags from the tag dictionary or special plotting tags such as subscript/superscript. There are also other special tags which can be used.

The remaining parameters are optional. Although these are optional it is necessary to keep the parameters in order, i.e. you can omit `font` and include the rest but cannot omit `colour` and keep `angle, justify, digits` and `font`. If necessary specify all the parameters explicitly.

Defaults for the optional parameters if left blank are:
Plotting basics

height = legendTextSize

colour = 'black'

angle = 0 (degrees)

justify = 0

digits = -3 (if numbers are being printed, this provides some control over the format: default is three figures/decimal places depending on the type of number involved; the negative sign indicates that trailing zeros will be removed)

font = the font (name or number, 1-44), or the current font if undefined.

plotnumber is the plot number (starts at 1) for which the text applies. 'auto' means all plots. The plot number is the sequential number of the plot and is always printed on the first line of the info block if that whole block is printed. For more detail on the numbering see the extraText summary. Text is truncated to 200 characters. This includes any text used to specify tags (see below). Enclose the text within paired single or double quotes if the text contains a delimiter (space, comma or tab). This will keep the text as a single string entity.

x and y can be made dynamic by making use of the <pxmin> etc tags which are set after the plotting limits have been determined., e.g. first set

numericTags

<px> = "<pxmin>+0.3*(<pxmax>-<pxmin>)"

then use <px> in place of say x in the extraText file.

The y-position always refers to the main y axis. If a point needs to refer to the 2y axis, then the corresponding y value must be calculated separately. This can be done using a numeric tag expression like:

<p2y> = "(<2y>-<p2ymin>)/(<p2ymax>-<p2ymin>)*(<pymax>-<pymin>) + <pymin>"

where <2y> is the wanted position on the 2y axis and should be defined using numericTags. The <p2y> tag can then be used in an extraText or extraSymbolsLines file.

When printing, leading tabs are replaced with three spaces and subsequent tags replaced by single spaces. Multiple spaces are also reduced to a single space. Spaces might appear with unpredictable length with proportional fonts.

justify: refers to the justification of the string (0= left, 1= centre, 2= right) irrespective of the angle, i.e. as if rotated about the appropriate character, so justify=2 with angle=180 will rotate around the last character whereas justify=0 will rotate about the first character. The x and y coordinates therefore by default specify the left-hand baseline (bottom left-hand corner) of the text. x defines the left-hand edge; y defines the baseline of the text string ('g' falls below the baseline). Where there is more than one line because of the use of embedded line breaks, <br>, y refers to the first line. If the text goes off the bottom of the screen, increase yoffset to reposition the plot higher, or increase the page size. Remember the page size of the printing or viewing device controls what part of the plot will actually be seen.

digits: specifies the number of digits to print when a numeric tag is substituted by a value. It is specified by an integer, n. n gives the number of digits printed after the decimal point (valid range 0>=n>=16). 0 prints the nearest integer. If a negative integer is given, trailing zeros are removed. If the absolute value of the number is less than 1e-3, it is printed in scientific (x.xEee) format. This format applies to all numeric tags in the text string (default n = -3).

font: The font is either specified by a defined font name (see fonts.dat, if present) or by a font number, numbered consecutively across and down the fonts.dat table, if present, e.g. Times-Roman or 17. If the font or font number is unrecognised, the current font is used if defined else it reverts to the number 1 (Helvetica) font.
A special piece of text for copying all or part of the text from the input file(s) next to a plot is:

- \texttt{<input>}
  - prints all lines from the input file(s).
- \texttt{"<input:str1>"}
  - prints all lines from the input file(s) starting at the first line containing \texttt{str1} to the last line of the CHEMISTRY section.
- \texttt{"<input:str1,str2>"}
  - prints all lines from the input file(s) starting at the first line containing \texttt{str1} to the first line containing \texttt{str2} inclusive.

\texttt{str1} and \texttt{str2} are ASCII character strings. Case is significant for both \texttt{str1} and \texttt{str2}. Use the quotes for the second form otherwise the comma will be parsed and \texttt{str2} will be set to \texttt{col} and then the text ignored completely. Justification, if present, is ignored. The input searched is the final concatenated input from all input files with comments and blank lines removed, include files expanded but tags not substituted.

The printed text will be left justified, vertically aligned with the top of the text at \texttt{ypos} and starting at \texttt{x}, \texttt{x}, \texttt{y} may be outside the plot axes. \texttt{mjust} is ignored – the text is always left justified. Multiple spaces and tabs are replaced by single spaces except any leading spaces (indentation) are preserved. A leading tab is replaced by three spaces. Blank lines are omitted.

The first example will print the entire input file. This is based on the input file as input - not the expanded input file that is created after expanding any \texttt{include files}.

The second example will print the input file starting from the first line containing \texttt{str1} and finishing at the first line containing \texttt{str2}. If \texttt{str1=\"\"}, then printing starts at the first line; if \texttt{str1} is non-blank and \texttt{str2=\"\"}, printing continues to the last line. If \texttt{str1} is found but not \texttt{str2}, then the text is printed from \texttt{str1} to the end of file but if \texttt{str2} is found and \texttt{str1} is not, then nothing will be printed.

\texttt{<br>} is not recognised within this tag. If \texttt{str1=\texttt{str2}}, only that line is printed.

Angle is the angle in degrees from the horizontal, rotating clockwise.

The size and \texttt{colour} of the text is either given on the \texttt{<input>} line or by default takes on the values of \texttt{legendTextSize} and ‘black’, respectively. The legend can be turned off by setting \texttt{labelColor} to ‘nd’.

If \texttt{info(1)} is set to ‘nd’ then this input text will not be printed. This allows clean plots to be produced without editing the individual \texttt{extraText} files, possibly by using the \texttt{override.set} file.

\texttt{<loop> or <loop...>}

Substitutes the current numeric value of the appropriate \texttt{loop} variable. The primary loop variable is \texttt{<loop>} which if a regular sequence, can either be defined in an input file using the \texttt{loopMin}, \texttt{loopMax}, \texttt{loopInt} and \texttt{loopLogVar} settings.

If the sequence of loop values is irregular or if more than one variable needs to be set on each iteration, use the loop file approach instead (Section 4.6.2). \texttt{<loop>} is equivalent to \texttt{<loop1>}, the first column of numbers in the loop file. Successive numeric columns are set to the \texttt{<loop2>}, \texttt{<loop3>} etc tags if there is no header with column names present. If a header row is present, this is used to generate the tag names.

The loop variable is printed as an integer if it is an integer otherwise it is printed in floating point format. No extra spaces are added to the number. The value of the loop variable used is the value at the time of plotting.

\texttt{<legend>}

Inserts the legend for custom plots at \texttt{x}, \texttt{y} rather than in its normal position to the right of the plot. If found, all other text in the string is ignored. Height and \texttt{colour} can be included but if
absent, the defaults are:

- text height = the **legendTextSize** setting
- colour = black.

e.g.

```
auto 4 -12 "<b>Key</b><legend>" 2 blue
```

will insert the top left-hand corner of the legend box at approximately (4, -12) based on the plot scale. Legend lines will be left-justified at 4 and the top line's baseline placed at -12. Symbols, if plotted, are justified slightly to the right of this. If a legend box is drawn, the top left-hand corner will be at (4, -12).

Any text before `<legend>` is treated as the legend title and will override the **legendTitle** setting. The legend title will therefore be 'Key' in the above example. If no legend is defined in the `<legend>` line, **legendTitle** will be used.

Text after `<legend>` is ignored. Text tags such as `<b>` will be ignored if split by a `<br>`.

This option allows the legend to be moved inside the plot area, or by using out-of-range x and/or y coordinates, removed completely from the plot.

**<mainspecies>**

 Inserts the name of the current main species into the text string, e.g. "<mainspecies>" would substitute "Fe" if that was the main species.

### 7.12.3 Formatting numbers in plots – varying the number of significant figures displayed

Tags can be used to write numbers to a plot file when numeric tags are placed within a text string that is to be plotted. This applies to tags in the upper (**PhreePlot**) section of an input file.

The default representation for floating point numbers in plotted output is to round them to 3 significant figures and to remove trailing zeros. A floating point format (e.g. 7.98) is used when 3 significant can be maintained but the exponent (E) format is used for larger and smaller numbers.

It can be useful to be able to control the number of significant figures displayed explicitly, e.g. 6.58928137 could be displayed as 6.59 (3 significant figures) or 6.589281 (7 significant figures). This can be controlled by appending an underscore to the number or tag followed by the number of significant figures required, e.g. 6.58928137_3. The number of significant fig-

<table>
<thead>
<tr>
<th>Text string</th>
<th>Appearance in plot</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;6.58928137&quot;</td>
<td>6.59</td>
</tr>
<tr>
<td>&quot;6.58928137_4&quot;</td>
<td>6.589</td>
</tr>
<tr>
<td>&quot;6.58928137S6&quot;</td>
<td>6.58928E+00</td>
</tr>
<tr>
<td>&quot;125734.123_4&quot;</td>
<td>125700</td>
</tr>
<tr>
<td>6.100</td>
<td>6.1</td>
</tr>
<tr>
<td>&quot;6.1_3&quot;</td>
<td>6.10</td>
</tr>
<tr>
<td>&quot;6.1-3&quot;</td>
<td>6.1</td>
</tr>
<tr>
<td>6.153</td>
<td>6.10E+00</td>
</tr>
<tr>
<td>&quot;6.1_3&quot;</td>
<td>6.1_3</td>
</tr>
<tr>
<td><code>&lt;a&gt;</code> where <code>&lt;a&gt;=1.23456789e-20</code></td>
<td>0</td>
</tr>
<tr>
<td><code>&lt;a&gt;_5</code></td>
<td>1.2346E-20</td>
</tr>
<tr>
<td><code>&lt;bb&gt;</code> where <code>&lt;bb&gt;=1.23456789e20</code></td>
<td>-1.23E+20</td>
</tr>
<tr>
<td><code>&lt;bb&gt;_6</code></td>
<td>-1.23457E+20</td>
</tr>
<tr>
<td><code>&lt;bb&gt;$6</code></td>
<td>-1.23457E+20</td>
</tr>
</tbody>
</table>
ures \( \text{sigfigs} \) can vary from 1 to 16 (although normally only 14 are significant). A negative value for the number of significant figures will format to \( \text{ABS(sigfigs)} \) and then remove trailing zeros.

Very large \( \text{ABS}(x) > 1e15 \) numbers will always default to the exponent format. Very small numbers, numbers less than an absolute value of 1e-15, are plotted as 0.

If an exponent format is wanted, use $ instead of _, e.g. \( 1.3463$3 will plot as \( 1.35E+00 \) and \( 1.34600$-5 \) as \( 1.346E+00 \).

This approach is especially useful for formatting numeric strings derived from a tag in an extraText file or in a plot or axis title. Some examples are shown in Table 7.1. It is used to format fit output in the extraText file of the demo\kineticsSi\kineticsSifit1.ppi demo file.

### 7.12.4 Making fancy plots

There is a limited capability in PhreePlot to add additional features to a plot (see Section 8.9). Firstly it is possible to add text, lines or symbols using the extraText and extraSymbolsLines keywords.

Secondly it is possible to add lines or points to a predominance or contour plot by including the lines or points keywords (but not the 2y equivalents) and making sure that the data are read in correctly. This could include the following additions to your input file:

```plaintext
extradat filename.dat # load this file with a header line giving column names
customxcolumn xcol # make this column the x-variable
lines ycol # make this column a y-variable (there can be several)
points col # ibid
linecolor red
pointcolor blue
```

The file containing the additional data could have been generated during the main plotting stage or read in from an external file. Several ppi files can be run and then the combined data plotted in some way. Batch files can orchestrate this.

Thirdly, it is possible to overlay one or more plots on top of each other using the overlay keyword.

Finally it is possible to change the plot in ways that PhreePlot cannot do by editing one of the plot files using an interactive image editor such as Inkscape. It would also be possible to run PhreePlot from within another program such as Python or R and use the output in one of their plots.
8 Predominance plots

8.1 SETTING UP A FILE TO CALCULATE A PREDOMINANCE DIAGRAM

PhreePlot offers a flexible approach to drawing predominance plot. The criteria for selection as a predominant species are user-defined in USER_PUNCH blocks (often in inc files) that are easily edited. These must simply return a series of name-value pairs and its the pair with the greatest value that is considered predominant. The values returned could be simply species concentrations whether dissolved, solid, surface or gas. They could be based just on aqueous species, they could sum all adsorbed species and treat the total as a ‘super’ species, or they might take some other factor, such as redox state, into account.

In order to draw a predominance diagram, PhreePlot expects that the last simulation executed will return the predominant species in the format expected. There can be other pre-loop and main loop simulations but these must precede this final simulation.

From that point onwards, PhreePlot treats all data in the same way and attempts to draw a 2D diagram showing the predominance fields. In fact, PhreePlot contains two distinct methods for drawing predominance plots – the ‘grid’ and ‘hunt and track’ approaches – and these differ in the way that they sample the 2D data space to identify the field boundaries.

Another approach is to contour some variable such as the total dissolved concentration of an element or species - also possible in PhreePlot - but the predominance diagram remains a beguilingly simple way of describing the chemical behaviour of complex systems.

The results should be quite close to those obtained using traditional analytical approaches such as used by the Geochemist’s Workbench (Bethke, 2005). However, they will not be exactly the same since the simplifying conditions required in the analytical approach are often unrealistic and can be difficult to achieve in practice. The full speciation approach requires that all parts of the domain under study be accessible via a plausible and specifiable reaction path. Activities are seen to reflect all the interactions in the system rather than being simply imposed on the system. Achieving constant activities along a boundary usually requires variable quantities of an element to be present whereas more realistic systems have constant total concentrations of elements and variable activities.

The results from PhreePlot should be similar to other programs that produce predominance diagrams using a rigorous approach to speciation. HYDRA/Medusa is one such program that is free to use. It produces a number of geochemical diagrams including predominance plots using the HALTAFALL (or SOLGASWATER) speciation program and a simple but effective user interface (including a Java version). The domain of interest is sampled on a regular grid. One noticeable feature of the program is that when drawing predominance diagrams, it traces the speciation above and below the ‘water limits’ with ease. This is a region where Phreeqc fails because there is simply little or no aqueous solution left. The difference here seems to be that Phreeqc keeps a mass balance on the water whereas Medusa apparently does not. This means that Medusa not only ignores the decomposition of water at the extremes of pe but that it presumably does not account for the loss of water in mineral (e.g. gypsum) formation or due to forced evaporation. However, given this limitation, the simplification is particularly useful when drawing traditional pe(Eh)-pH predominance diagrams since water is unstable in as much as half of the domain. Adding the water limits to the diagram shows the user the region where water decomposition occurs.
8.1.1 The ‘grid’ and ‘ht1’ approaches

**Grid approach**

PhreePlot is able to calculate predominance diagrams based on a full speciation of the system using the grid and ht1 algorithms. The grid approach is a direct search approach that calculates speciation on a grid while the ht1 approach finds and tracks the field boundaries. If the total quantity is kept constant, then the concentrations of dominant species at boundaries will be close to half the total concentration while at triple points will be close to one third of the total concentration.

The grid method is a ‘brute force’ method in that PhreePlot simply calculates the speciation at each point on a rectangular grid and reports the dominant species. The range and spacing of the grid points is determined by the $x_{min}$, $x_{max}$, $y_{min}$, $y_{max}$ and resolution parameters. This method requires $n_{res^2}$ speciation calculations plus any pre-loop calculations, e.g. initial solution calculations.

The matrix of predominant species can be rendered directly by colouring each species differently but this does not identify field boundaries and results in rather large image files. PhreePlot uses a ‘pixel aggregation’ technique to identify the boundaries. This enables the fields to be polygon-filled with colour and so avoids pixelation of the generated image This results in better rendering of the plots and much smaller file sizes.

One or more main species can be specified. With the ‘grid’ approach, a new set of speciation calculations is repeated for each main species even though the grid is the same. With the ‘grids’ approach, the speciation calculations are made for all main species in one pass and the results written to individual ‘out’ files in the normal way. Progress during the speciation calculations is shown by a series of dots on the screen. Each dot represents 10 speciation calculations plus any pre-loop calculations, e.g. initial solution calculations.

This approach needs different user_punch statements to export the data since the ‘grids’ option requires that the full speciation for all elements be returned and written to the appropriate ‘out’ files in one call to user_punch. ‘grid’ uses the ‘ht1.inc’ file or similar, while ‘grids’ uses the ‘grids.inc’ file or similar.

It is possible to add a z-loop to ‘grid’ and ‘grids’ plots. Normally the z-loop is inside the main species loop but in ‘grids’ plots, the results of all main species calculations are returned in one speciation calculation and so in this case the z-loop must be outside the main species loop. This special case is intercepted when the first z-loop is executed. This is reflected in the output in the log file, and affects the calculation of the printed timings of individual plots. These timings are estimated by dividing the total speciation time by the number of main species to give an approximate time taken per main species.

The ‘grids’ approach carries considerably more overheads but can be faster for two or more main species especially when the speciation calculations are themselves relatively slow. See the demo\grids directory for an example.

**’Hunt and track’ approach**

The ht1 method uses a ‘hunt and track’ approach to find and track the field boundaries. It is based on the assumption that all such boundaries can be reached from the domain (‘axis’) boundaries, i.e. that there are no ‘islands’. In fact, islands can occur (see Example 43) and so while this method is usually quicker than the grid approach, it is not so reliable and so the results of the ht1 approach should always be compared against the grid approach.

The ht1 approach first works along the domain boundaries looking for a change in the dominant species. This is the ‘hunting’ mode. Once it has found a change-over on the boundary, it tracks internally along it. During this tracking, it only makes evaluations (speciation calculations) on a fixed grid defined by the same parameters that the grid method uses. It bounces along the boundary keeping track of where changes in the dominant species occur and noting
where triple points - the intersection of three equally dominant species - occur. Where possible, more precise boundary positions are estimated by interpolation along a cell edge using the logarithm of the dominant and subdominant concentrations.

It is possible to define constraints that override the normal predominance criteria. The traditional ‘water limits’ are commonly applied in presenting predominance diagrams. The _ht1.inc_ code provides a check to see if these have been exceeded and _PhreePlot_ elevates them to the top of the list if they have. Note that the limits specified in _ht1.inc_ are for a pressure of >1 atm H₂(g) on the reducing side and >1 atm O₂(g) on the oxidising side. These limits can be easily changed by editing _ht1.inc_ or equivalent file. A check is also made on the methane partial pressure since this can be high in strongly reducing, carbon-containing systems.

The overall strategy is best appreciated by examining the _ht1.inc_ and _ht1c.inc_ files which provide generic pieces of _Phreeqc_ code for returning the predominant species. These are used by both the hunt and track and grid methods. _ht1.inc_ is the simpler of the two scripts in that it treats all adsorbed species as distinct species whereas _ht1c.inc_ combines all adsorbed species into a single ‘super’ species for the purposes of counting and display.

_PhreePlot_ expects the _SELECTED_OUTPUT_ to have a specific format in order to be able to draw a predominance diagram. The list of data required consists of five different blocks of data, each of which can contain a variable number of species name-number (often a concentration) pairs. The counts for each of these blocks is given by five numbers at the end of the list. These can be zero if no pairs are returned. This ensures that _PhreePlot_ knows how to read the data list returned. All the data are written by _USER_PUNCH_ statements in _ht1.inc_. Do not write any output using the _SELECTED_OUTPUT_ keyword data block as this will contravene the expected structure and will lead to a mismatch in the expected and found number of columns received.

The structure of the _ht1.inc_ file is summarised in Figure 8.1.

<table>
<thead>
<tr>
<th>nout1</th>
<th>nout2</th>
<th>nout3</th>
<th>nout4</th>
<th>nout5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominant species</td>
<td>Dominant minerals</td>
<td>Constraints</td>
<td>Carry variables</td>
<td>5 system variables</td>
</tr>
</tbody>
</table>

Figure 8.1. Data structure expected to be returned to _PhreePlot_ in the selected output in order to calculate a predominance diagram using the _ht1_ and grid plot types.

_ht1c.inc_ also gives the option of calculating either a predominance or stability diagram (Section 8.1.5). In a stability diagram, mineral species assume predominance over solution species no matter what their relative concentrations. This is done by commenting out either line 20 or 30 (using a # or REM statement).

The script defines the five blocks of species which must be returned by _PUNCHing_ them. These blocks are:

1. the three dominant species (solution, mineral, gas or adsorbed), (normally _nout1<=3_ although any number can be returned and _PhreePlot_ will sort them to find the largest)

2. the three dominant mineral species, i.e. those which account for the largest concentrations of the main species (_nout2<=3_)

3. any constraints: these will override all other considerations and if the constraint is true will force the species to be treated as the dominant one. Limits on the partial pressures of H₂(g), O₂(g) and CH₄(g) are usually included (_nout3=3_ usually) and can be used to plot the usual ‘water limits’. Other constraints are possible. See the carbonate example with constraints on the total carbonate in the system (Example 25)

4. any ‘carry’ variables (_nout4_)

5. the ‘system’ variables - pH, pe, the log partial pressures of O₂(g) and H₂(g) and temperature (°C) (always these five pairs of values in that order).
The final items punched are the five counts, \texttt{nout1-nout5}. \texttt{PhreePlot} receives the five groups of species, their values and their counts as one long list. The counts tell \texttt{PhreePlot} how to read the list.

The minimum output required to prepare a predominance diagram is one dominant species (\texttt{nout1 = 1}) and five system variables (\texttt{nout5 = 5}).

‘Carry’ variables are user-defined numeric variables that are wanted to be output but which play no part in the calculation of the predominance diagram. They are sent as name-value pairs. For example, the script \texttt{htminerals.inc} script can be used to list all the minerals that precipitate somewhere in a predominance diagram. This can be used to reduce the mineral phases considered during the speciation calculations. In this example, a list of the summary statistics (count, minimum, mean and maximum value) for each species output as a variable is listed in the log file provided the out file has been turned on (\texttt{out T}). The ‘carry’ variables are written to the out and track files and automatically added to the tag dictionary. The track file uses the species names found on the first iteration as the column heading and so you must ensure that the same ordered list is produced on all subsequent iterations.

The \texttt{ht1.inc} and \texttt{htlc.inc} scripts use the \texttt{SYS()} function to return a list of the concentration of all the species of the element of interest, the so-called ‘main species’. This list is returned from \texttt{Phreeqc} pre-sorted in decreasing amount of the main species element. This is not necessarily in terms of decreasing species concentrations where polynuclear species are involved.

The top three of these are sent back to \texttt{PhreePlot} as block 1 via the \texttt{SELECTED_OUTPUT} ‘file’ by \texttt{PUNCH}ing them in the sequence:

\texttt{name1 concn1 name2 concn2 name3 concn3}

where \texttt{name1} has the highest concentration and \texttt{name3} the lowest. If less than three species exist, then either one or two is returned depending on the number available (the predominance diagram is trivial if there is only one species).

The remaining output is written to the selected output ‘file’ in a similar manner, i.e. always as a species name followed by a numeric value, usually a concentration.

An example where a large number of carry variables are written at each calculated point of a predominance plot is given in \texttt{demo/grid/gridhfo_with_carry.ppi}. Here the carry variables are contained within a separate file, \texttt{carry.inc}, and give a detailed breakdown of the system in terms of species concentrations for the ‘main species’ and the minerals and gases present at each point. It is fairly straightforward to tailor this file to your particular needs.

### 8.1.2 Using the \texttt{ht1.inc} code to return the dominant species

A predominance diagram is most easily calculated using the \texttt{ht1.inc} code described above. A simple example is given below for preparing a pe-pH diagram for Fe (see Example 3).

The \texttt{CHEMISTRY} section starts with the include file (its position within a given simulation is unimportant) and then has a \texttt{SOLUTION} keyword block to define the initial conditions - the total amount of each element present in the system. This is constant and so can be setup in a separate simulation.

This is followed by a second simulation which includes an \texttt{EQUILIBRIUM_PHASES} keyword block. This provides the mechanism for traversing the x- and y-axes, and for defining mineral phases that may precipitate (or dissolve). This means that the two tags, \texttt{<x_axis>} and \texttt{<y_axis>} have to be present, either explicitly or implicitly. Note that the x axis is controlled by \texttt{Fix_H+} (defined in \texttt{ht1.inc}) and is therefore the pH. The initial conditions include the total concentration of Fe$^{3+}$.

The pH is adjusted by adding (or subtracting) NaOH. Note that the initial pH is low (and is preferably set to less than the minimum pH required on the x axis) The y axis is linked to a fixed partial pressure of O$_2$(g) supplied by an external reservoir of O$_2$ (10 mol). The plot created is therefore one of log fO$_2$(g) vs pH. This runs faster than using a ‘\texttt{Fix_pe}’ approach.
Since the pe is always carried in the calculations and in the output files, a switch of y scales can be done either when the plot is being generated or afterwards by replotting using the yscale setting (pe, Eh or mV).

CHEMISTRY
include 'ht1.inc'
SOLUTION 1
  pH       1.8
  units   mol/kgw
  Fe(3)   1e-2
  Na      1e-1
  Cl      1e-1
END

USE SOLUTION 1
EQUILIBRIUM_PHASES 1
  Fix_H+ <-x_axis> NaOH 10
  -force_equality true
  O2(g)   <-y_axis> 1
  Fe(OH)3(a) 0 0
END

Note the negative sign directly in front of the <x_axis> tag. This enables the x axis to be specified in directly terms of pH. This works because the substitution of the tag is done without introducing extra spaces around the tag. This approach works providing the value of <x_axis> does not become negative. The alternative is to use a numericTags relation to define a new tag as -<x_axis>.

In summary, you have to supply the initial solution conditions (total concentrations), the means of traversing the x- and y-axes by including the <x_axis> and <y_axis> tags, and you must include any phases that might precipitate in an EQUILIBRIUM_PHASES keyword block.

It is possible to modify the ht1.inc file to alter the logic by which a species is promoted to the top. For example, the minstab1.inc and minstab2.inc files are designed to highlight the most abundant or most likely minerals present. It is also possible to alter the way that species names are presented, for example, for minerals, by adding the mineral formula underneath the mineral name (e.g. see for an example of this) or instead of the mineral name.

8.1.3 Problems with the ‘hunt and track’ approach – ‘unclosed polygons’

Although numerical problems in tracking were initially thought to be the largest likely problem with the ‘hunt and track’ approach, the greatest difficulty in practice has been marrying up the ‘hunt’ and ‘track’ parts of the procedure at the boundaries of the plot. If a field boundary coincides with, or is very close to, a boundary of the calculation domain then this can result in lots of ‘bubbles’ and ultimately will fail because of ‘unclosed polygons’. All polygons must be perfectly closed to enable the colouring and labelling.

Although PhreePlot may restart automatically with a different resolution, thereby potentially shifting the field boundary away from the domain boundary, this is no guarantee of success. Increasing the size of the calculation domain (xmin, ymin, ...) is a better solution – the plotting domain (pxmin, pymin, ...) does not have to coincide with the calculation domain.

It is best to avoid field boundaries being very close to the domain boundaries where possible. A common problem is when the upper water limit set at exactly 1 atm O2(g) in ht1.inc and then ymax for (log O2(g)) is set to 0. This places the boundary for calculations exactly on a field boundary – in this case, the artificial boundary created by the upper water limit. It is better to either set the upper limit (ymax) of the calculation domain to be somewhat greater, say 1 or more, or to alter the water limit slightly. This is why the default water limit in ht1.inc is set at -0.05 or 0.9 atm - far enough away from 0 to not cause boundary problems with ymax = 0 but close enough to 1 atm to be able to label the boundary as such. The alternative is to set the water limit at exactly 1 atm but to set ymax somewhat higher, say ymax = 1, for example.
8.1.4 Optimising the calculation efficiency

Minimizing the size of the tag dictionary can significantly increase the speed of computations and therefore unused tags should be removed wherever possible. Minimizing the number of mineral phases considered will also improve the calculation speed.

8.1.5 Predominance vs mineral stability diagrams

A predominance diagram always shows the predominant species (unless overridden by a constraint). The predominant species is the species accounting for the largest number of moles of the main species. A stability diagram is similar except that if a mineral species is present, it overrides all solution species in precedence no matter what their concentration. The diagrams are not too different since minerals are often quite insoluble and often dominate the overall speciation not far from solution-mineral boundaries. If more than one mineral species is present, the most abundant mineral (in terms of moles of the main species) takes precedence.

The choice of calculating predominance and mineral stability diagrams can be readily implemented by including the `ht1c.inc` code in the `Phreeqc` code.

If a mineral stability diagram containing only mineral species is wanted then the main species should be set to the special value ‘minerals’ and most importantly, the `minstab1.inc` include file used instead of `ht1.inc` (Example 54). Precedence is then calculated in terms of the most abundant mineral species (in terms of moles of mineral), if present. Precedence in this case can therefore depend on the mineral formula used.

8.1.6 Using `ht1minerals.inc` to determine the minerals present

Many of the minerals in a database will never actually become stable anywhere in a predominance plot even where all the necessary components are present in the system. They may never be anywhere near saturation or may be protected from precipitating by a closely-related but more stable mineral.

These non-precipitating minerals can be excluded from the speciation calculations without any loss of accuracy. This is significant since excluding these unnecessary minerals from the `EQUILIBRIUM_PHASES` data block can speed up calculations considerably.

The include file, `ht1minerals.inc`, is similar to `ht1.inc` except that it also writes as ‘carry’ variables all minerals that have a saturation index of zero or above for each speciation calculation, i.e. these are the species that are either saturated or supersaturated.

`PhreePlot` automatically analyses all the ‘carry’ variables in the out file, if present, and sends summary statistics to the log file. In this case, there will be a table of all the minerals species for which $SI=0$ at some point. Looking at the ‘max’ column in this table gives the required information. All species which have a max value of 0 (or very close to it) have precipitated and must have already been included in `EQUILIBRIUM_PHASES`. Those with a max value significantly greater than 0 might precipitate if included and are therefore potentially relevant.

The supersaturated minerals will be those that exist in the database and which might have precipitated but have not done so since they have not been included in the `EQUILIBRIUM_PHASES` data block. Re-running with all of these minerals included will enable a minimum set of minerals to be identified, i.e. those that then have a max value close to zero.

Using `ht1minerals.inc` therefore identifies all minerals in a database that may be relevant to the present system – as well as those that are definitely not. This includes all minerals, not just those containing the main species.

This option requires that the out file is actually produced – it is not by default. This is achieved by setting ‘out T’ in the input file. Other than that just replace ‘ht1.inc’ by ‘`ht1minerals.inc`’ in the input file.

Since this approach only analyses the speciation calculations actually carried out, it is necessary to cover the whole domain of interest fairly systematically. This is best done with the ‘grid’
method.
An alternative approach is to automatically include all possible minerals in the EQUILIBRIUM_PHASES data block. This is the approach taken by the htlallminerals.inc include file and the demo\minstab\allminerals.ppi example.

8.2 THE 'GRID' APPROACH

The grid approach (Example 1) is simple and reliable but relatively slow especially at high resolutions. One advantage is that it does not rely on being able to access all internal boundaries from the domain boundaries nor does it require relatively noise-free boundaries. The ht1 approach requires both of these conditions. The disadvantage is that it can spend a lot of time analyzing parts of the domain where there is no change in dominant species.

The resolution specified for a grid plot divides the x- and y-axis ranges into a square grid with the specified number of nodes in each direction. There are therefore resolution-1 grid cells in each direction, each with a node at its centre. Since the cells are centered on the nodes and the plot is clipped on the domain boundaries, the peripheral set of cells around the plot are actually plotted as half cells.

Speciation calculations are carried out at each node and the dominant species identified for each node-centered cell. The results of these calculations are stored in the track file. This is the file that is used for replotting.

Each cell or pixel is 'coloured' according to the dominant species and the boundaries between species located to give polygons. This approach does not go down to the vector (individual line segment) level which means that it is not possible to simplify the boundaries using the same algorithm as used in the ht1 approach. The characteristic steps in the original pixel boundaries are therefore retained.

By default all of the polygons are coloured and labelled. This includes polygons such as H2(g) and the 'not available' or an NA field that is produced when the Phreeqc speciation fails. Any field can be omitted from a replot by setting the species (sp) number in the labels file (*.lab) to a negative value.

The polygons are rendered in order of decreasing size, largest first.

The results of the speciation are stored in the track ('trk') file. Since these are calculated in a well-defined manner, the calculations can be restarted from a partially complete set of calculations produced by a crash or by using an interrupt and stop. To do this, set calculationMethod to 3 and restart. This should resume calculations from where they left off and applies to both the track file and the out file if selected. This works with both the 'grid' and 'grids' calculation methods.

8.3 THE 'HUNT AND TRACK' APPROACH

8.3.1 Strategy

The 'hunt and track' approach finds and follows internal boundaries on predominance and mineral stability diagrams. It starts by hunting along the left-hand y axis until it finds a crossover of predominant species then uses this to track internally. Once this has been exhausted, it hunts along the remaining axes. Critically, this approach relies on the assumption that all such boundaries can be reached from a domain (axis) boundary, i.e. that there are no 'islands'. This is often the case (Example 1 and Example 3), but not always (Example 2 and Example 43).

The 'ht1' algorithm tracks the internal boundaries using single steps of fixed length along an imaginary grid. At each point, the algorithm requests a speciation calculation to be made and expects the speciation program to return the concentration and name of the top three species, i.e. the three most abundant species. These are transmitted via the SELECTED_OUTPUT file. The precise way that these are calculated is controlled by the user using statements within the
block of a Phreeqc input file. It is also possible to provide user-defined constraints that override the normal predominance sequence. This enables infeasible areas of the diagram to be clearly defined.

The input file also controls all other aspects of the chemistry including the initial chemical setup, e.g. total element concentrations and a list of all the gas, mineral and adsorbed phases potentially present, and the way in which the x- and y-axes of the diagram are to be traversed in response to requests from the ht1 algorithm.

The step size and resolution of the grid is controlled by the resolution parameter. The step size is simply the span of each axis (maximum value – minimum value) divided by the resolution-1. The resolution is always the same for both x- and y-axes.

8.3.2 Details of the ‘hunt and track’ algorithm

Applying the concepts of hunting and tracking at the practical level

We assume that presented with values for the two 'master' variables (representing the x and y axes on the predominance diagram), the speciation program can return either a complete speciation or at least, the predominant species. This is what the ht1.inc script does.

As mentioned above, the ht1 strategy for locating the field boundaries makes the assumption that all domain and axis boundaries are interconnected, i.e. there are no 'islands' (Kinniburgh and Cooper, 2004).

The grid approach on the other hand is rather inefficient since many of the points evaluated will be far from a boundary and will therefore contribute little useful information. It is also difficult to guarantee locating all of the fields without a fairly large computational effort. The ht1 approach concentrates on locating and tracking the various boundaries and puts no effort into computations far from the boundaries. It requires no initial grid of points. The algorithm is similar to that sometimes used to locate 'zero phase fraction lines' in phase diagrams and starts by systematically hunting along each axis boundary in turn until a change in predominance field is found. Once a change has been located, the algorithm tracks along the boundary until it reaches another axis, remembering any junctions with other predominant species as it goes. It then returns to these junctions to follow these paths until all of the internal boundaries have been followed. It then hunts along any remaining axes, again tracking internal boundaries until all four axes have been searched.

Our hunt and track algorithm is based on a fixed regular grid over the domain of interest defined by the x and y axes. The domain is usually rectangular, with its boundaries (axes) also being on the grid. The algorithm assumes that at least one predominance boundary actually crosses an axis (if none does, there must be just a single predominant species).

The first task is to identify such a crossing point by searching along boundaries. Starting at a selected corner of the domain, the predominant species is identified. The next point for computation is the nearest grid point on the axis defining the domain boundary, moving in a selected direction. If this has the same predominant species, the algorithm moves on to successive grid points on the axis until a change is found. This is the 'hunting' part of the algorithm. Once a point of change has been identified, the algorithm moves into "tracking" mode, following the predominance boundary within the domain.

Figure 8.2 shows an example with a predominance boundary (shown as a dashed line), whose precise location is unknown, crossing the y axis of a rectangular domain. Grid points where speciation calculations are undertaken are identified in Figure 8.2 by numbered open and filled circles. The numbering indicates the order in which the grid points of interest are visited.
Figure 8.2. Strategy for tracking the boundary between two fields. (a) after locating a change in dominance on a
domain boundary, cells are explored in the order a, b, c…. The numbers indicate the order and location where spe-
ciation calculations are undertaken and the filled and open symbols at the grid intersections indicate the dominant
species returned by these calculations; (b) method of linear interpolation used to establish boundary location in cell
d).

In this example, the first grid point in the sequence is on the y axis where the algorithm is in
hunting mode. It continues through points 2 and 3, where there is no change in predominant
species, reaching point 4 where a change is identified. There must therefore be an intersection
of the y axis with the predominance boundary between points 3 and 4. The tracking mode
now begins with predominance values calculated at points 5 and 6, which complete rectangle
a on the grid. The predominance boundary must exit through one of the remaining three sides
of this rectangle. This is immediately identified as the side linking points 5 and 6. This exit
side for rectangle a becomes the entry side for rectangle b, the second to be constructed. Cal-
culation of predominant species at points 7 and 8 shows that the exit side for rectangle b is the
side linking points 5 and 7. Rectangle c is built on this side, with exit side linked by points 7
and 10, on which rectangle d is constructed.

This technique tracks a sequence of grid squares through which the predominance boundary
runs. It does not provide coordinates through which the line passes. These can be approxi-
mated by linear interpolation along each exit or entry side. Four values are required to carry
out this interpolation (Section 8.2). These are the concentrations of the dominant and sub-
dominant species at the end points of the side. Denote the end points (x1,y1) and (x2,y2), and
the dominant and subdominant log10 concentrations are d1, s1, d2 and s2. Then the linearly
interpolated approximate location (xb,yb) of the predominance boundary as it crosses the line
joining (x1,y1) and (x2,y2) is:

On a rectangular grid, either x1=x2 or y1=y2, so one or other of the equations will always be
degenerate, with either xb=x1 or yb=y1. It is straightforward to extend this concept to three
components. This improves the all-important location of triple points.

The successive construction of exit/entry lines, and rectangles on the fixed grid continues until
it either exits from the domain, or a junction on the predominance boundary is identified. A
junction is indicated when more than two species are identified as dominant amongst the four
corners of a rectangle. If, for example, there are three dominant species then these define two
exit sides from a single rectangle. The two boundary lines from this rectangle are then tracked
in turn, with appropriate flagging of the necessary sequence of operations. For more complex
examples, a predominance boundary may return to a previously identified junction. The algo-
 rhythm keeps track of when this occurs. Finally, the line segments are assembled into polygons.

The domain boundary should be exhaustively searched in hunting mode to ensure no pre-
dominance boundaries are missed. Once a crossover point has been established, its location is
progressively refined with sub-grid accuracy. It is sometimes necessary to increase the grid res-
olution to ensure that the hunting and tracking sequences join up properly. This is done auto-
matically but requires a complete restart.

The present approach of stepping sequentially through every domain boundary grid point is recognised as inefficient, though guaranteed to detect all crossings within the resolution (grid spacing) chosen. If the grid is too coarse, the location of junctions may be poorly estimated. For a fixed grid algorithm of this sort there will be a trade-off between computational efficiency and accuracy. A grid spacing equivalent to a resolution of 1/500 of the domain boundary normally gives smooth and reliable curves. The advantage of using a fixed grid is that quite complex curves may be tracked. However, the scheme does not take full account of the known characteristics of some predominance boundaries inherent in the underlying chemical model. This can lead to some inefficiency. We make no assumptions about the curvature of the field boundaries which are often straight but which can be curved and can even show very sharp changes of direction.

A similar tracking procedure can be followed when the stability criterion is used or when an artificial boundary is added as a result of a user-defined constraint, e.g. \( \text{H}_2(\text{g}) = 1 \text{ atm} \). In these cases, linear interpolation can no longer be used to refine the point of intersection and so the mid-point is always chosen.

**Definitions of 'predominance' and mineral stability**

In classical mineral stability diagrams, the position of the mineral-solution boundary depends on the assumed activity of the solution components at the boundary. Garrels and Christ, following Pourbaix, defined the boundary as the point where the 'sum of the activities of the ions in equilibrium with the solid exceeds some chosen value'. They chose 10-6 as a default value on the basis that if it is less than this value, the solid will tend to behave as an immobile constituent in the environment. In the full speciation approach, the activity at the boundary is determined by the system specification and the speciation calculation and will normally be specified by either a fixed total amount of the various components or by a fixed activity/fugacity, e.g. for gases. There is no need for excessive simplification of the system and so such diagrams can be customised for systems of particular interest. The only decision is whether to draw the boundary for the predominance domain or the stability domain.

When adsorption reactions are included, it is possible to compare the total number of moles of the main species in each of the phases (gas, solid, solution, adsorbed) to determine the predominant phase (out of the four possibilities). Alternatively, more detail can be revealed if each species is considered independently as is normally done in solution-only predominance diagrams. The predominant species is then defined as the species accounting for the greatest number of moles in the whole system. The `htl.inc` code treats each adsorbed species as a separate entity, just like a solution species. The definition of 'species' becomes more ambiguous when an 'adsorbed' phase is considered since the normal definition of an adsorbed species is in terms of the type of binding site and so would make the diagrams very sensitive to adsorption model adopted. Probably a more useful approach is to lump all adsorbed species together into one 'super species' for the purposes of ranking. This is the approach adopted in the `htlc.inc` code. If the detailed adsorbed speciation is of particular interest, then `htl.inc` should be used.

**8.3.3 Failure of the 'hunt and track' approach**

One advantage of the 'hunt and track' approach is that it can create a vectorised diagram directly. This leads to a small plot file size and after smoothing of the field boundaries can produce smooth-looking boundaries as opposed to the jagged boundaries of the grid approach.

In principle the 'hunt and track' approach can produce better quality diagrams with less computational effort - the effort depends on the length of the boundaries that must be tracked rather than on the square of the resolution as with the grid approach. But the important caveat is that it can miss potentially important fields if they do not cross any of the domain boundaries. This can occur when 'wedges' intersect the domain boundary.

Such islands appear to be quite rare and the phase rule indicates that they are likely to be con-
fined to solution species. One example is given in Example 43. We have checked all the other diagrams in this report and have found no other ‘islands’.

Nevertheless, it is always important to check that there are no islands by switching the calculationType from ‘ht1’ to ‘grid’ and reducing the resolution setting to a more reasonable value.

A second problem is that in order to generate the required field boundaries and associated polygons, it is necessary that all the various line segments produced during tracking fit together exactly. Because of numerical errors implicit in the numerical approach used by Phreeqc, this is not always the case and it is possible to get lack of closure of some polygons.

It is also sometimes problematic connecting the ‘hunt’ part which is moving along the domain boundaries and the ‘track’ part which is moving along the internal boundaries. This is especially true when a field boundary happens to intersect, or runs very close, to a domain boundary. This can lead to a failure to close all the polygons, i.e. to create the clean polygons necessary for colour filling. This can sometimes be avoided by changing the resolution of the plot – and PhreePlot will attempt to do this to some extent automatically – but if this fails then manually changing the domain boundaries so that there is no intersection very close should help. Alternatively try the ‘grid’ approach.

8.4 FEASIBLE DOMAINS AND THE PREPARATION OF EH (PE) -pH DIAGRAMS

8.4.1 General principles

The classical Pourbaix diagrams extend from pH 0 or less to pH 14 and over a wide range of redox conditions effectively ranging in oxygen partial pressures, say from more than 1e0 to less than 1e-100 atm. Extreme conditions such as these may not be physically realistic and are only considered in classic Pourbaix diagrams because the constraints of a full speciation are not imposed. This is also not entirely within the domain within which Phreeqc is able to operate (at least using its ion association activity model) especially under the extremes of redox conditions where water is itself not stable – making the concept of ‘dissolved’ species a nonsense. Phreeqc keeps track of the mass of water and when most of it has evaporated, the concentration of the remaining solutes can become extremely high in the small amount of water remaining. This domain of failure can be mapped and is shown using the ht1 approach (Figure 8.3). The grid approach gives a similar result.

At 25°C, the failure occurs at below log $f_{O_2}(g)$ of -96 atm and arises because of the strong decomposition of water under these conditions – it decomposes releasing H₂(g). This failure occurs at a higher log $f_{O_2}(g)$ at higher temperatures, e.g. at log $f_{O_2}(g)$ of -86 at 60°C. There is also a limit to the maximum $O_2(g)$ partial pressure that can be sustained in aqueous systems since it combines with H⁺ to produce water. Phreeqc will eventually fail under these extreme conditions, though not without trying hard (and taking a lot of time attempting to find a solution).

As a consequence of these reactions and of limits to the practical limits of $O_2(g)$ that are reasonable, a typical Eh(pe)-pH has upper and lower diagonal lines that demarcate regions where the diagram is not evaluated.

On an orthogonal log $f_{O_2}(g)$-pH diagram, these ‘no evaluation’ regions can be simply excluded from being calculated by specifying appropriate upper and lower limits to the $O_2(g)$ fugacity.

Therefore a pe-pH diagram can be most easily prepared by specifying the y-axis variable to be log $f_{O_2}(g)$ and then converting the redox scale with the yscale setting. It is possible to drive the y axis with the pe by defining the pe in an analogous way to that used for pH:

\[
\text{Fixed}_e - e^- = e^- \\
\text{log } K = 0.0
\]

and then
**8.4.2 Domain tags - avoiding speciation calculations**

The calculation domain for a predominance diagram or a contour plot is always a rectangular area defined by \((x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}})\). It can be useful to omit certain parts of this domain from the speciation calculations, e.g. because the speciation is known to be unsuccessful or unnecessary in some region(s).

A special set of tags called *domain tags* can be used to clip the speciation domain, e.g.

```plaintext
<domain1_value> = "<x_axis>+<y_axis>" \
<domain1_min> = -2 \
<domain1_max> = 22
```

There can be up to 9 sets of domain tags, namely `<domain1_value>` up to `<domain9_value>` and their corresponding min and max tags.

These tags are defined in the usual `numericTags` block. `<domain_n_value>` is evaluated before each speciation calculation and compared with the corresponding minimum and maximum values set with the `<domain_n_min>` and `<domain_n_max>` tags. If any of the values are outside this range, then the speciation calculation is skipped.

Note that the domain value needs to be determined *before* the speciation calculations and so
the definition of the `<domain1_value>` tag should only use those tags that are known before speciation. The most obvious ones are the `<x_axis>` and `<y_axis>` tags.

The above test is only applied to main loop simulations. Pre-loop simulations will always be calculated in full.

These tags can be useful when generating a pe-pH diagram in order to eliminate speciation calculations from areas outside of the lower and upper bounds for the stability of water.

The 'ht1' method of calculating a predominance diagram requires that access to the boundaries is available from one of the domain boundaries. So it is not possible to reduce the domain size on all four sides since there will then be no access to the inner region. However, the 'grid' approach would work with these conditions.

The skipped calculation is by default assigned the 'species' name "Skip" and speciation values are set to UNDEFINED. The results are included in the track file as usual. A '-' sign precedes the iteration number of the rolling summary shown on the screen during the calculations.

The species name and therefore the label used can be renamed by editing the labels file and replotting, or by adding a special character tag with the name, `<domain1_name>`. The name can be the empty string, "". The name of the field used is taken from the first out-of-domain criterion searching from 1 to 9.

### 8.4.3 Speciation failure when there is not enough reactant present

This most frequently happens when trying to fix the pH using the `Fix_pH` ploy. Phreeqc will fail to converge at low pH if the reaction being used to achieve the desired pH is not feasible. For example, if NaOH is being used to change a solution from pH 2 to pH 1, it will likely eventually fail. In the presence of a background electrolyte containing Na such as NaCl, this failure will not occur at exactly pH 2 but at a somewhat lower pH depending on the amount of Na present. Phreeqc will attempt to achieve the low pH by withdrawing NaOH (negative NaOH additions) until all the Na has disappeared at which point it will fail. This problem can be solved by linking the Na to a large reservoir of a Na-containing mineral (Section 6.5.5) such as NaCl (halite) but this itself can have undesirable side effects. The problem can be more complicated when other side reactions such as redox reactions are themselves producing/consuming protons. Then when both pe and pH are changed, it is not necessarily obvious whether acid or base needs to be added to change the pH.

### 8.5 CHOICE OF THE RESOLUTION OF THE PLOT

The speed of calculation depends on many factors including the complexity of the chemistry, especially the number of mineral phases, the length of the USER_PUNCH code and the resolution of the plot. A reasonable approach is to start at a low resolution, say 50-100 for a ht1 plot or 20-50 for a grid plot, and increase it when a production quality plot is required. The resolution must be 10 or greater and should normally be less than 2000. The ht1 algorithm can fail to resolve junctions at low resolutions which can lead to a failure to close all the polygons properly.

If more detail is required for a particular area, zoom in by reducing the domain size with the `xmin`, `xmax`, `ymin`, or `ymax` parameters and recalculate rather than just replotting at the new scale.

PhreePlot sometimes overrides the resolution originally set in the input files and either increases or decreases it. It does this when it either needs more resolution to resolve apparent 4-way junctions or when a junction is too close to a domain boundary: Changing the relevant domain boundaries (`xmin`, `xmax`, `ymin`, or `ymax`) would avoid the latter problem. A reduction in resolution is sometimes necessary if the output from Phreeqc is for any reason unstable.
8.6 MONITORING THE PROGRESS OF A ‘HUNT AND TRACK’ RUN

Providing that the screen output has not been disabled, progress of the tracking will be displayed on the screen. An example is given below:

```plaintext
*** PhreePlot *** Pre-release 0.01 (3 Jan 2008)
Incorporating the Phreeqc library by DL Parkhurst, SR Charlton (USGS), & CAJ Appelo (Amsterdam)
Hunt & Track by DG Kinniburgh, BGS and DM Cooper, CEH (NERC)
Fitting by MJD Powell (University of Cambridge)
Postscript plotting by KE Kohler

<mainspecies> = Se
1 2.0000  -80.0000  11 Se          H2Se        -3.0010      -5.6388
2 2.0000  -76.8000  11 Se          H2Se        -3.0000      -7.2388
3 2.0000  -73.6000  11 Se          H2Se        -3.0000      -8.8388
4 2.0000  -70.4000  11 Se          H2Se        -3.0000      -10.439
5 2.0000  -67.2000  11 Se          H2Se        -3.0000      -11.847
6 2.0000  -64.0000  11 Se          H2Se        -3.0000      -13.447
7 2.0000  -60.8000  11 Se          H2Se        -3.0000      -15.047
8 2.0000  -57.6000  11 Se          H2Se        -3.0000      -16.647
9 2.0000  -54.4000  11 Se          H2Se        -3.0000      -18.247
10 2.0000  -51.2000  11 Se          H2Se        -3.0000      -18.105
11 2.0000  -48.0000  11 Se          H2Se        -3.0000      -14.905
12 2.0000  -44.8000  11 Se          H2Se        -3.0000      -11.705
13 2.0000  -41.6000  11 Se          H2Se        -3.0000      -8.5055
14 2.0000  -38.4000  11 Se          H2Se        -3.0000      -5.3055
15 2.0000  -35.2000  11 H2SeO3      HSeO3-      -3.1277      -3.5938
16 2.0000  -36.8000  11 H2SeO3      HSeO3-      -3.1333      -3.7055
17 2.0000  -36.0000  11 H2SeO3      HSeO3-      -3.1333      -3.5938
18 2.0000  -36.4000  11 Se          H2SeO3      -3.1333      -3.7055
19 2.0000  -36.0000  11 H2SeO3      HSeO3-      -3.1277      -3.5938
20 2.0000  -36.4000  11 H2SeO3      HSeO3-      -3.1277      -3.5938
21 2.0000  -36.4000  11 H2SeO3      Se          -3.3055      -3.4738
22 2.0000  -36.4000  11 H2SeO3      HSeO3-      -3.1277      -3.5938
23 2.0000  -36.4000  11 Se          H2SeO3      -3.3055      -3.4738
24 2.0000  -36.4000  11 H2SeO3      Se          -3.3055      -3.4738
25 2.0000  -36.4000  11 H2SeO3      HSeO3-      -3.1277      -3.5938
26 2.0000  -36.4000  11 H2SeO3      HSeO3-      -3.1277      -3.5938
27 2.0000  -36.4000  11 Se          H2SeO3      -3.1277      -3.5938
28 2.0000  -36.4000  11 H2SeO3      HSeO3-      -3.1277      -3.5938
29 2.0000  -36.4000  11 Se          H2SeO3      -3.1277      -3.5938
30 2.0000  -36.8000  21 Se          H2SeO3      -3.1333      -3.7055
31 2.0000  -36.0000  22 H2SeO3      HSeO3-      -3.1277      -3.5938
32 2.0800  -36.0000  23 H2SeO3      HSeO3-      -3.1496      -3.5356
33 2.0800  -36.0000  24 Se          H2SeO3      -3.1496      -3.5356
34 2.0800  -36.0000  22 H2SeO3      HSeO3-      -3.1496      -3.5356
35 2.0800  -36.0000  23 H2SeO3      HSeO3-      -3.1496      -3.5356
36 2.0800  -36.0000  24 Se          H2SeO3      -3.1496      -3.5356
37 2.0800  -36.0000  22 H2SeO3      HSeO3-      -3.1496      -3.5356
38 2.0800  -36.0000  23 H2SeO3      HSeO3-      -3.1496      -3.5356
39 2.0800  -36.0000  24 Se          H2SeO3      -3.1496      -3.5356
40 2.1600  -36.0000  22 H2SeO3      HSeO3-      -3.1744      -3.4804
41 2.1600  -36.0000  23 H2SeO3      HSeO3-      -3.1744      -3.4804
42 2.1600  -36.0000  24 Se          H2SeO3      -3.1744      -3.4804
43 2.2400  -36.0000  23 H2SeO3      HSeO3-      -3.2026      -3.4285
44 2.2400  -36.0000  24 Se          H2SeO3      -3.2026      -3.4285
45 2.2400  -36.0000  22 H2SeO3      HSeO3-      -3.1638      -3.7055
```

The columns are from left to right:

- **iteration number** (number of speciation calculations)
- **x-axis value**
- **y-axis value**
- **type of move**
  - **sign:** a -ve sign means a 'constraint' (a forced value) is in operation
  - **first digit:** 1 = hunting along an edge  2 = tracking an internal boundary
  - **second digit:** 1-4, side being traversed (starting with the left-hand y axis as 1 and counting clockwise).

- '00' is used for 'grid' plotting, '20' for a 'multi-point' point
- dominant species name
- subdominant species name
log concentration (mol/kgw) of dominant species
log concentration (mol/kgw) of subdominant species.

Where a constraint is operating, the indicated ‘dominant species’ is the constraint species, such as a gas or in the case of a mineral stability diagram, a mineral. The indicated ‘subdominant’ species is the species that would be dominant in the absence of the constraint. The numeric values of constraints are the log’s of the constraint value. In the case of pure phases (gases and minerals), this is the saturation index.

In the above example, the algorithm starts hunting along the y axis, finds a boundary crossing the axis between -36.4625 and -36.8 and then starts tracking inwards along that boundary. The boundary being tracked is between H₂SeO₃ and Se.

A full record of the tracking is recorded in the track file.

A graphical display of the grid or ht1 tracking can be obtained by using the ‘\texttt{esc p}’ combination. A plot file ‘\texttt{plot.ps}’ will then be written to the input file directory showing progress. In a ht1 plot, the blue filled circle shows the current calculating position. This plot only records the tracking phase, not the hunting phase.

\section*{8.7 Plotting and Replotting}

\texttt{calculationMethod} 1 does a full set of speciation calculations and will generate all the files necessary for plotting.

\texttt{calculationMethod} 2 does not recalculate the speciation or redo the calculation of the polygon boundaries but reads in these results from the polygon file and the label names and positions from the labels file.

\texttt{calculationMethod} 3 does not recalculate the speciation but does recalculate the polygons and label placement.

\section*{8.8 Modifying the Appearance of Predominance Plots}

The appearance of the plot can be modified through many of the keywords. These can be changed in the input file and the file rerun with one of the two replot options - there is no need to redo the calculations unless a different resolution is required.

The colours of the fields can be modified by editing the appropriate fill colour dictionary - that is a file with the default name of \texttt{fillcolor.dat}.

Fields are known primarily by the number assigned to them. The labels file translates this number into a field name. The colouring of individual polygons can be turned off by setting the species number in the labels file to zero.

The appearance of internal boundary lines is controlled by the the settings: \texttt{linecolor}, \texttt{lineWidth}, \texttt{lineType} and \texttt{dashesPerInch}. The ‘ht1’ method uses the vector file to plot the boundary and only plots each boundary once. The ‘grid’ method uses the polygon file to plot the boundaries and common boundaries will be plotted twice. This is likely to affect the appearance of dashed lines in ‘grid’ plots since the overwriting may have different starting points.

Individual polygons, and their boundaries, can be removed from the plot by setting the species number for all points (‘lines’) for the polygons of interest in the polygon file to zero or a negative number (reversing the sign in a text editor is the most convenient way of doing this).

The label name, position and orientation can be changed by editing the labels file. This can include changing the species name including to a blank field name, ‘\texttt{"}’. The label is known by its species number. Normally the label attributes are recalculated each time new field boundaries are generated (\texttt{calculationMethod} 1) but this can be changed by setting \texttt{useLabelsFile} to \texttt{TRUE}. This ensures that the readings from the labels file will be used if possible. If the file does not exist, it will be created. \texttt{calculationMethod} 2 has the same effect for a replot.
It is possible to change the y scale (native, pe, mV or V) without recalculation using `yscale`. Plot limits can be changed using `pxmin` etc but beware that if a larger range is specified there will be a blank area around the plot and if a smaller domain is chosen the field boundaries will appear correspondingly coarse – would be better to recalculate at the new resolution which will also calculate the label positions better.

The 'steppiness' of the boundaries in 'ht1' plots can be controlled with the `simplify` keyword. The default value is 1 so choose a larger value (up to 10 say) to smooth the line more. Recalculate the boundaries and replot using `calculationMethod 3` rather than 2 since the smoothing has to be redone. A `simplify` value of 0 will show all the boundary points. 'grid' plots are by definition 'steppy' and `simplify` has no effect on this.

Additional text, including labels, can be added with `extraText`. There is full control over the plot on which to apply the text plus the font, size, colour, justification and orientation of the added text.

Additional data from other files can be added to the plot using the `points` and `lines` keywords combined with the `extradat` keyword to add the additional data files to the search path. These data must be in regular tabular output format. The x-column of each file should be labelled with 'x' in the header since this is the label that is always used in predominance plot files. There is no automatic labelling or generation of a legend for lines or points added in this way. `extraText` can be used to add labelling manually. `extraSymbolsLines` can be used where more control is wanted over the symbols used or the line widths.

The entire plot can be rescaled with `plotFactor`.

Some of the main settings for `ill` are shown in Figure 8.4. For more complete control over the appearance of the plot transfer the data to a more powerful plotting package. This can be done at either the image (ps file) or data (pol and vec files) level.

Figure 8.4. Typical 'ht1' predominance plot showing some of the keywords that control the appearance of the plot.
8.9 ADDING LINES AND POINTS TO A PREDOMINANCE PLOT

The usual way of adding lines and points to a plot is through an `extraSymbolsLines` file. However, it is also possible to add lines and points to a predominance plot using the `lines` and `points` keywords. However, these data are not included in the auto scaling – this will be determined by the predominance plot alone. `customXcolumn` will need to be defined though this does not need to match the x-axis variable in the predominance plot. No legend will be produced. The 2y axis option does not operate in this mode.

It is also possible to overlay one or more existing graphics (in the form of PhreePlot-generated ps files) on top of the existing plot using an `overlay`. This can be useful for adding lines from another predominance plot.

8.10 CONTROLLING THE LABELLING OF PLOTS AND THE PLOTTING OF FIELDS

The label positions are taken from the labels file and so may not be centered if the plot has been rescaled. They will not be plotted if their centres are out of the plot area. In such cases, either do a full recalculation or just a recalculation of label positions(`calculationMethod`, 1 or 3, respectively) or edit the labels file. Fine tuning of label positions can either be done by editing the labels file or using a `nudge` file. The easiest way to do this is to set `nudge` `TRUE` and replot to create an empty nudge file containing all the labels. Then edit this file and add it to an input file with the `nudgeFile` keyword, and replot. You can specify positions either in an absolute sense (x, y position in mm etc) or in a relative sense (change in x, y position in mm etc.). This will be applied to all labels with the specified name so is less likely to be useful where several polygons have the same label.

Turning the printing of individual labels on and off can be automatically controlled with the `minimumAreaForLabelling` based on the size of the fields (useful for excluding the labelling of small fields) or by editing the labels files and setting the species number to a negative value. All labelling can be turned off by setting `labelSize` to zero. A single label can be omitted by nudging it to oblivion. The plotting of entire polygons can be turned off with the exclude list of the `pol` keyword. In a ‘ht1’ plot, the domain boundaries can be turned off with `domain` - most relevant when the y axis has been used leaving the original domain boundaries within the plot.

The plotting of labels also depends on the `calculationMethod` setting – choose method 2 or 3 after editing the labels file (Table 8.1). The labels file provides the link between species number and species name so editing this file can change the appearance of labels and their position during a replot. The exclude list of the `pol` keyword can be used to omit specific fields completely including their boundary lines, coloured infill and labels.

Label names are regenerated whenever `calculationMethod` 1 is used. If you want to override a label name at this stage, then you can edit the `ht1.inc` file to make the appropriate substitution. For example, in the demo\Fe\hfo.ppi file, say you wanted to rename Fe(OH)3(a) to Hfo then add the following line

```
131 IF (n$(i) = "Fe(OH)3(a)") THEN n$(i) = "Hfo"
```

and it will intercept the name and change it. Save the file with a different name and use this in the ppi file. Alternatively, for a one-off, you can add the above line directly to your ppi file...

```
include 'ht1.inc'
131 IF (n$(i) = "Fe(OH)3(a)") THEN n$(i) = "Hfo"
```

since the Basic interpreter executes the lines in line number order.

The `domain` keyword controls whether the domain boundaries will be plotted or not. The
default is T(RUE) which will plot the boundaries. If the native scale and automatic axis scaling are chosen, the domain boundaries will coincide with the axes boundaries. It can be useful to switch the domain boundaries off when a plot is rescaled and a constraint such as a gas partial pressure exists, e.g. to remove the outer lines specifying the water limits in pe-pH plots.

The default x- and y-limits to a predominance plot are set to the calculation domain as specified by xmin, xmax etc and any fields with out-of-plot boundaries are clipped. The plot limits can be reset with pxmin, pxmax etc.

Since predominance plots calculated with the 'grid' approach have cell-centered speciation values, the first and last rows and columns of the grid are half clipped when the default (auto) x- and y-limits are specified.

The labels are allowed to extend beyond the plot domain by 15% of the axis length. Anything beyond that is clipped.

8.11 WHY DO I NOT SEE METHANE GAS WHEN USING llnl.dat?

Unlike when using the wateq4f.dat database, you are unlikely to see methane gas (CH4(g)) as a predominant species when using the llnl.dat database even under strongly reducing conditions and in the presence of carbon. Why is this since it appears to be in the database?

The answer is that the llnl.dat has many more ‘exotic’ species in its database compared with wateq4f.dat. This includes the dissolved species CO, C2H4 and C3H6 as well as CO(g). Perhaps surprisingly, given the relative abundance of methane in the natural environment, these species are calculated to be thermodynamically dominant over methane in many reducing environments. The fact that they are not so abundant in most natural environments suggests that kinetic factors may be preventing their dominance in the ‘real’ world.

Therefore in order to see methane in predominance diagrams using PhreePlot, remove these ‘exotic’ species from consideration either by adding the species to an input file and changing the log_value to make it insignificant or, for a more permanent solution, delete them from the database file by commenting them out.

Of course this all becomes clearer if you plot a predominance diagram with C as the main species!

8.12 FAILURE TO COMPLETE A PREDOMINANCE DIAGRAM

The grid approach should always produce a valid diagram although its quality will be deter-
mined by the resolution chosen. There is no requirement for any spatial continuity between adjacent cells.

There is no guarantee that the 'hunt and track' approach will always work. Either Phreeqc may fail or the tracking may fail. Possible reasons for the failure of Phreeqc have been discussed elsewhere (Section 6.5.5).

The simple 'hunt and track' algorithm used in PhreePlot assumes that the speciation is returned without error and tracks accordingly. Clearly since all speciation programs, including Phreeqc, use numerical methods to derive their solution, the boundaries between fields must contain some 'noise'. This may mean that the fields reported near field boundaries may themselves be in error, e.g. varying 1212 rather than 1122. This may in turn mean that the fields (polygons) cannot be closed properly and so cannot be plotted as coloured polygons. This is par for the course and is not an error in Phreeqc per se but in the way it is being used.

In Phreeqc, the convergence_tolerance parameter in the KNOBS keyword block specifies a relative error for an element's mass balance and this controls when a valid solution is deemed to have been achieved. Typically this parameter is set at 1e-12 when SELECTED_OUTPUT; -high_precision is set to true. Either use KNOBS to increase it (e.g. to 1e-8) or set -high_precision to false. It may be helpful to change this setting where there is a problem in convergence typically seen when the residuals are very small but exceed 1e-12. Other possibilities have been discussed earlier.

If the tracking grid falls on or very close to a true predominance boundary, PhreePlot could end up tracking noise, or more likely, some of the results of the speciation calculation could be erroneous resulting in confusion for the tracking algorithm as indicated above. This can also occur when a field boundary–domain boundary intersection is very close to one of the four domain corners. If detected, PhreePlot attempts to get round this automatically by reducing the resolution of the plot by about 10% and recalculating but this does not always work. A '*' in the n column of the pp.log file indicates that an automatic restart has been made. The Mn.ppi, fluoritepredominance.ppi and fluoritestability.ppi demos are examples of this.

It also sometimes occurs that the 'ht1' method requires a greater resolution than given to resolve a junction or to close all polygons. In these cases, PhreePlot will automatically restart with a doubling of the resolution. This increase will be repeated if necessary up to a maximum resolution of 2000.

Other actions that can be used to resolve failures of the method involve moving the grid in other ways: changing the domain of the calculations (xmin, xmax, ymin, ymax) or by reducing the resolution more radically.

In rare cases, Phreeqc does not converge at all. This is usually, but not necessarily, clearly signalled by PhreePlot and can often be seen by the failure to write the SELECTED_OUTPUT file. Convergence can sometimes be helped by (i) changing the -high_precision setting in the SELECTED_OUTPUT section of the ht1.inc file (if used) to FALSE; (ii) reducing the convergence tolerance, (iii) reducing the pe_step_size (under KNOBS), or (iv) increasing the allowed maximum number of iterations (also under KNOBS). However, by far the most common reason for failure to converge is because of a poorly-constructed Phreeqc input file, i.e. inconsistent chemistry somewhere.

With debug >= 1, failure, if detected, will result in an immediate halt to execution and a dump of the offending Phreeqc.out file, and in the case of debug = 2 and 3, of all the Phreeqc output to that point. With debug = 0, the failed region will be mapped as its own species ('NA') and a question mark will be added to the pp.log file to indicate a failure.
9 Contour plots

9.1 WHAT ARE CONTOUR PLOTS?

Contour plots are the sort of plots you see when looking at a topographic map, i.e. a diagram showing lines of equal height. Each height is called a ‘contour’. Contours do not normally cross each other and always change in a regular sequence, one contour followed by one of its nearest neighbours. There should be no missing, intervening contours.

This concept of ‘height’ can of course be generalised to any continuous variable and that is what is done here. Note however that not all geochemical variables are continuous variables. Phase changes for example can introduce step changes and this can cause problems in contouring.

Contour plots provide another way of viewing x, y, z data where z is the variable being contoured. In some senses, they complement predominance diagrams.

Contour plots only work well for relatively smooth, continuously varying variables of which the latter characteristic is the most important. Sometimes, gradients in geochemical variables can either be so large that they do not appear continuous or the reporting of the variable has been truncated to such an extent that the variable is no longer continuous. Then contouring does not work well.

Contours are produced by generating a uniform grid of data over the given x- and y-domain. It is assumed there are no missing or undefined points in this data set.

Also, it is assumed that all the contoured data is valid data, i.e. that all Phreeqc calculations have converged properly. If the x and/or y variable are being ‘fixed’ by adding an equilibrium phase, e.g. as in Fix_H+, then it is wise to check that the target value has actually been achieved. Sometimes, Phreeqc does not trigger an error when there has been a failure to converge; it just gives a warning. It will also not issue an error if the reservoir has run out. If in any doubt, add the ‘-force_equality TRUE’ identifier to all phases.

This check can often be done by adding the value(s) achieved and the target value(s) to the selected output, e.g. if the x-axis is supposed to be the pH, then comparing the actual pH achieved, $\text{-la(“H+”)}$, with the $<x\_axis>$ tag will indicate if convergence has occurred. The track file will also give blank output values where there has been a failure to converge and where an error return has been indicated.

Contours can in principle be viewed from various angles. Here we deal only with a view directly from above rather like in a topographic map.

9.2 IMPLEMENTATION

9.2.1 Generating the contour data

The contouring algorithm requires smooth, regularly-gridded data. This is generated by PhreePlot in the same way that is done for ‘grid’ plots using xmin, xmax, ymin, ymax and resolution. The resolution must be greater than 1 and normally should be in the range 10–100. The gridded data is picked up from the ‘out’ file.

A contour plot is usually made by specifying the calculationType to be ‘contour’ and by specifying a contourZVariable. This is the name of one of the variables in the ‘out’ file. A special type of contour plot, namely a residual sum of squares or function plot, can also be specified.
from the ‘fit’ calculationType with the fitMethod set to ‘contour’.

You must provide the Phreeqc code to calculate a z-value at each x, y point. This is normally done using the selected_output and user_punch data blocks. The name of this variable is given by the contourZvariable keyword defined in an input file and should correspond with one of the user_punch ‘out’ file headings.

There can be more than one contourZvariable. If there is more than one mainSpecies specified on an input line, this will cause the mainSpecies loop to operate, on iteration for each of the given mainSpecies. A different contourZvariable is picked from this list for each iteration. If the contourZvariable list is shorter than the mainSpecies list, then it is recycled.

If the contourZvariable is not defined or not found, then a fatal error is issued.

It is not necessary to output the corresponding x- and y-variable values since these are defined implicitly by their respective ranges, the grid resolution and the iteration number. However, if the output data are to be used by some other software package, it may be helpful to output their values. Simply add the <x_axis> and <y_axis> tags to the list of variables to be ‘punched’.

Note that because the x- and y-values are implicit, you have to be careful to make sure that you regenerate the whole set of data when changing any of the parameters that define the domain of the grid or the speciation calculations. The contouring package has no knowledge of these x- and y-values explicitly – it calculates their values from their position in the ‘out’ file. If the speciation fails and there is no output, an UNDEFINED value placeholder is normally written to the out and trk files. This can be prevented by setting writePlaceholder to FALSE. A FALSE setting is useful when a failed speciation is repeated as in the \demo\switch examples.

The z-data can be smoothed using a five point nearest-neighbour moving average algorithm. This is specified with the contourOptions smooth 1 statement. Somewhat greater smoothing can be gained using smooth 2. This is a 9-point nearest neighbour algorithm. The smoothed data are not stored externally but are generated from the raw data as needed. A setting of smooth 0 turns off all smoothing.

9.2.2 Choosing the contour levels

The contour levels are specified with the contours keyword. This can either be a list of user-defined values in ascending order, or can be one of several automatic selected sets of values. See contours for details. The default is for a list of 17 values to be automatically generated by sub-dividing the range of the z-data values into 16 equal intervals (empirical values). It is also possible to choose the contour values based on percentiles.

The main requirement for the selection of a set of contour values is that the values are well separated – they should not be so close to each other that they enter the area of ‘noise’ created by the numerical procedures used in generating the data being contoured. It is also unwise to set a contour exactly on a value which for some reason (chemical buffering, numerics) is very common. This strategy will avoid the problem of the contour running along a boundary leading to a sensitivity to numerical errors.

The number of contours used will be automatically reduced if neighbouring contour values are deemed to be too close to each other. Successive contours should always differ by at least 1e-8 x average value of successive pairs of values. This is to try and minimise the effect of small numerical errors on the drawing of the contours. The ‘simplified percentiles’ auto option requires a greater separation between contour values and prunes the set of contour values even more aggressively (see contours).

9.2.3 Types of contour plot

Two types of contour plots are available:

(i) a plot consisting of contour lines with colour-fill in between the contours (sometimes called a ‘level’ plot)
(ii) a 'lines only' plot.

The **contourOptions** keyword determines which of these two types of plot is used.

A plot with colour fills is usually the most pleasing and can be customised in terms of the number and position of the contour levels, the fill and line colours, and the line, label and legend styles.

The lines only plot (**contourOptions fill FALSE joinSegments FALSE**) uses the most straightforward way of producing a contour plot since it does not require the various segments produced by the contouring algorithm to be combined together and with border segments to produce the polygons required for colour filling. The 'lines only' plot comes in two flavours: the simplest one is merely a plot of the short segments produced by the contouring algorithm scanning across the domain at each contour level. In this case, there is no requirement for the segments to be plotted in any particular order. This is an advantage in noisy, high resolution plots where defining the numerous polygons can be rather slow and the numerous line segments can sometimes be difficult to assemble in the proper order – the **joinSegments FALSE** option avoids this possibility and is therefore the ‘fall back’ option.

If you change the **joinSegments** option, you will have to regenerate the plot from scratch (not just replot it) since the structure of the vector file varies between the two approaches.

The lack of joined-up line segments can lead to visibly poor curves since the better-looking line joins cannot be used. It is also not possible to take account of the full length of the line when calculating the best position for dashes in dashed lines resulting in poor dashed lines especially when the resolution is high. In particular with dashed lines each line segment will have at least one ‘dash’ and so the dash density may be greater than specified by the **contourDashesPerInch** setting (unlike in the legend). It is best to avoid dashed lines with this option.

Finally no line simplification is undertaken giving more intricate contours and larger plot files.

A better strategy is to at least sort the segments into continuous contour lines which can then be simplified and plotted as a single curved line with proper line joins (**contourOptions fill FALSE joinSegments TRUE**).

Adding the various boundary segments to the sorted contour lines then enables the polygons to be closed and so filled with colour (**contourOptions fill TRUE**). It also enables a place for an in-line contour label to be put. This is the default plot type.

Of course the fill colours can be set to 'nd' or 'white' to produce what looks like a contours only plot. However there are still subtle differences in the way that the contours are labelled and the type of legend produced.

If a colour fill plot is selected but **PhreePlot** fails to close all the polygons properly for some reason, then the polygons that could not be closed are not drawn. Sometimes changing the smoothing option from 0 to 1 or 2, or vice versa, is sufficient to allow the plot to complete properly. A lines only plot should always work.

### 9.2.4 Changing the appearance of the contour lines.

The **contourLineType** keyword allows the line types for the contour lines to be specified using the usual recycling rules. **contourDashesPerInch** defines the dash density, **contourLineColor** defines the color and **contourLineWidth** defines the line width. For normal (**contourOptions fill TRUE**) plots, **contourLabelSize** controls whether a label is printed in the contour line and how large it is. If the **fill** option is **FALSE**, the contour values will only be given in the legend so normally **fill** should be **TRUE**.

### 9.2.5 Colouring the plot

The contour lines, contour fill and contour label can all be independently coloured using **contourLineColor**, **contourFillColour** and **contourLabelColor** as lists of colours that are recycled as necessary. Each of these has an 'auto' setting by default meaning that **PhreePlot** will choose
the colours.
All colours in a given list, or individual ones, can be ‘turned off’ by setting the colour to ‘not
drawn’ (‘nd’). For colour fills, it is safer to set an unwanted colour to the background colour,
e.g. ‘white’ rather than using ‘nd’ since in the case of closed contour levels, the colours levels
are filled from the outside in and the colouring relies on successively overwriting an outer
colour by the next inner colour.

9.2.6 Labels and Legend
If a lines only plot is produced, the legend or key simply consists of lines with the colour and
width of the corresponding contour lines and labelled with the contour value. The size of this
text is controlled by the contourLabelSize setting.
The labels are placed within the longest line segment on the simplified contour but the position
of all contour labels can be quickly changed by changing the labelPosition option in
contourOptions. If this setting is set to ‘centre’ rather than ‘longest’, the labels will be placed
at the centre of each plotted line.
A contour may consist of several non-connected segments. At most, one label is present per
segment. The position of the label on each segment can be moved using the contourShiftLabel
keyword. The label font, size and colour can be changed with contourLabelFont, contourLa-
belSize and contourLabelColor, respectively. The number of significant digits in the labels is
controlled by contourLabelFigs.
If a colour-filled plot is produced then the legend consists of a colour key with the range of
values for each of the fields.
A legend or key title can be specified with legendTitle. LabelColor determines the colour of
the legend title. The legendTextSize keyword controls the size of the text used for the legend
title (if any) and the legend text (the numbers). It also determines whether a legend is pro-
duced or not – a positive value automatically places a legend at the top right of the plot. A zero
size means no legend is produced.
A legend box can be added with legendBox.
The legend key can be moved by using the <legend> tag in an extratext file. This can also pro-
vide the legend title. If present, this overrides any title given by legendTitle.

9.2.7 Flow of data
The plotting of contours is carried out in three steps: (i) calculation of the regular grid of spe-
ciation data with Phreeqc; (ii) contouring of these data to produce a list of unsorted segments
denoting the contours, and (iii) sorting of these segments into their contours, adding bound-
ary segments to produce polygons for colour filling. This sequence is shown in Fig. 9.1.

9.2.8 What if Phreeqc fails?
Contour plots are best if the data are smooth and continuous and if there are no ‘missing data’
in the set of grid data to be contoured.
If there are ‘missing data’ but a value such as -99999 or -99 has been substituted as happens
when Phreeqc fails to converge, then the plot will still be attempted but will pay no special
heed to the missing value. It will be treated as a valid value and will attempt to contour around
it, probably by plotting it as a very low area or ‘hole’. This may still produce an informative
plot though it is unlikely to be worthy of publication.
If Phreeqc fails to converge and fails to produce any output, then the z-value is assumed to be
undefined (this is stored in memory for plotting) and a single undefined value (-99999.000)
written to the ‘out’ file for reploting.
A simple example is to gain another view of the solubility of Fe (total Fe = 1e-2 mol/kgw) in a system where the mineral Fe(OH)$_3$(a) may precipitate. This complements the predominance diagram produced by the input file demo/Fe/hfo.ppi. The input file would look something like:

```
SPECIATION
  calculationType                      "contour"
  calculationMethod                    1
  contourZvariable                     FeT
  xmin                                 2.0
  xmax                                 12.0
  ymin                                 -90.0
  ymax                                 0.0
  resolution                           50

PLOT
  plotTitle                            "Fe solubility\(a\) percentile contours"
  xtitle                               pH
  ytitle                               "log <i>f</i> O_2(g) (atm)"
  extraText                            extratextcontour_hfo.dat

CHEMISTRY
  # first simulation - fixed bits
  PHASES
  Fix_H+; H+ = H+; log_k 0.

  SELECTED_OUTPUT
  -reset FALSE
  -high_precision TRUE

  USER_PUNCH
  -headings FeT
  10 PUNCH log10(TOT("Fe"))

  SOLUTION 1
    pH        1.8
    units     mol/kgw
    Fe(3)     1e-2
    Na        1e-1
    Cl        1e-1
END

  # second simulation - loop on this
  USR solution 1
  EQUILIBRIUM_PHASES 1
```

Figure 9.1. Flow of data used in generating a contour plot.
The keywords controlling the appearance of the contour plot are all implicitly set at their default values as read from the \texttt{pp.set} file. The following are these default settings:

\begin{verbatim}
contours                             auto 17 p
contourFillcolor                     auto
contourLineWidth                     auto
contourLineColor                     auto
contourShiftLabel                    c
contourLabelSize                     2
contourLabelFont                     "Helvetica"
contourLabelColor                    auto
contourlabelFigs                     auto
\end{verbatim}

The plotting domain is from pH 2 to pH 12 (x axis) and from -90 to 0 atm $\log f_{O_2}(g)$ (y axis) and the z-variable is defined by FeT. The principal task for the user is to set up the calculation of the z-data. This requires defining (a) a z-variable, here the total dissolved Fe, FeT, in the \texttt{USER_PUNCH} data block, and (b) a resolution, here a 50 x 50 grid.

FeT is calculated with:

\begin{verbatim}
USER_PUNCH -headings  FeT
10 PUNCH log10(TOT("Fe"))
\end{verbatim}

The name of this column (FeT) matches that in the \texttt{contourZvariable} setting and this provides the necessary link. The value of the x- and y-variables are implicit and not required but these and others variables could also be exported, for example, with

\begin{verbatim}
USER_PUNCH -headings  x y FeT Fe(OH)3(a) water
10 PUNCH <x_axis>, <y_axis>, log10(TOT("Fe")), MOL("Fe(OH)3(a)"), TOT("water")
\end{verbatim}

If this is done, then the \texttt{USER_PUNCH} data block must be moved from the first to the second simulation otherwise \texttt{<x_axis>} and \texttt{<y_axis>} will not get updated on each iteration.

The plot produced from this input file is shown in Figure 9.2(a). The legend has been omitted. This plot has been made with auto-generated contour values and the default colouring which ranges from dark blue (low) to dark red (high). It is possible to specify any set of contour values and any set of colours.

The main task in generating a contour plot is to decide on a set of suitable contour values. With the defaults operating, this is done by dividing the z-data into 17 approximately equally-occupied intervals. This is called the 'percentile' or 'p' option as in Figure 9.2(a).

The major variation in the concentration of dissolved Fe shown by the plot occurs where Fe(OH)$_3$(a) has precipitated. Under more acidic and more reducing conditions where Fe(OH)$_3$(a) does not precipitate, the concentration of dissolved Fe is necessarily very close to that added, namely 1e-2 mol/kgw. This results in several contours with the same truncated label (-2) since the actual contour values used (from the log file) are -2.000191900824, -2.000191868200, and so on. The small but systematic variation in Fe concentration in the absence of Fe(OH)$_3$(a) precipitation is due to small changes in the mass of water which in turn reflects various hydrolysis reactions.

A second option when auto-generating a set of contour values is to start with the percentile values and then to eliminate any values which are fairly close in value to each other (within a relative difference of less than 1e-4 of the z-data range). This is called the 'simplified percentile' (or 's') option (Figure 9.2(b)).

A third auto option is to define the contours by dividing the range of z-values (maximum z - minimum z) by the number of contours, by default, 17, to give arithmetically-spaced inter-
Figure 9.2. Four options for defining contour intervals for the same z-data: (a) contours ‘auto’ option with percentile contours (‘p’); (b) same with simplified percentile options (‘s’); (c) same with empirical (‘e’) option; (d) a user-supplied set of contour values.

vals. This is called the ‘empirical’ (or ‘e’) choice (Figure 9.2(c)).

Of course, a list of any number of contour values can also be entered explicitly (Figure 9.2(d)). However derived, the set of contour values defines a set of lower class values, namely, <contour(1), contour(1)-contour(2), contour(2)-<contour(3), ... >contour(n). There will always be one more class for colour filling than the number of contours specified. Internally, an additional very large contour value is automatically added to the set of contours so that all data values will fit into one of the contour intervals defined. If a value sits exactly on a contour boundary, it is allocated to the higher class.
9.4 SOME DETAILS OF THE DATA PROCESSING

9.4.1 Algorithm

The speciation data generated by *Phreeqc* are stored in the ‘out’ file as normal. These data are scanned line by line to produce the contours. The quality of the plot obviously depends on the resolution of the grid. Normally this should be 10 or greater for draft, and 50 or greater for production.

The contouring procedure, **CONREC** (Bourke, 1987), used to calculate the contour location is a ‘local’ one and only makes use of the enclosing four grid points. This may not be as efficient as some more sophisticated interpolation procedures but has proven to be robust given some of the extreme changes in slope and discontinuities encountered in geochemical-based contour plots.

After the z-data have been generated on the requisite grid, the domain is scanned horizontally, cell by cell, top down, and the various contour crossings established. In each cell where there is a crossing, the contour level forms a horizontal plane that intersects an inclined plane formed by the z-surface. This intersection is output as a short segment and eventually all such segments for a given contour level are joined together to form the contour. The raw contour data are treated in much the same way as in a ‘ht1’ plot in that the data first undergo a line simplification to reduce the number of points. The intention is to reduce the file size without substantially changing the accuracy of the plot. This can usually be achieved with a `simplify` value of 1. Smaller values will include more points, larger values, fewer points. A value of 0 will include all of the original points, i.e. no simplification.

The simplified vectors are written to the vector (‘vec’) file along with the boundary vectors. The individual vectors or line segments are assembled into polygons (‘pol’ file) for colouring. The vector and polygon files include ‘direction’ information – walking forward along the contour in the sequence given in the file, the high side is always on the right. This is sometimes necessary to differentiate between ‘peaks’ and ‘holes’. The polygons are also listed (and plotted) in order of decreasing polygon area.

A given contour may give rise to more than one polygon if it leaves and enters the plotting domain more than once.

Finally, the domain boundary segments are added for each contour level to enable closed polygons to be formed and therefore for the polygons to be filled with colour.

Once these ‘vec’ and ‘pol’ files have been written, they are used to produce a plot.

9.4.2 Problematic cases

The contouring is normally successful but the very steep boundaries and step changes that can be produced by geochemical data, e.g. at mineral-solution boundaries, can give rise to practically convergent contours. The reverse situation can also occur: the surface can be extremely flat (and inevitably somewhat ‘noisy’) leading to a relatively large error in locating the contours. This can lead to ‘wiggly’ contours. This often happens when contour values are auto-generated with the ‘percentiles’ option. Judicious choice of contour values can usually reduce the impact of both of these problems.

These problems can lead to attempts to contour along a boundary that is particularly ‘noisy’. This in turn can lead to an excessive amount of jumping across a contour boundary and will eventually lead to a termination of the plotting with the message:

"Error: too many separate contour segments. 'Noisy' contour? Try changing contour values, e.g. auto 10 e."

This problem can usually be avoided by changing the contour values so that the positions of the boundaries do not correspond with the noisy region, for example, as suggested by choosing a set of equally-spaced ‘empirical’ contour values.
A good example of this is when contouring the variation of a saturation index of a potentially precipitating mineral in a given pH-log $f\text{O}_2(\text{g})$ domain. The saturation index will be negative over the areas where the mineral is unstable but will be zero or ‘very close to it’ where the mineral is stable. Depending on the convergence settings in Phreeqc, the value of ‘very close to it’ will vary but can be $-3e-14$ to $3e-14$ or smaller. It is entirely reasonable that the SI value can be anywhere in this range. Therefore if a contour value of $0.0$ (exactly) is chosen either manually or automatically, there are likely to be an excessive number of contour crossings as the contour attempts to track the boundary.

This is made worse in this case because of the discontinuity in the slope at the mineral precipitation boundary and the fact that the change is from a fairly steep slope while undersaturated to a slope of exactly zero when the mineral is present.

In this and similar cases, setting a contour value of exactly $0.0$ should be avoided. A small negative value, e.g. $-1e-6$, would probably avoid these problems and still identify the precipitation boundary accurately enough.

A zero (or very small) contour value is likely to be selected in this case when the auto percentile option is chosen since many of the SI values are likely to be $0.0$ or very close to it. Either set the contour values manually as described above to avoid the ‘noisy’ region around $0.0$ or choose the ‘empirical’ option for the auto selection of contour values which will probably avoid this region anyway.

In order to fill the contour levels with colour, it is necessary to closed the polygons for filling. If PhreePlot fails to close a polygon, including the domain boundaries, a message to that effect is issued and only the closed polygons plotted. The problem may disappear if a different set of contours is chosen or if the resolution is increased.

### 9.5 Modifying the Appearance of the Plot

The number of contours, their values and the overall appearance of the plot can be changed explicitly with a number of contour-related keywords. These are listed in Table 9.1 and discussed in more detail under their individual headings in Section 14.
In principle, apart from `contourShiftLabel`, the lists above have a length that is equal to the number of contours + 1. A list of these contours is given in the log file. If the specified list is shorter than required, it is recycled. If it is longer, the excess is ignored.

The property associated with each contour is then simply picked off the appropriate list based on the corresponding position of the contour of interest within the list of contours.

In this way it is easy to repeat sequences of properties, e.g. if `contourLineWidth` 0.3 -0.3, then this would alternate a full line and a dashed line.

It is also possible to add extra text, lines and symbols with `extraText` and `extraSymbolsLines` in the normal way.

Figure 9.3(a) shows the above example plotted with user-defined contour values and default values for most of the other settings. `contourFillColor` has been set to ‘nd’ in order to give a black and white plot. This could also be achieved using the `contourOptions fill FALSE` setting but in this case there would be no inline contour labels.

Figure 9.3(b) shows the same example but with some tweaking – the grid resolution has been increased from 50 to 200 to remove the ‘wiggle’ in the -2.10 contour, the label size has been reduced, the number of digits in the label reduced to 2 and the labels moved to increase legibility, the contour line width has been reduced and now alternates full line–dashed line with corresponding colours black-gray4. The settings were:

<table>
<thead>
<tr>
<th>Table 9.1. Contour-related keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keyword</strong></td>
</tr>
<tr>
<td>contours</td>
</tr>
<tr>
<td>contourFillColor</td>
</tr>
<tr>
<td>contourLineWidth</td>
</tr>
<tr>
<td>contourShiftLabel</td>
</tr>
<tr>
<td>contourLabelSize</td>
</tr>
<tr>
<td>contourLabelFigs</td>
</tr>
<tr>
<td>contourLabelFont</td>
</tr>
<tr>
<td>contourLabelColor</td>
</tr>
<tr>
<td>contourOptions</td>
</tr>
</tbody>
</table>
As with the predominance plots, it is possible to change various plotting parameters and to replot without going through the potentially slow process of regenerating the geochemical data. The various points of entry are shown in Figure 9.1.

calculationMethod = 1 uses Phreeqc to produce the ‘out’ file containing the z-grid of values based on the specified speciation calculations. It then produces the intermediate ‘vec’ and ‘pol’ data files and the plot. Since the calculations start at the beginning, any of the contouring parameters can be changed (maximum computing effort)

calculationMethod = 2 starts with the existing ‘vec’ and ‘pol’ files and uses these to generate a new plot. There is no recalculation of contours or line simplification. You can change plotting parameters including shifting the labels but not ‘calculation’ parameters such as the grid resolution or the number or values of the contour levels.

calculationMethod = 3 starts with the existing ‘out’ data file, recontours the data and regenerates the ‘vec’ and ‘pol’ files with renewed line simplification before replotting (intermediate computing effort).

Therefore, if the number or values of the contour intervals is changed or the line simplification factor altered or a different smoothing option used, it is necessary to use calculationMethod = 3 otherwise the somewhat quicker calculationMethod = 2 can be used.

If the geochemical model has changed in any way or the resolution changed, it will be necessary to regenerate the geochemical data with calculationMethod = 1.
9.6.2 Smoothing the z-data

A 'low pass' moving-average filter is available to smooth the z-data before contouring. This is invoked with one of the `contourOptions smooth=0` options: smooth=0 applies no smoothing; 1 applies N-S-E-W nearest neighbour (4) plus central cell averaging, and 2 averages all the 8 nearest neighbours plus the central cell. The smoothed data are not saved but generated internally from the raw (unsmoothed) data each time the data are plotted.

It is also possible to simplify contour boundaries with the `simplify` setting.

9.7 OVERLAPPING OR MISPLACED LABEL

One label is normally plotted for each distinct contour segment. The position of the label is normally chosen to be at the centre of the longest, 'straightest' part of the contour. This is derived from the output of the line simplification procedure if it has been used. Such a strategy often produces a satisfactory placement, but certainly does not always. For example, no account is taken of the spatial relationship between the various labels – they may overlap.

If this is unsatisfactory, the quickest change to make is to place the contour labels near the centre of their contour using the `contourOptions labelPosition=centre`.

For a given plot, the size of the contour label and its likelihood of overlapping other labels, is governed in part by the label text height (`contourLabelSize`) and in part by the number of digits actually printed in the label (`contourLabelFigs`). If PhreePlot thinks that there is not enough space to print a contour in the desired position, it will not be printed at all.

It is common, especially where there are rapidly changing conditions near mineral-solution boundaries, that two or more labels will overlap rendering them illegible. One strategy for dealing with this (other than changing the contour intervals or reducing the label size) is to move one or more of the offending labels with `contourShiftLabel`, even moving them so far that they are not plotted at all. With a little effort, it should be able to derive satisfactory label positions.

The `contourLabelSize` can also be set to 0, or the `contourLabelColor` set to 'na'. This will suppress all labelling of this contour. However, the 'shift' approach is more versatile where there is more than one label per contour since it addresses individual contour labels rather than all the labels for a given contour level.

9.8 WHAT HAPPENS IF Phreeqc FAILS DURING CONTOURING CALCULATIONS?

The contouring package expects a full set of regularly-spaced data and currently cannot deal with 'missing data'. The calculations stop and no plot is produced.
10 Custom plots

10.1 OVERVIEW

‘Custom’ plots do not have any of the intricacies of predominance diagrams or of fitting. They simply take the selected output that has accumulated from one or more Phreeqc simulations and make a plot using the columns of data found. The challenge is to get the results to produce a well-formed ‘out’ file ready for plotting.

One column has to be defined as an ‘x axis’ and all the other chosen columns are plotted using a common ‘y axis’. A secondary y axis can be chosen if wanted. The headings given in the selected output file become column labels in the normal Phreeqc fashion and these are used to select the column (‘variable’) to plot and to label it. Additional text, lines and symbols can be added to the plot through ‘extra’ text and data files.

The simplest approach is to use the User_Punch keyword data block to only send the results that need plotting, or at least, to ensure that these results are the last thing written following any output from any initial solution etc calculations. If the problem is similar to one of the examples given here, use the example as a template to get started.

Keep it simple to begin with, and use the log file with debug = 2 to take a close look at what is being done. Decide which calculations can be put into pre-loop simulations and which belong best to the main loop.

A loop file provides a flexible way of defining variables if more than one is wanted since the variables do not have to increment in any particular way, and any number of variables can be made to change ‘in parallel’. Each row in the loop file results in a separate iteration. The results are accumulated in the ‘out’ file which can be used to produce a custom plot. Data from pre-existing ‘out’ files (or other files with a similar tabular format) can be imported and added to the plot.

If no selected output file is created or the plotFactor is set to zero or calculationMethod is negative, then no plot will be created. Therefore custom-type plots can be used to do Phreeqc type calculations in the normal way without any plotting. PhreePlot has the advantage that it is possible to do some simple looping and so generate a stream of output suitable for viewing or for input to other software. For example, it is possible to accumulate all of the normal Phreeqc output from a series of runs in the *.all file (see Example 71).

10.2 PREPARATION OF THE INPUT FILE

10.2.1 Introduction

Custom plots are used for all x-y plots other than predominance plots. This includes species plots and fit plots.

A custom plot calculation normally expects Phreeqc to produce selected output data with an x-variable in the column defined by customXcolumn, and y1...yn-1 in the other columns where n=no. of columns. Data for plotting are selected from these columns using the lines and points keywords.

The ‘out’ file is the primary file used for plotting but additional files can be included by using the extradat keyword. The specified customXcolumn must be present in each file.

Selected data are copied from the main selected output to the ‘out’ file. Normally this will be
just the last line that is copied but more lines can be transferred using the selectedOutputLines keyword. PhreePlot gets the label names from the selected output header line. Only the first 198 characters will be used to define tag names.

The Phreeqc headings identifier which should be included as part of the selected_output data block has the format

```
-headings xxxx xxxx ...
```

where xxxx should not contain whitespace (spaces or tabs) or \ or /. Commas are not required as separators and so should not be included.

Values transferred with the value missingValue are treated as missing data.

If no data are found within the plot area, the label is printed in the legend but no entry is sent to the lineColor.dat file and no label is plotted on the graph.

The size of the legend labels is controlled by labelSize – the same setting as used for the labels. If a species appears in the legend but is not shown on the plot, this is because it is not found within the plot area.

Labels are taken from the names of column headers as sent from Phreeqc to the selected output using the header identifier: The labels are by default interpreted as Phreeqc species and formatted accordingly (see Section 6.4.2). This behaviour can be overridden with the convertLabels keyword. In order to avoid generating situations such as <sub>.<sub>...</sub>.</sub>, no attempt will be made to translate column headers if they contain <sub> or <sup> before translation.

The position of labels can be adjusted using a 'nudge file' (but not for contour plots).

The <x_axis> increment is controlled by xmin, xmax and resolution.

10.2.2 Controlling the scope of custom plots

Execution of custom plots is controlled by:

```
xmin, xmax etc controls the x-axis scope.
```

```
ymin, ymax does not matter as y is calculated
```

```
loopMin, loopMax and loopInt are used as an additional loop variable (the z-dimension) by the <loop> t1.inc.
```

If loopInt=0., then only one circuit round the loop is made. This uses <loop>=loopMin. This is useful for checking the input file for a specific setting. Alternatively the <loop> variable can be omitted from the CHEMISTRY section which will then only run through the loop once assuming the normal default settings.

10.3 SIMPLE LOOPING

One use of a custom plot procedure is to simply loop on a calculation many times changing one or more variables on each iteration. It is not necessary to actually produce a plot. This is something that can be awkward to do in Phreeqc at present. The Phreeqc_looping\Pespe-

```SPECIATIONcalculationType                 "custom"
calculationMethod               -1
xmin                            -10.0
xmax                            -4.0
resolution                      3
all                             T
CHEMISTRYPRINT-reset false # don't output initial solution calculation
PHASES```
Fix_H+
H+ = H+
log_k 0

SOLUTION 1

pH 2
units mol/kgw
Fe 1e-2
Na 1e-1
Cl 1e-1

# no reaction so no need to SAVE solution 1
END

USE solution 1
PRINT
- equilibrium_phases true
- species true
EQUILIBRIUM_PHASES
Fe(OH)3(a) 0 0
Fix_H+ <x_axis> NaOH
-force_equality true
END

where the pH is controlled by the <x_axis> tag which is generated from xmin, xmax and resolution. Phreeqc will be run resolution times with the values -10, -6 and -4 being substituted in turn for the <x_axis> tag.

The Phreeqc code is split into two simulations, an initialization (pre-loop) simulation and a second iteration which is the one that is iterated. all is set to T so as to create the *.all file which contains the accumulated Phreeqc.0.out output from all iterations. All of the normal Phreeqc output is first turned off with -reset false and then the species output is turned on to minimize the file size.

The calculationMethod is set to -1 so as not to produce a plot. No selected output is produced.

It is also possible to use a loop file to provide the successive values of the loop variable(s) (see loopFile). Another approach is to use the 'simulate' plot type which takes loop values from a fit data file.

One of the challenges of running custom calculations for more complex examples is to ensure that a 'well-formed' outfile is produced so that any plotting that is needed produces the desired results. This can usually be achieved by splitting the Phreeqc input part into the 'correct' number of simulations (using END's), using mainLoop to define the divide between pre-loop and main loop simulations thereby controlling the looping and selected output from these simulations, and finally using selectedOutputLines to control the number of lines (rows) of data sent to the 'out' file (Section 4.6).

10.4 CALCULATING SPECIATION ON A 2-D GRID

Although the 'grid' approach to calculating predominance diagrams and the 'contour' method both calculate speciation on a grid, they go on to do other things. The most straightforward way to simply calculate speciation on a 2-D grid is to use the 'custom' method with the x-axis and y-axis parameters both defined. Use selected_output and user_punch to define what you want output. These will then be written to the 'out' file.

10.5 MODIFYING THE APPEARANCE OF CUSTOM PLOTS

10.5.1 Overview

Many of the keywords can be used to control the appearance of the final plot (Figure 10.1). Axis scaling can either be 'auto' or can be forced using keywords such as pxmin, pxmax etc. Note the use of the x-axis scale factor, a dividing factor, which is automatically used by PhreePlot for very small or large numbers. Since this approach always produces some confusion, it
is usually better to avoid the problem altogether by rescaling (changing units) in the selected
output to bring the scale somewhere in the range 1e-3 to 1e3.

A second, independent y-axis scale can be used for the right-hand y axis. Point and lines that
use this scale are specified with `points2y` and `lines2y` in the same way as for `points` and `lines`.

Tags can be used to give super and subscripted text, italics, bold or line breaks. Single Greek
characters can be entered with the \letter notation or more generally Greek text can be added with `\g` . . . `\g` (Section 7.6.3).

Additional data from other files can be added to the plot using the `points` and `lines` keywords
combined with the `extradat` keyword to add the additional files to the search path. These data
must be in regular tabular output format. `extraSymbolsLines` can be used where more control
is wanted over the symbols used or the line widths.

![Figure 10.1. Some of the keywords used to control the appearance of custom plots.](image)

10.5.2 Customising the plot

Providing `legendTextSize` is greater than zero, the default is to print a simple legend just out-
side of the plot area close to the top right corner of the plot. This legend provides a key for the
plotted lines and symbols. The position of the legend can be moved inside or outside the plot
area using the `<legend>` tag in the `extraText` file. `auto` for x or y coordinate position chooses
the default value.

The labels associated with each item in the legend are derived in various ways. Most simply
they are generated from the column names in the plot data file from which the plotted data
were derived. Often these are derived from the `\out` file which is automatically generated from
the Phreeqc selected output. The labels appear in column output order which itself is deter-
mained by the `\punch` order. The data and labels used can also be derived from another file pro-
vided it is in the tabular out file format and that it is defined in the search path given by `extradat`.
Where there are completely blank lines in the plot data file, these generate line breaks and each line is labelled separately in the legend. Consecutive blank lines count as a single break. Comment lines do not produce breaks. By default, the column name is appended with 
"_n" where n starts at 1 and increments by 1 for successive datasets.

These names can be replaced with your own names by using the labels keyword. This provides a list, one name for each successive value of the loop variable. This list is recycled as needed, or names can be read from a loop file or from the plot data file used in simulations and fitting. If only one loop name is given, then the plot data file providing the data plotted is searched to see if this name appears as a column label. If it does, then this column is used to provide the legend label for that dataset. Since only one name is needed per dataset, this is taken from the first row of each dataset. Variable values can also be ‘posted’ next to plotted points using the post keyword.

A legend is automatically placed to the right of a plot if the legendTextSize is greater than zero. It is also often useful to include a heading for the legend. This can be done using the legendTitle keyword or the <legend> approach in the extraText file which can also be used to move the legend to somewhere else. Any text in the text string that precedes <legend> is used as the legend title. This can include text enhancement tags and line breaks. If a dataset has the ‘na’ colour then that dataset will not be drawn and no entry in the legend will be made.

Line and point colouring can either be left to PhreePlot or defined in the line colour dictionary. Line curves are automatically labelled if labelSize is greater than zero and the labelColor is not ‘na’. When there are many overlapping lines finding the ‘optimal’ label placement can be slow. The optimization can be turned off by reducing labelEffort to zero. Label positions can be micro-adjusted using a ‘nudge file’. A red ‘tracking’ symbol is used to show the label anchors. These can be turned off by setting trackSymbolSize to zero.
11 Species plots

11.1 What is special about a ‘species’ plot?

A ‘species’ plot is a special type of custom plot which shows the distribution of species for a
particular element in terms of their concentration or percentage distribution versus some master
variable, such as pH. The element is specified with the mainspecies keyword. The percentage
is calculated in terms of the moles of an element in a given species as a percentage of the
total number of moles of that element present in all species.

Often the mainspecies is a chemical element such as ‘Fe’ but there are a number of special
return subsets of the system (see the SYS() function in ‘The Basic interpreter’ section of the
Phreeqc guide for details of each of these subsets); (ii) ‘all’ which will plot the percentages or
concentrations of all species for all elements. These options provide considerable flexibility in
the subsets of species included.

Two common types of species plots are shown in Figure 11.1 and discussed in more detail in
the Examples section (Example 73 and Example 74)

![Species plot](image)

Figure 11.1. Two common types of species plots produced using the ‘species’ method showing the distribution of Cd species with pH. The two plots use different ‘include’ files to generate the selected output.

A species vs pH plot is obtained by setting calculationType to ‘species’ and by including one
of the special ‘include’ files in the CHEMISTRY section. An example of using the ‘species-
vsph.inc’ include file is given in Figure 11.1 (left). The ‘logspeciesvsph.inc’ file is used to
generate a plot of log species concentrations vs pH (Figure 11.1, right). These files can be
edited to filter out unwanted species – H and O species which normally dominate aqueous
systems are filtered out by default.

It was necessary to make the species plot a special type of custom plot since Phreeqc cannot
automatically generate a set of heading names (column titles) at run time to correspond with
the species found. Therefore it is necessary to write the species name to the selected output file.
along with each value. Furthermore, the \texttt{SYS()} function that is used to extract the set of species present and their concentrations is written to the selected output file in order sorted by concentration rather than sorted alphabetically. This means that the sort order is likely to vary and must be reordered to enable species distribution curves to be plotted.

The 'species' method expects a specific type of input to be returned in the selected output. This consists of a series of species name-value pairs. The species names are used for the labels while the values are plotted. The \texttt{customXcolumn} should be set to the column where the x-axis variable is located. The other pairs of columns are assumed to contain y-values (species percentages or concentrations) to be plotted. For example, if the x-axis variable is \texttt{PUNCH}ed first, then the \texttt{customXcolumn} would be set to 2 and the default name of the x axis is taken from the name given in column 1. The remaining columns are the names-species pairs to plot.

Six \texttt{include files} for making species plots are included in the system directory. These are for analysing solution species or adsorbed species and report in either relative concentrations (\%) or in absolute concentrations (mol/kgw), see Table 11.1. The files can easily be edited to customise their behaviour. For example, instead of pH as the independent variable, other variables can be chosen, or the files can be edited to exclude various species from the plot.

<table>
<thead>
<tr>
<th>Name of file</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{speciesvsph.inc}</td>
<td>\texttt{&lt;mainspecies&gt;} specifies the 'element', or 'elements', to analyse, e.g. &quot;Cd&quot; or &quot;aq&quot;, &quot;phases&quot;, &quot;elements&quot;, &quot;surf&quot;, &quot;s_s&quot;, &quot;gas&quot;, &quot;all&quot; for all species for a specific element or aqueous, solid, etc species. Output is in terms of the relative concentration (in %) of all species, including minerals and adsorbed phases, containing the main element vs pH. If the \texttt{&lt;mainspecies&gt;} is &quot;elements&quot; then the analysis is done at the element level for all elements.</td>
</tr>
<tr>
<td>\texttt{logspeciesvsph.inc}</td>
<td>As for \texttt{speciesvsph.inc} but the output is in terms of the log$_{10}$(mol/kgw) concentration of each species.</td>
</tr>
</tbody>
</table>

These files pick up the main species from the \texttt{mainspecies} setting and use the \texttt{SYS()} function to obtain a list of the species present for this 'element' and their amounts (in moles). These are then either converted to percentages of the total amount present or converted to log concentrations and punched in name-value pairs to the selected output file which, in turn, is copied to the \texttt{out} file, one row per pH.

There can be more than one main species. A separate plot is produced for each.

The column headings in the 'out' file are those given by \texttt{Phreeqc} for unnamed columns, namely, \texttt{no_heading_1} for column 1 and so on. The species are written to the 'out' file starting with the most abundant species followed by all the other species in order of decreasing abundance. The order of the species written will tend to vary with chemistry of the system.

A sorted table of the speciation results is written to the \texttt{pts} file if the 'pts' setting has been set to \texttt{TRUE} in one of the main input files. This can be used for plotting the results with other software or with the \texttt{lines} or \texttt{points} keywords.

\texttt{PhreePlot} automatically displays the distribution of all species as lines. No \texttt{lines} line is required in the input file. The colour of the lines drawn is determined by the normal line colouring algorithm, i.e. using auto colour selection or the line colour dictionary depending on the \texttt{useLineColorDictionary} setting. The line width is controlled by the \texttt{lineWidth} setting. The \texttt{minimumYValueForPlotting} setting is useful to remove minor species from the plot.

If \texttt{mainspecies} is set to "elements", the distribution is over the total concentrations of all dissolved elements and valence states (solution master species) in the system rather than for the concentrations of all species of a particular element.

The 'adsorbed' option considers the distribution of adsorbed species vs pH either for just one element or of all elements, as determined by the \texttt{mainspecies} setting.
A list of main species can also be specified to give multiple plots. The multipageFile setting controls whether all the ps and pdf plots produced will be combined into one file or not.

If the main species is set to 'all', then the percentages or log concentrations of all species of all elements is plotted (with the exception of certain species including water - see the Basic code in speciesvsph.inc and logspeciesvsph.inc). Percentages are calculated as the percentage of the total amount of the main 'element' being considered (calculated with the SYS() function). It is possible to filter out other unwanted species by modifying the Basic code in the *.inc files.

Much the same effect can be achieved using an ordinary custom plot but here the sorting has to be done with basic code in the user_punch block (Example 60).

The 'species' calculation method discussed above can be modified to plot a variety of things providing the selected output is set up to produce a fixed set of name-value pairs.

Note that in Figure 11.1 (left) the curves for several minor species have been omitted for clarity. This was achieved by setting minimumYValueForPlotting to a value of 5. Only curves with a maximum value exceeding 5% are plotted.

If the x-axis variable wanted is not pH but some other variable then it is straightforward to edit the include files to ensure that the required variable is the first one to be punched to the selected output, and to associate customXcolumn with it.

It is possible to make multiple plots by using the loop variable.

11.2 MODIFYING THE APPEARANCE OF SPECIES PLOTS

If the default x-axis title is for a species plot is set to 'auto' then the x-axis title is set to the name of the customXcolumn, e.g. "pH" and that for the y-axis title is either "% in species" for a linear scale plot or "log concentration (mol/kgw)" when a log scale has been inferred, i.e. when minimumYValueForPlotting is less than or equal to zero and there are no 2y axis variables to plot. The plot title is automatically set to "distribution of <mainspecies> with pH". All of these defaults can be overridden from the input files.

Species plots are produced with the custom plot procedure and so all the same keyword settings discussed for custom plots apply. Many of these are illustrated in Figure 11.2.

There can be a bewildering array of lines in species plots and it may help if the labelling is coloured as well as the lines. This can be achieved by setting labelColor to 'auto'. Lines and labels will then have the same colour in both plot and legend.

11.3 ADDING OTHER VARIABLES TO A SPECIES PLOT

It is possible to plot other variables on a species plot.

Add any additional variables to the list of punch'ed items in the speciesvsph.inc file. They can be added anywhere providing the x-column remains the first to be punch'ed. Normally, they would be added before or after the actual species have been punch'ed. This makes sure that they will be included in the outfile. Thereafter they are treated as if they are ordinary species, i.e. they will be plotted on the main y axis.

The column names in the outfile from a species plot will be in pairs beginning no_heading_1, no_heading_2 .... for the most abundant species followed by the other species in order of decreasing abundance.

For species type plots, a table with fixed column positions is required. This is given in the 'pts' file. This file is automatically added to the search path and so the indicated species can be plotted with the lines or points settings.
Cd speciation (using speciesvspH.inc)

\[
\begin{align*}
\text{Cd}^{2+} & \\
\text{CdCl}^+ & \\
\text{CdNO}_3^+ & \\
\text{CdOH}^+ & \\
\text{Cd(OH)}_2 & \\
\text{Cd(OH)}_3^- & \\
\text{Cd(OH)}_4^{2-} & \\
\text{CdOHCl} & \\
\end{align*}
\]

\[\text{pH} \]

\[\% \text{ in species}\]

Figure 11.2. Some of the keywords used to control the appearance of custom plots.
12 Fitting and simulations

12.1 INTRODUCTION

Calibrating models is a key part of modelling but can be rather daunting in the beginning. Fitting with PhreePlot definitely belongs to the ‘advanced’ category of tasks. You have to get quite a few things working correctly together to be successful and while it would be nice to have a ‘fire-and-forget’ approach for this, it is rarely that straightforward.

The setup for fitting is very flexible. Each observation can be simulated using either the same block of Phreeqc simulations (containing variable tags) or a different block of simulations as specified in the fit data file. This enables different models to be applied to different observations – a form of ‘global’ optimization. For example, several ‘metals’ could be fitted to a similar model, or each ‘metal’ could be fitted to a different model, at the same time. It is also possible to globally fit parameters by combining data from potentiometric titrations for proton binding with sorption data for metal binding, for example.

The overriding principle is that each line in a fit data file corresponds with an observation and that these must be paired one-to-one with corresponding calculated values in the ‘out’ file. Then there must therefore be ways of defining how to calculate the expected values for each observation and of weighting the various observations, or more specifically, the residuals.

PhreePlot includes a number of easy-to-use optimization algorithms that can be used to fit models to data, i.e. they automatically adjust a set of parameter values defined in a Phreeqc input file to their optimal values based on a set of observations (data) and an objective function to provide a measure of the overall goodness of fit. The objective function used here for this is the weighted sum of squares of the residuals.

The algorithm to be used is chosen with fitMethod. The five optimizers currently available are:

(i) ‘nlls’: this modified Levenberg-Marquadt procedure is described in Powell (1965). The version used here is based on the VA05 routine from the Harwell Subroutine Library (HSL) Archive;

(ii) ‘lm’: the Levenberg-Marquadt procedure as implemented in the MINPACK-1 package by More, Garbow and Hillstrom. It uses a forward difference method for approximating the Jacobian.

(iii) ‘newuoa’ (NEW Unconstrained Optimization Algorithm): this is not specifically a least squares optimizer although it is used here as such. NEWUOA is a general-purpose optimizer that uses a quadratic model to approximate the objective function in a ‘trust region’ (Powell, 2007). The ‘trust region’ is a restricted part of the objective function in which the quadratic approximation is ‘trusted’ to be correct. This region is progressively enlarged as the approximation improves. NEWUOA is robust and has proven to be capable of dealing with a large number, i.e. hundreds, of adjustable parameters. It has been largely superseded by BOBYQA.

(iv) ‘bobyqa’ (Bound Optimization BY Quadratic Approximation): BOBYQA is similar to NEWUOA except that it allows lower and upper bounds to be specified for each adjustable parameter (Powell, 2009).

(v) ‘subplex’ (Simplex algorithm which searches in subspaces) by Rowan (1990) is a generalization of the Nelder-Mead simplex method. It is said to be well-suited for situations in which the functions are noisy and discontinuous at the solution.
A list of these methods can be given to fitMethod and then the methods will be run sequentially on the same data set with all settings the same (although some parameters may take on somewhat different meanings).

All of these methods are derivative-free meaning that you do not need to provide functions to calculate the derivatives. This makes the fitting of new models much easier – and even possible – but there is a penalty in terms of speed of execution and more importantly, stability. When numerical derivatives are themselves calculated from a numerical model which itself carries estimation errors, as all Phreeqc calculations do, care has to be taken to ensure that the derivatives are obtained with sufficient accuracy to be useful. In most of the above implementations, this is largely outside of the control of the user but can be controlled to some extent by the convergence criterion chosen and with the 'nlls' optimizer by the fitFiniteDiffStepSize parameter.

The parameters to be optimised, and the independent variables used, are identified using tags defined by the user and placed in the Phreeqc input code of the Chemistry section of the main input file. The adjustable parameters are each systematically varied in such as way as to find the minimum value of the objective function. This should correspond with an optimal fit between calculated and observed values of the dependent variable(s) taking into account the weight assigned to each observation.

Weighting options are: unit weighting (all weights are automatically assigned a value of 1), relative weighting (weights are 1/observed value), or the weights for each observation can be read from the data file.

The calculations can be run either in simulation ('simulate') or fitting ('fit') mode. Simulations (calculationType 'simulate') in the present context mean using the same setup as a fit input file but only calculating the 'dependent' variable once for each observation rather than using these values to refine the values of the adjustable parameters. This is useful to get some idea of what results the initial set of parameter values will give.

The simulation mode is also useful where a particular Phreeqc input file is wanted to be run with various sets of parameter values or variables (sometimes called 'independent variables'). These values can be entered in the 'data file' and picked up from there with each row in the data file corresponding to a separate simulation. The headings of the data file are used to generate tag names. The corresponding tags are used to mark the position of a substitution in the input file.

If the number of degrees of freedom in a fit file is 0, i.e. there are the same number of adjustable parameters as data points, then least squares optimization is not appropriate as there is an exact solution. PhreePlot assumes that the problem is then one of 'root finding' and will attempt to find the solution by iteration. This also applies when no data file is given, i.e. the fit data file name is blank. In this case, the 'target' value(s) is/are automatically set to zero.

12.2 FITTING IS SPECIAL

Fitting includes its own form of looping and is outside the normal looping built into PhreePlot. So the x-, y-, z (loop) and mainspecies loops do not operate. And as described below, the way that it treats multi-simulation input files is also different from normal. The division between pre-loop simulations and main loop simulations is based on a subset of simulations, potentially unique to each data point, rather than the whole set of simulations. This provides maximum flexibility in the way that different types of data can be globally optimized.

12.3 APPROACH TO FITTING

There are two distinct components to fitting:

(i) defining the objective function, i.e. defining what is to be minimized subject to any constraints on the values that the function variables ('adjustable parameters') may take. Here we use a sum of squares of the weighted residuals. Other functions are possible;
(ii) adjusting the given set of variables in such a way as to locate an acceptable minimum value of
the objective function;

Given (i), the challenge of (ii) is simply defined but finding fast and reliable approaches has
exercised numerical analysts for a long time. All optimizers should in principle converge to the
same minimum given the same (i).

In our case, the objective function itself consists of three components:

(i) the chemical model used to calculate the value of the dependent variable(s);
(ii) the corresponding observations of the dependent variable(s);
(iii) the weighting applied to each observation or more particularly to the residuals (the difference
between (i) and (ii)) for each observation.

Again, given a well-defined chemical model (including the thermodynamic database) and the
same weighting scheme, all fitting programs should in principle converge to the same final
solution – the ‘best’ fit. This can be tested by comparing the values of the fitted parameters
and the individual residuals. The relative degree of success of fitting by various procedures can
be judged by a measure of the overall ‘goodness of fit’ such as the root mean square error
(RMSE) or coefficient of determination (R²). The aim is that the fit should at least be better
than assuming the mean value of the dependent variable throughout – it can of course be
worse giving an at first seemingly improbable negative R².

Ultimately, the details of exactly how the model calculated the individual contributions to the
objective function should make no difference. It is assumed that this has been done reliably,
though of course this is a critical assumption, and is itself a measure of model quality.

Pitfalls in the fitting involve converging to a local not a global minimum, and not finding a
unique minimum perhaps because of two or more highly correlated variables. The latter situation
leads to the optimiser ‘wandering along a long flat valley bottom’ and is a sign of an over-
parameterised model. This does not mean that the model is wrong, just that there are insuffi-
cient data to uniquely define all variables. The usual solution to this is to reduce the number
of variables by fixing one or more of the variables at an acceptable value, or by adding one or
more constraints on the values that the variables can take ('constrained optimization').

12.4 PRACTICAL SETUP

12.4.1 Approach

It is helpful to break down the overall task into several more manageable sub-tasks:

1. Organise your data

A file needs to be made containing the data (‘observations’) that are to be fitted. This
should be an ASCII-coded flat file. It can be easily prepared in a spreadsheet and
exported in tab or csv format. It should have one line per observation with columns
which include all the independent variables. The first line should be a header contain-
ing the column labels – keep these simple by avoiding spaces and other special charac-
ters, e.g. use only upper and lowercase characters, numbers, the dollar sign (‘$’) and the
underscore (‘_’). The data should start on the second valid line (not counting com-
ments and blank lines). Columns can contain either numeric or character data (not
mixed within a column). By default, all columns are assumed to be numeric. If the last
character of a column label is a dollar sign, then that column is assumed to be a charac-
ter column. If a variable in this second line was assumed to be numeric but a character
(non-numeric) value is found in the first data line, then the whole column will be
assumed to be character. If necessary, rearrange the order of rows or insert a dummy
‘format’ line to make sure that this is so. Also decide which data separator(s) is (are) to
be used and tell PhreePlot either with the dataSeparators keyword or with the optional
second parameter of the dataFile keyword. Beware of tabs and other ‘invisible’ charac-
ters in your data file. If you want multiple consecutive tabs to signify missing data set
the data separator explicitly to "\t" although a complete line of tabs with no other characters is always treated as a blank line. If you want tabs and spaces to be treated equally, and multiple tabs and spaces to be treated as a single separator, use "\t". This is often an appropriate option.

If additional lines need to be included but not used in the calculations, turn them into comments by making the first non-blank character a hash (#).

When reading entries, quotes are stripped from the entry and the entry is then read as numeric or character according to the format established above. Non-numeric values are assumed to be character. Quotes alone do not define a character value, e.g. "1.23", "4.56"..., will be interpreted as valid numeric values.

Empty fields are given the missingValue if the column is designated numeric or as the empty string ("") if designated character. An ‘na’ in a numeric field is also given the missing value. The missing value is treated as a valid value in calculations unless it is assigned the value of UNDEFINED (99999) in an input file. The default missing value is set as UNDEFINED.

If fitting, you will need to decide how you want to weight the observations and if necessary include a 'weights' column.

2. Get the chemical model working

   Each observation is associated with a block of simulations, i.e. a contiguous set of Phreeqc simulations. Each observation can refer to the same block of simulations (with some tag(s) within the block varying the actual code for each observation) or each observation can have its own distinct block of simulations, or anywhere between these two extremes. This enables global optimization – fitting essentially unrelated datasets/models all at the same time.

   Each block is split into two parts:

   (i) zero or more pre-loop simulations

   (ii) one or more main loop simulations.

   All the pre-loop simulations for all the specified blocks are run sequentially just once per run before the fitting proper begins. This enables static code such as loading databases or preparing solutions to be executed just once. Where the model is similar for all observations, it is often convenient to associate pre-loop simulations with just the first observation. The blocks of code for each observation and the pre-loop/main loop division are specified along with the observations using the blockRangeColumn and mainLoopColumn in the fit data file as described below.

   Each observation and its block of Phreeqc code will include tags specifying the independent variables and will ultimately output the value of the dependent variable in a USER_PUNCH keyword block. Each block of code will therefore usually contain at least the SOLUTION and USER_PUNCH keyword blocks. Check that the code is working properly by checking the calculation for a single point. It might be convenient to develop this using Phreeqci after manually substituting the tags with reasonable values. The column headings defined in the USER_PUNCH data block are used to name the columns in the output files (and so can be used for identifying which columns to use for plotting) as well as for telling PhreePlot where to expect the calculated value of the dependent variable. Columns can be referred to by column name or column number.

   If debug is set to 2, then a table showing the Phreeqc simulations that are executed as pre-loop and main loop simulations is sent to the log file. This contains an entry for each observation.

3. Make up the PhreePlot input file

   If possible, choose an existing input file of a similar fitting problem as a template and edit accordingly. This will remind you of the parameters that need to be considered. You will need to define the name of the data file (dataFile), the column in that file con-
Fitting and simulations

12.4.2 Flow of data and information during fitting

A summary of the overall flow of data and information during fitting is shown in Figure 12.1. Data are read in from the data file (dataFile). If necessary, the data are transformed before being stored in memory (logVariableIn). These data contain values for the dependent variable (for a fit), the weight to be applied to each observation if applicable and any independent variables needed in the calculations. It can contain additional columns of data but these are ignored. The column headings are used to define special ‘fit data’ tags. These are used in the CHEMISTRY part of the input file to identify the variables to be substituted when calculating the value of the dependent variable. It is also necessary to identify which column contains the dependent variable. This is done by the column position or name (dependentVariableColumnObs). The Phreeqc simulation(s) used for each data point are indicated by the integer value

taining the observations (dependentVariableColumnObs) and the column in the selected output file containing the value of the dependent variable (dependentVariableColumnCalc). Each column in the data file is converted into a tag which can be used in the Phreeqc model code to indicate where a substitution needs to be made.

Decide which fitting algorithm to use ('fitMethod') and which variables need to be defined as model parameters – usually this is all the parameters that may need to be adjusted at some stage – and define the model names and other parameter switches accordingly (numberOfFitParameters, fitParameterNames, fitAdjustableParameters). The numberOfFitParameters should be set to the number of distinct parameters specified by tags in the input file – it should include both fixed and adjustable parameters. Set the initial parameter values for all parameters (fitParameterValues) and the bounds on these parameters if appropriate. It is important that this combination of parameter values works so if necessary check with Phreeqc or by other means. You may have to fine tune some of the fitting algorithm's operational settings controlling such things as the step size, the maximum number of iterations etc. Finally decide what the most diagnostic plot will be and define customXcolumn and points accordingly. Any columns defined in the 'pts' file can be used for plotting including the automatically-generated 'observed', 'calculated' and 'residuals' columns. Refinements include adding extra text (extraText) and labelling individual points using post.

Each data point will have to be associated with a block of one or simulations which will be used to calculate the fitted value for that observation. It is best to (a) identify all 'constant' Phreeqc blocks, i.e. those blocks that are always the same and do not change – these can be put in a pre-loop simulation; (b) identify all Phreeqc blocks that are constant for all observations (data points) but which may vary during fitting – these can be included in the simulation for observation 1; (c) identify the simulation or range of simulations that will be used for each observation – these should mainly contain non-constant data blocks, i.e. those containing a tag that will vary. This can include extra simulations too.

For maximum speed, in step (c) above, associate all of the observations with the same range of simulations, e.g. 2-11 and use the onePass TRUE option. This means that all the simulations will be calculated in a single call to Phreeqc.

4. Run the file

Run the file using debug = 2 or 3 if necessary. You may need to adjust some of the fitting parameters particularly fitFiniteDiffStepSize. The R2 value is a guide to how well the model is actually fitting the data. If the model seems to be running properly but PhreePlot refuses to adjust the adjustable values, check the model and data carefully, e.g. by looking at the table of observed and calculated values near the end of the log file. This type of failure is usually because the model cannot work out how to improve the fit and is often a sign that there is something wrong with the model. When you think you have a good fit, confirm its uniqueness by starting from a different set of initial parameter values, or even by using one of the other algorithms.

12.4.2 Flow of data and information during fitting
or integer range found in the column of the data file given by the blockRangeColumn. These simulations can be shared by different observations or can be different for each observation. There can also be a column in the fit data file to indicate which simulations within each block range are pre-loop simulations and which are the main loop simulations. Like the blockRangeColumn, this can be specified separately for each observation. The name of the column is given by the mainLoopColumn keyword. If this is not specified and the mainLoop keyword has a valid setting, then this setting is used for all observations. This setting is relative to the start of the block of simulations used for each observation.

The default setting of mainLoop is auto which for fitting and simulations is set to ‘1’, i.e. all simulations will be run on each iteration. Remember that ‘1’ refers to the first simulation in the specified block not the first simulation in the whole set of simulations.

All pre-loop simulations are executed only once per run – at the very beginning of the run. Pre-loop simulations should include static data such as database blocks. If there are data blocks that vary but apply to all data points in a given set of function evaluations and so only need to be run once per iteration, include them in the block of simulations specified for the first data point. The simulations specified for the second and subsequent data points should only refer to the simulations specifically required for those data points.

The input files define the number and names of parameters used by the chemistry model to calculate the value of the dependent variable (numberOfFitParameters and fitParameterNames) for a given set of conditions. Parameters can be fixed or adjustable (fitAdjustableParameters) and may be fitted as log parameter values (fitLogParameters). The parameter values set (fitParameterValues) are either used as initial estimates if adjustable or as fixed values.

The CHEMISTRY section contains the code that is used to calculate the dependent variable and this is normally output via the main selected output ‘file’. The column for this is given by dependentVariableColumnCalc and whether it should be transformed or not by logDepVariable. The observations from the data file are compared with calculated value of the dependent variable and the difference (the ‘residual’) multiplied by the given or calculated weight (fitWeightingMethod and weightColumn). This weighted residual is passed to the optimizer.

The optimizer adjusts the adjustable parameters, identified by fitAdjustableParameters until the convergence criterion of some other factor signals an exit. After convergence, summary statistics are calculated, some tabular output produced and a plot made. The plot is made from the ‘pts’ file with the columns used specified by the lines and points keywords. Data from other files, including the ‘out’ file, can be used by adding the relevant files to the search path with extradat. The value of customXcolumn controls the x-axis variable. If no satisfactory convergence is achieved, a message is issued accordingly.

12.4.3 The parameters

Parameters are variables that remain constant during a simulation, i.e. they have the same value for all observations with a given data set. Parameters can either be fixed or adjustable depending on whether they are to be adjusted by PhreePlot to provide the best fit or not.

The number of parameters should be defined first using the numberOfFitParameters keyword. Each parameter has various attributes associated with it such as its name, its value and whether it is fixed or adjustable, whether the log of the parameter should be optimized, and any constraints (lower and upper bounds). These are specified by a series of lists, one entry per parameter.

Unlike most other settings, the order of the numberOfFitParameters keyword in the input file is important. Specifically this setting should always come before any of the other parameter list settings in the input file since it automatically re-initialises all these other parameter settings to arrays of the specified length and with the system default values. This is to ensure that all the parameter lists have the correct length. The parameter list settings (all have length of numberOfFitParameters) are: fitParameterNames, fitLogParameters, fitAdjustableParameters, fitParameterValues, fitLowerParameterValues, and fitUpperParameterValues.
Figure 12.1. Flow of data and information during fitting.

The parameter tag names are defined by the fitParameterNames keyword in an input file, one name for each parameter. Once defined, these names can be used as tags in the Phreeqc input file (the tag is generated by enclosing the name in angle brackets). These names must not be used for other tag definitions. The values assigned to these parameters is determined by the fitting procedure in PhreePlot.

The initial values of each of the parameters are defined by the fitParameterValues keyword. fitStepSize controls the largest permissible change in a parameter value during an iteration for the 'nlls' algorithm. It defines RHOBEG, the initial radius of the 'trust region' in the 'newuoa' and 'bobyqa' algorithms. In the 'lm' and 'subplx' methods it defines the initial step bound. The parameter values are either fixed or automatically adjusted by PhreePlot to provide the best possible fit. This option is controlled by the fitAdjustableParameters keyword.

12.4.4 Variables

Variables can vary for each observation within a given dataset. The dependent variable is the variable that is calculated from the model(s) and a combination of the given parameters and
the independent variables. It is the variable which is fitted when optimisation is being undertaken. Fitting minimises the weighted sum of squares of the differences between the observed and calculated values of the dependent variable. It is assumed that the independent variable contains all the errors, both errors of observation and model errors.

There can only be one dependent variable per line of input in the data file plus any number of independent variables on the same line giving the fit input file a tabular or spreadsheet layout.

The independent variables are the variables that are fixed by the user and control the value of the dependent variable. It is assumed that the independent variables are known without error. It is possible to have no independent variables as in a ‘root finding’ problem or where the model itself varies from observation to observation.

12.4.5 Passing the fitted values from Phreeqc to PhreePlot: preparing the input file

There are limitations to the structure of the Phreeqc input file that can be used during fitting. This particularly relates to the use of multi-simulation input code and the way in which the value of the dependent variable is calculated.

There are two principal calculation options: either one pass through the Phreeqc code is made to calculate a single value of the dependent variable, or all dependent variable values are calculated in a single, multi-simulation pass. There is no half-way house. The latter is computationally faster but the input file is more complex and less flexible since it must either include a separate simulation to generate each data value or must include a Phreeqc keyword that generates a sequence of data via its own iterations, e.g. REACTION and TRANSPORT.

Typically many iterations (or function evaluations) are made through the whole data set during fitting and so the former approach will make \( n_{\text{data}} \times n_{\text{iterations}} \) calls to Phreeqc while the latter approach will make just \( n_{\text{iterations}} \) calls.

The difference between these two approaches can be seen in the iso (Example 80) and ison (Example 81) examples.

One pass to generate a single data value

When onePass is FALSE during ‘simulate’ or ‘fit’ calculations, only one data value should be written to the selected output at a time. This may be because the whole input file must be repeated each time in order to calculate one value or because a different block of simulations is used to calculate each value. If there is more than one line of selected output data, the last line in the selected output is always used.

The SELECTED_OUTPUT keyword block (optionally plus a USER_PUNCH block) should generate at least one line of data in the selected output file for each calculation (i.e. after “any initial solution, initial exchange-composition, initial surface-composition, or initial gas-phase-composition calculation and after each step in batch-reaction or each shift in transport calculations”).

The PhreePlot keyword selectedOutputLines is not used since only one line will ever be picked up. The PRINT -selected_output statement (or similar in SELECTED_OUTPUT) can be used to control which Phreeqc simulations send data to the selected output file and so which simulation actually sends the last line of data.

The dependentVariableColumnCalc keyword controls the column where the dependent variable will be found.

The Phreeqc code can include several simulations, for example, different types of simulations can be combined into a single global optimization, but only one can be used to generate the value for each data point. The block of simulations to be used are set by the value or range set in the column defined by the blockRangeColumn keyword providing onePass is set to FALSE. This means that completely different models can be used to generate the various values of the dependent variable for each data point.
One pass to generate all data values

When onePass is TRUE, all of the data values is expected to be written to the selected output file in a single pass through the designated block of Phreeqc code. If there are more lines of data than required to pair off with the number of observations in the fit data file (\(n\)), then the last \(n\) lines of selected output will be selected.

It is up to you to ensure that the USER_PUNCH code does actually produce the required output. The 'one pass' option is most often used for REACTION, KINETICS, ADVECTION and TRANSPORT calculations where Phreeqc has its own built-in iterators. Here a single call to USER_PUNCH transfers all of the required output values. Alternatively, a whole set of different simulations can be run in one call to Phreeqc which will also result in a multi-line block of selected output containing all of the required dependent variable values. This approach uses the mainLoop setting to select more than the last simulation to iterate on.

Since the onePass TRUE option along with the default oneSimulationAtATime switch of the mainLoop keyword set to FALSE means that all the main loop simulations are expected to be produced in a single call to Phreeqc, there is normally no updating of tag values between simulations. If variables need to be passed from one simulation to another, use Phreeqc's own PUT/GET mechanism or set both the onePass and oneSimulationAtATime switches to TRUE.

The PRINT -selected_output statement can be used to control which Phreeqc simulations send data to the selected output file and so which simulation actually sends the last line(s) of data.

In the case of a fit, the 'auto' option controlling the number of selected output lines is set to one when onePass is FALSE and to the number of data points when onePass is TRUE.

The selected output produced by a given input file can be easily viewed by setting debug to 2 – the output is written to the screen and also written to the file selected_1.0.out (default name) which will be found in the working directory.

Skipping unwanted Phreeqc selected output

A USER_PUNCH block may lead to unwanted lines of data being sent to the selected output file an so prevent the 1:1 match between the sequence of observations in the fit data file and the computed values in the selected output file.

Excess lines of selected output are automatically deleted counting from the bottom upwards and so some initialization output may not matter. However, sometimes unwanted data are produced in the middle of an output file and so would matter.

It is often possible to suppress unwanted output by either turning off all selected output for a simulation with the PRINT; -selected_output FALSE statements or for individual simulations by including a test at the beginning of a USER_PUNCH block followed if necessary by a jump over the PUNCH statement. For example, testing for valid values of STEP_NO or SIM_NO can avoid inclusion of the output from initial solution calculations.

Where this is not possible, e.g. in TRANSPORT calculations with onePass set to TRUE, it may be possible to amend the input data file containing the observations so that it includes dummy (placeholder) values to match the unwanted lines of output and then to include weights of zero for these unwanted 'observations'. They will then not affect the fitting. The 1:1 matching of lines of input and output is a prerequisite for fitting with PhreePlot.

Global optimization - switching models

Where global optimization over various data sets and models is to be undertaken, then there needs to be a way to switch the code (model) used to give the calculated values in the USER_PUNCH block.

This can be achieved in various ways depending on the degree to which the models vary.

(i) Using an independent variable set in the fit data file
where the fit data file looks like this

observed  weight  solute
# Ca
-5.00E+000  1  Ca
-5.10E+000  1  Ca
-5.00E+000  1  Ca
...

# SO4 - the blank line in the data file above breaks the line in a plot
-5.20E+000  1  Mg
-5.10E+000  1  Mg
-4.98E+000  1  Mg
...

As each of the observed values is read from this data file, a tag is made of the other variables
(columns) in the file, namely <weight> and <solute>, and the values of these changed in parallel
to correspond with each observation. The value of <solute> is then substituted in the
USER_PUNCH block above. This file could also define one or more numeric values to use, potentially
different for each observation.

The corresponding weight is not transmitted through its tag value but by naming the weight
column with the weightColumn keyword.

The 'IF (STEP_NO > 0)' above is a means of avoiding sending any output for the initial solution
calculations.

(ii) Explicitly using the simulation number as a switch

For example, using the same model setup but optimizing different parts of it, e.g. different elements. Simulation 1 does the calculation for Ca and simulation 2 does it for Mg.

USER_PUNCH

10 IF (SIM_NO = 1) THEN PUNCH log(TOT("Ca")), "Ca", SIM_NO ELSE PUNCH log(TOT("Mg")), "Mg", SIM_NO

The simulations may contain keyword blocks such as REACTION or KINETICS which themselves
generate multiple observations.

(iii) Giving a range of simulations to use in the fit data file

If each data point requires a different model (= set of simulations) to calculate the dependent
variable, then these can be specified in a separate column in the fit data file.

observed  weight  sim
# Ca
-5.00E+000  1  1-2
-5.10E+000  1  3-4
-5.00E+000  1  5-8
...

# SO4 - blank line above breaks line in plot
-5.20E+000  1  13-14
-5.10E+000  1  25-26
-4.98E+000  1  16-17

The blockRangeColumn keyword is here defined as sim and the sim column defines the sim-
ulations to use for each observation. These do not have to be in order or to contain the same
number of simulations. They are to some extent independent but will inherit Phreeqc data
structures from simulation to simulation in the normal way. Each of these mini-blocks of sim-
ulations can have its own pre-loop and main loop simulations as defined by a separate column
specified with the mainLoopColumn keyword.

12.4.6 Data file

The data file consists of the observations, one line per observation. It needs to be sorted in x-
column in order to make the plot that is automatically produced sensible (the fitted values are
plotted as a continuous line).

The data separator needs to be specified to match that found in the data file (see ‘Organise
your data’).

Each line in the data file contains: values for each of the independent variables (if present) and
the single value of the dependent variable (if present). The first line contains a list of headers,
one per column variable. The header names are automatically converted to tags so must
adhere to the tag naming convention, i.e. they must not contain operators (+-*^) and will be
case sensitive. If the dependent variable is not present, then only simulated values can be cal-
culated.

Anything following a number sign (i.e. pound sign or hash symbol, #) is treated as a comment
and is ignored. One or more blank lines indicates a break in the data set. The break is pre-
served in the ‘out’ file and produces a break in line plots.

The data file can also contain columns containing alphanumeric data (descriptive columns).
These also produce tags which can be used for substituting character strings in the input files.
The type of column (numeric or character) is determined by analysing the first valid row of
data. If necessary rearrange the row order to ensure that the correct column type is indicated in
row 1 of the data.

If the descriptive text contains spaces, embed in quotes. Only the first 20 characters are trans-
ferred, 18 if the text contains embedded spaces and quotes need to be used.

If the blockRangeColumn has been set to a positive integer or valid column name, then this
column (counting from the left) is assumed to contain the Phreeqc simulation number (or
range of simulations) to be used for calculating the dependent variable for this line of data.
The default value of blockRangeColumn is 0 (undefined) in which case each observation is
assumed to use all simulations. Similarly the mainLoopColumn if defined contains the posi-
tion of the beginning of the main loop simulations within the block of simulations that is used
for an observation. If mainLoopColumn is not defined, mainLoop defaults to the mainLoop
setting in the input files. Not defining a blockRangeColumn and a mainLoopColumn may
work but may repeat unnecessary calculations.

For example, say the fitting requires three simulations, the first two simulations to define static
data such as the database and the third to calculate and output the dependent variable for each
observation. This third simulation also contains the tags, <PT> and <CaT>, which define the
values of the two independent variables.

Then the data file for the first two observations will look something like this:

<table>
<thead>
<tr>
<th>...</th>
<th>PT</th>
<th>CaT</th>
<th>sim_num</th>
<th>main_loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>6.3e-4</td>
<td>1e-2</td>
<td>1-3</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>6.6e-4</td>
<td>2e-2</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

The blockRangeColumn is set to ‘sim_num’ and the mainLoopColumn is set to ‘main_loop’.
The first observation is defined by three simulations (1-3) and the main loop starts at the third
simulation in this block, i.e. simulation 3. So simulations 1-2 must be pre-loop simulations.
The second observation is defined by just one simulation, simulation 3, and the main loop
starts at the first simulation in this block, i.e. simulation 3.
The sequence of simulations executed is therefore as follows:

(i) pre-loop 1: execute simulation 1, update tags
(ii) pre-loop 2: execute simulation 2, update tags

(iii) main loop: execute simulation 3 for observation 1, update tags
    main loop: execute simulation 3 for observation 2, update tags
    [repeat (iii) above] until
    ... main loop: execute simulation 3 for last observation, update tags

On subsequent iterations, the sequence starts at (iii) above.

If some Phreeqc code needs to be repeated just once on each iteration of the dataset, then include it in the main loop of the first observation, not as a pre-loop simulation of this observation.

The data file can also be used in simulation mode (calculationMethod = simulate) which has a similar setup to that of ‘fit’ but does not compare observed and calculated values and does no ‘fitting’. No dependent variable need be defined though it can be. A summary of the selected output is sent to the ‘out’ file. Simply changing the calculationMethod from ‘fit’ to ‘simulate’ will produce an output file with results for all the observations using the initial parameter values. This can be useful for quickly checking how reasonable the initial estimates are.

The log file contains the mean and standard deviation of the numeric data in the input data file. It also contains the sum of the relative standard deviations of each of the columns. This figure can be used as a crude ‘checksum’ to indicate whether two data files are the same or not.

The number of observations (lines of data) in the data file (n) fixes the number of lines to be read from the ‘out’ file: if onePass is TRUE, then the last n lines will be read (irrespective of the selectedOutputLines setting). If onePass is FALSE, then only the last line will be read from the ‘out’ file. These two options are demonstrated in the demo\kineticsSi\kineticsSifit.ppi and demo\kineticsSi\kineticsSifit1.ppi examples, respectively. The latter option is used for fitting kinetic models when the observations are at irregular time intervals.

12.5 THE OPTIMIZATION ROUTINES

12.5.1 Choice of fit algorithm

Providing that reasonably good initial estimates of the adjustable parameters can be given, then the ‘nlls’ algorithm is likely to be fastest of the available algorithms. This algorithm was especially developed for minimizing nonlinear functions involving sums of squares and is noted for its ability to rapidly home in on a solution when close to it. However, it suffers the disadvantage that it can be confused by local minima and so should be started from different starting positions to ensure the solution given is the global solution. Its behaviour is controlled by a rather small, but not-too-obscure, number of settings.

If the ‘nlls’ algorithm fails to converge either because of poor initial parameter estimates or because of its tendency to stray into undesirable territory (e.g. negative concentrations), then it is worth trying the ‘bobyqa’ or ‘subplex’ algorithms. These algorithms were designed for large-scale optimization problems with hundreds of adjustable parameters (‘variables’). They also have more general application than the specialised least squares optimizers such as ‘nlls’ and should be more efficient than the ‘nlls’ algorithm when the Gauss-Newton approach performs less well. This tends to be true for large residual problems with nonlinear terms in the sum of squares.

At present, ‘bobyqa’ is the only one of the algorithms that can apply constraints to the adjustable parameter values, in this case, simple box constraints.

Comparing the different algorithms on the same problem can provide an insight into the quality of the parameter estimates and the fitting algorithms. Ideally, they should all converge to the same set of parameter values. However, the comparisons are rarely straightforward as the convergence criteria differ to some extent. Nevertheless simple timing tests will quickly pro-
vide insight into major differences.

It is possible to apply several approaches to the same dataset by adding the list of routines wanted to the the `fitMethod` setting. If set to do so, each method will produce its own track file and plot.

**12.5.2 Scaling of parameters**

These optimization algorithms have to estimate successive steps in multi-dimensional space in order to reduce the value of the objective function to a minimum. This requires the calculation of distances between points. The estimation of derivatives may also involve a fixed shift in parameter values (‘nulls’). Therefore it may be necessary to scale the adjustable parameters to ensure that the magnitudes of their expected changes are all similar. Ideally the parameter values should all be close to one. Re-scaling could be done by a change in units, for example, or where appropriate, by taking logs (see `fitLogParameters`). Other than the option of taking logs, this scaling has to be done outside of PhreePlot.

**12.5.3 Constrained optimization**

The `bobyqa` `fitMethod` allows simple upper and lower bounds to be put on the parameter values using the `fitLowerParameterValues` and `fitUpperParameterValues` keywords.

Implicit constraints can also be imposed indirectly, e.g. by fitting the log of parameter value. This will constrain the parameter value to positive values.

**12.5.4 Control parameters**

Details of the algorithms and their underlying control parameters can be found in the library and online documentation of the routines (‘nulls’, ‘lm’, ‘newuoa’, ‘bobyqa’ and ‘subplx’). Most of the critical control parameters have been translated into PhreePlot settings so that a large degree of control over each algorithm’s behaviour can be achieved from within PhreePlot (Table 12.1). In particular, the meaning of the convergence criterion varies (sometime it refers to a change in the residual sum of squares, sometimes to the change in the parameter estimates).

Parameters that are to be fitted or made to be easily adjusted should be entered as parameters with names (up to 30 characters) (‘fitParameterNames’) and values (‘fitParameterValues’). These values are assumed to be either fixed values or initial estimates to be adjusted during fitting. This distinction is set by the `fitAdjustableParameters` keyword (0 = fixed; 1 = adjustable).

The log10 of the parameter value can also be easily fitted. Set the appropriate `fitLogParameters` keyword value to 1 otherwise set it to 0. This can provide a useful form of scaling to bring very large or very small numbers into the same order of magnitude as other parameters. It also is a simple way of constraining a parameter to a positive value.

It is important that each of the above parameter lists should have the same length. This can be set with the `numberOfFitParameters` keyword but can also be allowed to be automatically set from the length of any of the above lists as they are entered. If present, the `numberOfFitParameters` keyword should precede all the parameter lists in the input file.

**Finite difference step size (‘nulls’ only)**

This controls the step size of each parameter value used in the estimation of partial derivatives by finite differences numerically. The value of the step size should be large enough to achieve a significant change in the objective function while being small enough to give derivatives with sufficient accuracy. The correctness of the choice can best be seen from the first few iterations where each of the adjustable parameters is adjusted one by one by the specified step size. This should produce a significant change in the objective function for at least some of the parameters (preferably all). If it does not, increase the step size by an order of magnitude. Default 1e-6. Where approximate (numerical) methods are being used to generate the dependent variable
values, as in Phreeqc, there will inevitably be some noise in the generated values and so a larger value may be appropriate to ensure sufficiently accurate derivatives.

**Convergence criterion**

The meaning of this varies with the optimization method. Read the original documentation for guidance. In all cases, this parameter determines when to stop the iterations and accept (or not) a fit. A small value will tend to increase the accuracy of the fit, albeit with more iterations, but there will be a limit when the other numerical procedures (both in the calculation of partial derivatives and in Phreeqc) will limit the accuracy that can be obtained. The default value is 1e-6.

<table>
<thead>
<tr>
<th>PhreePlot keyword and action</th>
<th>'nlls'</th>
<th>'newuoa'</th>
<th>'bobyqa'</th>
</tr>
</thead>
<tbody>
<tr>
<td>fitFiniteDiffStepSize</td>
<td>DSTEP=fitFiniteDiffStepSize</td>
<td>Small values will mean less chance of straying too far on the first step and better estimates of derivatives but if too small there may not be a significant change in WRSS given the background noise. Choose judiciously.</td>
<td>not used</td>
</tr>
<tr>
<td>fitConvergenceCriterion</td>
<td>ACC=fitConvergenceCriterion^2</td>
<td>Converges when the predicted value of WRSS is &lt; ACC above the true minimum or when there is little predicted change in parameters. WRSS is a sum that depends on the number of data points.</td>
<td>RHOEND=fitConvergenceCriterion Final radius of 'trust region' determines the final accuracy in the parameter estimates. Also RHOEND&lt;=RHOBEG</td>
</tr>
<tr>
<td>fitStepSize</td>
<td>DMAX=fitStepSize</td>
<td>Maximum 'distance' of initial estimate from the solution (not scaled). Also the minimum size of the Marquadt parameter = fitConvergenceCriterion/fitStepSize.</td>
<td>RHOBEG = fitStepSize</td>
</tr>
<tr>
<td>fitMaxIterations</td>
<td>MAXFUN=fitMaxIterations</td>
<td></td>
<td>MAXFUN=fitMaxIterations</td>
</tr>
<tr>
<td>fitLowerParameterValues</td>
<td>not used</td>
<td>not used</td>
<td></td>
</tr>
<tr>
<td>fitUpperParameterValues</td>
<td>not used</td>
<td>not used</td>
<td></td>
</tr>
<tr>
<td>fitAdjustableParameters</td>
<td>Counting 1's gives the number of adjustable parameters, N.</td>
<td>As for 'nlls' but N is also used to determine the parameter, NPT, when it has not been defined explicitly, see below.</td>
<td></td>
</tr>
<tr>
<td>fitnpt</td>
<td>not used</td>
<td></td>
<td>if fitnpt is UNDEFINED then if (N&lt;6) then NPT= (N+2)<em>(N+1)/2 else NPT=2</em>N+1.</td>
</tr>
</tbody>
</table>

Table 12.1. Translation of PhreePlot settings into the control parameters for three of the optimization algorithm's
**Step size**

This usually controls the initial step size for each parameter and for ‘nlls’ is the maximum size of any step. A large value (say 100) will enable a large area to be searched but may lead to the focus wandering away from the best solution and even lead Phreeqc to fail because of unrealistic parameters.

With the **NEWUOA** and **BOBYQA** methods, the step size sets the initial size (radius) of the trust region and is an important parameter. A small value will constrain the search area to be close to the original parameter estimates which is fine provided the initial estimates were good but may lead to misleading results otherwise. A large value may lead to physically unrealistic parameter estimates during fitting and therefore to problems in Phreeqc convergence. The default value is 1.0.

**Maximum iterations**

The maximum number of function evaluations. If convergence has not been achieved by the time this limit has been reached, PhreePlot will exit the optimization gracefully and move on. If set to 1, this will force an immediate exit from the optimization routine but it will give an indication of the correctness of the initial estimates. The default value is 5000. Some of the methods make additional function evaluations in order to estimate the derivatives in the Jacobian.

**Weighting method**

`fitWeightingMethod` determines how the objective function is calculated from the residuals. The residuals are multiplied by weights, one for each observation. See the definition of the `fitWeightingMethod` for more details.

It has the following options:

- **0** absolute error: all weights = 1.
- **1** relative error: weights = abs(1/fitted value) if fitted value is not equal to 0 else it is abs(1/observed value).
- **2** from the input data file: weights in the column given by `weightColumn`.

The weights should be related to the quality of each observation in terms of the size of the observation error. Weights are normally given by the inverse of the standard deviation of each observation or something proportional to that

\[
\begin{align*}
    w_i &= 1/\sigma_i
\end{align*}
\]

where the objective function to minimize is given by

\[
\begin{align*}
    \text{minimize} \sum [w_i(f_{obs,i} - f_{calc,i})]^2
\end{align*}
\]

### 12.6 PREPARING AN INPUT FILE

#### 12.6.1 Simple example

The following is a simple example based on the `iso.ppi` file found in the `\demo\fit` directory. This example fits a Langmuir isotherm to data for Zn sorption by Hfo at constant pH (pH 5.5). The `iso.dat` data file looks like this:

<table>
<thead>
<tr>
<th>Znsorbed</th>
<th>Znconcn</th>
<th>pHobs</th>
<th>wt</th>
<th>sim#</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td>0.030</td>
<td>5.5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1.40</td>
<td>0.069</td>
<td>5.5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1.95</td>
<td>0.118</td>
<td>5.5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2.51</td>
<td>0.166</td>
<td>5.5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
where the units are mmol Zn/mol Fe for Zn\text{\textsubscript{sorbed}}, mmol Zn/L for Zn\text{\textsubscript{concn}} and pH\text{\textsubscript{obs}} is dimensionless. \textit{wt} is the relative weight assigned to each observation and \textit{sim\#} is the \texttt{Phreeqc} simulation number. Simulation 1 is used for all the calculations and so all simulation numbers have been set to 1.

The \texttt{Phreeqc} input file part is

12.6.2 Automatic updating of parameter values in an input file

If the updateFitParameters switch has been set to \texttt{TRUE} and if the fit has been successful, then the newly fitted parameter values will replace the existing ones in the input file. A backup of the original input file (***.bak.*** ) is made if a backup file of the same name does not already exist.

12.7 INTERRUPTING OR STOPPING THE FITTING

The ‘\texttt{Esc}’ key will interrupt the fitting once the calculations for the current iteration have all been finished. You are then given the option to continue or to stop. If you stop and a plot is expected, then this will be produced using the best set of fitted values at the time of interruption.

12.8 FORCING RELATIONS BETWEEN PARAMETER VALUES

It is possible to force relations between parameter values by using the \texttt{numericTags} keyword to define the desired relations and then substituting the corresponding tags in the input file in
the normal way.

For example, say there are four parameters and that these are referred to in the input file by their corresponding tag names: \(<p1>\), \(<p2>\), \(<p3>\) and \(<p4>\). If you want to force \(p4 = p1\) then reduce the number of parameters from 4 to 3 (\(p1, p2, p3\)) and add \(<p4> = <p1>\) in the numericTags block. \(<p4>\) should be left unchanged in the input file. Its value will be updated based on the numericTags expression on each iteration.

Other more complex relations may be specified in a similar way.

Note that it is important not to include the redefined parameter (\(<p4>\) above) in the set of fit parameters since the optimizer expects to have full control over all parameter values and does not expect them to be changed by an external procedure. \(<p4>\) is now a tag variable rather than a fit parameter. There is no connection between the tag variables and fit parameters during optimization.

\subsection{12.9 Multi-objective fitting}

It is possible to fit parameters to multiple different models at the same time. These models may share some common parameters, or may not. The principle is the same as before: each observation must take one line of data in the fit data file and produce one result with the input and output matching line-by-line. The data file must have a spreadsheet-like structure with each variable having a separate column. Where variables vary between the models, there will be rows containing columns with no data. It is easiest to set up such files in a spreadsheet and export as a tab delimited file. Then read the data into PhreePlot with "fn.dat \t" to indicate its tab-delimited format. Adjacent tabs will then serve as placeholders for blank columns and will not be merged.

In principle, each line of data can be derived from a different model since it is possible to specify the range or block of simulations to use for each observation in the fit data file using the blockRangeColumn. More often, groups of observations use a common set of simulations. The weights should also be specified in the weightColumn with fitWeightingMethod set to 2. Alternatively, the relative error option (fitWeightingMethod 1) may be suitable. There is a problem of mixing 'apples and oranges' here so the weighting is critical. Be wary of over-interpreting the summary statistics such as the standard errors.

The \texttt{\~demo\fit\multiobjective.ppi} file gives an example of fitting two quite distinct data sets (demonstrated individually elsewhere) at the same time and demonstrates how to set up the fit data and input files.

\subsection{12.10 Output files}

Other than the log and plot files, the useful files produced during fitting and simulations are the 'out' file and the 'pts' file. The 'trk' file will save a copy of the convergence monitoring during fitting and can be used to prepare a plot of this using the 'extraout' keyword to read in these *.trk files.

The 'out' file contains a copy of the selected output, one record per observation. During fitting, this normally contains just one block of results containing one record per observation – the results from the last iteration (which is not necessarily the best-fitting iteration). Simulations always just contain one block of results since there is no iteration. Fits can be made to store the results of all iterations by changing the rewind data separator for the 'out' file to null ("\") instead of the default '\r'. This prevents the rewind before writing results and also introduces a blank line after each block of results. This behaviour is controlled by the fifth parameter of the dataSeparators keyword.

The 'pts' file contains a comprehensive list of results for each observation including both the input data, fitted, observed and weighted residuals and the results of the selected output, all from the best-fitting iteration. This is the file that is normally used for plotting.
12.11 RESPONSE IN THE EVENT OF A FAILURE OF PHREEQC TO CONVERGE

The response if Phreeqc should fail to converge during fitting depends on the debug setting. If debug is 0 or less, then the offending point is deleted from the input and the fitting restarted without it. When this happens, a '?' is appended to the number of iterations in the pp.log file to indicate that this has happened.

If debug is greater than zero, then PhreePlot will stop if Phreeqc fails. A list of any offending points is sent to the log file. This gives the physical line numbers of the offending points as found in the fit data file (counting the header line).

12.12 PLOTTING THE RESULTS OF THE FITTING

12.12.1 Plots of observed and calculated results

The 'pts' file is the primary plot data file for fitting and simulations though other files can be added using the extradat keyword.

The 'pts' file contains data from three sources: (i) the fitting (5 columns: row number, observed, calculated, residual, weighted residual); (ii) all the variables read in from the data file; and (iii) all the columns from the selected output with values from the 'best' fit.

A plot of the observed points (as points) and the fitted values (as a continuous line) is often useful. The lines and points which are plotted is controlled by the lines and points keywords, respectively. 'Lines' plots only make sense when the points are contiguous.

The customXcolumn setting is also important as this fixes the x-variable.

The accumulated output from the selected output is stored in the 'out' file if present - this is produced for debug > 0. If the fifth data separator is "\r", the file is rewound at the beginning of each set of function evaluations so only the last set will be found on the file (not necessarily for the 'best' fit since this may have occurred in an earlier iteration). Any other character means that there is no rewind and so all selected output results will be accumulated on the file.

The 'trk' file, if requested, contains a copy of the nlls monitoring results.

The use of the labels and post/postSize keywords can be useful for making a key and for 'posting' values beside each individual point. Such posting can be useful for identifying outliers.

Additional text can be added with extraText and additional data from other files with extradat. After a successful fit, three special system tags are populated. These are <R2>, <RMSE> and <nFit> which contain the R^2, RMSE and number of data points. These tags can be used to annotate the plot using the extraText approach.

An example of a fit plot with some of the more commonly used keywords is shown in Figure 12.2. Since the 'pts' file contains both observed and fitted data, it provides a useful source of data to plot. Observations are often plotted with points and calculated values with lines.

Where the dependent variable depends on more than one independent variable, it can be difficult to choose a suitable plot. In these cases, a plot of calculated or residual values vs observed values can be useful.

12.12.2 Contour plot of the residual sum of squares

It can be useful to view how the residual sum of squares varies with changes in the value of the model parameters. A contour plot can show the variation of the residual sum of squares versus two model parameters and may help you to understand convergence problems.

For a residual sum of squares plot, the data are read in as normal for fitting and compared with the model calculated values to calculate the weighted residual sum of squares (WRSS) surface. The plot shows how the WRSS varies with the variation of two user-selected variables, most usefully two model parameters. The variation is driven by the normal x- and y-axis parameters xmin, xmax, ymin, ymax and resolution. These generate changes of the <x_axis> and
tags which should then be used in the input file to vary the model output, again most usefully by changing the fitParameterValues. The contourZvariable must be set to 'rss', 'log10(rss)', 'wrss', 'log10(wrss)' depending on whether the residuals have unit weights or not and whether (W)RSS is to be logged or not.

The number of times the input chemistry is run is \( n_{\text{res}} \times n_{\text{res}} \times \text{number of data points} \) since the rss is calculated at each of the grid points.

The demo example `\fit\contour_rss.ppi` shows how the residual sum of squares (RSS) plot for the iso.ppi isotherm fitting example varies with the two model parameters, \( \log_k \) and \( M_1 \). Here the RSS has been logged before plotting. The plot is invoked by setting calculationType...
162 PhreePlot Guide

The plot shows a deep, banana-shaped valley where the optimal combination of parameters is found. The minimum is found at $\log_k = 3.9954$ and $M_1 = 9.9011$. The long valley reflects the relatively high correlation ($r = 0.987$) between the two parameters and shows the difficulty that the fitting routine has in finding the optimal combination of parameters.

The data generated for the plot are written to the track file, `contour_rss.trk`, in the current directory in x-, y- and z-column format.

12.13 SIMULATIONS

Simulations use essentially the same setup as fitting except that there are no observations to compare with calculated values and so no fitting takes place. Typically simulations are used after fitting to plot a calculated curve based on the fitted parameters. This is done by changing the `calculationType` from 'fit' to 'simulate'. Values of the independent variables are still read in from a data file and tags assigned, exactly as for fitting.

This mechanism provides a way of running a given piece of Phreeqc code for a disparate range of samples read from a file. It is somewhat similar to the use of `SOLUTION_SPREAD` in Phreeqc but has more flexibility in the way that the data are read in.

12.14 ROOT FINDING – SOLVING EQUATIONS, ADDING CONSTRAINTS

You may want to estimate the value of one or more unknown parameters in a fully-defined chemical model. Most chemical models are too complex to calculate the unknown values explicitly and so the values have to be estimated numerically. When there are more data points than adjustable parameters, the system is over-determined and 'fit' attempts to find the optimal solution based on minimizing some kind of least squares objective function.

When the number of data points is the same as the number of unknown parameters or fewer,
there are zero degrees of freedom and the problem is then one of finding the unique set of parameter values that minimizes the objective function. This is a minimization problem with the proviso that the ‘observed’ values area all effectively zero. Some methods (subplex, bobyqa, cobyla, newuoa) work happily with no extra effort but other methods (nlins, lm) need some ‘observations’. Simply set up a file with observed values of zero. The objective function then becomes \( w_i (0 - f(x,y))^{**2} \) where \( f(x,y) \) and providing all function values are positive, the minimum in this squared surface will be at the minimum of the function itself.

It is also straightforward to contour the function in two dimensions using the ‘contour’ calculation type, see the \( \text{\texttt{\demo\contour\trigp.ppi}} \) example.

For example, assume that we want to find the volume of acid required to reach a certain pH. We set up the chemical model such that the volume of acid added is a variable and represented by some tag, say \(<\text{titre}>\). The selected output is arranged to contain the pH after adding this amount of acid. The pH is the dependent variable. In this example, there is one dependent variable and no independent variables.

Then we make a fit data file with a single data point which specifies the end point pH of interest. ‘fit’ then adjusts the single adjustable parameter, \(<\text{titre}>\), so that the match between the input and output pH is very close. The closeness is controlled by the normal convergence criteria.

The fit data file, \texttt{fittitration.dat}, simply looks like this:

\[
\text{pHwanted} \\
4.5
\]

It takes 1.761 mL of acid. In this case, there is only one Phreeqc simulation involved and the default is to use only the first simulation in fitting. If two or more simulations were involved,
then it would be necessary to add a column to the fit data file with the range of simulations to use (e.g. "1-2") and to set blockRangeColumn to point to this column.

This example can be found in demo\titration\fittitration.ppi. While this particular example could be solved pretty closely using a more direct Phreeqc approach, the principle is general and applies to more complex examples where such direct approaches are not possible.

In principle, it is possible to extend this approach to search for a solution with many unknowns. The setup has to be slightly different since the objective function must be defined in terms of a single dependent variable. Therefore this has to be defined. A convenient measure to minimize is the RMSE (root mean square error) which can be formed from the deviations of calculated values from their target values.

The following code fragment uses Na$_2$CO$_3$, NaCl and CuCl$_2$ to make 1kg of a solution with a pH of 9, a Na molality of 0.01 mol/kgw and a free Cu$^{2+}$ activity of $10^{-10}$ (pCu = 10), all in the presence of CO$_2$(g). Basic lines 10-30 calculate the three residuals and line 40 forms the objective function, the RMSE. Note that no data file is needed – it is assumed in this special case with no degrees of freedom that the ‘observed’ or target values are all zero.

You may need to apply some bounds on the solution – in the example below, only non-negative quantities are permitted – and to scale the various residuals so that they all participate in the solution. Parameter values all close to one are best. The initial values of the three parameters have each been set to one though note that the e-3 following the Cu entry effectively scales this parameter to 1000 times less than the fitted value.

Also the initial and final values of the ‘trust’ region radius, ‘rhobeg’ and ‘rhoend’, may need adjusting. ‘rhobeg’ controls the initial shift in parameter values and ‘rhoend’ controls the termination.

The above example can obviously be generalised with the full array of functions built into Phreeqc’s powerful Basic interpreter available for use. There is of course no guarantee of suc-
cess – the problem may simply not be solvable in which case the ‘nearest’ solution should be found. When successful, the final RMSE should be very small say less than 1e-8. Always check the Phreeqc output to make sure that the problem has been solved as intended.
13 The input file pre-processor

13.1 USE OF THE PRE-PROCESSOR

Although the use of tags can eliminate the need for repetitive blocks of text in an input file, this may avoid the more efficient internal looping mechanisms provided by Phreeqc and so might be undesirable especially when speed is an issue. This most obviously occurs during fitting where the onePass TRUE option is much more efficient than the onePass FALSE option.

For example, consider the test fitting example shown below. This is similar to the demo\fit-preprocessor\isopp.ppi example:

```plaintext
PRINT
  -reset false
SURFACE_MASTER_SPECIES
 Surf Surf
SURFACE_SPECIES
 Surf = Surf
  log_k 0

 Surf + Zn+2 = SurfZn+2
  log_k <log_k>                      # from fitParameterNames
SELECTED_OUTPUT
  -high_precision true
  -reset false
USER_PUNCH
  -headings sorbZn pH molZn step
 10 sorbedZn=SURF("Zn","Surf")
 20 if (step_no=0) THEN punch sorbedZn, -la("H+"), tot("Zn")*1e3, step_no
SOLUTION_SPREAD
  -units mmol/L
  -pH <pH>
include 'isopp.dat'
SURFACE
 Surf <M1>                          # from fitParameterNames
  -no_edl
  -equil 1
END
SURFACE
 Surf <M1>
  -no_edl
  -equil 2
END
SURFACE
 Surf <M1>
  -no_edl
  -equil 3
END
...
SURFACE
 Surf <M1>
  -no_edl
  -equil 10
END
```

where the ten observations are defined as a series of SOLUTIONS 1-10 using the
**SOLUTION_SPREAD** keyword data block. The surface is equilibrated in turn with each of the solutions using a repetitive block of code that looks something like:

```
SURFACE
  Surf <M1>
  -no_edl
  -equil n
END
```

where `n` is the solution number.

The pre-processor provides a very simple mechanism for generating these repetitive blocks of code using a simple form of numeric substitution. This is sufficient, for example, to generate a series of simulations varying only by their `SOLUTION` number, moles of `REACTANT`, or time interval.

In the above example, the repeating blocks would be replaced by:

```
<repeatStart1> 1 10 1                # startvalue, endvalue, increment
SURFACE
  Surf <M1>
  -no_edl
  -equil <repeatValue1>
END
<repeatEnd1>
```

The critical parts are the `<repeatStart1>` tag which defines the start of the repeat block, the `<repeatEnd1>` tag which defines its end, and `<repeatValue1>` which is a placeholder for where the generated value is to be substituted. All values must be numeric though not necessarily whole numbers.

The three parameters on the `<repeatStart1>` line define a simple looping mechanism: start value, end value, increment value. The generated values are substituted at the `<repeatValue1>` point and the whole repeat block is then added to the input with the substituted value.

All three parameters must be numeric. The sign of the increment is not important. If the second value is larger than the first, the value counts up by the given increment and if the second value is smaller than the first, the value counts down.

The ‘1’ block identifier appended to the end of `<repeatStart` in the above example can be any unique character string of any length.

More than one repeat block can be given but these blocks either must not overlap or if they do, they must be nested ‘properly’. Each named loop must have one and only one start and end tag so that each loop is unique. The blocks are expanded top down.

This expanded input is fed into the input parser in the normal way. The parser knows nothing about the pre-processing. The results of the pre-processor’s expansion are written to the log file.
14 Keywords

14.1 SUMMARY OF AVAILABLE KEYWORDS

Table 14.1 gives a summary of the available keywords. The keywords are arranged in their

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPECIATION</td>
<td>section heading</td>
</tr>
<tr>
<td>PhreePlotVersion</td>
<td>version of PhreePlot</td>
</tr>
<tr>
<td>unrecognisedKeywordIsFatal</td>
<td>determines if unrecognised text in an input file counts as a fatal error or not</td>
</tr>
<tr>
<td>checkForUpdate</td>
<td>checks PhreePlot website for a more recent version of PhreePlot</td>
</tr>
<tr>
<td>jobTitle</td>
<td>title of this job for log file</td>
</tr>
<tr>
<td>speciationProgram</td>
<td>speciation program used</td>
</tr>
<tr>
<td>speciationProgramVersion</td>
<td>version of speciation program used</td>
</tr>
<tr>
<td>database</td>
<td>thermodynamic database to use</td>
</tr>
<tr>
<td>dateDatabase</td>
<td>date or version of thermodynamic database to use - no longer used</td>
</tr>
<tr>
<td>pdfMaker</td>
<td>file path of ps to pdf conversion program</td>
</tr>
<tr>
<td>fillColorDictionary</td>
<td>file path for fill colour dictionary used in predominance diagrams</td>
</tr>
<tr>
<td>lineColorDictionary</td>
<td>file path for line colour dictionary used in custom and fit plots</td>
</tr>
<tr>
<td>blockRangeColumn</td>
<td>name of column in a data file defining the range of a block of simulations</td>
</tr>
<tr>
<td>mainLoopColumn</td>
<td>name of column in a data file defining the start of the main loop simulations</td>
</tr>
<tr>
<td>selectedOutputFile</td>
<td>logical switch which determines if the selected output file is written to disk</td>
</tr>
<tr>
<td>Phreeqc.0.out</td>
<td>explicit switch which determines if the Phreeqc.0.out file is written</td>
</tr>
<tr>
<td>all</td>
<td>logical switch which determines if the *.all file is written</td>
</tr>
<tr>
<td>log</td>
<td>logical switch for the log file</td>
</tr>
<tr>
<td>trk</td>
<td>logical switch for the track file</td>
</tr>
<tr>
<td>pts</td>
<td>logical switch for the points file</td>
</tr>
<tr>
<td>pplog</td>
<td>logical switch for the pp.log file</td>
</tr>
<tr>
<td>pol</td>
<td>logical switch for the polygon file</td>
</tr>
<tr>
<td>labelFile</td>
<td>logical switch for the labels file</td>
</tr>
<tr>
<td>vec</td>
<td>logical switch for the vectors file</td>
</tr>
<tr>
<td>nudge</td>
<td>logical switch for the nudge file</td>
</tr>
<tr>
<td>out</td>
<td>logical switch for the output (or out) file</td>
</tr>
<tr>
<td>writeAllInputFiles</td>
<td>logical switch controlling the number of input files written to the log file</td>
</tr>
<tr>
<td>dataSeparators</td>
<td>controls the separator(s) used for data input files and the format of output files</td>
</tr>
<tr>
<td>calculationType</td>
<td>type of calculations and plotting to do</td>
</tr>
<tr>
<td>calculationMethod</td>
<td>whether to calculate and plot or just replot</td>
</tr>
<tr>
<td>mainspecies</td>
<td>main species in a predominance or mineral stability plot</td>
</tr>
<tr>
<td>xmin</td>
<td>minimum x-value for calculations</td>
</tr>
<tr>
<td>xmax</td>
<td>maximum x-value for calculations</td>
</tr>
<tr>
<td>ymin</td>
<td>minimum y-value for calculations</td>
</tr>
<tr>
<td>ymax</td>
<td>maximum y-value for calculations</td>
</tr>
<tr>
<td>loopFile</td>
<td>file path for file containing values for the z-loop variable</td>
</tr>
<tr>
<td>loopMin</td>
<td>minimum value for the z-loop variable</td>
</tr>
<tr>
<td>loopMax</td>
<td>maximum value for the z-loop variable</td>
</tr>
<tr>
<td>loopInt</td>
<td>interval or increment for the z-loop variable</td>
</tr>
<tr>
<td>Keyword</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>loopLogVar</td>
<td>determines whether the z-loop variable value is to be anti-logged (10^z)</td>
</tr>
<tr>
<td>loopIndexStartNumber</td>
<td>initial number of loop index used e.g. for naming files</td>
</tr>
<tr>
<td>resolution</td>
<td>number of intervals on which x- and y-axis interval is divided</td>
</tr>
<tr>
<td>debug</td>
<td>controls response to errors and extent of reporting made to log file</td>
</tr>
<tr>
<td>omitAccumulate</td>
<td>filter out all lines containing the given string(s) from being sent to Phreeqc</td>
</tr>
<tr>
<td>printScreenFrequency</td>
<td>frequency with which progress in h expansion calculations is sent to screen</td>
</tr>
<tr>
<td>plotFrequency</td>
<td>frequency with which plot.ps file showing plotting progress is written</td>
</tr>
<tr>
<td>selectedOutputLines</td>
<td>controls how many selected output lines are sent to the ‘out’ file</td>
</tr>
<tr>
<td>mainLoop</td>
<td>controls the division between pre-loop and main loop Phreeqc simulations</td>
</tr>
<tr>
<td>dominant</td>
<td>dominant or subdominant predominance diagram</td>
</tr>
<tr>
<td>numericTags</td>
<td>number of numeric tag definitions to follow</td>
</tr>
<tr>
<td>characterTags</td>
<td>number of character tag definitions to follow</td>
</tr>
<tr>
<td>initialValue</td>
<td>sets the value of all undefined numeric tags</td>
</tr>
<tr>
<td>unrecognisedKeywordIsFatal</td>
<td>determines whether an unrecognised keyword produces a fatal error or not</td>
</tr>
<tr>
<td>stopOnFail</td>
<td>determines whether calculations stop after Phreeqc has failed or not</td>
</tr>
<tr>
<td>writePlaceholder</td>
<td>add a placeholder to the ‘out’ and ‘trk’ files if Phreeqc fails</td>
</tr>
<tr>
<td>FIT</td>
<td>section heading</td>
</tr>
<tr>
<td>dataFile</td>
<td>file path for the data file containing dependent and independent variables</td>
</tr>
<tr>
<td>onePass</td>
<td>determines if all dependent variable values are calculated in one pass or not</td>
</tr>
<tr>
<td>logDepVariable</td>
<td>indicates whether dependent variable is entered on a linear or log scale</td>
</tr>
<tr>
<td>logVariableIn</td>
<td>indicates whether independent variables are entered on a linear or log scale</td>
</tr>
<tr>
<td>dependentVariableColumnObs</td>
<td>column from which to read the dependent variable from the fit data file</td>
</tr>
<tr>
<td>dependentVariableColumnCalc</td>
<td>column from which to read the dependent variable from the selected output</td>
</tr>
<tr>
<td>skip</td>
<td>controls the number of records read in from the fit data file</td>
</tr>
<tr>
<td>fitMethod</td>
<td>choose optimization/plotting procedure(s)</td>
</tr>
<tr>
<td>fitFiniteDiffStepSize</td>
<td>finite difference step size for estimating first derivatives in fitting routine</td>
</tr>
<tr>
<td>fitConvergenceCriterion</td>
<td>convergence criterion in fitting routine</td>
</tr>
<tr>
<td>fitStepSize</td>
<td>starting step size and/or maximum step size in fitting routine</td>
</tr>
<tr>
<td>fitMaxIterations</td>
<td>maximum iterations in fitting routine</td>
</tr>
<tr>
<td>fitWeightingMethod</td>
<td>weighting method to use in fitting routine</td>
</tr>
<tr>
<td>weightColumn</td>
<td>column giving the weights in the fit data file</td>
</tr>
<tr>
<td>numberOfFitParameters</td>
<td>number of parameters that are defined and tagged for fitting</td>
</tr>
<tr>
<td>fitParameterNames</td>
<td>names of fit parameters</td>
</tr>
<tr>
<td>fitLogParameters</td>
<td>determines whether to log transform fit parameters or not</td>
</tr>
<tr>
<td>fitAdjustableParameters</td>
<td>determines whether fit parameters are fixed or adjustable</td>
</tr>
<tr>
<td>fitParameterValues</td>
<td>initial values of fit parameters</td>
</tr>
<tr>
<td>fitnpt</td>
<td>number of interpolation points used by the NEWUOA and BOBYQA optimizers</td>
</tr>
<tr>
<td>fitLowerParameterValues</td>
<td>lower constraint on fit parameters (not currently used)</td>
</tr>
<tr>
<td>fitUpperParameterValues</td>
<td>upper constraint on fit parameters (not currently used)</td>
</tr>
<tr>
<td>updateFitParameters</td>
<td>determines if fitted parameter values are written to the input file or not</td>
</tr>
<tr>
<td>PLOT</td>
<td>section heading</td>
</tr>
<tr>
<td>units</td>
<td>units to use for all dimensions</td>
</tr>
<tr>
<td>paperSize</td>
<td>paper size to write to Postscript file</td>
</tr>
<tr>
<td>backgroundColor</td>
<td>colour of background within the plot boundaries and for the rest of the page</td>
</tr>
<tr>
<td>colorModel</td>
<td>colour model to use for all colours</td>
</tr>
<tr>
<td>ps</td>
<td>logical switch for the ps format plot file</td>
</tr>
<tr>
<td>pdf</td>
<td>logical switch for the pdf format plot file</td>
</tr>
<tr>
<td>png</td>
<td>logical switch for the png format plot file</td>
</tr>
<tr>
<td>screen</td>
<td>logical switch for screen output and setting for close down time on failure</td>
</tr>
<tr>
<td>epsi</td>
<td>logical switch for the epsi format plot file</td>
</tr>
<tr>
<td>eps</td>
<td>logical switch for the eps format plot file</td>
</tr>
</tbody>
</table>
Table 14.1. Available keywords (contd)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>jpg</code></td>
<td>logical switch for the jpg format plot file</td>
</tr>
<tr>
<td><code>ppa</code></td>
<td>obsolescent</td>
</tr>
<tr>
<td><code>overlay</code></td>
<td>adds one or more graphic images (ps files) on top of the one being generated</td>
</tr>
<tr>
<td><code>plotTitle</code></td>
<td>title to be placed at top of plot</td>
</tr>
<tr>
<td><code>plotTitleColor</code></td>
<td>color of plot title</td>
</tr>
<tr>
<td><code>plotTitleSize</code></td>
<td>height of plot title</td>
</tr>
<tr>
<td><code>xtitle</code></td>
<td>x-axis title</td>
</tr>
<tr>
<td><code>ytitle, 2ytitle</code></td>
<td>y(2y)-axis title</td>
</tr>
<tr>
<td><code>xoffset</code></td>
<td>distance from left hand edge of page to left-hand y axis</td>
</tr>
<tr>
<td><code>yoffset</code></td>
<td>distance from bottom of page to lower x axis</td>
</tr>
<tr>
<td><code>pageOrientation</code></td>
<td>portrait or landscape</td>
</tr>
<tr>
<td><code>multipageFile</code></td>
<td>for multiplot runs, determines whether the plots are all in one file or not</td>
</tr>
<tr>
<td><code>customLoopManyPlots</code></td>
<td>make many separate plots (if T) or just one (if F) when multiple z-loops specified</td>
</tr>
<tr>
<td><code>xaxisLength</code></td>
<td>length of x axis</td>
</tr>
<tr>
<td><code>yaxisLength</code></td>
<td>length of y axis</td>
</tr>
<tr>
<td><code>pxmin</code></td>
<td>minimum value of x axis on plot and start of major ticks</td>
</tr>
<tr>
<td><code>pxmax</code></td>
<td>maximum value of x axis on plot</td>
</tr>
<tr>
<td><code>pxmajor</code></td>
<td>interval between major tick marks on x axis</td>
</tr>
<tr>
<td><code>pxdec</code></td>
<td>controls number of figures after decimal point on x-axis labelling</td>
</tr>
<tr>
<td><code>pxminor</code></td>
<td>x-axis interval between minor (unlabelled) tick marks</td>
</tr>
<tr>
<td><code>yscale</code></td>
<td>determines the yscale of predominance plots: native, pe or Eh (V or mV)</td>
</tr>
<tr>
<td><code>pymin, p2ymin</code></td>
<td>maximum value of y(2y)-axis on plot</td>
</tr>
<tr>
<td><code>pymax, p2ymax</code></td>
<td>minimum value of y(2y)-axis on plot and start of major ticks</td>
</tr>
<tr>
<td><code>pymajor, p2ymajor</code></td>
<td>interval between major tick marks on y(2y)-axis</td>
</tr>
<tr>
<td><code>pydec, p2ydec</code></td>
<td>controls number of figures after decimal point on y(2y)-axis labelling</td>
</tr>
<tr>
<td><code>pyminor, p2yminor</code></td>
<td>y(2y)-axis interval between minor (unlabelled) tick marks</td>
</tr>
<tr>
<td><code>gridLines</code></td>
<td>controls whether to plot x- and y-axis grid lines</td>
</tr>
<tr>
<td><code>gridColor</code></td>
<td>colour of x- and y-axis grid lines</td>
</tr>
<tr>
<td><code>gridDashesPerInch</code></td>
<td>number of dashes per inch for x- and y-axis grid lines</td>
</tr>
<tr>
<td><code>gridLineType</code></td>
<td>line styles for major and minor x- and y-axis grid lines</td>
</tr>
<tr>
<td><code>customXcolumn</code></td>
<td>column name or number of x-axis variable for plotting</td>
</tr>
<tr>
<td><code>lines</code></td>
<td>list of column names or column numbers to plot as lines</td>
</tr>
<tr>
<td><code>lineWidth</code></td>
<td>line width</td>
</tr>
<tr>
<td><code>dashesPerInch</code></td>
<td>number of dashes per inch for dashed lines (separate version for 2y axis)</td>
</tr>
<tr>
<td><code>lineType</code></td>
<td>line style for lines in custom plots (separate version for 2y axis)</td>
</tr>
<tr>
<td><code>lineColor</code></td>
<td>set initial colours in the line colour sequence</td>
</tr>
<tr>
<td><code>changeColor</code></td>
<td>determines if the colour changes automatically for subsets of data in custom plots</td>
</tr>
<tr>
<td><code>useLabelsFile</code></td>
<td>determines if an existing labels file is used or is regenerated in predominance plots</td>
</tr>
<tr>
<td><code>useLineColorDictionary</code></td>
<td>is the line colour dictionary used for colours and label positions</td>
</tr>
<tr>
<td><code>restartColorSequence</code></td>
<td>controls color sequence between plots &amp; within subsets of the same data column</td>
</tr>
<tr>
<td><code>plotOrder</code></td>
<td>controls the order of plotting of lines and points</td>
</tr>
<tr>
<td><code>points</code></td>
<td>list of column names or column numbers to plot as points</td>
</tr>
<tr>
<td><code>pointType, pointType2y</code></td>
<td>list of number or names of symbols used in custom plots</td>
</tr>
<tr>
<td><code>pointSize, pointSize2y</code></td>
<td>size of symbols used in custom and fit plots</td>
</tr>
<tr>
<td><code>pointColor</code></td>
<td>starting colour of symbols used in custom and fit plots</td>
</tr>
<tr>
<td><code>rimFactor</code></td>
<td>widths of the rims of symbols (filled circles) as a fraction of symbol sizes</td>
</tr>
<tr>
<td><code>rimColor</code></td>
<td>colours of the rims of filled circle symbols</td>
</tr>
<tr>
<td><code>pointsSameColor</code></td>
<td>controls whether all symbols have the same colour</td>
</tr>
<tr>
<td><code>tickSize</code></td>
<td>length of the tick marks and controls plotting of the grid lines</td>
</tr>
<tr>
<td><code>tickColor</code></td>
<td>colour of the tick marks</td>
</tr>
<tr>
<td><code>axisNumberSize</code></td>
<td>height of the axis numbers</td>
</tr>
<tr>
<td><code>axisNumberColor</code></td>
<td>colour of the axis numbers</td>
</tr>
<tr>
<td><code>axisTitleSize</code></td>
<td>height of the axis title</td>
</tr>
</tbody>
</table>
three major sections: SPECIATION, FIT and PLOT based on their function.

14.2 CONVENTIONS

The PhreePlot keywords are listed below in alphabetic order. Keywords are not case sensitive. Each keyword has one or more attributes associated with it. The type of these attributes is fixed. Each one belongs to one of the following types:

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>logical</td>
<td>T (RUE) or F (ALSE)</td>
</tr>
<tr>
<td>integer</td>
<td>a whole number (no decimal point)</td>
</tr>
<tr>
<td>number</td>
<td>an integer or floating point number</td>
</tr>
</tbody>
</table>

Table 14.1. Available keywords (contd)

axisTitleColor  colour of the axis title
axisLineWidth   width of the axis lines
axisLineColor   colour of the axis lines
labels          list of names to be used for the lines labels in custom plots
labelSize       height of the labels used for labelling lines
labelColor      colour of the labels used for labelling lines
nudgeFile       file to 'nudge' labels to a new position
info            colour for the text of the 'info' data accompanying each plot
legendBox       inserts a box around a legend on a custom or contour plot
legendTitle     text for legend (key) title in custom plots
legendTextColor colour of the legend text
legendTextSize  height of text in the legend for custom, fit and grid plots
labelEffort     controls the effort (and time) taken to improve automatic label placement
trackSymbolSize size of symbol used for a tracking plot and for labelling anchor positions
domain          determines if the domain boundaries are plotted in a ht1 predominance plot
customXcolumn   number or column name pointing to the x-column in a custom or fit plot
font             base font and character encoding to be used for all text
plotFactor      scaling factor to be used for all plot elements
missingValue    dummy value to signal a missing value
minimumAreaForLabeling minimum size of field (as a %age of total area) to plot a label in a ht plot
minimumYValueForPlotting minimum maximum y-value for which to plot a curve in a custom plot
beep            turn sound on or off
simplify        controls the degree of polyline simplification in ht plots
convertLabels   interpret label names as Phreeqc formulae or not
extraSymbolsLines path name for file containing additional symbols and line data to add to plot
extraText        path name for file containing additional text to add to plot
extradat         list of path names for files to add to the search path for variables used in plotting
post             list of names to be used for the posted text or a data file column tag name
postSize         size of the posted text
contourZvariable name of the variable/column in the outfile that contains the z-data for contouring
contours         list of values at which to draw the contour lines
contourFillColor list of colours to fill the contour levels
contourLineColor list of the widths of the contour lines
contourLineColor list of the colours of the contour lines
contourShiftLabel list of contour labels to move and the distance to move them
contourLabelSize  list of the size (height) of the contour labels
contourLabelFigs list of numbers specifying the number of digits to use in the contour labels
contourLabelFont list of fonts used for printing the contour labels
contourLabelColor list of colours used for the text of the contour labels
string a character string (with or without enclosing quote marks)
filename a valid filename which may include the path (system dependent)
filepath a valid filepath without a filename (system dependent)
color a valid colour

14.3 KEYWORD DESCRIPTION

The function and use of each keyword is given below. Keywords have been ordered alphabetically. Aliases are alternative names for the keywords. The default is the value set internally by PhreePlot and read from the pp.set file. These default values can be overridden from the input files (*.ppi or override.set) or during an interrupt ('Esc') while running. Values given in square brackets are optional.

---

**all**

Value 'auto' or a logical (TRUE or FALSE) [filename]
Description Switch to determine if the standard *.all file is written
Aliases System default 'auto'
Use Explicitly sets the switch that determines if the *.all file is definitely written (TRUE) or not (FALSE). This file accumulates all of the printed output from Phreeqc and will be written to on every Phreeqc iteration. This file can get very large and can slow down execution times. FALSE will cause the file to be deleted on termination if present.

The 'auto' value sets the all switch depending on the debug level, FALSE if ABS(debug) < 2 else TRUE. When TRUE, the *.all file will always be created.

'auto' is the default setting.

The second, optional, parameter is the filename given to this file. This could be Phreeqcall.out which was the default name given to this file before December 2015.

---

**axisLineColor**

Value Cohort colour
Description Determines the colour of the axes.
Aliases
System default auto
Use Enables the line colour of the axes to be changed. Colours should be chosen from the colour palette.

'auto' reverts to 'black'.
Example 38
**axisLineWidth**

Value: non-negative number  
Description: Determines the width of the axes.  
Aliases: axislw  
System default: 0.3  
Use: Enables the line width of the axis lines to be changed. This setting also controls the line width of the axis tick marks and grid lines (which are special long ticks). This is the same as the width of the axis line for the major ticks, and half the width for the minor ticks.

Example: 38

**axisNumberColor**

Value: Cohort colour  
Description: Determines the colour of the numbers on the axis scales.  
Aliases: numberColor  
System default: auto  
Use: Enables the line colour of the axis numbering to be changed. Colours should be chosen from the colour palette. ‘auto’ reverts to ‘black’.

Example: 38

**axisNumberSize**

Value: non-negative number  
Description: Determines the size of the axis numbers.  
Aliases: numberSize  
System default: 3  
Use: Enables the size of the axes to be changed.

Example: 38

**axisTitleColor**

Value: Cohort colour  
Description: Determines the colour of the titles of the axis scales.  
Aliases
System default: auto

Use: Enables the colour of the axis titles to be changed. Colours should be chosen from the colour palette.

'auto' reverts to 'black'.

Example: 38

---

**axisTitleSize**

Value: non-negative number

Description: Determines the size of the axis titles.

Aliases: axisTitleHt

System default: 3

Use: Enables the size of the axis titles to be changed.

Example: 38

---

**backgroundColor**

Value: Cohort colour [cohort colour]

Description: Determines the background colour of the plot and optionally of the page.

Aliases: background

System default: nd nd

Use: Enables the background colours to be changed. This plot background is the area bounded by the x and y axes. The plot background is overwritten by any text, lines or fills produced by the plot.

The page background is the whole page. The page background colour, if drawn, will be overplotted by the plot background colour, if drawn.

nd, or equivalently "", will suppress the drawing of the colour.

Example: 70

---

**beep**

Value: logical

Description: Determines whether the sound is on or off.

Aliases: τ

System default: τ

Use: Switches the sound on (τ) or off (τ). A high-pitched beep is produced on successful completion of a plot. A low-pitched beep is produced when Phreeqc fails to converge or when the plot fails to complete. This option can be useful to signal progress when multiple plots are being produced in the background or to highlight when Phreeqc fails. It can also be irritat-
ing. Placing \texttt{beep FALSE} in the override file will ensure that no sound is heard from any run no matter what the setting in the \texttt{pp.set} and input files!

Because of the duration of the beep, repeated low frequency beeps can significantly slow down execution. The beeping can be turned off during execution using the \texttt{'Esc i beep F'} sequence. Alternatively, setting \texttt{beep F} in the \texttt{override.set} file will ensure that the sound is off in all subsequent runs.

---

### blockRangeColumn

<table>
<thead>
<tr>
<th>Value</th>
<th>zero or a positive integer or a column name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specifies the column number (counted from the left) or column name in the fit data file in which the range of \texttt{Phreeqc} simulations number to use to calculate each observation will be found.</td>
</tr>
<tr>
<td>Aliases</td>
<td>\texttt{fitSimulationColumn}, \texttt{fitSimulationNumberPosition}, \texttt{fitSimulationPosition}</td>
</tr>
<tr>
<td>System default</td>
<td>0</td>
</tr>
<tr>
<td>Use</td>
<td>Only used in \texttt{simulate} and \texttt{fit} calculations. When \texttt{onePass} is \texttt{FALSE}, the dependent variable for each line of data is calculated from its own \texttt{Phreeqc} simulation, or range of simulations. This variable specifies the column name in the fit data file which contains the \texttt{Phreeqc} simulation number(s) to use for each observation. A range is entered in the form \texttt{<code>m-n</code>} or \texttt{<code>m_n</code>} without any spaces. Since a range such as \texttt{1-2} is non-numeric, the column must be read as a character string rather than as a number. Force this by adding \texttt{$$} to the end of the column header, e.g. \texttt{sim$} rather than \texttt{sim}, or by ensuring that the first line of data in a fit data file is character by enclosing the value or range in quotes, e.g. \texttt{&quot;2&quot;} or \texttt{&quot;1-2&quot;}.</td>
</tr>
</tbody>
</table>

In principle, every line of the data file could be specified to use a different block or set of \texttt{Phreeqc} simulations. This would make a multi-objective function fit.

When \texttt{onePass} is \texttt{TRUE}, \texttt{blockRangeColumn} is not checked or used since the selected output from all simulations is automatically used for each observation.

When this setting is set to \texttt{0} (the default) or a negative integer, it is assumed that the block range used for all observations is from \texttt{1-n}, where \( n = \) number of simulations in the input file, i.e. the entire block of simulations is computed in a single \texttt{Phreeqc} run for each observation.

Use \texttt{mainLoopColumn} to specify a column in the data file in which the division between the pre-loop calculations and main loop calculations within each block can be set for each observation.

**Examples**  
80, 83
**calculationMethod**

- **Value**: 1, 2 or 3 and their negatives
- **Description**: Determines whether to undertake the calculations and plot or just replot existing results
- **Aliases**: method, plotMethod
- **System default**: 1
- **Use**:
  - 1 = calculate and plot
  - 2 = replot only (necessary results files must be present). For ht1 plots, the existing polygon file is used. Do not re-optimize label positions in custom plots or recalculate contours in a contour plot.
  - 3 = do not re-speciate but reprocess the output data and replot. With pre-dominance plots, this generates new polygon and label files from the points file ('ht1') or the track file ('grid' and 'grids'). With contour plots, this re-reads the raw data from the out file and recalculates the contours.

Typically the plot files are generated first time through with a setting of 1 then set to 2 (or 3) during fine-tuning of the appearance of plots. This saves speciation time but not labelling time.

Negative values of `calculationMethod` will do the same as their positive counterparts except that no plot will be produced.

Use of `calculationMethod` 2 or 3 (replot) will not generate ‘run time’ values for those tags assigned values at run time, namely the user punch tags and the tags automatically created during a fit run. In these cases, user punch tags that are not defined values will not be recognised as valid tags in subsequent simulations and so will either plot as their literal text string, i.e. no substitution will be undertaken, or will produce an error message where a numeric value is required. In these cases, recalculating with `calculationMethod` 1 is the only option.

For custom plots, `calculationMethod` 2 does not reoptimize label positions; `calculationMethod` 3 does.

For grid plots, `calculationMethod` 3 can be used to resume calculations when there has been a crash or an interrupt and stop. The track file is read in and calculations resumed where they left off.

Dummy user punch tags will be defined from the second (first non-header) line of the appropriate ‘out’ file, if present, and fit tags generated from the second line of the data file, if present. This is necessary since all tags are updated just before plotting and some of these tags may be involved in tag definitions. Clearly the values will have little significance in a replot.

If the tags cannot be defined, an error will result.

In order to get dynamic tags – i.e. those generated during a run and whose effects are not stored in the various data files used for plotting – properly substituted, it is necessary to do the calculations from scratch (`calculationMethod` 1) each time a plot is wanted.
calculationType

Value: one of ‘grid’, ‘ht1’, ‘custom’, ‘species’, ‘simulate’ or ‘fit’

Description: Determines the calculations carried out and the type of plot drawn

Aliases: plotType, type, calculation

System default: “custom”

Use: Specifies one of the six calculation types available: ‘grid’, ‘ht1’, ‘custom’, ‘species’, ‘simulate’ or ‘fit’. This keyword should be specified in each input file.

Examples: 1, 3, 55, 73, 80

changeColor

Value: logical

Description: Determines the extent to which auto-generated colours vary for multiple sets of data especially for subsets of the same variable (column)

Aliases: chgCol

System default: FALSE

Use: Colours for custom plots are picked from a 15-long list of colours. Colours specified by one of the colour settings, e.g. lineColor, are promoted to the top of the line colour list.

If changeColor is set to FALSE and if the line colour dictionary is not being used, all line and point colours will be those specified by the lineColor(m) and pointColor(n) settings, where m, n are the positions of the variable in the corresponding lines and points sequences, respectively, i.e. lineColor(1) applies to lines(1), lineColor(2) applies to line(2), etc. If the lineColor() list is shorter than the lines() list, then the autocolor sequence is used. The lineColor list is recycled if necessary. All subsets of data for a given variable will have the same base colour although the colour density may vary.

If changeColor is set to TRUE and if the line colour dictionary is not being used, the colours used for successive subsets of data and for different variables will be automatically chosen from the appropriate points or lines colour lists. In general, changeColor TRUE aims to ensure that a different colour is used for each dataset plotted, however generated.

With species plots where the species are auto-generated and there is no explicit lines variable, the colours specified by the lineColor keyword are used, and recycled if necessary.

The colour ‘auto’ (the default) will use auto-generated colours for each distinct dataset using the colour sequence for the data type specified (lines, points, lines2y, points2y) – there is no true colour ‘auto’. This is equivalent to setting changeColor to TRUE but only applies to that specific
data type.

If `restartColorSequence` is TRUE, the colour sequence is reset for each new plot or each new data type (lines, points etc). This can be used to ensure that the same colour is used for the same variable within and between plots. If `restartColorSequence` is FALSE and the color sequence contains generic colours, i.e. colours specified without a colour density e.g. `red` rather than `red4`, then the density of the chosen colours will be cycled through 4, 6, 8, 2, 4 ... for the different subsets, e.g. `red4`, `red6`, `red8`, `red2`, ...

These lists start with the colours specified with the `lineColor` and `pointColor` lists and then continue with the unused colours in their normal order. The list is recycled as necessary changing the colour density on each cycle if generic colours (e.g. `red`) have been specified.

Fine tuning of colours is done by editing the line colour dictionary.

**Example**

```
62
```

---

**characterTags**

<table>
<thead>
<tr>
<th>Value</th>
<th>A list of character tag definitions, all on one logical line.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Character tags can be used to substitute strings.</td>
</tr>
<tr>
<td>Aliases</td>
<td><code>characterTag</code>, <code>numberOfCharacterTags</code></td>
</tr>
<tr>
<td>System default</td>
<td><code>&quot;&quot;</code></td>
</tr>
<tr>
<td>Use</td>
<td>Used to enter user-defined character tags with a similar syntax to <code>numericTags</code>, as list of triplets:</td>
</tr>
<tr>
<td></td>
<td><code>tagname = expression</code>, e.g. <code>characterTags</code> &lt;myTextColor&gt; = &quot;blue4&quot; \</td>
</tr>
<tr>
<td></td>
<td>&lt;mySymbolColor&gt; = &quot;red4&quot;</td>
</tr>
<tr>
<td></td>
<td>If none is to be defined, use a blank string: <code>characterTags &quot;&quot;</code></td>
</tr>
</tbody>
</table>

See **Section 5.3.2** for a definition of valid tag names. Spaces around the `=` sign are optional. The tag expression should be a simple character string, embedded in quotes if necessary. It can also be another character tag. The initial integer, if present, and the definitions should all be on a single logical line, hence the use of \ above. A \ continuation character enables each tag definition to be placed on a separate physical line. This can aid legibility.

The tag expression must be a single 'word'. Enclose in quotes if it contains one or more spaces. A null string can be entered as `""` or `"`. Character tag names are case sensitive.

When a character tag is substituted into an input file, no enclosing quotes are included. Therefore in situations where enclosing quotes are needed to force the string to be interpreted as a character string, enclose the tag expression itself in quotes, e.g. if `<data>="Clear Lake"`

```
USER_PUNCH
- headings pH dataset
10 PUNCH -log10("H+"), "<data>
```
Character tags can also be used to replace lists of character or numeric items in an input file, e.g.

```plaintext
characterTags <col> = "green6 purple" \ 
<contours> = "0 2 4 6 8 10"
```

This keyword can be repeated and each instance will be appended to the last rather than replacing it.

---

**checkForUpdate**

**Value**
logical [non-negative number]

**Description**
Determines whether PhreePlot checks the PhreePlot website for a more recent version of the program.

**Aliases**
checkForUpdates

**System default**
FALSE 1

**Use**
If TRUE, the PhreePlot website is contacted and the date of the latest version obtained. This is compared with the current version and if a more recent version is available, the option is given for opening the website for download. No download is actually made – this is left for the user to do.

The optional second parameter determines the minimum time gap between successive checks (in days) when checking is set on. Set to 0 to check whenever PhreePlot is run or a large number to check infrequently. The default of 1 means check once every day.

The checking is largely silent unless `debug > 0` in which case a message is given when a check is being made.

The checking requires an internet connection.

---

**colorModel**

**Value**
rgb, b&w or gray

**Description**
Sets the colour model used

**Aliases**
color

**System default**
rgb

**Use**
Three colour models are available:

- `rgb` => the red-green-blue colour model
- `b&w` => black and white only
- `gray` => a grayscale

rgb colour names are based on the Cohort colour scale. b&w converts all colours to black. gray attempts to convert colours to a grayscale based on
their hue. The Postscript file produced reflects the colour model used, i.e. a black-and-white file can never be made to produce a colour plot. On the other hand, a colour plot file can often be made to produce a grayscale print on a black and white-only printer.

The 'b&w' and 'gray' colour models translate any 'coloured' colours to their black and white or gray equivalents at plot time. The original colours are written to the appropriate colour dictionary so that the plot can be replotted in full colour if desired.

If full control over the gray colours used is wanted, these should be entered explicitly in the colour dictionary.

An alternative way of getting a grayscale image is to prepare the image file as a rgb coloured image in the normal way and then to use the options available in many printer drivers to export to a file using only grayscale or black and white colours. You don't have to actually have the printer attached, just have the printer driver installed, e.g. the Adobe pdf printer driver.

---

**contourDashesPerInch**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of non-negative numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the dash density for contour lines</td>
</tr>
<tr>
<td>Aliases</td>
<td>System default: 10</td>
</tr>
<tr>
<td>Use</td>
<td>The list of numbers should be the same length as the number of contours. If shorter, the list is recycled. If longer, it is truncated.</td>
</tr>
</tbody>
</table>

**contourFillColor**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of colours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the colours used for filling in between the contour lines</td>
</tr>
<tr>
<td>Aliases</td>
<td>auto</td>
</tr>
<tr>
<td>System default</td>
<td>The list of colours should be the same length as the number of contours plus one. If shorter, the list is recycled. If longer, it is truncated.</td>
</tr>
<tr>
<td>Use</td>
<td>The colours are paired off in sequence, one per contour class. Use 'nd' for 'not drawn' or no colour.</td>
</tr>
</tbody>
</table>

'auto' generates a list of blue-red colours centered on sky1-red1 with the lowest class being the darkest blue and the highest class being the darkest red. The blue list is expanded sky1, sky2, ..., sky9 and the red list is expanded red1, red2, ..., red9. As the number of colours required is increased, additional colours are added to the low and high end: spring, magenta, cyan, purple, green, orange, blue and maroon. There are therefore a maximum of 90 distinct colors. This corresponds with 89 specified contour levels. The last two colors are recycled if there are more than 90
classes.

Colours should be chosen from the *colour palette*.

---

**contourLabelColor**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of colours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the colours used for the labels to the contour lines</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td><em>auto</em></td>
</tr>
<tr>
<td>Use</td>
<td>The list of colours should be the same length as the number of contours. If shorter, the list is recycled. If longer, it is truncated. The colors are paired off in sequence, one per contour value. Use ‘na’ for ‘not drawn’ or no colour. This will suppress the drawing of the label(s) for this contour level. ‘<em>auto</em>’ will copy the colour specified by <em>labelColor</em>. Colours should be chosen from the <em>colour palette</em>.</td>
</tr>
</tbody>
</table>

---

**contourLabelFigs**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of non-negative integers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines the number of significant figures used in the numeric contour labels</td>
</tr>
<tr>
<td>Aliases</td>
<td><em>auto</em></td>
</tr>
<tr>
<td>System default</td>
<td><em>auto</em></td>
</tr>
<tr>
<td>Use</td>
<td>The list of integers should be the same length as the number of contours. If shorter, the list is recycled. If longer, it is truncated. The integers are paired off in sequence, one per contour value. The format used depends on the size of the value of the label. Exponential format is automatically used for very large or small numbers. Precede the integer with “_” to force the number to be written in floating point format and with “$” for exponential format, e.g. $2$. Trailing zeros will be removed. Use 0 to force the value to be printed to the nearest integer. ‘<em>auto</em>’ will use 3 or less figures depending on the value and is the default.</td>
</tr>
</tbody>
</table>

---

**contourLabelFont**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of fonts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines the font used by the contour labels</td>
</tr>
</tbody>
</table>
The list of fonts should be the same length as the number of contours. If shorter, the list is recycled. If longer, it is truncated.

The fonts are paired off, one per contour value.

The fonts are specified by font names or font family names (see `font`).

**contourLabelSize**

- **Value**: list of numbers
- **Description**: Determines the size (height) of the contour labels
- **Aliases**: System default `auto`
- **Use**: The list of label sizes should be the same length as the number of contours. If shorter, the list is recycled. If longer, it is truncated.

  The sizes are paired off, one per contour value.

  The units of `contourLabelSize` depend on the units in effect when `contourLabelSize` is set.

  'auto' copies the size given by `labelSize`.

**contourLineColor**

- **Value**: list of colours
- **Description**: Sets the colours used for the contour lines
- **Aliases**: System default `auto`
- **Use**: The list of colours should be the same length as the number of contours. If shorter, the list is recycled. If longer, it is truncated.

  The colors are paired off, one per contour value.

  Use ‘nd’ for ‘not drawn’ or no colour. This will suppress the drawing of the contour(s) for this contour level.

  ‘auto’ will match the colour specified by `lineColor`.

  Colours should be chosen from the [colour palette](#).

**contourLineType**

- **Value**: list of numbers from 0 to 20
- **Description**: Sets the line type used for contour lines
The list of numbers should be the same length as the number of contours. If shorter, the list is recycled. If longer, it is truncated. The 20 line styles are shown here.

Other settings such as line colour, line width and dash density will affect the appearance of the lines.

**contourLineWidth**

- **Value**: list of numbers
- **Description**: Sets the width of the contour lines
- **Aliases**: contourOption
- **System default**: auto
- **Use**

Use 'na' for 'not drawn' or no colour. This will suppress the drawing of the contour(s) for this contour level.

'auto' will copy the width specified by lineWidth.

**contourOptions**

- **Value**: string of options
- **Description**: Sets various options for the contour plot
- **Aliases**: contourOption
- **System default**: smooth=0 fill=TRUE joinSegments=TRUE labelPosition=straightest
- **Use**

This currently sets up to four options affecting the generation and display of a contour plot. Include only those keyword-value pairs that need changing. Separate keyword-value by either an equals sign or one or more spaces, tabs or commas. Number and order of pairs given is optional.

The options are:

(i) smooth 0 | 1 | 2

After generating the z-data, or reading it from an outfile, it is possible to smooth it using a low pass filter. This is either a five point (smooth = 1) or nine point (smooth = 2) moving average based on equal weighting of the central cell and either the four nearest neighbours (i.e. the cells N-S-E-W of the central cell) or the eight nearest neighbours (i.e. the cells N-S-E-W-NW-NE-SE-SW of the central cell). A value of 0 means no smoothing is carried out. It is not necessary to regenerate the z-data to test these various options. Once the data have been generated and plotted, use the calcula-
tionMethod 3 option to smooth again and replot.

(ii) fill TRUE | FALSE

If TRUE, the colour fills are plotted with the simplified lines taken from the vector file and the coordinates for the polygon fills taken from the polygon file. If FALSE, this produces a lines only plot with no colour fill between contour levels. It comes in two flavours depending on the third option:

(iii) joinSegments TRUE | FALSE

If this is set true, then the lines are joined up and then plotted. If it is set false, then the short line segments making up a contour are plotted separately and in the order produced by the scanning algorithm used to determine the contours.

This difference can be subtle and is best seen by looking closely at the line joins which are smoother with the TRUE option. Dashed lines are often much better looking with the TRUE option. The advantage of the FALSE option is that its output is derived directly from the contouring routine and does not require post-assembly into a continuous line, which can be difficult. Therefore if the joining of line segments is a problem, try the option which misses this step.

If the joinSegments FALSE option is used to generate the original plot, then the TRUE option cannot be used for a simple replot (calculationMethod 2) since no polygon file will have been produced. Recalculate (calculationMethod 1) or relabel and replot with calculationMethod 3 to produce the polygon file.

(iv) labelPosition longest | straightest | centre | center

This controls the position of the labelling for all contour lines, either close to the longest straight section (default) or near the centre of the line. Individual labels can be shifted with contourShiftLabels.

If a plot with inline labels and no fill colour is wanted, choose the fill=TRUE and contourFillColor nd.

---

**contours**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Aliases</th>
<th>System default</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>list of numbers in strictly ascending order or auto</td>
<td>Sets the values of the contours to plot</td>
<td></td>
<td>auto 17 e</td>
<td>This keyword defines the contour levels used. The main task is to get a set of contours that displays what you want without getting confused by numerical noise, especially around phase boundaries. There are several ways of defining the number and value of the contour levels. The first way is simply a list of user-defined contour values in a strictly ascending order (cannot be equal). The list can be of any length. The second way, auto [n [p</td>
</tr>
</tbody>
</table>
value of the contours. \( n \) contours are chosen (default \( n = 17 \)).

These values are selected based on the \( z \)-data being contoured. This is done in one of three ways: (i) by percentiles (‘\( p \)’) in which the various contour classes are approximately equally occupied, (ii) by simplified percentiles (‘\( s \)’) which is the same as the ‘\( p \)’ option except that any pair of adjacent contour values that are ‘rather close’ to each other (within less than 1e-2 of the \( z \)-data range) are replaced by a single average value, or (iii) by dividing the \( z \)-range (\( z_{\text{max}} - z_{\text{min}} \)) into \( n \) equally-spaced contours (‘\( e \)’).

The \( z \)-data statistics used in calculating contour levels using the \texttt{auto} option are based on all the data read from the data file not a selection based on any redefinition of the plotting domain by changing \texttt{xmin} etc.

Aside from the simplified percentiles approach which explicitly ‘simplifies’ or reduces the number of contours, successive pairs of contour values must always differ from each other by at least 1e-8 of their average value. If specified or generated contour values are closer than this, the pair of values will be replaced by their average value.

The actual number of contours plotted may therefore be fewer than the number specified. This occurs when the contours are deemed to be ‘too close’ to each other as described above.

The empirical (‘\( e \)’) approach is the default.

The contour values actually used are always reported in the log file.

Example 84

---

### contourShiftLabel

<table>
<thead>
<tr>
<th>Value</th>
<th>‘c’ or ‘n’ then zero or more triplets of integers which specify the plot number, contour number and shift amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Display label position and specify which labels to move and by how much for contour plots with inline labels (filled colour plots)</td>
</tr>
<tr>
<td>Aliases</td>
<td>System default ‘c’ Use The first character determines what the label is:</td>
</tr>
<tr>
<td></td>
<td>‘c’ ‘c’ for contour. The contours are labelled with their values (default). The number of digits is controlled by \texttt{contourLabelFigs}.</td>
</tr>
<tr>
<td></td>
<td>‘n’ ‘n’ for number. The labels show the contour line number rather than the contour value. The vertices of the contours (after line simplification) are shown with the track symbol (filled circle) of colour \texttt{trackSymbolColor} and size \texttt{trackSymbolSize}(1). The sequence number of each vertex is written above the vertex using \texttt{trackSymbolColor} and half \texttt{labelSize} and with the line number appended in parentheses.</td>
</tr>
</tbody>
</table>

If triplets of integers follow, these define a contour label and how much to move it from its default position. The default position is either centered within the longest straight section of each contour, or in the middle. This is controlled by one of the \texttt{contourOptions}. |
The triplets are specified as follows:

**first integer** (non-negative): the absolute value defines the plot number of interest. Each plot is numbered sequentially as plotted. The plot number is given in the log file and is printed at the top left of the info block. The sequence number increments on cycling through the main species (outer loop) and then the loop variables (inner loop).

**second integer** (non-negative): gives the contour number. The contours are plotted from the ‘vec’ file with the list of vertices of each contour being separated by a blank line. The contours are numbered sequentially as read from this file. The contour number refers to this sequential position. Each contour can in principle be represented by more than one vector when it intersects a domain boundary, hence the reason to specify by this number rather than simply the contour class.

**third integer** (non-negative): gives the ‘distance’ to shift or move the label from its default position in terms of vertices. The default position places the label at the centre of the longest straight segment in the simplified vector. A shift of +1 moves the label to between the next pair of points, +2 to two points forward etc. -1 moves the position backwards one point etc. ‘Forward’ is always ‘moving forward with the high side on the right’.

In order to estimate the shift settings, it is simplest to first plot with the ‘n’ option. This will mark all the points on the contours with coloured circles. The number of points to shift by can then be easily determined, shifting forward from the default position using positive integers and backwards using negative integers. Once the shift has been determined, the shift type can be reversed from ‘n’ to ‘c’ to plot the contour values.

Simplified straight line segments will have few vertices. The number can be increased by reducing simplify. The printing of all labels for a given contour level can be suppressed by setting the contourLabelSize to 0. or the contourLabelColor to ‘nd’.

The figure below shows the contours labelled with their consecutive line numbers (1-14) using the ‘n’ option. The colour and size of the circle symbols have been specified with the trackSymbolColor and trackSymbolSize(1) settings.

Moving the ‘9’ and ‘10’ labels forward one point is achieved with:

\[
\text{contourShiftLabel} \ n \ 1 \ 9 \ 1 \ 1 \ 10 \ 1
\]

as seen in the figure below:

This moves the labels to the centre of the next interval. If the shift places the label outside the list of points, the label is not drawn. This is how the plotting of individual overlapping labels can be entirely suppressed.
contourZvariable

Value
name or number of an outfile column (case sensitive)

Description
Specifies the variable (column) in the outfile to be used as a source of z-data for contouring

Aliases
System default
"

Use
The contourZvariable must be present in the outfile. It can be specified by name or number (the position in the outfile). Normally the name is set
by the -headings setting in a USER_PUNCH data block but it may be defined dynamically in which case it may be necessary to specify the column by number or by using the default heading such as “no_heading_1”.

For a residuals sum of squares contour plot, contourZvariable must be either ‘rss’, ‘log10(rss)’, ‘wrss’ or ‘log10(wrss)’ depending on whether the residuals have non-unit weights and whether the (W)RSS is to be logged before writing to file and plotting.

There must be nres x nres rows of z-data in the outfile where nres is the resolution. The speciation must not fail anywhere. If it does, the calculations stop immediately and no plot is produced.

Example

---

**convertLabels**

<table>
<thead>
<tr>
<th>Value</th>
<th>logical [logical]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines whether an attempt is made to interpret plot labels as Phreeqc chemical formulae with subscripts and superscripts etc</td>
</tr>
<tr>
<td>Aliases</td>
<td>convertLabelNames</td>
</tr>
<tr>
<td>System default</td>
<td>T T</td>
</tr>
</tbody>
</table>

Use

If the first setting is set to T(RUE), label names are checked to see if they are plausible Phreeqc chemical formulae and is so, they are converted. This does not check the thermodynamic database for formulae names but does check for basic things such as the first character must be a (, [or upper case letter, and it must not contain any of the characters: ¬¬¬#@?"£$%^&'.

If a label does not look like a formula or if convertLabels is FALSE, then no attempt to interpret it as formula. For examples of the conversion see Section 6.4.2.

If some labels are to be interpreted as formulae and some not, then convertLabels should be set to TRUE and the individual label names should be written in such a way as to prevent them being interpreted as a formula, e.g. begin names with a lower case character or include one of the ‘illegal’ characters.

The ¬ character can be read but will not be printed with the ASCII encoding and so if added to the end of a name will prevent its conversion without being seen in the final plot.

This setting applies to all label names including loop names.

The second optional setting controls the interpretation of colons in formulae. If this is set to TRUE, then colons (:) are replaced with a period (.). This is to deal with formulae such as ZnCO3:H2O and UF4:2.5H2O. Again, it deals with all translations. If a block on a single label translation is wanted, edit the appropriate line in the labels file so that the a non-printing character such as ¬ (ASCII encoding) precedes the colon.

Example

84
customLoopManyPlots

Value logical switch
Description Determines if each new value of the z-loop variable produces a new plot or not.

Aliases System default FALSE
Use Only used for custom plots. The default (FALSE) means that each value of the loop variable will normally produce a separate curve on a single plot. This can be messy and so when this option is set to TRUE, a new plot is produced for each value of the loop variable. These can be combined into a single file using the multipageFile option.

This does not apply to species plots where each loop value, if used, will always produce a separate plot.

customXcolumn

Value positive integer or column name (case sensitive)
Description Determines which column of data in the outfile is used for the x axis during custom plotting.

Aliases xColumn, customXPosition
System default 1
Use Ensures that the correct column is used as the x-axis variable for plotting. The order of output is determined by the order of user_punch statements and the choice of other selected_output parameters in the Phreeqc code. The columns are counted from the left. Used for custom and species plots. In species plots, customXcolumn should point to the column of either the species name or the numeric value of the pair wanted.

If data from two or more data files are to be used in the plot, make sure that they have the same name for the x column. If this is not possible, or if two or more separate plots are wanted, plot in separate runs, possibly creating a batch file or script to execute the two runs consecutively.

Examples 55, 73, 80

dashesPerInch, dashesPerInch2y

Value list of non-negative numbers
Description Determines the number of dashes (and dots) per inch for dashed lines.

Aliases dashes
System default 10
Use

This gives the number of dashes (and dots) per inch for dashed lines in custom plots. Dashed lines are indicated by the respective lineType style.

In custom plots, the dashesPerInch for a line is picked from the lines list in order (and recycled where necessary), i.e. the first line takes the first dashesPerInch, the second line takes the second dashesPerInch etc.

dashesPerInch2y is similar to dashesPerInch but applies to the 2y axis.

Dashed lines of varying density can also be used for the grid lines of a plot (see gridDashesPerInch).

Examples

---

**database**

<table>
<thead>
<tr>
<th>Value</th>
<th>filename</th>
</tr>
</thead>
</table>
| Description | Controls which thermodynamic database file is used by Phreeqc.
| Aliases   | wateq4f.dat |
| System default | wateq4f.dat |
| Use      | Set the filename of the database that is to be used by Phreeqc. Note that the database should be set with this keyword rather than using the Phreeqc DATABASE keyword. |
| Example  | 40 |

---

**databaseVersion**

<table>
<thead>
<tr>
<th>Value</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Gives information about the version of the database selected.</td>
</tr>
<tr>
<td>Aliases</td>
<td>none</td>
</tr>
<tr>
<td>System default</td>
<td>none</td>
</tr>
<tr>
<td>Use</td>
<td>If the log file is activated, this string is printed to it. It is also printed in the info block of a plot if that is selected to be printed. These provide a record of which database was used in the calculations used to make the plot. The string can be a date but does not have to be. This setting does not affect the computations.</td>
</tr>
</tbody>
</table>

---

**dataFile**

<table>
<thead>
<tr>
<th>Value</th>
<th>filename [data separator]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specifies the file containing data for the simulate and fit calculations and optionally the data separator(s) used to parse it.</td>
</tr>
<tr>
<td>Aliases</td>
<td>fitDataFile</td>
</tr>
<tr>
<td>System default</td>
<td>fitDataFile</td>
</tr>
</tbody>
</table>
Use

This text file contains the observations (dependent variable), if present, and the independent variables if present. The first line should consist of the column headers. These are converted to tags and so should conform to the tag naming conventions (e.g. arithmetic operators like + or - and other illegal characters will be automatically replaced by an underscore).

The total number of columns to be read is determined from the number of entries in the column header. This should also match the number of entries specified by the logVariableIn setting.

Contiguous blanks are treated as a single separator but multiple other separators (e.g. , ,) will introduce null fields. Comments can be included using a # as usual. Blank lines are ignored.

Where only some non-essential data are absent, some form of missing data identifier can be used to identify such cells, essentially acting as placeholders. For example, a blank field can be entered as a null string, "", or by a missing value. A null string will be set to zero if it represents a numeric field.

The header line and the remaining part of the file will be parsed, by default, according to the separator(s) specified by the first entry in the dataSeparators keyword. If the optional data separator string is specified with the data file name, then this is used. The options for this separator are given in Section 5.2.7.

The location of the column containing the dependent variable if present is specified by the dependentVariableColumnObs keyword.

The other columns that have been read in are given tags defined by their column headers with numeric or character type depending on the type of the corresponding entry in the first valid data line. These tags can then be used in the Phreeqc input file. Each iteration of Phreeqc will read in one line of the data file, generate a full set of tags and their corresponding tag values, make any tag substitutions in the Phreeqc input and run. A maximum of 20 characters is output by Phreeqc - longer strings will be truncated.

If calculationType = 'simulate', then the data file can be used to supply a list of tag values to the input file without undertaking any fitting. There need not even be a dependent variable.

If the filename is blank and the calculation type is 'fit', then it assumed that there are no degrees of freedom and the problem is a 'root finding' one. In this case, the 'target' value is always set to 0.0, i.e. \( f(x) = 0 \). The \( f(x) \) is set in a USER_PUNCH block and the initial estimate(s) are adjusted to achieve the target value.

Examples

80, Using the 'simulate' calculationMethod

---

dataSeparators

Value
six separate 1- or 2-character strings

Description
Defines the separators used in data input and formatted output files.

Aliases
sep

System default
input="\" output="\t" xyblank="\p" loop="\p" rewind="\r"
Use

The standard input files use a space, tab or comma for separators but this can be too flexible for formatted data files. Also it is useful to be able to specify whether blank lines (indicating a break in a plotted curve) are written to the ‘out’ file after a change in a loop variable. This keyword contains options that can be used to control the interpretation of various data separators.

With whitespace separators (\b, \w and \), contiguous separators are treated as a single separator but in the case of tab (\t)- or comma (\,)-delimited input they are treated as single separators and so can be used to read in files with blank or missing fields, eg. “,”.

If the keyword=value format shown above is not used then all six separators must be included on a single line in the order given. If keywords are used, then one or more options can be specified and in any order.

Quoted fields are treated as a single field even if they contain spaces.

The six possible entries are:

1. input: default separator(s) for reading data input files, e.g. by ‘fit’ and ‘speciate’ as well as loop and extradat files (the separator can also be specified separately for each file, see Section 5.2.7);

2. output: separator to be used in formatted output files notably the ‘out’, ‘trk’ and custom ‘pts’ files. The same separator will also be used later for reading these files;

3. xyblank: controls whether a blank line is inserted into the ‘out’ file for each new value of <x_axis> or <y_axis>;

4. loop: controls whether a blank line is inserted into the ‘out’ file for each new value of the loop variable (z-value). "\p" will insert a blank line, "" will not;

5. rewind: controls whether the ‘out’ file should be rewound at the beginning of each iteration or not.

6. break: a ‘break variable’ used to introduce one or more breaks in a column of data based on the change in slope of this variable.

The input separator defines the separator for reading ‘user-prepared’ data files (as opposed to input files with keywords, extra text files or dictionary files). The three data files that use this setting at present are the fit data file used during fitting and simulations, the loop data file and any extradat data files.

Use any single character or "\t" for a tab, "\b" for a blank space or "\w" for both (whitespace). A null string, entered as a pair of quotes with nothing in between (""), indicates that either a tab, space(s) or comma is interpreted as a valid separator. This is equivalent to "\". Consecutive blanks are treated as a single separator but multiple other single character separators are not, i.e. they will be interpreted as missing data or blanks depending on their type. This allows blank fields to be read. Tabs alone are useful when an input file has been pasted into a text file from a spreadsheet and when spaces and commas are present in text strings and the strings are not embedded in quotes. Although quoted strings will preserve whatever is inside them, the quotes can get lost on pasting to and from a spreadsheet. "", " should be used for the first setting in comma-separated (csv) files.
The output separator specifies the separator to use for formatted output files, notably the 'out', 'trk' and 'pts' files. Tabs ("\t"), commas (",") or spaces ("\b") are the most commonly used separators. Enclose the specified separator in quotes if necessary. Tabs are useful when the file is to be pasted into a spreadsheet. Spaces and commas are preserved in strings without the need to quote them.

When a space ("\b") is used for the output separator, then additional spaces are inserted to justify the columns. This format is better for reading and printing. However, character strings are truncated to 18 characters to allow the strings to be quoted. Use tabs if this is a problem.

Note that "" is not the same as " ". The former is the null string; the latter is a space.

The xyblank and loop separators refer to the between-the-line separator used in the 'out' file. If these two strings are "\p", then a blank line is inserted into the out file after (i) a change in the x- and y-axis loop variable (third separator); or (ii) a change in the z-loop (or <loop>) variable (fourth separator). Any other characters, including the null string, "", means that no blank line is inserted at these points. These settings can be used to control the breaks made when plotting curves. If a loop file is used, it is also possible to insert a blank line into the 'out' file by inserting a blank line into the corresponding position in the loop file.

The rewind separator determines if the 'out' file should be rewound at the beginning of a fit iteration or not. A rewind is indicated by the "\r" combination otherwise there is no rewind. The net result is that if the file is rewound at the beginning of each fit iteration, the 'out' file will only be left with the results from the last set of calculated values returned from Phreeqc. Otherwise the results from each iteration, i.e. all Phreeqc calculations, will accumulate in this file.

The break separator is a break variable that can be useful for separating the plotting of a column of data in the out file into two or more separate curves (as can be done with a blank line) when there is no blank line in the out file. Sometimes it is not possible to introduce a blank line in the out file where it is needed. For example, if more than one Phreeqc simulation is executed during a single PhreePlot iteration (using the main-Loop) the selected output from the two simulations is run together without a break. However, it is usually possible to arrange for some variable to indicate when a break is needed. This could be one of the loop variables or the simulation step number.

This break parameter is the name of the numeric column in the outfile that is used to identify a break. This only applies to the file containing the break variable, and only one such break variable can be specified. A break is signalled when the direction of the 'slope' in this break variable changes. The reference direction is determined from the first two rows of data and is positive, negative or zero (signalled by an absolute difference of less than 1E-8). A break is made at every change of direction in slope with each block of data being considered independently. Care should be taken to avoid choosing a variable in which the difference between adjacent data values is subject to significant numerical errors.

The data separator used for reading input data for custom plots is always the separator set by the output separator since the file used for plotting is generated with this format. Care has to be taken to not edit the plot data
files in such a way as to bring this relationship out of synchronization. If in doubt, regenerate the files, calculationMethod 1.

Examples 82, Using the 'simulate' calculationMethod, \demo\sis.ppi

dateDatabase

<table>
<thead>
<tr>
<th>Value</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Gives the date of the specified database</td>
</tr>
<tr>
<td>Aliases</td>
<td>databaseVersion</td>
</tr>
<tr>
<td>System default</td>
<td>''</td>
</tr>
<tr>
<td>Use</td>
<td>No longer used. The date is now obtained from the file itself and is the date that the file was last modified. It is only used for printing to the log file and the info data block.</td>
</tr>
</tbody>
</table>

debug

<table>
<thead>
<tr>
<th>Value</th>
<th>0, 1, 2, 3, 99</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Controls the amount of information sent to the log file and the action taken when an error is encountered.</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td>'0'</td>
</tr>
<tr>
<td>Use</td>
<td>The higher the value, the greater is the amount of information that is sent to the log file.</td>
</tr>
</tbody>
</table>

There are many switches controlled by debug but the main actions are:

0 = provides the least logging and therefore gives the fastest execution times. In this case, all Phreeqc input/output is via memory rather than via disk files. This debug setting is the normal value for production runs. With predominance plot calculations, the species returned in the case of errors in speciation are all given the label 'NA'.

1 = as above and also writes the values of all the tags substituted to the log file and saves a copy of the speciation output from the last simulation in the file Phreeqc.0.out providing this has not been set to false explicitly. With predominance plot calculations, any error in speciation triggers an immediate halt to the calculations. Use this setting to check that the correct lines of Phreeqc code have been executed each time and that the tag values are as expected.

2 = as above and also accumulates the Phreeqc.0.out data for each simulation in the *.all file providing the all keyword has not been set to false explicitly. Use this setting to see the detailed output from Phreeqc. This file can get very large.

3 = as above but also echoes the Phreeqc input to the screen just before it is executed. The Phreeqc output is also inserted into the log file. This can produce a very large log file with slow execution times.
Debug also controls the output when the default file switches for Phreeqc.0.out and all are set to 'auto'. These are detailed in Table 14.2.

Table 14.2. The effect that the debug setting has on the generation of Phreeqc output files.

<table>
<thead>
<tr>
<th>Phreeqc output file</th>
<th>debug keyword setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phreeqc.0.out *</td>
<td>&lt;=0 1 2 3</td>
</tr>
<tr>
<td>Error Phreeqc.0.err</td>
<td>FALSE TRUE TRUE TRUE</td>
</tr>
<tr>
<td>Log Phreeqc.0.log</td>
<td>FALSE TRUE TRUE TRUE</td>
</tr>
<tr>
<td>Selected output (e.g. selected_1.0.out)</td>
<td>FALSE ** TRUE ** TRUE **</td>
</tr>
</tbody>
</table>

* with predominance plots, there is a special case when resolution = 1 in that this forces Phreeqc.0.out to be written whatever. This file will contain a list of all the possible mineral phases ready for pasting into the input file.

** can be overridden by the selectedOutputFile keyword.

The special value of '99' does not run the input files but rather checks the files for the use of any keyword aliases. If found these are listed along with the respective main keywords. This function operates from the point of the definition of debug onwards. Therefore to check all the files move debug to the top of the pp.set file and give it a value of 99.

**dependentVariableColumnObs**

- **Value**: zero or a positive integer or column name (case sensitive)
- **Description**: Used in fitting to specify which column in the data file holds the dependent variable.
- **Aliases**: dependentVariablePositionIn, dependentVariableColumnIn
- **System default**: **
- **Use**: When fitting data to a Phreeqc model, there is always one dependent variable containing the observations and this identifies where in the data file it will be found. The columns are counted from left to right.

A value of zero should be used when a simulation is being carried out and there is no dependent variable present. If a simulation is being carried out and the data file contains a column with dependent variable data then dependentVariableColumnObs should point to this column so that it can be skipped when reading the file. This makes it possible to switch between fit and simulate without changing the data file.

If dependent variable is to be read in, then dependentVariableColumnObs should be given the value zero.

**Examples**: 80. Using the 'simulate' calculationMethod
**dependentVariableColumnCalc**

**Value**
zero or a positive integer or column name (case sensitive)

**Description**
Used in fitting to specify which column in the selected output file holds the dependent variable.

**Aliases**
dependentVariablePositionOut, dependentVariableColumnOut

**System default**
"

**Use**
When fitting data to a **Phreeqc** model or when calculating the results of a simulation, there is usually one dependent variable that must be calculated and sent to the selected output file. **dependentVariableColumnCalc** identifies where in the selected output file this will be found. The columns are counted from the left.

The column specified depends on the sequence of the **PUNCH** statements in the **USER_PUNCH** keyword block and whether other selected output parameters have been selected. If in doubt, set debug 2 or 3, run and interrupt after a few iterations. Then look at the selected output file to determine its position or use the column name set in the **USER_PUNCH** block. This will also be printed in the log file.

If no selected output is wanted, then **dependentVariableColumnCalc** should be given the value zero.

**Example**
80, Using the 'simulate' **calculationMethod**

---

**domain**

**Value**
logical [colour [number [ number [number]]]]

**Description**
Determines if and how domain boundaries are plotted in a predominance diagram.

**Aliases**
plotDomain

**System default**
F auto UNDEFINED UNDEFINED

**Use**
The domain boundaries are the vectors going from **xmin** to **xmax**, **ymin** to **ymax** etc delineating the calculation domain. This keyword is useful for deciding whether the boundaries should be shown in **ht1** plots (this is where one of the two species codes in the vectors file is 99). With the native y-scale and default plot scales, the boundaries will normally be overdrawn by the axis lines and so this setting has little visible effect. However, this keyword setting is useful for suppressing the drawing of these boundaries in pe-pH plots calculated with a non-native pe axis (e.g. using **O2**(g)). If you want to see the boundary vectors, set **domain** to **TRUE**.

Alternatively it is possible to simply comment out unwanted field boundaries in the vector file with a leading # and replotting with **calculationMethod** 2.

The second and subsequent parameters are for the colour of the line, the line type, dashes per inch and line width. If absent, "auto" or undefined,
the defaults are taken from the current line parameters of the plot (lineColor, lineType, dashesPerInch, lineWidth).

Example 46

### dominant

<table>
<thead>
<tr>
<th>Value</th>
<th>logical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines if the dominant or sub-dominant domain boundaries are plotted in a predominance plot</td>
</tr>
<tr>
<td>Aliases</td>
<td>τ</td>
</tr>
<tr>
<td>System default</td>
<td>dominant set to TRUE plots the normal predominance or stability diagram with the fields showing the dominant species.</td>
</tr>
<tr>
<td>Use</td>
<td>dominant set to FALSE plots fields for the second most abundant (sub-dominant) species instead.</td>
</tr>
</tbody>
</table>

Example 2

### eps

<table>
<thead>
<tr>
<th>Value</th>
<th>logical [epstype]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines whether the plot output (if any) is converted to a file in the Encapsulated PostScript (eps) format.</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td>F</td>
</tr>
<tr>
<td>Use</td>
<td>Encapsulated Postscript (eps) files are Postscript files that are one page long and have a bounding box included. They will only be produced for the first page in multipage files (multipageFile).</td>
</tr>
<tr>
<td></td>
<td>eps files can be useful for embedding in other documents as they are always closely cropped but getting reliable eps files has become more difficult in recent years and it may be worthwhile resorting to high resolution png files instead.</td>
</tr>
<tr>
<td></td>
<td>An eps file can only be produced if Ghostscript/GSview is installed. If the optional epstype parameter is set to ‘gs’ (or ‘GS’) then the Ghostscript version of the eps file is produced using the eps2write device. Otherwise PhreePlot makes use of the Ghostscript bbox device to generate the required values of the bounding box and then adds these to the native ps file. This version is normally smaller in file size and better in quality.</td>
</tr>
<tr>
<td></td>
<td>Sometimes the bounding box is incorrectly calculated. This can be manually fixed by editing the bounding box at the top of the eps file. Alternatively, other versions of Ghostscript can be tried.</td>
</tr>
</tbody>
</table>
**eps**

**Value** logical

**Description** Determines whether the plot output (if any) is converted to a file in the Adobe Encapsulated PostScript Interchange (epsi) format.

**Aliases**

**System default** F

**Use** An epsi file can only be produced if Ghostscript/GSview is installed. PhreePlot makes use of the Ghostscript ps2eps1 utility to produce this file.

The Adobe epsi format is a type of eps file which includes a coloured bitmap preview image as well as the bounding box. It is therefore clipped at the boundaries of the plot which makes it convenient for inserting into documents. The plot is visible when inserted into software such as Microsoft Word. The preview is stored in a platform-independent format.

Unlike ps files, eps and epsi files can only be one page long. As a result, multipage ps files produced by PhreePlot cannot be converted to epsi files. If eps and multipageFile are set to TRUE and more than one page is produced, no epsi file is produced.

The file size tends to be quite large but the quality tends to be better than the eps format files produced by PhreePlot.

The file created is given the extension epsi.

An example of output in epsi format is given below.
**extradat**

**Value**  
filename [data separator]

**Description**  
Name of a data file which is added to the search path when looking for variables used in a custom plot, or for tag definitions. Unlike for other keywords, there can be multiple occurrences of the `extradat` keyword, each adding a new file to the search path. An optional data separator can be specified which will be used to parse the file.

**Aliases**  
extraOut

**System default**  
```
```

**Use**  
By default, the data to be plotted are sought by checking the column names in the header of the `out` file. `extradat` allows the search path to be extended to other files. The first occurrence of the variable in the search path is used.

Columns can still be referred to by numbers in these extra files. The numbering continues consecutively based on the order of the files in the `extradat` list.

These additional files should have the same format as an `out` file, namely a matrix layout with row 1 as column headers. These define the variable names. It is preferable to make these variable names unique, i.e. different from the variable names in any of the other `out` files. If not, the first found is chosen.

The header line and the remaining part of the file will be parsed, by default, according to the separator(s) specified by the first entry in the `dataSeparators` keyword. If the optional data separator string is specified with the file name, then this is used. The options for this separator are given in Section 5.2.7.

Each file used must contain a column of data with the same specified `customxColumn` heading. Each y-column uses the x-column from the same
file so that the lengths of x and y arrays will always be the same.

The extradat files can also be used as a source of the post and loop columns.

They can also be used to define tag expressions. The first header line is used to generate the tag name and the second line defines the corresponding tag values, either numeric or character. The extradat file can be generated from an earlier simulation.

Example

### extradat

<table>
<thead>
<tr>
<th><strong>Value</strong></th>
<th>filename [data separator]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>An optional file containing details of additional symbols and lines to be added to the plot.</td>
</tr>
<tr>
<td><strong>Aliases</strong></td>
<td>symbols, extraSymbols</td>
</tr>
<tr>
<td><strong>System default</strong></td>
<td>““</td>
</tr>
<tr>
<td><strong>Use</strong></td>
<td>The file is useful for adding extra symbols and lines to the plot with more control than can be had by using 'points' and 'lines'. For example, symbol and line properties can be varied from point to point, and tags can be used within the file.</td>
</tr>
</tbody>
</table>

Normal PhreePlot input file conventions apply. No header line is required but can usefully be included – the first non-comment line is ignored if the first word is not an integer (i.e. plot number). Columns should be consistently numeric or character.

The optional data separator determines the parsing of the data file. By default it is treated as a standard input file in which any (commas, tabs or spaces) are treated as a valid separators and multiple white space characters are merged. If a specific separator is specified, e.g. “\t” for the tab character as from a file created in a spreadsheet, this will be used instead. In this case, adjacent separators will not be merged. This enables empty data fields to be read.

Each row (line) represents a data point. A blank row signifies a break in the data. This will introduce a break in a line. The line is only actually drawn when all of the data for the block of data have been read.

The last defined parameters (width, colour and type) override any previously defined parameters. All line and symbol properties remain in force until redefined, i.e. they persist between data blocks.

Providing a line consists of three or more points, i.e. at least two line segments, and is a full line (lineType = 1), the plotted line will have rounded joins and ends. If butt (square) ends are wanted, make sure that only single line segments are drawn, i.e. insert a line break after every pair of points.

This file is often best prepared in a spreadsheet and saved as a tab-delimited file (add /t after the filename to indicate the format).

The file has the following columns of data:

```
plotnumber,x,y, [lw, [linecol, [isymb, [sizesymb, [symbolcol, [rim-```
color, [rimfactor, [linetype, [dashesperinch]]]],]]]

The file must have at least 3 columns to plot a line and at least 6 columns
to plot a symbol. If an error occurs while reading a column, a default
value is substituted.

plotnumber(column 1) is the plot number (starts at 1) for which the data
applies. The plot number is sequential and is printed in the info at the
bottom left-hand corner of each plot. auto or '999' means all plots. 0
means no plots.

x(column 2) and y(column 3) are the x- and y-coordinates for the points
in plot coordinates (i.e. as seen on the screen or page).

lw(column 4) is the line width in whatever length units are in force. Negative
line widths produce dashes if the linetype has not been explicitly
defined. Default = lineWidth.

linecol(column 5) is the colour of the line (including nd). Default = lineColor.

isymb(column 6), sizesymb(column 7) and symbolcol(column 8) refer to
the symbol code number or name, the symbol size and the symbol colour.
Default = 1, 0.0 and pointColor, respectively.

rimcolor(column 9), rimfactor(column 10) define the appearance of
the rim in those symbols (1-6) that have one. Default = 'nd' and 0.05
respectively.

linetype (column 11), dashesperinch (column 12) are the line type
(default = lineType(1)) and dashes per inch (default = dashesPerInch(1)).

Additional columns are ignored.

By selecting the appropriate size and colour of the symbols and lines it is
possible to have combinations of lines and symbols of any colour.

The default is for the symbols and lines to be clipped to the plot area. If
the symbols and lines are to be plotted outside the plot area, put the string
' noclip' (case insensitive) on a separate line anywhere in the appropriate
block of data, most logically at the beginning. Clipping of lines will be
turned off from this block of data forward. It can be turned on again by
adding the string 'clip' at the appropriate place in the file. Clipping of
symbols, which do not depend on a set of data, obeys the preceding
'noclip/clip' statement.

See Section 7.12 for more details and a display of the available symbols
and their symbol codes (isymb) and names.

Comments can be included in the file by using a # symbol but blank com-
ment lines do not count as blank lines.

See Section 2.4.3 for how to ensure that the file will be found.

Examples

66, 69

extraText

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename [data separator]</td>
<td>An optional file containing details of additional text to be added to the</td>
</tr>
</tbody>
</table>
plot.

Aliases

System default

Use

The file is useful for adding extra text to the plot. Each logical line will plot a piece of text. There is no limit to the number of lines. Normal PhreePlot input file conventions apply. Comments can be included in the file by using a # symbol.

See ‘Section 2.4.3’ for how to ensure that the file will be found.

The second optional parameter is the data separator used for reading the file, e. g. ”\w” for whitespace (tab or space), “\t” for tab only, “\” for whitespace or comma, the default.

Each line in the file has the following structure:

plotnumber,x,y,text[,size[,colour[,angle[,justify[,digits[,font]]]]]]

plotnumber is the plot number (starts at 1) for which the text applies. If the info block is printed, the plot number will be printed at the beginning. ‘auto’ means all plots. A value of 0, or any other out of range number, would suppress plotting.

The plotnumber increments by one for each plot produced. The outer loop is the z loop and the inner (most rapidly changing) loop is the main species loop. so in a run with m-elements and n-loop (z) values, the order of plots will be:

z1-el1, z1-el2,...z1-elm, z2-el1, z2-el2, ...z2-elm, ...zn-el1,zn-el2,...zn-elm.

x and y are the x- and y-coordinates for the text position in plotting coordinates (i.e. as seen on the screen or page). x and y can take on special values: x = ‘auto’ (case insensitive) sets x just to the right of the x axis (at the start of the key); y = ‘auto’ sets y to just below the key. The text is horizontally justified according to the justify parameter (0=left, default; 1=centre; 2=right justification) and is always vertically aligned such that y refers to the baseline of the first line of text. The default is therefore for x and y to refer to the bottom left of the top line (if the text string contains any <br>s). If x is set to ‘last’ (case insensitive) then the last x value is used. If y is set to ‘last’ then y is set to one line below the last line. Setting both x and y to ‘last’ means that the text is continued below where it previously finished. This is useful to overcome the 200 character limit to the length of a plotted text string.

text is the text string, enclosed in quotation marks if necessary. It can be of any length but is truncated to a maximum length of 200 characters including tags when actually plotted. It can include the normal text enhancement tags for sub- and superscripts, line breaks and so on (Section 7.6.3). In addition, there are a number of special tags to load variable text relating to the current simulation and plot. These tags are: <input:s1,s2> to copy Phreeqc text from the input file; <legend> to move the legend to a different place; <mainspecies> to plot the name of the main species and <loop> to plot the value of loop variable. When these tags are used, they may impose their own layout rules which override the given ones.

Other tags can also be included and will be substituted at plot time if defined; if not defined, they will be plotted as is.
size is the size of the text in the current units.

color is the colour of the text.

angle is the angle of the text measured in degrees from the horizontal rotating clockwise. 0 is horizontal and upright. The text is rotated about the left, centre, right position of the baseline of the first line of text depending on justification.

justify is the justification with respect to the x, y coordinates. 0 = left-justified, 1 = centre, 2 = right-justified.

digits refers to the number of decimal places when substituting values for numeric tags (1<=digits<=16). If digits is a negative number, then trailing zeros will also be removed. Exponential format is normally used when abs(value) is less than 0.001. This value is ignored for non-numeric text strings.

font refers to the font (name or number) given by the font keyword, or 'Helvetica' if undefined.

The special plot tags which are evaluated after computations are <pxmin>, <pxmax>, <pymin> and <pymax>. These contain the current values of pxmin, pxmax etc and can be used to generalise plotting positions.

The size of the text, its colour, angle, justification and digits are optional though the order must be maintained. For example, if the justify parameter needs to be included, then all of the other preceding ones also do.

Default values for the parameters are:

\[\text{size} = \begin{cases} \text{legendTextSize} & \text{if } >0 \\ \text{labelSize} & \text{if } >0 \\ \text{plotTitleSize}/2 & \end{cases} \]

color = black

angle = 0 (horizontal and upright)

justify = 0 (=left)

digits = -3

Once the value for a parameter has been set, this value stays in force for all subsequent entries in the file until it is redefined.

Examples 3, 51, 55

fillColorDictionary

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>filename</td>
</tr>
<tr>
<td>Description</td>
<td>A text file that is used to store a list of chemical species and their associated fill colours. This is used during the plotting of predominance and mineral stability diagrams.</td>
</tr>
<tr>
<td>Aliases</td>
<td>fillFile, FillDict, FillColor</td>
</tr>
<tr>
<td>System default</td>
<td>fillColor.dat</td>
</tr>
<tr>
<td>Use</td>
<td>This is a file containing a list of species names and associated colours used for the colouring of polygons. If the specified file exists, it will be used and any additional species auto-</td>
</tr>
</tbody>
</table>
Colours can be changed by editing this file and replotting. If the specified file does not exist, it will be created in the input file directory with the filename given by the `fillColorDictionary` setting.

### FIT

<table>
<thead>
<tr>
<th>Value</th>
<th>none (section heading)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Optional section header</td>
</tr>
<tr>
<td>Aliases</td>
<td>System default</td>
</tr>
<tr>
<td>Use</td>
<td>Can be used in the input file to highlight the beginning of FIT keywords. There is no attribute associated with it.</td>
</tr>
</tbody>
</table>

### fitAdjustableParameters

<table>
<thead>
<tr>
<th>Value</th>
<th>list of 0 or 1's</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Flags whether fit parameters are fixed (0) or adjustable (1)</td>
</tr>
<tr>
<td>Aliases</td>
<td>System default</td>
</tr>
<tr>
<td>Use</td>
<td>The list should be of length given by the number of fit parameters. Each parameter has the value of 0 or 1 signifying whether it is fixed or adjustable.</td>
</tr>
<tr>
<td>Examples</td>
<td>80, 83</td>
</tr>
</tbody>
</table>

### fitConvergenceCriterion

| Value | positive number ['rss' | 'absdiff' | 'function' | 'L1' | 'L2'] |
|-------|------------------------|
| Description | Specifies the convergence criterion used during fitting and the type of objective function. |
| Aliases | fitConvergenceCriterion, RHOEND |
| System default | 1e-6 rss |
| Use | Controls when fitting is deemed to have converged and, optionally, the objective function used. L1 is a synonym for absdiff and L2 is a synonym for rss. The default objective function is 'rss' (the L2 norm, least squares) unless one of the other options is specified and the fitting method is appropriate. 'absdiff' uses the sum of the weighted absolute differences (the L1 norm) whereas 'function' simply uses the calculated value and can be used for finding the minimum in a 'function'. The 'nlls' and 'lm' methods are specifically designed for the L2 norm and cannot be used with the L1 norm. |
Interpretation depends on the algorithm chosen (see Section 12.5.4 and Table 12.1). For 'nlls', a 'normal' termination is triggered when the objective function is predicted to be less than \( \text{fitConvergenceCriterion}^2 \). Since the objective function is itself a summation this will depend on the number of observations.

For the 'trust region' methods, \( \text{fitConvergenceCriterion} \) defines the radius of the final trust region, \( \text{RHOEND} \). This should indicate the accuracy that is required in the final values of the parameters.

A value of 1e-6 (default) to 1e-3 is normally reasonable.

Small values will always provide more accurate fits but at a cost in terms of the number of iterations required.

Normally if this setting is set at a very small value, the lack of a significant change in parameter values will also trigger termination.

**Examples**  
80, 83

---

### fitFiniteDiffStepSize

<table>
<thead>
<tr>
<th>Value</th>
<th>positive number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specifies the size of the interval that is used by the 'nlls' fitting routine to calculate numerical derivatives by finite differences.</td>
</tr>
<tr>
<td>Aliases</td>
<td>finiteDiffStepSize</td>
</tr>
<tr>
<td>System default</td>
<td>1e-3</td>
</tr>
<tr>
<td>Use</td>
<td>The size should be sufficiently large to give rise to a significant change in the response of the dependent variable since each of the ( n ) adjustable parameters is adjusted by this amount during the first ( n+1 ) iterations. However, if the setting is too large then the fitting may wander too far from the optimal solution, and possibly into territory where Phreeqc fails or where convergence of the fitting is not achieved.</td>
</tr>
</tbody>
</table>

**Examples**  
80, 83

---

### fitLogParameters

<table>
<thead>
<tr>
<th>Value</th>
<th>a list of 0 and 1's</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specifies whether the fit parameters are to be log10 transformed (1) or not (0) during fitting.</td>
</tr>
<tr>
<td>Aliases</td>
<td>all 0's</td>
</tr>
<tr>
<td>System default</td>
<td>The list should be the same length as indicated by the number of fit parameters. Each parameter should have a 0 or 1 associated with it.</td>
</tr>
<tr>
<td>Use</td>
<td>A value of 0 fits the parameter as given. A value of 1 indicates that the parameter will be anti-logged ( (10^x) ) before being substituted in the model so the parameter values specified by \text{fitParameterValues} \ should be log10 values.</td>
</tr>
</tbody>
</table>

**Examples**  
80, 83
This option can be useful to: (i) restrict a parameter to positive values; (ii) fit parameters that can vary by orders of magnitude.

Although this option will not necessarily affect the final fit, it can affect it because the rescaling will affect how the parameters are adjusted between steps. For example, the `fitFiniteDiffStepSize` applies to the parameters in the original log space as does the `fitStepSize`.

Examples  

**fitLowerParameterValues**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specifies the minimum allowable value of each adjustable parameter during fitting.</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td>UNDEFINED UNDEFINED</td>
</tr>
<tr>
<td>Use</td>
<td>Used for constrained optimization ( = ‘bobyqa’ only).</td>
</tr>
</tbody>
</table>

There should be one value for each parameter, corresponding one-to-one with the other parameter lists such as that of `fitParameterValues` and length defined by `numberOfFitParameters`. Values should be included for ‘fixed’ parameters to maintain the correspondence of the lists. These values will not be used.

A value of UNDEFINED (or -99999) means that no constraint will be applied (it is automatically set to a huge negative value).

**fitMaxIterations**

<table>
<thead>
<tr>
<th>Value</th>
<th>positive integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Controls the maximum number of iterations allowed during fitting</td>
</tr>
<tr>
<td>Aliases</td>
<td>fitMaxIteration, maxIterations, maxiteration</td>
</tr>
<tr>
<td>System default</td>
<td>500</td>
</tr>
<tr>
<td>Use</td>
<td>An ‘iteration’ is one set of calculations of the residuals for all observations. <code>fitMaxIterations</code> can be varied to either allow more time for convergence or to deliberately force an early exit, e.g. after one iteration.</td>
</tr>
</tbody>
</table>

**PhreePlot** will attempt to exit gracefully after the maximum iterations have been reached. This includes reporting the optimal parameter values (so far) and their errors, and plotting the results. A low-pitched beep will be given if the sound is on and an error message written to the log file and screen, if active.

Sometimes a few extra function calls are made after the maximum iterations have been reached in order to restore the previous best estimates of the fit.

Examples  

### fitMethod

<table>
<thead>
<tr>
<th>Value</th>
<th>List of one or more of ‘nlls’, ‘lm’, ‘newuoa’, ‘bobyqa’, ‘subplx’ or ‘contour’ (case is not significant)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specifies the optimization (fitting) procedure(s) used</td>
</tr>
<tr>
<td>Aliases</td>
<td>‘nlls’</td>
</tr>
<tr>
<td>System default</td>
<td>‘nlls’</td>
</tr>
<tr>
<td>Use</td>
<td>Five optimization procedures are currently available: ‘nlls’, ‘lm’, ‘newuoa’, ‘bobyqa’, ‘subplx’. These are all derivative-free methods. They are all unconstrained except ‘bobyqa’ which accepts lower and upper bounds on the parameters. See &quot;Fitting and simulations&quot;, p. 143. The ‘nlls’ method is the only one that will give estimates of the standard errors of the fitted parameters, and the correlation matrix between these parameters. If more than one optimization method is given, then the fit will be re-run with all settings the same except for the optimization method. A plot will be produced for each run. The method, e.g. “_nlls”, will be appended to the various output filenames to distinguish them. Beware that some of the stepping and convergence parameters have somewhat different meanings with the different methods and if this is important, run each fit with a separate file and adjust the parameters accordingly. The ‘contour’ option is used to produce a contour plot of the objective function versus two user-defined variables, most usefully two of the model parameters being fitted. The objective function, defined by the contourZvariable, is the z-variable and xmin, xmax, ymin, ymax and the resolution drive the calculation of the objective function. The &lt;x_axis&gt; and &lt;y_axis&gt; tags must be present explicitly in the list of fitParameterValues. These will be substituted at run time. If necessary, these values can be passed to other tags used in the model via the numericTags settings. They must also be given parameter names. The resolution, nres, should be set to 2 or greater.</td>
</tr>
</tbody>
</table>

### fitnpt

<table>
<thead>
<tr>
<th>Value</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specifies the number of conditions or interpolation points (NPT) used by the NEWUOA and BOBYQA optimization algorithms.</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td>UNDEFINED</td>
</tr>
<tr>
<td>Use</td>
<td>This value must be in the interval [n+2, (n+2)*(n+1)/2] where n = the number of adjustable parameters. A larger value will provide more accuracy but at a cost. If the value is set to UNDEFINED, then the chosen value will depend on the value of n:</td>
</tr>
</tbody>
</table>
if \( n < 6 \), then \( NPT = \frac{(n+2)(n+1)}{2} \) (the maximum) 
else \( NPT = 2n+1 \) (the recommended value for large problems).

---

**fitParameterNames**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of character strings (up to 30 characters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specifies the names of each of the fit parameters</td>
</tr>
<tr>
<td>Aliases</td>
<td>“”</td>
</tr>
<tr>
<td>System default</td>
<td>“”</td>
</tr>
<tr>
<td>Use</td>
<td>Specifies the names of the parameters (fixed or adjustable) used in the model. These names are used to make tags which can be used within the CHEMISTRY section (the <em>Phreeqc</em> code). These names must not be used in other tag definitions, e.g. numericTags.</td>
</tr>
</tbody>
</table>

**Examples** 80, 83

---

**fitParameterValues**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specifies the initial values of each of the fit parameters.</td>
</tr>
<tr>
<td>Aliases</td>
<td>missing value</td>
</tr>
<tr>
<td>System default</td>
<td>missing value</td>
</tr>
<tr>
<td>Use</td>
<td>A value must be assigned to each parameter. It will either be treated as fixed or adjustable depending on the values of fitAdjustableParameters. Normally during fitting, the optimizer takes complete control of setting these values so they cannot be manipulated from outside – you cannot assign tag values to them. The exception is with the fitMethod ‘contour’ when you can – indeed, must – since in this case the optimizer is not being used. Rather the parameter values are being driven by the x- and y-axis variables and so their corresponding tags (&lt;x_axis&gt; and &lt;y_axis&gt;) are required here.</td>
</tr>
</tbody>
</table>

**Examples** 80, 83

---

**fitStepSize**

<table>
<thead>
<tr>
<th>Value</th>
<th>positive number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>During fitting, controls the maximum size of a step that can be taken.</td>
</tr>
<tr>
<td>Aliases</td>
<td>RHOBEG, fitMaxStepSize, stepSize</td>
</tr>
<tr>
<td>System default</td>
<td>100</td>
</tr>
<tr>
<td>Use</td>
<td>The interpretation of this parameter depends on the fitMethod used. With ’nlls’, it controls the minimum size of the Marquardt parameter</td>
</tr>
</tbody>
</table>

---
(along with the fitConvergenceCriterion) and so influences the size of the steps taken – the larger its value, the larger the step sizes. With the ‘trust region’ methods, it defines RHOBEG, the initial radius of the ‘trust region’. This radius is subsequently reduced as the algorithm converges to a solution. RHOBEG and so fitStepSize should be about one tenth of the expected greatest change to an adjustable parameter (hence the importance of some approximate scaling of the adjustable parameters).

This parameter should not be so small as to lead to an insignificant shift in the objective function during fitting or so small as to make progress painfully slow. Neither should it be so large that it allows the parameter values to wander into ‘undesirable’ territory causing Phreeqc to fail to converge.

Examples $80, 83$

---

### fitUpperParameterValues

**Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of numbers</th>
</tr>
</thead>
</table>

**Description**

Specifies the maximum allowable value of each parameter during fitting.

**Aliases**

System default: UNDEFINED

**Use**

Used for constrained optimization (= ’bobyqa’ only).

There should be one value for each parameter, corresponding one-to-one with the other parameter lists such as that of fitParameterValues and length defined by numberOfFitParameters. Values should be included for ‘fixed’ parameters to maintain the correspondence of the lists. These values will not be used.

A value of UNDEFINED (or -99999) means that no constraint will be applied (it is automatically set to a huge positive value).

---

### fitWeightingMethod

**Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>0, 1 or 2</th>
</tr>
</thead>
</table>

**Description**

Controls how the residuals are weighted in the objective function used by the fitting algorithms.

**Aliases**

weighting

**System default**

0

**Use**

The objective function to be minimized, $s$, is given by:

$$ s = \sum_i \left[ w_i \times (f_i - \hat{f}_i) \right]^2 $$

where $f_i = \text{observed value for observation } i$, $\hat{f}_i = \text{fitted value for observation } i$, $w_i = \text{weight for observation } i$. Note $w_i$ is inside the square.
The `fitWeightingMethod` setting can take the following values:

0 = unit weighting (all \( w_i = 1 \))

1 = relative error weighting (\( w_i = 1/\hat{f}_i \))

2 = weights (\( w_i \)) read from the data file

Ideally, the weights should be equal to the standard deviation of each observation. This can be estimated from repeat measurements. It may vary with the magnitude of \( f_i \).

If `fitWeightingMethod = 2` is used, then the column containing the weights is given by the `weightColumn` setting.

Examples: 80, 83

---

### font

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[font] [character encoding]</code></td>
<td>Optionally one or both of, the name or number of the font family or font (up to 40 characters) and/or the character encoding to use, all case insensitive. The encoding must follow the font if present.</td>
</tr>
</tbody>
</table>

**Aliases**

- System default: Helvetica Latin-1

**Use**

All 35 standard Postscript fonts are available. The font can be defined by its name, number or font family. Only one font family is used for the basic text in a plot (axis titles and numbering etc.). However, entries in `extraText` files can define which font to use.

The character encoding can be either ‘Standard’, ‘Latin-1’ (or ‘Latin1’) or ‘ASCII’ (see Appendix 4). See the discussion of how to input special characters in Section 7.6.1.

Fonts are numbered consecutively based on their order of appearance in `PhreePlot`'s default font table, or in the `fonts.dat` file, if present, scanning across and down the table, i.e. 1 = Helvetica, 2 = Helvetica-Oblique, 3 = Helvetica-Bold, ..., 44 = Dingbats (see Table below). Font numbers outside this range lead to a fatal error.

The following eight font families are the default in `PhreePlot`: Helvetica, Helvetica-Narrow, Bookman, Avantgarde, Times, Palatino, NewCentury-Schoolbook and Courier. The Chancery (only the italic font), Symbol and Dingbats fonts are also defined. The Symbol font contains Greek characters and some common plotting symbols. The Ghostscript distribution normally contains all these fonts.

The principal text fonts look like this:

**Dingbats** are mostly iconic symbols and are only approximately centered.

The above font families and their various faces make up the 35 standard Postscript fonts.

The specified font is checked against the current fonts table first checking for a specific font and if this is not found, checking for the name of a font family. If a font family is found, the ‘regular’ face of this font is used as the
212  PhreePlot Guide

Helvetica: The quick brown fox jumps over a lazy dog 0123456789 italic bold
Helvetica-Narrow: The quick brown fox jumps over a lazy dog 0123456789 italic bold
Bookman: The quick brown fox jumps over a lazy dog 0123456789 italic bold
Avantgarde: The quick brown fox jumps over a lazy dog 0123456789 italic bold
Times: The quick brown fox jumps over a lazy dog 0123456789 italic bold
Palatino: The quick brown fox jumps over a lazy dog 0123456789 italic bold
NewCenturySchoolbook: The quick brown fox jumps over a lazy dog 0123456789 italic bold
Courier: The quick brown fox jumps over a lazy dog 0123456789 italic bold
Chancery: The quick brown fox jumps over a lazy dog 0123456789 italic bold

Standard fonts and their numbering as defined in fonts.dat

<table>
<thead>
<tr>
<th>Font_family</th>
<th>Regular</th>
<th>Italic</th>
<th>Bold</th>
<th>Bold-italic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dingbats</td>
<td>41. Dingbats</td>
<td>42. Dingbats</td>
<td>43. Dingbats</td>
<td>44. Dingbats</td>
</tr>
</tbody>
</table>

base font (i.e. plain text). If the name of a specific font face is found, this will be used as the 'base font'. It is usually best to use the name of a font family or the name of the regular face font as the base font. Then bold and italic can be turned on with the tags `<b>` and `<i>`, respectively. If a bold font is chosen as the base font, then the bold cannot be turned off using the tags.

How software reacts to these fonts depends on whether the output device recognises the specified fonts either in terms of having the font of the exact same name available or being able to provide an appropriate substitution font (there is no standardization of font names and so similar fonts can have more, or less, different names).

The fonts.dat file contains the names of the 11 font families and the 35 fonts defined in PhreePlot. These are the names of the fonts actually written to the Postscript file. This file can be edited to give the required font names for the device of interest based on the available fonts.

The fonts.dat file has a standard format and consists of eleven rows of
data, each line containing the family name of the font (as used by \texttt{font}) and then the font names for the regular font, the italic (or oblique) font, the bold font and the bold, italic font. A blank string (""") is a placeholder that indicates that no such font is defined. This file is read in free format.

It is in principle possible to use different font families for plain text, italic, bold and italic-bold by editing the \texttt{fonts.dat} file.

The search path for the font file is the current directory followed by the system directory.

If \texttt{font} is not one of the specified font families, then a close match is sought – if the given font includes a font family name within it, this font family is chosen. Also if \texttt{font} contains any of the words below, the corresponding font family is selected:

\begin{itemize}
  \item arial
  \item Helvetica font family
  \item roman
  \item Times font family
\end{itemize}

Text tags provide a way of specifying changes to the appearance of individual characters (bold, italic) and groups of characters. They can also specify Greek characters (these are converted to the Symbol font).

The 'Standard' character encoding consists of the standard ASCII 7-bit character set ('ASCII' encoding, decimal codes 0-127) plus a variable range of extended characters often including the permil sign and the oe ligature. Not all codes are defined in Postscript fonts.

The 'Latin-1' character encoding consists of the ANSI (default for your system) 8-bit character set (decimal codes 0-255), often known as the ISO-8859-1 encoding. This includes most of the accented characters used in Western European languages (but not the euro sign or the oe ligature) and apart from these exceptions, is broadly similar to the commonly-used Windows-1252 character set.

A table of the current character set and font can be produced if \texttt{plotTitle} is set to 'character set', \texttt{labelsize} set to about 2 mm and 'A4' or 'letter' \texttt{paperSize} chosen. Most other settings are ignored.

\section*{gridColor}

\begin{tabular}{|l|p{10cm}|}
\hline
\textbf{Value} & One to six cohort colours \\
\hline
\textbf{Description} & Defines the colour of the grid lines for the six axes \\
\hline
\textbf{Aliases} & gridColors, gridLineColor \\
\hline
\textbf{System default} & \texttt{black} \\
\hline
\textbf{Use} & Defines the colours used for drawing grid lines. \\
\hline
The six colours refer in order to the following six axes: major x, minor x, major y, minor y, major 2y and minor 2y. \\
\hline
If less than six colour are entered, then the missing colours are automatically filled in by recycling the colours given. See \texttt{tickSize} for the recycling rules that apply.
\end{tabular}
gridDashesPerInch

- **Value**: One to six non-negative numbers
- **Description**: Defines the number of dashes per inch for the six grid lines
- **Aliases**: System default 20
- **Use**: The six numbers refer in order to the following six axes: major x, minor x, major y, minor y, major 2y and minor 2y.

If less than six numbers are entered, then the missing numbers are automatically filled in by recycling the numbers given. See tickSize for the recycling rules that apply.

Grid lines are only drawn if the corresponding gridLines setting is true or the corresponding tickSize is very large.

See gridLineType below for choosing the line style.

gridLines

- **Value**: One to six logical values
- **Description**: Defines whether a grid line is drawn or not
- **Aliases**: gridline, grid
- **System default**: FALSE
- **Use**: This keyword is used to determine if a grid line is drawn (true) on a plot or not (false). The values refer in order to the following six axes: major x, minor x, major y, minor y, major 2y and minor 2y.

The major axes are where the major ticks and axis numbers are positioned; the minor axes are where the minor ticks are positioned.

If less than six values are entered, then the missing values are automatically filled in by recycling the values given. See tickSize for the recycling rules that apply.

The 2y ticks and grid lines are only drawn if some 2y lines or points are actually plotted.

Grid lines can also be set by specifying very large tick sizes (see tickSize).

Other settings (gridLineType, gridDashesPerInch and gridColor) affect the appearance of the grid lines and even whether they are drawn. AxisLinewidth controls the width of the grid lines.

gridLineType

- **Value**: Form one to six integers in the range 0 to 20
Depicts the styles of line drawn for the major and minor grid lines

**Aliases**: gridLineTypes

**System default**: 1

**Use**: This keyword is used to determine the style of a grid line. The default (1) is for the grid lines not to be dashed. **Line styles** can be solid, dashed, dotted and dot-dash.

The six numbers refer in order to the following six axes: major x, minor x, major y, minor y, major 2y and minor 2y.

If less than six numbers are entered, then the missing numbers are automatically filled in by recycling the numbers given. See **tickSize** for the recycling rules that apply.

Grid lines are only drawn if the corresponding **gridLines** setting is `TRUE`.

See **lineType** for a full description of the 20 line types, e.g. useful styles are 0 = no line; 1 = full line; 6 = dashed line; 11 = dotted line; 15 = dash-dot line.

---

**info**

**Value**: Cohort colour for the info data block

**Description**: Sets the colour of the ‘info’ block printed at the bottom left-hand corner of the plot.

**Aliases**: infoColor

**System default**: nd [nd]

**Use**: One or two **colours** must be specified: the first is for the whole ‘info’ block. The second is for the file path of the input file and is only used when the first colour is ‘nd’ or blank. If the second colour is omitted or the first colour is not ‘nd’ or blank, then the first colour is used for both.

The info block contains summary information about the figure. The information given varies slightly depending on the type of calculation. An example is shown below for a ‘ht1’ calculation:

```
1 Main species = Fe     Temperature = 25.0 °C
Resolution = 250     Speciation calculations = 2544; Time = 0.251 min
C:\Program Files\PhreePlot\0.01\demo\Fe\hfo_Fe1.ps
PhreePlot version = Pre-release 0.01 (27 Jun 2008)
Speciation program = PHREEQC (4 April 2007)
Database = wateq4f.dat (8 Sept 2006)
12:25:47 27 June 2008
```

The printing of the ‘info’ block can be turned off by setting the first colour to ‘nd’ or blank and giving no second colour.

‘nd’ as the first parameter also turns off the printing of any “<input:” text specified in an **extraText** file (“<input:” enables a copy of all or some of the input file(s) to be printed with the plot). ‘nd’ overrides any input settings in this file and so can be used to produce ‘clean’ plots without editing the individual extraText files. This can be set in the override.set file to ensure that this text is missing from all plot files whatever the input file states.

The size of the info text is automatically determined from the **labelSize** setting. If **labelSize**>=0.1 inch (or the equivalent on other scales) then
the text size will be \(0.6 \times \text{labelSize}\). If it is less than 0.1, then it is fixed at 0.05 inch.

Example: \(38\)

### initialValue

<table>
<thead>
<tr>
<th>Value</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Value given to all undefined numeric tags</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td>UNDEFINED (-99999)</td>
</tr>
<tr>
<td>Use</td>
<td>A numeric tag should normally be defined in terms of constants and other tag values that have already been defined, i.e. that precede the given tag in terms of evaluation order. However sometimes this is not appropriate and it is necessary to set an initial value, e.g. (&lt;n&gt; = &quot;&lt;n&gt; + 1&quot;)</td>
</tr>
</tbody>
</table>

### jobTitle

<table>
<thead>
<tr>
<th>Value</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Description of job</td>
</tr>
<tr>
<td>Aliases</td>
<td>job</td>
</tr>
<tr>
<td>System default</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>Use</td>
<td>Free text used to describe the job (up to 200 characters long). This string is printed in the log file so can be used for comments about the job.</td>
</tr>
</tbody>
</table>

### jpg

<table>
<thead>
<tr>
<th>Value</th>
<th>logical [number]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines whether the plot output (if any) is converted to a file in the jpg format.</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td>true</td>
</tr>
<tr>
<td>Use</td>
<td>A jpg file can only be produced if Ghostscript/GSview is installed. PhreePlot makes use of the Ghostscript \texttt{jpeg} device to produce this file. The second, optional parameter specifies the resolution (in dpi) to use when making the conversion. The default is 300 dpi. An example of output in jpg format is given below (the resolution seen here has been reduced during conversion to pdf – the quality of the original jpg was better than this).</td>
</tr>
</tbody>
</table>
**Example**  

---

### labelColor

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Cohort colour</td>
</tr>
<tr>
<td>Description</td>
<td>Sets the colour of labels in custom-type plots</td>
</tr>
<tr>
<td>Aliases</td>
<td>labCol</td>
</tr>
<tr>
<td>System default</td>
<td>black</td>
</tr>
<tr>
<td>Use</td>
<td>Specify the colour to be used for labels. Use ‘nd’ (not drawn) if no labels are wanted. This also controls the default color of the text used by the <code>&lt;input...&gt;</code> tag. Colours should be chosen from the colour palette. If ‘auto’ is set as the colour, then the colour of the label in the plot and legend will match that of the line, or symbol rim color if the line has not been plotted. If the rim colour is not defined or is ‘white’ or ‘nd’, the point colour is chosen instead. This setting does not apply to contour plots which have their own mechanism for labelling contours, and does not make much sense for predominance plots since the label will be largely hidden. It is principally of use for custom plots and their derivatives, e.g. species plots.</td>
</tr>
<tr>
<td>Example</td>
<td>51</td>
</tr>
</tbody>
</table>

---

### labelEffort

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0, 1, 2 or 3 [number]</td>
</tr>
<tr>
<td>Description</td>
<td>Controls the amount of effort used and time taken to place the labels for lines plotted with custom and fit plots so as to minimize the amount of overlapping and other aspects of poor label placement.</td>
</tr>
<tr>
<td>Aliases</td>
<td>effort</td>
</tr>
</tbody>
</table>
System default: 1

Use:
- Controls the amount of effort (and time) used to optimize the placement of labels for lines in custom and fit plots.
  - 0: No optimization; labels are placed half-way along the x axis
  - 1: Basic
  - 2: Moderate
  - 3: High

If the optional second parameter is present, the time taken for label placement in custom or fit plots will be at most this time in sec.

The time taken varies greatly with the number of labels, their size and the degree of overlap found in the default placement (near the centre point of the curve). In order to reduce overlap, reduce the label size, change the axis scale, or reduce the number of curves plotted.

The Esc key can be used to interrupt the labelling. If the plotting is stopped at this point, the best placement found so far is used.

The labels can be repositioned manually by editing the line colour dictionary and using calculationMethod 2. The line colour dictionary contains the position of the centre of each label and the line colour. useLineColorDictionary should be set to 2 to give the dictionary priority over automatic labelling.

A value of 0 rarely produces worthwhile output.

The labels can be turned off altogether by setting labelColor to 'na' or labelSize to be 0.

Example:

```
45
```

---

**labelFile**

<table>
<thead>
<tr>
<th>Value</th>
<th>logical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Now redundant.</td>
</tr>
</tbody>
</table>

**Aliases**

System

Use:
- Originally a switch that controlled the creation and deletion of the labels (*.lab) file that is used by ht1. The labels file contains the name, position and angle of all of the labels used to label the fields in predominance plots.
- Since this file is essential for the operation of 'ht1' plots, the labels file is automatically created and retained with these plots.

---

**labels**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of strings (up to 30 characters each).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Can be used to override the default names used for labelling the curves on custom and fit plots</td>
</tr>
</tbody>
</table>
Keywords 219

Aliases label, loopNames, loopName, loopNumberNames

System default "" (use default names)

Use A list of names to be associated with each set of data plotted in a multi-loop or multi-curve custom plot. The order in which the curves are plotted is in the order shown in the legend. This order is the order that the data have been written to the appropriate plot data file(s) (the 'out' file or 'pts' file depending on the type of plot, and any extradat files) and will run sequentially through each z-loop value for each of the variables to be plotted.

This list of labels can be used to rename the entries in a custom plot legend – the labels are simply picked from this list in turn and used in the legend.

A single curve is designated by a contiguous set of records in a single column in the data file being read. Multiple curves are designated by having one or more blank lines in a column of data or by the data being derived from several columns, or both of these.

Blank lines are normally inserted in 'out' files at each change in the value of the z-loop variable. The third and fourth dataSeparators settings control the insertion of line breaks in generated data files during looping.

If the data are derived from fit data files, then blank rows are copied across from the input files to the output files preserving the column breaks.

Names are picked off the labels list as needed, one by one, for each data-set. All curves for a given variable and a given z-loop variable will be plotted first, then curves with different values of the z-loop variable will be plotted. The labels list is recycled if short.

If the labels keyword is set blank, then the column headers of the data being read will be used to generate default label names. If more than one subset of data is being plotted, then a subset identified will be appended to the label name. This takes the form columnHeader_subsetIdentifier. The subset identifier is the subset number. This is often the z-loop number.

Labels set with the labels keyword take precedence over those set by specifying an alphanumeric variable as the first column of a loopFile. These label names will always be used as is – they are never appended with the subset number, e.g. "_1" and so on.

The loop names can be used to label each iteration of the loop variable, or in the case of a data file, each separate subset of data as indicated by a break (blank line) in the plot data file (the 'out' and 'extradat' files for custom plots and the 'pts' file for fit plots).

The convertLabels setting controls whether there is an attempt to convert the loop names to Phreeqc formulae for the label plotting, or not.

Examples 61, 84

labelSize

Value non-negative number

Description Controls the size of the labels used in the plots and the size of the info
text.

<table>
<thead>
<tr>
<th>Aliases</th>
<th>labelHt, labelTextSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>System default</td>
<td>2</td>
</tr>
<tr>
<td>Use</td>
<td>Controls the size of the labels used in the ht1, fit and custom plots. The size of the text is also controlled by the units being used (default is mm). Also controls the size of the vertex number labels in a contour plot with the <code>contourShiftLabel</code> ‘n’ option. The size of the info text is also automatically set to 0.5 x <code>labelSize</code>.</td>
</tr>
<tr>
<td>Example</td>
<td>40, 67, 74</td>
</tr>
</tbody>
</table>

**legendBox**

<table>
<thead>
<tr>
<th>Value</th>
<th>box line width or <code>auto</code> [box line colour [box background colour [number [number [number [number]]]]]]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Adds a box around a legend</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td>0 black nd 0 0 0 0</td>
</tr>
<tr>
<td>Use</td>
<td>Adds a box around the legend in custom or contour plots. It is possible to specify a line width, line colour and background colour for the box though only the line width is mandatory. The minimum requirement is to set a line width. The default is zero which means no box is drawn. ‘auto’ for line width uses the <code>axisLineWidth</code>. The default line colour is black. The default background colour is ‘nd’ which makes the box transparent. To hide the background, choose a colour, including white. The final four numbers adjust the position of the legend box (but not its contents). These are the adjustments in inches, in the order: bottom, right, top and left. Measurements start from the left and bottom. These adjustments are made to the calculated positions and may be positive or negative.</td>
</tr>
</tbody>
</table>

**legendTextColor**

<table>
<thead>
<tr>
<th>Value</th>
<th>A Cohort colour.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines the colour of the legend text in custom, contour and grid plots.</td>
</tr>
<tr>
<td>Aliases</td>
<td><code>keyTextColor</code>, <code>keyTitleColor</code>, <code>legendTitleColor</code></td>
</tr>
<tr>
<td>System default</td>
<td>“auto”</td>
</tr>
<tr>
<td>Use</td>
<td>Enables the colour of the legend text to be changed.</td>
</tr>
<tr>
<td>Example</td>
<td></td>
</tr>
</tbody>
</table>
**legendTextSize**

<table>
<thead>
<tr>
<th>Value</th>
<th>non-negative number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines the size of the legend text in custom, contour and grid plots.</td>
</tr>
<tr>
<td>Aliases</td>
<td>keyTextSize, keyText, legendTitleSize</td>
</tr>
<tr>
<td>System default</td>
<td>“auto”</td>
</tr>
<tr>
<td>Use</td>
<td>Enables the size of the legend text to be changed. The units are defined by the units keyword. The default units are mm. “auto” sets the size to be the same as labelSize or 0.0 if this is undefined. If set to zero, no legend is drawn. legendBox draws a box round the legend.</td>
</tr>
</tbody>
</table>

**Example**

74

---

**legendTitle**

<table>
<thead>
<tr>
<th>Value</th>
<th>Character string or ‘auto’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>One way of specifying the legend title in custom, contour and grid plots.</td>
</tr>
<tr>
<td>Aliases</td>
<td>keyTitle, key, legendText</td>
</tr>
<tr>
<td>System default</td>
<td>“”</td>
</tr>
<tr>
<td>Use</td>
<td>Adds a title to the legend in custom plots. Maximum length is 200 characters. This can contain text enhancements such as <code>&lt;b&gt;...&lt;/b&gt;</code> and line breaks (<code>&lt;br&gt;</code>). It is justified to the left but this can be altered by adding leading spaces. The legend text can also be added with the <code>&lt;legend&gt;</code> tag in the extraText file. If present, this will override any legendTitle setting. By default, the legend is placed to the right of the plot. If you want to place it somewhere else, including inside the plot, use the <code>&lt;legend&gt;</code> approach. The colour of the legend title is controlled by legendTextColor. The ‘auto’ option only applies to residual sum of squares plots. If used here, the legend title is taken from the contourZvariable setting.</td>
</tr>
</tbody>
</table>

**Example**

73

---

**lineColor, lineColor2y**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of one or more Cohort colours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Controls the colours used for lines in ht1, custom and fit plots for the main y and 2y axes.</td>
</tr>
<tr>
<td>Aliases</td>
<td>col</td>
</tr>
<tr>
<td>System default</td>
<td>black</td>
</tr>
<tr>
<td>Use</td>
<td>In ht1 plots, lineColor(1) controls the colour of the line separating the</td>
</tr>
</tbody>
</table>
fields. In custom, species and fit plots, the effect of `lineColor` depends on the `useLineColorDictionary` setting. If `useLineColorDictionary` is 0, then `lineColor` sets the colours for the first n lines plotted where n = length of the list colours set by `lineColor`. Additional line colours are selected sequentially from the PhreePlot colour sequence ([Line colours and auto line colouring](#)). In effect, `lineColor` promotes the given colours up the colour sequence list.

If `useLineColorDictionary` is 1 or 2 and the species being plotted is specified in the line colour dictionary, then the dictionary colour is used in preference. If the species is not in the dictionary then the colour is chosen from the PhreePlot line colour sequence.

Colours should be chosen from the [colour palette](#).

---

### lineColorDictionary

<table>
<thead>
<tr>
<th>Value</th>
<th>filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specifies the filename for the line colour dictionary.</td>
</tr>
<tr>
<td>Aliases</td>
<td>linefile, lineDict</td>
</tr>
<tr>
<td>System default</td>
<td>lineColor.dat</td>
</tr>
<tr>
<td>Use</td>
<td>If the filename is blank or the specified file cannot be found, then the system default filename is used. The line colour dictionary is used to record and, if specified, control the colour of points and lines in plots. It also stores the coordinates of any line labels. The specified file is read and updated if found; otherwise it is created. Whether the colours and coordinates in the dictionary are used is determined by the <code>useLineColorDictionary</code> setting (see Section 7.9).</td>
</tr>
</tbody>
</table>

---

### lines, lines2y

<table>
<thead>
<tr>
<th>Value</th>
<th>character list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specifies which columns should be plotted as lines for datasets plotted on the main y (left) and 2y (right) axes.</td>
</tr>
<tr>
<td>Aliases</td>
<td>plotLines; plotLines2y, lines2y, line, line2y</td>
</tr>
<tr>
<td>System default</td>
<td>&quot;&quot;</td>
</tr>
</tbody>
</table>
| Use         | The list should contain the column names or column numbers of columns for which the lines are to be plotted. The names are case dependent. The names or numbers refer to the column of the file being used for plotting, e.g. the 'out' file for custom plots or the 'pts' file for fit plots. The order of plotting the lines is determined by the column order in the
'out' file.

Additional files can be added to the search path using the `extradat` keyword. These files must be in tabular format with a single header row defining the column names. One of these columns must be the same as the `customXcolumn` defined elsewhere.

Additional lines can be added to predominance diagram plots as well as custom plots (including fit and species plots). No legend is produced for lines added to predominance plots.

The 2y axis is the right-hand y axis which can have a different scale from the left-hand or main y axis.

Examples 55, 65

---

**lineType, lineType2y**

<table>
<thead>
<tr>
<th>Value</th>
<th>A list of numbers in the range 0 to 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Defines the styles of lines in predominance and custom plots</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td>1</td>
</tr>
<tr>
<td>Use</td>
<td>This keyword is used to determine the style of lines defined by lines in custom plots. Line styles can be solid, dashed, dotted and dot-dash. The default (1) is for the line to be a solid line. Styles in the range 2-10 give an increasing length of space to dash; style 11 is pure dots and styles 12-20 are dot-dash style with an increasing length of dash. By varying the <code>dashesPerInch</code>, <code>lineWidth</code> and <code>lineColor</code> settings, a wide variety of line styles can be achieved. The 2y version applies to the second y-axis. These line styles can also be used for the grid lines of a plot. Here the corresponding <code>gridLineType</code> keyword is used.</td>
</tr>
<tr>
<td>Example</td>
<td>See the \demo\linetype\linetype.ppi example.</td>
</tr>
</tbody>
</table>

---

**lineWidth, lineWidth2y**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Controls the line widths in plots including custom plots with datasets plotted according to the main y (left) and 2y (right) axes.</td>
</tr>
<tr>
<td>Aliases</td>
<td>width, lw</td>
</tr>
<tr>
<td>System default</td>
<td>0.3</td>
</tr>
<tr>
<td>Use</td>
<td>This keyword sets all line widths in the plot to the given values. The list controls the line widths for successive curves in a custom-based plot. The position of the entry used is based on the corresponding position of the variable name in the <code>lines</code> keyword. The list is recycled as necessary. The actual line width drawn is determined by the units in operation. The system default units are mm.</td>
</tr>
</tbody>
</table>
A value of 0.0 means that no line will be drawn.

A negative value draws a dashed line of width $ABS(lineWidth)$ with 10 dashes per inch by default.

The number of dashes per inch to use for a dashed line is set by dashesPerInch.

The width of lines separating fields in predominance diagrams is 0.66 times lineWidth (1).

Examples

61, 55, 23

---

**log**

<table>
<thead>
<tr>
<th>Value</th>
<th>logical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines whether a log file is produced.</td>
</tr>
<tr>
<td>Aliases</td>
<td>logFile</td>
</tr>
<tr>
<td>System default</td>
<td>T</td>
</tr>
</tbody>
</table>

Use

The log file is a text file containing feedback about the run and is especially useful for debugging. The amount of information sent to the log file is controlled by the debug parameter. This increases as debug is changed from -1 or 0, 1, 2 and 3. It can be a very large file when debug = 3.

The default value is TRUE and so some output may be sent to the log file before the redefined value comes into effect. Set the log setting in the .pp.set file to avoid this early output being sent to the log file.

---

**logDepVariable**

<table>
<thead>
<tr>
<th>Value</th>
<th>0 or 1 or -1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>During fitting, determines whether the calculated values of the dependent variable returned by Phreeqc should be log-transformed or un-log-transformed (exponentiated) before being used or not</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td>0</td>
</tr>
</tbody>
</table>

Use

If logDepVariable is set to 1, then the calculated value of the dependent variable returned by Phreeqc is log-transformed before being used in the objective function. This option avoids having to store log-transformed values of the dependent variable in the data file. Similarly if logDepVariable is -1, the value of the dependent variable returned by Phreeqc is un-log-transformed, i.e. $10^x$.

If logDepVariable is 0, the calculated value is used without transformation.

Note that the observed values of the dependent variable must be on the transformed scale.

log transforming the dependent variable has approximately the same effect as using a relative error weighting.
It is possible to return log values of the dependent variable by using the log10 function in the USER_PUNCH data block. In this case, logDepVariable should usually be set to zero since no further transformation needs to be applied.

There are a number of examples of logDepVariable in the \demo\iso\isologx.ppi series of files.

**logVariableIn**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Aliases</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>A list of 0, 1, -1, -, X (or x)</td>
<td>Determines the format by which to read columns of data from the data files used by fit and simulate, and whether to transform these data or not.</td>
<td>‘’ (empty string)</td>
<td>The default, an empty list, means that the format is determined by the first row of data in the data file. This classifies each column as either numeric or character depending on the data found. If anything other than this is wanted, for example forcing a numeric value to be read as a character or transforming the data, or skipping columns, then the format must be specified explicitly.</td>
</tr>
</tbody>
</table>

Format options for each column are:

0=numeric data  
1=numeric and positive data; apply a log10 transformation  
-1=numeric data, apply an anti-log10 (10**x) transformation  
X=character data (X is not case sensitive)  
-=skip column

The default empty list means that each column of data is read according to the formats derived from the first row of data. Reading in character data with a numeric format will produce a fatal error (the first 10 such errors will be reported).

If a list of formats is given this overrides the format derived from the first data row. If this list contains \( n \) items and is shorter than the number of columns found then only the first \( n \) columns will be read. If \( n \) is greater than the number of columns found, then only the first \( n \) items in the format list will be used.

The above list will also determine which columns of data are transferred to the 'out' and 'pts' files.

If requested, the data are transformed immediately on reading in. No knowledge of this is used further on in the processing.

If the dependent variable is to be read in from the file (i.e. observations are present), then its column should be given by the dependentVariableColumnObs setting and the logDepVariable should be used to indicate whether this variable needs to be log-transformed or not. However, the appropriate logVariableIn setting should also be set correctly. Care should be taken to ensure that both the observations and the calculated values of
the dependent variable are on the same scale. It is not possible for PhreePlot to check this at run time.

In a simulation (calculationType simulate), the dependent variable may or may not be present in the fit data file. If it is absent, then the logDepVariable will determine whether the calculated value is transformed or not.

There are a number of examples of different logVariableIn settings in the \demo\iso\isologx.ppi series of files.

---

**loopFile**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Aliases</th>
<th>System default</th>
<th>Use</th>
</tr>
</thead>
</table>
| valid file path to an existing loop file [data separator] | Optional name of a file containing a list or table of loop values. | “ (empty string) | If loopFile is defined and present, then loop values are read from this file, one or more values per line. If a header line is present in the file, this is used to name the tags that are created. These column headers must be unique – they should not clash with the names of other tags.

If the data are all numeric and no header line is present, then the data associated with column 1, 2, ...are automatically labelled with loop tags, <loop1>, <loop2>, ... This will take preference over the single <loop> tag generated from loopMin, loopMax and loopInt. In this case, <loop1> is copied to <loop> so that <loop> remains defined.

In general, for mixed tables with both numeric and character columns, PhreePlot attempts to determine the structure of the file based on the first two columns and first two rows of data. It is safest if the first row is a row of character values which serve as a header line and which are used to generate tag names for the columns of data below. Similarly, if the first column of data is all character values then this is treated as a column of row labels which can be used in plotting.

The header line (if present) and the remaining part of the file will be parsed, by default, according to the separator(s) specified by the first entry in the dataSeparators keyword. If the optional data separator string is specified with the loop file name, then this is used. The options for this separator are given in Section 5.2.7.

Columns should consist of all numeric or all character data. If the analysis of the first row of data indicates that a row is numeric, then any subsequent reading of non-numeric data in this column will result in any error and the entire row of data are discarded.

If the first two rows of the first column contains character variables (up to 30 characters), the rows are assumed to contain row names in the first column. These are used for labelling any plot that is produced from this data. However, if label names have been assigned by the labels keyword then these are used in precedence over those defined by a loop file.

A special case is a single column of character variables. Here the normal rules are ambiguous and the following assumption is made: the first row is assumed to be the column name (and will be used as a tag name) and the
remaining rows are character values. These are accessible by using the tag name. The loop tags, <loop1> etc, remain undefined since these are reserved for numeric variables.

If numeric values like 1, 2, 3... are wanted as label names, force them to be read as character strings by adding quotes, e.g. “1”, “2” etc. This only applies to column 1. Alternatively, if the labels are just used for plotting, you can add a non-plotting character (ASCII encoding) to the string, e.g. 1¬ to force it to be read as a character variable.

Quotes tend to get stripped from variables when read or may need to be added when a substituted character tag is to be read by Phreeqc so it may be necessary to include the tag in quotes, e.g. "<description>" or even use the two types of quotes, '"<description>"'.

The key feature of a loop file is that it ‘loops’. A new row of loop values will be associated with the appropriate tags on each iteration of the z-loop variable. This includes row names (potential labels) if present.

loopLogVar operates as normal but it operates over all of the numeric variables.

The normal search path is used for locating the specified loop file.

If loopFile is defined but the specified file is not found, then this is treated as a fatal error.

Normal conventions for reading input files apply to the loop file. This includes not using # or ; in label names. Comment lines can be included and will be skipped.

The values of the loop variables read from the loop file will be written to the log file and can be checked there.

Example

---

**loopIndexStartNumber**

Value: positive integer
Description: The initial z-loop number used to define a filename
Aliases: zstart, loopStartNumber
System default: 1
Use: By default this 1. This specifies the starting number used in the file extension when a series of files is output. This enables single plots to be created with any number, thus enabling the reploting of just one file when many were produced in a series (not tested).

---

**loopInt**

Value: non-negative number
Description: The value of the z-loop interval (>=0) used to calculate the value of the z-loop variable, <loop>.
Aliases: | zint
---|---
System default: | UNDEFINED
Use: If `loopInt > 0`, then the value of the `<loop>` variable is varied from `loopMin` to `loopMax` in steps of `loopInt`. For example, given the following loop parameters (`loopMin`, `loopMax`, `loopInt`, the `<loop>` values generated will be:

262, `<loop>` values generated: 2, 4, 6
263, `<loop>` values generated: 2, 5

If `loopInt=0`, then `<loop>` is set to `loopMin` and only one iteration is made. `loopMax` is therefore not be used and should not be defined otherwise a fatal error will be signalled (unless `loopMin=loopMax`).

`loopInt<0` is incorrect if `loopMax=loopMin` but will be corrected using:

```
LOOPINT = sign(LOOPINT,LOOPMAX-LOOPMIN).
```

The above assumes that `loopLogVar` is 0 (linear scale). If `loopLogVar=1` then the loop variable will be set to $10^{<loop>}$. `<logloop>` always contains the current value of $\log_{10}<loop>$.

If `<loop>` is used in an input file and if `loopInt` (or `loopMin` or `loopMax`) is undefined, then an error is reported.

**Example** 61

---

**loopLogVar**

<table>
<thead>
<tr>
<th>Value</th>
<th>0 or 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines whether the z-loop variable and z-loop interval are based on a log10 scale or not</td>
</tr>
<tr>
<td>Aliases</td>
<td>zvar, looplog, logLoopVar</td>
</tr>
<tr>
<td>System default</td>
<td>0</td>
</tr>
</tbody>
</table>
| Use | Either has a value of 0 or 1.  
0 = a linear scale; 1 = log10 scale. |

The following examples show how `loopLogVar` controls the z-loop variable, `<loop>`, given the values of `loopMin`, `loopMax`, `loopInt` shown below:

`loopLogVar = 0`

0, 10, 2 will generate values of 0, 2, 4, 6, 8, 10

`loopLogVar = 1`

-2, 2, 1 will generate values of 0.01, 0.1, 1, 10, 100

**Example** 61
**loopMax**

Value: number

Description: Maximum value of the z-loop variable

Aliases: zmax

System default: not defined

Use: Determines the finishing value of the z-loop variable. `loopMax` must be defined if `loopInt` is non-zero.

Example: 61

**loopMin**

Value: number

Description: Minimum value of the z-loop variable

Aliases: zmin

System default: not defined

Use: Determines the starting value of the z-loop variable.

Example: 61

**mainLoop**

Value: integer or ‘auto’ or ‘last’ [logical]

Description: Defines the division, if any, between ‘pre-loop’ Phreeqc simulations and ‘main loop’ simulations and, optionally, the oneSimulationAtaTime switch determines whether the main loop simulations should be executed as individual simulations one at a time or all together in one run.

Aliases: loopSimulationStartNumber, simulationStartNumber, simulationStart, start

System default: ‘auto’ [FALSE]

Use: `mainLoop` defines the division between ‘pre-loop’ simulations and ‘main loop’ simulations. The number given defines the first of the main loop Phreeqc simulations numbered from the top of the appropriate block of simulations downwards. Where a data file is used to specify a separate block of simulations for each line of data as in fitting, `mainLoop` is always counted relative to the top of the block not the absolute simulation number, i.e. 1 will always point to the first simulation.

‘auto’ is set by PhreePlot: normally it refers to the last simulation but for calculationType's ‘fit’ and ‘simulate’ it is set to 1. ‘last’ refers to the last simulation.

The optional second parameter is the oneSimulationAtaTime switch
which if set to `TRUE` will run each main loop simulation separately. The default is `FALSE` which means that all the main loop simulations are run in a single call to `Phreeqc`. Running each simulation separately enables tags to be defined and used between simulations.

The `mainLoop` simulation and all subsequent simulations will be ‘looped’ over by the x- and y-axis loops as controlled by the `<x_axis>` and `<y_axis>` tags. These main loop iterations are intended to be run fast, with minimum overheads, trying to avoid the repeating of unnecessary calculations. This distinction also affects what output is stored and in particular eliminates the accumulation of unwanted output data in the ‘out’ file. The ‘out’ file has to be well-formed to go into the plotting or fitting phases successfully.

With the second `oneSimulationAtaTime` parameter set to `FALSE`, as by default, all simulations within the ‘main loop’, i.e. those simulations numbered `mainLoop` and greater, will be executed in one call to `Phreeqc`. Tags are only substituted before entering this block of code and only updated after exiting it. They cannot be updated between these simulations since execution is not returned from `Phreeqc` to `PhreePlot` until all the simulations sent for the run have been executed. If tag values need to be passed from one simulation to another, set `oneSimulationAtaTime` to `TRUE`.

Output is only sent to the ‘out’ file (the main plotting file) from the last simulation or set of simulation to be run. The number of lines sent from the chosen `SELECTED_OUTPUT` block is controlled by `selectedOutputLines`. ‘Pre-loop’ `Phreeqc` simulations are processed one-by-one with tags generated between simulations but no pre-loop output is ever sent to the ‘out’ file.

‘Pre-loop’ simulations are intended to be one-off simulations in which solutions, equilibrium phases, surfaces, reactions, databases etc. are defined and initialised while the main loop is where the `PhreePlot`-style iterations are done.

This strategy gives the main loop advantages in terms of speed in that the overheads are reduced when as many `Phreeqc` simulations are executed in one `Phreeqc` run as possible and when calculations that do not need to be repeated are not. The price paid is that the tags are not updated between the individual simulations making up the main loop (just before and after) and so cannot be used to pass newly-acquired output data from one simulation to a later one. This type of calculation should be done wherever possible in the pre-loop section.

If there are no tags in an input file, the `mainLoop` setting should make no difference to the results although in principle the calculations should be somewhat faster with a setting of 1.

With fits and simulations, the `mainLoop` setting may be set individually for the blocks of simulations associated with each data point. This overrides any setting by `mainLoop` although the value of `oneSimulationAtaTime` is still inherited from `mainLoop`. With fits and simulations, `mainLoop` refers to the position relative to the start of the specified block of simulations for an observation not to its position in the main input file.
**mainLoopColumn**

Value: integer, string  
Description: Defines the column number or name in the 'fit' data file which contains the main loop number  
Aliases:  
System default: 0  
Use: Defines the column number or name in the 'fit' data file which contains the main loop number. The string should refer to a column header in the 'fit' data file.

**mainspecies**

Value: list of strings (maximum length 32 characters each), optionally in quotes  
Description: Determines the 'main species' used in the calculation of predominance and species diagrams  
Aliases: main  
System default: '' (empty string)  
Use: This setting controls the outermost (slowest moving) loop in a predominance diagram (ht1 and grid plots). This is a 'character loop' – the only one – and it uses each of the strings in the list in turn. The setting can be used to specify a list of main 'species', e.g. "Fe", "Mn", "Zn". This can also be input as "Fe, Mn, Zn" or "Fe Mn Zn" or simply Fe Mn Zn without quotes. It also controls the 'element' analysed in a species plot. The list of strings may itself be in quotes with individual elements separated by spaces, tabs or commas, e.g. “Fe,Mn,Zn”.  

The 'main species' is the species for which predominance is to be calculated, e.g. Fe will give a diagram containing only Fe species while Cu would give a diagram with only Cu species. Actually the 'main species' is not a normal chemical species but a component or 'master species' in Phreeqc terminology. It should not include secondary master species such as Fe(+2) or Cu(+2).  

At least one main species must be defined to create a predominance diagram. A list of up to 50 main species can be entered. They will be looped in the order given. Separate by spaces or commas.  

A special main species is 'minerals' which acts as a 'superspecies' - actually more like a 'do nothing' species. If used in conjunction with an appropriate script, e.g. minstab1.inc, it can be used to produce a 'minerals only' or mineral stability plot. This will plot the most abundant mineral no matter what elements it contains.

Examples:  
1, 3, 54
minimumAreaForLabeling

Value: non-negative number
Description: Either defines the minimum percentage of the plot area that is necessary for a field to be labelled with h1 or defines the minimum y-value for a line to be plotted in a species or custom plot.
Aliases: minArea, minimumArea
System default: 0.1
Use: Sometimes some very small fields are found by the hunt and track algorithm either because they actually exist or because they are produced as a result of the numerical error inherent in the numerical method employed in calculating the speciation. This setting can be used to prevent them being labelled (but not from being plotted).

In species plots, there may be many minor species present. These would obscure the plot so this setting can be used to ignore all species for which the maximum value (% species or log concentration) is less than the set value. Neither the line nor the label are plotted in such cases.

Example: 23

minimumYValueForPlotting

Value: number [number]
Description: Defines the minimum y-value for a dataset to be plotted in a species or custom plot. The optional second parameter applies the same test to datasets plotting according to the 2y axis.
Aliases: minY
System default: UNDEFINED UNDEFINED
Use: Points or lines datasets will only be plotted if the maximum value in the dataset is equal to or greater than this value. Each dataset is initially clipped to the plotting domain insofar as it is known at the time.

A ‘dataset’ in this context is a whole column of data as read in from a data file. This dataset may contain line breaks (internally represented by -99998) and so may consist of more than one line (plotted curve). It is not possible to apply this criterion to each separate line in a dataset.

This setting can be used to exclude the plotting of datasets where all the data are close to zero, for example.

A setting of UNDEFINED means that no test is applied.

Example: 75
**missingValue**

<table>
<thead>
<tr>
<th>Value</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A value used in data files to signify a missing data value</td>
</tr>
<tr>
<td>Aliases</td>
<td>miss</td>
</tr>
<tr>
<td>System default</td>
<td>-99999</td>
</tr>
</tbody>
</table>

**Use**

Missing values are substituted as place markers in some data output files where a proper value is not available, e.g. because the speciation has failed so no valid concentration can be written.

PhreePlot also uses an **UNDEFINED** value in other places when it has to print a value without having a value. This is currently set as -99999 and cannot be changed.

**multipageFile**

<table>
<thead>
<tr>
<th>Value</th>
<th>logical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Where more than one plot is produced, determines whether a separate plot file is produced for each file or a single, multipage plot file is produced.</td>
</tr>
<tr>
<td>Aliases</td>
<td>multipage</td>
</tr>
<tr>
<td>System default</td>
<td>F</td>
</tr>
</tbody>
</table>

**Use**

Looping through multiple ‘main species’ produces multiple plots which can either be stored separately one file per plot or as a single multiple page file.

This is also true for multiple predominance plots produced with the `<loop>` variable. However, for custom plots the `<loop>` variable is used to produce multiple curves per plot and so the `<loop>` variable does not trigger the production of a multipage file.

Such files can be viewed with Ghostscript and Adobe Illustrator and can be translated to multiple page pdf files. These are very compact and can be viewed with Adobe Reader.

Set to **TRUE** for a multipage file. It is often best to produce single page files initially as these will be stored as soon as they are produced in a run and can be viewed separately. It is also only possible to follow progress with a tracking plot if this setting is set to **FALSE**. Once the set of plots is complete, a multipage file can be generated quickly by setting `calculation Método` 2 and `multipageFile` to **TRUE** and replotting.

This setting applies to ps, pdf and png files. It may not be possible to view multipage png (mng) files as most png viewers, including GsView, only render the first page. epsi files are intrinsically single page.

For an example of the making of a multipage file see `\demo\multipage-file\multipagefile.ppi`. 
### nameSpeciationProgram

<table>
<thead>
<tr>
<th>Value</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The name of the speciation program being used</td>
</tr>
<tr>
<td>Aliases</td>
<td>program</td>
</tr>
<tr>
<td>System default</td>
<td>Phreeqc</td>
</tr>
<tr>
<td>Use</td>
<td>Since the speciation program used is currently fixed, the default value should not be changed. It is only used in the log file as a matter of record.</td>
</tr>
</tbody>
</table>

### nudge

<table>
<thead>
<tr>
<th>Value</th>
<th>logical switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A switch to automatically create a ‘nudge file’ for adjusting label positions.</td>
</tr>
<tr>
<td>Aliases</td>
<td>FALSE</td>
</tr>
<tr>
<td>System default</td>
<td>FALSE</td>
</tr>
<tr>
<td>Use</td>
<td>If set to TRUE, a ‘do nothing’ template file called ‘nudge_&lt;input_filename&gt;.dat’ will be automatically created with all the labels found in the plot. This can then be edited and used by specifying this file for the nudgeFile setting.</td>
</tr>
</tbody>
</table>

### nudgeFile

<table>
<thead>
<tr>
<th>Value</th>
<th>filename [data_separator]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The name of a nudge file for adjusting the position of labels in a plot</td>
</tr>
<tr>
<td>Aliases</td>
<td>nudgeLabelsFile, nudgeLabels</td>
</tr>
<tr>
<td>System default</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>Use</td>
<td>To adjust the position of labels in a plot (except for a contour plot, q.v., which has its own ways of doing this). There are two possible parameters: (1) the name of the file following normal conventions including search path; (2) optional data separator indicating how the nudge file should be parsed. Default is “\” which specifies any whitespace or multiple commas. The easiest approach is to create a ‘do nothing’ nudge file by setting the nudge switch to TRUE (but not for contour plots). This can then be used as a template for ‘nudging’ label positions. The format of a nudge file is: line 1: a header line (compulsory). Its actual content is optional but would normally consist of the column headings, e.g.</td>
</tr>
</tbody>
</table>

``` `nudgeFile` nudgeFileSwitch
```

``` `nudgeFile` nudgeFileSwitch
```

``` `nudgeFile` nudgeFileSwitch
```

``` `nudgeFile` nudgeFileSwitch
```
plot label type x y angle pos

Leave a blank line if no headings. The headings are for your information only.

line 2 and subsequent: a spreadsheet-like file with the columns consecutively given as:

(1) plot number: plot number for which this line of adjustments will be used. Used 'auto' for all plots.

(2) label species name exactly as given in the labels file, fill color file etc. Don't include <sub> etc tags unless explicitly specified.

(3) type is either 'diff' or 'abs'.

(4) x coordinate or delta x: in the distance units in force, e.g. mm.

(5) y coordinate or delta y: in the distance units in force, e.g. mm

(6) optional (as long as (7) not needed), angle to rotate from the horizontal, measured clockwise (in degrees). Default = 0.0. For example, to rotate the label for the water limits in predominance plots, a positive number must be given. The degrees to rotate are given in the log file.

For 'abs' lines only:

(7) optional, x-position of the 'anchor' point given in (4) above: 0 = beginning of label string (left justified), 1 = centre of string, 2 = end of string (right justified). Default = 1. The x,y-coordinates refer to the position on the baseline of the first line of a multi-line string.

There can be any number of lines in a nudge file. Blank lines and comment lines will be ignored. The speciation calculations do not have to be redone – use calculationMethod 2 or 3 for simply replotting.

The program GSview is useful for getting precise coordinate positions from a ps file. Minor adjustments are most easily made using the 'diff' option to nudge a label by a small amount.

---

**numberOfFitParameters**

Value | non-negative integer
--- | ---
Description | Defines the number of parameters specified in a 'fit' calculation (the number can also be implicitly defined by the length of the various fit parameter lists).

Aliases

System default | 0 (but set to 2 in the distributed pp.set file)
Use | This specifies the number of parameters, each with its own tag, that will be defined and which may be used in the Chemistry section of the input file. These parameters may be fixed or adjustable.

If this setting is used, it should precede all of the other fit parameter lists in the input file since if it has a positive value, it will reset the values of all the parameter lists to their system defaults. There are six such lists (all must have a length of numberOfFitParameters): fitParameterNames, fitLogParameters, fitAdjustableParameters, fitParameterValues, fitLowerParameterValues, and fitUpperParameterValues.
**numericTags**

Value: A list of tag definitions, all on one logical line

Description: Numeric tags can be used to substitute numeric values within the Phreeqc part of the input file, and used in plots.

Aliases: numericTag, numberOfNumericTags

System default: ‘’

Use: The general form for the definition of a tag with the name ‘mytag’, say, is:

\[
\text{<mytag>} = "\text{tag expression}"
\]

where the spaces surrounding the ‘=’ are optional. The tag name is case dependent. The tag expression can itself contain numeric tags providing that they have already been defined. Tags are defined and evaluated in the order of their definitions, effectively ‘top down’. Numeric tag names are case sensitive.

The tag name must not be the same as the name of a fit parameter.

Any number of tag definitions can be included on a line. While the tags and their definitions must all be on a single logical line, it improves legibility if they are split, one definition per physical line, e.g.

\[
\text{numericTags <tag1> = 20} \backslash
\text{<tag2> = "2*<tag1>"}
\]

The tag expression must be one ‘word’ so if it contains spaces, it should be embedded in quotes. Otherwise the quotes are optional. The tag expressions can contain the mathematical functions, \text{abs}, \text{exp}, \text{log10}, \text{log}, \text{sqrt}, \text{sin}, \text{cosh}, \text{tanh}, \text{sin}, \text{cos}, \text{tan}, \text{asin}, \text{acos}, \text{atan}, \text{rand} and \text{nrand}.

For historic reasons, the list of tag definitions may optionally be preceded by an integer giving the number of tag definitions to follow. This option will be removed at some date.

This keyword can be repeated and each instance will be appended to the last rather than replacing it.

**omitAccumulate**

Value: list of strings, each up to 32 characters long

Description: Filters out lines of Phreeqc input if it contains any of the strings (case sensitive).

Use: If any of the strings defined with this keyword is found in the Phreeqc input, then the entire logical line is omitted from input to the Phreeqc processor.
Trailing blanks are not significant. Leading blanks are.
This can be used to omit lines containing the word **undefined** which could have been introduced when a tag to be substituted is **undefined**.

---

### onePass

**Value**  
logical

**Description**  
Used by 'fit' and 'simulate' to determine if the all the values of the dependent variable are calculated in one pass through the **phreeqc** code or not.

**Aliases**  
System default: F

**Use**  
This keyword only has any effect during 'simulate' or 'fit' calculations.

This switch affects how the simulations are run and how the selected output is read.

If **onePass** is **true**, the **phreeqc** code should deliver at least \( n \) lines of selected output (excluding the header) where \( n = \) number of data points (the last \( n \) lines will be picked).

If **onePass** is **false**, the **phreeqc** code should deliver just 1 line of selected output (excluding the header). The code block is iterated \( n \) times to produce the required data.

**onePass** might be appropriate because internal **phreeqc** looping produces multi-line selected output (e.g. the **KINETICS**, **REACTION** or **TRANSPORT** keyword data blocks) or because the **PhreePlot** CHEMISTRY section contains multiple simulations that between them produce the required number of lines of output.

The actual lines picked from the selected output produced by each simulation can be specified with the **selectedOutputLines** keyword. 'auto' will attempt to pick the correct number but if this does not work it should be entered explicitly.

Fitting is usually considerably faster with the **onePass true** setting since this requires fewer calls to **phreeqc** and less overheads. However, this is at the expense of a more complex set up. The **input file pre-processor** may reduce the effort in setting up repetitive parts of the input file.

Also since tag values can only be updated when execution is returned from **phreeqc**, simulations containing tags that need to be updated every iteration must be included in the main loop simulations. It may therefore be necessary to use **mainLoop** (or **mainLoopColumn**) to ensure that the required simulations are included in the main loop and so updated on every iteration.

It is also possible to force **PhreePlot** to run each simulation within a block of simulations used to calculate a data point separately, even with the **onePass** set to **true**. This is done by setting the optional **oneSimulationAtATime** switch of **mainLoop** to **true**.

**Example**  
81
out

Value: logical
Description: Determines if the out output file is created
Aliases: outputFile, output, outFile
System default: T

Use:
This file contains the selected output from the last run, i.e. the last simulation or set of simulations run together. It is the main file for saving data for plotting and is always in a spreadsheet type format. Its precise form depends on the type of plot made.

Output is only sent to the ‘out’ file for main loop simulations (not pre-loop simulations) and when there is more than one main loop simulation and these are executed oneSimulationAtaTime (see mainLoop), output is normally only sent from the very last simulation.

In order to get all of the selected output sent to the ‘out’ file from all simulations, set mainLoop to 1, oneSimulationAtaTime to FALSE, and selectedOutputLines to ‘auto’.

A new ‘out’ file is started (or old file rewound) whenever the main loop value (iz) is 1. ‘species’ calculations also begin a new file on every iteration of the z-loop.

With predominance plots, the ‘out’ file contains the species–value pairs in the number and order specified by the five counts found at the end of the line. This is controlled by the htl.inc file or similar.

With custom and fit plots, the file contains the accumulation of the selected ‘selected output’ with the headings that were sent to the selected output. A blank line separates custom datasets with different loop values. Other methods are available for inserting blank lines (see dataSeparators). The number of lines to be sent to the ‘out’ file is controlled by the type of calculation and the selectedOutputLines keyword.

Example

overlay

Value: list of filenames of PhreePlot-generated Postscript plot files
Description: Plots one of these files on top of the main plot
Aliases: **
System default: **
Use:
This keyword provides the names of plot files to be added to the main plot(s) and so can be useful for combining several PhreePlot plots into one. The list of filenames should match the number of plots generated in the run. If there are not enough files in the list to match the number of plots, the list is recycled. The number of each plot can be seen by using the info block.
This list of files is used, one per ‘page’ (or plot) in turn. This means that each of the individual plots produced can be overwritten with a different plot.

A null string, "", means that no overlay is added to that plot for that position and so can act as an empty placeholder.

This feature assumes a specific format for the ps file(s) to be overlaid. It only applies to ps files generated by PhreePlot in an earlier run. It does not apply to ps files in general or to PhreePlot-generated ps files that have been edited by other software.

The overlay plot(s) will be drawn on top the main plot.

The ps file to be overlaid should be a ‘single page’ file. If a ‘multipage file’ is specified, the overlay will only use the plot from the first page.

The resulting plots are automatically named with the main filename and the suffix, "_overlay.ps", e.g. hfo_Fe1_overlay.ps. The original plot file, e.g. hfo_Fe1.ps, without the overlay is generated as normal.

If other format plot files, such as pdf, are generated these will include the overlay(s).

Although PhreePlot does not support transparency (because Postscript has only limited transparency options), it is possible to produce some pseudo-transparency. For example, the colour ‘nd’ (‘not drawn’) can be useful for fill colours in overlay files since it allows other colours below to be seen. It is the default if no colour background for plots is specified. Note that ‘nd’ (100% transparent) is different from ‘white’ (100% opaque) in this respect.

The overlay option can be useful for adding more information (and complexity) to predominance plots, e.g. the underlying aqueous speciation. An example is shown in the figure below. This was produced in two stages: (i) prepare a ‘bare’ diagram showing just the boundaries for the aqueous speciation (no solid phases present), no axes drawn and a gray colour for the boundaries and labels; (ii) prepare a full diagram with the solid phases, axes, boundaries and labels in black and overlay with (i) above. Tweak the labelling.

An example of this is shown below where the aqueous speciation is overlayed on the main predominance diagram.

An overlay can also be used for ‘watermarking’ your plot.

Unlike most keywords, multiple instances of the ‘overlay’ keyword in input files will not result in overwriting the earlier setting(s) but will enable multiple ps plot files to be overlayed on top of the main ps plot file. Each instance can have its own list of files to be picked one-by-one in sequence. This produces an $n \times m$ matrix of filenames where $n$ is the number of different files to add to a plot and $m$ is the number of the plot produced in a multiplot file. This enables each file in a multiplot file to have multiple, but different, overlay plots for each plot (or page in a multipage file).

For example, if the input file produces four separate plots and the ‘overlay’ keyword is used five times, namely

```
overlay File1.ps
overlay File2.ps File3.ps
overlay File4.ps File5.ps
```
then applying the recycling rules, the four plots will be successively overlaid with the following overlay files:

<table>
<thead>
<tr>
<th>Plot 1</th>
<th>Plot 2</th>
<th>Plot 3</th>
<th>Plot 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>File1.ps</td>
<td>File1.ps</td>
<td>File1.ps</td>
<td>File1.ps</td>
</tr>
<tr>
<td>File2.ps</td>
<td>File3.ps</td>
<td>File2.ps</td>
<td>File3.ps</td>
</tr>
<tr>
<td>File4.ps</td>
<td>File5.ps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>File10.ps</td>
<td>File11.ps</td>
<td>File10.ps</td>
<td></td>
</tr>
</tbody>
</table>

The positioning of the ‘overlaid’ plot on the page will depend on their position in the originating Postscript file, i.e. it will depend on the offsets, axis lengths and orientation defined by the PhreePlot input file used to produce them. By varying the position of individual plots on a page, it is possible to produce complex page layouts with multiple plots.

Example

See \demo\Fe\hfo_with_overlay\hfo-aq.bat and the plot shown below.

---

**pageOrientation**

<table>
<thead>
<tr>
<th>Value</th>
<th>0 or 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines the page orientation of plot files.</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td>0</td>
</tr>
</tbody>
</table>
Use 0 signifies portrait mode while 1 signifies landscape mode.

**paperSize**

**Value** One of the standard paper sizes given by the codes below (case not significant).

**Description** Determines the paper size written to the Postscript file.

**Aliases** paper

**System default** a4

**Use** Sets the paper size. The following paper sizes are available:

<table>
<thead>
<tr>
<th>Code</th>
<th>Dimension (width x height)</th>
</tr>
</thead>
<tbody>
<tr>
<td>11x17</td>
<td>11 x 17 inch</td>
</tr>
<tr>
<td>A0</td>
<td>841 x 1189 mm</td>
</tr>
<tr>
<td>A1</td>
<td>594 x 841 mm</td>
</tr>
<tr>
<td>A2</td>
<td>420 x 594 mm</td>
</tr>
<tr>
<td>A3</td>
<td>297 x 420 mm</td>
</tr>
<tr>
<td>A4</td>
<td>210 x 297 mm</td>
</tr>
<tr>
<td>A5</td>
<td>148 x 210 mm</td>
</tr>
<tr>
<td>B4</td>
<td>250 x 353 mm</td>
</tr>
<tr>
<td>B5</td>
<td>176 x 250 mm</td>
</tr>
<tr>
<td>Ledger</td>
<td>17 x 11 inch</td>
</tr>
<tr>
<td>Letter</td>
<td>8.5 x 11 inch</td>
</tr>
<tr>
<td>Legal</td>
<td>8.5 x 14 inch</td>
</tr>
<tr>
<td>Note</td>
<td>8.5 x 11 inch</td>
</tr>
</tbody>
</table>

**Example**

**pdf**

**Value** logical [logical]

**Description** Determines whether the plot output (if any) is converted to a file in the Adobe Portable Document (pdf) format.

**Aliases** pdfFile

**System default** FALSE

**Use** The first logical switch determines if a pdf file is produced. The second, optional switch determines if the file is ‘linearised’ or not (aka ‘Fast webview’). Both switches are FALSE by default.

A pdf file can only be produced if Ghostscript/GSview is installed and available.
PhreePlot makes use of Ghostscript to produce the pdf file. Pdf files are compact graphics files that can be viewed using Gsview, Adobe Reader and other software. The file created is given the extension pdf. It is also possible to create a pdf file from Gsview directly using its Convert facility. An example of pdf output is given below.

Example

pdfMaker

Value: file path
Description: Path for the Ghostscript executable file used for converting the ps file to other graphical formats including pdf.

Aliases
System default: ""

Use: Ghostscript is not strictly necessary for the operation of PhreePlot since PhreePlot produces native Postscript (.ps) files. However, Ghostscript is necessary for the automatic conversion to other formats such as png or eps.

The Ghostscript executable is named `gswin32c.exe` or `gswin64.exe` (older versions of PhreePlot used `ps2pdf14.bat`). Its location is set in one of three ways:

(i) if Ghostscript has been downloaded during installation, this version of Ghostscript is used by default. This can be found alongside the pp executable in the appropriate Program Files folder. It consists of two files: `gswinxxc.exe` and `gsdllxx.dll` where `xx = 32` or `64`. Replacing these two files with any other version of Ghostscript will cause these to be used;

(ii) if the GSC environment variable has been set to a valid path, then this is used;

(iii) if pdfmaker is not blank and has been set to a valid path, then this will version of Ghostscript be used, e.g. pdfMaker "C:\Program Files\gs\gs9.26\bin\gswin64c.exe".
After normal installation, the **Ghostscript** executable will be installed into the same folder as the **PhreePlot** executable, `pp.exe` or `pp`. For Windows, this will be in the *Program Files* (or *Program Files (x86)*) folder.

The embedded executable usually has a path such as `C:\Program Files\gs\gsx.xx\bin\gswin64c.exe` and is installed along with its associated dynamic link library `gsdll64.dll` (Windows), where `x.xx` is the version number.

The embedded version is usually the latest version of **Ghostscript**. In any case, Version 9.16 or later is required for reliable use.

### IPhreePlotVersion

<table>
<thead>
<tr>
<th>Value</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>PhreePlot version</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td>current version</td>
</tr>
<tr>
<td>Use</td>
<td>Not currently version used.</td>
</tr>
</tbody>
</table>

### Phreeqc.0.out

<table>
<thead>
<tr>
<th>Value</th>
<th>'auto' or a logical (TRUE or FALSE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Switch to determine if the standard Phreeqc.0.out file is written</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td>'auto'</td>
</tr>
<tr>
<td>Use</td>
<td>Explicitly sets the switch that determines if the Phreeqc.0.out file is definitely written (TRUE) or not (FALSE). This will be written on every iteration and can slow down execution times. <strong>FALSE</strong> will cause the file to be deleted on termination if present.</td>
</tr>
</tbody>
</table>

The 'auto' value sets the Phreeqc.0.out switch depending on the debug level, **FALSE** if `ABS(debug) = 0` else **TRUE**. When **TRUE**, the Phreeqc.0.out file will always be created.

'auto' is the default setting.

This file is copied to the *.all file if created and so may be temporarily created even when it is not wanted itself at the end.

### PLOT

<table>
<thead>
<tr>
<th>Value</th>
<th>none (section heading)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Optional section heading for input file</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
</tbody>
</table>
System default
Use None other than to offer chance to structure file

---

**plotFactor**

Value non-negative number
Description Scaling factor for all plot dimensions
Aliases factor
System default 1.0
Use All plot dimensions (titles, axis lengths, line widths, symbol sizes etc) but excluding offset and yoffset are scaled by this factor. This scaling is done just before plotting and so if more than one plotFactor has been specified, only the latest is used. A value of zero will prevent any plotting.
Example 71

---

**plotFrequency**

Value positive integer
Description Frequency of automatically writing the plot.ps file during computations
Aliases plotFreq, plotx
System default 1000000
Use Determines the frequency with which the plot.ps file is automatically written during computations. The file is written every plotFrequency'th point calculated. This file can be used to view the status of the ht1 and grid calculations. A large number means very infrequently; the default effectively means ‘never’. This file can also be forced to be written using the interrupt key (Esc).

---

**plotOrder**

Value list of ‘lines’, ‘lines2y’, ‘points’, ‘points2y’ (or their singular), in any order
Description Controls the general order of plotting lines and points in custom plots
Aliases System default lines lines2y points points2y
Use The order of plotting of points and lines can be important since it controls the over-printing – the last feature plotted will appear on top. The default is for points to appear on top of lines.

Within a given class, say ‘points’, the order of plotting is controlled by the order of definition in the keyword setting.

Although plotOrder does control the order of plotting of separate sets of lines and points, when the same variable is plotted as both a line and a set...
of points, the line is always plotted first. This means that the points will always overprint the line.

---

### plotTitle

<table>
<thead>
<tr>
<th>Value</th>
<th>string (maximum 200 characters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Title at the top of a plot</td>
</tr>
<tr>
<td>Aliases</td>
<td>title</td>
</tr>
<tr>
<td>System default</td>
<td>&quot;&quot; (empty string)</td>
</tr>
<tr>
<td>Use</td>
<td>Gives the title string placed at the top of the plot. This can contain text tags such as <code>&lt;sup&gt;</code>...<code>&lt;/sup&gt;</code>. Its placement is fixed. Use extraText for other placements. A blank string means that an auto-generated title will be used. Turn-off by setting plotTitleColor to 'na', plotTitleSize to 0 or setting the title to a non-printing character such as ‘¬’ (ASCII encoding). A special case is to set plotTitle to 'character set' and this will produce a plot of the current character set (see Appendix 4).</td>
</tr>
<tr>
<td>Example</td>
<td>55</td>
</tr>
</tbody>
</table>

### plotTitleColor

<table>
<thead>
<tr>
<th>Value</th>
<th>Cohort colour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Colour of the plot title.</td>
</tr>
<tr>
<td>Aliases</td>
<td>titleColor</td>
</tr>
<tr>
<td>System default</td>
<td>blue4</td>
</tr>
<tr>
<td>Use</td>
<td>Colour of the plot title. Colours should be chosen from the colour palette.</td>
</tr>
</tbody>
</table>

### plotTitleSize

<table>
<thead>
<tr>
<th>Value</th>
<th>A non-negative number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Size of the plot title</td>
</tr>
<tr>
<td>Aliases</td>
<td>titleSize</td>
</tr>
<tr>
<td>System default</td>
<td>3</td>
</tr>
<tr>
<td>Use</td>
<td>Uses the length units in force at the time of plotting</td>
</tr>
</tbody>
</table>

### png

<table>
<thead>
<tr>
<th>Value</th>
<th>logical [number]</th>
</tr>
</thead>
</table>

Description | Make a png (portable network graphics) copy of the plot
Aliases | pngFile
System default | F
Use | Uses Ghostscript to convert from the ps file. The Ghostscript path for the conversion script is taken from pdfMaker and assumes that the default Ghostscript directory structure given by the installation of a recent release Ghostscript is unchanged.

The second, optional parameter specifies the resolution (in dpi) to use when making the conversion. The default is 300 dpi.

An example of output in png format is given below.

![Example Image]

---

**pointColor, pointColor2y**

Value | list of colours
Description | Symbol colours used for plotting points
Aliases | symbolColor
System default | red4
Use | These define the colour of any points that are to be plotted. Colours should be chosen from the colour palette.

If pointsSameColor is TRUE and a line has been drawn, uses the same colour for the points as for the line. Otherwise the point colour sequence is used.

Whether the colour changes for later datasets depends on the changeColor setting (q.v.). Point colours can be changed by editing the line colour dictionary and forcing the dictionary to be used by setting useLineColorDictionary to a value of 1 or more. The line colour dictionary is automatically written and updated as plotting takes place so it may be necessary to generate the plot first then edit the line colour dictionary and replot.
The colours specified by \texttt{pointColor} are promoted to be the first colours used. If further colours are needed, these are taken successively from the auto-generated list of colours (see Section 7.9).

If \texttt{useLineColorDictionary} is set to a value of 0, the line colour dictionary is not used and the colour sequence for points is either taken from the \texttt{pointColor} setting, or if that list is exhausted, automatically set by \texttt{PhreePlot}.

---

**points, points2y**

<table>
<thead>
<tr>
<th>Value</th>
<th>character list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specifies which columns should be plotted as points on the main y and 2y axes.</td>
</tr>
<tr>
<td>Aliases</td>
<td>plotPoints, plotPoints2y, 2ypoints, point, point2y</td>
</tr>
<tr>
<td>System default</td>
<td>' '</td>
</tr>
<tr>
<td>Use</td>
<td>The list should contain the column names or column numbers of columns for which the points are to be plotted. The names are case dependent. The names or numbers refer to the column of the file being used for plotting, e.g. the 'out' file for custom plots or the 'pts' file for fit plots. The names can contain tags, most usefully character tags. Additional files can be added to the search path using the \texttt{extradat} keyword. These files must be in tabular format with a single header row defining the column names. One of these columns must be the same as the \texttt{customXcolumn} defined elsewhere. Points can be added to predominance diagram plots as well as custom plots (including fit and species plots). The 2y axis is the right-hand y axis which can have a different scale from the left-hand or main y axis.</td>
</tr>
</tbody>
</table>

**Examples**

55, 80

---

**pointsSameColor**

<table>
<thead>
<tr>
<th>Value</th>
<th>logical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines if the colour used for points is the same as that use for the corresponding lines</td>
</tr>
<tr>
<td>Aliases</td>
<td>symbolSameColor, sameColor</td>
</tr>
<tr>
<td>System default</td>
<td>\texttt{F}</td>
</tr>
<tr>
<td>Use</td>
<td>When both lines and points are drawn for a particular column/curve, setting this to \texttt{TRUE} will force the lines and points for a particular column to use the same \texttt{colour}. This will be the colour of the lines irrespective of the \texttt{pointColor} setting (see Section 7.9). This only applies to datasets plotted on the same axis.</td>
</tr>
</tbody>
</table>
**pointSize, pointSize2y**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of non-negative numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the size of points (symbols) for each dataset plotted on the main y (left) and 2y (right) axes for custom plots</td>
</tr>
<tr>
<td>Aliases</td>
<td>symbolSize, symbol</td>
</tr>
<tr>
<td>System default</td>
<td>2</td>
</tr>
</tbody>
</table>

**Use**

Symbols are used to plot points in custom plots (including fit plots) and their size, and that of all other symbols, are controlled by the `pointSize` setting. The actual size of a plotted symbol depends on the length units in force at plotting time and the way that the symbol fills the allotted symbol space.

The `rimFactor` and `rimColor` settings, and their 2y counterparts, control the widths and colours of the rim around each filled circle, respectively. The rim factors are specified as a fraction of the corresponding symbol sizes.

**Example**

55

---

**pointType, pointType2y**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of symbol numbers or names</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Used to define the symbols used in custom plots</td>
</tr>
<tr>
<td>Aliases</td>
<td>symbolType, symbol</td>
</tr>
<tr>
<td>System default</td>
<td>2</td>
</tr>
</tbody>
</table>

**Use**

Symbols are used to plot points in custom plots (including fit plots) and the symbols used are controlled by the `pointType` (main y axis) and `pointType2y` (2y axis) settings.

The y and 2y lists are maintained and used separately.

Symbols can either be specified by their symbol numbers (see Figure 7.5) or by their symbol names (Appendix 3).

**pol**

<table>
<thead>
<tr>
<th>Value</th>
<th>logical [exclude list]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines if a polygon file is created during predominance plot and contour plot calculations, and whether any polygons should be excluded from plotting in predominance diagrams.</td>
</tr>
<tr>
<td>Aliases</td>
<td>polygonFile, polygon</td>
</tr>
<tr>
<td>System default</td>
<td>T</td>
</tr>
</tbody>
</table>
Use

Controls the creation and deletion of the polygon (*.pol) file that is used by *ht1* and *grid*. The polygon file contains the x, y coordinates of field boundaries used to plot the predominance fields. The header also contains the resolution used to generate the file (e.g. x500) and the pe for each point.

Since this file is essential for the operation of *ht1*, *grid* and *contour* plots, the polygon file is automatically created and saved for these plots. This means that the setting of this logical switch is ignored.

The 'exclude list' is a list of polygon/species names (from the list of names appearing in the labels file) that are not to be infilled with colour.

Contours fill areas are automatically named as "1", "2" etc. However, while contour fill areas can be excluded from filling with their ascribed colours, the areas may still be filled with an adjacent colour since the colouring of contour fill areas relies on the overprinting of larger areas by smaller areas. This is why the polygons are coloured strictly in order of decreasing polygon size and why this option is not recommended for contour plots.

An alternative approach to excluding one or more polygons from plotting is to set the species numbers for the excluded polygons to 0 or less in the polygon file and replotting.

The labelling of a polygon (but not the plotting of the polygon) can also be suppressed by using the `minimumAreaForLabeling` criterion or more generally by setting the species number to a negative value in the plot file and replotting without recalculating the labels (`calculationMethod = 2`).

---

**post**

**Value**

List of strings (up to 30 characters each) or a single column name from one of the plotting data files.

If two strings are present and the second one is an integer, then this integer is interpreted as the number of significant figures to use when printing floating point numbers (see below).

**Description**

Can be used to 'post' a numeric value or character string next to each plotted point in custom and fit plots.

**Aliases**

`postName`, `postNames`

**System default**

"" (use default names)

**Use**

This only applies to variables plotted using the `points` or `point2y` keywords.

The behaviour of this setting is somewhat similar to the `labels` setting but it offers the possibility of posting values to individual points (symbols) in custom and fit plots, most commonly using another variable column to provide the posted character string.

Posted values will always be written above and to the right of ('NE') of the associated symbol.

If there is either one post name, or two with the second being an integer, then the first name is tested to see if it corresponds with a column name
in one of the plot data files (the ‘out’ or ‘pts’ file or an _extradat_ file). This test is case-sensitive.

If it does, then posted values for each point are taken from the specified column and the corresponding row of the identified file. Both posted variable and data variable must come from the same file, i.e. be of similar length. If the posted variable exists in more than one file, as would the x column, only the points from the post file are used.

The post column pointed to can be numeric or character.

If more than one post name is specified, then these are assumed to be a list of character strings to post against each point. These are recycled as necessary starting with the first string specified above always starting the cycle for each new dataset.

The interpretation of post strings as _Phreeqc_ formulae is automatically turned off.

If multiple points data sets are defined from the same file as the posted column, each set is posted with the same list of posted values.

Floating point values are reduced to 3 significant figures unless the second (and last) item in the post list is an integer with an absolute value between 1 and 7 whereupon that value is used for the number of significant figures, e.g. if the mass value is 12.764897

- `post mass` will print ‘12.8’
- `post mass 2` will print ‘13’

‘Trailing zeros are removed to save space, even if deemed ‘significant’ by the above definition if the number of significant figures specified above is negative, i.e. between -1 and -7.

‘Large’ and ‘small’ numbers will be printed in E format. In this case, the second item absolute value if present will control the number of figures after the decimal point. For example, 1.2345678E-09 will print ‘1.235E-9’ for a second item value of 3.

The size of the posted text is given by _postSize_. The colour of the text is always black.

Posted text is always printed last and so will overprint any other text.

---

**postSize**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Aliases</th>
<th>System default</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The size of the posted text in the units of length in force at the time of reading.</td>
<td>2</td>
<td>Used with <em>post</em> to post text alongside plotted symbols. The colour of the text is always black and the text is always placed to the top right of the symbol.</td>
<td></td>
</tr>
</tbody>
</table>
**ppa**

Value: logical  
Description: Obsolescent  
Aliases: ppaFile  
System default: F  
Use: No longer used.

**pplog**

Value: logical  
Description: Controls whether a line is written to the pp.log file or not  
Aliases: pplogFile  
System default: T  
Use: The pp.log file keeps a summary of each PhreePlot run. It is especially useful for seeing the results of multiple runs executed from a batch file, such as demo.bat.

This setting should be set to FALSE if several instances of PhreePlot are run simultaneously to minimise interfering interactions.

**printScreenFrequency**

Value: integer  
Description: Frequency of automatically writing a summary of the output to the screen during computations  
Aliases: screenx  
System default: 1  
Use: The absolute value of printScreenFrequency determines the frequency with which summary results are automatically written to the screen during computations. Output is written every abs(printScreenFrequency)th point calculated. This output can be used to view the status of the h1t and grid calculations, and to monitor progress during fitting. A large number means very infrequently.

A value of 0 during fitting means that the value is temporarily set to the number of adjustable parameters. This is because a new direction is only chosen every n'th function evaluation, where n is the number of adjustable parameters.

A negative value turns off the continual updating of the pp.log file that occurs on every iteration of a run.
ps

Value logical
Description Turns on or off the output to the Postscript (ps) file
Aliases psFile
System default T
Use This is the native format for graphical output in PhreePlot. Since several other file formats are derived from the ps file, this file will be produced as an intermediate file if necessary even when this parameter is set to F(ALSE). In such cases, the file is deleted at the end of the run. If set to T(RUE), a copy of the latest ps file is also stored as plot.ps. If ps is set to F(ALSE), then not only will the ps file not be produced but if a ps file of the same name is present, then this will be deleted. The same is true of the plot.ps file.

pts

Value logical
Description Make (and deletes) a points file
Aliases pointsFile
System default F
Use The input value is overridden where a points file is necessary, e.g. in fit, species and ht1 plots.

pxdec

Value integer
Description Determines the number of digits after the decimal point for x-axis numbers
Aliases xaxisDecimalPts
System default auto
Use Can be used to override the value automatically assigned. A value of -1 indicates that it will write a number without a decimal point (i.e. an integer). ‘auto’ supplies a default value (up to 6 decimal points).
### pxmajor

<table>
<thead>
<tr>
<th>Value</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Plot x-axis interval between axis numbers</td>
</tr>
<tr>
<td>Aliases</td>
<td>pxint, xaxisint</td>
</tr>
<tr>
<td>System default</td>
<td>auto</td>
</tr>
<tr>
<td>Use</td>
<td>Defines the separation of the major tick (labelled) interval in plot units on the x axis. 'auto' supplies a default value.</td>
</tr>
<tr>
<td>Example</td>
<td>64</td>
</tr>
</tbody>
</table>

### pxmax

<table>
<thead>
<tr>
<th>Value</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Plot x-axis maximum value</td>
</tr>
<tr>
<td>Aliases</td>
<td>xaxismax</td>
</tr>
<tr>
<td>System default</td>
<td>auto</td>
</tr>
<tr>
<td>Use</td>
<td>Defines the maximum value of the x axis. 'auto' supplies a default value.</td>
</tr>
<tr>
<td>Example</td>
<td>64</td>
</tr>
</tbody>
</table>

### pxmin

<table>
<thead>
<tr>
<th>Value</th>
<th>number [number]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Plot x axis minimum value</td>
</tr>
<tr>
<td>Aliases</td>
<td>xaxismin</td>
</tr>
<tr>
<td>System default</td>
<td>&quot;auto&quot; &quot;undefined&quot;</td>
</tr>
<tr>
<td>Use</td>
<td>Defines the minimum value of the x axis. 'auto' supplies a default value. The optional second parameter (pxminstart) specifies the location of the first major tick mark (and axis numbering). pxminstart should always be greater than or equal to pxmin. The first major tick is at pxmin if pxminstart is not defined or at pxminstart if this is defined. This makes it easier to label the axis with 'pretty' numbers.</td>
</tr>
<tr>
<td>Example</td>
<td>64</td>
</tr>
</tbody>
</table>

### pxminor

<table>
<thead>
<tr>
<th>Value</th>
<th>non-negative number</th>
</tr>
</thead>
</table>
**Description**
Plot x-axis interval between minor (unlabelled) ticks

**Aliases**
xaxisMinorTickInt

**System default**
auto

**Use**
Defines the axis interval of the minor ticks on the x axis. Choosing half the major tick interval is often sensible. ‘auto’ supplies a default value. A value of 0 turns off the minor x-ticks.

**Example**
75

---

**pydec, p2ydec**

**Value**
positive integer

**Description**
Determines the number of digits after the decimal point for y(2y) axis numbers

**Aliases**
yaxisDecimalPts, 2yaxisDecimalPts

**System default**
auto

**Use**
Can be used to override the value automatically assigned. A value of -1 indicates will write a number without a decimal point (i.e. an integer). ‘auto’ supplies a default value (up to 6 decimal points).

The 2y axis is the right-hand y axis which can be separately defined from the main y axis. The 2y axis is used for variables defined with points2y and lines2y.

**Example**
83

---

**pymajor, p2ymajor**

**Value**
number

**Description**
Plot y(2y)-axis interval between axis numbers

**Aliases**
pyInt, yaxisInt, p2yInt, 2yaxisInt

**System default**
auto

**Use**
Defines the separation of the major tick (labelled) interval in plot units on the y axis. ‘auto’ supplies a default value.

The 2y axis is the right-hand y axis which can be separately defined from the main y axis. The 2y axis is used for variables defined with points2y and lines2y.

---

**pymax, p2ymax**

**Value**
number

**Description**
Plot y(2y)-axis maximum value

**Aliases**
yaxisMax, 2yaxisMax
System default: auto

**Use**

Defines the maximum value of the y axis. ‘auto’ supplies a default value.

The 2y axis is the right-hand y axis which can be separately defined from the main y axis. The 2y axis is used for variables defined with `points2y` and `lines2y`.

### pymin, p2ymin

<table>
<thead>
<tr>
<th>Value</th>
<th>number [number]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Plot y(2y)-axis minimum value</td>
</tr>
<tr>
<td><strong>Aliases</strong></td>
<td><code>yaxisMin</code>, <code>2yaxisMin</code></td>
</tr>
<tr>
<td><strong>System default</strong></td>
<td>“auto” “undefined”</td>
</tr>
<tr>
<td><strong>Use</strong></td>
<td>Defines the minimum value of the y axis. ‘auto’ supplies a default value. The optional second parameter (<code>pyminstart</code>, <code>p2yminstart</code>) specifies the location of the first major tick mark (and axis numbering). <code>pyminstart/p2yminstart</code> should always be greater than or equal to <code>pymin/p2ymin</code>. The first major tick is at <code>pymin</code> if <code>pyminstart</code> is not defined or at <code>pymin-start</code> if this is defined. This makes it easier to label the axis with ‘pretty’ numbers. The same applies to the 2y axis. The 2y axis is the right-hand y axis which can be separately defined from the main y axis. The 2y axis is used for variables defined with <code>points2y</code> and <code>lines2y</code>.</td>
</tr>
</tbody>
</table>

### pynminor, p2yminor

<table>
<thead>
<tr>
<th>Value</th>
<th>non-negative number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Plot y(2y)-axis interval between minor (unlabelled) ticks</td>
</tr>
<tr>
<td><strong>Aliases</strong></td>
<td><code>yaxisMinorInt</code>, <code>2yaxisMinorInt</code></td>
</tr>
<tr>
<td><strong>System default</strong></td>
<td>auto</td>
</tr>
<tr>
<td><strong>Use</strong></td>
<td>Defines the axis interval of the minor ticks of the y axis. Choosing half the major tick interval is often reasonable. ‘auto’ supplies a default value. A value of 0 turns off the minor y(2y)-ticks. The 2y axis is the right-hand y axis which can be separately defined from the main y axis. The 2y axis is used for variables defined with <code>points2y</code> and <code>lines2y</code>.</td>
</tr>
</tbody>
</table>

### resolution

<table>
<thead>
<tr>
<th>Value</th>
<th>non-negative integer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Controls the x- and y axis step size used by the hunt and track algorithm and ‘custom’ calculations. When ‘custom’ calculations are made and no</td>
</tr>
</tbody>
</table>
**Aliases**

res, nres

**System default**

1

**Use**

The given x- and y-ranges are divided into a rectangular grid with resolution-1 cells along each axis, i.e. resolution points or nodes on each axis. htl uses a fixed step size (one grid cell) and so resolution is one of the primary determinants which determine the time to make a plot. The speed of the chemical calculations and the length of the boundaries are the others.

The larger the specified resolution, the more detailed the resolution of boundaries but the slower the calculations. Normally a resolution in the range 50-500 is reasonable. Although high resolutions produce more calculation points than lower resolutions, the number of points retained depends on the degree of simplification subsequently used and so will not necessarily lead to significantly larger file sizes.

Resolutions of less than 10 are not allowed by the hunt and track routine. Low resolutions may not be able to resolve certain junctions, particularly close to the domain edges, and may lead to a failure of the htl algorithm to close all the polygons. If this happens, it is not possible to colour the polygons appropriately and so a black and white plot is produced from the vector file. Try increasing the resolution or altering the domain boundaries (xmin, xmax, ymin or ymax) to achieve polygon closure.

Resolution also controls the number of iterations used in custom calculations and plots. If <x_axis> or <y_axis> are present in an input file, then resolution + 1 iterations are made with the following values:

\[
x_{\text{int}} = (\text{xmax}-\text{xmin})/\text{resolution}
\]

for \(i\) in 0:resolution \[x_i = \text{xmin} + i*x_{\text{int}}\]

where xint is the x interval and \(x_i\) is the value of x taken on successive iterations.

e.g. xmin=0, xmax=10, resolution=5 will give values of \(x_i=0, 2, 4, 6, 8, 10\).

If <x_axis> or <y_axis> are not defined, then the interval is set to 0.

Some *Phreeqc* keywords generate their own iterations internally and so produce a SELECTED_OUTPUT file with multiple lines of data suitable for plotting, e.g. the REACTION keyword can do this.

If a full listing of the normal *Phreeqc* output file is wanted, set the Phreeqc.0.out keyword to 'T'. This will write the output from the latest simulation or set all to 'auto' and debug>1. This will then create the *.all file which contains the Phreeqc.0.out output from all of the iterations.

Resolution = 1 will give a single iteration and automatically forces output of the Phreeqc.out file. This is useful for checking *Phreeqc* output. With predominance plot calculations, this will also produce a list of all possible mineral phases in the Phreeqc.out file ready for pasting into the CHEMISTRY section of an input file. A resolution of 1 should normally be used when simple looping calculations are being undertaken and no plot is
produced, i.e. when the `<loop>` tag has been used but the `<x_axis>` and `<y_axis>` tags have not and when `plotFactor` has been set to 0.

resolution = 0 will cause an immediate exit from the calculations and will produce no plot and no error messages.

A contour plot must have a resolution of at least two. Normally a value of 10–100 provides reasonable plots.

Examples 3, 55

---

**restartColorSequence**

<table>
<thead>
<tr>
<th>Value</th>
<th>logical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>When there are multiple plots/datasets per run, determines whether the line color sequence for auto-generated colours is restarted from the beginning of the sequence or not for each plot/dataset.</td>
</tr>
<tr>
<td>Aliases</td>
<td>System default: FALSE</td>
</tr>
<tr>
<td>Use</td>
<td>If the colours for auto-generated colours symbols and lines in multiple plots, or multiple line types within a single plot, need to follow the same sequence, then setting this keyword to TRUE ensures that the colour sequence is started at the beginning of the sequence defined by <code>lineColor</code> or <code>pointColor</code> for each plot. This also applies to whether the sequence should be restarted for any 2y plots or not. The default (FALSE) is to continue the y sequence. Set to TRUE to start 2y sequence with 'red'. If the various plots need lines and symbols to have different colours between plots, then this should be set to FALSE. This will ensure the continuation of the two sequences from where they left off in the previous plot. A more precise determination of colours can be made by editing the line colour dictionary and setting <code>useLineColorDictionary</code> to 1 or 2.</td>
</tr>
</tbody>
</table>

---

**rimColor**

<table>
<thead>
<tr>
<th>Value</th>
<th>list of Cohort colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines the rim colours for point (filled circles) symbols</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td>nd</td>
</tr>
<tr>
<td>Use</td>
<td>Each set of points (or ‘curve’) defined by <code>points</code> can have its own symbol with a separate rim. The colour of the rim is defined by this list and the line width of the rim is given by <code>rimFactor</code>. Colours should be chosen from the colour palette. A rim is only drawn if the symbol itself is drawn.</td>
</tr>
</tbody>
</table>
rimFactor

Value
list of non-negative numbers, normally less than one (fractional sizes)

Description
Determines the line width used for rim colours with point (filled circle) symbols

Aliases

System default
0.08

Use
Each set of points (or ‘curve’) defined by points can have its own symbol. If these are filled circles, they can have a separate rim. The line widths of the rims are defined by this list. The factor given represents the width as a fraction of the symbol size with the rim centered on the circumference of the original filled circle. Normally a value of 0.05 to 0.1 is about right. If the list of rim sizes is shorter than required, the list is recycled.

The rim colour is defined by rimColor. Open circles can be drawn by making the point colour white and the rim colour some other colour. A rim is only drawn if the symbol itself is drawn.

screen

Value
logical [non-negative integer]

Description
Turns on or off all screen output (except fatal errors during reading input files). The optional second argument is the close down time in seconds.

Aliases

System default
T 5

Use
Set to FALSE to prevent any screen output. The system default is TRUE and so any output that is sent before a FALSE has been set will normally be output. This can be disabled by setting the screen setting in pp.set to FALSE.

The optional integer value gives the close down time. This is the number of seconds counted down at the end of a run that has ‘failed’ for some reason. It allows the screen output to be inspected or paused before disappearing from sight. If the Esc key is pressed during closedown, then PhreePlot will stop immediately.

The width of the console window is best set to at least 85 characters wide. This way the scrolling output during a predominance plot will not wrap. The defaults can be set by right-clicking on the top frame of the console window and changing the Defaults.
selectedOutputFile

**Value** logical

**Description** A logical switch which forces the selected output file to be written to a physical file or not.

**Aliases** FALSE

**System default** If this switch is set to TRUE, the selected output file(s) will be written to disk irrespective of the debug setting (normally it is only written for debug > 1). The name of the file is taken from the file name defined by the SELECTED_OUTPUT; -file filename defined in the Phreeqc input file (or its default value) for the particular simulation.

The amount of data sent to a selected output file(s) depends on many factors – the number of simulations executed, their SELECTED_OUTPUT and USER_PUNCH settings, the selectedOutputLines setting and whether all the simulations are executed in one block or one at a time (see mainLoop).

selectedOutputLines

**Value** a non-negative integer or auto

**Description** This defines the number of lines to be copied from the chosen selected output to the ‘out’ file.

**Aliases** selectedOutput

**System default** 1

**Use** This figure gives the number of lines of selected output to be sent to the ‘out’ file and is counted backwards from the last (most recent) line. So the default value of 1 will pick off the last line. Any preceding lines are ignored.

Sometimes more than one line must be sent to the ‘out’ file and selectedOutputLines enables this to be specified. The most important Phreeqc keywords generating such multiline files are REACTION, KINETICS and TRANSPORT.

The value of ‘auto’ means that all lines found in the selected output are sent to the ‘out’ file.

These setting does not affect the number of lines actually written to the selected output – this is controlled by USER_PUNCH and the PRINT; - selected_output switch – but rather it controls the number of such lines written to the ‘out’ file.

Setting a value to zero will turn off copying any selected output. If a value exceeds the number of lines produced, it will copy all the lines actually-produced.

These data will be sent to the ‘out’ file, for plotting or use in fitting if, and only if, these data are part of main loop calculations. Data from any pre-
loop calculations are never sent.

When there is more than one simulation in the main loop and `oneSimulationAtATime` has been chosen (e.g., `mainLoop 1 TRUE`), then selected output is only written to the 'out' file from the last simulation.

The 'auto' option setting is set internally for 'simulate' and 'fit' calculations with the `onePass` option. When `onePass` is `FALSE`, only one value must be exported per full iteration of the Phreeqc code; when `onePass` is `TRUE`, at least `n` lines must be sent where `n` is the number of observations. In this case, if there are more lines than needed only the last `n` are used. It is up to the `selectedOutputLines` settings to ensure that this is the case.

'ht' and 'grid' calculations automatically set `selectedOutputLines` to a value of 1 as this is what is expected by PhreePlot.

Where there are several `SELECTED_OUTPUT/USER_PUNCH` blocks defined in an input file, `selectedOutputLines` only applies to the blocks defined with the largest user number, `n`. This does not have to be the last to be executed though it often is. Giving any `SELECTED_OUTPUT/USER_PUNCH` block a high user number will force the 'out' output to come from this block.

It is often convenient to make sure that no redundant output is sent to the selected output in the first place. This can be done in the Phreeqc code by using

```plaintext
PRINT -selected_output FALSE
```

for all simulations where no output is wanted and then by turning on the output for simulations where it is wanted

```plaintext
PRINT -selected_output TRUE
```

### simplify

<table>
<thead>
<tr>
<th>Value</th>
<th>non-negative number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Controls the degree of simplification of the field boundaries in prominence diagrams</td>
</tr>
<tr>
<td>Aliases</td>
<td>simplificationFactor, simplification</td>
</tr>
<tr>
<td>System default</td>
<td>1</td>
</tr>
<tr>
<td>Use</td>
<td>This is mainly used to straighten 'squiggly' boundaries in ht1 and contour plots. It only applies for <code>calculationMethod 1</code> or <code>3</code> and while plotting 'vectors' (line segments), not whole polygons. In grid plots, the only line simplification that is done is the removal of 'in line' or redundant vertices. It cannot be used to reduce the 'steppiness' of grid plots. The grid stepping is still retained. Line simplification is applied for all <code>simplify</code>'s greater than 0.0.</td>
</tr>
<tr>
<td>For ht1 diagrams and contour plots, a value of <code>simplify</code> of 1 often provides reasonable plots. Larger numbers, say 3, introduce more simplification and may be useful for removing low-angled, jagged boundaries. Smaller values, say 0.1, retain many more points. Somewhere in the range of 0.1 to 10 is usually reasonable.</td>
<td></td>
</tr>
<tr>
<td>Simplification can significantly reduce the size of output files (data and</td>
<td></td>
</tr>
</tbody>
</table>
plot files) and can produce more visually pleasing plots by eliminating unwanted noise. However, for the most accurate plots, it is usually better to reduce the 'steppiness' by increasing the resolution of the calculations rather than by increasing the simplification factor.

The retained points can be viewed by making the track symbol viewable (trackSymbolSize(2) >0 and trackSymbolColor not 'nd'). Setting simplify to 0.0 will show the result without any line simplification.

You can change the simplification factor without recalculating the speciation by using calculationMethod 3 (not 2).

---

**skip**

<table>
<thead>
<tr>
<th>Value</th>
<th>non-negative integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Keep every skips data record when reading in data for fitting or simulation.</td>
</tr>
<tr>
<td>Aliases</td>
<td>numberOfSkipRecords</td>
</tr>
<tr>
<td>System default</td>
<td>1</td>
</tr>
<tr>
<td>Use</td>
<td>Used to select a small subset of the data for more rapid fitting. Useful when exploring the initial estimates of the adjustable parameters. The records to be skipped are calculated from mod(ndatar-nstart,skip)/=0 where ndatar is the number of data records read, nstart is the sequential number of the first record read (normally 1) of the data block and skip is the defined parameter. New data blocks begin after a blank line so the first record after a break is normally included providing the number of records in the block is equal to or greater than skip. Only valid data lines are counted for skipping, i.e. blank and comment lines are filtered out first before counting for skip. To keep all data use skip 1. To keep roughly 1 in 10 use skip 10, etc. A skip value of zero is treated as 1.</td>
</tr>
</tbody>
</table>

---

**SPECIATION**

<table>
<thead>
<tr>
<th>Value</th>
<th>none (section heading)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Section heading only</td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
</tr>
<tr>
<td>System default</td>
<td>Optional; does nothing.</td>
</tr>
<tr>
<td>Use</td>
<td></td>
</tr>
</tbody>
</table>
**speciationProgram**

Value: string  
Description: The name of the speciation program to use  
Aliases: program  
System default: Phreeqc  
Use: The name of the speciation program to use for speciation calculations. Currently only Phreeqc.

**speciationProgramVersion**

Value: string  
Description: Specifies the current version of the speciation program being used  
Aliases: dateProgram  
System default:  
Use: None specified by the program. The program version is now obtained directly from the library being used so this keyword is no longer necessary. Any text given here will be prepended to the rest of the version information. It is only used for printing in log file and info data block.

**startTemperature**

Value: positive number  
Description: The starting 'temperature' for the simulated annealing option for fitting (not implemented)  
Aliases:  
System default: 100  
Use: This is no longer used.

**stopOnFail**

Value: 0, 1 or 2  
Description: Determines whether calculations should continue after a failure in speciation  
Aliases:  
System default: 2  
Use: This keyword controls the behaviour after Phreeqc has failed as indicated
by a non-zero error return.

The default value is 2 which allows PhreePlot to decide whether to continue or not. For most types of calculations, PhreePlot will stop after a non-zero (error) return from Phreeqc but for predominance and contour plots with debug 0, it will continue - effectively mapping the NA area.

To always continue, set to 0. To always stop on an error, set to 1.

### tickColor

<table>
<thead>
<tr>
<th>Value</th>
<th>Cohort colour [colour [colour [colour [colour [colour [colour]]]]]]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specifies the colours of the major and minor axis ticks</td>
</tr>
<tr>
<td>Aliases</td>
<td>tickCol</td>
</tr>
<tr>
<td>System default</td>
<td>auto</td>
</tr>
<tr>
<td>Use</td>
<td>From one to 6 colours need to be specified for the major x-axis, minor x-axis, major y-axis, minor y-axis tick marks, respectively (tickSize is similarly specified, see below). These can all be defined explicitly or implied as follows depending on the number of colours entered. See tickSize for the recycling rules. If a colour is 'auto', it takes on the corresponding axis line colour. The tick marks can also be used to produce a grid over the whole plot area (see tickSize below) but the preferred method is to use gridLines, gridColor etc. The 2y colours are used for the ‘opposite y’ axis if the 2y-axis scale is specified by one of the sets of lines or points, else the tick colours are the same as the ‘normal’ y axis. Colours should be chosen from the colour palette.</td>
</tr>
<tr>
<td>Example</td>
<td>38</td>
</tr>
</tbody>
</table>

### tickSize

<table>
<thead>
<tr>
<th>Value</th>
<th>[number [number [number [number [number [number [inside or outside]]]]]] (but not completely blank)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the length of the major and minor ticks and their placement</td>
</tr>
<tr>
<td>Aliases</td>
<td>ticklength, tick</td>
</tr>
<tr>
<td>System default</td>
<td>2</td>
</tr>
<tr>
<td>Use</td>
<td>From zero to six numbers plus an optional ‘inside’ or ‘outside’. There must be at least one number or inside/outside. Sets the length of the ticks and the placement of the ticks. Six tick lengths are defined, in order: major x-axis, minor x-axis, major y-axis, minor y-axis, major 2y-axis, and minor 2y axis. These can all be defined explicitly or implied as follows depending on the number of numbers entered: one number: defines major x-axis</td>
</tr>
</tbody>
</table>
minor x-axis = 0.5 x major x-axis
major y-axis = major x-axis
minor y-axis = minor x-axis
major 2y-axis = major x-axis
minor 2y-axis = minor x-axis

two numbers: defines major and minor x-axes
major y-axis = major x-axis
minor y-axis = minor x-axis
major 2y-axis = major x-axis
minor 2y-axis = minor x-axis

three numbers: defines major and minor x-axes, and major y-axis
minor y-axis = minor x-axis
major 2y-axis = major y-axis
minor 2y-axis = minor x-axis

four numbers: defines major x-axis and minor x-axes, and major and
minor y-axes
major 2y-axis = major y-axis
minor 2y-axis = minor y-axis

five numbers: defines major x-axis and minor x-axes, and major and
minor y-axes, and major 2y-axis
minor 2y-axis = minor y-axis

six numbers: defines major and minor x-axes, major and minor y-axes,
and major and minor 2y-axes

Ticks are always plotted with full lines not dashed lines. Zero tick size
turns off the tick.

Minor ticks are by default placed half way between the major ticks but
this can be changed with the pxminor, pyminor and p2yminor keywords.

If the tick size is equal to 0.5 or more times the length of the shorter of the
corresponding axis lengths (xaxisLength and yaxisLength), then grid lines
will be plotted. Therefore using an arbitrary very large tick size makes that
tick a grid line. Using large negative values makes a dashed grid line.
However, grid lines are preferably specified with gridLines, gridLineType
and gridDashesPerInch.

If the tick marks are put on the outside, then both ticks and grid lines will
be plotted. In this case, the major ticks will be based on tickSize(1) in
the pp.set file with the minor ticks 0.5 times this size.

The line thickness of the major grid lines is given by axisLineWidth and
the thickness of the minor grid lines is 0.5 x axisLineWidth.

The default is for the tick marks to be placed inside the plotting area. This
can be changed to outside by appending the word 'outside' to the end of
the sequence of numbers entered above. 'inside' forces placement on the
inside.

The 2y tick sizes are used for the 'opposite y' axis if the 2y-axis scale is
specified by one of the sets of lines or points plotted, else the tick sizes are
the same as the 'normal' y axis.

Example

38
trackSymbolColor

Value: Cohort colour
Description: The colour of the tracking symbol.
Aliases: trackCol
System default: red
Use: Used for indicating visited sites in the intermediate plot.ps file produced when calculations are interrupted (‘Esc’) during a predominance plot (see definition of plotFrequency).

Colours should be chosen from the colour palette.

trackSymbolColor is also used for plotting the label anchor in custom plots and for showing the vector vertices in ‘ht1’ plots and contour plots with the contourShiftLabel option ‘n’.

The track symbols can be turned off by setting the colour to ‘nd’ or the size to zero.

trackSymbolSize

Value: non-negative number [non-negative number]
Description: Size of the tracking symbol.
Aliases: tracksize
System default: 1.0 0.0
Use: In custom plots, the track symbol shows the position of the label anchor. This only applies to labels that have been placed in the current plot (calculationMethod 1). In contour plots and with contourShiftLabel set to ‘n’ (for number), the track symbol (1) is drawn at all the vertices to aid deciding by how many vertices to shift the label.

The track symbol is also used in the plot.ps file that is produced by pressing ‘p’ during predominance diagram calculations and shows the progress so far. In ‘grid’ mode, only the perimeter vertices of the visited cells are shown. A double-sized blue or red circle highlights the last calculated point.

The first number, trackSymbolSize(1), gives the size of both of these symbols.

The second optional number, trackSymbolSize(2), either gives (i) the size of the symbols used to show the vertices that have been written to the polygon file for predominance plots, i.e. those vertices that remain after any line simplification, or (ii) the size of the anchor symbol (a filled circle) showing the centre of the plotted labels. These anchor symbols have the colour specified by trackSymbolColor.

In a grid plot, the vertices on the domain boundary will be clipped since the grid is offset by half a cell and so extends beyond the domain bounda-
ries. This means that the boundary vertices are not on the domain boundaries and consequently will not be shown.

**Example**

```plaintext
44
```

---

### trk

**Value** logical

**Description** Determines if the ‘track’ file is retained or not

**Aliases** track, trackFile

**System default** F

**Use**

The track file keeps a record of the summary results of each speciation calculation and records the top three species in terms of molar abundance for the purposes of predominance calculations. This is a more complete version of the summary output that is sent to the screen during a predominance plot calculation.

Output is only sent to the track file for main loop simulations (not pre-loop simulations) and only one line of output is sent per main loop simulation, i.e. if there is more than one main loop simulation and these are executed oneSimulationAtaTime (see mainLoop), output is only sent from the last simulation.

The file also includes various system variables (x, y, pH, pe and temperature depending on the calculationType). If any ‘carry variables’ are specified, these are also appended.

The track file is the primary data file that is used to prepare a ‘grid’ type of predominance plot and is also used to store contour-generated data.

---

### units

**Value** ‘mm’, ‘inch’, ‘pt’

**Description** Units used for all length measurements (mm, inch, pt) that follow its definition.

**Aliases**

**System default** mm

**Use**

Choose between ‘mm’, ‘inch’ or ‘pt’ (one point = 1/72 inch). It is best to decide on the units at the outset and set this in the pp.set file or at the top of an input file.

All settings with a length dimension are read in as pure numbers without dimensions but are immediately converted to the units in force at the time of reading.

Unlike other keywords, the position of this keyword in the input file(s) is important since all length measurements following this keyword will be assumed to be specified using this length scale. This includes keywords, such as extraText, which read some parameters in terms of lengths or sizes from a file.
If a mixture of length units is required then they can be varied within an input file by using this keyword to redefine the units before the change is required.

Internally all dimensions are converted to inches for plotting but the default units in *PhreePlot* and its demo files is 'mm'.

---

### recognisedKeywordIsFatal

<table>
<thead>
<tr>
<th>Value</th>
<th>logical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines if an unrecognised keyword is treated as a fatal error or not.</td>
</tr>
</tbody>
</table>

**System default**: `TRUE`

**Use**

The default (TRUE) is to treat unrecognised keywords as fatal errors in which case they have to be corrected for *PhreePlot* to run.

Alternatively, by setting this switch to `FALSE`, the keyword and its arguments will be ignored. This could be useful in future when new keywords have been added to *PhreePlot* and a script using these keywords is run with an older version of *PhreePlot*.

---

### updateFitParameters

<table>
<thead>
<tr>
<th>Value</th>
<th>logical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Determines whether after a successful optimization the updated parameter values are written back into the relevant input file (TRUE) or not (FALSE).</td>
</tr>
</tbody>
</table>

**System default**: `F`

**Use**

If `updateFitParameters` is set to `TRUE` and if a line containing `fitParameterValues` was present in the original input file, as it normally would be for a fit, then the final fit values will be written back into the input file if the fit has been successful.

The original input file is copied and saved with the template `***.bak.***` where the `***`'s are the original filename and extension providing that this backup file does not already exist.

Clearly it is wise to make an independent backup of all important input files.

---

### useLabelsFile

<table>
<thead>
<tr>
<th>Value</th>
<th>logical</th>
</tr>
</thead>
</table>
| Description | Determines whether an existing labels file is used to position labels in a predominance plot particularly when the fields are being regenerated (calc-
With predominance plots, the default behaviour (FALSE) is for calculationMethod = 1 or 3 to recalculate the positions and orientation of labels each run. This keyword enables the readings in an existing labels file to take precedence and so preserve any editing. If TRUE, the labels file is read but not rewritten.

**useLineColorDictionary**

- **Value**: 0, 1 or 2
- **Description**: Determines whether the line colour dictionary is used to determine the line and points (symbol) colours, and label positions, or not.
- **Aliases**: useColorDictionary, coldict
- **System default**: 0
- **Use**: Increasing numbers indicate increasing dependence on the line colour dictionary, if present.
  
  0 = do not use the line colour dictionary. Use the lineColor setting. If there is more than one line and changeColor is FALSE, use the auto-selected colour sequence defined by PhreePlot and the position of the lineColor variable in the list of variables.
  
  1 = use the line colour dictionary if present but only for colours
  
  2 = use the line colour dictionary if present for both colours and label positions. Labels are only used for lines. In multiplot files, the positions of the labels refer to the last positions recorded for each species. Use post for labelling points.

**Example**: 83

**vec**

- **Value**: logical
- **Description**: Determines if a vector file created during ‘hunt and track’ calculations is retained at the end of computations.
- **Aliases**: vectors, vectorsFile
- **System default**: F
- **Use**: The vectors file will always be created in h1 calculations but this setting will determine if it is deleted at the end of the calculations. It can be regenerated from the points file using the re-process data option (calculationMethod 3).
**weightColumn**

<table>
<thead>
<tr>
<th>Value</th>
<th>non-negative integer or column name (case sensitive)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Column number of the weight variable for the fit method</td>
</tr>
<tr>
<td>Aliases</td>
<td>weightPosition</td>
</tr>
<tr>
<td>System default</td>
<td>0</td>
</tr>
<tr>
<td>Use</td>
<td>Only used in fitting mode. If it is not a positive number, i.e. zero, then the default weighting (unit weighting) is used.</td>
</tr>
<tr>
<td>Example</td>
<td>81</td>
</tr>
</tbody>
</table>

**writeAllInputFiles**

<table>
<thead>
<tr>
<th>Value</th>
<th>logical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Controls whether only the main input files are printed to the log file (T) or all input files (F)</td>
</tr>
<tr>
<td>Aliases</td>
<td>writeInputFiles</td>
</tr>
<tr>
<td>System default</td>
<td>FALSE</td>
</tr>
<tr>
<td>Use</td>
<td>The main input file is the file named on the command line, usually having the ppi extension. Other input files are those ‘included’ in the main input file as well as the override.set file. Sometimes the include files are long and essentially constant and so would extend the log file unnecessarily if always printed. This setting provides the option to switch on or off the printing of these ancillary files. FALSE only echoes the ppi and override.set files; TRUE echoes all the files.</td>
</tr>
</tbody>
</table>

**writePlaceholder**

<table>
<thead>
<tr>
<th>Value</th>
<th>logical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Controls whether a placeholder (a blank line or one containing an UNDEFINED value) is written to the out and trk files when no output is produced by Phreeqc</td>
</tr>
<tr>
<td>Aliases</td>
<td>writePlaceholder</td>
</tr>
<tr>
<td>System default</td>
<td>TRUE</td>
</tr>
<tr>
<td>Use</td>
<td>The default value (TRUE) is is useful when generating contour plots or grid-based predominance diagrams as it maintains the strict order required for plotting from these files even when there has been a failure of Phreeqc to converge. It can also be useful in other calculations to record where a failure has occurred.</td>
</tr>
</tbody>
</table>
However, it may produce unwanted effects, for example, when the input script engineers that a failing calculation is repeated with a different input to produce the wanted output. Here the failing output placeholder would ‘contaminate’ the output files and so should not be written.

Example

See the demo\switch\switch1.ppi example.

---

**xaxisLength**

<table>
<thead>
<tr>
<th>Value</th>
<th>positive number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Length of x axis</td>
</tr>
<tr>
<td>Aliases</td>
<td>xlength</td>
</tr>
<tr>
<td>System default</td>
<td>90</td>
</tr>
</tbody>
</table>

Use

The length is used to determine the length of the x axis in the units in operation (see units).

Example

---

**xmax**

<table>
<thead>
<tr>
<th>Value</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Maximum value of the x-axis variable used during the calculations</td>
</tr>
<tr>
<td>System default</td>
<td>UNDEFINED</td>
</tr>
</tbody>
</table>

Use

Controls the final value for the <x_axis> variable.

Example

3

---

**xmin**

<table>
<thead>
<tr>
<th>Value</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Minimum value of the x-axis variable used during the calculations</td>
</tr>
<tr>
<td>System default</td>
<td>UNDEFINED</td>
</tr>
</tbody>
</table>

Use

Controls the starting value for the <x_axis> variable.

Example

3

---

**xoffset**

<table>
<thead>
<tr>
<th>Value</th>
<th>non-negative number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Distance of the origin of the plot (lower left-hand corner) from the left</td>
</tr>
</tbody>
</table>
Use: Distance of the origin of the plot (lower left-hand corner) from the left side page margin.

Example: 40

---

**xtitle**

<table>
<thead>
<tr>
<th>Value</th>
<th>string ([second string])</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Title of the x axis</td>
</tr>
<tr>
<td>Aliases</td>
<td>'auto' ' '</td>
</tr>
<tr>
<td>System default</td>
<td>'auto' ' '</td>
</tr>
<tr>
<td>Use</td>
<td>The first string is the main x-axis title. The second string is appended to the first string with the 'exponential' scaling factor placed in between. For example if the x-data range from 0 to 1200 then the data could be automatically rescaled by dividing by 1e2. For example, xtitle 'Distance ( /102 m)' gives Distance ( /102 m) or in general, trim(string) // trim(exptext) // trim(second string) where // is the concatenation operator. The title strings can be of any length but their maximum combined length including any text tags and the inserted scale factor is 200 characters. 'auto' gives a blank title for predominance and contour plots. For custom, fit and species plots 'auto' takes the value of customXcolumn label. For a residual sum of squares plot, it takes the title from the x-variable/parameter name. A blank string means that an auto-generated title will be used. Turn-off by setting axisTitleColor to 'na', axisTitleSize to 0 or setting the title to a non-printing character such as '¬' (ASCII encoding).</td>
</tr>
<tr>
<td>Example</td>
<td>66</td>
</tr>
</tbody>
</table>

---

**yaxisLength**

<table>
<thead>
<tr>
<th>Value</th>
<th>non-negative number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Length of x axis</td>
</tr>
<tr>
<td>Aliases</td>
<td>ylength</td>
</tr>
<tr>
<td>System default</td>
<td>90</td>
</tr>
<tr>
<td>Use</td>
<td>The length is used to determine the length of the y axis in the units in</td>
</tr>
</tbody>
</table>
operation (see units).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
<th>Aliases</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ymax</strong></td>
<td>number</td>
<td>Maximum value of the y-axis variable used during the calculations</td>
<td>UNDEFINED</td>
<td>Controls the final value for the (&lt;y_axis&gt;) variable.</td>
</tr>
<tr>
<td><strong>ymin</strong></td>
<td>number</td>
<td>Minimum value of the y-axis variable used during the calculations</td>
<td>UNDEFINED</td>
<td>Controls the starting value for the (&lt;y_axis&gt;) variable.</td>
</tr>
<tr>
<td><strong>yoffset</strong></td>
<td>non-negative number</td>
<td>Distance of the origin of the plot (lower left-hand corner) from the bottom page margin</td>
<td>140</td>
<td>Distance of the origin of the plot (lower left-hand corner) from the bottom page margin.</td>
</tr>
<tr>
<td><strong>yscale</strong></td>
<td>character</td>
<td>Determines the y-scale to use for predominance and contour plots</td>
<td>“native”</td>
<td>The y-scaling during the plotting of predominance and contour diagrams has the following three options:</td>
</tr>
</tbody>
</table>
"native" scale determined by the y-axis variable
"pe" uses the pe scale
"Eh" uses the Eh scale (in V)
"mV" uses the Eh scale (in mV)

If one of the pe-related scales is used, then the calculated pe-related value is used rather than the native y-axis variable.

This setting only applies to predominance and contour plots and is ignored by other plots. It only makes sense if the native y-scale is driven by a variable that directly controls the redox, most commonly \(O_2(g)\) and a precondition for the conversion is that the pe must be known; for the Eh and mV scales, the temperature must also be known. These 'system' variables are always output in predominance plots via the ht*.inc files but these are not mandated in a contour plot. Therefore, in order to make a conversion to one of these pe scales, a column with the heading “pe” and values (-la("e-")) must be PUNCH’ed to the output file, or read from an extradat file. Also for the Eh and mV conversions, a column with heading “rc” and values of temperature in Celsius (rc) must also be PUNCH’ed.

These headings are case sensitive so must be exactly as indicated. Failure to specify the required columns will result in an error. Temperatures within a given plot must be constant for the pe to Eh (mV) conversion to be applied otherwise a warning is given and the first value used.

The calculated pe values, or the conversion parameters, are stored in all of the critical output files. Unknown pe/Eh/mV values are estimated from the known pe values just before plotting. For example, this may be necessary to specify label positions.

The y-scale setting can be changed either before a plot is generated or before replotting. In predominance plots, use calculationMethod = 3 to recalculate the label positions, or for manual adjustment by editing the third column of the labels file, use calculationMethod = 2.

In order to make plots using one of the pe-related scales, it may be necessary to estimate the pe where the speciation has failed. This is not always possible to do reliably and it may therefore not be able to close the various polygons outlining the fields properly. Either reduce the domain to one where the pe can be estimated reliably at all times or drive the yaxis variable with the pe directly using the pe, i.e. use '\(\text{Fix}_e-\)' analogous to '\(\text{Fix}_H^+\)'. If this is done, set the yscale to 'native'.

Example 18

**ytitle, 2ytitle**

<table>
<thead>
<tr>
<th>Value</th>
<th>string [[second string] third string (max. 40 characters)]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Title of the y(2y)-axis, its units and the character attached to label names to indicate that they refer to the 2y axis.</td>
</tr>
<tr>
<td>Aliases</td>
<td>title2y</td>
</tr>
<tr>
<td>System default</td>
<td>'auto' '' '*'</td>
</tr>
<tr>
<td>Use</td>
<td>The title strings can be any length but a maximum of 200 characters including any text tags are used for plotted.</td>
</tr>
</tbody>
</table>
It behaves the same as for `xtitle` except that for predominance plots, if `yscale` is set to “pe” and `ytitle` is ‘auto’, then `ytitle` will be automatically set to “pe”. Similarly for `yscale` and “Eh” or “mV”.

For custom plots with `ytitle` set to ‘auto’, the first label name is used for the y-title. Similarly for `2ytitle`.

The 2y axis is the right-hand y axis which can be separately defined from the main y axis. The 2y axis is used for variables defined with `points2y` and `lines2y`.

See `xtitle` for the meaning of the optional second string.

A blank string means that an auto-generated title will be used. Turn-off by setting `axisTitleColor` to ‘nd’, `axisTitleSize` to 0 or setting the title to a non-printing character such as ‘¬’ (ASCII encoding).

The optional third parameter is a character string that is appended to all label names that refer to variables plotted according to the 2y axis. It is an asterisk by default but it can be set to null (‘’’) if no character is wanted or can include plot tags. This means that `<sup>+</sup>` would be valid.

Examples 3, 66
Examples

These examples are included in the \demo directory and can be run individually or they can all be run with the demo.bat file. The plot filename can be seen at the bottom left-hand corner of each plot.

The examples are arranged in sections based on the type of calculations undertaken. The sections are themselves ordered in terms of the calculationType.

The PhreePlot code that produced each plot is shown below each plot. Note that long lines in the code may wrap to a second line so be careful if copying. If in doubt, refer to the original file in the demo directory.

The examples demonstrate many of the features built into PhreePlot and are intended as templates to be modified for your own particular problem. In most cases, the choice of concentrations etc. are arbitrary and are not intended to have any environmental significance.

The results of geochemical calculations are entirely dependent on the choice of thermodynamic database. Phreeqc makes it easy to select different databases, and to modify them either temporarily or permanently. In some cases, we have mixed databases in order to extend them. This can be dangerous and will almost certainly introduce inconsistencies. We have not made any effort to re-evaluate log K’s in order to achieve internal consistency though in any real application this should be seriously considered.

The other side of the coin is almost equally bad – to accept the status quo and to ignore a significant reaction because it cannot be modelled perfectly can also lead to serious errors – errors of omission rather than commission. If something is important to you, check it out carefully and if necessary, try and improve it. Pass your experiences on to others. Ultimately of course the database you use is your responsibility and you have to be able to defend its use – caveat emptor.
Predominance plots (grid approach)

Predominance plots are a type of two-dimensional plot showing the predominant species as a function of two master variables plotted on the x- and y-axes.

**PhreePlot** uses a full speciation approach (based on **Phreeqc**) to calculate the predominant species. These examples use the ‘grid’ or brute force approach (Section 8.2).
1 Fe-H₂O (grid approach)

This is a predominance diagram for the Fe-H₂O system. Only Fe(OH)₃(a) (also known as hydrous ferric oxide and ferrihydrite) has been allowed to precipitate in this example – the more stable iron oxides such as goethite and hematite would take precedence if they were included in the list of minerals considered. The diagram shows that Fe(OH)₃(a) dissolves under both acidic and reducing conditions.

The grid approach is the simplest way of calculating such predominance diagrams. Just calculate the speciation on a regular grid and plot the results directly, here on a 101 x 101 grid. However, there is much ‘wasted’ effort away from the boundaries and it takes quite a high grid resolution to make a good plot. The resultant plot file size is also quite large especially if the boundaries are not vectorised and simplified. Here the basic pixel map that is produced by the grid approach has been vectorised to give the polygon boundaries. This reduces the resultant file size. However, the boundaries have not been simplified, hence the ‘jaggies’.

It is natural to want to refine the boundaries with a finer grid. This can be done but is expen-
sive – the computational load increases with the square of the resolution. This logically leads to a search for a more efficient approach such as the ‘hunt and track’ approach. However, the grid approach is the most reliable approach (see the caution about the ‘hunt and track’ approach, Section 8.3). Its quality is only limited by the resolution.

The main task of the input file is to define what Phreeqc calculations do and what and how the results are plotted. calculationType defines the type of calculations to be done, here ‘grid’ specifies a predominance diagram using the grid approach. mainspecies defines the ‘species’ or component for which the diagram is to be produced, here Fe. There could be a list of main species in multi-component systems, one for each ‘species’ present (other than H and O).

xmin, xmax, ymin and ymax define the range of the <x_axis> and <y_axis> tags, the domain of the calculations (not necessarily of the plotting - they are controlled by pxmin etc). resolution controls the size of the grid over which speciation calculations will be made, here 101 x 101.

The ‘title’ keywords control the various titles placed on the plot while extraText specifies a file which contains information from which to plot additional, user-supplied text anywhere on the page.

The Phreeqc code starts with the ht1.inc include file which writes data to the selected output ‘file’ in the format expected by PhreePlot for a predominance-type calculation. Note that while this file is called ht1.inc, it works equally well for both grid- and ht1-type calculations.

The rest of the code defines the total concentrations in the system and any minerals or gases that should be considered. Note that the initial pH of the solution (pH 1.8) is lower than the lowest pH of the plot (pH 2). This ensures that Fix_H+ will be able to reach any required pH by the addition of NaOH. If the consequences of this are not wanted then the initial solution would need to have sufficient Na in it to support negative additions of NaOH.

The code is split into two simulations for speed. The first simulation does the initial solution calculation while the second simulation does the individual speciation calculations for each point on the plot. By default, PhreePlot only loops on the final simulation in a multi-simulation file. That is why <x_axis> and <y_axis> are found in the final simulation.

Note that the small, dark blue field (‘FeOH2+’) at pH = 2.8 and log fO2(g) = -22 has not been labelled. This is because it occupies less than 0.1% of the plot area, the default value of minimumAreaForLabeling.
# produces a predominance diagram for the Fe-H2O system using the grid approach

SPECIATION
   JobTitle                             "Fe-H2O"
# uses the 'grid' or brute force approach - slow but more reliable than 'hunt and track'
   calculationType                      "grid"
   calculationMethod                    1
   # diagram is for Fe
   mainSpecies                          Fe
   # pH range
   xmin                                 2.0
   xmax                                 12.0
   # fO2(g) range (controls the redox)
   ymin                                 -90.0
   ymax                                 0.0
   # 101 x 101 grid
   resolution                           101

PLOT
   plotTitle                            "Fe-H<sub>2</sub>O<br>(grid approach)"
   xtitle                               pH
   # this will produce a plot with the native y-scale, fO2(g)
   ytitle                               "log <i>f</i> O<sub>2</sub>(g) (atm)"
   # can use the 'yscale' keyword to plot using pe, mV or Eh scales
   extraText                            extratextFeOH.dat

CHEMISTRY
# standard file for calculating predominant species - from system directory
   include 'ht1.inc'
# same as used by the hunt and track approach

SOLUTION 1
   pH        1.8
   units     mol/kgw
   Fe(3)     1e-2
   Na        1e-1
   Cl        1e-1
   SAVE solution 1
   END

USE solution 1
EQUILIBRIUM_PHASES 1
# N.B. this works because <x_axis> is substituted without any leading spaces
   Fix_H+ --x_axis> NaOH 10
# for negative values of <x_axis> would have to form a new tag to avoid --value
   -force_equality true
   O2(g)  <y_axis> 0.1
# the only mineral allowed to form - must be in the database used
   Fe(OH)3(a)  0 0
# 0 0 means achieve SI=0 and size of initial reservoir is 0 mol Fe(OH)3(a)
   END
# i.e. allow to precipitate but none there to dissolve
2 Cu-S-C (‘island’ found with ‘grid’)

This predominance diagram is for solution-only species; no minerals have been allowed to precipitate.

The diagram has been produced with the ‘grid’ approach. It illustrates an example where the ‘hunt and track’ approach fails to find all fields (see the next Example). The field not found is the Cu+ field in the lower part of the diagram. This ‘island’ is not accessible from any of the domain boundaries and so the hunting part of the ht1 algorithm fails to find it.

Sections through the island at log $f_{O_2}(g) = -63$ atm showing the Cu (Figure Ex2.1) and Cl (Figure Ex2.2) speciation as a function of pH indicate that the island occurs where the only two significant Cu species are Cu+ and CuCl2−. The Cl speciation is also dominated by the CuCl2− species and so the Cl concentration fixes the CuCl2− concentration which in turn fixes the Cu+ concentration.

It appears in practice that such ‘islands’ are not that common. None of the other ‘ht1’ examples in this guide have an ‘island’. Nevertheless, it is always wise to check an ‘ht1’ diagram
using the ‘grid’ approach just to be sure.
Thanks to Hans Meeussen for finding this example.
SPECIATION
    calculationType                      grid
    calculationMethod                    1
    mainSpecies                          Cu
    xmin                                  2
    # upper pH
    xmax                                  10
    ymin                                  -80.0
    ymax                                  0.0
    #
    resolution                            100
PLOT
    plotTitle                            "Cu-S-C-H<sub>2</sub>O<br>(no minerals)"
    xtitle                               pH
    ytitle                               "log <i>f</i>O<sub>2</sub>(g)"
    extraText                            "extratextCuS.dat"

CHEMISTRY
    # standard hunt and track file also works for grid plots
    include 'ht1.inc'

SOLUTION 1
    Temp                                 20
    # set just below lowest pH
    pH                                    1.8
    units                                 mol/kgw
    # total concns
    Cu                                     1e-1
    S(6)                                   1e-1
    # background electrolyte
    Na                                      1e-1
    Cl                                      1.032e-1
    SAVE solution 1
    END
    # loop on last simulation by default
    USE solution 1
EQUILIBRIUM_PHASES 1
    Fix_H+                                 -<x_axis> NaOH
    -force_equality true
    O2(g)                                  <y_axis> 0.1
    # N.B. no minerals
    CO2(g)                                 -3.5      1.0
    END
Predominance plots (ht1 approach)

These examples use the 'ht1' or 'hunt and track' approach (Section 8.3) for calculating predominance diagrams.
3 Fe-H$_2$O (‘Hunt and track’ approach)

This figure is for the same system as in the previous example but has been calculated using the ‘hunt and track’ approach rather than the ‘grid’ approach. This approach seeks out the change in predominant species along the domain boundaries (outside edges) and from these intersections tracks inside, bouncing along the boundary. Ultimately all the internal boundaries have been tracked and linked together to give the required polygons or fields. This approach assumes that there are no ‘islands’ – fields that cannot be reached from the edges – an assumption that is often but not always valid (see Example 43) so care is needed when applying this approach.

This approach is quicker than the grid approach for high quality plots. The effort depends on the length of the boundaries and varies more or less linearly with the resolution (not the square as with the grid approach).

The only change required from the previous example is to change the calculationType from ‘grid’ to ‘ht1’.
# produces a predominance diagram for the Fe-H2O system using the hunt and track approach, uses native y-scale

**SPECIATION**

- **JobTitle**: "Fe-H2O"
- **calculationType**: "ht1"
- **calculationMethod**: 1
- **mainSpecies**: Fe
- **pH range**: x_min = 2.0, x_max = 12.0
- **fO2(g) range**: y_min = -90.0, y_max = 0.0
- **resolution**: 250

**PLOT**

- **plotTitle**: "Fe-H$_2$O (hunt and track approach)"
- **xtitle**: pH
- **ytitle**: "log $f$O$_2$(g) (atm)"

**CHEMISTRY**

- **include**: 'ht1.inc'

**SOLUTION 1**

- **pH**: 1.8
- **Na**: 1e-1
- **Cl**: 1e-1

**USE solution 1**

**EQUILIBRIUM_PHASES 1**

- **Fix H+**: -c_x_axis NaOH 10
- **O2(g)**: y_x_axis 0.1

The only mineral allowed to form - must be in the database used

- **Fe(OH)$_3$(a)**: 0 0

# i.e. allow to precipitate but none there to dissolve
Close-up of a portion of an interesting part of the predominance diagram from the previous figure but this time using the llnl.dat database.

This database file is specified with the database keyword. The database file should either be in the system directory or in the ppi directory or in a directory for which the full file path is specified with the keyword.

The llnl.dat database has a more extensive set of polynuclear Fe species than wateq4f.dat. However, the larger database makes the calculations significantly slower.

Note the appearance of a second very small and unlabelled FeOH2+ field to the right of the main FeOH2+ field.

It is best to recalculate close-ups anew rather than merely replotting them by using a changed pxmin etc.
# closeup of the Fe-H2O system based on the llnl.dat database

SPECIATION

JobTitle                             "Fe-H2O-O2"
calculationType                      ht1
calculationMethod                    1

# species and their names vary with database
database                             llnl.dat
fillColorDictionary                  fillcolorllnl.dat
mainSpecies                          Fe

# pH range (x-axis) from 2-4 in 250 steps
xmin                                 2.0
xmax                                 4.0

# log f(O2(g)) range (y-axis) from -20 to 0 in 250 steps
ymin                                 -30.0
ymax                                 0.0
resolution                           250

PLOT

plotTitle                            "Fe-O\(_2\)-H\(_2\)O \(\text{closeup}\)"
xtitle                               pH
ytitle                               "log <i>f</i> O\(_2\) (g)"
extraText                            "extratextFeOHclose.dat"

CHEMISTRY

# first simulation - initial solution calculation

# special predominance diagram file that adds "(s)" to the names of all minerals
include 'htis.inc'

# this is helpful because in the llnl.dat database, "Fe(OH)3" could be confused
# with an aqueous species

SOLUTION 1

# initial solution pH is less than pHmin
pH                                   1.8
units                                mol/kgw

# total Fe
Fe(3)                                 1e-2

# background electrolyte
Na                                    1e-1
Cl                                     1e-1

SAVE solution 1
END

# second simulation - iterate on this final simulation

USE solution 1

EQUILIBRIUM_PHASES 1

Fix_H+ -<x_axis> NaOH 10
-force_equality true
O2(g) <y_axis>

# note the mineral is Fe(OH)3 not Fe(OH)3(a) in this database
Fe(OH)3 0 0

END
5 Fe-H$_2$O: close-up (stability criterion)

This is the same as in the previous Example but calculated using the ‘stability’ criterion. Note the slightly different diagram with the appearance of an aqueous Fe(OH)$_3$ field.
Examples  287

# closeup of the Fe-H2O system based on the llnl.dat database

SPECIFICATION
  JobTitle                             "Fe-H2O-O2"
calculationType                      ht1
calculationMethod                    1
# species and their names vary with database
database                             llnl.dat
 fillColorDictionary                 fillcolorllnl.dat
 mainSpecies                          Fe
# pH range (x-axis) from 2-4 in 250 steps
 xmin                                 2.0
 xmax                                 4.0
# log f(O2(g)) range (y-axis) from -20 to 0 in 250 steps
 ymin                                 -30.0
 ymax                                 0.0
 resolution                           250

PLOT
 plotTitle                            "Fe-O<sub>2</sub>-H<sub>2</sub>O<br>(closeup)"
xtitle                               pH
 ytitle                               "log <i>f</i> O<sub>2</sub>(g)"
 extraText                            "extratextFeOHclose.dat"

CHEMISTRY
# first simulation - initial solution calculation
# standard stability diagram file
 include 'ht1stability.inc'

SOLUTION 1
# initial solution pH is less than pHmin
 pH        1.8
 units     mol/kgw
# total Fe
 Fe(3)     1e-2
# background electrolyte
 Na        1e-1
 Cl        1e-1
 SAVE solution 1
 END

# second simulation - iterate on this final simulation
 USE solution 1
 EQUILIBRIUM_PHASES 1
   Fix_H+  -<x_axis> NaOH 10
     -force_equality true
   O2(g)  <y_axis>

# note the mineral is Fe(OH)3 not Fe(OH)3(a) in this database
 Fe(OH)3 0 0
 END
6 Fe-H$_2$O (pe scale)

Same as in Example 2 but plotted with the pe scale rather than the O$_2$(g) fugacity scale. This is set with the \texttt{yscale} keyword. \texttt{pymin} and \texttt{pymajor} have also been set to ensure a reasonable y-scale. If \texttt{yscale} is set to "pe", then the default y-axis title is automatically set to "pe".
# produces a predominance diagram for the Fe-H2O system using the hunt and track approach, pe scale
#

SPECIATION
jobTitle "Fe-H2O-O2"
# the 'hunt and track' method
calculationType ht1
# 1=calculate, 2=replot, 3=resimplify and replot
calculationMethod 1
# make a Fe diagram
mainSpecies Fe

# pH range (x-axis) 2-12
xmin 2.0
xmax 12.0

# log f(O2(g)) range (y-axis) -90 to 0
ymin -90.0
ymax 0.0

resolution 250

PLOT
plotTitle "Fe-O<sub>2</sub>-H<sub>2</sub>O<br>(only HFO precipitates: pe scale)"
xtitle pH
# use pe scale - default title is 'pe'
yscale pe
# min y value on plot scale
pymin -15
# pymajor 5 # interval between major ticks and labels on y-scale
# make a pdf file (assumes Ghostscript installed)
pdf T

CHEMISTRY

# first simulation - initial solution calculation only calculated once
include 'ht1.inc'

SOLUTION 1
pH 1.8
units mol/kgw
# total Fe in system
Fe(3) 1e-2
Na 1e-1
Cl 1e-1
SAVE solution 1
END

# second (final) simulation - the final simulation is iterated many times as required by the hunt and track procedure
USE solution 1
EQUILIBRIUM_PHASES 1
Fix_H+ -<x_axis> NaOH 10
-force_equality true
O2(g) <y_axis>

# the only mineral considered
Fe(OH)3(a) 0 0
END
7 Fe-H₂O (Eh scale)

Same as in Example 2 but plotted with an Eh scale rather than the O₂(g) fugacity scale. This makes use of the yscale setting. pymin and pymajor have been set to ensure a reasonable y-scale. If yscale is set to "Eh", then the default y-axis title is set to "Eh (V)". If it is preferred to have the yscale in mV rather than 'V', set yscale to 'mV'.
example 291

# produces a predominance diagram for the Fe-H2O system using the hunt and track approach, Eh (V) scale
#

SPECIATION
jobTitle                             "Fe-H2O-O2"
# the 'hunt and track' method
calculationType                      ht1
# 1=calculate, 2=replot, 3=resimplify and replot
calculationMethod                    1
# make a Fe diagram
mainSpecies                          Fe

# pH range (x-axis) 2-12
xmin                                 2.0
xmax                                 12.0

# log f(O2(g)) range (y-axis) -90 to 0
ymin                                 -90.0
ymax                                 0.0

# searching takes place over a 250 x 250 grid
resolution                           250

PLOT
plotTitle                            "Fe-O<sub>2</sub>-H<sub>2</sub>O<br>(only HFO precipitates; Eh (V) scale)"
xtitle                               pH
# use Eh scale (V) - default title is 'Eh (V)'
ytitle                               Eh
# min y value on plot scale
pymin                                -1
# interval between major ticks and labels on y-scale
pymajor                              0.5
# make a pdf file (assumes Ghostscript installed)
pdf                                   T

CHEMISTRY

# first simulation - initial solution calculation only calculated once
include 'ht1.inc'

SOLUTION 1
pH        1.8
units     mol/kgw
# total Fe in system
Fe(3)     1e-2
Na        1e-1
Cl        1e-1
SAVE solution 1
END

# second (final) simulation - the final simulation is iterated many times as required by the hunt and track procedure
USE solution 1
EQUILIBRIUM_PHASES 1
Fix_H+  -<x_axis> NaOH 10
-<x_axis> H2O 10
-force_equality true
O2(g)  <y_axis>
# the only mineral considered
Fe(OH)3(a) 0 0
END
A classical predominance diagram for arsenic showing the change of arsenic species with both pH and redox. This diagram is only solution for solution species since no minerals have been included in the EQUILIBRIUM_PHASES data block.
SPECIATION
  calculationType                      ht1
  calculationMethod                    1
  mainSpecies                          As
  xmin                                 2.0
  xmax                                 12.0
  ymin                                 -90.0
  ymax                                 0.0
  resolution                           200
  
PLOT
  plotTitle                            "As-H2O-O2-H2O\((\text{dominant species})\)"
  xtitle                               pH
  ytitle                               "log f(O2(g)"
  extraText                            "extratextAs.dat"

CHEMISTRY
  # standard 'hunt and track' file
  include ht1.inc

SOLUTION 1
  temp      20
  units     mol/kgw
  # total As
  As        1e-3
  # background electrolyte
  Na        1e-1
  Cl        1e-1 charge
  SAVE solution 1
  END

USE solution 1

EQUILIBRIUM_PHASES 1
  # <x_axis> is pH so reverse sign
  Fix_H+     -<x_axis> NaOH
              -force_equality true
  O2(g)      <y_axis> 0.1
  END
This diagram has been calculated for the same system as for the previous diagram but this time the sub-dominant species (the second most abundant species) have been plotted. The dominant species is still included in the calculations but it has been demoted so that it is not selected. Note that this often produces six-way intersections as opposed to the three-way intersections of the classical plots.

This type of plot may not be terribly enlightening but if the dominant species is of interest, then the sub-dominant species might also be of some interest. As with the normal predominance diagram, the diagram triggers questions about why particular fields are where they are and this in itself can aid understanding of the processes involved.

It is not possible to calculate this type of diagram with the classical (analytical) approach for calculating predominance diagrams.
# sub-dominant plot (second most abundant species - a bit different from normal!)

**SPECIATION**
- calculationType: ht1
- calculationMethod: 1
- mainSpecies: As
- xmin: 2.0
- xmax: 12.0
- ymin: -90.0
- ymax: 0.0

# need a high resolution to get good straight sloping lines - otherwise use 'simplify 10'
- resolution: 500

# this picks the sub-dominant species
- dominant: F

# three fields are not labelled since they are 'cross-over' fields of other fields and included in a single polygon sequence... and there is one label per polygon sequence

# some two-point segments are sent to the polygon file but are not plotted

**PLOT**
- plotTitle: "As-H$_2$O-O$_2$-H$_2$O (sub-dominant species)"
- xtitle: pH
- ytitle: "log $f$ O$_2$(g)"
- extraText: "extratextAs.dat"

**CHEMISTRY**

# use standard predominance file which returns top three species - PhreePlot deals with this
- include 'ht1.inc'

**SOLUTION 1**
- temp: 20
- units: mol/kgw
- As: 1e-3
- Na: 1e-1
- Cl: 1e-1 charge

**EQUILIBRIUM_PHASES 1**
- Fix H$^+$: $-x_axis$ NaOH
- $-x_axis$ NaOH
- $-force_equality$ true
- O$_2$(g): $y_axis$ 0.1

END
This is the pe-pH diagram for As in a Fe-As-C-S system. This is a complex system with Fe, As, C and S minerals precipitating in various places.

Note that the ppi code (see next page) that produced this diagram actually produces diagrams for all four 'main species' or components (Fe, As, C and S). These are produced in a single ps file since `multipageFile` has been set to true. Looking at all four diagrams together gives a good indication of the interactions involved, and the reasons why the minerals predominate where they do.

For example, realgar is only stable in highly acidic and reducing systems where HS⁻ and H₃AsO₃ activities are relatively high. It does not therefore predominate when pyrite is present since this drops the sulphide activity too low or in oxidising conditions where SO₄²⁻ and As(V) species predominate.
SPECIATION

calculationType          ht1
calculationMethod         1

# produce predominance diagrams for these four elements
mainSpecies              Fe As C S
xmin                      2.0
xmax                      10.0
ymin                      -85.0
ymax                      0.0
resolution                200

PLOT
plotTitle                "Fe-As-C-S<br>(minerals but no sorbed As)"
xtitle                   pH
ytitle                   pe
pymin                    -10.0

# this changes to the pe scale
yscale                   pe
extraText                "extratextFeAsCS.dat"

# put all the plots into a single file - only applies to ps and pdf
multipagefile            t

CHEMISTRY

INCLUDE 'ht1.inc'

SOLUTION 1

temp         25
pH            1.8
units mol/kgw
S(6) 5e-3
Fe 5e-3
As 1e-2
Na 1e-1
Cl 1e-1 charge

SAVE solution 1

END

USE solution 1

EQUILIBRIUM_PHASES 1

Fix_H+ -<x_axis>  NaOH  10
-force_equality true
O2(g) <y_axis>  0.1
CO2(g) -3.5     10

Realgar 0 0
Orpiment 0 0
As2S3(am) 0 0
Halite 0 0
Arsenolite 0 0
Claudetite 0 0
Pyrite 0 0
Mackinawite 0 0
FeS(ppt) 0 0
Sulfur 0 0
Fe(OH)2.7Cl.3 0 0

# Goethite
Pe(OH)3(a) 0 0

# Hematite
Greigite 0 0
Magnetite 0 0
Nahcolite 0 0
Siderite 0 0
Maghemite 0 0
Siderite(d)(3) 0 0
Scorodite 0 0
Natron 0 0
Thermonatrite 0 0
Fe3(OH)8 0 0
<table>
<thead>
<tr>
<th>Mineral</th>
<th>Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thenardite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Melanterite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mirabilite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>As$_2$O$_5$(cr)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Trona</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Jarosite-Na</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>JarositeH</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

END
This the classical pe-pH diagram for As in Fe-As systems – it does not take into account any possible adsorption of As by Hfo which is precipitated above pH 4 and at high pe. Scorodite is only stable below pH 4 and high pe since precipitation of Hfo reduces the Fe$^{3+}$ activity at higher pH values. Reduction of Fe$^{3+}$ to Fe$^{2+}$ reduces Fe$^{3+}$ activities at low pe and so scorodite is not stable there either.
SPECIATION
  # aqueous and minerals
  jobTitle "Fe-As-O2-H2O"
calculationType htl
calculationMethod 1
mainSpecies As
xmin 2.0
xmax 12.0
ymin -85.0
ymax 0.0
resolution 200

PLOT
  plotTitle "Fe-As-O\textsubscript{2}-H\textsubscript{2}O"
xtitle "pH"
  # force the minimum axis y-scale
  pymin -20
  yscale pe
  extraText "extratextscorodite.dat"

CHEMISTRY
  # first simulation

  include 'ht1.inc'

  SOLUTION 1
  # start at a low pH (less than xmin)
  pH 1.8
  units mol/kgw
  Fe(3) 1e-1
  Na 1e-1
  Cl 1e-1
  # total As
  As 1e-2
  SAVE solution 1
END

  # second simulation

  USE solution 1
  EQUILIBRIUM_PHASES 1
  Fix_H\textsubscript{+} -<x_axis> NaOH 10
  -force_equality true
  O2(g) <y_axis> 0.1
  # but no adsorbed As
  Fe(OH)\textsubscript{3}(a) 0 0
  # possible As minerals
  Arsenolite 0 0
  Claudetite 0 0
  Scorodite 0 0
  As2O5(cr) 0 0
END
This diagram is for a similar chemistry to the previous example but uses the Dzombak & Morel (1990) DL model for Hfo to estimate As adsorption by Hfo. The adsorbed As consists of one As(III) and three As(V) species.

The adsorbed species have been included by using the hfo.inc file which links the precipitation of Fe(OH)₃(a) to the Hfo surface.

In working out the predominant species, this example counts and displays each adsorbed species separately. This is how the ht1.inc include file has been coded.
SPECIATION
# aqueous, minerals and surface species
jobTitle                             "Fe-As-O2-H2O"
calculationType                      ht1
calculationMethod                    1
mainSpecies                          As
xmin                                 2.0
xmax                                 12.0
ymin                                 -85.0
ymax                                 0.0
resolution                           200

PLOT
plotTitle                            "Fe-As-O2-H2O<br>(represented by individual adsorbed species)"
xtitle                               pH
pymin                                -20
yscale                               pe
extraText                            "extratextscorodite.dat"

CHEMISTRY
# first simulation

# ht1.inc treats all Hfo-As surface species as separate species for plotting
include 'ht1.inc'

SOLUTION 1
pH        1.8
units     mol/kgw
Fe(3)     1e-1
Na        1e-1
Cl        1e-1
# total As
As        1e-2
SAVE solution 1
END

# second simulation

# add Hfo surface
include 'hfo.inc'

USE solution 1
EQUILIBRIUM_PHASES 1
  Fix_H+ -<x_axis> NaOH 10
  -force_equality true
  O2(g) -<y_axis> 0.1
# only consider this oxide
  Fe(OH)3(a)  0 0

# possible As minerals
  Arsenolite  0 0
  Claudetite  0 0
  Scorodite   0 0
  As2O5(cr)   0 0
END
This is the same system as in the previous example but all the adsorbed As species have been lumped together into a single adsorbed species, Hfo-As. This change is made by using the \texttt{ht1combined.inc} include file rather than \texttt{ht1.inc} file.

The adsorbed species have been included by including the \texttt{hfo.inc} file which links the amount of Hfo surface to the amount of precipitated Fe(OH)$_3$(a), as often occurs in nature.

The Hfo-As field closely follows the combined boundaries of the four adsorbed As species. It is actually very slightly larger than the sum of the four fields because of the larger number of moles of As in the combined Hfo-As field than in individual fields and hence its greater competitiveness against other As species.
SPECIATION
# aqueous, minerals and surface species
jobTitle "Fe-As-O2-H2O"
calculationType   htl1
calculationMethod 1
mainSpecies As
xmin 2.0
xmax 12.0
ymin -85.0
ymax 0.0
resolution 200

PLOT
plotTitle "Fe-As-O2-H2O (represented by a single adsorbed species)"
xtitle pH
pymin -20
yscale pe
extraText "extratextscorodite.dat"

CHEMISTRY

# first simulation

# treat all Hfo-As surface species as one ‘super’ species for plotting
include 'ht1combined.inc’

SOLUTION 1
pH 1.8
units mol/kgw
Fe(3) 1e-1
Na 1e-1
Cl 1e-1
# total As
As 1e-2
SAVE solution 1
END

# second simulation

# add Hfo surface
include 'hfo.inc’

USE solution 1
EQUILIBRIUM_PHASES 1
  Fix_H+ -<x_axis> NaOH 10
  -force_equality true
  O2(g) -<y_axis> 0.1
  # only consider this oxide
  Fe(OH)3(a) 0 0

  # possible As minerals
  Arsenolite 0 0
  Claudetite 0 0
  Scorodite 0 0
  As2O5(cr) 0 0
END
14 Fe-S-As (without sorption)

This example involves Fe-S-As and allows precipitation of the Fe(OH)$_3$(a) which is linked to the Hfo mineral phase. However, there is no code to link arsenic adsorption to this phase, i.e. no 'include hfo.inc' line, or similar. In this example, arsenic minerals only predominate under strongly reducing conditions.

There were a few 'beeps' when running this example with the default convergence parameters. Phreeqc repeatedly failed to converge at a point around pH 9.55 and log $f_{O_2}$(g) -84.7. No selected output was received by PhreePlot and the failed region was recorded as an 'NA' field. The problem was solved by relaxing the convergence tolerance from its default value of 1e-12 to 1e-10. This was done by changing the -convergence_tolerance setting of the KNOBS keyword data block for the second simulation.
Examples 307

# predominance diagram for As in the presence of HFO which adsorbs As according to the Dzombak & Morel DL adsorption model

SPECIATION

calculationType                     ht1
calculationMethod                    1
mainSpecies                          As

# range of pH
xmin                                 2.0
xmax                                 12.0

# range of log fO2(g) to control redox
ymin                                 -90.0
ymax                                 0.0

# controls the resolution (big resolution means small step size)
resolution                           500

PLOT

plotTitle                            "Fe-As-S-H<sub>2</sub>O (HFO DLM)"
xtitle                               pH
ytitle                               "log <i>f </i>O<sub>2</sub>(g)"
extratext                            extratextFeAsS.dat

CHEMISTRY

# simulation 1

# standard predominance-calculating file
include 'ht1.inc'

# initial solution calculation
SOLUTION 1
Temp      20
pH        1.8
units     mol/kgw

# total concns
Fe(3)     3.5e-1
As        1e-2

# sulphide minerals can form but not adsorb
S(6)      1e-2
Na        1e-1
Cl        1e-1

SAVE solution 1
END

include 'hfo.inc'

# simulation 2

USE solution 1
EQUILIBRIUM_PHASES 1
Fix_H+     -<x_axis> NaOH
-force_equality true
O2(g)      <y_axis>

# most likely minerals given the database
As_native                      0 0

# hematite, magnetite removed to make goethite stable
Orpiment                        0 0
Realgar                         0 0
As2S3(am)                       0 0
Pyrite                          0 0
Arsenolite                      0 0
Claudetite                      0 0
Mackinawite                     0 0
FeS(ppt)                        0 0
Sulfur                          0 0
Fe(OH)3(a)                      0 0
Greigite                        0 0
Scordite                        0 0
Mirabilite                      0 0
Melanterite                     0 0
Thenardite 0 0
As2O5(cr) 0 0
Jarosite-Na 0 0
JarositeH 0 0
# Goethite 0 0

END
15  Fe-S-As (low Fe, without surface speciation)

Similar to the previous example but calculated using the htlcombined.inc file which bulks together all As species adsorbed by Hfo into a single species, Hfo-As. This includes both As(V) and As(III) surface species.
# predominance diagram for As in the presence of HFO which adsorbs As according to
# the Dzombak & Morel DL adsorption model

SPECIATION
calculationType                htl1
calculationMethod              1
mainSpecies                    As

# range of pH
xmin                            2.0
xmax                            12.0

# range of log fO2(g) to control redox
ymin                            -90.0
ymax                            0.0

# controls the resolution (big resolution means small step size)
resolution                      500

PLOT
plotTitle                       "Fe-As-S-H<sub>2</sub>O (HFO DLM)"
xtitle                          pH
tytitle                         "log <i>f </i>O<sub>2</sub>(g)"
extratext                       extratextFeAsS.dat

CHEMISTRY

# simulation 1

# standard predominance-calculating file
include 'ht1combined.inc'

# initial solution calculation
SOLUTION 1
Temp      20
pH        1.8
units     mol/kgw

# total concns
Fe(3)     3.5e-1
As        1e-2

# sulphide minerals can form but not adsorb
S(6)      1e-2
Na        1e-1
Cl        1e-1

SAVE solution 1
END

include 'hfo.inc'

# simulation 2
USE solution 1
EQUILIBRIUM_PHASES 1
Fix_H+    -<x_axis> NaOH
-force_equality true
O2(g)     <y_axis>            0.1

# most likely minerals given the database
As_native                      0 0

# hematite, magnetite removed to make goethite stable
Orpiment                        0 0
Realgar                         0 0
As2S3(am)                       0 0
Pyrite                          0 0
Arsenolite                      0 0
Claudetite                      0 0
Mackinawite                     0 0
FeS(ppt)                        0 0
Sulfur                          0 0
Fe(OH)3(a)                      0 0
Greigite                        0 0
Scorodite                       0 0
Mirabilite                      0 0
Melanterite                     0 0
<table>
<thead>
<tr>
<th>Compound</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thenardite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>As2O5(cr)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Jarosite-Na</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>JarositeH</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td># Goethite</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

END
An example of a predominance diagram for As in which the CD-MUSIC model has been used to calculate the adsorption of As(V) and As(III) species on goethite. The CD-MUSIC model parameters were taken from Stachowicz et al. (2006).

This diagram differs from the previous diagram in two main ways: (i) As adsorption has been calculated using the CD-MUSIC model rather than the diffuse layer model; (ii) adsorption is linked to goethite not HFO. Goethite is far less soluble than HFO.

The very insoluble nature of goethite and the strong adsorption of As species to goethite means that adsorbed As predominates throughout most of the domain, more so than for the more soluble HFO. Only under strongly reducing conditions at low pH, and at very high pH, do soluble As species predominate. Reducing conditions and a low pH leads to a marked increase in goethite solubility by reductive dissolution while a high pH leads to a strong electrostatic repulsion of the negatively-charged adsorbed As(V) species and the negatively-charged goethite surface at high pH. This reduces adsorption and so increases the apparent solubility.
# predominance diagram for As in the presence of goethite which adsorbs As \ according to the CD-MUSIC adsorption model

**SPECIFICATION**

- **calculationType**: htl1
- **calculationMethod**: 1
- **mainSpecies**: As

# range of pH
- **xmin**: 2.0
- **xmax**: 12.0

# range of log fO2(g) to control redox
- **ymin**: -90.0
- **ymax**: 0.0

# controls the resolution (big resolution means small step size)
- **resolution**: 101

**PLOT**

- **plotTitle**: "Fe-As-S-H\(_2\)O (CD-MUSIC)"
- **xtitle**: pH
- **ytitle**: "log \(\textit{fO}_2\)(g)"

**CHEMISTRY**

# simulation 1

# standard predominance-calculating file, also sets -high_precision true
include 'ht1.inc'

# CD-MUSIC database, resets convergence criterion, therefore must FOLLOW htl1.inc
include 'cdmusic_hiemstra.dat'

# initial solution calculation
**SOLUTION 1**

- **Temp**: 20
- **pH**: 1.8
- **units**: mol/kgw

# total concns
- **Fe(2)**: 3.5e-1
- **As**: 1e-2

# sulphide minerals can form but not adsorb
- **S(6)**: 1e-2
- **Na**: 1e-1
- **Cl**: 8e-1 charge

**END**

# simulation 2

USE solution 1

**EQUILIBRIUM_PHASES 1**

- **Fix H+**
- **-force_equality true**
- **O2(g)**: 0.1

# most likely minerals given the database
- **As\_native**

# hematite, magnetite removed to make goethite stable
- **Orpiment**
- **Realgar**
- **As2S3(am)**
- **Pyrite**
- **Arsenolite**
- **Claudetite**
- **Mackinawite**
- **FeS(ppt)**
- **Sulfur**
- **Fe(OH)3(a)**
- **Greigite**
Scorodite          0 0
Mirabilite         0 0
Melanterite        0 0
Thenardite         0 0
As2O5(cr)          0 0
Jarosite-Na        0 0
JarositeH          0 0

# common Fe oxide under oxidising conditions
Goethite           0 0

# Magnetite         0 0  # stable below log fO2(g) = -60 \ 
but not in cd-music database (yet) so ignore!!

SURFACE 1
# 3.5 sites/nm2, 98 m2/g, MWt = 88.855 g/mol
  Goe_uniOH1.5  Goethite 0.049886874 8707.79
# 2.7 sites/nm2
  Goe_triOH0.5  Goethite 0.039041901 8707.79
  -cd_music
# C1  C2  (in F/m2)
  -cap  0.85  0.75
  -equil 1

END
17 Fe-As-S (high Fe)

This is similar to Example 13 except that the Fe/As mole ratio is 10:1 rather than 35:1 and the As, Fe and S concentrations are an order of magnitude lower. This means that the region where adsorbed As is a dominant species is much smaller and there is also insufficient SO$_4^{2-}$ to lead to the precipitation of scorodite. However, realgar – an arsenic sulphide – has a small field at low pH and under strongly reducing conditions.
# predominance plot for As including As sorbed by Hfo

**SPECIATION**

calculationType          htl1
calculationMethod        1

# plots for these two species
mainSpecies             As Fe
xmin                    2.0
xmax                    12.0
ymin                    -90.0
ymax                    0.0
resolution              200

**PLOT**

plotTitle              "Fe-As-S-O\textsubscript{2}-H\textsubscript{2}O\textbackslash\textbackslash (with
individual sorbed As species)"
xtitle                  pH
ytitle                 "log \textit{i)f }<i>O<sub>2</sub>(g)"
extraText              "extratextFeAsS2.dat"

**CHEMISTRY**

# first simulation

**SOLUTION 1**

Temp    20
pH      1.8
units   mol/kgw
Fe      1e-2

# total concns
As      1e-3
S(6)    1e-3
Na      1e-1
Cl      1e-1

SAVE solution 1
END

# second simulation

# standard predominance plot file
include 'ht1.inc'

# this includes adsorbed species (calcd according to the D&M DLM)
include 'hfo.inc'

**USE solution 1**

**EQUILIBRIUM_PHASES 1**

Fix H+  -<x_axis> NaOH
-force_equality true
O2(g)    <y_axis> 0.1

As\textit{ native}  0 0
Orpiment       0 0
Realgar        0 0
As\textit{2S3 (am)} 0 0
Pyrite         0 0
Arsenolite     0 0
Claudetite     0 0
Mackinawite    0 0
Fe\textit{3 (ppt)} 0 0
Sulfur         0 0
Fe(OH)\textit{3 (a)} 0 0
Greigite       0 0
Scorodite      0 0
Mirabilite     0 0
Meleramite     0 0
Thenardite     0 0
As\textit{2O5 (cr)} 0 0
Jarosite-Na    0 0
JarositeH      0 0

END
This is a customised predominance plot using the analytical concentrations of elements from a specific groundwater source (shown by the large red filled circle) to construct the predominance diagram. The diagram shows the stability fields for Fe for this water.

It is possible to overlay the plot with symbols designating the pe-pH location of a sequence of sample points, as here, using the extra keyword. This names a file containing the pH-pe coordinates of the groundwater source in question (in red) as well as of other sources in the same aquifer (displayed in blue).

Because the pH of the various sources is near neutral and the sodium concentration in many of the waters is low, it is necessary to add a source of Na in order to achieve the low pHs by subtracting NaOH. This is achieved by adding a nominal source of ‘salt’, NaCl. The code for this is in the PHASES and EQUILIBRIUM_PHASES data blocks. If the interactions of Na or Cl are important in defining the chemistry of interest, perhaps through an ionic strength effect, then non-interacting pseudo-elements can be used instead of Na and Cl (see Section 6.5.5).
# demonstrates how to add sample points to a predominance diagram

**SPECIATION**

jobtitle                        "Fe at Site 10 compared with other groundwater sources"
calculationType                 ht1
calculationMethod               1
mainspecies                     Fe
xmin                            3
xmax                            10
ymin                            -80.0
ymax                            0.0
resolution                      200

**PLOT**

plottitle                       "Site 10"
xtitle                          pH
# tab-delimited
extrasymbolslines               "extrasymbolslincslst.dat" "\t"
extratext                       "extratextmainspecies.dat" "\"
yscale                          pe
pxmin                           3
pxmax                           10
pxmajor                         1
pymin                           -10
pymax                           20
debug 0

**CHEMISTRY**

# standard file for making a predominance plot
include 'ht1.inc'

**PHASES**

Salt
  NaCl = Na+ + Cl-
  log_k 0

# calculates predominance diagram according to these total concns - but only based on the 1 sample is not commented out
# this approach means that it is easy to calculate diagrams for the whole set by removing the comment from one sample at a time

**SOLUTION_SPREAD**

| NumberDescription | temp | pe | Ba | CaCl(0) | F | Fe(4) | Mg | Mn | Na(3) | N(5) | Si(6) | Sr mg/kgw | Mg/kgw | Mg/kgw | Mg/kgw | Mg/kgw | Mg/kgw | Mg/kgw | Mg/kgw | Mg/kgw |
|-------------------|------|----|----|---------|---|-------|----|----|-------|------|-------|----------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| #1Site 111.27.083.630.03691761140.30.2940.673443.45130.029475.30.120.0773.96166 | 0.816 |
| #2Site 27.17.033.380.04222091020.170.2030.853554.5412.40.034793.50.09870.2653.7227 | 0.713 |
| #3Site 310.27.043.020.040818986.20.170.1450.5583456.3611.20.030642.80.03230.2643.382010.573 |
| #4Site 410.17.014.580.009816855.90.110.2630.03383324.74110.007838.30.05122.55337.1680.694 |
| #5Site 510.07.063.920.013618459.50.110.3760.1233533.61210.011448.60.61230.3413.591960.947 |
| #6Site 610.87.181.670.027213246.60.140.5580.4133122.8913.20.06651.90.239-0.054.021421.13 |
| #7Site 79.97.141.480.0169142464.30.170.2250.6433403.2524.60.06853.90.347-0.054.91991.85 |
| #8Site 810.07.033.003892041020.310.210.8993794.4312.30.0406910.9750.2093.47262.0.695 |
| #9Site 910.37.057.630.00717055.52180.10.00253102.71760.006626-0.0074.5322127.0.373 |
| #10Site 1010.07.095.200.008317855.62820.0840.02523102.96570.003926.7-0.0075.233.371310.348 |
Examples 324

#11Site 1110.37.296.470.0091176561.890.0950.00253152.857.370.0005327.50.01464.23
3.341340.39
#12Site 1210.27.13.810.006817371.61.340.1350.02513073.076.880.0009540.40.0194
3.111370.27
#13Site 1311.07.093.500.0123169588.740.0590.00252634.5.973.380.000823.6-0.00716.3
2.91100.27
#14Site 1410.27.221.930.0132176420.210.2720.2093072.688.090.005622.30.0747-0.05
2.741710.679
#15Site 1510.27.183.400.014216845.10.210.3550.1323183.338.480.004729.90.0942-0.05
2.871630.698
#16Site 166.27.093.150.013516055.70.140.154.3462.6215.10.065736.60.1-0.05424
1530.81
#17Site 1711.26.513.900.012416645.20.210.3620.3363173.198.330.008429.30.0911-0.05
2.911590.708
#18Site 1810.47.033.060.015917346.70.220.3980.1543143.489.260.004733.20.106-0.05
3.11670.803
#19Site 199.37.237.530.007818145.40.10.2131.363113.17.040.018419.60.0232-0.05292
1780.538
#20Site 2010.17.076.860.008918344.90.10.2410.1323173.097.390.005920.40.033-0.05
2.911750.602

EQUILIBRIUM_PHASES 1
  Fix_H+ -<x_axis> NaOH 10
    -force_equality  true
  O2(g)  <y_axis> 10
# atmospheric CO2
  CO2(g) -3.5
# this ensures that Na does not run out when fixing low pH’s
  Salt  -12 10 dissolve

# list of the most probable minerals that might form
  Calcite         0 0
  Gypsum          0 0
  Dolomite        0 0
  Siderite        0 0
  Quartz          0 0
  Fe(OH)3(a)      0 0
  Hausmannite     0 0
  Pyrolusite      0 0
  Manganite       0 0
# disordered
  Rhodochrosite(d) 0 0
  Pyrochroite     0 0
  Bixbyite        0 0
# Birnessite      0 0  # more stable than
  Pyrolusite -
  Bixbyite        0 0
  Fluorite        0 0
  Barite          0 0
END
This example shows an example of the Fe-Ni-S system with the low-angled wedges characteristic of predominance diagrams involving sulphide minerals.

The 'steppy' nature of the low-angled boundaries could be reduced either by increasing the resolution from 200 to say 500, or by increasing the simplification factor, say from 1 to 3.

If the resolution is increased then it is necessary to start the calculations from scratch.

If only the simplification factor is changed, then it is not necessary to recalculate the speciation rather change `calculationMethod` to 3 (resimplify and replot) and change `simplify` to 3.
SPECIATION

# this is a larger database, meaning more species, meaning slower
Database  llnl.dat
calculationType  ht1
calculationMethod  1
mainSpecies  "Ni"
xmin  2.0
xmax  12.0
ymin  -75.0
ymax  0.0
resolution  200

PLOT
plotTitle  "Fe-Ni-S<br>(using llnl.dat)"
xtitle  pH
ytitle  "log <i>f </i>O<sub>2</sub>(g)"
pointSize  1.5
# file with additional text to be added to plot
extraText  "extratextFeNiS.dat"

CHEMISTRY

# standard predominance plot file
include 'ht1.inc'

# first simulation - initial solution calculation
PHASES
SOLUTION 1
Temp  80
pH  1.8
units  mol/kgw
density  1
Fe(2)  1e-3
Na  1e-1
# total concns
Ni  1e-3
S(6)  1e-2
Cl  1e-1 charge
SAVE solution 1
END

# second simulation - only repeats this simulation while tracking
USER solution 1
EQUILIBRIUM_PHASES 1
# drives the x-axis
Fix_H+  -<x_axis> NaOH
-force_equality true
# drives the y-axis
O2(g)  <y_axis>  0.1
# list of possible minerals
Polydymite  0 0
Vaesite  0 0
Fyrite  0 0
Millerite  0 0
Heazlewoodite  0 0
Troilit  0 0
Pyrrhotite  0 0
Ni  0 0
S  0 0
Nickelbischofite  0 0
NiCl2:4H2O  0 0
NiCl2:2H2O  0 0
PeO  0 0
Fe(OH)2  0 0
Wustite  0 0
Goethite  0 0
Bunsenite  0 0
Lawrencite  0 0
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ni(OH)₂</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NiCl₂</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fe(OH)₃</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hematite</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Magnetite</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NaFeO₂</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trevorite</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Molysite</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Melanterite</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mirabilite</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Morenosite</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NiSO₄·6H₂O (alpha)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Na</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thenardite</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FeSO₄</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NiSO₄</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Na₂O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Na₃H(SO₄)₂</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jarosite-Na</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fe₂(SO₄)₃</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

END
This is a predominance diagram for the Zn-Fe-C-H\(_2\)O system with adsorption of Zn by Hfo and precipitation of ZnO(a).

\[ \log f_{CO_2(g)} = -3.5 \] subject to the constraint that not more than 1 mole of CO\(_2\)(g) is used. Many minerals, including many more stable forms of iron oxide and siderite (FeCO\(_3\)), have been suppressed for this example. Zn is adsorbed by the Hfo when it is stable. This example uses the 'ht1.inc' include file for sending Phreeqc output to PhreePlot. All Zn species adsorbed by Hfo have been treated as individual species for the purposes of ranking.

If a combined adsorbed Zn field had been wanted, then the 'ht1combined.inc' include file should have been used rather than 'ht1.inc' (see the next Example).
# Zn predominance diagram with Zn adsorbed onto Hfo
# all adsorbed Zn on Hfo are treated as separate species in terms of predominance (mol) counting

SPECIATION
pdf T
calculationType htl
calculationMethod 1
# diagram for Zn
mainSpecies Zn

# pH (x-axis) range 2-10
xmin 2.0
xmax 10.0
# log fO2(g) from -75 to 0
ymin -75.0
ymax 0.0
# jumps about on a 500 x 500 grid when tracking
resolution 500

PLOT
plotTitle "Fe-Zn-C-H\textsubscript{2}O<br>(showing individual sorbed species)"
xtitle pH
ytitle "log f\textsubscript{O2}(g)"
extraText "extratexthfoZnC.dat"

CHEMISTRY

# this standard file calculates the predominant species based on
include 'htl.inc'
# treating all sorbed species as separate species

SOLUTION 1
pH 1.8
units mol/kgw
Fe(3) 1e-1
Zn 1e-3
Na 1e-1
Cl 1e-1
SAVE solution 1
END

USE solution 1
EQUILIBRIUM_PHASES 1
# driven by the x-axis parameters defined above
Fix H\textsuperscript{+} -<x_axis> NaOH 10
-force_equality true
# driven by the y-axis parameters defined above
O2(g) <y_axis>
# fix log PCO2(g) at -3.5 subject to the constraint that
CO2(g) -3.5 1
# a maximum of 1 mol CO2 is supplied

# list of possible minerals (many are commented out)
Pe(OH)\textsubscript{3}(a) 0 0
#Magnetite 0 0
#Hematite 0 0 # this is likely to be the stable Fe(III)-oxide mineral
#Goethite 0 0
#Fe(OH)\textsubscript{2}Cl\textsubscript{0.7} 0 0
#Fe\textsubscript{3}(OH)\textsubscript{8} 0 0
#Siderite 0 0
# Maghemite 0 0
# Siderite(d)(3) 0 0
Trona 0 0
Natron 0 0
Thermonatrite 0 0
Nahcolite 0 0
# Zincite(c) 0 0
ZnO(a) 0 0
# ZnO3:H2O 0 0
# Zn(OH)2-e 0 0
Smithsonite 0 0
# Zn(OH)2-g 0 0
# Zn(OH)2-b 0 0
# Zn(OH)2-c 0 0
# Zn(OH)2-a 0 0
# Halite 0 0
Zn2(OH)3Cl 0 0
ZnS(OH)8Cl2 0 0
ZnCl2 0 0
ZnMetal 0 0

SURFACE 1
# Dzombak & Morel (1990) Hfo database
Hfo_sOH Fe(OH)3(a) equilibrium_phase 0.005 53300
Hfo_wOH Fe(OH)3(a) equilibrium_phase 0.2
END
This is the same as the previous example except that all adsorbed Zn species have been combined into a single composite species, Hfo-Zn, for the purposes of ranking. This makes the diagram somewhat more straightforward in appearance and avoids the division into individual surface species which in many cases is poorly constrained and in any case may only be of interest to the surface chemist. This approach uses the ‘ht1combined.inc’ include file. The overall area of predominance of the adsorbed species is very similar.

The longer BASIC script required for this approach makes the calculations significantly slower per iteration (some 20% slower), but because the total length of the boundaries is less in the composite approach, the overall computation time is actually slightly faster.
Examples 335

# Zn predominance diagram with Zn adsorbed onto Hfo
# all adsorbed Zn on Hfo are treated as separate species in terms of predominance (mol) counting

SPECIATION
pdf T
calculationType h1
calculationMethod 1
# diagram for Zn
mainSpecies Zn

# pH (x-axis) range 2-10
xmin 2.0
xmax 10.0

# log fO2(g) from -75 to 0
ymin -75.0
ymax 0.0

# jumps about on a 500 x 500 grid when tracking
resolution 500

PLOT
plotTitle "Fe-Zn-C-H<sub>2</sub>O<br>(showing individual sorbed species)"
xtitle pH
ytitle "log <i>f</i> O<sub>2</sub>(g)"
extraText "extratexthfoZnC.dat"

CHEMISTRY
# this standard file calculates the predominant species based on
include 'ht1.inc'
# treating all sorbed species as separate species

SOLUTION 1
pH 1.8
units mol/kgw
Fe(3) 1e-1
Zn 1e-3
Na 1e-1
Cl 1e-1
SAVE solution 1
END

USE solution 1
EQUILIBRIUM_PHASES 1
# driven by the x-axis parameters defined above
Fix_H+ -<x_axis> NaOH 10
-force_equality true
# driven by the y-axis parameters defined above
O2(g) <y_axis>
# fix log PCO2(g) at -3.5 subject to the constraint that
CO2(g) -3.5 1
# a maximum of 1 mol CO2 is supplied

# list of possible minerals (many are commented out)
Fe(OH)3(a) 0 0
#Magnetite 0 0
#Hematite 0 0 # this is likely to be the stable Fe(III)-oxide mineral
#Goethite 0 0
#Fe(OH)2.7Cl.3 0 0
#Fe3(OH)8 0 0
#Siderite 0 0
Examples 336

#Maghemite 0 0
#Siderite\{d\}(3) 0 0
    Trona 0 0
    Natron 0 0
    Thermonatrite 0 0
    Nahcolite 0 0
#Zincite\{c\} 0 0
    ZnO(a) 0 0
    ZnCO_3\cdot H_2O 0 0
#Zn(OH)\{2\}-e 0 0
    Smithsonite 0 0
#Zn(OH)\{2\}-g 0 0
#Zn(OH)\{2\}-b 0 0
#Zn(OH)\{2\}-c 0 0
#Zn(OH)\{2\}-a 0 0
#Halite 0 0
    Zn_2(OH)_3Cl 0 0
    Zn_5(OH)_8Cl_2 0 0
    ZnCl_2 0 0
    ZnMetal 0 0

SURFACE 1
# Dzombak & Morel (1990) Hfo database
    Hfo_{sOH} Fe(OH)_3(a) equilibrium_phase 0.005 53300
    Hfo_{wOH} Fe(OH)_3(a) equilibrium_phase 0.2

END
22  Ca-Fe-Na-X-HFO (adsorption and ion exchange)

Shows how Fe, mostly as Fe$^{2+}$, indirectly affects the dominant Ca speciation through Ca$^{2+}$-Fe$^{2+}$ competition on the cation exchanger and on Hfo surface sites.

In principle, by combining one or more ion exchangers, some adsorbed species and some mineral species, it is possible to simulate the active species in many soils and sediments.
# produces a set of predominance diagrams for Ca using the &lt;loop&gt; variable \
 to systematically vary the total Fe in the system 
# system contains a reduced Fe mineral
# Ca is present in solution, on an ion exchanger (e.g., clay), sorbed by Fe(OH)3(a) \
 and potentially in a mineral (calcite)

SPECIATION
jobTitle                             "Fe-Ca-H2O redox"
# produce a predominance diagram using the hunt and track approach
calculationType                      ht1
calculationMethod                    1
mainSpecies                          Ca   Fe # diagram for Ca
# pH range adjusted through the &lt;x_axis&gt; variable
xmin                                 2.0
xmax                                 10.0
# f(O2(g)) range adjusted through the &lt;y_axis&gt; variable
ymin                                 -80.0
ymax                                 0.0
# low resolution - jumps around on a 50 x 50 grid
resolution                           50

# min, max and step size for FeT (the z-loop variable)
loopMin                              0.0
loopMax                              0.1
loopInt                              5.0E-02

PLOT
plotTitle                            "Ca-Fe-Na with sorption and ion exchange\n(includes Fe(OH)2(s) from llnl.dat)"
xtitle                               pH
ytitle                               "log <i>f</i> O<sub>2</sub>(g)"
pxmax                                10.0
# additional text on the plot
extraText                            "extratextFeOHCa.dat"
multipageFile                        F # each plot in a separate file

CHEMISTRY
include 'ht1.inc' # standard predominance calculating file

PHASES # temporarily add this to the database
Fe(OH)2(a)  # from llnl.dat (NB approximate only - not checked for consistency)
Fe(OH)2 + 2.0000 H+  =  + 1.0000 Fe++ + 2.0000 H2O
log_k           13.9045
-delta_H  -95.4089 kJ/mol    # Calculated enthalpy of reaction
-analytic -8.6666e+001 -1.8440e-002 7.5723e+003 3.2597e+001 1.1818e+002

SOLUTION 1
temp      25
pH        1.5
units     mol/kgw
Fe(3)     &lt;loop&gt; # FeT is controlled by the &lt;loop&gt; variable
Na        1e-2
Cl        1e-2 charge
Ca        1e-3 # CaT

EXCHANGE
-equilibrate 1
X 0.02 # 0.02 equiv of an exchanger

EQUILIBRIUM_PHASES 1
Fix_H+    -&lt;x_axis&gt; NaN # x-axis is pH
-force_equality true
O2(g)    &lt;y_axis&gt; # y-axis is log fO2(g)
CO2(g)   -2 0.3 # soil PCO2 - max CO2 supplied is 0.3 mol
Fe(OH)$_3$(a)  0 0  # possible minerals
Fe(OH)$_2$(a)  0 0  # ... added to database above
Calcite  0 0

SURFACE 1
  Hfo_sOH Fe(OH)$_3$(a)  equilibrium_phase 0.005  53300
  Hfo_wOH Fe(OH)$_3$(a)  equilibrium_phase 0.2
END
23  Acid mine drainage

This example shows the results of a predominance calculation starting with a water with a complex composition – it starts with some acid mine drainage (AMD). It is assumed that there is adsorption of As by HFO (hydrous ferric oxide) and HAO (hydrous aluminium oxide) and equilibrium with CO$_2$(g) subject to a constraint on the total amount of CO$_2$(g) used (simulating poor pathways for gas migration).

The \textit{wateq4f.dat} database is used for most of the thermodynamic data. There is no established database for metal adsorption by HAO and so we have adjusted the HFO database for a higher pzc and reduced specific surface area. This is only very approximate but is included to show the possible impact of HAO. In this case, HAO becomes the dominant As species under reducing conditions where HFO is no longer stable. An optimised HAO database is required before taking the results of these calculations any further. This example is slow to calculate because of the complexity of the calculations – there are many mineral and adsorbed species.

This diagram is just for As but a better appreciation of the reactions can be achieved by viewing the diagrams for other elements. The following example is a diagram for C calculated for the same overall conditions as for As above.
SPECIATION
  jobTitle                             "WJ AMD"
  Database                             "wateq4fhao.dat"
  calculationType                      ht1
  calculationMethod                    1
  mainSpecies                          As
  xmin                                 2.0
  xmax                                 10.0
  ymin                                 -80.0
  ymax                                 0.0
  resolution                           200

PLOT
  plotTitle                            "Wheal Jane AMD"
  xtitle                               pH
  yscale                               pe
  pymin                                -10
  lineWidth                            0.1
  minimumAreaForLabeling               1
  extraText                            "extratextwj.dat"

CHEMISTRY
  include 'ht1.inc'

PRINT
  -reset false

PHASES
  Hydrozincite
  Zn5(OH)6(CO3)2 + 10H+ = 5Zn+2 + 2CO2 + 8H2O
  #9.15
  log_k 45.75
  #Preis & Gamsjager 2001
  -delta_H  -256.5 kJ

  # complex water chemistry
  SOLUTION 1  WJ1
  temp      21.6
  pH        3.5
  #100mV from other sample
  pe        1.69
  # redox
  units     mg/L
  density   1
  # Alkalinity 12.7 as HCO3-  #probably Al
  C    10  as H2CO3 CO2(g) -2
  Cl   179
  F    44
  #reduction of this produces a lot of OH-  1390
  S(6) 1390  as SO4
  Ca   191
  Mg   43
  Na   93
  K    12
  #25 really
  Al   25
  Si   11.0
  Sr   1.87
  Ba   0.052
  Li   2.7
  Fe   346
  Mn   19.7
  As   2.1
  Zn   125
  # kg
  -water  1
  SAVE solution 1
END
USE solution 1
EQUILIBRIUM_PHASES 1
   Fix_H+ -<x_axis> NaOH 10
   -force_equality true
   O2(g) <y_axis> 0.1
# gives CH4 and can reduce to native As
   CO2(g) -3.5 0.01
# maintains Na in the system for functioning of Fix_H+
   Halite -6.34 10

Al(OH)3(a) 0 0
As_native 0 0
Ba3(AsO4)2 0 0
Barite 0 0
Calcite 0 0
Dolomite 0 0
Fe(OH)3(a) 0 0
Fluorite 0 0
Halloysite 0 0
Hausmannite 0 0
Hydrozincite 0 0
Jarosite(ss) 0 0
Jarosite-H 0 0
Jarosite-Na 0 0
Manganite 0 0
Orpiment 0 0
Pyrite 0 0
Pyrochroite 0 0
Pyroclusite 0 0
Realgar 0 0
Rhodochrosite 0 0
Siderite 0 0
Sphalerite 0 0
Strontianite 0 0
ZnO(a) 0 0

SURFACE 1
# standard Hfo of D&M
   Hfo_sOH Fe(OH)3(a) equilibrium_phase 0.005 53300
   Hfo_wOH Fe(OH)3(a) equilibrium_phase 0.2

# glorified guesswork only
   Hao_sOH Al(OH)3(a) equilibrium_phase 0.005 7800
   Hao_wOH Al(OH)3(a) equilibrium_phase 0.2
END
This diagram has been calculated for the same conditions as in the previous example but is for C rather than As. This has been done by changing the value of mainspecies from As to C.

The computations are slow because of the large number of mineral boundaries and possible minerals.

There are many fields close together at high pH which makes labelling difficult. Some of the species have several fields, e.g. HCO₃⁻ and the fields are small. Therefore the number of labels plotted has been reduced by changing the minimumAreaForLabeling from 0.1% (the default set in pp.set) to 1%.
SPECIATION

jobTitle "WJ AMD"

# includes Hao surface definitions
Database "wateq4fhao.dat"
calculationType htl
calculationMethod 1
mainSpecies C
xmin 2.0
xmax 10.0

# use PO2(g) to control redox
ymin -80.0
ymax 0.0
resolution 250

PLOT

plotTitle "Wheal Jane AMD"

# use pe scale
xtitle pH
yscale pe

# minimum pe on plot
pymin -10

LineWidth 0.1

# don’t label small fields (diagram v complex)
minimumAreaForLabeling 1
extraText "extratextwj.dat"

CHEMISTRY

include 'ht1.inc'

PRINT

-reset false

PHASES

# a possibility
Hydrozincite

Zn5(OH)6(CO3)2 + 10H+ = 5Zn2+ + 2CO2 + 8H2O

#9.15
log_k 45.75

# Preis & Gamsjager 2001
-delta_H -256.5 kJ

SOLUTION 1 WJ1

temp 21.6
pH 3.5

# 100 mV from other sample
pe 1.69

# redox pe
units mg/L
density 1

# Alkalinity 12.7 as HCO3-
C 10 as H2CO3 CO2(g) -2
Cl 179
F 44

# reduction of this produces a lot of OH- 1390
S(6) 1390 as SO4

Ca 191
Mg 43
Na 93
K 12

# 25 really
Al 25
Si 11.0
Sr 1.87
Ba 0.052
Li 2.7
Pe 346
Mn 19.7
As 2.1
Zn 125
# kg
-water 1
SAVE solution 1
END

USE solution 1
EQUILIBRIUM_PHASES 1
  Fix H+ <-x_axis> NaOH 10
  -force_equality true
  O2(g) <-y_axis> 0.1
# gets CH4 and native As
  CO2(g) -3.5 0.01
  Halite -6.34 10

# possible minerals
Al(OH)3(a) 0 0
As_native0 0
Ba3(AsO4)2 0 0
Barite 0 0
Calcite 0 0
Dolomite0 0
Fe(OH)3(a) 0 0
Fluorite0 0
Halloysite0 0
Hausmannite0 0
Hydrozincite0 0
Jarosite(ss) 0 0
Jarosite-H0 0
Jarosite-As0 0
Manganite0 0
Orpiment0 0
Pyrite 0 0
Pyrochroite0 0
Pyrolusite0 0
Realgar0 0
Rhodochrosite0 0
Siderite0 0
Sphalerite0 0
Strontianite0 0
ZnO(a) 0 0

# two surfaces
SURFACE 1
  Hfo_sOH Fe(OH)3(a) equilibrium_phase 0.005 53300
  Hfo_wOH Fe(OH)3(a) equilibrium_phase 0.2

# mainly guesswork but would be wrong to ignore completely
  Hao_sOH Al(OH)3(a) equilibrium_phase 0.005 7800
  Hao_wOH Al(OH)3(a) equilibrium_phase 0.2
END
This example shows the use of a user-defined constraint to cut off part of the diagram, namely the region at high pH where the total carbonate concentration is greater than 0.3 mmol/kg water. This is done in the `ht1combinedCO3.inc` file by adding an additional constraint to the type 3 (constraints) section of the `ht1` code:

```plaintext
581 IF(TOT("C")/h2o > 0.3) THEN 582 ELSE 590
582 PUNCH "CO3 > 0.3 mol/kgw",TOT("C")/h2o
583 nout3 = nout3+1
```

This approach is exactly the same as that used for identifying the normal oxygen and hydrogen constraints for defining the ‘water limits’. The area where the constraint applies is treated just like any other field. It inherits its name from the column heading written by the `ht1combinedCO3.inc` file and the colour from the fill colour dictionary.

The `simplify` keyword could be used to reduce the steps seen in some of the field boundaries. This is done by changing the simplify setting from its default value of 1 to a greater value, say 2.
SPECIATION

jobTitle       "C predominance in the presence of selected Ca, Mg and Zn carbonates"
calculationType ht1
calculationMethod 1
mainSpecies C
xmin       4.0
xmax       12.0
ymin       -5.0
ymax       -1.0
resolution 200

PLOT

plotTitle  "Ca-Mg-Zn-CO$_3$"
xtitle  pH
ytitle  "log $<i>f</i>$ CO$_2$(g)"
simplify 1.0
extraText "extratextcarbonates.dat"

CHEMISTRY

PHASES
Fix_H+
H+ = H+
log_k 0.0
Hydrozincite
Zn$_5$(OH)$_6$(CO$_3$)$_2$ + 10H+ = 5Zn+2 + 2CO$_2$ + 8H$_2$O
log_k 45.0
# Preis & Gamsjager 2001
-delta_H -256.5 kJ

SELECTED_OUTPUT

-high_precision true
-reset false

# includes CO2 constraint
include 'ht1combinedCO3.inc'

SOLUTION 1

temp      25
pH        7
pe        5
redox     pe
units     mmol/kgw
density   1

# kg
-water    1
Ca        5
Mg        2
Zn        5
Cl        24 charge

SAVE solution 1
END

USE solution 1
EQUILIBRIUM_PHASES 1

# ... but no Na present as a background electrolyte
Fix_H+ -<x_axis> NaOH 10
-force_equality true
O2(g)    -0.68 0.1
# limit at high pH (corners)
CO2(g)   <y_axis> 2

# This is necessary to provide a source of Na for Fix_H+ when it goes -ve
Halite   -12 0.1

#ZnCO3:H2O 0 0
Dolomite 0 0
Calcite 0 0
Smithsonite 0 0
#Aragonite 0 0
#Dolomite(d) 0 0
Magnesite 0 0
Zincite(c) 0 0
ZnO(a) 0 0
#Zn(OH)2-e 0 0
#Zn(OH)2-g 0 0
#Zn(OH)2-b 0 0
#Natron 0 0
#Zn(OH)2-c 0 0
#Nesquehonite 0 0
#Zn(OH)2-a 0 0
Hydrozincite 0 0
#Huntite 0 0
Brucite 0 0
#Artinite 0 0
Portlandite 0 0
#Zn2(OH)3Cl 0 0
#Hydromagnesite 0 0
#Zn5(OH)8Cl2 0 0
#ZnCl2 0 0
#ZnMetal 0 0
END
This is a low resolution plot (resolution = 50), hence the uneven boundaries. A low resolution is useful for quickly seeing what is involved before replotting at a higher resolution.
SPECIATION
jobTitle                             "Uranium redox and speciation"
calculationType                     "ht1"
calculationMethod                   1
mainSpecies                          "U"
xmin                                 2.0
xmax                                 10.0
ymin                                 -80.0
ymax                                 0.0
loopmin                              -6
loopmax                              -6
loopint                              0
looplogvar                           1
resolution                           50

PLOT
plotTitle                            "Uranium hydrolysis and redox<br>(low resolution=50)"
xtitle                               "pH"
ytitle                               "log <i>f</i> O<sub>2</sub>(g)"
extraText                            "extratextUCO3.dat"

CHEMISTRY
include 'ht1.inc'

SOLUTION 1
temp  25
pH    1.8
units mol/kgw
U     <loop>
Na    1e-1
Cl    1e-1
SAVE solution 1
END

USE solution 1
EQUILIBRIUM_PHASES 1
Fix_H+  <x_axis>  NaOH 10
         -force_equality true
O2(g)    <y_axis>  0.1
CO2(g)   -3.5      1.0
B-UO2(OH)2                  0 0
Gummite                           0 0
Na4UO2(CO3)3                    0 0
Nahcolite                         0 0
Natron                             0 0
Rutherfordine                     0 0
Schoepite                          0 0
Thermonatrite                     0 0
Trona                              0 0
U3O8(c)                            0 0
U4O9(c)                            0 0
UO2(a)                             0 0
UC3{(gamma)                       0 0
Uraninite(c)                      0 0
END
This example shows how temperature affects calcite-dolomite stability as a function of pH and CO$_2$(g) partial pressure. The results are shown as a series of predominance diagrams. The temperature varies from 0 to 80°C in increments of 10°C. This is achieved by using loopMin, loopMax and loopInt to define the looping variable <loop>. This loop variable is then equated to <temp> in the numericTags definition (for convenience) and <temp> is substituted in the appropriate place in the SOLUTION definition below.

The extraText file includes a line which uses <temp> to write the current temperature in the bottom left of each plot.

The multiPageFile setting has been set to true so that only one plot file is recorded. This contains the nine plots, page by page. The plot above is for the eighth plot which is for 70°C.
SPECIATION
  calculationType                      ht1
  calculationMethod                    1
  mainSpecies                          Ca
  xmin                                 5.0
  xmax                                 10.0
  ymin                                 -8.0
  ymax                                 0.0
  loopMin                              0
  loopMax                              80
  loopInt                              10
  resolution                           200
# change this for different temperatures
numericTags                          <temp> = <loop>

PLOT
  multiPageFile                        true
  plotTitle                            "Ca-Mg-CO$_3$ at a given temperature"
  xtitle                               pH
  ytitle                               "log $<i>f</i>$ CO$_2$(g)"
# includes the <temp> tag for putting temperature on plot
extraText                            "extratextcalcite.dat"

CHEMISTRY
# standard 'hunt and track' file
include ht1.inc

SOLUTION 1
  temp      <temp>
  pH        7
  units     mmol/kgw
  density   1
# kg
  -water    1
  Ca        1
  Mg        1
  Na        100
  Cl        100
SAVE solution 1
END

USE solution 1
EQUILIBRIUM_PHASES 1
  Fix_H+   -<x_axis> NaOH 10
  -force_equality true
# limit CO2 supply at high pH
  CO2(g)   <y_axis> 1
  Brucite                        0 0
  Calcite                        0 0
  Aragonite                      0 0
  Magnesite                      0 0
  Dolomite                       0 0
#Dolomite(d)                   0 0
  Artinite                       0 0
  Nesquehonite                   0 0
  Portlandite                    0 0
  Huntite                        0 0
  Nahcolite                      0 0
  Hydromagnesite                 0 0
  Natron                         0 0
  Thermonatrite                  0 0
  Trona                          0 0
END
A log $f_{O_2}(g)$-pH predominance diagram for phosphorus in the P-Fe-Al-Ca system. It is for aqueous species only – this is determined by the lack of any P (or other) minerals in the EQUILIBRIUM_PHASES data block. In that sense, the diagram is unrealistic since many minerals would precipitate at various places within the domain explored. This can be seen in the next example.
SPECIFICATION

jobTitle                            "Fe-Al-Ca-P"
calculationType                    htl1
calculationMethod                  1
mainSpecies                        "P"
xmin                                2.0
xmax                                10.0
ymin                                -80.0
ymax                                0.0
resolution                          500

PLOT

plotTitle                            "P-Fe-Al-Ca<br>(only aqueous species)"
xtitle                               pH
ytitle                               "log <i>f </i>O<sub>2</sub>(g)"
extraText                            "extratextP.dat"

CHEMISTRY

include 'ht1.inc'

# first simulation - initial solution calculation
SOLUTION 1

Temp      20
pH        1.8
units     mol/kgw
density   1
P         1e-3
Ca        1e-1
Fe        1e-1
Al        1e-1
Na        1e-1
Cl        1e-1 charge
SAVE solution 1
END

# second simulation - reaction and equilibration
USE solution 1
EQUILIBRIUM_PHASES 1

Fix_H+     -<x_axis> NaOH     
-force_equality true
O2(g)      -<y_axis>

# no minerals - aqueous species only
CO2(g)     -3.5
END
This diagram is for similar conditions to the previous example but crucially a variety of minerals has been allowed to precipitate and P adsorption onto HFO has been taken into account. There has been a deliberate policy of only allowing the more soluble (less stable) minerals to precipitate so for example, the more stable iron oxides such as hematite and goethite have been removed from consideration by commenting them out.

Adsorbed P dominates in the region where Fe(OH)$_3$ is stable. Under acid, oxidising conditions where Fe(OH)$_3$ dissolves, strengite (Fe(III)PO$_4$·2H$_2$O) is stable. Under reducing conditions, vivianite (Fe(II)$_3$(PO$_4$)$_2$·8H$_2$O) and hydroxyapatite (Ca$_5$(PO$_4$)$_3$OH) precipitate.

Dissolved P species only predominate under some of the most extreme conditions of high pH and strongly reducing conditions. The presence of CO$_2$(g) leads to calcite precipitation above pH 7.6 which ultimately leads to the lowering of the Ca$^{2+}$ activity to such an extent that hydroxyapatite becomes unstable.
SPECIATION

jobTitle "Fe-Al-Ca-P"
calculationType ht1
calculationMethod 1
mainSpecies "P"

# x-axis calculation range
xmin 2.0
xmax 10.0

# y-axis calculation range
ymin -80.0
ymax 0.0

# tracks on a 500 x 500 grid
resolution 500

PLOT

plotTitle "P-Fe-Al-Ca\n\(including\) mineral and adsorbed
species"
xtitle pH
ytitle "\(\text{log} c_{O_2(g)}\)"
extraText "extratextP.dat"

CHEMISTRY

include 'ht1.inc'

# first simulation - initial solution calculation

SOLUTION 1

Temp 20
pH 1.8

units mol/kgw
density 1
P 1e-3
Ca 1e-1
Fe 1e-1
Al 1e-1
Na 1e-1
Cl 1e-1 charge

SAVE solution 1

END

# second simulation - reaction and equilibration

USE solution 1

EQUILIBRIUM_PHASES 1

# fix the pH
Fix_H+ -<x_axis> NaOH
-force_equality true
O2(g) <y_axis>

# atmospheric PCO2(g)
CO2(g) -3.5

# choose the minerals you want (from the database)
Hydroxyapatite 0 0
Magnetite 0 0
Hematite 0 0
Vivianite 0 0
Fe3(OH)8 0 0
Goethite 0 0
Fe(OH)2.7Cl.3 0 0
Diaspore 0 0
Gibbsite 0 0
Maghemite 0 0
Boehmite 0 0
Al(OH)3(a) 0 0

# assumed the metastable Fe-oxide mineral
Fe(OH)3(a) 0 0
Portlandite 0 0
Strengite 0 0
Calcite 0 0
Siderite 0 0
SURFACE
# phosphate adsorbed by Hfo
Hfo_sOH Fe(OH)3(a)  equilibrium_phase 0.005  53300
Hfo_wOH Fe(OH)3(a)  equilibrium_phase 0.2

END
This is a pe-pH predominance diagram for Mn in which no minerals have been allowed to precipitate. The system is in equilibrium with CO₂(g) at close to its atmospheric partial pressure making MnCO₃(aq) stable at high pH where carbonate activities are high.

Permanganate (MnO₄⁻) becomes stable in a small region at high pH and under strongly oxidising conditions.

This is an example where the hunt and track algorithm has to automatically readjust the resolution in order to track the boundaries properly. It increases the resolution from 400 to 746.
# Mn predominance diagram for aqueous species only - CO2 included

**SPECIFICATION**

- **jobTitle**: "Mn-CO2-H2O"
- **calculationType**: htl1
- **calculationMethod**: 1

# diagram for Mn

- **mainSpecies**: "Mn"
- **pH range**: 2-10
  - **xmin**: 2.0
  - **xmax**: 10.0
- **log f(O2(g)) range**: -90 to 0
  - **ymin**: -90.0
  - **ymax**: 0.0

# track on a 300 x 300 grid

- **resolution**: 300

**PLOT**

- **plotTitle**: "Mn-CO2-H2O (CO2(g) but no minerals)"
- **xtitle**: pH
- **yscale**: pe
- **pymin**: -15
- **extraText**: "extratextMn.dat"

**CHEMISTRY**

# standard predominance calculating code

- **include 'ht1.inc'**

# first simulation - initial solution calculation

**SOLUTION 1**

- **Temp**: 20
- **pH**: 1.8
- **Mn**: 1e-2
- **Na**: 1e-1
- **Cl**: 1e-1 charge

**SAVE solution 1**

**END**

# second (final) simulation

**USE solution 1**

**EQUILIBRIUM_PHASES 1**

- **Fix_H+**: \(-<x_axis>\) NaOH
- **O2(g)**: \(-<y_axis>\)

# NB no minerals specified

- **CO2(g)**: \(-3.5\) 1
This is somewhat similar to the previous example except that minerals have been allowed to precipitate and no CO$_2$(g) is present. It shows the stability region of the various Mn oxides, some of which contain Mn in a mixed valence state.
# Mn predominance diagram including Mn oxide minerals (see Mn.ppi for aqueous species only)

**SPECIATION**

jobTitle                             "Mn-CO2-H2O"
calculationType                      htl1
calculationMethod                    1
#
## diagram for Mn
mainSpecies                          "Mn"
## pH range 2-10
xmin                                 2.0
xmax                                 10.0
## log f(O2(g)) range -90 to 0
ymin                                 -90.0
ymax                                 0.0
## track on a 400 x 400 grid
resolution                           400

**PLOT**

plotTitle                            "Mn-H$_2$O (oxide minerals included)"
xtitle                               pH
# drive redox with fO2(g) but use pe for plot yscale
yscale                               pe
# force plot y min at pe = -15
pymin                                -15
extraText                            "extratextMn.dat"

**CHEMISTRY**

# standard predominance calculating code
include 'ht1.inc'

# first simulation - initial solution calculation
**SOLUTION 1**

Temp      20
# initial pH is less than pHmin so adding NaOH should always work
pH        1.8
units     mol/kgw
# total Mn
Mn        1e-2
# background electrolyte
Na        1e-1
Cl        1e-1 charge

SAVE solution 1

**END**

# second (final) simulation
**USE** solution 1
**EQUILIBRIUM_PHASES 1**
Fix_H+     -<x_axis> NaOH
-<y_axis> O2(g)

# this list of minerals is considered
Pyrochroite                    0 0
Manganite                      0 0
Pyrolysite                     0 0
Neutite                        0 0
Birnessite                     0 0
Bixbyite                       0 0
Hausmannite                    0 0

**END**
This is similar to the previous example but in this case, CO$_2$(g) is present. This leads to the formation of rhodochrosite (MnCO$_3$) at high pH and under moderately to strongly reducing conditions. In this case, Hausmannite and Pyrochroite are no longer predominant Mn minerals.

The CO$_2$(g) is reduced to CH$_4$(g) under strongly reducing conditions.
# Mn predominance diagram including Mn oxide and carbonate minerals (see Mn.ppi for aqueous species only)

SPECIFICATION
    jobTitle        "Mn-CO2-H2O"
calculationType  htl1
calculationMethod 1
    # diagram for Mn
    mainSpecies     "Mn"
    # pH range 2-10
    xmin             2.0
    xmax             10.0
    # log f(O2(g)) range -90 to 0
    ymin             -90.0
    ymax             0.0
    # track on a 400 x 400 grid
    resolution       400

PLOT
    plotTitle        "Mn-H2O<br>(oxide minerals included)"
xtitle            pH
    # drive redox with fO2(g) but use pe for plot yscale
    yscale            pe
    # force plot y min at pe = -15
    pymin             -15
    extraText         "extratextMn.dat"

CHEMISTRY
    # standard predominance calculating code
    include 'ht1.inc'

    # first simulation - initial solution calculation
    SOLUTION 1
        Temp      20
        # initial pH is less than pHmin so adding NaOH should always work
        pH        1.8
        units     mol/kgw
        # total Mn
        Mn        1e-2
        # background electrolyte
        Na        1e-1
        Cl        1e-1 charge
    SAVE solution 1
END

    # second (final) simulation
    USE solution 1
    EQUILIBRIUM_PHASES 1
        Fix_H+    -<x_axis> NaOH
        -force_equality true
        O2(g)     -<y_axis>
        # atmospheric pCO2(g) - include up to 1 mole CO2 max
        CO2(g)    -3.5       1
            MnCl2:4H2O        0 0
            Pyrochroite      0 0
            Rhodochrosite    0 0
            Rhodochrosite(d) 0 0
            Manganite        0 0
            Pyrolusite        0 0
            Nautite           0 0
            Birnessite        0 0
            Bixbyite          0 0
            Hausmannite       0 0
END
This log $f_{O_2}(g)$-pH predominance diagram for plutonium ($10^{-12}$ mol/kgw Pu$_T$) in the presence of fluoride and carbonate demonstrates the extreme insolubility of PuO$_2$ under a wide range of conditions. It also shows that fluoride and carbonate form strong complexes with Pu(IV) and can maintain relatively high concentrations of Pu in solution. Reduction of Pu(IV) to Pu(III) under reducing and acidic conditions also enhances Pu solubility.
SPECIATION

jobTitle                             "Plutonium redox and speciation"
Database                             llnl.dat
epsi                                 T
calculationType                      ht1
calculationMethod                    1
mainSpecies                          "Pu"
xmin                                 2.0
xmax                                 10.0
ymin                                 -80.0
ymax                                 0.0
resolution                           100

PLOT

plotTitle                            "Plutonium hydrolysis and redox\(\text{br}\) (using llnl.dat database)"
xtitle                               pH
ytitle                               "\log <i>f</i> \textsubscript{O_2}(g)"
lableSize                            2.0
simplify                             10
extraText                            "extratextPu.dat"

CHEMISTRY

include 'ht1.inc'

SOLUTION 1

temp 25
pH 1.8
units mol/kgw
Pu 1e-12
Na 1e-1
Cl 1e-1
S 1e-3
F 1e-3
SAVE solution 1
END

USE solution 1

EQUILIBRIUM_PHASES 1

 Fix_H+ \rightarrow_x \text{NaOH} 10
-force_equality true
O2(g) \rightarrow_y 0.1
CO2(g) \rightarrow -3.5 1.0

PuO2 0 0
Pu(OH)4 0 0
NaHcolite 0 0
Natron 0 0
Na2CO3:7H2O 0 0
Thermonatrite 0 0
Pu(OH)3 0 0
PuO2OH(am) 0 0
Na2CO3 0 0
PuO2(OH)2 0 0
C 0 0
Pu2O3 0 0
Na 0 0
Na2O 0 0
Pu 0 0
END
This is one of three log $f(O_2(g))$-pH predominance diagrams for U ($10^{-6}$ mol/kgw $U_1$) which demonstrate how predominance diagrams provide a useful way of comparing thermodynamic databases, here made with \textit{wateq4f.dat}.

Uranium speciation is strongly dependent on the pH and redox conditions with the highly insoluble mineral Uraninite dominating reducing conditions. Uranium(VI) forms strong complexes with carbonate which enhances U solubility in the presence of CO$_2(g)$ and high pH.

The extraText file, \texttt{extratextUCO3.dat}, also adds the text in the top left corner of the diagram and demonstrates features such as subscripts, superscripts, italics, line breaks ('\texttt{<br>}' and the use of multiline input through the use of the '\' continuation character.
SPECIATION

jobTitle                             "Uranium redox and speciation"
Database                             "wateq4f.dat"
calculationType                      "ht1"
calculationMethod                    1
mainSpecies                          "U"
xmin                                 2.0
xmax                                 10.0
ymin                                 -80.0
ymax                                 0.0
loopmin                              -6
loopmax                              -6
loopint                              0
looplogvar                           1
resolution                           500

PLOT
plotTitle                            "Uranium hydrolysis and redox: wateq4f.dat"
xtitle                               "pH"
ytitle                               "log <i>f</i> O<sub>2</sub>(g)"
extraText                            "extratextUCO3.dat"

CHEMISTRY

include 'ht1.inc'

SOLUTION 1

temp  25
pH    1.8
units mol/kgw
U     1e-6
Na    1e-1
Cl    1e-1

SAVE solution 1
END

USE solution 1

EQUILIBRIUM_PHASES 1

Fix_H+ -<x_axis>  NaOH 10
-force_equality true
O2(g)  <y_axis>  0.1
CO2(g) -3.5      1.0

Uraninite(c)                            0 0
UO2(a)                                   0 0
U4O9(c)                                  0 0
Schoepite                                0 0
B-UO2(OH)2                               0 0
UO3(gamma)                               0 0
Nahcolite                                0 0
Gummite                                  0 0
Rutherfordine                            0 0
Natron                                   0 0
Thermonatrite                            0 0
U3O8(c)                                  0 0
Trona                                     0 0
Na4UO2(CO3)3                             0 0

END
35 U-C-H₂O (NAPSI)

The same as for the previous diagram but here made with the NAPSI database.
SPECIATION
jobTitle                           "Uranium redox and speciation"
Database                           "NAPSI_290502.DAT"
calculationType                   "ht1"
calculationMethod                 1
mainSpecies                        "U"
xmin                               2.0
xmax                               10.0
ymin                               -80.0
ymax                               0.0
loopmin                            -6
loopmax                            -6
loopint                            0
looplogvar                         1
resolution                         500

PLOT
plotTitle                          "Uranium hydrolysis and redox (NAPSI)"
xtitle                             "pH"
ytitle                             "log $<i>f</i> O_{2}(g)"
extraText                          "extratextUCO3.dat"

CHEMISTRY
include 'ht1.inc'

SOLUTION 1
temp 25
pH 1.8
units mol/kgw
U 1e-6
Na 1e-1
Cl 1e-1
SAVE solution 1
END

USE solution 1
EQUILIBRIUM_PHASES 1
Fix_H+ -<x_axis> NaOH 10
-force_equality true
O2(g) <y_axis> 0.1
CO2(g) -3.5 1.0

END
The same as for the previous diagram but here made with the llnl.dat database.

Note that one of the fields in the centre of the diagram has not been labelled (light grey, \(\text{UO}_2.3333^{(\beta)}\)) because it occupies less than the minimum area required to plot a label (as given by the keyword `minimumAreaForLabeling` which is 1% by default).

This sequence of diagrams demonstrates the quite large differences in speciation models for U in the various databases. This applies not only to the minerals but also the aqueous species.
SPECIATION
jobTitle                             "Uranium redox and speciation"
Database                             "llnl.dat"
calculationType                      "ht1"
calculationMethod                    1
mainSpecies                          "U"
xmin                                 2.0
xmax                                 10.0
ymin                                 -80.0
ymax                                 0.0
loopmin                              -6
loopmax                              -6
loopint                              0
looplogvar                           1
resolution                           500

PLOT
plotTitle                            "Uranium hydrolysis and redox: llnl.dat"
xtitle                               "pH"
ytitle                               "log \( <i>f</i> \) O\(_2\) (g)"
minimumAreaForLabeling               1.0
extraText                            "extratextUCO3.dat"

CHEMISTRY
include 'ht1.inc'

SOLUTION 1
  temp  25
  pH    1.8
  units mol/kgw
  U     1e-6
  Na    1e-1
  Cl    1e-1
SAVE solution 1
END

USE solution 1

EQUILIBRIUM_PHASES 1
  Fix_H+ -<x_axis>  NaOH 10
  -force_equality true
  O2(g)  <y_axis>  0.1
  CO2(g)  -1.5  1.0

  Ice                                 0 0
  Uraninite                           0 0
  UO2.25                               0 0
  UO2.25(beta)                        0 0
  UO2.3333{beta}                      0 0
  UO2(am)                             0 0
  Schoepite                           0 0
  UO3:.2H2O                            0 0
  UO2(OH)2(beta)                      0 0
  Schoepite-dehy(.9)                 0 0
  UO3:.9H2O(alpha)                     0 0
  Schoepite-dehy(.85)                0 0
  Schoepite-dehy(1.0)                 0 0
  UO2ClOH:2H2O                        0 0
  Schoepite-dehy(.648)                0 0
  UO2.6667                            0 0
  UO2Cl                               0 0
  Schoepite-dehy(.393)                0 0
  UO3(gamma)                          0 0
  UO3(beta)                           0 0
  UO3(alpha)                          0 0
  UO2Cl2:3H2O                         0 0
  NaUO3                               0 0
  UOCl2                               0 0
  UO2Cl2:H2O                          0 0
  U5O12Cl                             0 0
UO2C12  0  0
Na2U2O7  0  0
UOC13   0  0
Na2UO4 (alpha)  0  0
(UO2)2C13  0  0
UOC1   0  0
UC14   0  0
UC13   0  0
U2O2C15  0  0
Na   0  0
UC15   0  0
Na3UO4  0  0
Na2O   0  0
UC16   0  0
U   0  0
UH3 (beta)  0  0
END
This is a uranium (log $U_T = -9$) pe-pH predominance diagram with mineral formation and adsorption of U and carbonate on hydrous ferric oxide (HFO). There is also competition from carbonate complexation in solution at high pH.

The U adsorption is approximate in the sense that the U adsorption parameters have been taken from a source that is not necessarily consistent with the aqueous speciation database used here (see the code below for details). In principle, consistent databases should be used although this is often not possible and can be difficult to maintain.

Carbonate species are also adsorbed by HFO although they never become the dominant C species in the system. Maximum carbonate adsorption is at about pH 6.5.
SPECIATION

jobTitle                  "Uranium redox and speciation"
mainSpecies               "U"
calculationType           ht1
calculationMethod         1

# relatively recently revised database for U
database                  NAPSI_290502.DAT
 fillColorDictionary       "fillcolor.dat"

# minimum pH
xmin                      2.0
# maximum pH
xmax                      9.0
# minimum PO2(g) to generate variable redox
ymin                      -75.0
# maximum PO2(g)
ymax                      0.0
resolution                400

PLOT

plotTitle                 "U-Fe-C-H₂O<br>(with adsorbed speciation)"
xtitle                    pH
# use pe for the y-scale
yscale                    pe
# on the pe scale
pymin                     -10.0
extraText                 "extratextUhfo.dat"

CHEMISTRY

TITLE U Sorption to ferrihydrite according to DLM and database derived and used by

include 'U-Hfo.dat'
include 'ht1.inc'

# first simulation - initial solution calculations
SOLUTION 1

temp          25
# initial pH is just less than pHmin
pH            1.8
units         mol/kgw
Na            0.1
Fe            1e-4
U(6)          1e-6
Cl            0.1 charge

EQUILIBRIUM_PHASES
# NB name of related mineral is different from wateq4f.dat
Fe(OH)₃(am) 0 0

SURFACE 1

Hfo_sOH     Fe(OH)₃(am)  equilibrium_phase 0.005 53300
Hfo_wOH     Fe(OH)₃(am)  equilibrium_phase 0.2

SAVE solution 1
SAVE surface 1
END

# second simulation - loops on the final simulation
USE solution 1
USE surface 1

EQUILIBRIUM_PHASES 1

Fix_H+       <-x_axis> NaOH 10
-force_equality true
O₂(g)        <-y_axis> 0.1
CO₂(g)       -3.5 1.0

Graphite     0 0
UO₂(s)        0 0
Examples 384

38 U-Fe-C (risk colours)

The same as the previous example but with labelling and colouring more appropriate for conveying the risk of uranium mobilization to a non-technical audience.

Dissolved species are coloured red (‘Mobile’); adsorbed species are coloured orange (‘May be mobile’) and the mineral species are coloured green (‘Immobile’). The criteria for determining predominance are entirely set in the USER_PUNCH file, ‘risk.inc’, and can be changed to give whatever priorities you like – see the next example where the redox state is the defining criterion. In general, the SYS() function is your friend here.

Many of the axis settings have also been changed.
SPECIATION

jobTitle                           "Uranium redox and speciation"
mainSpecies                        "U"
calculationType                    htl1
calculationMethod                  1
# relatively recently revised database for U
database                           NAPS_290502(260802).DAT
fillColorDictionary                "fillcolor.dat"
# minimum pH
xmin                                2.0
# maximum pH
xmax                                9.0
# minimum PO2(g) to generate variable redox
ymin                                -75.0
# maximum PO2(g)
ymax                                0.0
resolution                          400

PLOT

plotTitle                           \\n   "U-Fe-C-H_2O<br>(with adsorbed speciation)"
xtitle                               "Acidity (pH)"
ytitle                               "Reduction/oxidation potential (pe)"
# use pe for the y-scale
yscale                               pe
# on the pe scale
pymin                                -10.0
plotTitleColor                      red
plotTitleSize                       5
axisNumberSize                      4
axisNumberColor                     "blue"
axisTitleSize                       4
axisTitleColor                      "blue"
axisLineWidth                       0.4
axisLineColor                       "blue"
tickSize                             3
tickColor                            "blue"
info                                 "nd" "blue"
extraText                            "extratextUhfo(risk).dat"

CHEMISTRY

TITLE U Sorption to ferrihydrite according to DLM and database derived and used by Luo et al. Journal of Contaminant Hydrology 92, 129-148 (2007)

include 'U-Hfo.dat'
# this uses simple 'traffic light' classification for aqueous, adsorbed and mineral phases
include 'risk.inc'

# first simulation - initial solution calculations
SOLUTION 1

 temp      25
# initial pH is just less than pHmin
pH        1.8
units     mol/kgw
Na        0.1
Fe        1e-4
U(6)      1e-6
Cl        0.1  charge

EQUILIBRIUM_PHASES

# NB name of related mineral is different from wateq4f.dat
Pe(OH)_3(am)  0 0
SURFACE 1

Hfo_sOH Fe(OH)$_3$(am)  equilibrium_phase 0.005  53300
Hfo_wOH Fe(OH)$_3$(am)  equilibrium_phase 0.2

SAVE surface 1
END

# second simulation - loops on the final simulation
USE solution 1
USE surface 1

EQUILIBRIUM_PHASES 1

Fix_H+ -&lt;x_axis&gt; NaOH 10
   -force_equality true
O2(g)  &lt;y_axis&gt;  0.1
CO2(g) -3.5  1.0

Graphite  0  0
UO2(s)    0  0
#Goethite  0  0

Siderite  0  0
FeCO3(pr)  0  0
#Fe(OH)$_3$(mic)  0  0

Schoepite  0  0
Rutherfordine  0  0
Fe(OH)$_3$(am)  0  0
#Fe(cr)  0  0

#Hematite  0  0

#Magnetite  0  0

END
This example shows a predominance diagram based solely on the redox state of the aqueous species for a given 'master species', here the element U. The example file loops over three total U concentrations and produces a multipage ps file.

This diagram has been produced by using the $SYS()$ function within the $ht1redox.inc$ file to give the moles of all possible redox states from -8 to +8, sorting them, and then returning the top three of these to the tracking routine within PhreePlot. These figures define the predominant 'species', albeit an aggregation of all species with the given valence state.

This redox speciation can include, in principle, solid and adsorbed species based on the valence of the species given in their dissolution (mineral) or association (surface) reactions. For example, magnetite which contains both Fe(2) and Fe(3) will contribute to both of these 'super' species. For a mineral to be actually present depends on its inclusion within an EQUILIBRIUM_PHASES block; similarly, adsorbed phases depend on SURFACE blocks.

However, be aware that for some minerals, such as the strongly covalent arsenic sulphides, the valence state of the elements within the mineral phase is in reality rather poorly defined. Also some tabulated dissolution reactions, such as that for Realgar, Manganite and CuMetal in watex4f.dat, involve release of an electron which means that the valence of these solid phases may be reported incorrectly.
# Predominance diagram for U in the presence of CO2(g).
# All species are aggregated into their valence states, e.g. U(4), U(5), U(6).
# Thanks to Remi Marsac (INE) for the suggestion.

SPECIFICATION

Database "NAPSI_290502.DAT"
calculationType "ht1"
calculationMethod 1
mainSpecies "U"
xmin 2.0
xmax 10.0
ymin -80.0
ymax 0.0
loopmin -6
loopmax -2
loopint 2
looplogvar 1
resolution 200

# used in extraText file
numericTag <logUt> = <logloop>

PLOT

plotTitle "Uranium redox states"
xtitle "pH"
ytitle "log <i>f</i> O<sub>2</sub>(g)"
extraText "extratextUredox.dat"
multipagefile t

CHEMISTRY

PHASES
Fix_H+
H+ = H+
log_k 0.

#include 'ht1.inc'
include 'ht1redox.inc'

SOLUTION 1

temp 25
pH 1.8
units mol/kgw
U <loop>
Na 1e-1
Cl 1e-1
SAVE solution 1
END

# main loop - iterate here
USE solution 1
EQUILIBRIUM_PHASES 1

Fix_H+ -<x_axis> NaOH 10
-force_equality true
O2(g) <y_axis> 0.1
CO2(g) -3.5 1.0

END
This is an example of a predominance diagram for U in a system containing many U species including U adsorbed to HFO and other components. The input file also demonstrates how to generate a number of diagrams (U, P, F, Fe, C) for the same system. The next two examples give the corresponding fluoride and phosphorus outputs for this system.

The calculations were made using the NAPSI_290502(260802).DAT database for most species but since this does not include surface species, it was combined with a revised database for HFO-U surface species. This is found in U.dat. These data from Luo et al. (2007) use the Dzombak and Morel (1990) DLM but the surface reaction for U is described in terms of bidentate binding.

The NAPSI database does not include such an extensive set of minerals as the wateq4f.dat database and is completely lacking in some trace elements. Therefore some second order interactions may be missed.

The total uranium concentration is set by the loop variable although in this case only one
value is selected, -6. This is converted to $10^{-6}$ before it is substituted by setting loopLogVar to 1. This value is then used in the SOLUTION data block with the <loop> tag.
SPECIATION

jobTitle                            "Uranium complexation"
database                             NAPSI_290502.DAT
calculationType                      ht1
calculationMethod                    1
# calculate diagrams for these 5 elements
mainSpecies                          U P F Fe C
# minimum pH etc used to generate the plot
xmin                                 2.0
xmax                                 10.0
ymin                                 -80.0
ymax                                 0.0
loopMin                               -6.0
loopMax                               -6.0
loopLogVar                            1
loopInt                               1
resolution                            200

PLOT

plotTitle                            "U complexation in the presence of F and P (loops on the main species)"
xtitle                               pH
ytitle                               "log <i>f</i> O<sub>2</sub>(g)"
xoffset                              40.0
yoffset                              150.0
labelSize                             1.7
extraText                             "extratextUPF.dat"
# make one ps file per element not one file overall
multiPagefile                         f

CHEMISTRY

# revised database for U surface species - also other surface species
include 'U-Hfo.dat'
# standard code to generate predominance diagrams
include 'ht1.inc'

SOLUTION 1

temp  25
pH    1.3
units mol/kgw
U     <loop>
Na    le-1
Cl    le-1
F     le-2
S(6)  le-2
P     le-4
Fe    le-2

EQUILIBRIUM_PHASES 1
Fe(OH)3(am)       0  0

SAVE solution 1

END

# second simulation - loops on the final simulation

#PRINT; reset true
USE solution 1
USE surface 1
EQUILIBRIUM_PHASES 1
Fix_H+ -<x_axis> NaOH 10
-force_equality true
O2(g)  <y_axis>  0.1
CO2(g) -3.5      1

# PSI database
Pyrite                               0  0
Troilite                              0  0
<table>
<thead>
<tr>
<th>Substance</th>
<th>Code</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>UF₄·2.5H₂O(cr)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Graphite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S(rhomb)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>#Goethite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>UO₂(s)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>#Fe(OH)₃(mic)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fe(cr)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fe(OH)₃(am)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Siderite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fe₂O₃(pr)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Chernikovite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>#Hematite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Schoepite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>#Magnetite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Melanterite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rutherfordine</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>U(OH)₂SO₄(cr)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(UO₂)₃(PO₄)₂·4H₂O(cr)</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

END
41 U-F-P (F)

The fluoride view of the previous example.
The phosphorus view of the previous example. Phosphorus adsorption is calculated using the DLM with default Dzombak & Morel (1990) model parameters.
This example was computed with the same conditions as in the previous example but used the ‘ht1’ approach rather than the ‘grid’ approach (the only difference was in the calculation Method setting; the same ht1.inc file was used for generating the predominant boundaries in both diagrams.

This diagram has failed to identify the Cu⁺ field since it is an ‘island’ and is not accessible by hunting along the domain boundaries or tracking internal boundaries. This field is found using the ‘grid’ approach which provides a more reliable but slower approach.
SPECIATION
  calculationType                      ht1
  calculationMethod                    1
  mainSpecies                          Cu
  xmin                                 2
  xmax                                 10
  ymin                                 -80
  ymax                                 0
  resolution                           100

PLOT
  plotTitle                            "Cu-S-C-H₂O₂<sub>2</sub>(no minerals)"
  xtitle                               pH
  ytitle                               "log <i>f </i>O₂(g)"
  extraText                            extratextCuS.dat

CHEMISTRY
  include 'ht1.inc'
  SOLUTION 1
    Temp      20
    pH        1.8
    units     mol/kgw
    Cu        1e-1
    S(6)      1e-1
    Na        1e-1
    Cl        1.032e-1
  SAVE solution 1
  END

USE solution 1
EQUILIBRIUM_PHASES 1
  Fix_H⁺   -<x_axis> NaOH
            -force_equality true
  O₂(g)    <y_axis> 0.1
  CO₂(g)   -3.5     1.0
  END
The regular ‘jagged’ lines for the Chalcocite-Djurleite and other boundaries, highlighted by the red symbols (pointSize = 1.5), are because of the low angle of the slopes of the boundaries in relation to the chosen resolution (resolution = 400). The resolution controls the spacing of the imaginary grid where evaluations take place. Straight boundaries with a low angle therefore tend to result in regular steps. This is not a property of the underlying speciation program but of the hunt and track algorithm used. The regular steps can be eliminated by increasing the simplification factor (see the next example). This does not require recalculation of the data just replotting (calculationType = 3) with a larger value of the simplification factor, simplify which is set to 1 here. Alternatively, recalculating with a higher resolution will reduce the step size.

Sometimes boundaries, especially mineral boundaries, can be ‘noisy’. This is a reflection of the speciation program but again the boundaries can be smoothed by increasing the simplification factor.
SPECIATION

  calculationType                      ht1
  calculationMethod                    1
  mainSpecies                          Cu
  xmin                                 2.0
  xmax                                 10.0
  ymin                                 -80.0
  ymax                                 0.0
  resolution                           400

PLOT

  plotTitle                            "Cu-S-C-H\textsubscript{2}<sub>2</sub>O<br>(simplify = 1)"
  xtitle                               pH
  ytitle                               "log <i>f</i> O\textsubscript{2}(g)"

# sets the sizes of the symbols used for an intermediate plot and
  trackSymbolSize                      1.5  1.5

# normal default of 1 - but note the jaggies in some of the low-angled boundaries
  simplify                             1
  extraText                            "extratextCuS.dat"

CHEMISTRY

# standard hunt and track file
  include 'ht1.inc'

SOLUTION 1

  temp      20
  pH        1.8
  units     mol/kgw

# total concns
  Cu        1e-1
  S(6)      1e-1

# background electrolyte
  Na        1e-1
  Cl        1e-1 charge

SAVE solution 1

END

USE solution 1

EQUILIBRIUM_PHASES 1

  Fix_H+     --\textless  x\_axis > NaOH
  -force_equality true
  O2(g)      --\textless  y\_axis > 0.1

# includes carbonate species
  CO2(g)     -3.5  1.0

# permitted minerals
  Chalcocite                     0 0
  Djurleite                      0 0
  Anillite                       0 0
  BlaubleiII                     0 0
  BlaubleiI                      0 0
  Covellite                      0 0
  CuMetal                        0 0
  Sulfur                         0 0
  Nantokite                      0 0
  Cuprite                        0 0
  Tenorite                       0 0
  Cu(OH)\textsubscript{2}        0 0
  Melanothallite                 0 0
  Nahcolite                      0 0
  Atacamite                      0 0
  CuCO3                          0 0
  Natron                         0 0
  Thermonatrite                  0 0
  Thenardite                     0 0
  Mirabilite                     0 0
  Chalcanthite                   0 0
  Malachite                      0 0
CuSO₄       0 0
Cu₂SO₄      0 0
Trona       0 0
CuOCuSO₄    0 0
Antlerite   0 0
Azurite     0 0
Brochantite 0 0
Langite     0 0

END
This is the same example as the previous example except that it uses a greater value of the simplification factor, simplify, compared with the previous example has eliminated the obvious stepping. The value of 'simplify' is normally set to 1. Values greater than 1 reduce the number of vertices used to draw the polygons while a value less (normally in the range 0.1 to 1) will give more. A value of 0 does no line simplification at all.

A value of 3 was chosen here which almost completely eliminates the intermediate points although some of the boundaries seem unnaturally sharp suggesting that a greater resolution would also help. Note that it is not necessary to recalculate the points in order to change the simplification but it is necessary to use *calculationMethod* 3 not 2.
SPECIATION

calculationType          ht1
calculationMethod        1
mainSpecies              Cu
xmin                      2.0
xmax                      10.0
ymin                     -80.0
ymax                      0.0
resolution               400

PLOT
plotTitle                "Cu-S-C-H<sub>2</sub>O<br>(simplify = 3)"
xtitle                   pH
ytitle                   "log <i>f </i>O<sub>2</sub>(g)"

# sets the sizes of the symbols used for an intermediate plot and
trackSymbolSize          1.5 1.5
# custom plot label anchor symbols
# increase smoothing of field boundaries above normal default of 1
simplify                 3
extraText                "extratextCuS.dat"

CHEMISTRY

# standard hunt and track file
include 'ht1.inc'

SOLUTION 1

temp                     20
pH                       1.8
units                    mol/kgw

# total concns
Cu                       1e-1
S(6)                     1e-1

# background electrolyte
Na                       1e-1
Cl                        1e-1 charge

SAVE solution 1
END

USE solution 1

EQUILIBRIUM_PHASES 1

fix_H+                -<x_axis> NaOH
-force_equality true
O2(g)                  <y_axis> 0.1

# includes carbonate species
CO2(g)                 -3.5  1.0

# permitted minerals
Chalcocite                 0 0
Djурилит                   0 0
Anilite                    0 0
BlaubreilII                0 0
BlaubreilI                 0 0
Covellite                  0 0
CuMetal                    0 0
Sulfur                     0 0
Nantokite                  0 0
Cuprite                    0 0
Tenorite                   0 0
Cu(OH)2                    0 0
Melanothallite             0 0
Nahcolite                  0 0
Atacamite                  0 0
CuCO3                      0 0
Natron                     0 0
Thermonatrite              0 0
Thenardite                 0 0
Mirabilite                 0 0
Chalcanthite               0 0
Malachite                  0 0
CuSO4                          0 0
Cu2SO4                         0 0
Trona                          0 0
CuOCuSO4                       0 0
Antlerite                      0 0
Azurite                        0 0
Brochantite                    0 0
Langite                        0 0

END
The *wateq4f.dat* database does not contain any data for EDTA so it is necessary to use the *minteq.v4.dat* database which does. However, it is also necessary to add data for the aqueous solubility of $\text{H}_2(\text{g})$ since this is not included in the *minteq.v4.dat* database.
SPECIATION
# for EDTA
  Database                              "minteq.v4.dat"
calculationType                      ht1
calculationMethod                    1
mainSpecies                          Cu

# pH range
  xmin                                 2.0
  xmax                                 10.0

# O2(g) range
  ymin                                 -85.0
  ymax                                 0.0
  resolution                           300

PLOT
  plotTitle                            "Cu-EDTA<br>(using minteq.v4.dat)"
xtitle                               pH
  pymin                                -10.0
  # use pe scale even though redox controlled by PO2(g)
  yscale                               pe
  # omit domain boundary lines
  domain                               F
  extraText                            "extratextCuedta.dat"

CHEMISTRY

SOLUTION_SPECIES
  # can help convergence (see Phreeqc_3 manual, KNOPS p 117)
  H2O + 0.01e- = H2O-0.01; log_k -9.0

PHASES
  # not in minteq.dat
  H2(g)
    H2 = H2
  # solubility of H2(g)
    log_k    -3.150
    delta_h  -1.759 kcal
  # standard hunt and track file
  include 'ht1.inc'

  # initial solution calculation
  SOLUTION 1
    temp  25
    # start ing pH below minimum pH cos adding NaOH
    pH      1.8
    # total concentrations
    Cu    1e-4
    Edta  1e-4
    # background electrolyte
    Na    1e-1
    Cl    1e-1
  SAVE solution 1
  END

  # faster split into two simulations as here
  USE solution 1
    # - only loops on last simulation by default

EQUILIBRIUM_PHASES 1
  # negative sign converts pH to logH
    Fix_H+ -<x_axis>  NaOH 10
    -force_equality true
  # limit to 0.1 mol O2 max
    O2(g)  <y_axis>  0.1

  # possible minerals
  Atacamite       0 0
  Cu(OH)2        0 0
CuMetal  0  0
Cuprite  0  0
Melanothallite  0  0
Nantokite  0  0
Tenorite  0  0
END
In this example, a predominance diagram is drawn but the y-axis is not related to redox but is the partial pressure of CO$_2$(g). The diagram is one way of showing the competition between three Ca minerals as a function of CO2(g) and pH – at different points, a fluoride, a phosphate and a carbonate predominate.
SPECIATION
jobTitle "Calcium in the presence of fluoride, phosphate and bicarbonate"
calculationType ht1
calculationMethod 1
mainSpecies Ca
xmin 4.0
xmax 9.0
ymin -4.0
ymax -2.0
resolution 200
PLOT
plotTitle "Ca-F-P-CO$_2$-H$_2$O"
xtitle pH
ytitle "log $f$ CO$_2$(g)"
extraText "extratextCaF.dat"

CHEMISTRY
# standard 'hunt and track' file
include 'ht1.inc'

SOLUTION 1
temp 25
pH 1.8
units mol/kgw
Ca 1e-2
F 1e-2
P 3e-3
Na 1e-1
Cl 1e-1 charge
SAVE solution 1
END

USE solution 1
EQUILIBRIUM_PHASES 1
Fix_H+ -<x_axis> NaOH 1
-force_equality true
CO2(g) <y_axis> 1
-force_equality true
Calcite 0 0
Fluorite 0 0
Fluorapatite 0 0
Calcite 0 0
Aragonite 0 0
END
A redox-pH predominance diagram for Ni in the presence of C and S. Since some of the Ni minerals do not have mineral names in the database (e.g. Ni(OH)$_2$), the htl1s.inc include file has been used rather than htl1.inc file. This is the same as the htl1 file except that it appends '(_s)' to mineral names making it clear that Ni(OH)$_2$(s) is a mineral whereas NiCO$_3$ is not.
# predominance diagram for Ni in the presence of S and C (employing the Halite ploy)

SPECIATION

jobTitle                             "Ni-S-C-H2O"
calculationType                      ht1
calculationMethod                    1
mainSpecies                          "Ni"
# calculate pH 2-10
xmin                                 2.0
xmax                                 10.0
# calculate log f(O2(g)) -80 to 0
ymin                                 -80.0
ymax                                 0.0
# track on a 250 x 250 grid
resolution                           250

PLOT

plotTitle                            "Ni-S-C-O_{2}-H_{2}O at 20^oC"
xtitle                               pH
ytitle                               "log <i>f</i> O_{2}(g)"
extraText                            "extratextNiS.dat"

CHEMISTRY

# first simulation - initial solution calculation

# standard predominance diagram code
include 'ht1s.inc'

SOLUTION 1

Temp      20
# initial pH less than pHmin to hope that Fix_H+ works (but see below)
pH        1.8
units     mol/kgw
# total Ni etc
Ni        1e-2
S(6)      1e-2
# background electrolyte
Na        1e-1 charge
Cl        1e-1
SAVE solution 1
END

# second simulation

USE solution 1
EQUILIBRIUM_PHASES 1
# Fix_H+ defined in ht1.inc
Fix_H+     -<x_axis> NaOH
-<x_axis> O2(g)
-force_equality true
# limit max CO2 supplied to 1 mol
CO2(g)     -1.5       1
# list of possible minerals
Millerite                      0 0
Sulfur                         0 0
Ni(OH)2                        0 0
# reduction of S(6) produces a lot of OH- so may actually need HCl
Halite                         -12 1 dis
# (or -NaOH) to adjust pH - can’t specify +NaOH and +HCl so use -NaOH.
Bunsenite                      0 0
# This Halite phase ensures that there will always be some Na to take away.
NiCO3                          0 0
# To avoid this, either reduce initial pH to pH 1 or add more Na(Cl) or add less S(6).
<table>
<thead>
<tr>
<th>Mineral</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nahcolite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Morenosite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Retgersite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Natron</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Thermonatrite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Thenardite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mirabilite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Ni₄(OH)₆SO₄</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Trona</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

END
This is similar to the previous example except that it is for Zn not Ni. As before, the ‘htls.inc’ file was used so that ‘(s)’ has been appended to the mineral names, which in this case included Zn$_5$(OH)$_8$Cl$_2$. 
SPECIATION
jobTitle                             "Zn-C-H<sub>2</sub>O"
calculationType                      ht1
calculationMethod                    1
mainSpecies                          Zn
xmin                                 2.0
xmax                                 10.0
ymin                                 -80.0
ymax                                 0.0
resolution                           200

PLOT
plotTitle                            "Zn-S-C-H<sub>2</sub>O"
xtitle                               pH
ytitle                               "log <i>f</i> O<sub>2</sub>(g)"
extraText                            "extratextZnS.dat"

CHEMISTRY
include 'ht1s.inc'

PHASES
Hydrozincite
    Zn5(OH)6(CO3)2 + 10H+ = 5Zn+2 + 2CO2 + 8H2O
    #9.0
    log_k 45.0
    #Preis & Gamsjager 2001
    -delta_H -256.5 kJ

SOLUTION 1
Temp      20
# start at pH less than pHmin for Fix_H+
    pH        1.8
    units     mol/kgw
# total Zn
    Zn        1e-1
# also redox sensitive
    S(6)      1e-1
    Na        1e-1
    Cl        1e-1 charge
SAVE solution 1
END

USE solution 1
EQUILIBRIUM_PHASES 1
Fix_H+     -<x_axis> NaOH
-force_equality true
O2(g)      <y_axis>  0.1
CO2(g)     -3        1.0
# minerals that could ppt
    Sphalerite                          0 0
    Wurtzite                            0 0
    ZnS(a)                              0 0
# to maintain some Na at all times
    Halite                              -10 1 dis
    Sulfur                               0 0
    ZnO(a)                               0 0
    Zincite(c)                           0 0
    Zn(OH)2-e                            0 0
    Zn(OH)2-g                            0 0
    Zn(OH)2-b                            0 0
    Zn(OH)2-c                            0 0
    Zn(OH)2-2-a                          0 0
    ZnCl2                                0 0
    Zn2(OH)3Cl                           0 0
    Smithsonite                          0 0
    Nahcolite                            0 0
<table>
<thead>
<tr>
<th>Mineral</th>
<th>ZnMetal</th>
<th>Natron</th>
<th>Goslarite</th>
<th>Thermonatrite</th>
<th>Bianchite</th>
<th>ZnSO4·H2O</th>
<th>Thenardite</th>
<th>Mirabilite</th>
<th>Zn₅(OH)₈Cl₂</th>
<th>Znocosite</th>
<th>Zn₂(OH)₄SO₄</th>
<th>Trona</th>
<th>Zn₄(OH)₆SO₄</th>
<th>Zn₃O(SO₄)₂</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

END
In this example, simplify was set to a high value (10) in order to eliminate the 'steppiness' of the low-angled Greenockite-Otavite boundary.
SPECIATION
  jobTitle                             "Cd in the presence of sulphur"
calculationType                      htl1
calculationMethod                    1
mainSpecies                          Cd
# logH range
  xmin                                 2.0
  xmax                                 10.0
# O2(g) range
  ymin                                 -80.0
  ymax                                 0.0
resolution                           100
# straightens out the low-angled boundary
  simplify                             10

PLOT
  plotTitle                            "Cd-S-C-H<sub>2</sub>O at 20<sup>o</sup>C"
  xtitle                               pH
  ytitle                               "log <i>f</i> O<sub>2</sub>(g)"
  extraText                            "extratextCdS.dat"

CHEMISTRY
  # standard 'hunt and track' file
  include htl1.inc

SOLUTION 1
  Temp      20
  pH        1.8
  units     mol/kgw
  # total Cd
    Cd        1e-1
    S(6)      1e-1
  # background electrolyte
    Na        1e-1
    Cl        1e-1
  SAVE solution 1
END

USE solution 1
EQUILIBRIUM_PHASES 1
  Fix_H+       -<x_axis> NaOH
               -force_equality true
  O2(g)       <y_axis> 0.1
  CO2(g)      -3.5     1.0

  Greenockite                        0 0
  CdCl2:2.5H2O                       0 0
  CdCl2:2H2O                         0 0
  CdCl2                             0 0
  Sulfur                             0 0
  CdOHCl1                            0 0
  Cd(OH)2                            0 0
  Cd(OH)2(a)                         0 0
  Monteponite                        0 0
  Cd(OH)2:2(a)                       0 0
  CdMetal                             0 0
  Cd(gamma)                           0 0
  Otavite                             0 0
  Nahcolite                           0 0
  Natron                              0 0
  Thermonatrite                      0 0
  CdSO4:2.7H2O                       0 0
  CdSO4:H2O                          0 0
  Thenardite                          0 0
  Mirabilite                          0 0
  CdSO4                               0 0
  Trona                               0 0
  Cd3(OH)4SO4                         0 0
  Cd4(OH)6SO4                         0 0
  Cd3(OH)2(SO4)2                     0 0
This is quite a ‘challenging’ diagram to generate. It has with competing mineral phases and some close, low-angled boundaries.

The colour of the info text and the field labels have been set to blue with `info` and `labelColor`, respectively.
SPECIATION

jobTitle                             "Lead speciation"
# large database
Database                             llnl.dat
# hunt and track approach
calculationType                      ht1
calculationMethod                    1
mainSpecies                          "Pb"
xmin                                 2.0
xmax                                 10.0
ymin                                 -80.0
ymax                                 0.0
# tracks on a 500 x 500 grid
resolution                           500

PLOT

plotTitle                            "Pb-S-C-P-H<sub>2</sub>O<br>(two simulations)"
xtitle                               pH
ytitle                               "log <i>f</i> O<sub>2</sub>(g)"
# colour of the field labels
labelColor                           blue
# colour of the info and filename
info                                 nd blue
extraText                            "extratextPb.dat"

CHEMISTRY

include 'ht1.inc'

# first simulation - initial solution calculation only
SOLUTION 1

  temp  25
  pH    1.8
  units mol/kgw
  Pb    5e-4
  S     2e-4
  F     5e-4
  P     1e-5
  Na    1e-1
  Cl    1e-1 charge

SAVE solution 1

END

# second simulation - equilibrate
# saves recalculation
USE solution 1

EQUILIBRIUM_PHASES 1

  Fix_H+  -<x_axis>  NaOH 10
  -force_equality true
  O2(g)  <y_axis>  0.1
  CO2(g) -3  1.0

# these minerals were obtained by first running with resolution = 1 & PRINT -true
Pb(H2PO4)2                     0 0
Pb4O(PO4)2                     0 0
Pyromorphite-OH                0 0
Pb3(PO4)2                      0 0
PbHPO4                          0 0
Galena                          0 0
#Ice                             0 0
Matlockite                      0 0
PbFCl                           0 0
#Halite                          0 0
Cotunnite                       0 0
Paralaurionite 0 0
S 0 0
Pb 0 0
PbF2 0 0
C 0 0
Litharge 0 0
Massicot 0 0
Nahcolite 0 0
Cerussite 0 0
Anglesite 0 0
Phosgenite 0 0
Pb2Cl2CO3 0 0
Mirabilite 0 0
Pb4Cl2(OH)6 0 0
Thenardite 0 0
Natron 0 0
Na2CO3·7H2O 0 0
Thermonatrite 0 0
Na2CO3 0 0
Lanarkite 0 0
PbCO3·PbO 0 0
Na 0 0
Plattnerite 0 0
Pyromorphite 0 0
Pb3SO6 0 0
Na2O 0 0
Pb4S07 0 0
Na3H(SO4)2 0 0
Hydrocerussite 0 0
Minium 0 0
Burkeite 0 0
END
This is also an awkward example to specify properly because Sb(OH)$_3$ is present as both a solution species and a mineral species in the llnl.dat database. It is necessary to ensure that the species names are actually different not just for plotting but also so that the tracking is able to differentiate between them. As before, this is achieved by using the hts.inc include file which appends '(s)' to all mineral species names. In this case, the differentiation is not just useful for making the plot more legible but is also important for actually generating the proper boundaries.

The resolution has been set at 400 which produces the plot without problems. However, PhreePlot fails to converge at some lower resolutions such as 100 because of problems in the lower left-hand corner. There is a narrow sliver of Sb(s) which is uncomfortably close to the lower axis boundary. PhreePlot therefore automatically increases the resolution until it converges. The ‘steppiness’ of the low-angled boundaries at low resolutions can be reduced or eliminated by increasing the resolution or the simplification factor using simplify.
SPECIFICATION

jobTitle                 "Sb-O2-H2O"
# has Sb data
Database                 llnl.dat
calculationType          htl
calculationMethod        1
mainSpecies              "Sb"
xmin                     5.0
xmax                     10.0
ymin                     -85.0
ymax                     0.0
# need an even higher resolution to get smooth low-angled boundaries
resolution               400

PLOT

plotTitle                "Sb-S-O<sub>2</sub>-CO<sub>2</sub>-H<sub>2</sub>O<br>(demonstrates increased weighting of Fix_H+)"
xtitle                   pH
ytitle                   "log <i>f</i> O<sub>2</sub>(g)"
extratext                "extratextSbS.dat"

CHEMISTRY

# first simulation

# adds "...(s)" for mineral phases to avoid confusion with aq species
include 'ht1s.inc'

PHASES

Fix_H+

H+ = H+
log_k 0.0

# Kerfoot, GWB 15 Dec 2005
SOLUTION 1
Temp      18
pH        3.00
units     mol/kgw
# total Sb
Sb        3e-7
S(6)      4.4e-4
Na        0.0027
K         0.00020
Mn        1e-4
Ca        9.93e-4
Fe(2)     1e-4
Cl        5e-5
SAVE solution 1
END

# second simulation
USE solution 1
EQUILIBRIUM_PHASES 1

Fix_H+  -<x_axis> NaOH
         -<force_equality true
O2(g)    <y_axis> 0.1
CO2(g)   -3.523 1.0

Pyrite              0 0
Stibnite            0 0
Sb                  0 0
Sb(OH)3             0 0
S                  0 0
Pe(OH)2             0 0
Mn(OH)2(am)         0 0
Sb2O3               0 0
Pe(OH)3             0 0
Sb2O4               0 0
<table>
<thead>
<tr>
<th>Mineral Type</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sb4O6 (cubic)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rhodochrosite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Siderite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Calcite</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sb4O6 (orthorhombic)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sb2O5</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

END
This plot was produced with the wateq4f.dat database. The low-angled boundaries for Se(s) was rather ‘steppy’ and so simplify was set to 3. Selenium metal is stable under a wide range of reducing conditions.
SPECIATION
  jobTitle                             "Plutonium redox and speciation"
  "contains Se species"
  Database                             wateq4f.dat
  calculationType                      htl1
  calculationMethod                    1
  mainSpecies                          "Se"
  # pH 2 to 10
  xmin                                 2.0
  xmax                                 10.0
  # redox
  ymin                                 -80.0
  ymax                                 0.0
  # some low-angled boundaries could benefit from higher resolution or more smoothing
  resolution                           200

PLOT
  plotTitle                            "Selenium protonation and redox<br>(no adsorption)"
  xtitle                               pH
  ytitle                               "\log f O_2(g)"
  labelSize                            2.0
  extraText                            "extratextSe.dat"

CHEMISTRY

  # first simulation - initial solution calculation
  # each adsorbed species is counted separately
  include 'ht1.inc'

SOLUTION 1
  temp 25
  # start out at pH<xmin
  pH 1.8
  units mol/kgw
  # total Se
  Se 1e-3
  Na 1e-1
  Cl 1e-1
  S 1e-3
  SAVE solution 1
END

  # second simulation - equilibrate
  USE solution 1
  EQUILIBRIUM_PHASES 1
    Fix H+ <x_axis> NaOH 10
    -force_equality true
    O2(g) <y_axis> 0.1
    CO2(g) -3.5 1.0
  # possible Se minerals
    Se(s) 0 0
    Mirabilite 0 0
    Thenardite 0 0
    Sulfur 0 0
    SeO2 0 0
END
PhreePlot is not recommended for drawing the type of mineral stability diagrams often used in mineral geochemistry as the large range of activities often involved can lead to problems of convergence. However, in principle, such diagrams can be calculated and this example is one such calculated using the ‘ht1’ procedure. It plots the most abundant mineral at any particular point. The logic for determining the boundaries is in the include file ‘minstab1.inc’.

The main species has been set to ‘mineral’ as it is counting all minerals, not just minerals of a particular element. The x- and y-axes are driven by ‘fixing’ the \( H_4SiO_4 \) activity and the \( K^+/H^+ \) activity ratio, respectively, using fictitious phases defined in the PHASES keyword block.

The diagram shows the predominant mineral species (in terms of moles). Pure phases fix the activity or activity product of their constituent species. The indicated mineral is often the only mineral present (except on the phase boundaries). This is reflected by the Na code that appears on the screen for the sub-dominant species. If no mineral is stable, the field is labelled ‘No minerals present’. This can be demonstrated in this example by changing the units of concentration from mol/kgw to umol/kgw.

It can be difficult to fix the activity ratios over a wide range of values using the present approach and numerical errors can mean that the boundaries are rather ragged (see bottom right-hand corner). In such cases, the ‘grid’ approach may be a better option. An alternative is to plot the mineral with the largest theoretical supersaturation (see the minstab2.ppi demo).
Examples 437

# an example of a classical mineral stability diagram - diagram only includes minerals (not aqueous etc species)

SPECIATION
jobTitle                             "Clay mineral stability diagram"
Database                             "phreeqc.dat"
# use this approach for finding field boundaries of most abundant minerals
calculationType                      hti
calculationMethod                    1
# NB "minerals" is a special case that invokes this type of plot
mainSpecies                          "minerals"
xmin                                 -5.0
xmax                                 -1.0
ymin                                 -2.0
ymax                                 8.0
resolution                           200
PLOT
plotTitle                            "Clay mineral stability diagram
mainSpecies = minerals"
xtitle                               "log H\_4SiO\_4"
ytitle                               "log (K\^+ /H\^+)"

CHEMISTRY
# first simulation - initial solution calculation
# special file for generating mineral stability diagrams
include 'minstabl.inc'

PHASES
# used for driving the x-axis variable
Fix_Si
 H\_4SiO\_4 = H\_4SiO\_4
log_k 0.0

# used for driving the y-axis variable
Fix_H/K
 KOH = K\^+ + H\_2O - H\^+
log_k 0.0

PRINT
 reset FALSE
SOLUTION 1
# this also controls the diagram
pH       7
 units    mol/kgw
# added by reaction
 K       0
 Na      1e-2
 Cl      1e-2
 Al      1e-2
# added by reaction
 Si      0
 Ca      1e-2
SAVE solution 1
END

# second (final) simulation - iterates on this simulation when driving the x- and y-axes
USE solution 1
EQUILIBRIUM_PHASES 1
# fix H\_4SiO\_4 activity
 Fix_Si <x_axis> H\_4SiO\_4 10
# fix H/K activity ratio
 Fix_H/K <y_axis> KOH 10

# list of minerals considered
<table>
<thead>
<tr>
<th>Mineral</th>
<th>SiO2(a)</th>
<th>Si activity fixed by</th>
<th>Ca-Montmorillonite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kaolinite</td>
<td>0 0</td>
<td></td>
<td>0 0</td>
</tr>
<tr>
<td>K-feldspar</td>
<td>0 0</td>
<td># can’t add this</td>
<td></td>
</tr>
<tr>
<td>K-mica</td>
<td>0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gibbsite</td>
<td>0 0</td>
<td>cos Si</td>
<td></td>
</tr>
</tbody>
</table>

END
Custom plots

Custom plots refer to plots created directly from USER_PUNCH code. Phreeqc provides a very versatile mechanism for writing output to the selected output 'file' using BASIC code within USER_PUNCH keyword data blocks. This output is accumulated in the 'out' file which is then used for plotting.

The following examples give a guide as to how to create custom plots.
55  Gibbsite solubility vs pH

A custom plot showing the concentration of Al complexes as a function of pH. The species plotted have been explicitly defined in the input file. The saturation index for gibbsite has also been plotted.

This example demonstrates simple looping using the x axis variable. $\text{Xmin}$ and $\text{xmax}$ control the range of values taken by the $\langle \text{x_axis} \rangle$ tag. $\text{Resolution}$ determines the number of sub-divisions within the x-axis and so directly controls the number of points calculated for each curve. Here the resolution is 200 which is more than enough to get smooth curves.

Species names and plot labels have been defined by the headings given in the $\text{USER_PUNCH}$ keyword data block. This writes the headings to the selected output file which are then copied to the 'out' file which is then used for plotting. Note that text enhancement tags such as subscript can be used in the headings and passed through to the plot.

The curves plotted have been defined with the $\text{lines}$ keyword and have auto colour selection. The order of plotting and the order in which the labels are printed in the legend is determined.
by the order punched in the selected output file. The legend has been moved from its default position to the right of the plot into the plot area using the <legend> tag in the extraText file.
SPECIFICATION

calculationType                      custom
calculationMethod                    1
xmin                                 2.0
xmax                                 12.0

# determines the number of 'points' on the curves (i.e. PHREEQC runs)
resolution                           101

PLOT

# <br> causes a line break
plotTitle                            "Al solubility vs pH<br>(plotting named species)"
xtitle                               pH
ytitle                               "log concn (mol/kgw)"

# force the y-axis range
pymin                                -10.0
pymax                                2.0

# in the units currently in force
lineWidth                            0.4

# x-axis variable -‘pH’ must match the name of one of the punched columns below
customXcolumn                        pH

# y-axis variables in legend order
lines                                Al+3 AlF+2 Al(OH)4- SI<sub>Gibbs</sub>
Al<sub>T</sub> Gibbsite

extraText                            "extratextAlvsph.dat"

# turns off the little red label 'anchors'
trackSymbolSize                      0

CHEMISTRY

SELECTED_OUTPUT
reset false
high_precision true

PHASES

Fix_H+

H+ = H+
log_K 0.0

SOLUTION 1 Total Al

units mol/kgw
Al  1e-3
F  1e-3
S(6) 1e-3
Na  1e-1
Cl  1e-1

USER_PUNCH

# this is where 'pH' and all the y-axis variables are defined
headings pH  Al+3 AlF+2 Al(OH)4- SI<sub>Gibbs</sub> Al<sub>T</sub> Gibbsite

-start
10 totel = SYS(“Al”,n,n$,t$,c)
20 mol_gibbsite = equi("Gibbsite")
30 REM -99999 is PhreePlot’s UNDEFINED value
40 IF (mol_gibbsite > 0) THEN log_gibbsite = log10(mol_gibbsite) ELSR log_gibbsite
   = -99999
50 PUNCH -la("H+"), lm("Al+3"), lm("AlF+2"), lm("Al(OH)4-"), SI("Gibbsite"),
   log10(tot("Al")), log_gibbsite
-end

END

USE solution 1

EQUILIBRIUM_PHASES 1

Fix_H+          -<x_axis> NaOH 10
-force_equality true

# these are the possible minerals considered
Gibbsite  0 0
# Al(OH)3(a)  0 0
Basaluminite  0 0
Boehmite  0 0
Jurbanite  0 0

END
56  Acid titration of groundwater (using ‘REACTION’)

This example demonstrates how a single iteration of Phreeqc can generate a multiline
SELECTED_OUTPUT file. Each of the reaction steps produces a line of output. RXN gives the
moles of reactant used at each step and this is converted to mmoles for plotting. resolution has
been set to 1 because only iteration is used.

The selectedOutputLines setting has been set to auto which signals that all lines in the
selected output are transferred to the ‘out’ file rather than just the last line.

The labels normally attached to each line have been suppressed by setting labelSize to 0. The
legend has been moved inside the plot with the <legend> tag in the extraText file.

It is much faster to use Phreeqc’s internal looping like this compared with PhreePlot’s looping
mechanisms. Having said that, calculation times are often so short that speed is not an issue
for simple calculations like this.
SPECIATION

calculationType custom

calculationMethod 1

# get as many lines as there are -> out file

selectedOutputLines auto

PLOT

PlotTitle "Acid titration of 50 mL of groundwater (using REACTION keyword)"

xtitle "ml 0.16M HCl"

ytitle pH

customXcolumn ml

pxmax 3

# from selected output lines

points pH Gran

# from selected output points

gran

# suppress curve labels inside plot

labelSize 0

lineColor blue

pointColor green

extraText extratexttitration.dat

CHEMISTRY

SELECTED_OUTPUT

-reset false

# Groundwater

# the groundwater to titrate with HCl

SOLUTION 1

pH 7.05

units mg/L

# temp 10.5

water 0.050 kg

Na 6

K 0.6

Ca 124

Mg 1.6

Cl 11

Alkalinity 348 as HCO3

S(6) 3 as SO4

Si 5.8

REACTION 1 Add HCl to the soln

# 1 mL of 0.16M HCl

# this takes into account the dilution since it includes water

HCl 0.16e-3

H2O 55.5e-3

3 in 50 steps

USER_PUNCH

-headings ml pH water Gran

# assumes density = 1

10 VT = TOT("water")*1000

20 V = VT-50

30 pH = -la("H+")

40 Gran = VT*(10^-pH)*30

50 punch V, pH, VT, Gran

GAS_PHASE

-fixed_volume

# 10 mL gas + 50 mL solution

-volume 0.01

# equilibrate with solution 1 to begin with - this leads to some initial degassing

-equilibrate 1

CO2(g)

END
57 Acid titration of groundwater (using PhreePlot looping)

This is essentially the same example as the previous example but has been calculated using one of PhreePlot’s own looping mechanisms. This involves using a 1 mol/kgw solution of HCl to titrate the groundwater. The titration is achieved using the MIX keyword.

This approach includes the dilution brought about by the titration (the REACTION approach essentially titrates with ‘solid’ HCl). In this case, the dilution is very small.
calculationType: custom
calculationMethod: 1
xmin: 0.0
xmax: 3.0E-03
resolution: 50
numericTags: <titre> = "<x_axis>"

PLOT
plotTitle: "Acid titration of 50 mL of groundwater<br>(using MIX keyword)"
xtitle: "ml 0.16M HCl"
ytitle: pH
lineColor: blue
customXcolumn: ml
lines: pH Gran
points: Gran
pointColor: green
labelSize: 0
extraText: "extratexttitration.dat"

CHEMISTRY

SELECTED_OUTPUT
-reset false

TITLE Acid titration of groundwater (assumes no CO2 loss)

# Groundwater
SOLUTION 1
  pH  7.05
  units mg/L
  temp 10.5
  water 0.050 kg
  Na  6
  K  0.6
  Ca 124
  Mg  1.6
  Cl 11
Alkalinity 348 as HCO₃
S(6) 3 as SO₄
Si 5.8

SOLUTION 2
 units mol/kgw
# 0.16 mol/kgw HCl
 pH 1 charge
Cl 0.16
END

# mix two solutions, the sample and the acid
MIX 1 Add 0.1M HCl to the soln
 1 1
# driven by x loop parameters, see above
 2 <titre>

USER_PUNCH
-headings ml pH VT Gran
 1 pH = -la("H+")
 10 V = <titre>*1000
 11 VT = (0.05 + <titre>)*1000
 20 Gran = VT*(10^-pH)*30
 30 punch V, pH, VT, Gran

GAS_PHASE
-fixed_volume
# 10 mL gas + 50 mL solution
-volume 0.01
# equilibrate with solution 1 to begin with - this leads to some initial degassing
-equilibrate 1
 CO2(g)

END

Redox sequence

This example (from Appelo and Postma, 2005, Fig. 9.17) shows how a single iteration of Phreeqc (using the reaction keyword) can generate a series of points that can be assembled to give the ‘redox ladder’ plot indicated. The reaction keyword generates its own internal looping and so there is no need for PhreePlot loops.

The curves show the successive reduction of various solutes as the groundwater is titrated with
C. (as in organic matter) in the presence of a small amount of goethite and pyrolusite.

The label names have been set explicitly by making them the names for the headings in the selected output. These names get passed to the ‘out’ file which is then used for plotting. Note that the default assumes that all labels are species names and so are interpreted with superscripts etc. accordingly. This behaviour can be suppressed by setting \texttt{convertLabels} to \texttt{FALSE}.

The order of species plotted and in the legend is determined from the order punched to the selected output. FeS(ppt) is the only mineral that is allowed to form.

The script could be generalised by using tags to define the number of steps used, the mol of C added and the initial solution concentrations. For example, to define just the first two of these, the following changes could be made:

(i) add to the \texttt{PhreePlot} section

\begin{verbatim}
numericTags                           <steps> = 100 \ 
<molCadded> = 0.572e-3
\end{verbatim}

(ii) change line 10 in the \texttt{USER\_PUNCH} data block

10 addedc=step_no*<molCadded>*1e3/<steps>

(iii) change the \texttt{REACTION} data block

\begin{verbatim}
REACTION 1
  CH2O; <molCadded> in <steps> steps
INCREMENTAL\_REACTIONS true
END
\end{verbatim}
# titrate with C (like glucose)

**SPECIATION**

- **jobTitle**: Development of redox zones (A&P, Fig 9.17)
- **calculationType**: custom
- **calculationMethod**: 1

# just one iteration since REACTION has its own looping mechanism

# copy all lines in selected.out to out file

**PLOT**

- **plotTitle**: Development of redox zones (using REACTION)
- **xtitle**: Carbon added (mmol/L)
- **ytitle**: Concentration (mmol/L)

**CHEMISTRY**

**SELECTED_OUTPUT**

- reset false
- high_precision true

**USER_PUNCH**

- headings C O2 NO3- Mn+2 Fe+2 SO4-2 S-2 CH4 As(V) As(III)

- start

10 addedc=step_no*0.572/100
20 punch addedc, 1000*tot("O(0)")/2, 1000*tot("N(5)"), 1000*tot("Mn"),
1000*tot("Fe(2)")\,
1000*tot("S(6)"), 1000*tot("S(-2)"), 1000*tot("C(-4)"), 1000*tot("As(5)"),
1000*tot("As(3)")

- end

**SOLUTION 1**

- pH 7.1
- Na 1.236
- K 0.041
- Mg 0.115
- Ca 0.067
- Cl 1.467
- N(5) 0.058
- S(6) 0.085
- As(5) 0.075
- Alkalinity 0.26
- O(0) 0.124

**EQUILIBRIUM_PHASES**

- start with some
  - Goethite 0 2.5e-3
- start with none
  - FeS(ppt) 0 0
- start with some
  - Pyrolusite 0 4e-5

**REACTION 1**

- internal looping by REACTION data block

  - CH2O; 0.572e-3 in 100 steps

- INCREMENTAL_REACTIONS true

END
Kd’s for trace metals as a function of pH

This example (from Appelo and Postma, 2005, Fig. 11.19) also uses the REACTION keyword to generate a series of curves showing the variation of solid/solution partition coefficient (Kd) as a function of pH for U, Zn and Cd.

It uses the SURF() function to get the total number of moles of each element adsorbed to a particular mineral surface (here Hfo) and TOT() to get the total number of moles of each element remaining in solution. Cd and Zn are also bound by ion exchange reactions on kaolinite. The total bound includes both adsorbed and exchanged species so these must be added together to calculate the Kd.

The initial solution is a sample of acid mine drainage in equilibrium with a quartz-rich sediment. This is progressively neutralized with NaOH. The trace metals are bound to Hfo and kaolinite and the Kd’s reflect how binding to these two surfaces changes with pH. Cd and Zn are mostly bound to kaolinite at low pH and this is modelled as a simple pH-independent cation exchange reaction. At high pH, binding to ferrhydrite becomes important. U sorption is out-competed by other trace metals on ferrhydrite at low pH. At high pH, various negatively charged U species dominate in solution which works against their sorption at high pH.
# plots the solid/solution partition coefft (Kd) for the sorption of metals by HFO as a function of pH

**SPECIFICATION**

- **jobTitle**: "Kd's of trace metals in neutralized AMD (A&P, Fig 11.19)"
- **calculationType**: custom
- **calculationMethod**: 1  # 1 = calculate and plot
- **resolution**: 1  # just one iteration of x- and y-axis variables
- **selectedOutputLines**: auto  # auto = results will be on the last line of the selected output

**PLOT**

- **plotTitle**: "Kd's of trace metals in neutralized AMD"
- **xtitle**: pH
- **ytitle**: "Distribution coefficient, Kd"
- **customXcolumn**: pH  # from the out file
- **lines**: KdU  KdZn  KdCd  # from the out file
- **file - plot these three as lines**
- **extraText**: "extratextkd.dat"  # additional text on/by plot

**CHEMISTRY**

**SELECTED_OUTPUT**

- reset false
- high_precision true

**USER_PUNCH**

- headings pH  KdU  KdZn  KdCd  # these columns of data accumulate in the out file
- **-start**
- **-end**

**SOLUTION 1 AMD**

- **-temp**: 10
- **-units**: mmol/kgw
- **pH**: 2.3  # analysis from some Acid Mine Drainage

<table>
<thead>
<tr>
<th>Element</th>
<th>Concentration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Na</td>
<td>23.8</td>
</tr>
<tr>
<td>K</td>
<td>0.1</td>
</tr>
<tr>
<td>Mg</td>
<td>2.0</td>
</tr>
<tr>
<td>Ca</td>
<td>11.6</td>
</tr>
<tr>
<td>C</td>
<td>1.7e-4</td>
</tr>
<tr>
<td>Cl</td>
<td>13</td>
</tr>
<tr>
<td>P</td>
<td>0.08</td>
</tr>
<tr>
<td>S(6)</td>
<td>52.8</td>
</tr>
<tr>
<td>Al</td>
<td>6.5</td>
</tr>
<tr>
<td>Cd</td>
<td>0.01</td>
</tr>
<tr>
<td>Fe(3)</td>
<td>10.7</td>
</tr>
<tr>
<td>Fe(2)</td>
<td>0.27</td>
</tr>
<tr>
<td>U(6)</td>
<td>0.18</td>
</tr>
<tr>
<td>Zn</td>
<td>1.5</td>
</tr>
</tbody>
</table>

**SURFACE 1**

- **Hfo_w**: 2e-3 600 0.89
- **Hfo_s**: 5e-5
- **-equil**: 1
EXCHANGE_SPECIES
H+ + X- = HX
log_k 1.0
-gamma 9.0 0.0

EXCHANGE 1
X 50e-3
-equil 1

REACTION 1
NaOH 1
105e-3 in 100 steps
# the number of steps controls the resolution of the plot
INCREMENTAL_REACTIONS

END
Cd speciation vs pH

**Phreeqc** cannot automatically generate columns headings containing the species names. This means that it is not possible to automatically write the correct header in the ‘out’ file when writing species that are generated automatically, for example, by the *SYS()* function.

However, a custom plot needs to be able to pick up the correct label names from the header line in order to be able to label the plot properly. Communicating the species names to the plot file therefore becomes a problem. There are two ways round this: (i) put the label names explicitly (manually) in the *-headings* line of a *USER_FUCNT* block, or (ii) write them as a separate data column in the ‘out’ file, i.e. as name-value pairs.

The first approach is illustrated in this example. This requires that you know which species will be output in the first place. The *SYS()* function in the *Cdvsph.inc* file makes it easy to output all of the species involved automatically. These are output in descending amount order (largest amounts first) and so this order will change with pH. The species therefore need to be sorted. This is done with the *sort.inc* file. The species will then always be output in ascending alphabetic order (ignoring parentheses) and the *-heading* list should reflect this order. It is normally necessary to run a problem like this twice: firstly to get the species involved, and sec-
ondly to make the plot. This example also illustrates the use of nested include files. The \texttt{minimumYValueForPlotting} keyword eliminates all curves which do not rise above 5%.

The next example illustrates the second approach which is normally easier to implement.
SPECIATION
jobTitle "Cd speciation vs pH\(\lt;br\)&lt;br&gt;\(\text{\texttt{using}} \) \(\text{\texttt{explicit}} \) \(\text{\texttt{naming of species to plot}}\)"
calculationType custom
calculationMethod 1
xmin -13.0
xmax -6.0
# determines the number of points at which speciation is calculated
resolution 100

PLOT
plotTitle "Cd-Cl-H\(_2\)O speciation vs pH"
xtitle pH
ytitle "% species"
pxmax 13
# explicit naming of species - order defined in user_punchCd.inc
lines Cd(OH)\(_2\) Cd(OH)\(_3\) - Cd(OH)\(_4\) - 2 Cd+2 Cd2OH+3 \ CdCl+ CdCl2 CdCl3 - CdNO\(_3\) - CdOH+ CdOHCl
lineColor "blue"
pointSize 5.0
# use first column as defined by include files - this is pH
customXcolumn 1
# this prevents minor species being plotted
minimumYValueForPlotting 5.0
extraText "extratextCdspeciation.dat"

CHEMISTRY
include 'Cdvsph.inc' # nested includes
This example, based on Example 8 in the Phreeqc distribution, demonstrates the use of the \texttt{<x_axis>} tag to loop over a range of pH and the loop variable to loop over a range of Zn concentrations. The \texttt{logLoopVar} has been used to transform the loop variable to $10^z$. The plot shows the percentage of Zn adsorbed as a function of pH in 0.1M NaNO$_3$.

The \texttt{<x_axis>} tag and the \texttt{resolution} determine the range and step size for the x-axis variable (pH). The \texttt{USER_PUNCH} data block produces a block of selected output for each pH-Zn combination. With the default setting of \texttt{selectedOutputLines}, the last line of this block of output is copied to the 'out' file for plotting. A blank line is written to the 'out' file for each new value of the loop variable but not for each new value of the x-axis variable. The data are therefore plotted as a series of curves with a new curve after each change of the loop variable.

The normal legend or key has been suppressed by setting \texttt{legendTextSize} to zero. A new legend has been placed in the top-left corner using a line of the \texttt{extraText} file. The new 'legend' text has been placed on a series of lines using the continuation character, \texttt{\}, to concatenate lines and give the single text string required. Note that the maximum total length of the text string,
including any text enhancement tags such as \(<\text{sub}>\), is 200 characters. Labels have been used to number the curves.
# %sorption vs pH for Zn on Hfo
# Modelled after 'Example 8' from the PHREEQC example set

SPECIATION
  calculationType                      custom
  calculationMethod                    1
  # x-axis (pH) range
  xmin                                 5.0
  xmax                                 8.0
  # z-loop (log ZnT), one curve for each ZnT
  loopMin                              -6.0
  # from -6 to -2 in steps of +1
  loopMax                              -2.0
  loopInt                              1.0
  # 1 = value of loop variable is exponentiated (10^<loop>) before use
  loopLogVar                           1
  # number of calculations (PHREEQC simulations) for each curve
  resolution                           100

PLOT
  plotTitle                            "%sorbed vs pH curves<br>(split into two
PHREEQC simulations)"
  xtitle                               pH
  ytitle                               "%sorbed"
  # this variable in the 'out' file is plotted as a line (%sorbed is a valid column
header)
  lines                                %sorbed
  lineWidth                            0.4
  changeColor                          T
  # used in order for label names on the plots
  labels                               1 2 3 4 5
  labelSize                            2.0
  legendTextSize                       0.0
  customXcolumn                        pH
  # adds customised legend text
  extraText                            "extratextpcsorption.dat"

CHEMISTRY

# simulation 1 - initial surface calculation is run but no selected output is pro-
#duced or read
TITLE Example 8.--Sorption of zinc on hydrous iron oxides.
SURFACE_SPECIES
  Hfo_sOH + H+ = Hfo_sOH2+
  log_k  7.18
  Hfo_sOH = Hfo_sO- + H+
  log_k  -9.82
  Hfo_sOH + Zn2+ = Hfo_sOZn+ + H+
  log_k  0.66
  Hfo_wOH + H+ = Hfo_wOH2+
  log_k  7.18
  Hfo_wOH = Hfo_wO- + H+
  log_k  -8.82
  Hfo_wOH + Zn2+ = Hfo_wOZn+ + H+
  log_k  -2.32
SURFACE 1
  Hfo_sOH 5e-6  600.  0.09
  Hfo_wOH 2e-4

PHASES
  Fix_H+
  H+ = H+
  log_k  0.0

# first simulation
USE solution none

SELECTED_OUTPUT
-reset false
USER_PUNCH
# determines column headers in the ‘out’ file
-heading pH %sorbed sorbed
10 sorbed = SURF("Zn","Hfo")
20 totZn = SYS("Zn")
30 pcsorbed = 100*sorbed/totZn
40 punch -la("H+"), pcsorbed, sorbed
END

# simulation 2 - loops on this simulation to produce the output required for graphing

USE surface 1
SOLUTION 1
-units mol/kgw
pH 8.0
# ZnT
Zn <loop>
Na 0.1
N(5) 0.1 charge
EQUILIBRIUM_PHASES 1
# fixes the pH
Fix_H+ -<x_axis> NaOH 10.0
-force_equality true
END
This example shows the surface speciation for Zn adsorbed to Hfo in the same system as that of the previous example. Curves are produced for total Zn concentrations of $10^{-7}$ M and $10^{-4}$ M. Adsorbed speciation is calculated by *PUNCHing* the log concentrations of the adsorbed species directly. A similar plot could also be made using the 'species plot' procedure (see the demo\example8 directory) with the logadsspeciesvsph.inc include file.

Since there are two loops for each species, the labelling appends an underscore and the loop number to the species name to help to differentiate between the curves.

*pxmajor* has been set to one since the auto setting would produce major tick marks (and axis labels) at every 0.5 pH unit.

The legend has been suppressed by setting the legend text size to 0 and the colour to ‘nd’ in the *<legends>* line of the extraText file. It could also have omitted by setting the *legendTextSize* to 0.

*changeColor* is by default false and *useLineColorDictionary* has been set to 0 (the default) so that the default colour sequence is automatically used starting at red2, blue2 etc as given by
their respective positions in the \texttt{lineColor} list in the input file. On the second loop, the colours are kept the same but the density is increased to 4, e.g. \texttt{red4}, \texttt{blue4}, ... .
SPECIFICATION

calculationType    custom
calculationMethod   1
xmin               5.0
xmax               8.0
# minimum value of <loop> tag
loopMin            -7.0
# maximum value of <loop> tag
loopMax            -4.0
# increment of <loop> tag per iteration
loopInt            3.0
# 1 = antilog loop value, ie <loop> = 10^<loop>
loopLogVar         1
resolution         100

PLOT

plotTitle          "Speciation of zinc sorbed on HFO"
xtitle             pH
ytitle             "log conc (mol/kgw)"
pxmajor            1.0
customXcolumn      pH
# lines to plot from out file - headings defined below
lines              Zn+2 Hfo_sOZn+ Hfo_wOZn+
lineWidth          0.6
# starting colours and colour densities
lineColor          red2 blue2 green2
labelSize          1.8
trackSymbolSize    2.0

CHEMISTRY

# Similar to PHREEQC Example 8
TITLE Example 8.--Sorption of zinc on hydrous iron oxides.
# <loop> iterates on all simulations - this is the outer loop

SELECTED_OUTPUT

- reset false
- high_precision true

SURFACE_SPECIES

Hfo_sOH + H+ = Hfo_sOH2+
log_k    7.18
Hfo_sOH   = Hfo_sO- + H+
log_k    -8.82
Hfo_sOH + Zn+2 = Hfo_sOZn+ + H+
log_k    0.66
Hfo_wOH   + H+ = Hfo_wOH2+
log_k    7.18
Hfo_wOH = Hfo_wO- + H+
log_k    -8.82
Hfo_wOH + Zn+2 = Hfo_wOZn+ + H+
log_k    -2.32

SURFACE 1

Hfo_sOH         5e-6    600.    0.09
Hfo_wOH         2e-4

PHASES

Fix_H+
H+ = H+
log_k    0.0
# Initial setup simulation

SOLUTION 1

- units mol/kgw
# only for the initial solution calculation
pH      8.0
Zn      <loop>
# background electrolyte
Na 0.1 charge
N(5) 0.1

USER_PUNCH
-heading pH Zn+2 HfO_wOZn+ HfO_sOZn+
10 punch -la("H+"), lm("Zn+2"), lm("HfO_wOZn+"), lm("HfO_sOZn+")
END

# <x_axis> iterations only execute the last iteration by default
USE solution 1
USE surface 1
EQUILIBRIUM_PHASES 1
   Fix_H+ -<x_axis> NaOH 10.0
END
As-HFO: reduction in surface area

This example shows how the dissolved As concentration could evolve as the surface area of HFO declines (ageing). The example demonstrates the use of user-defined tags to pass information from one simulation to the next. An alternative approach involves using the $\text{PUT()}$ and $\text{GET()}$ BASIC functions.

The total amount of As is defined by the first simulation and then the adsorbed As (and any adsorbed P) is redistributed in the second simulation assuming closed conditions (apart from $\text{H}^+$). The dissolved As (and P) from the first simulation is discarded and the adsorbed As redistributed as the surface area decreases. It is assumed that while the surface area of the HFO decreases, the surface properties of the HFO remain unchanged (unlikely to be true in practice).

The various tag definitions in $\text{numericTags}$ calculate the number of sites at any particular stage based on the given initial surface characteristics. These values are substituted in the $\text{Phreeqc}$ code during each iteration.

The $\text{PhreePlot}$ looping is only over the second (final) simulation.
# calculates how the solution concn of As changes as the surface area of Hfo
# (but not the surface properties) is reduced in a closed system.

SPECIATION
JobTitle                       "Diagenesis"
calculationType              custom
calculationMethod           1
xmin                           10.0 # minimum surface area, see below
xmax                           600.0 # maximum surface area
resolution                     100
numericTags                   
 &lt;mass&gt;  = 1 \
 &lt;molecular_wt&gt; = 89 \
 &lt;initial_site_density_per_mol&gt; = 0.2 \
 &lt;initial_surface_area&gt; = 600 \
 &lt;initial_site_density_per_g&gt; = \
 &lt;initial_site_density_per_mol_per_label&gt;/&lt;molecular_wt&gt; \ 
 &lt;initial_sites&gt; = \
 &lt;initial_site_density_per_g&gt;/&lt;initial_surface_area&gt; \
 &lt;surface_area&gt; = &lt;x_axis&gt; \
 &lt;sites&gt; = \
 &lt;surface_area&gt;*&lt;site_density_per_m2&gt;*&lt;mass&gt;

PLOT
plotTitle                    "As desorption as the surface area \ decreases"
xtitle                      "Surface area (m² g⁻¹)"
ytitle                      "As (µg L⁻¹)"
pymax                        5000.0 # truncate the highest values
customxColumn               surface_area
lines                        As
lineWidth                   0.6
lineColor                   red
legendTextSize              0.0
extraText                   "extratextsurfacearea.dat"

CHEMISTRY
PRINT
# -reset false
PHASES
Fix_H+
  H+ = H+
  log_k 0.0

SELECTED_OUTPUT
 -high_precision   true
 -reset            false

USER_PUNCH
 -headings   totAs totP
 -start
10 totAs=SYS("As",n,n$,t$,c)
20 totP=SYS("P",n,n$,t$,c)
30 punch totAs, totP
 -end

# first simulation - set up initial conditions
SOLUTION 1
  temp      25
  pH        7.0
  units     mol/kgw
  density   1
  Na        le-2
  N(5)      le-2
# Equilibrate Hfo with low As and P
As        1 ug/kgw
P         0 ug/kgw # P has an important effect
 -water    1 kg
EQUILIBRIUM_PHASES 1
    Fix_H+ -7.0 NaOH 10
    -force_equality true
O2(g) -0.67 10

SURFACE 1
    Hfo_w &lt;initial_sites&gt; &lt;initial_surface_area&gt; &lt;mass&gt;
    -equilibrate 1
END

# second simulation - now start reducing surface area always starting from the \
    initial state

USER_PUNCH
    -headings surface_area As
    -start
10 As=tot("As")*74.9216*1e6
20 punch &lt;surface_area&gt; As
    -end

SOLUTION 1
    temp 25
    pH 7.0
    units mol/kgw
    density 1
    Na 1e-2
    N(5) 1e-2
    As &lt;totAs&gt; # tag name from selected output file headings above
    P &lt;totP&gt; # mol/kgw
    -water 1 # kg

EQUILIBRIUM_PHASES 1
    Fix_H+ -7.0 NaOH 10 # keep a constant pH
O2(g) -0.67 10

SURFACE 1
    Hfo_w &lt;sites&gt; &lt;surface_area&gt; &lt;mass&gt;
END
CD-MUSIC: As(III) adsorption on goethite

This example shows the calculated concentration of As(III) remaining in solution after adsorption of As(III) on goethite (98 m²/g) as a function of pH. Calculations are based on the CD-MUSIC model and parameters of Stachowicz et al. (2006) and reproduces their Figure 4. As(III) loadings were varied by varying the solid/solution ratio and the initial As(III) concentration.

Thermodynamic data for the aqueous As species are retrieved from the ecosat.inc include file. The arsenic species in the default database have been removed from consideration by defining all As(III) reactions in terms of a new element, [As₃]. The CD-MUSIC model is defined in the cdmusic.inc include file. Many parameter values are set with tags in the main input file for convenience.

**pxmin** and **pxmajor** override the default x-axis scaling which would give an x-axis ranging from 2 to 12 with labelling every 2 units. **pxmin** forces the x-axis to start at 3 while **pxmajor** forces the axis labelling to be every 3 pH units. Similarly **pymin** and **pymax** force the y-axis scaling to the desired range.

The label names are derived from the loop names which themselves are defined in column 1 of the loopfile. **changeColor** has been set to **TRUE** to ensure that the different curves of the same data column have different colours.
SPECIATION
    calculationType              custom
    calculationMethod            1
    xmin                          3.0
    xmax                          12.0
    resolution                   100
    loopFile                     "loopfig4.dat"
    numericTags                  <mass>       = <loop1> \\
                                 <AsT>        = <loop2>

PLOT
    plotTitle                    "CD-MUSIC: As(III) remaining in solution<br>(after Stachowicz et al., 2006, Fig. 4)"
    xtitle                       pH
    ytitle                       "As(III) in solution, log (mol/L)"
    pxmin                        3
    pxmajor                      3
    pymin                        -7
    pymax                        -3
    customxcolumn                pH
    lines                        As3
    changeColor                  T
    labelsize                    1.5
    extratext                    "extratextfig4.dat"

CHEMISTRY

include 'cdmusic_hiemstra.dat'

SELECTED_OUTPUT
    -reset false

SOLUTION 1
    Temp       25
    pH         2.9
    units      mol/kgw
    # total As
    As(3)      <AsT> mmol/kgw
    # background electrolyte
    Na         1e-1
    # N(5) is not thermodynamically stable with As(3)
    [N5]        1e-1

USER_PUNCH
    -headings pH As3
10 PUNCH -la("H+"), log10(tot("As"))

PHASES ; Fix_H+; H+ = H+ ; log_k 0

SURFACE 1
    # sites/nm2  m2/g  g
    Goe_uniOHH0.5 3.45 98 <mass>
    # C1  C2 (in P/m2)
    -cap  0.85  0.75
    Goe_trioH0.5  2.7
    -cd_music
    -sites_units density

EQUILIBRIUM_PHASES 1
    # to ensure all As(3)
    O2(g)     -70
    Fix_H+    -<x_axis> NaOH
    -force_equality true

END
65  CD-MUSIC: As(V) adsorption on goethite

This is similar to the previous example except it is for the adsorption of As(V) rather than As(III). The figure shows the calculated amount of As(V) remaining in solution after adsorption on goethite (98 m$^2$/g) as a function of pH. The calculated curves were based on the CD-MUSIC model and the parameters of Stachowicz et al. (2006). This figure replicates the calculated curves of their Fig. 6.
SPECIFICATION

calculationType                      custom
calculationMethod                    1
xmin                                 3.0
xmax                                 12.0
resolution                           100

# defines <loop1> and <loop2> tags for mass and AsT
loopfile                             "loopfig6.dat"
numericTags                          <mass> = <loop1> \ 
<AsT> = <loop2>

PLOT
plotTitle                            "CD-MUSIC: As(V) remaining in solution (3 species)<br>(after Stachowicz et al., 2006, Fig. 6)"
xtitle                               pH
ytitle                               "As(V) in solution, log (mol/L)"
# plot limits
pxmin                                3
pxmax                                12
pymin                                -7
pymax                                -3

# column name from selected output file: see below
customxcolumn                        pH
# ibid
lines                                As5
changeColor                          T
# removes legend [key] from plot
legendTextSize                       0
# additional text for plot
extratext                            "extratextfig6.dat"

CHEMISTRY

SELECTED_OUTPUT

-SOLUTION 1
Temp      25
pH        3
units     mol/kgw
As(5)     <AsT> mmol/kgw
Na        1e-1
N(5)      1e-1

# the results here differ slightly from the published ones due to small differences
in the databases used
# note particularly the sensitivity of the 3 g/L curve at low pH
include ‘cdmusic_hiemstra.dat’

USER_PUNCH

-PHASES ; Fix_H+; H+ = H+ ; log_k 0

SURFACE 1

# sites/nm2  m2/g  g
Goe_uniOHH0.5  3.45  98 <mass>

EQUILIBRIUM_PHASES 1

Fix_H+  -<x_axis> NaOH
-<x_axis> O2(g) -0.67
This example shows the kinetics of the oxidation of pyrite from Appelo and Postma (2005) p 455-456, Fig. 9.28 (the calculated O$_2$(g) curve here does not quite agree with A&P’s because of small changes in the Phreeqc.dat database used).

The resolution only has to be set to 1 since the KINETICS data block has an internal looping mechanism (like REACTION) which produces a multi-lined selected output ‘file’. selectedOutputLines has therefore been set to auto so that all the lines in the selected output are picked up.

The calculated points are plotted as a continuous curve using the lines keyword with two variable (column) names, namely O$_2$(g) and CO$_2$(g). These names are defined in the USER_PUNCH headings line and are automatically passed through to the ‘out’ file which is then used for the plotting.

The data points are plotted using an extra file. legendTextSize has been set to 0 to eliminate the legend. The Times-Roman font has been selected with the font keyword.

The x-axis scaling and title includes a scaling factor (x10$^2$) which indicates that the true scale actually varies from 0–1400 hours.
SPECIFICATION
jobTitle "Pyrite oxidation kinetics, A&P (2005) p 455-6"
database phreeqc.dat
calculationType custom
calculationMethod 1
# only need to do one calculation as KINETICS block has its own looping
resolution 1
# multiline selected.out so copy all the lines produced into the out file
selectedOutputLines auto

PLOT
plotTitle "Pyrite oxidation kinetics\n(Appelo and Postma, 2005)"
xtitle "Time (\" hours)"
ytitle \"P\"_{O2(g)} or \"P\"_{CO2(g)} (%)"
# fix upper limit of y-axis
pymax 9
# omit legend
legendTextSize 0.0
# from out file
customXcolumn Time
# modelled results - labels from USER_PUNCH block -> out file
lines CO2(g) O2(g)
# plot experimental data points
extraSymbolsLines "pyritekineticsdata.dat"

CHEMISTRY
SELECTED_OUTPUT
-reset false
-high_precision true

USER_PUNCH
# defines column heading in out file
-headings Time CO2(g) O2(g)
-start
10 PUNCH total_time/3600
20 PUNCH 100*10^{si("CO2(g)"), 100*10^{si("O2(g)")}
-end

RATES
# the kinetic model
Pyrite
-start
1 A=15e3*m0
10 if SI("Pyrite")>0 then goto 100
20 fH=mol("H+")
30 fFe2=(1+tot("Fe(2)")/1e-6)
40 if mol("O2")<1e-6 then goto 80
# rate with oxygen
50 rO2=10^{-8.19}*mol("O2")*fH^{-0.11}
60 rO2_Fe3=6.3e-4*tot("Fe(3)")^{0.92}*fFe2^{-0.43}
70 goto 90
80 rem
# rate without oxygen
81 rFe3=1.9e-6*tot("Fe(3)")^{0.28}*fPe2^{-0.52}*fH^{-0.3}
90 rate=A*(m/m0)^{0.67}*(rO2+rO2_Fe3+rFe3)*(1-SR("Pyrite"))
# must include this
100 save rate*time
-end

SOLUTION_SPECIES
# force one way - dissolved N2(g) does not make nitric acid!
2NO3^- + 12H^+ + 10e^- = N2 +6H2O; log_K 500

SOLUTION 1
# kg
-water 0.0069239
-temp 20
pH 7 charge; pe 14 O2(g) -1.0878
Ca 1 Calcite; C 1 CO2(g) -2.6021
Fe 1e-3 Goethite 2; N 1.3 N2(g) -0.0382

EQUILIBRIUM_PHASES
   Goethite 2; Calcite 0; Gypsum 0 0

GAS_PHASE 1
   -fixed_pressure
   -volume 0.02127; temp 20
   CO2(g) 0.0025; O2(g) 0.0817; N2(g) 0.9157

KINETICS 1
   Pyrite; -m0 1.32e-4; -step 0 5e5 5e5 5e5 5e5 5e5 5e5 5e5 5e5 5e5 5e5
   INCREMENTAL_REACTIONS true

END
This example demonstrates how a single iteration of the KINETICS data block generates a multiline selected output file which is read in with selectedOutputLines set to auto.

The label in the plot and the key to the right of the plot have been suppressed with labelSize and legendTextSize both set to 0.

The extraText file also includes the use of various text enhancements – italics, subscripts, superscripts and a Greek character.
Examples 485

# plot of the dissolution of quartz vs time based on the PHREEQC kinetics model

SPECIFICATION

calculationType          custom
calculationMethod         1
selectedOutputLines      auto

PLOT

plotTitle                   "Kinetics of quartz dissolution<br>(from Appelo 'Get-going' sheet 11)"
xtitle                      "Time (year)"
ytitle                      "Si (mmol/kgw)"
# plot x = time
customXColumn               time
# plot y(line) = Si
lines                       Si
linecolor                   blue
linewidth                   0.6
# suppresses label
labelSize                   0
# suppresses key
legendTextSize              0
numericTags                 \<log_k> = -13.7
# additional text (including some Greek symbols)
extraText                   extratextkinetics.dat

CHEMISTRY

# Kinetics of quartz dissolution
# from Appelo 'Get-going sheet #11'

RATES

#1  dQu/dt = -k * (1 - SR(Quartz)). k = 10^-13.7 mol/m2/s (25 C)
#2  parm(1) = A (m2), parm(2) = V (dm3) recalculate to mol/dm3/s

# rate name
Quartz
-start
10 moles = parm(1) / parm(2) * (m/m0)^0.67 * 10^<log_k> * (1 - SR("Quartz"))
# integrate. save and time must be in rate definition
20 save moles * time
# moles count positive when added to solution
-end

# Sediment: 100% qu, grain size 0.1 mm, por 0.3, rho_qu 2.65 kg/dm3

KINETICS

# rate name
Quartz
-Formula SiO2
# initial moles of quartz
-m0 102.7
# parameters for rate eqn. Here:
-parms 22.7 0.162
# Quartz surface area (m2/kg sediment), water filled porosity (dm3/kg sediment)

# 1.5768e8 seconds = 5 years
-step 1.5768e8 in 100 steps
# integration tolerance, default 1e-8 mol
-tol 1e-8

# start integration from previous step
INCREMENTAL_REACTIONS true

SOLUTION 1

SELECTED_OUTPUT

-user_punch
# these are accumulated in the out file ready for plotting
-heading time Si SIQtz
-start
10 IF (step_no>0) THEN punch total_time/3.1536e7, tot("Si")*1e3, SI("Quartz")
-end

END
This example does no geochemistry but simply plots a grid of symbols with their codes. The symbols are defined on an invisible grid in the `extraSymbolsLines` file called `symbols.dat` while the code numbers printed below are defined in the `extraText` file called `extratextsymbols.dat`.

The surrounding box is drawn with four gray lines specified at the end of the `extra` file. `axis-NumberSize` and `axis` have been set to 0 to suppress the plotting of the axis numbering and the axis lines.
SPECIATION
  calculationType: custom
  calculationMethod: 1

PLOT
  # make bigger
  plotfactor: 2
  xaxislength: 90
  yaxislength: 110
  units: "mm"
  plotTitle: "Symbols, dingbats and lines"
  # it should not look like an x-y plot
  xtitle: ""
  ytitle: ""
  # plot area large enough not clipped gray border
  pxmin: -2
  pxmax: 17
  pxmajor: 1
  pxminor: 1
  pymin: -100
  pymax: 120
  pymajor: 10
  pymminor: 5
  xoffset: 7.5
  yoffset: 10
  tickSize: 0
  axisNumberSize: 0
  axisLineWidth: 0
  # contains the list of symbols and their positions etc
  extraSymbolsLines: "symbols.dat"
  # number the positions
  extraText: "extratextsymbols.dat"
  # gridlines
  t
  # axislinewidth
  0.05
This example demonstrates how different lines and sets of points of custom plots can have different attributes associated with them. Some acidic, Al-F-containing water is titrated with Ca(HCO₃)₂. Eight variables are punched to the selected output. pH is used as the x-axis and the other variables are plotted on the y- and 2y-axes according to the lists given in points, lines and lines2y. Attributes are pulled off the associated lists (pointColor, pointType, pointSize etc.) until exhausted in which case the list is automatically extended (colours) or recycled (other attributes). There is a default sequence of 15 colours (red, blue, green, orange, cyan, magenta, brown, sky, purple, gray, yellow, maroon, lawn, spring, black) for each type of list (points and lines, y and 2y axes). These lists are rearranged according to the individual input colour lists which effectively promote their members to the top of the respective sequence. If there are more than one subsets of data, the colour density is cycled (not here).

The attributes chosen for a particular dataset depend on the position of that dataset in the input list. For example, Alk is third in the points list so will pick up the third pointColor, third pointType, third pointSize, and so on. Symbol types are defined either by number (see the preceding Example) or name (see Appendix 3). The first six symbols can have a separate rimColor. The width of the rim is defined in terms of the fraction of the overall symbol width (rimFactor). Dashed lines are signified by negative line widths and here are on the 2y axis (*).
This is the basic method of assigning attributes. Colour selection can be modified from this by using `changeColor`, `pointsSameColor` and `restartColorSequence`. The colours and other attributes can also be set using the line colour dictionary and ensuring its use with `lineColor-Dictionary` 1 or 2.
# demonstrates the use of different symbols, colours and line types in a plot.

# For reference, the 15 auto color sequence is:
# red, blue, green, orange, cyan, magenta, brown, sky,
# purple, gray, yellow, maroon, lawn, spring, black

SPECIATION
calculationMethod 1
calculationType custom
xmin 5
xmax 8
resolution 20

PLOT
xtitle pH
ytitle "concentration (mmol kgw<sup>-1</sup>)"
2ytitle "Saturation index"
# this is used as the x-axis
customXcolumn pH
p2ymax 1
# these will be plotted as points
points Ca(HCO3)2 Al<sub>T</sub> Alk Ca
# these will be plotted as lines
lines Ca(HCO3)2 Al<sub>T</sub> Alk Ca
lines2y SI<sub>calcite</sub> SI<sub>fluorite</sub> SI<sub>Al(OH)3(a)</sub>
# symbols to use in sequence for 'points' datasets, recycled
pointType 1 2 6
# colors to use in sequence for 'points'
pointColor gray2 green2 red2 sky2
# size of symbols, recycled
pointSize 3
# explicit rim color for first 2 datasets then recycle
rimColor black green6
# fractional rim width, recycled
rimFactor 0.08
# colors to use in sequence for 'lines'
lineColor black green6 red blue
# colors to use in sequence for 'lines2y' then auto sequence
lineColor2y purple brown6
# line width to use for 'lines2y', negative for dashed, recycled
lineWidth2y -0.4
changeColor f

CHEMISTRY

# titrate acidic Al-rich water with Ca(HCO3)2

PHASES
Fixed_H+ H+ = H+
log_k 0.

SOLUTION
-units mmol/kgw
pH 4.5
Al1
P2
C13 charge
SAVE solution 1
END

USE solution 1
EQUILIBRIUM_PHASES
Fixed_H+ -<x_axis> Ca(HCO3)2
Al(OH)3(a) 0 0
Fluorite 0 0
Calcite 0 0
CO2(g) -3.5
SELECTED_OUTPUT
-reset FALSE

USER_PUNCH
-headings Ca(HCO3)2 pH Al<T> Alk Ca Si<sub>calcite</sub> Si<sub>fluorite</sub> Si<sub>Al(OH)3(a)</sub>
10 IF (STEP_NO = 1) THEN PUNCH (10-EQUI("Fixed_H+"))/TOT("water")*1e3, -la("H+"), TOT("Al")*1e3, \ Alk*1e3, TOT("Ca")*1e3, SI("Calcite"), SI("Fluorite"), SI("Al(OH)3(a")
END
This example also does no geochemistry but demonstrates some of the capabilities of justifying and rotating text as well as enhancing it with subscripts and superscripts and Greek characters and text. These are all defined in the `extratext.dat` file.

The light yellow background colour has been set with the `backgroundColor` setting. It only occupies the 'plot' area.
SPECIATION
   calculationType          custom
   calculationMethod        1

PLOT
   backgroundColor        "yellow0"
   plotTitle              "<b>Text: demonstrating <i>justification</i>,
                           <i>orientation</i> and <i>character sets</i></b>"
   # not supposed to look like an x-y plot
   xtitle                 ""
   ytitle                 ""
   xoffset                60
   pxmin                  -2.0
   pxmax                  12.0
   p xmax                  1.0
   pymin                  -20.0
   pymajor                100.0
   tickSize               0.0
   axisNumberSize         2
   axisLineWidth          0.05
   pointSize              0.0
   # file containing special text
   extraText              "extratexttext.dat"
   gridlines T
   gridlinecolor gray4

   font Bookman
   # font Latin-1

CHEMISTRY
# Simple looping on one variable, log (H+) activity
# Fe speciation at pH 4, 7, 10

SPECIATION
  calculationType        "custom"
  xmin                   -10
  xmax                   -4
  resolution            3  # i.e. pH 10, 7, 4
  all                   t  # writes the *.all file
  selectedOutputLines   1 0 0  # no SELECTED_OUTPUT expected

CHEMISTRY

PRINT
  -reset FALSE       # don't output initial solution calculation

PHASES

Fix_H+
  H+ = H+
  log_k 0

SOLUTION 1

pH     2
units  mol/kgw
Fe(3)  1e-6
Na     1e-2
Cl     1e-2

# no reaction so no need to SAVE solution 1

END

USE solution 1

PRINT
  -equilibrium_phases true
  -species TRUE

EQUILIBRIUM_PHASES

  Fe(OH)3(a) 0 0
  Fix_H+ <x_axis> NaOH

END

and the output from the three iterations of Phreeqc are accumulated in *.all which, when using the wateq4f.dat database looks like this:

--- Simulation 1 ---

--- Phase assemblage ---

<table>
<thead>
<tr>
<th>Phase</th>
<th>SI</th>
<th>log IAP</th>
<th>log KT</th>
<th>Initial</th>
<th>Final</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fe(OH)3(a)</td>
<td>-0.00</td>
<td>17.91</td>
<td>17.91</td>
<td>0.000e+000</td>
<td>7.572e-007</td>
<td>7.572e-007</td>
</tr>
<tr>
<td>Fix_H+</td>
<td>-10.00</td>
<td>-10.00</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Distribution of species

<table>
<thead>
<tr>
<th>Species</th>
<th>Molality</th>
<th>Activity</th>
<th>Log Molality</th>
<th>Activity</th>
<th>Gamma</th>
</tr>
</thead>
<tbody>
<tr>
<td>OH-</td>
<td>1.134e-004</td>
<td>1.001e-004</td>
<td>-3.945</td>
<td>-4.000</td>
<td>-0.054</td>
</tr>
<tr>
<td>H+</td>
<td>1.133e-007</td>
<td>1.000e-007</td>
<td>-6.946</td>
<td>-7.000</td>
<td>-0.054</td>
</tr>
<tr>
<td>H2O</td>
<td>5.551e+001</td>
<td>9.995e-001</td>
<td>1.744</td>
<td>-0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Cl-</td>
<td>9.998e-003</td>
<td>8.798e-003</td>
<td>-2.000</td>
<td>-2.056</td>
<td>-0.056</td>
</tr>
<tr>
<td>FeCl+</td>
<td>4.580e-008</td>
<td>4.042e-008</td>
<td>-17.339</td>
<td>-17.393</td>
<td>-0.054</td>
</tr>
<tr>
<td>FeCl2+</td>
<td>3.414e-006</td>
<td>2.071e-006</td>
<td>-25.467</td>
<td>-25.684</td>
<td>-0.217</td>
</tr>
<tr>
<td>FeCl3</td>
<td>9.221e-005</td>
<td>8.138e-005</td>
<td>-27.035</td>
<td>-27.090</td>
<td>-0.054</td>
</tr>
<tr>
<td>Fe(OH)+</td>
<td>1.192e-005</td>
<td>1.052e-005</td>
<td>-14.924</td>
<td>-14.978</td>
<td>-0.054</td>
</tr>
<tr>
<td>Fe+2</td>
<td>5.487e-007</td>
<td>3.328e-007</td>
<td>-15.261</td>
<td>-15.478</td>
<td>-0.217</td>
</tr>
<tr>
<td>Fe(OH)2-</td>
<td>8.916e-006</td>
<td>8.949e-006</td>
<td>-16.050</td>
<td>-16.048</td>
<td>0.002</td>
</tr>
<tr>
<td>FeOH+</td>
<td>1.192e-006</td>
<td>1.052e-006</td>
<td>-14.924</td>
<td>-14.978</td>
<td>-0.054</td>
</tr>
<tr>
<td>Fe+3</td>
<td>2.400e-008</td>
<td>7.793e-009</td>
<td>-15.621</td>
<td>-16.108</td>
<td>-0.488</td>
</tr>
<tr>
<td>FeCl2+</td>
<td>3.414e-006</td>
<td>2.071e-006</td>
<td>-25.467</td>
<td>-25.684</td>
<td>-0.217</td>
</tr>
<tr>
<td>FeCl3</td>
<td>9.221e-005</td>
<td>8.138e-005</td>
<td>-27.035</td>
<td>-27.090</td>
<td>-0.054</td>
</tr>
<tr>
<td>Fe2(OH)2+</td>
<td>5.027e-010</td>
<td>6.807e-010</td>
<td>-32.299</td>
<td>-33.167</td>
<td>-0.868</td>
</tr>
<tr>
<td>Fe3(OH)4+</td>
<td>0.000e+000</td>
<td>0.000e+000</td>
<td>-40.269</td>
<td>-41.626</td>
<td>-1.353</td>
</tr>
</tbody>
</table>

### Phase assemblage

<table>
<thead>
<tr>
<th>Phase</th>
<th>SI log IAP</th>
<th>log KT</th>
<th>Initial</th>
<th>Final</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fe(OH)3(a)</td>
<td>0.00</td>
<td>17.91</td>
<td>17.91</td>
<td>0.000e+000</td>
<td>9.596e-007</td>
</tr>
<tr>
<td>Fe(OH)3</td>
<td>4.043e-008</td>
<td>2.135e-008</td>
<td>2.134e-008</td>
<td>-7.671</td>
<td>-7.669</td>
</tr>
<tr>
<td>Fe(OH)2-</td>
<td>3.410e-006</td>
<td>2.071e-006</td>
<td>-16.467</td>
<td>-16.684</td>
<td>-0.216</td>
</tr>
<tr>
<td>Fe+2</td>
<td>5.487e-007</td>
<td>3.328e-007</td>
<td>-15.261</td>
<td>-15.478</td>
<td>-0.217</td>
</tr>
<tr>
<td>Fe(OH)2-</td>
<td>8.916e-006</td>
<td>8.949e-006</td>
<td>-16.050</td>
<td>-16.048</td>
<td>0.002</td>
</tr>
<tr>
<td>FeOH+</td>
<td>1.192e-005</td>
<td>1.052e-005</td>
<td>-14.924</td>
<td>-14.978</td>
<td>-0.054</td>
</tr>
<tr>
<td>Fe+3</td>
<td>2.400e-008</td>
<td>7.793e-009</td>
<td>-15.621</td>
<td>-16.108</td>
<td>-0.488</td>
</tr>
<tr>
<td>FeCl2+</td>
<td>3.414e-006</td>
<td>2.071e-006</td>
<td>-25.467</td>
<td>-25.684</td>
<td>-0.217</td>
</tr>
<tr>
<td>FeCl3</td>
<td>9.221e-005</td>
<td>8.138e-005</td>
<td>-27.035</td>
<td>-27.090</td>
<td>-0.054</td>
</tr>
<tr>
<td>Fe2(OH)2+</td>
<td>5.027e-010</td>
<td>6.807e-010</td>
<td>-32.299</td>
<td>-33.167</td>
<td>-0.868</td>
</tr>
<tr>
<td>Fe3(OH)4+</td>
<td>0.000e+000</td>
<td>0.000e+000</td>
<td>-40.269</td>
<td>-41.626</td>
<td>-1.353</td>
</tr>
</tbody>
</table>

### Summary

- **NaOH** is reactant: $1.000e+001$ 9.989e+000-1.114e-002
- **Simulation 2**
--- Simulation 3 ---

### Phase assemblage

<table>
<thead>
<tr>
<th>Phase</th>
<th>SI</th>
<th>log IAP</th>
<th>log KT</th>
<th>Initial</th>
<th>Final</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fe(OH)₃(a)</td>
<td>-1.44</td>
<td>16.47</td>
<td>17.91</td>
<td>0.000e+000</td>
<td>0.000e+000</td>
<td></td>
</tr>
<tr>
<td>Fe₂⁺</td>
<td>-4.00</td>
<td>-4.00</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NaOH is reactant**

1.000e+01 9.989e+000-1.102e-002

### Distribution of species

<table>
<thead>
<tr>
<th>Species</th>
<th>Molality</th>
<th>Activity</th>
<th>Log Molality</th>
<th>Log Activity</th>
<th>Gamma</th>
</tr>
</thead>
<tbody>
<tr>
<td>H⁺</td>
<td>1.113e-004</td>
<td>1.000e-004</td>
<td>-3.954</td>
<td>-4.000</td>
<td>-0.046</td>
</tr>
<tr>
<td>OH⁻</td>
<td>1.133e-010</td>
<td>1.001e-010</td>
<td>-9.946</td>
<td>-10.000</td>
<td>-0.054</td>
</tr>
<tr>
<td>H₂O</td>
<td>5.551e+001</td>
<td>9.995e-001</td>
<td>1.744</td>
<td>-0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Cl⁻</td>
<td>9.998e-003</td>
<td>8.802e-003</td>
<td>-2.000</td>
<td>-2.055</td>
<td>-0.055</td>
</tr>
<tr>
<td>FeCl⁺₂</td>
<td>1.243e-009</td>
<td>7.551e-010</td>
<td>-8.906</td>
<td>-9.122</td>
<td>-0.216</td>
</tr>
<tr>
<td>FeCl⁺</td>
<td>3.363e-011</td>
<td>2.969e-011</td>
<td>-10.473</td>
<td>-10.527</td>
<td>-0.054</td>
</tr>
<tr>
<td>FeCl₂⁺</td>
<td>5.990e-013</td>
<td>5.288e-013</td>
<td>-12.223</td>
<td>-12.277</td>
<td>-0.054</td>
</tr>
<tr>
<td>FeCl⁻</td>
<td>2.604e-014</td>
<td>2.613e-014</td>
<td>-13.584</td>
<td>-13.583</td>
<td>0.002</td>
</tr>
<tr>
<td>FeCl⁺₂</td>
<td>7.224e-011</td>
<td>7.551e-010</td>
<td>-8.906</td>
<td>-9.122</td>
<td>-0.216</td>
</tr>
<tr>
<td>FeCl⁻</td>
<td>5.990e-013</td>
<td>5.288e-013</td>
<td>-12.223</td>
<td>-12.277</td>
<td>-0.054</td>
</tr>
<tr>
<td>FeOH⁺</td>
<td>1.166e-013</td>
<td>1.170e-013</td>
<td>-22.933</td>
<td>-22.932</td>
<td>0.002</td>
</tr>
<tr>
<td>Fe(OH)₂⁻</td>
<td>4.922e-030</td>
<td>4.346e-030</td>
<td>-29.308</td>
<td>-29.362</td>
<td>-0.054</td>
</tr>
<tr>
<td>Fe₂(OH)₂⁺</td>
<td>9.997e-007</td>
<td>8.607e-007</td>
<td>-6.163</td>
<td>-6.217</td>
<td>-0.054</td>
</tr>
<tr>
<td>Fe(OH)⁺</td>
<td>6.432e-007</td>
<td>6.067e-007</td>
<td>-6.163</td>
<td>-6.217</td>
<td>-0.054</td>
</tr>
<tr>
<td>Fe(OH)₂⁺</td>
<td>3.017e-007</td>
<td>1.833e-007</td>
<td>-6.520</td>
<td>-6.737</td>
<td>-0.216</td>
</tr>
<tr>
<td>Fe₃⁺</td>
<td>8.718e-009</td>
<td>2.841e-009</td>
<td>-8.060</td>
<td>-8.547</td>
<td>-0.487</td>
</tr>
<tr>
<td>Fe(OH)⁺</td>
<td>1.243e-009</td>
<td>7.551e-010</td>
<td>-8.906</td>
<td>-9.122</td>
<td>-0.216</td>
</tr>
<tr>
<td>Fe₂(OH)₂⁺</td>
<td>7.784e-010</td>
<td>7.812e-010</td>
<td>-9.109</td>
<td>-9.107</td>
<td>0.002</td>
</tr>
<tr>
<td>FeCl₂⁺</td>
<td>3.363e-011</td>
<td>2.969e-011</td>
<td>-10.473</td>
<td>-10.527</td>
<td>-0.054</td>
</tr>
<tr>
<td>Fe₂(OH)₂⁺⁺</td>
<td>6.640e-012</td>
<td>9.045e-013</td>
<td>-11.178</td>
<td>-12.044</td>
<td>-0.866</td>
</tr>
<tr>
<td>FeCl⁻</td>
<td>2.604e-014</td>
<td>2.613e-014</td>
<td>-13.584</td>
<td>-13.583</td>
<td>0.002</td>
</tr>
<tr>
<td>Fe(OH)³⁻</td>
<td>8.065e-015</td>
<td>7.120e-015</td>
<td>-14.093</td>
<td>-14.147</td>
<td>-0.054</td>
</tr>
<tr>
<td>Fe₃(OH)₄⁺⁺</td>
<td>2.583e-015</td>
<td>1.146e-016</td>
<td>-14.588</td>
<td>-15.941</td>
<td>-1.353</td>
</tr>
<tr>
<td>H₂</td>
<td>0.000e+000</td>
<td>0.000e+000</td>
<td>-40.821</td>
<td>-40.819</td>
<td>0.002</td>
</tr>
<tr>
<td>Na⁺</td>
<td>2.101e-002</td>
<td>1.858e-002</td>
<td>-1.677</td>
<td>-1.731</td>
<td>-0.053</td>
</tr>
<tr>
<td>O₂</td>
<td>3.613e-011</td>
<td>1.813e-011</td>
<td>-10.743</td>
<td>-10.742</td>
<td>0.002</td>
</tr>
</tbody>
</table>

The output for pH 10, 7 and 4 shows that Fe(OH)₃(a) is precipitated at pH 10 and 7 but not at pH 4.

More flexible control of the `<loop>` variable can be had by reading in the values from a file using the `loopFile` keyword. The `FespeciesLoopFile.ppi` example in the demo directory provides an example of this approach.
72 Phreeqc mineral species

By default and for good reason, Phreeqc does not automatically precipitate all minerals for which the saturation index exceeds zero. Nor is it possible easily to encourage it to do so. The mineral species must be explicitly included in an EQUILIBRIUM_PHASES block. The actual mineral species that need to be considered will depend on the database being used (even the name of the same mineral species can vary between databases).

The following input file from demo\Phreeqc_minerals\FeZnminerals demonstrates how a list of all possible minerals can be created in the normal Phreeqc output file and used for pasting into the input file. The printphases.inc include file is used. This makes use of the Phreeqc SYS() function for enquiring about the status of the system:

```
# runs a single iteration of Phreeqc to generate a list of all possible Fe and Zn mineral phases
SPECIATION
calculationType                 "custom"
resolution                      0   # 1 iteration only
Phreeqc.0.out                   t   # writes Phreeqc.0.out file
selectedOutputLines             0   # tells PhreePlot not to expect any

CHEMISTRY
include 'printphases.inc'

SOLUTION 1
pH        1.95
units     mol/kgw
Fe        0.01
Zn        1e-6
Na        1e-1
Cl        1e-1
END

and printphases.inc looks like this:

PRINT
-reset false
-user_print true
USER_PRINT
-start
10 totm = SYS("phases", nm, nm$, tm$, cm)
20 FOR i = 1 TO nm
30    k = INSTR(nm$(i),"(g)")
40    k = k+INSTR(nm$(i),"Fix")+INSTR(nm$(i),"FIX")+INSTR(nm$(i),"fix")
50    IF(k > 0) THEN 70
60    PRINT"    ", CHR$(35)+PAD(nm$(i), 30), "0", "0"
70 NEXT i
-end

CHR$(35) is a convenient way of including the # symbol.

The output in Phreeqc.out using the wateq4f.dat database looks like this:

```
Reading input data for simulation 1.

# runs a single iteration of Phreeqc to generate a list of all possible Fe and Zn mineral phases

PRINT
-reset false
-user_print true
USER_PRINT
-start
10 totm = SYS("phases", nm, nm$, tm$, cm)
20 FOR i = 1 TO nm
30    k = INSTR(nm$(i),"(g)")
40    k = k+INSTR(nm$(i),"Fix")+INSTR(nm$(i),"FIX")+INSTR(nm$(i),"fix")
50    IF(k > 0) THEN 70
60    PRINT"    ", CHR$(35)+PAD(nm$(i), 30), "0", "0"
70 NEXT i
-end

CHR$(35) is a convenient way of including the # symbol.
The output for the minerals given above can be pasted into an input file and the required minerals un-commented. ‘0 0’ indicates that the mineral should precipitate when the saturation index exceeds 0 (first 0) but will not dissolve since the initial abundance is 0 (second 0) in all cases.

Therefore in order to find out all the possible mineral species in your system, edit the SOLUTION block of this file to include the components of interest, rename and run. Then look at the Phreeqc.out file.
It is often useful to know the relative abundance of all the chemical species in a system. ’species’ plots are a special type of custom plot that produce plots of the distribution or abundance of all the species for a particular element as a function of some master variable, often set to pH.

There are two variants of this plot in PhreePlot, one produces a percentage distribution and the other plots the log concentration. While these plots could be produced using the normal custom plot approach, the species plot approach does not require any prior knowledge of the names of the species present and is therefore usually easier to setup.
This produces similar output to the previous example but uses the ‘species’ calculationType to label the curves automatically. It uses the species-value paired output approach.

A species plot automatically generates a % species versus pH plot. This is a special type of custom plot in which PhreePlot expects the ‘out’ file to contain a succession of species name-%distribution pairs. It ignores any headings. Rather it gets the species name from the preceding column. It also expects the x-axis variable to be sent as the first pair of values. This is pH in this example but by changing the include file, any variable can be used.

The speciesvsph.inc include file specifies the ‘out’ or plot file. Pure phases (assumed to be mineral species) are appended with ‘(s)’ to distinguish them from aqueous species.

In order to generate the above type of plot, it is necessary to specify: (i) “species” as the plot type; (ii) a main species, here ‘Cd’; (iii) an initial chemistry and any equilibrium phases, and (iv) the x-axis variable and its range. It is also necessary to indicate with an <x_axis> tag how the x-axis variable changes the chemistry (here pH) and point customXcolumn to the pH column.

If any of the following plot titles are blank, they are set as follows: (i) xtitle = name accompanying the x-axis value (position defined by customXcolumn); (ii) ytitle = “% species”, and (iii) plotTitle = “Distribution of <mainspecies> with <xtitle>” where the angle brackets indicate the substitutions to be made. To omit, set the appropriate colour to ‘nd’.
SPECIATION

jobTitle                             "Speciation vs pH using 'species' plot type"
calculationType                      species
calculationMethod                    1
mainSpecies                          Cd
# logH range
xmin                                 -13.0
xmax                                 -6.0
resolution                           100

PLOT

plotTitle                            "Cd speciation\(\text{br}\); (using \"speciesvsph.inc\")"
# x-axis value is the second column - the first column is 'pH' (see out file)
customXcolumn                        2
# default is 14
pxmax                                13
# eliminates minor species
minimumYValueForPlotting             5.0
legendTitle                          "Cd species"
extraText                            "extratextCdspeciation.dat"

CHEMISTRY

# contains the logic for outputting the expected x-axis, y-axis (%distr) pairs
# expected by 'species' plot type
include 'speciesvsph.inc'

PHASES

Fix_H+
\[ \text{H}^+ = \text{H}^+ \]
\[ \log K = 0.0 \]

SOLUTION 1

temp      25
pH        7
units     mol/kgw
density   1
Cd        1e-4
Cl        2e-3
Na        0.1
N(5)      0.1 charge

USE solution 1

EQUILIBRIUM_PHASES

O2(g)     -0.677  0.1
 Fix_H+   &lt;x_axis&gt;  NaOH 10
-force_equality true

END
Cd speciation vs pH (log species plot)

This is a variant of the previous example but plots the log of the species concentrations vs pH rather than the %distribution. This change is made entirely within the include file which controls the values sent to the 'out' file. The include file has been changed to output log concentrations (see logspeciesvsph.inc) and ytitle has been given explicitly to override the default ' % species' title.

The size of the legend (key) text has been reduced using legendTextSize and its placement has been set in the extraText file.
Another example of a ‘species’ plot. Oxidising conditions have been maintained by equilibrating with a high partial pressure of oxygen. This is set by the \(<pO2>\) tag. Similarly the CO\(_2\) partial pressure is set with the \(<pCO2>\) tag.

The U speciation is dominated by U(VI) species under these conditions. Note the carbonate species at high pH.

useLineColorDictionary has been set to 1 to force the colour dictionary (linecolor.dat) to be read. This is where the colour of each of the lines is defined.

The minimumYValueForPlotting has been set to 5\% so that minor species are not plotted.

The labelSize has been set to 1.5 mm. The label anchors shown as red dots by each of the labels can be removed by setting trackSymbolColor to ‘nd’.
SPECIATION

jobTitle                             "Speciation vs pH using 'species' plot type"
calculationType                      species
calculationMethod                    1
mainSpecies                          "U"
xmin                                 3.0
xmax                                 10.0
resolution                           100
numericTags                          <pCO2> = "-3.5" \
# atmospheric PO2
<pO2> = "-0.677"

PLOT

plotTitle                            "U speciation vs pH (oxidizing conditions)<br>(using speciesvsph.inc)"
pxmin                                3.0
pxmajor                              1.0
labelSize                            1.5
# only plot species with max(conc)>5%
minimumYValueForPlotting             5.0
extraText                            "extratextUsupeciation_ox.dat"

CHEMISTRY

# PHREEQC code for outputting species-concn pairs
include 'speciesvsph.inc'

PHASES
Fix_H+
   H+ = H+
   log_k 0.0

SELECTED_OUTPUT

# omit all default output
  -reset                false

SOLUTION 1

temp      25
pH        7
pe        4
units     mol/kgw
# total U
  U         1e-6
# background electrolyte
  Cl        0.01
  Na        0.01

END

# second (final) simulation - loops on this one
USE solution 1
EQUILIBRIUM_PHASES

O2(g)     <pO2>  0.1
Fix_H+    -<x_axis> NaOH 1
   -force_equality true
# Uraninite(c) 0 0

solution only
  CO2(g)   <pCO2>  1.0

END
This is similar to the previous plot but \( \log P_{O_2} \) has been set to a low oxygen fugacity \( \log P_{O_2} = -70 \). Most of the plotted species are U(IV) species but \( UO_2(CO_3)_3^{4-} \) is a U(VI) species and \( UO_2^+ \) is a U(V) species.

It would be necessary to go to an even lower \( \log P_{O_2} \) value (-75) to ensure that no U(VI) species are plotted.
SPECIATION

jobTitle "Speciation vs pH using 'species' plot type"
calculationType species
calculationMethod 1
mainSpecies "U"
xmin 3.0
xmax 10.0
resolution 100
numericTags "\pCO2 = "-3.5" \n\pO2 = "-70"

PLOT

plotTitle "U speciation vs pH (reducing conditions)<br>(using speciesvspH.inc)"
pxmin 3.0
pxmajor 1.0
labelSize 1.5
minimumYValueForPlotting 5.0
extraText "extratextUspeciation_red.dat"

CHEMISTRY

# PHREEQC code for outputting species-concn pairs
include 'speciesvsph.inc'

PHASES

Fix_H+
H+ = H+
log_K 0.0

SELECTED_OUTPUT

# omit all default output
-reset false

SOLUTION 1

temp 25
pH 7
pe 4
units mol/kgw

# total U
U 1e-6

# background electrolyte
Cl 0.01
Na 0.01

END

# second (final) simulation - loops on this one

USE solution 1

EQUILIBRIUM_PHASES

O2(g) <pO2> 0.1
Fix_H+ -<x_axis> NaOH 1
-force_equality true

# Uraninite(c) 0 0

END

# omit for solution only

CO2(g) <pCO2> 1.0

END
Carbon speciation vs pH

This uses a ‘species’ plot to show the variation in carbon species with pH in an oxidising environment. It uses the ‘speciesvsph_inc’ include file and so plots % in species vs pH.

The Phreeqc output is accumulated in ‘species name, species concentration’ pairs, one line per pH. The order of the species output is based on decreasing C concentration (i.e. highest concentration first) and so the order changes as the pH changes. Because it is a ‘species’ plot, PhreePlot expects this type of paired output and sorts the data so that the column positions for all species are fixed. This enables them to be plotted.

With the given include file and input file setup, the first column is defined as the x-axis variable and so the value of customXcolumn is ignored.
SPECIATION

# plot %C species vs pH

calculationType  

calculationMethod  1 

mainSpecies  C 

# controls the range of pH plotted 
xmin  2 
xmax  13 

# controls the number of points on each curve 
resolution  100 

PLOT

plotTitle  "C speciation vs pH using \ 
\ speciesvsph.inc" 

eextraText  "extratextcarbonspeciation.dat" 

CHEMISTRY

# this file exports the required x-axis(pH), y-axis (%) value pairs. Edit as 

include 'speciesvsph.inc'

PHASES

Fix_H+ 
H+ = H+ 
log_k 0.0

SOLUTION 1

# carbonate speciation at least depends on temperature 

temp  10 
pH  7 
units  mol/kgw 

# total C 
C(4)  1e-1 as HCO3 

# background electrolyte 
Na  0.1 charge 
Cl  0.1 

END

USE solution 1

EQUILIBRIUM_PHASES

Fix_H+ -&lt;x_axis&gt;  NaOH 10 

-force_equality true 

END
This figure shows the surface speciation of As(V) adsorbed on goethite as a function of pH. The calculated curves were based on the CD-MUSIC model and the parameters of Stachowicz et al. (2006). This figure replicates the calculated curves of their Fig. 7 for one of the three surface loadings (1.70 μm/m², 3 g/L) studied. The input file generates plots for the other two surface loadings as separate files and the extraText file picks off the appropriate text based on the plot number.

In the other plots, the contribution of several of the species is everywhere less than 10%. These curves could be omitted by setting the minimumYValueForPlotting setting to 10.
SPECIATION
# plot all As species f(pH) including adsorbed species
  calculationType species
  calculationMethod 1
#
  mainSpecies "As"
xmin 3.0
xmax 12.0
# controls the number of points at which speciation is calculated
resolution 100
# defines <loop1> and <loop2> which are used below
loopFile "loopfig7.dat"
numericTags
  <AsT> = <loop1>
  <mass> = <loop2>

PLOT
  plotTitle "CD-MUSIC: As(V) surface speciation<br>(after Stachowicz et al., 2006, Fig. 7)"
xtitle pH
# 2nd column in selected output created by adsspeciesvsph.inc
customxcolumn 2
# plot xmin
pxmin 3
labelSize 1.5
# can be used to omit plotting of species that are always below this value
minimumyvalueforplotting 0.1
# here the As(3) species
extratext "extratextfig7.dat"
# turn off legend to the right of the plot
legendTextSize 0

CHEMISTRY
# this controls exactly what is plotted (see the system directory for the file)
include 'adsspeciesvsph.inc'
# must 'punch' x-axis, y-axis pairs

SELECTED_OUTPUT
  -reset false

SOLUTION 1
  Temp 25
  pH 2.9
  units mol/kgw
# total As(5) concn
  As(5) <AsT> mmol/kgw
# background electrolyte
  Na 1e-1
# N(5) only stable in oxidising environments
  N(5) 1e-1
include 'cdmusic_hiemstra.dat'
#
# Arsenate - these are the actual figures used in the SHR2006 paper and change the
speciation from the above database slightly
#
SURFACE_SPECIES
  Goe_uniOH-0.5 + 2H+ + AsO4-3 = Goe_uniOAsO2OH-1.5 + H2O
  SHR2006 26.60  SHR2008
  log_k 26.62
  -cd_music 0.30 -1.30 0 0 0

  2Goe_uniOH-0.5 + 2H+ + AsO4-3 = (Goe_uniO)2AsO2-2 + 2H2O
  SHR2006 29.77  SHR2008
  log_k 29.29
  -cd_music 0.47 -1.47 0 0 0
\[2\text{Goe}_\text{uniOH}^{0.5} + 3\text{H}^+ + \text{AsO}_4^{3-} = \{\text{Goe}_\text{uniO}\}2\text{AsOOH}^- + 2\text{H}_2\text{O}\]

# SHR2006  33.00  # SHR2008

\[
\text{log}_k = 32.69
\]

-\text{cd\_music}  0.58  -0.58  0  0  0

**PHASES**

\text{Fix\_H}^+; \text{H}^+ = \text{H}^+ ; \text{log}\_k 0

**SURFACE 1**

# sites/nm2  m2/g  g

\text{Goe}_\text{uniOH}^{0.5}  3.45  98 <\text{mass}>

# C1  C2  \{in F/m2\}

-\text{cap}  0.85  0.75

\text{Goe}_\text{triOH}^{0.5}  2.7

-\text{cd\_music}

-\text{sites\_units}  \text{density}

**EQUILIBRIUM\_PHASES 1**

\text{Fix\_H}^+  -<\text{x\_axis}> \text{NaOH}

-force\_equality  true

\text{O}_2(g)  -0.67

**END**
**79 Test plot output formats**

This example is primarily to demonstrate how plot files can be created in different formats: **ps**, **eps**, **eps1**, **jpg**, **pdf** and **ai**. The conversions from the native **ps** format are all carried out by **Ghostscript** and so can only be produced if **Ghostscript** is properly installed and the **pdfMaker** setting is pointing to a valid directory and file.

The example shows a species distribution plot for aqueous Fe species in the Fe-Cl-H2O system as a function of pH.

The **minimumYValueForPlotting** has been set to -10 to eliminate minor species from the plot.
**SPECIATION**

jobTitle       "log speciation vs pH using ‘species’ plot type"

# plots concn/% of all species containing the main species

calculationType    species

calculationMethod 1

mainSpecies       Fe

xmin              -12.0

xmax              -2.0

resolution        100

minimumYValueForPlotting  -10

pdf               T

png               T

epsi              T

jpg               T

eps               T

**PLOT**

plotTitle         "Fe-H$_2$O species vs pH (using logspeciesvsph.inc)"

**CHEMISTRY**

# this controls exactly what is plotted (see the system directory for the file)

include 'logspeciesvsph.inc'

# must ‘punch’ x-axis, y-axis pairs

**PHASES**

Fix_H+

H+ = H+

log$_K$ 0.0

**SOLUTION 1**

temp            10

# pH for initial solution calculations only. Changed by <x_axis>

pH        7

units     mol/kgw

# background electrolyte

Na        0.1 charge

Cl        0.1

# total concn of element

Fe(3)     1e-3

END

USE solution 1

**EQUILIBRIUM_PHASES**

O2(g)       -0.677  0.1

# controls logH from xmin to xmax

Fix_H+ <x_axis> NaOH 10

-force_equality true

END
Fitting models to data

PhreePlot provides quite a versatile method for fitting chemical models to data (observations). This is sometimes called 'optimization' or 'parameter identification'.

Other methods of optimizing Phreeqc models have been used, e.g. PEST, but the integration within PhreePlot is tight and the additional learning step is a relatively small one once the basics of producing custom plots have been mastered.

Nevertheless successful optimization, whatever the software used, is always a bit of an art that takes some time and experimentation to become proficient. The examples included here can be used as a starting point to experiment with. The golden rule is to start out simple.
This example demonstrates the fitting of adsorption data to a Langmuir isotherm. Only one point is calculated per speciation calculation. This approach therefore requires \( n_{data} \times n_{iterations} \) Phreeqc runs. The onePass setting is set to false in order to tell PhreePlot that it will require more than pass through the Phreeqc code in order to calculate dependent variable values for the complete set of data. The next example shows how to calculate all data points in one Phreeqc run thus reducing the number of Phreeqc runs to \( n_{iterations} \). This latter approach speeds up the calculations at the expense of a more complex input file.

The input file must describe how to read in values for the dependent variable and all independent variables from the fit data file. The columns can be specified by column number or column name (from the header).

Fine tuning of the fitting parameters often helps convergence. The choice of \( \text{fitFiniteDiffStep-Size} \) is often critical to get the fitting started. It is also important to decide how the residuals are going to be weighted (the error model), here unit weighting has been used.

The plot can use any column from the ‘pts’ file. This includes special columns labelled ‘observed’, ‘calculated’, ‘residuals’ and ‘weightedResiduals’ as well as all the columns from the fit data file and from the selected output.
# simple example of fitting to a Langmuir isotherm using unit weighting
# basic fit with isotherm plot

SPECIATION
  jobTitle                             "Test fitting: absolute errors (unit weighting)"
  calculationType                      fit
  calculationMethod                    1
 FIT
# contains a list of observations and independent variables
dataFile                             iso.dat
# does a separate PHREEQC simulation for each observation -
onepass                              FALSE
# slow but easy to set up

# name of column in fit data file containing the observations (dep variable)
dependentVariableColumnObs           Znsorbed
# name of column in selected output file containing the calcd values of the dep
variable
dependentVariableColumnCalc          sorbZn
# 0 = unit weights
fitWeightingMethod                   0
# initial step size for each adjustable parameter
fitFiniteDiffStepSize                1.0E-3
# column header in fit data file for which PHREEQC simulation(s) to use for each
observation
mainLoopColumn                       sim
numberOfFitParameters                2
fitParameterNames                    log_k  M1
# 0 = parameters on a linear scale, 1 = log10 parameters before fitting
fitLogParameters                     0 0
# 1 = adjustable
fitAdjustableParameters              1 1
# initial values (starting point)
fitParameterValues                   3.0 1.0

PLOT
  plotTitle                            "Zn sorption on Hfo"
xtitle                               "Zn concn (mmol/L)"
ytitle                               "Zn sorbed (mmol Zn/mol Fe)"
# plot isotherm (x = Znconc, y = sorbed)

# y = line from calculated values from out file with column heading = ‘calculated’
lines                                calculated
# y = points from observed values from out file with column heading = ‘observed’
points                               observed

lineWidth                            0.4
lineColor                            red
# no labels on plot
labelSize                            0.0
# no legend (key)
legendTextSize                       0.0
pointSize                            4.0
# x = Znconc column in out file
customXcolumn                        Znconc
# extra text on plot
extraText                            "extratextiso.dat"

CHEMISTRY

PHASES
Fix_H+
  H+ = H+
  log_k 0.0

SURFACE_MASTER_SPECIES
  Surf Surf
SURFACE_SPECIES
Surf = Surf
log_k 0.0
# Langmuir model
Surf + Zn^{2+} = SurfZn^{2+}
# this is where log_k (updated parameter value) is substituted
log_K <log_k>

SELECTED_OUTPUT
-high_precision true
-reset false
PRINT
-reset false

USER_PUNCH
# fit Langmuir isotherm
# these are the column headings used for tag names: NB this is the output pH not the input pH
-headings sorbZn pH mmolZn step_no
10 sorbedZn=Surf("Zn","Surf")
# NB variable name (used internally) is distinct from column header
20 if sorbedZn>0 THEN punch sorbedZn, -la("H^{+}"), tot("Zn")*1e3, step_no

SOLUTION 1
-pH <pHobs>
-units mmol/L
# 1 M NaNO\textsubscript{3} background electrolyte
Na 1000
N(\textsubscript{5}) 1000
# <Znconcn> from iso.dat
Zn <Znconcn>

SURFACE
# this is where the max number of sites (updated parameter) is substituted
Surf <M1>
-equil 1
-no_edl

EQUILIBRIUM_PHASES
# pHobs from the fit data file
Fix_H+ -<pHobs> NaOH
-force_equality true

END
This is exactly the same as the previous example except that all 10 points are calculated in a single pass of Phreeqc. This requires the onePass setting to be set TRUE. It also requires some rearrangement of the way that the input data are presented. There are two critical differences compared with the ‘one point per pass’ approach:

(i) in the ‘one point per pass’ approach, the CHEMISTRY part of the input file has just one Phreeqc simulation (one END at the end of the file) whereas in the ‘calculate all points in one pass’ approach there is one simulation for each data point, i.e. multiple END’s;

(ii) in the first approach, selectedOutputLines points to just the last line, e.g.

\texttt{selectedOutputLines 1}

whereas with the ‘calculate all points at once’ it points to many lines with the ‘auto’

\texttt{selectedOutputLines auto}

where ‘auto’ selects all the lines found in the selected output which should therefore contain one line per point. Each line of output represents one simulation.

Note that the Phreeqc setup in the input file contains many simulations which are near-repeats. This could become tedious to setup for large datasets and would probably require some form of automatic generation. PhreePlot includes a simple input file pre-
processor which can generate the necessary input file (see demo\fitpreprocessor\ppiso.ppi) from a simplified skeleton file.
SPECIFICATION

jobTitle                             "Test fitting"
calculationType                      fit
calculationMethod                    1

FIT

# fit data file
dataFile                             ison.dat
# says that all dependent variable values output in one PHREEQC pass (see below)
onePass                              TRUE
# NB no half way house - either one calculation per pass or the whole lot
mainloop                             2
# selectedOutputLines                0 1 1 1 1 1 1 1 1 1 1 1 1 # not necessary
since pre-loop doesn't write selected output
# observations from the fit data file
dependentVariableColumnObs          Znsorbed
# from selected output
dependentVariableColumnCalc          sorbZn
# initial step size
fitFiniteDiffStepSize                1.0E-3
# from the fit data file
weightColumn                         wt
# blockRangeColumn                   sim                          # from the fit
data file - defaults ok
# mainLoopColumn                      main                          # from the fit
data file - defaults ok
numberOfFitParameters                2
fitLogParameters                     0 0
# 1 = adjustable, 0 = fixed
fitAdjustableParameters              1 1
# starting values
fitParameterValueValues              3.0 1.0

PLOT

plotTitle                            "Zn sorption on Hfo<br>(onePass = TRUE)"
xtitle                               "Zn concn (mmol/L)"
ytitle                               "Zn sorbed (mmol Zn/mol Fe)"
# from the out file
lines                                calculated
# from the out file
points                               observed
lineWidth                            0.4
lineColor                            red
labelSize                            0.0
legendTextSize                       0.0
pointSize                            4.0
# from the out file
customXcolumn                        Znconcn
extraText                            extratextiso.dat

CHEMISTRY

#1
PHASES
Fix_H+
H+ = H+
log_k  0.0

SURFACE_MASTER_SPECIES
Surf Surf
SURFACE_SPECIES
Surf = Surf
log_k  0
END

#2
SURFACE_SPECIBS
   Surf + Zn^2+ = SurfZn^2+
# <log_K> is binding constant - substituted from the parameters defined above
   log_K <log_k>
PRINT
   -selected_output true
#   -reset false
SELECTED_OUTPUT
   -high_precision true
   -state true
#   -m Zn^2+ SurfZn^2+
SOLUTION_SPREAD
   -pH <pHobs>
# this is an easy way to put in data - paste in here from a spreadsheet etc
   -units mmol/L
   # strictly PHREEQC names, separated by tabs
   NumberNaN(5)ZnpH
# this is the pH used
   1100010000.035.5
   # data separated by tabs
   2100010000.695.5
   3100010000.1185.5
   4100010000.1665.5
   5100010000.2175.5
   6100010000.275.5
   7100010000.3255.5
   #100010000.3895.5
   9100010000.4535.5
   10100010000.5125.5
USER_PUNCH
# fit Langmuir isotherm
# this is the output pH
   -headings sorbZn pH molZn step
# dependent variable calculated here
10 sorbedZn = SURF("Zn","Surf")
20 if sorbedZn>0 THEN punch sorbedZn, -la("H+"), tot("Zn")*1e3, step_no
END

#3
SURFACE
# <M1> is the number of sites - substituted from the parameters defined above
   Surf <M1>
   -no_edl
# this uses solution Number 1 above
   -equil 1
END

#4
SURFACE
   Surf <M1>
   -no_edl
# this uses solution Number 2 above etc
   -equil 2
END

#5
SURFACE
   Surf <M1>
   -no_edl
   -equil 3
END

#6
SURFACE
   Surf <M1>
   -no_edl
-equil 4
END

#7
SURFACE
Surf <M1>
-no_edl
-equil 5
END

#8
SURFACE
Surf <M1>
-no_edl
-equil 6
END

#9
SURFACE
Surf <M1>
-no_edl
-equil 7
END

#10
SURFACE
Surf <M1>
-no_edl
-equil 8
END

#11
SURFACE
Surf <M1>
-no_edl
-equil 9
END

#12
EQUILIBRIUM_PHASES
   Fix_H+ <-pHobs> NaOH
SURFACE
Surf <M1>
-no_edl
-equil 10
END
This is similar to the `\demo\iso\ison.ppi` except that the near-repetitive blocks are generated automatically by invoking the PhreePlot pre-processor (Section 13) to expand the various blocks containing `<repeatStart n>` and `<repeatEnd n>` tags where `n` is a positive integer.

# This demonstrates an efficient (fast) method of fitting data to a PHREEQC model.
# It uses the PhreePlot pre-processor which replicates blocks of PHREEQC code with minor changes.
# This enables the 'one pass' option to be used and avoids the copying normally necessary (cf ison.ppi).
# The expanded input file is written to the log file.

**SPECIATION**

```plaintext```
jobTitle                             "Test fitting"
calculationType                    "fit"
calculationMethod                  1
```plaintext```
FIT
# fit data file - has observations
dataFile "isopp.dat"
# this produces a block of selected output with 10 lines of data (not 1)
onePass TRUE
mainLoop 1
# column in isopp.dat
dependentVariableColumnObs sorbed
# column in selected output
dependentVariableColumnCalc Znsorbed
# initial step size for parameter adjustment
fitFiniteDiffStepSize 1.0E-03
# column in fit data file with the weights
weightColumn wt
numberOfFitParameters 2
fitParameterNames "log_k" "M1"
fitLogParameters 0 0
# l= adjustable, 0 = fixed
fitAdjustableParameters 1 1
# initial values
fitParameterValues 3. 1.

PLOT
plotTitle "Zn sorption on Hfo<br>(all in one pass of the input preprocessor)"
xtitle "Zn concn (mmol/L)"
ytitle "Zn sorbed (mmol Zn/mol Fe)"
# plot this column from the out file as lines
lines calculated
points observed
lineWidth 0.4
lineColor "red"
labelSize 0.0
legendTextSize 0.0
pointSize 4.0
customXcolumn Znconcn

CHEMISTRY
# to see input and selected output on screen
PRINT
 reset false
- selected_output false
SURFACE_MASTER_SPECIES
 Surf Surf
SURFACE_SPECIES
 Surf = Surf
 log_k 0
 Surf + Zn+2 = SurfZn+2
# from fitParameterNames
 log_k <log_k>

SELECTED_OUTPUT
- high_precision true

USER_PUNCH
- headings Znsorbed pH molZn step
10 sorbedZn=SURF("Zn","Surf")
20 if (step_no = 0) THEN punch sorbedZn, -la("H+"), tot("Zn")*1e3, step_no

SOLUTION_SPREAD
DescriptionZnpHNaN(5)
mmol/kgwmmol/kgwmmol/kgw
13.00E-02 5.51 000 1000
26.90E-02 5.51 000 1000
31.18E-01 5.51 000 1000
41.66E-01 5.51 000 1000
52.17E-01 5.51 000 1000
62.70E-015.510001000
73.25E-015.510001000
83.88E-015.510001000
94.53E-015.510001000
105.12E-015.510001000
END

PRINT
     -selected_output true

<repeatStart1>  1 10
SURFACE
# from fitParameterNames
   Surf <M1>
     -no_edl
     -equil <repeatValue1>
END
<repeatEnd1>
This example fits sorption data for Ni sorption by goethite from data by Sherman and Peacock (unpublished). Ni sorption was measured as a function of pH at a constant solid solution ratio of goethite and constant background electrolyte concentration (0.1M NaNO₃). The dependent variable was the final solution Ni concentration after sorption. The independent variable was the final pH.

The data were fitted to the Dzombak and Morel (1990) diffuse double layer model as implemented in Phreeqc (surface activities are defined in terms of mole fractions). The Ni was assumed to bind to two types of bidentate surface sites as inferred from EXAFS data. The edl SURFACE option was used. Two log K values were fitted.

The finiteDiffStepSize was set to 1e-2 which is a large enough shift in the log K's to give a small but significant change in the objective function while still giving reliable derivatives.

The fit is good but there is really not enough data to provide a convincing test of the model. The line colour (blue) and points colour (red) are both determined by the line colour dictionary since useLineColorDictionary has been set to 1. This means ‘use the dictionary if present and if the species are defined’, which they are.
# fit some Ni sorption to goethite data

SPECIATION
calculationType  fit
calculationMethod  1

FIT
# fit data file - fit the final Ni concn not the amount sorbed
dataFile "Nisolnfrsh.dat"
# final Ni concn is in column 2 of fit data file
dependentVariableColumnObs  2
# NB this is often a good way of fitting sorption data where possible
dependentVariableColumnCalc  2
# size of initial adjustment of parameters when fitting
fitFiniteDiffStepSize  1.0E-02
fitConvergenceCriterion  1.0E-12
fitStepSize  1.0
# 0 = unit weighting for all points
fitWeightingMethod  0
numberOfFitParameters  2
fitParameterNames "log_k1" "log_k2"
# 0 = linear parameters
fitLogParameters  0 0
# 1 = adjustable
fitAdjustableParameters  1 1
# initial values of log_k1 and log_k2
fitParameterValues  8.0 0

PLOT
plotTitle "Ni sorption by goethite (edl)"
xoffset  60
xtitle pH
ytitle "soln Ni (mol/L)"
# p or plot limits
pxmin  4.0
pxmax  9.0
pymin  0
pymax  5.E-04
# number of decimal places
pydec  4
lineColor blue
lineWidth  0.4
# calculated column from out file
lines calculated
# observed column from out file
points observed
# key size
legendTextSize  1.9
pointSize  3.0
# suppress labelling
labelSize  0
# column 6 of out file
customXcolumn  6
extraText "extratextfitNi.dat"

CHEMISTRY
TITLE Goethite surface lpK model, Ni sorption 1 site.
PHASES
Fix_H+
H+ = H+
log_k 0.

SURFACE_MASTER_SPECIES
Fes_ Fes_OH-0.5

SURFACE_SPECIES
# surface charging
Fes_OH-0.5 = Fes_OH-0.5
log_k 0.0
\[ \text{Fes}_\text{OH}^{-0.5} + \text{H}^+ = \text{Fes}_\text{OH}^2+0.5 \]
\[ \log_k 8.50 \]

# bidentate model
\[ 2\text{Fes}_\text{OH}^{-0.5} + \text{Ni}^{+2} = (\text{Fes}_\text{OH})^2\text{Ni}^+ \]
\[ \log_k <\log_k1> \]

\[ 2\text{Fes}_\text{OH}^{-0.5} + \text{Ni}^{+2} + \text{H}_2\text{O} = (\text{Fes}_\text{OH})^2\text{NiOH} + \text{H}^+ \]
\[ \log_k <\log_k2> \]

SELECTED_OUTPUT
  high_precision true
  reset false

USER_PUNCH
headings pH Ni
-start
# pH and total dissolved Ni (calcd)
10 punch -la("H+"), TOT("Ni")
# fitting compares TOT("Ni") with obsd total, eg by ICP-AES
-end

SURFACE 1
# -diffuse_layer
# -no_edl
# goethite parameters
  \text{Fes}_\text{OH}^{-0.5} 1.09e-3 32.7 3.33

SOLUTION 1
units mol/kgw
# NiT
  Ni 0.4258e-3
# background electrolyte
  Na 0.1
  N(5) 0.1 charge

EQUILIBRIUM_PHASES
  \text{O}_2(\text{g}) -0.67 0.1
  \text{Fix}_\text{H}^+ -<pHobs> \text{NaOH}
  -force_equality true
  \text{Ni(OH)}_2 0 0

END
84  **As(V) sorption on hydrous ferric oxide**

This example uses the As data used by Dzombak and Morel (1990) to fit the three surface log K's for As(V) sorption by HFO. A global fit has been used. The weighting factors have all been set to one.

Line breaks in the input data file are used to defined line breaks in the plots – essentially six different data sets. The line and point colours are read from the line colour dictionary.

Some of the fits are not very good - EAs4 (orange points, orange line), for example, shows quite a large deviation between observations and fitted values.

Note that the in-plot labels are only plotted for the lines not the points. The lines and points for each dataset could be given the same colour by editing the line colour dictionary and replotting. convertLabels has been set to false to prevent the labels being interpreted as species names and therefore subscripting the final number.

changeColor has been set to `TRUE` to ensure that the various curves, which are all subsets of data from the same column, are given separate colours. This applies equally to both the points and lines sets of data – hence they follow the same colour sequence.
# Example of fitting some As sorption on Hfo data from Dzombak and Morel (1991)

SPECIATION
  calculationType                      fit
  calculationMethod                    1
  labels                               "EAs1" "EAs2" "EAs3" "EAs4" "EAs5" "EAs6"
  "EAs1" "EAs2" "EAs3" \ 
  # used in turn for labelling points then curves in plot
  "EAs4" "EAs5" "EAs6"

FIT
  # file containing observations and independent variables
  dataFile                             "1eAsv.dat"
  # dep variable is in column 6
  dependentVariableColumnObs           6
  # this where the calcd values are foun in selected output - see below
  dependentVariableColumnCalc          4
  # size of initial shift in parameter values looking for response
  fitFiniteDiffStepSize                1.0E-02
  # controls when convergence has been achieved
  fitConvergenceCriterion              1.0E-03
  fitStepSize                          1.0
  # 2 = take weights from fit data file
  fitWeightingMethod                   2
  # weights in column 7 in fit data file
  weightColumn                         7
  # column 8 defines which PHREEQC simulation (see below) to use for each point
  blockRangeColumn                     8
  numberOfFitParameters                3
  fitParameterNames                    "log_K1" "log_K2" "log_K4"
  # 0 = linear parameter values (ie don’t use log param)
  fitLogParameters                     0 0 0
  # 1 = adjustable (0 = fixed)
  fitAdjustableParameters              1 1 1
  # initial values
  fitParameterValues                   29.31 23.51 10.58

PLOT
  plotTitle                            "Refitting As(V) sorption data for
                                      Hfo<br>(1eAsv.dat)"
  xtitle                               pH
  ytitle                               "% bound"
  # the 'calculated' column in the 'out' file is plotted as a line
  lines                                calculated
  # the 'observed' column in the 'out' file is plotted as points
  points                               observed
  # prevents the labels being interpreted as species
  convertLabels                        F
  # give subsets a sequence of difft colours
  changeColor                          T
  # 0 = do NOT use the line colour dictionary for colours
  useLineColorDictionary               0
  # symbols will be 3 mm (nominal)
  pointSize                            3.0
  # x-axis variable is in column 8 of the 'out' file
  customXcolumn                        8
  # can use this to scale whole plot
  plotFactor                           1.0
  # additional text for plot
  extraText                            "extratextfithfoAsv.dat"

CHEMISTRY

PHASES
  Fix_H+
  H+ = H+
  log_k 0.
  Fe(OH)3(a)                  112
  Fe(OH)3 + 3H+ = Fe+3 + 3H2O
  log_k                      4.891
# prevents Fe(OH)₃(a) from dissolving
- add_constant -10

**SURFACE_SPECIES**

# Arsenate

\[ \text{Hfo}_{\text{wOH}} + \text{AsO}_4^- + 3\text{H}^+ = \text{Hfo}_{\text{wHAsO}_4} + \text{H}_2\text{O} \]

# the first parameter is substituted here

\( \log_k <\log_K_1> \)

\[ \text{Hfo}_{\text{wOH}} + \text{AsO}_4^- + 2\text{H}^+ = \text{Hfo}_{\text{wHAsO}_4^-} + \text{H}_2\text{O} \]

\( \log_k <\log_K_2> \)

\[ \text{Hfo}_{\text{wOH}} + \text{AsO}_4^- = \text{Hfo}_{\text{wOHAsO}_4^-} \]

\( \log_k <\log_K_4> \)

**SELECTED_OUTPUT**

high_precision true
reset false

**USER_PUNCH**

# fourth column (% sorbed) is compared with observations
- headings pH Hfo AsT %sorbed
10 Hfo = equi("Fe(OH)₃(a)")
20 totAs = SYS("As")
30 pcsorb = 100 * SURF("As", "Hfo") / totAs
40 PUNCH -la("H⁺"), Hfo, totAs, pcsorb

**SURFACE 1**

# D&M Hfo parameters

\( \text{Hfo}_{\text{wOH}} \text{Fe(OH)₃(a)} \text{equilibrium_phase} 0.005 53300 \)

\( \text{Hfo}_{\text{wOH}} \text{Fe(OH)₃(a)} \text{equilibrium_phase} 0.2 \)

**SOLUTION**

units mol/kgw

# <I>, <FeT> and <AsT> are from the fit data file
Na <I>
N(5) <I> charge
Fe <FeT>
As <AsT>

EQUILIBRIUM_PHASES

O₂(g) -0.67 0.1

# <pH> from the fit data file
Fix_H⁺ -<pH> NaOH
-force_equality true
Fe(OH)₃(a) 0 0

END
Contour plots provide a way of viewing the variation of some factor in two dimensions. The contour plots generated by Phreeplot are simple, classical 2D views of the surface. The viewing angle of the generated surface is always looking down directly from above.

The user has control over many of the plotting parameters such as the choice of the contour levels, and the size and colour of many of the plot attributes.

The challenge is to generate a set of data and choose a set of contour levels that produces a good-looking plot while avoiding trying to trace numerical noise. This is largely controlled by the choice of resolution used to generate the regular grid of values, and the choice of the contour values. Plots based on geochemical data can produce areas with both extremely large and extremely low gradients, both of which can be challenging to contour.
85 Contour two metals at three resolutions

Besides demonstrating the generation of contour plots, this example (demo\contour\contour_hfo-metalx.ppi) shows the use of tags in the PhreePlot section of the main input file. These are used to produce contour plots for two metals, Zn and Pb, at three resolutions, 10, 25 and 100. The data contoured are the percentage of metal adsorbed by HFO in the presence of a fixed amount of metal, Fe and background electrolyte. The two variables, pH and O$_2$(g) fugacity, change over a wide range leading to the variable dissolution/precipitation of HFO on which the Zn and Pb are adsorbed. This combination produces a total of six plots.

The calculationType keyword is set to ’contour’. The <mt> tag defined in numericTags defines the total concentration of metal in the system and the system-defined <mainspecies> tag defines the metals of interest. This tag is generated from the mainspecies keyword list.

The Phreeqc calculations are relatively straightforward. A solution of either Zn or Pb plus 0.01 mol/kgw Fe and 0.1 mol/kgw NaCl background electrolyte is brought to a particular pH and log fO$_2$(g) using NaOH and O$_2$(g), respectively, as defined by the EQUILIBRIUM_PHASES keyword data block. This uses the <x_axis> and <y_axis> tags. The values of these are generated on a regular grid by the ’contour’ routine using xmin, xmax, ymin, ymax and resolution. The ’hfo.inc’ include file is retrieved from the system directory and adds the Hfo surface and the DLM adsorption model.

The USER_PUNCH block calculates the % adsorbed using TOT() for the total dissolved metal concentration and SYS() for the total number of moles of the metal in the system. The total dissolved concentration has to be multiplied by the total amount of water (in kg) in the system to convert it to the number of moles of dissolved metal. The pH and percent adsorbed (%s) are output to the selected output once per iteration.

The selected output results accumulate in the ’out’ file based on the USER_PUNCH selected output. It is this file that provides the z-data to contour. Note that the x- and y-data are not output – they are defined implicitly by their position in the ’out’ file, the calculation domain and the grid resolution. The contourZvariable is defined as ’%s’. The heading of the outfile is searched for this string to determine the column position of the z-variable used in contouring.

A loopfile, loopmetalx.dat, defines a loop variable, res, which in turn defines a corresponding <res> tag which successively takes on the values 10, 25 and 50. Note that a loop value derived from a loopfile overrides any setting of the loop variable using loopMin etc.

Four PhreePlot loops are involved in the calculations, each based on a generated tag value. These are (from the least rapidly changing outermost loop to the most rapidly changing innermost loop): <mainspecies>, <res>, <y_axis> and <x-axis>. The <res> tag is used in the PhreePlot section of the input file and is updated once per loop iteration.

The plotTitle uses the <mainspecies> and <res> tags and these are substituted in the title just before plotting. Similarly, the extraText file contains the <mt> and <mainspecies> tags and these are substituted just before use.

Each mainspecies-res combination generates its own outfile so there are six such outfiles. The multipageFile t setting means that each of the six plots is written to a single multi-page ps file, one plot per page. The six plots are shown in Figure Ex85.1. Pb is adsorbed more strongly than Zn resulting in a larger area with >99% adsorption. There is no adsorption at low pH or low fO$_2$(g) due to the instability of HFO under those conditions. As the resolution increases, the clarity of the boundaries increase. The default placement of the labels is acceptable so there is no need to move them.
Figure Ex85. Contour plots for the percentage of Zn and Pb adsorbed by HFO as a function of pH and log f O$_2$(g) for three total metal concentrations.
# produces a contour plot for the Fe-H2O system with FeT = 0.01 mol/kg and Fe(OH)3(a) as a possible mineral phase

SPECIATION

<table>
<thead>
<tr>
<th>Speciation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>loopfile</td>
<td>loopmetalx.dat</td>
</tr>
<tr>
<td>calculationType</td>
<td><em>contour</em></td>
</tr>
<tr>
<td>calculationMethod</td>
<td>1</td>
</tr>
</tbody>
</table>

# see USER_PUNCH

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>contourZvariable</td>
<td>%s</td>
</tr>
<tr>
<td>mainspecies</td>
<td>Zn Pb</td>
</tr>
<tr>
<td>xmin</td>
<td>2.0</td>
</tr>
<tr>
<td>xmax</td>
<td>12.0</td>
</tr>
<tr>
<td>ymin</td>
<td>-90</td>
</tr>
<tr>
<td>ymax</td>
<td>0</td>
</tr>
</tbody>
</table>

# updated from loop file once per z-loop

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>resolution</td>
<td>&lt;res&gt;</td>
</tr>
<tr>
<td>numerictags</td>
<td>&lt;&lt;pet&gt; = 0.01 \ &lt;mt&gt;=10^&lt;M&gt;</td>
</tr>
</tbody>
</table>

PLOT

<table>
<thead>
<tr>
<th>Plotting Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>plotTitle</td>
<td>&quot;%&lt;mainspecies&gt; adsorbed by HFO&lt;br&gt;resolution</td>
</tr>
<tr>
<td>xtitle</td>
<td>pH</td>
</tr>
<tr>
<td>ytitle</td>
<td>( \log \ i{f} &lt;i&gt;f(i) = 0&lt;sub&gt;2&lt;/sub&gt;(g) ( \text{atm})&quot;</td>
</tr>
<tr>
<td>contours</td>
<td>1 10 20 50 80 90 99</td>
</tr>
<tr>
<td>extratext</td>
<td>extratextmetalx.dat</td>
</tr>
<tr>
<td>multipagefile</td>
<td>t</td>
</tr>
</tbody>
</table>

CHEMISTRY

# one simulation

# add standard Hfo DLM model

include hfo.inc

PHASES

Fix_H+; H+ = H+; log_k 0.

SELECTED_OUTPUT

-reset FALSE

-high_precision TRUE

USER_PUNCH

-headings pH %s

10 PUNCH -la("H+"), 100*(1 - TOT("<mainspecies>"))\*TOT("water")/{SYS("<mainspecies>")})

SOLUTION 1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>pH</td>
<td>1.8</td>
</tr>
<tr>
<td>units</td>
<td>mol/kgw</td>
</tr>
<tr>
<td>Fe(3)</td>
<td>&lt;Pet&gt;</td>
</tr>
<tr>
<td>&lt;mainspecies&gt; &lt;mt&gt;</td>
<td>Na 1e-1</td>
</tr>
<tr>
<td>Cl</td>
<td>1e-1</td>
</tr>
</tbody>
</table>

EQUILIBRIUM_PHASES 1

Fix_H+ -<x_axis> NaOH 10

-force_equality true

O2(g) <y_axis>

Fe(OH)3(a) 0 0
The \texttt{wateq4f.dat} database

The \textit{log} $f_{O_2}$-$\text{pH}$ predominance diagrams were calculated using the \texttt{ht1} method for each of the 32 components (excluding H, O and Humate) in the \texttt{wateq4f.dat} database. Humate was not included since Fulvate was included and all of the entries for Humate match those of Fulvate.

The total amount of each of the 32 components was set to 1e-2 mol/kgw.

Each of the possible 311 mineral species present in the database was allowed to precipitate if its saturation index indicated such. In practice, only 54 minerals actually did so.

The calculation is not meant to be particularly significant in terms of environmental chemistry. However, it is a fairly demanding test for the speciation program (\texttt{PHREEQC}) since the whole sequence of diagrams originally needed more than one week of continuous computing to calculate (more recent runs are considerably faster due to improvements in processor speed and the \texttt{PhreePlot} and \texttt{PHREEQC} code).

These calculations also provide a unique insight into the essential character of the \texttt{wateq4f.dat} database, something that is remarkably difficult to achieve by simply staring at a table of numbers.

The following 32 components were considered: Ag, Al, As, B, Ba, Br, C, Ca, Cd, Cl, Cs, Cu, F, Fe, Fulvate, I, K, Li, Mg, Mn, N, Na, Ni, P, Pb, Rh, S, Se, Si, Sr, U and Zn.

The following 311 minerals were considered: Acanthite, Adularia, Ag$_2$CO$_3$, Ag$_2$O, Ag$_2$SO$_4$, Ag$_3$PO$_4$, AgF:4H$_2$O, AgMetal, Al$_2$O$_3$(OH)$_3$, Albitine, AlumK, Alunite, Alunocile, Angleite, Anhydrite, Anilite, Aninite, Anorthite, Antlerite, Aragonite, Arsenolite, Artinite, As$_2$O$_3$(cr), As$_2$S$_3$(am), AsI$_3$, Atacamite, Autunite, Azurite, Ba$_3$(AsO$_4$)$_2$, BaF$_2$, Barite, Basaluminate, BasSeO$_3$, Bassettite, Beidellite, Bianchite, Birnessite, Bixbyte, BlaubleiI, BlaubleiII, Boehmite, Brochantite, Bromyrite, Brucite, Bunzenite, B-UO$_2$(OH)$_2$, Ca$_3$(AsO$_4$)$_2$, Calcite, CaSeO$_3$, Cd$_2$(BO$_2$)$_2$, Cd$_2$(gamma), Cd$_2$(OH)$_2$(a), Cd$_2$(OH)$_2$(b), Cd$_3$$(SO_4)$$_2$, Cd$_3$(OH)$_4$SO$_4$, Cd$_3$(PO$_4$)$_2$, Cd$_4$(OH)$_6$SO$_4$, CdBr$_2$:2H$_2$O, CdCl$_2$, CdCl$_2$:2H$_2$O, CdCl$_2$:3H$_2$O, CdCl$_2$:H$_2$O, CdF$_2$, CdI$_2$, CdMetal, CdOHCl, CdSO$_4$, CdSO$_4$:2H$_2$O, Celestite, Cerargyrite, Cerrusite, Chalcanthite, Chalcedony, Chalcocite, Chalcopyrite, Chlorite14A, Chlorite7A, Chrysothile, Claudetite, Clinohumite, Closomophosphate, Coffinite, Cotunnite, Covellite, Cristobalite, Cu$_2$(OH)$_2$, Cu$_2$(OH)$_3$NO$_3$, Cu$_2$SO$_4$, Cu$_3$(AsO$_4$)$_2$:6H$_2$O, Cu$_3$(PO$_4$)$_2$, Cu$_3$(PO$_4$)$_2$:2H$_2$O, CuBr, CuCo$_3$, CuF, CuF$_2$, CuF$_2$:2H$_2$O, CuI, CuMetal, CuOCuSO$_4$, CupricFerrite, Cuprite, CuprousFerrite, CuSO$_4$, Diaspore, Dioptase, Djurulite, Dolomite(d), Dolomite, Epsomite, FCO$_3$Apatite, Fe$_2$O$_3$:7Cl:3, Fe$_2$(OH)$_3$(am), Fe$_2$(SeO$_3$)$_3$, Fe$_3$(OH)$_8$:8Fe$_2$(ppt), FeSe$_2$, Fluorapatite, Fluorite, Forsterite, Galena, Gibbsite, Goethite, Goslarite, Greenalite, Grencrskite, Greigite, Guminite, Gypsum, Halite, Halloysite, Hausmannite, N-Antunite, Hematite, Hinsdalite, Huntite, Hxypyrsmophite, Hydrocerrusite, Hydromagnesite, Hydroxypatite, Illite, Iodide, Jarosite, JarositeH, Jarosite-K, Jarosite-Na, Juranite, Kaolinite, K-Antunite, Kmicite, Langite, Larnakite, Laumontite, Laurionite, Leonhardite, Litharge, Mackinawite, Magadiite, Maghemite, Magnesite, Magnetite, Malachite, Mangane, Massicot, Matlockite, Melanothallite, Melanterite, Millerite, Minum, Mirabilite, Mn$_2$(SO$_4$)$_3$, Mn$_3$(AsO$_4$)$_2$:2H$_2$O, Mn$_3$(PO$_4$)$_2$, MnCl$_2$:4H$_2$O, MnHPO$_4$, Mn$_3$(Green), MnSO$_4$, Montepone, Montmorillonite-Aberdeen, Montmorillonite-BelleFourche, Montmorillonite-Ca, Morenosite, Na$_4$UO$_2$(CO$_3$)$_3$, Na-
Autunite, Nahcolite, Nantokite, Natron, Nesquehonite, Ni(OH)$_2$, Ni$_2$SiO$_4$, Ni$_3$(AsO$_4$)$_2$:8H$_2$O, Ni$_3$(PO$_4$)$_2$, Ni$_4$(OH)$_6$SO$_4$, NiCO$_3$, Ningyoite, Nautrite, Orpiment, Otavite, Pb(BO$_2$)$_2$, Pb(OH)$_2$, Pb$_2$(OH)$_3$Cl, Pb$_2$O(OH)$_2$, Pb$_2$O$_3$, Pb$_2$CO$_3$, Pb$_2$SiO$_4$, Pb$_3$(AsO$_4$)$_2$, Pb$_3$(PO$_4$)$_2$, Pb$_3$O$_2$CO$_3$, Pb$_3$O$_2$SO$_4$, Pb$_4$(OH)$_6$SO$_4$, Pb$_4$O$_3$SO$_4$, PbBr$_2$, PbBrF, PbF$_2$, PbFPO$_4$, PbI$_2$, PbMetal, PbO:0.3H$_2$O, PbSiO$_3$, Phillipsite, Phlogopite, Phosgenite, Plattnerite, Plumbogummite, Portlandite, Prehnite, Przhevalskite, Pyrite, Pyrochroite, Pyroslite, Pyrophyllite, Quartz, Realgar, Retgersite, Rhodochrosite(d), Rhodochrosite, Rutherfordine, Saleeite, Schoepite, Scorodite, Se(s), SeO$_2$, Sepiolite(d), Sepiolite, Siderite(d)(I), Siderite, Silicagel, SiO$_2$(a), Smithsonite, Sphalerite, Sr-Autunite, SrF$_2$, Strengite, Strontianite, Sulfur, Talc, Tenorite, Thenardite, Thermonatrite, Torbernite, Tremolite, Trona, Tsumebite, U(HPO$_4$)$_2$:4H$_2$O, U(OH)$_2$SO$_4$, UO$_3$(c), UO$_2$(c), UF$_4$(c), UF$_4$:2.5H$_2$O, UO$_2$(a), UO$_2$HPO$_4$:4H$_2$O, UO$_3$(gamma), (UO$_2$)$_3$(PO$_4$)$_2$:4H$_2$O, Uramphite, Uraninite(c), Uranocircite, Uranophane, Vivianite, Wairakite, Willemite, Wurtzite, Zincite(c), Zincoxite, Zn(BO$_2$)$_2$, Zn(NO$_3$)$_2$:6H$_2$O, Zn(OH)$_2$-a, Zn(OH)$_2$-b, Zn(OH)$_2$-c, Zn(OH)$_2$-e, Zn$_2$(OH)$_2$:8SO$_4$, Zn$_2$(OH)$_3$:Cl, Zn$_3$(AsO$_4$)$_2$:2.5H$_2$O, Zn$_3$(PO$_4$)$_2$:4H$_2$O, Zn$_3$(SO$_4$)$_2$, Zn$_4$(OH)$_6$:SO$_4$, Zn$_5$(OH)$_8$:Cl$_2$, ZnBr$_2$:2H$_2$O, ZnCl$_2$, ZnCO$_3$:H$_2$O, ZnF$_2$, ZnI$_2$, ZnMetal, ZnO(a), Zn$_3$(a), ZnSiO$_3$ and ZnSO$_4$:H$_2$O.

The usual 'water limits' of 0.21 atm O$_2$(g) and 1 atm H$_2$(g) were used. All calculations were carried out for a temperature of 25°C. In practice, given the conditions used, only 54 of the 311 possible minerals precipitated somewhere within the range of conditions imposed. Inclusion of the other 257 minerals therefore has no impact on the predominance diagrams produced and in principle could be excluded. This results in approximately a four-fold increase in calculation speed illustrating that the number of pure phases considered can have a strong impact on the speed of PHREEQC calculations.

In these examples, we found that on average, about half the time was spent hunting along the domain boundaries and half was spent tracking along the internal boundaries. Of course this varies considerably from diagram to diagram.
Ag

All elements

\[ \text{O}_2(g) \cdot p_{\text{O}_2} \text{ atm} \]

\[ \text{pH} \]

\[ \log f_{\text{O}_2(g)} \]

\[ \text{AgMetal} \]

\[ \text{iodyrite} \]
2  Al

All elements

\begin{center}
\begin{tikzpicture}
\begin{axis}[
    width=\textwidth,
    height=\textwidth,
    xlabel={pH},
    ylabel={\(\log f_{O_2}(g)\)},
    xmin=2, xmax=10,
    ymin=-80, ymax=0,
    xtick={2,4,6,8,10},
    ytick={0,-20,-40,-60,-80},
    xticklabels={2,4,6,8,10},
    yticklabels={0,-20,-40,-60,-80},
    grid=both,
    grid style={line width=0.2pt, draw=gray!40},
    major grid style={line width=0.2pt, draw=gray!40},
    minor grid style={line width=0.2pt, draw=gray!10},
    axis lines=middle,
    axis line style={-},
    axis on top,
    legend style={at={(0.5,0.95)},anchor=north},
]

% Alunite
\node at (axis cs:3.5,-70) {Alunite};
% Al\textsuperscript{2+}
\node at (axis cs:4.5,-60) {Al\textsuperscript{2+}};
% Pyrophyllite
\node at (axis cs:5.5,-50) {Pyrophyllite};
% Diaspore
\node at (axis cs:6.5,-40) {Diaspore};
% O\textsubscript{2}(g) > 0.21 atm
\node at (axis cs:7.5,-30) {O\textsubscript{2}(g) > 0.21 atm};
\end{axis}
\end{tikzpicture}
\end{center}
3  As

![Graph showing the distribution of arsenic species under different pH and $fO_2(g)$ conditions. The graph includes species such as $H_2AsO_4^{-}$, $H_3AsO_4$, $H_2AsO_3$, $Ba_3(AsO_4)_2$, and As native. The pH range is from 2 to 10, and the $fO_2(g)$ range is from -80 to 0. The graph shows the stability field for all elements under varying conditions.]
\[ \text{PhreePlot Guide} \]

4 B

All elements

\[
\begin{array}{c}
\text{pH} \\
2 \quad 4 \quad 6 \quad 8 \quad 10
\end{array}
\]

\[
\begin{array}{c}
\log f_{O_2(g)} \\
-80 \quad -60 \quad -40 \quad -20 \quad 0
\end{array}
\]

\[ \Phi(g) = 0.21 \text{ atm} \]

\[ \text{H}_3\text{BO}_3 \]

\[ \text{H}_2\text{BO}_3^- \]

All elements
5  Ba

All elements

Ba$_3$(AsO$_4$)$_2$

O$_2$(g) - 0.21 atm

Ba$^{2+}$

Barite

Ba$^{2+}$
All elements

\[
\text{Br} \quad \text{O}_2(\text{g}) \rightarrow \text{Br}^+ \text{Br}^- + \cdot21 \text{ atm}
\]

\[
\begin{align*}
\text{pH} & \quad 0 & \quad -20 & \quad -40 & \quad -60 & \quad -80 \\
\log f \text{O}_2(\text{g}) & \quad 2 & \quad 4 & \quad 6 & \quad 8 & \quad 10
\end{align*}
\]
8  **Ca**

---

![Diagram showing the distribution of Ca across pH and log $f_O_2(g)$](image-url)
9  Cd

All elements

\[ \text{O}_2(g) > 0.21 \text{ atm} \]

\[ \text{log} f_{\text{O}_2(g)} \]

\[ \text{pH} \]

- Cd
- Cd\(^{2+}\)
- CdFulvate
- Otavite
- Greenockite
10 Cl

All elements

\[ \text{O}_2(\text{g}) \to \text{Cl}_2, 0.21 \text{ atm} \]

\[ \log f_{\text{O}_2(\text{g})} \]

\[ \text{pH} \]

2  4  6  8  10

-80  -60  -40  -20  0
11 Cs

All elements

$\text{Cs}^+$

$\text{O}_2(g) = 0.21 \text{ atm}$

$pH$

$\log f \text{O}_2(g)$

pH

$2 \quad 4 \quad 6 \quad 8 \quad 10$

$-80 \quad -60 \quad -40 \quad -20 \quad 0$
12 Cu

![Diagram showing the distribution of Cu phases as a function of pH and $\log f_{O_2}$]
The wateq4f.dat database

13 $F$

All elements

- $O_2(g) = 0.21$ atm
- $AlF_2^+$
- $AlF_3$
- Fluorite
- $F^-$

$pH$

$\log fO_2(g)$
14 Fe
15 Fulvate

![Diagram showing the distribution of Fulvate species across pH and log fO2(g) conditions.]

- Fulvate
- CuFulvate
- CdFulvate
- HFulvate
- Fulvate$^{2-}$

O$_2$(g) = 0.21 atm
All elements

\[ \log f_{O_2(g)} \]

\[ \text{pH} \]

\[ \text{O}_2(g) \rightarrow \text{0.21 atm} \]

lodyrite

0
-20
-40
-60
-80

2
4
6
8
10
The wateq4f.dat database

17 K

All elements

$\text{O}_2(\text{g}) = 0.21 \text{ atm}$

$\log f_{\text{O}_2(\text{g})}$

$\text{pH}$
18 Li

\[ \text{All elements} \]

\[ \text{Li}^+ \]

\[ \text{O}_2(g) \rightarrow \text{pH} \geq 2.21 \text{ atm} \]

\[ \log f_{\text{O}_2(g)} \]

pH

2 4 6 8 10
19 Mg

\[ \text{Mg}^{2+} \]

\[ \text{O}_2(\text{g}) = 0.21 \text{ atm} \]

All elements

\[ \log fO_2(\text{g}) \]

\[ \text{pH} \]

Magnesite

Chlorite
20  Mn

All elements

O_2(g) \rightarrow 21 \text{ atm}

Mn

Pyrolusite

Bixbyite

Rhodochrosite

Mn^{2+}

Mn(HPO_4)_2 \cdot 8H_2O

Mn\text{HPO}_4

Mn\text{HPO}_4

log \ f_{O_2(g)}
The wateq4f.dat database
22 Na

All elements

\( \text{Na} \)

\( \text{Na}^+ \)

\( \log f_{O_2} \text{(g)} \)

\( \text{pH} \)

\( \text{O}_2 \text{(g)} \rightarrow 0.21 \text{ atm} \)
The wateq4f.dat database

23 Ni

All elements

log fO₂(g) vs pH

Ni²⁺ Ni(OH)₂ Ni(CO₃)₂²⁻
All elements

-log fO₂(g)

pH

2 4 6 8 10

Clpyromorphite

FCC₃Apatite

Clpyromorphite

MnHPO₄

O_2(g) - 0.21 atm
25 Pb

All elements

\[ \log f_{O_2(g)} \]

\[ \text{pH} \]

O\(_2\) (g) = 8.21 atm

Clpyromorphite

Pb(OH)\(_2\)

PbMetal
26 Rb

All elements

$pH$ vs $\log f_{O_2(g)}$

$O_2(g) \rightarrow 0.21$ atm

$Rb^+$
All elements

O$_2$(g) > 0.21 atm

pH

log $f_{O_2(g)}$

-80

-60

-40

-20

0

Celestite

Barite

Chalcocite

Greenockite

SO$_4^{2-}$

S
28 Se

![Diagram of Se speciation](image)

All elements

- $\text{Se}(s)$
- $\text{H}_2\text{SeO}_3$
- $\text{HSeO}_3^-$
- $\text{SeO}_4^{2-}$
- $\text{SeO}_3^{2-}$

$pH$

$log f_{O_2}(g)$

$O_2(g) > 0.21 \text{ atm}$
29  Si

All elements

log $f_{O_2(g)}$

Quartz  Pyrophyllite  ZnSiO$_3$

O$_2(g)$ = 0.21 atm

pH

Chlorite$_{14A}$

Si
30 Sr

All elements

\[ \text{Sr}^{2+} \]

\[ \text{Sr} \]

\[ \text{O}_2(g) \text{ at } \frac{1}{2} \text{ atm} \]

Celestite

Strontianite

pH

log \( f_{O_2}(g) \)
31 U

All elements

pH

log fO₂(g)

O₂(g) = 0.21 atm

U₃O₈(c)

U₄O₉(c)

U₄(CO₃)₃⁴⁻

Na-Autunite

Schoepite

Uraninite(c)
32 Zn

All elements

log $f_{O_2}(g)$ vs pH

- $Zn^{2+}$
- $ZnSiO_3$
- $Zn$ (c)
References


Stachowicz, M., Hiemstra T., W. H. van Riemsdijk. 2006. Surface speciation of As(III) and As(V) in relation to charge distribution. J. Colloid Interface Sci. 302, 62–75.

Acknowledgements

PhreePlot inherits most of the hard work from others. Phreeqc does all of the geochemical calculations and is the work of David Parkhurst, Tony Appelo and Scott Charlton. Scott Charlton and David Parkhurst prepared the Phreeqc module that is embedded in PhreePlot. Phreeqc has become ever more powerful over the years and is a model of stability and reliability. It also comes with its own excellent documentation and databases. The Phreeqc format is now a standard format for thermodynamic databases.

The Postscript plotting library embedded in PhreePlot is from the late Kevin Kohler. This library enables PhreePlot to produce high quality, fully scalable plots. Ghostscript and GSview provide the perfect companions for rendering these files. Ghostscript provides a range of format conversions (pdf, png etc.) and can be configured to run directly from within PhreePlot.

Three of the fitting routines are by the late Mike Powell. The non-linear least squares routine (‘nlls’) has proved an invaluable and reliable assistant over many years, and the two newer routines are expected to be equally reliable. We also thank Tom Rowan for his ‘subpix’ code. The contouring routine is from Paul Bourke. After quite a lot of testing, we found that this rather simple and elegant algorithm proved the most reliable for contouring geochemical data. It has been slightly modified here to enable the contour regions to be filled with colour.

Geochemical modelling is nothing without the databases that go with it and so we would like to thank all of those who have helped to painstakingly prepare these for use in Phreeqc and elsewhere.

A number of smaller contributions have also been included. These are listed below with details of the sources and Conditions of Use:

Development of PhreePlot was begun while dgk was a full-time member of the British Geological Survey (Wallingford) and dmc was a member of the Centre of Ecology and Hydrology (Wallingford), both NERC Research Centres. We are grateful to these two institutions for their support.
<table>
<thead>
<tr>
<th>Function/software</th>
<th>Owner/source/Conditions of Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fitting</td>
<td>Harwell Subroutine Library (HSL Archive), VA05 routine by M J D Powell. <a href="http://www.cse.scitech.ac.uk/nag/hsl/hsl05.shtm">http://www.cse.scitech.ac.uk/nag/hsl/hsl05.shtm</a>. Free source code after registering; can freely distribute binaries derived from this but not the source itself; license file.</td>
</tr>
<tr>
<td>Postscript plotting</td>
<td>Kevin Kohler. <strong>PSPLOT</strong> subroutine plotting library. Available for download from <a href="http://www.nova.edu/ocean/psplot/">http://www.nova.edu/ocean/psplot/</a>. Egon Szondi has recently (2012) updated and ported the library to Fortran 95; follow the <strong>PSPLOT</strong> link given here.</td>
</tr>
<tr>
<td>Hash function</td>
<td>Rich Townsend, groups.google.com/group/comp.lang.fortran/browse_thread/thread/456a07645b776f78. Free source code.</td>
</tr>
<tr>
<td>The following are used but not embedded in <strong>PhreePlot</strong>:</td>
<td></td>
</tr>
<tr>
<td>Windows installer</td>
<td>Inno Setup, Copyright © 1997-2011 Jordan Russell. All rights reserved. <a href="http://www.jrsoftware.org/isinfo.php">http://www.jrsoftware.org/isinfo.php</a>. Open source code (Delphi) and binaries; license file. Free to use but copyrighted.</td>
</tr>
<tr>
<td>wget for Windows</td>
<td>wget.exe is used to check the PhreePlot server for the date of the latest version. It is part of the GnuWin package and is distributed under the GNU GPL (<a href="http://www.gnu.org/copyleft/gpl.html">http://www.gnu.org/copyleft/gpl.html</a>).</td>
</tr>
<tr>
<td>agrep</td>
<td>agrep.exe is used as an ‘approximate’ grep to assist with correcting keyword entry errors. It was developed by Sun Wu and Udi Manber at the University of Arizona. It is distributed with its own license conditions (<a href="http://www.tgries.de/agrep/#COPYRIGHT">http://www.tgries.de/agrep/#COPYRIGHT</a>).</td>
</tr>
</tbody>
</table>
Appendix 1. Glossary of terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch file</td>
<td>a text file containing batch commands that is executed by the command line interpreter (e.g. cmd.exe)</td>
</tr>
<tr>
<td>Chemistry section</td>
<td>all statements in the main input file following the line containing the word CHEMISTRY. This is made up of slightly modified Phreeqc code</td>
</tr>
<tr>
<td>custom plot</td>
<td>a type of xy-plot that is fully defined by the user</td>
</tr>
<tr>
<td>environment variable</td>
<td>a variable that is set and retrieved through the operating system and can be used by programs such as PhreePlot to configure the way in which they run. PhreePlot uses environment variables to define where the PhreePlot system directory is located and for defining the Paths to certain executable files such as the PhreePlot executable and Ghostscript executable</td>
</tr>
<tr>
<td>(fit) data file</td>
<td>a data file in tabular format that provides data (observations, independent variables) used in fit and simulate calculations</td>
</tr>
<tr>
<td>Ghostscript</td>
<td>Long-standing open source software for interpreting Postscript files under a number of operating systems and in many popular graphic file formats</td>
</tr>
<tr>
<td>grid approach</td>
<td>a way of calculating predominance and stability diagrams that simply calculates the speciation on a square grid of points</td>
</tr>
<tr>
<td>GSview</td>
<td>software that provides a pleasant user interface for running Ghostscript under Windows</td>
</tr>
<tr>
<td>hunt and track approach</td>
<td>a way of calculating predominance and stability diagrams that works by finding and then tracking the field boundaries from the domain boundaries inwards</td>
</tr>
<tr>
<td>include file</td>
<td>a file containing text that will be inserted into the Chemistry section at the point given by the include ‘filename’ statement</td>
</tr>
<tr>
<td>input file</td>
<td>a file containing PhreePlot keywords and settings</td>
</tr>
<tr>
<td>interrupt</td>
<td>the result of pressing the ‘Esc’ key during calculations. This enables the calculations to be paused, stopped or a keyword setting to be changed</td>
</tr>
<tr>
<td>job</td>
<td>a block of one or more runs</td>
</tr>
<tr>
<td>log file</td>
<td>a file that is normally generated by a PhreePlot run containing details of the run. The level of detail is controlled by the debug keyword</td>
</tr>
<tr>
<td>loop file</td>
<td>a data file in tabular format in which a row of data is read from the file for every iteration of the z-loop. Tags based on the column headings are generated from the line of data and may be used in the Chemistry section.</td>
</tr>
<tr>
<td>main input file</td>
<td>the principal file containing PhreePlot keywords and settings. It will also contain the Chemistry section, if present. It normally has the extension .ppi and is used to launch a job</td>
</tr>
<tr>
<td>main loop simulations</td>
<td>all simulations numbered mainLoop or greater. Used for iterating with the least overheads and with constant updating of the &lt;x_axis&gt; and &lt;y_axis&gt; tags</td>
</tr>
<tr>
<td>main species</td>
<td>a character string representing a main species variable that is controlled by the main species loop. Often used for a list of elements but can be used for a list of any character strings</td>
</tr>
<tr>
<td>mainspecies loop</td>
<td>the outermost alphanumeric loop controlled by the &lt;mainspecies&gt; tag and list of mainspecies</td>
</tr>
<tr>
<td>outfile or ‘out’ file</td>
<td>the selected output file. It is the default file used by custom calculations to make a plot</td>
</tr>
<tr>
<td>override file</td>
<td>an input file that contains PhreePlot keywords that is read immediately after the main input file and will override any settings in force at that point. The PhreePlot environment variable (PHREEPLOT) is required to tell your computer where to find the PhreePlot system directory. It is set during installation, through the Control Panel or with a program such as setx.exe. It should end in a backslash</td>
</tr>
<tr>
<td>PhreePlot executable</td>
<td>the file that contains the executable code for PhreePlot, normally called pp.exe</td>
</tr>
<tr>
<td>Term</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PhreePlot system directory</td>
<td>The directory containing many of the files needed by PhreePlot to run. It is normally called something like ...\PhreePlot\xx\system, where xx is the PhreePlot version number; it is normally stored in the PHREEPLOT environment variable.</td>
</tr>
<tr>
<td>Phreeqc</td>
<td>a popular geochemical speciation program (pH-redox equilibrium calculations in C) from the USGS that does all the geochemical calculations in PhreePlot.</td>
</tr>
<tr>
<td>Postscript</td>
<td>a page description language noted for its ability to produce scalable text and vector graphics of high quality. Postscript is the code that PhreePlot uses to define its plots. It can be rendered with the Ghostscript/GSview software combination and is readily converted to other formats. pdf is a popular descendant of Postscript and was also developed by Adobe. Postscript files normally have the extension .ps.</td>
</tr>
<tr>
<td>pp.log file</td>
<td>a file that contains a one-line entry for each PhreePlot run. It is automatically generated and is located in the PhreePlot system directory.</td>
</tr>
<tr>
<td>pp.set file</td>
<td>an input file that contains user-defined default settings for all the PhreePlot keywords. This is the first input file to be read and overwrites the program defaults. All simulations preceding the main loop simulation(s). Used for initialization calculations. Not repeated during execution of the x- and y-axis loops.</td>
</tr>
<tr>
<td>pre-loop simulations</td>
<td>a diagram, normally in two-dimensions, that shows the dominant chemical species for a particular 'component' over the domain of interest. The 'predominant' species is defined in PhreePlot as the most abundant species in terms of moles of component, irrespective of whether it is a solution, mineral or adsorbed species</td>
</tr>
<tr>
<td>predominance diagram</td>
<td>run a block of one or more simulations that are executed in a single call to Phreeqc tabular output generated by Phreeqc following some calculations. The output can be controlled using keywords such as SELECTED_OUTPUT and USER_PUNCH.</td>
</tr>
<tr>
<td>selected output file</td>
<td>a 'file' created by Phreeqc that is used to communicate between Phreeqc and PhreePlot. The actual name of the file is defined by the -file identifier in the SELECTED_OUTPUT keyword data block of Phreeqc. The default name is selected_1.0.out but it can be an unnamed virtual file in PhreePlot when there is minimal debugging taking place.</td>
</tr>
<tr>
<td>simulation number</td>
<td>a block of one or more Phreeqc keywords ending with an END or the end-of-file data block.</td>
</tr>
<tr>
<td>stability diagram</td>
<td>the sequence number of a simulation counted from the top.</td>
</tr>
<tr>
<td>tag</td>
<td>similar to a predominance diagram except that a mineral species, if present, always take precedence over any solution species.</td>
</tr>
<tr>
<td>x-axis loop</td>
<td>a character string (here 30 characters or less) that is enclosed by angle brackets, e.g. &lt;x_axis&gt;. Tags are used as placeholders in the Chemistry section and in certain PhreePlot keyword settings such as the plot title. They can be defined in a number of ways and can refer to either numeric or character variables. Tags are substituted by their current values before code is submitted either to Phreeqc for processing or to the in-built plotting routines for writing the plot file.</td>
</tr>
<tr>
<td>y-axis loop</td>
<td>the fourth innermost (and most rapidly changing) loop controlled by the &lt;y_axis&gt; tag.</td>
</tr>
<tr>
<td>z-loop</td>
<td>the second innermost loop controlled by the &lt;loop&gt; tag.</td>
</tr>
</tbody>
</table>
Appendix 2. Thermodynamic databases

The following tables show which elements are found in the various Phreeqc-format thermodynamic databases distributed with PhreePlot. The included elements are shown in red. Additional elements may be added by editing the databases or including the necessary code in an input file.

These database files can be found in the PhreePlot ‘system’ directory.

The ‘Summary of inorganic species’ for each database has been generated with the count_database_species demo.

Table A2a. Elements available in the Phreeqc.dat (USGS) database.†

<table>
<thead>
<tr>
<th>Group</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Period 2</td>
<td>Li</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Be</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Na</td>
<td>Mg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Period 3</td>
<td>K</td>
<td>Ca</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Period 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Period 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Period 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Period 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>** Lanthanoids</td>
<td>57</td>
<td>58</td>
<td>59</td>
<td>60</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td>70</td>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>** Actinoids</td>
<td>89</td>
<td>90</td>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
<td>97</td>
<td>98</td>
<td>99</td>
<td>100</td>
<td>101</td>
<td>102</td>
<td>103</td>
<td>104</td>
<td>105</td>
<td>106</td>
</tr>
</tbody>
</table>

† These files are included with the standard Phreeqc distribution. See http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/Phreeqc/.

Summary of inorganic species (elements) included

- number of primary master species = 25
- number of secondary master species = 18
- number of minerals = 57
- number of gases = 8
- number of aqueous species = 184
Table A2b. Elements available in the Amm.dat database.†

This database is similar to the original Phreeqc.dat except that Amm.dat also includes Amm as a master species and so breaks the assumed redox equilibrium between ammonium (N\(^{-3}\)) and other N species. Unlike the latest (2.17) version of Phreeqc.dat, it also excludes diffusion coefficients for some aqueous species.

Summary of inorganic species (elements) included

<table>
<thead>
<tr>
<th>Summary of inorganic species (elements)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>number of primary master species</td>
<td>25</td>
</tr>
<tr>
<td>number of secondary master species</td>
<td>17</td>
</tr>
<tr>
<td>number of minerals</td>
<td>56</td>
</tr>
<tr>
<td>number of gases</td>
<td>6</td>
</tr>
<tr>
<td>number of aqueous species</td>
<td>180</td>
</tr>
</tbody>
</table>

† This file is included with the standard Phreeqc distribution. See http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/Phreeqc/.
Table A2c. Elements available in the \texttt{wateq4f.dat} (USGS) database.†

<table>
<thead>
<tr>
<th>Period</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Li</td>
<td></td>
<td>B</td>
<td>C</td>
<td>N</td>
<td>O</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Na</td>
<td></td>
<td>Al</td>
<td>Si</td>
<td>P</td>
<td>S</td>
<td>Cl</td>
<td>Ar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>K</td>
<td>Ca</td>
<td>Sr</td>
<td>Y</td>
<td>Zr</td>
<td>Nb</td>
<td>Mo</td>
<td>Br</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Rb</td>
<td>Sr</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Lanthanoids

** Actinoids

† This file is included with the standard \texttt{Phreeqc} distribution. See \url{http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/Phreeqc/}.

Summary of inorganic species (elements) included

- number of primary master species = 33
- number of secondary master species = 29
- number of minerals = 311
- number of gases = 8
- number of aqueous species = 346
Table A2d. Elements available in the llnl.dat (Lawrence Livermore National Laboratory) database.†

<table>
<thead>
<tr>
<th>Group</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Period</td>
<td>1</td>
<td>1</td>
<td>H</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>B</td>
<td>7</td>
<td>C</td>
<td>9</td>
<td>O</td>
<td>10</td>
<td>F</td>
<td>11</td>
<td>Al</td>
<td>13</td>
<td>Si</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>Li</td>
<td>Be</td>
<td>6</td>
<td>8</td>
<td>N</td>
<td>10</td>
<td>Ne</td>
<td>12</td>
<td>Mg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>11</td>
<td>Na</td>
<td>12</td>
<td>Mg</td>
<td>13</td>
<td>Al</td>
<td>14</td>
<td>Si</td>
<td>15</td>
<td>P</td>
<td>16</td>
<td>S</td>
<td>17</td>
<td>Cl</td>
<td>18</td>
<td>Ar</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>19</td>
<td>K</td>
<td>20</td>
<td>Ca</td>
<td>21</td>
<td>Sc</td>
<td>22</td>
<td>Ti</td>
<td>23</td>
<td>V</td>
<td>24</td>
<td>Cr</td>
<td>25</td>
<td>Mn</td>
<td>26</td>
<td>Fe</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>37</td>
<td>Rb</td>
<td>38</td>
<td>Sr</td>
<td>39</td>
<td>Y</td>
<td>40</td>
<td>Zr</td>
<td>41</td>
<td>Nb</td>
<td>42</td>
<td>Mo</td>
<td>43</td>
<td>Tc</td>
<td>44</td>
<td>Ru</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>55</td>
<td>Cs</td>
<td>56</td>
<td>Ba</td>
<td>72</td>
<td>Hf</td>
<td>73</td>
<td>Ta</td>
<td>74</td>
<td>W</td>
<td>75</td>
<td>Re</td>
<td>76</td>
<td>Os</td>
<td>77</td>
<td>Ir</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>87</td>
<td>Fr</td>
<td>88</td>
<td>Ra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Lanthanoids 57 La | 58 Ce | 59 Pr | 60 Nd | 61 Pm | 62 Sm | 63 Eu | 64 Gd | 65 Tb | 66 Dy | 67 Ho | 68 Er | 69 Tm | 70 Yb | 71 Lu |

** Actinoids 89 Ac | 90 Th | 91 Pa | 92 U | 93 Np | 94 Pu | 95 Am | 96 Cm | 97 Bk | 98 Cf | 99 Es | 100 Fm | 101 Md | 102 No | 103 Lr |

† This file is included with the standard Phreeqc distribution. See http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/Phreeqc/.

Summary of inorganic species (elements) included

- number of primary master species = 81
- number of secondary master species = 137
- number of minerals = 1120
- number of gases = 91
- number of aqueous species = 1186
Table A2e. Elements available in the .minteq.dat (USEPA) database.†

<table>
<thead>
<tr>
<th>Group</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Li</td>
<td>Be</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Na</td>
<td>Mg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>K</td>
<td>Ca</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rb</td>
<td>Sr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cs</td>
<td>Ba*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fr</td>
<td>Ra**</td>
<td>**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Lanthanoids 57 La 58 Ce 59 Pr 60 Nd 61 Pm 62 Sm 63 Eu 64 Gd 65 Tb 66 Dy 67 Ho 68 Er 69 Tm 70 Yb 71 Lu

** Actinoids 89 Ac 90 Th 91 Pa 92 U 93 Np 94 Pu 95 Am 96 Cm 97 Bk 98 Cf 99 Es 100 Fm 101 Md 102 No 103 Lr

† This file is included with the standard Phreeqc distribution. See http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/Phreeqc/.

This database includes cyanide, DOM and some 30 organic ligands including EDTA, citrate and acetate.

Summary of inorganic species (elements) included

number of primary master species = 38
number of secondary master species = 41
number of minerals = 478
number of gases = 14
number of aqueous species = 484
Table A2f. Elements available in the minteq.v4.dat (USEPA) database.†

<table>
<thead>
<tr>
<th>Group</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Li</td>
<td>4</td>
<td>Be</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>Na</td>
<td>12</td>
<td>Mg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>K</td>
<td>20</td>
<td>Ca</td>
<td>21</td>
<td>Sc</td>
<td>22</td>
<td>Ti</td>
<td>23</td>
<td>V</td>
<td>24</td>
<td>Cr</td>
<td>25</td>
<td>Mn</td>
<td>26</td>
<td>Fe</td>
<td>27</td>
<td>Co</td>
</tr>
<tr>
<td>5</td>
<td>37</td>
<td>Rb</td>
<td>38</td>
<td>Sr</td>
<td>39</td>
<td>Y</td>
<td>40</td>
<td>Zr</td>
<td>41</td>
<td>Nb</td>
<td>42</td>
<td>Mo</td>
<td>43</td>
<td>Tc</td>
<td>44</td>
<td>Ru</td>
<td>45</td>
<td>Rh</td>
</tr>
<tr>
<td>6</td>
<td>55</td>
<td>Cr</td>
<td>56</td>
<td>Ba</td>
<td></td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>80</td>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>Fr</td>
<td>88</td>
<td>Ra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Lanthanoids  
** Actinoids

† This file is included with the standard Phreeqc distribution. See http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/Phreeqc/.

This database includes cyanide, cyanate and some 31 organic ligands including EDTA, citrate and acetate.

Summary of inorganic species (elements) included

number of primary master species = 40
number of secondary master species = 45
number of minerals = 541
number of gases = 15
number of aqueous species = 609
Table A2g. Elements available in the pitzer.dat (USGS) database.†

<table>
<thead>
<tr>
<th>Group</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Li</td>
<td>4</td>
<td>Be</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>Na</td>
<td>12</td>
<td>Mg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13</td>
<td>Al</td>
<td>14</td>
<td>Si</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>K</td>
<td>20</td>
<td>Ca</td>
<td>21</td>
<td>Sc</td>
<td>22</td>
<td>Ti</td>
<td>23</td>
<td>V</td>
<td>24</td>
<td>Cr</td>
<td>25</td>
<td>Mn</td>
<td>26</td>
<td>Fe</td>
<td>27</td>
<td>Co</td>
</tr>
<tr>
<td>5</td>
<td>37</td>
<td>Rb</td>
<td>38</td>
<td>Sr</td>
<td>39</td>
<td>Y</td>
<td>40</td>
<td>Zr</td>
<td>41</td>
<td>Nb</td>
<td>42</td>
<td>Mo</td>
<td>43</td>
<td>Tc</td>
<td>44</td>
<td>Ru</td>
<td>45</td>
<td>Rh</td>
</tr>
<tr>
<td>6</td>
<td>55</td>
<td>Cs</td>
<td>56</td>
<td>Ba</td>
<td>*</td>
<td>72</td>
<td>Hf</td>
<td>73</td>
<td>Ta</td>
<td>74</td>
<td>W</td>
<td>75</td>
<td>Re</td>
<td>76</td>
<td>Os</td>
<td>77</td>
<td>Ir</td>
<td>78</td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>Fr</td>
<td>88</td>
<td>Ra</td>
<td>**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Lanthanoids
** Actinoids

† This file is included with the standard Phreeqc distribution. See [http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/Phreeqc/](http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/Phreeqc/).

This database includes Pitzer coefficients and is designed to be used with the Pitzer model.

Summary of inorganic species (elements) included

- number of primary master species = 16
- number of secondary master species = 4
- number of minerals = 45
- number of gases = 2
- number of aqueous species = 26
Table A2h. Elements available in the NAPSI_290502(260802).DAT (Nagra-PSI) database.†

<table>
<thead>
<tr>
<th>Period</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Li</td>
<td>Be</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>C</td>
<td>N</td>
<td>O</td>
<td>F</td>
<td>Ne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Na</td>
<td>Mg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Al</td>
<td>Si</td>
<td>P</td>
<td>S</td>
<td>Cl</td>
<td>Ar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>K</td>
<td>Ca</td>
<td>Sc</td>
<td>Ti</td>
<td>V</td>
<td>Cr</td>
<td>Mn</td>
<td>Fe</td>
<td>Co</td>
<td>Ni</td>
<td>Cu</td>
<td>Zn</td>
<td>Ga</td>
<td>Ge</td>
<td>As</td>
<td>Se</td>
<td>Br</td>
<td>Kr</td>
</tr>
<tr>
<td>5</td>
<td>Rb</td>
<td>Sr</td>
<td>Y</td>
<td>Zr</td>
<td>Nb</td>
<td>Mo</td>
<td>Tc</td>
<td>Ru</td>
<td>Rh</td>
<td>Pd</td>
<td>Ag</td>
<td>Cd</td>
<td>In</td>
<td>Sn</td>
<td>Sb</td>
<td>Te</td>
<td>I</td>
<td>Xe</td>
</tr>
<tr>
<td>6</td>
<td>Cs</td>
<td>Ba</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Fr</td>
<td>Ra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Lanthanoids

** Actinoids


Summary of inorganic species (elements) included

- number of primary master species = 39
- number of secondary master species = 37
- number of minerals = 91
- number of gases = 6
- number of aqueous species = 391
Table A2i. Elements available in the sit.dat (ANDRA) database.†

This file is included with the standard Phreeqc distribution. See http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/Phreeqc/. It is based on the ThermoChimie v.7.b database, developed by Amphos 21, BRGM and HydrAsa for ANDRA, the French National Radioactive Waste Management Agency.

This database has been especially prepared for dealing with problems in radioactive waste management. It includes a table of SIT epsilon parameters for use with the Specific Interaction Theory (SIT) activity coefficient model of Grenthe et al. (1997).

Summary of inorganic species (elements)

number of primary master species = 54  
number of secondary master species = 55  
number of minerals = 775  
number of gases = 10  
number of aqueous species = 1013

<table>
<thead>
<tr>
<th>Group</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>20</td>
<td>56</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td>70</td>
<td>71</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
</tr>
<tr>
<td>6</td>
<td>55</td>
<td>56</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>80</td>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>88</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Lanthanoids
** Actinoids

† This file is included with the standard Phreeqc distribution. See http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/Phreeqc/. It is based on the ThermoChimie v.7.b database, developed by Amphos 21, BRGM and HydrAsa for ANDRA, the French National Radioactive Waste Management Agency.
### Table A2j. Elements available in the 050000c0.tdb (NEA) database.†

<table>
<thead>
<tr>
<th>Group</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1 H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3 Li</td>
<td>4 Be</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>11 Na</td>
<td>12 Mg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>19 K</td>
<td>20 Ca</td>
<td>21 Sc</td>
<td>22 Ti</td>
<td>23 V</td>
<td>24 Cr</td>
<td>25 Mn</td>
<td>26 Fe</td>
<td>27 Co</td>
<td>28 Ni</td>
<td>29 Cu</td>
<td>30 Zn</td>
<td>31 Ga</td>
<td>32 Ge</td>
<td>33 As</td>
<td>34 Se</td>
<td>35 Br</td>
<td>36 Kr</td>
</tr>
<tr>
<td>5</td>
<td>37 Rb</td>
<td>38 Sr</td>
<td>39 Y</td>
<td>40 Zr</td>
<td>41 Nb</td>
<td>42 Mo</td>
<td>43 Tc</td>
<td>44 Ru</td>
<td>45 Rh</td>
<td>46 Pd</td>
<td>47 Ag</td>
<td>48 Cd</td>
<td>49 In</td>
<td>50 Sn</td>
<td>51 Sb</td>
<td>52 Te</td>
<td>53 I</td>
<td>54 Xe</td>
</tr>
<tr>
<td>6</td>
<td>55 Cs</td>
<td>56 Ba</td>
<td></td>
<td>72 Hf</td>
<td>73 Ta</td>
<td>74 W</td>
<td>75 Re</td>
<td>76 Os</td>
<td>77 Ir</td>
<td>78 Pt</td>
<td>79 Au</td>
<td>80 Hg</td>
<td>81 Tl</td>
<td>82 Pb</td>
<td>83 Bi</td>
<td>84 Po</td>
<td>85 At</td>
<td>86 Rn</td>
</tr>
<tr>
<td>7</td>
<td>87 Fr</td>
<td>88 Ra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Lanthanoids: 57 La, 58 Ce, 59 Pr, 60 Nd, 61 Pm, 62 Sm, 63 Eu, 64 Gd, 65 Tb, 66 Dy, 67 Ho, 68 Er, 69 Tm, 70 Yb, 71 Lu
** Actinoids: 89 Ac, 90 Th, 91 Pa, 92 U, 93 Np, 94 Pu, 95 Am, 96 Cm, 97 Bk, 98 Cf, 99 Es, 100 Fm, 101 Md, 102 No, 103 Lr

† See [http://migrationdb.jaea.go.jp/english.html](http://migrationdb.jaea.go.jp/english.html) for the download home page and links to the privacy policy and copyright, and here for downloading this data. A variety of other related Phreeqc-format databases are available from the home page.

### Summary of inorganic species (elements) included

- number of primary master species = 54
- number of secondary master species = 4
- number of minerals = 404
- number of gases = 148
- number of aqueous species = 387
Table A2k. Elements available in the 011213c2.tdb (JAEA) database.†

<table>
<thead>
<tr>
<th>Group</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period</td>
<td>1</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Al</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Br</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>In</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Tl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Lanthanoids
  57 La
  58 Ce
  59 Pr
  60 Nd
  61 Pm
  62 Sm
  63 Eu
  64 Gd
  65 Tb
  66 Dy
  67 Ho
  68 Er
  69 Tm
  70 Yb
  71 Lu

** Actinoids
  89 Ac
  90 Th
  91 Pa
  92 U
  93 Np
  94 Pu
  95 Am
  96 Cm
  97 Bk
  98 Cn
  99 Lr
  100 Rf
  101 Db
  102 Sg
  103 Bh

† See http://migrationdb.jaea.go.jp/english.html for the home page and links to the privacy policy and copyright, and here for downloading this data. A variety of other databases are available from the home page.

This database also includes definitions for the organic ligands: oxalate, citrate and EDTA.

Summary of inorganic species (elements) included

- number of primary master species = 48
- number of secondary master species = 39
- number of minerals = 368
- number of gases = 72
- number of aqueous species = 497
Table A2k. Elements available in the 011213c2.tdb (JAEA) database.†

<table>
<thead>
<tr>
<th>Period</th>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
<th>Group 4</th>
<th>Group 5</th>
<th>Group 6</th>
<th>Group 7</th>
<th>Group 8</th>
<th>Group 9</th>
<th>Group 10</th>
<th>Group 11</th>
<th>Group 12</th>
<th>Group 13</th>
<th>Group 14</th>
<th>Group 15</th>
<th>Group 16</th>
<th>Group 17</th>
<th>Group 18</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>H</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Li</td>
<td>4</td>
<td></td>
<td>5</td>
<td>B</td>
<td>6</td>
<td>C</td>
<td>7</td>
<td>N</td>
<td>8</td>
<td>O</td>
<td>9</td>
<td>F</td>
<td>10</td>
<td>Ne</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>Na</td>
<td>12</td>
<td>Mg</td>
<td>13</td>
<td>Al</td>
<td>14</td>
<td>Si</td>
<td>15</td>
<td>P</td>
<td>16</td>
<td>S</td>
<td>17</td>
<td>Cl</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>K</td>
<td>20</td>
<td>Ca</td>
<td>21</td>
<td>Sc</td>
<td>22</td>
<td>Ti</td>
<td>23</td>
<td>V</td>
<td>24</td>
<td>Cr</td>
<td>25</td>
<td>Mn</td>
<td>26</td>
<td>Fe</td>
<td>27</td>
<td>Co</td>
</tr>
<tr>
<td>5</td>
<td>37</td>
<td>Rb</td>
<td>38</td>
<td>Sr</td>
<td>39</td>
<td>Y</td>
<td>40</td>
<td>Zr</td>
<td>41</td>
<td>Nb</td>
<td>42</td>
<td>Mo</td>
<td>43</td>
<td>Tc</td>
<td>44</td>
<td>Ru</td>
<td>45</td>
<td>Rh</td>
</tr>
<tr>
<td>6</td>
<td>55</td>
<td>Cs</td>
<td>56</td>
<td>Ba</td>
<td>57</td>
<td>72</td>
<td>73</td>
<td>Te</td>
<td>74</td>
<td>W</td>
<td>75</td>
<td>Re</td>
<td>76</td>
<td>Os</td>
<td>77</td>
<td>Ir</td>
<td>78</td>
<td>Pt</td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>Fr</td>
<td>88</td>
<td>Ra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Lanthanoids (57-71)
** Actinoids (89-103)

† See [http://migrationdb.jaea.go.jp/english.html](http://migrationdb.jaea.go.jp/english.html) for the home page and links to the privacy policy and copyright, and here for downloading this data. A variety of other databases are available from the home page.

This database also includes definitions for the organic ligands: oxalate, citrate and EDTA.

**Summary of inorganic species (elements) included**

- number of primary master species = 48
- number of secondary master species = 39
- number of minerals = 368
- number of gases = 72
- number of aqueous species = 497
Table A21. Elements available in the PCHatches.dat (NEA18, formerly Serco now Amec) database.†

<table>
<thead>
<tr>
<th>Group</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period</td>
<td>1</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Li</td>
<td>Be</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Na</td>
<td>Mg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>K</td>
<td>Ca</td>
<td>Sr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Rb</td>
<td>Sr</td>
<td>Y</td>
<td>Zr</td>
<td>Nb</td>
<td>Mo</td>
<td>Tc</td>
<td>Ru</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Cs</td>
<td>Ba</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Fr</td>
<td>Ra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Lanthanoids
** Actinoids

† This file is available from [http://www.hatches-database.com/main.htm](http://www.hatches-database.com/main.htm).

Although originally compiled for use in radiochemical modelling work, the HATCHES database also includes data suitable for many other applications e.g. toxic waste disposal, effluent treatment, chemical processing. It contains data for nearly 40 organic ligands.

Summary of inorganic species (elements) included

- number of primary master species = 59
- number of secondary master species = 96
- number of minerals = 946
- number of gases = 7
- number of aqueous species = 876
Appendix 3. Symbol numbers and names

The following symbols are available for use in PhreePlot. Symbols can be specified either by their symbol code or by their name (case independent). Enclose the name in quotes if it contains a space. See Figure 7.5 for a display of all of the symbols.

Table A3. Table showing the symbols available for plotting arranged by symbol code and name.

<table>
<thead>
<tr>
<th>font</th>
<th>symbol code</th>
<th>symbol name</th>
</tr>
</thead>
<tbody>
<tr>
<td>native</td>
<td>1</td>
<td>FILLED CIRCLE</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>FILLED SQUARE</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>FILLED TRIANGLE</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>FILLED UPSIDE DOWN TRIANGLE</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>FILLED DIAMOND</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>FILLED OCTAGON</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>OPEN CIRCLE</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>PLUS</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>MULTIPLY</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>STAR</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>NUMBER SIGN</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>PERCENT SIGN</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>ASTERISK</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>PLUS</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>MINUS</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>PERIOD</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>QUESTION MARK</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>PERPENDICULAR</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>UNDERSCORE</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>OVERSCORE</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>BAR</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>MINUTE</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>LESSEQUAL</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>CLUB</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>DIAMOND</td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>HEART</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>SPADE</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>ARROWLEFTRIGHT</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>ARROWLEFT</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>ARROWUP</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>ARROWRIGHT</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>ARROWDOWN</td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>SIMILAR</td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>COPYRIGHTSANS</td>
</tr>
<tr>
<td>font</td>
<td>symbol code</td>
<td>symbol name</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Zapf Dingbat</td>
<td>41</td>
<td>UPPER BLADE SCISSORS</td>
</tr>
<tr>
<td></td>
<td>42</td>
<td>BLACK SCISSORS</td>
</tr>
<tr>
<td></td>
<td>43</td>
<td>LOWER BLADE SCISSORS</td>
</tr>
<tr>
<td></td>
<td>44</td>
<td>WHITE SCISSORS</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>BLACK TELEPHONE</td>
</tr>
<tr>
<td></td>
<td>46</td>
<td>TELEPHONE LOCATION SIGN</td>
</tr>
<tr>
<td></td>
<td>47</td>
<td>TAPE DRIVE</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>AIRPLANE</td>
</tr>
<tr>
<td></td>
<td>51</td>
<td>ENVELOPE</td>
</tr>
<tr>
<td></td>
<td>52</td>
<td>BLACK POINTING RIGHT INDEX</td>
</tr>
<tr>
<td></td>
<td>53</td>
<td>WHITE POINTING RIGHT INDEX</td>
</tr>
<tr>
<td></td>
<td>54</td>
<td>VICTORY HAND</td>
</tr>
<tr>
<td></td>
<td>55</td>
<td>WRITING HAND</td>
</tr>
<tr>
<td></td>
<td>56</td>
<td>LOWER RIGHT PENCIL</td>
</tr>
<tr>
<td></td>
<td>57</td>
<td>PENCIL</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>UPPER RIGHT PENCIL</td>
</tr>
<tr>
<td></td>
<td>61</td>
<td>WHITE NIB</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>HEAVY BALLOT X</td>
</tr>
<tr>
<td></td>
<td>71</td>
<td>OUTLINED GREEK CROSS</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>HEAVY GREEK CROSS</td>
</tr>
<tr>
<td></td>
<td>73</td>
<td>OPEN CENTRE CROSS</td>
</tr>
<tr>
<td></td>
<td>74</td>
<td>HEAVY OPEN CENTRE CROSS</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>LATIN CROSS</td>
</tr>
<tr>
<td></td>
<td>76</td>
<td>SHADOWED WHITE LATIN CROSS</td>
</tr>
<tr>
<td></td>
<td>77</td>
<td>OUTLINED LATIN CROSS</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>MALTESE CROSS</td>
</tr>
<tr>
<td></td>
<td>101</td>
<td>STAR OF DAVID</td>
</tr>
<tr>
<td></td>
<td>102</td>
<td>FOUR TEARDROP-SPOKED ASTERISK</td>
</tr>
<tr>
<td></td>
<td>103</td>
<td>FOUR BALLOON-SPOKED ASTERISK</td>
</tr>
<tr>
<td></td>
<td>104</td>
<td>HEAVY FOUR BALLOON-SPOKED ASTERISK</td>
</tr>
<tr>
<td></td>
<td>105</td>
<td>FOUR CLUB-SPOKED ASTERISK</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td>BLACK FOUR POINTED STAR</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td>STRESS OUTLINED WHITE STAR</td>
</tr>
<tr>
<td></td>
<td>112</td>
<td>CIRCELLED WHITE STAR</td>
</tr>
<tr>
<td></td>
<td>113</td>
<td>OPEN CENTRE BLACK STAR</td>
</tr>
<tr>
<td></td>
<td>114</td>
<td>BLACK CENTRE WHITE STAR</td>
</tr>
<tr>
<td></td>
<td>115</td>
<td>OUTLINED BLACK STAR</td>
</tr>
<tr>
<td></td>
<td>116</td>
<td>HEAVY OUTLINED BLACK STAR</td>
</tr>
<tr>
<td></td>
<td>117</td>
<td>PINWHEEL STAR</td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>SHADOWED WHITE STAR</td>
</tr>
<tr>
<td></td>
<td>121</td>
<td>HEAVY ASTERISK</td>
</tr>
<tr>
<td></td>
<td>122</td>
<td>OPEN CENTRE ASTERISK</td>
</tr>
<tr>
<td></td>
<td>123</td>
<td>EIGHT SPOKED ASTERISK</td>
</tr>
<tr>
<td></td>
<td>124</td>
<td>EIGHT POINTED BLACK STAR</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>EIGHT POINTED PINWHEEL STAR</td>
</tr>
<tr>
<td></td>
<td>126</td>
<td>SIX POINTED BLACK STAR</td>
</tr>
<tr>
<td></td>
<td>127</td>
<td>EIGHT POINTED RECTILINEAR BLACK STAR</td>
</tr>
<tr>
<td></td>
<td>130</td>
<td>HEAVY EIGHT POINTED RECTILINEAR BLACK STAR</td>
</tr>
<tr>
<td></td>
<td>131</td>
<td>TWELVE POINTED BLACK STAR</td>
</tr>
<tr>
<td></td>
<td>132</td>
<td>SIXTEEN POINTED ASTERISK</td>
</tr>
<tr>
<td></td>
<td>133</td>
<td>TEARDROP-SPOKED ASTERISK</td>
</tr>
<tr>
<td></td>
<td>134</td>
<td>OPEN CENTRE TEARDROP-SPOKED ASTERISK</td>
</tr>
<tr>
<td></td>
<td>135</td>
<td>HEAVY TEARDROP-SPOKED ASTERISK</td>
</tr>
<tr>
<td></td>
<td>136</td>
<td>SIX PETALLED BLACK AND WHITE FLORETTE</td>
</tr>
<tr>
<td></td>
<td>137</td>
<td>BLACK FLORETTE</td>
</tr>
<tr>
<td></td>
<td>140</td>
<td>WHITE FLORETTE</td>
</tr>
<tr>
<td></td>
<td>141</td>
<td>EIGHT PETALLED OUTLINED BLACK FLORETTE</td>
</tr>
<tr>
<td></td>
<td>142</td>
<td>CIRCELLED OPEN CENTRE EIGHT POINTED STAR</td>
</tr>
<tr>
<td></td>
<td>143</td>
<td>HEAVY TEARDROP-SPOKED PINWHEEL ASTERISK</td>
</tr>
<tr>
<td></td>
<td>144</td>
<td>SNOWFLAKE</td>
</tr>
<tr>
<td></td>
<td>145</td>
<td>TIGHT TRIFOLIATE SNOWFLAKE</td>
</tr>
<tr>
<td></td>
<td>146</td>
<td>HEAVY CHEVRON SNOWFLAKE</td>
</tr>
<tr>
<td></td>
<td>147</td>
<td>SPARKLE</td>
</tr>
<tr>
<td>font</td>
<td>symbol code</td>
<td>symbol name</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>Zapf Dingbat</td>
<td>150</td>
<td>HEAVY SPARKLE</td>
</tr>
<tr>
<td></td>
<td>151</td>
<td>BALLOON-SPOKED ASTERISK</td>
</tr>
<tr>
<td></td>
<td>152</td>
<td>EIGHT TEARDROP-SPOKED PROPELLER ASTERISK</td>
</tr>
<tr>
<td></td>
<td>153</td>
<td>HEAVY EIGHT TEARDROP-SPOKED PROPELLER ASTERISK</td>
</tr>
<tr>
<td></td>
<td>154</td>
<td>BLACK CIRCLE</td>
</tr>
<tr>
<td></td>
<td>155</td>
<td>SHADOWED WHITE CIRCLE</td>
</tr>
<tr>
<td></td>
<td>156</td>
<td>BLACK SQUARE</td>
</tr>
<tr>
<td></td>
<td>157</td>
<td>LOWER RIGHT DROP-SHADOWED WHITE SQUARE</td>
</tr>
<tr>
<td></td>
<td>158</td>
<td>UPPER RIGHT DROP-SHADOWED WHITE SQUARE</td>
</tr>
<tr>
<td></td>
<td>159</td>
<td>LOWER RIGHT SHADOWED WHITE SQUARE</td>
</tr>
<tr>
<td></td>
<td>160</td>
<td>UPPER RIGHT SHADOWED WHITE SQUARE</td>
</tr>
<tr>
<td></td>
<td>161</td>
<td>BLACK UP-POINTING TRIANGLE</td>
</tr>
<tr>
<td></td>
<td>162</td>
<td>BLACK DOWN-POINTING TRIANGLE</td>
</tr>
<tr>
<td></td>
<td>163</td>
<td>BLACK DIAMOND</td>
</tr>
<tr>
<td></td>
<td>164</td>
<td>BLACK DIAMOND MINUS WHITE X</td>
</tr>
<tr>
<td></td>
<td>165</td>
<td>RIGHT BLACK SEMICIRCLE</td>
</tr>
<tr>
<td></td>
<td>166</td>
<td>LIGHT VERTICAL BAR</td>
</tr>
<tr>
<td></td>
<td>167</td>
<td>MEDIUM VERTICAL BAR</td>
</tr>
<tr>
<td></td>
<td>168</td>
<td>HEAVY VERTICAL BAR</td>
</tr>
<tr>
<td></td>
<td>169</td>
<td>HEAVY SINGLE TURNED COMMA QUOTATION MARK ORNAMENT</td>
</tr>
<tr>
<td></td>
<td>170</td>
<td>HEAVY SINGLE COMMA QUOTATION MARK ORNAMENT</td>
</tr>
<tr>
<td></td>
<td>171</td>
<td>HEAVY DOUBLE TURNED COMMA QUOTATION MARK ORNAMENT</td>
</tr>
<tr>
<td></td>
<td>172</td>
<td>HEAVY DOUBLE COMMA QUOTATION MARK ORNAMENT</td>
</tr>
<tr>
<td></td>
<td>173</td>
<td>CURVED STEM PARAGRAPH SIGN ORNAMENT</td>
</tr>
<tr>
<td></td>
<td>174</td>
<td>HEAVY EXCLAMATION MARK ORNAMENT</td>
</tr>
<tr>
<td></td>
<td>175</td>
<td>HEAVY HEART EXCLAMATION MARK ORNAMENT</td>
</tr>
<tr>
<td></td>
<td>176</td>
<td>HEAVY BLACK HEART</td>
</tr>
<tr>
<td></td>
<td>177</td>
<td>ROTATED HEAVY BLACK HEART BULLET</td>
</tr>
<tr>
<td></td>
<td>178</td>
<td>FLORAL HEART</td>
</tr>
<tr>
<td></td>
<td>179</td>
<td>ROTATED FLORAL HEART BULLET</td>
</tr>
<tr>
<td></td>
<td>180</td>
<td>BLACK CLUB SUIT</td>
</tr>
<tr>
<td></td>
<td>181</td>
<td>BLACK DIAMOND SUIT</td>
</tr>
<tr>
<td></td>
<td>182</td>
<td>BLACK HEART SUIT</td>
</tr>
<tr>
<td></td>
<td>183</td>
<td>BLACK SPADE SUIT</td>
</tr>
<tr>
<td></td>
<td>184</td>
<td>DINGBAT CIRCLED DIGIT ONE</td>
</tr>
<tr>
<td></td>
<td>185</td>
<td>DINGBAT CIRCLED DIGIT TWO</td>
</tr>
<tr>
<td></td>
<td>186</td>
<td>DINGBAT CIRCLED DIGIT THREE</td>
</tr>
<tr>
<td></td>
<td>187</td>
<td>DINGBAT CIRCLED DIGIT FOUR</td>
</tr>
<tr>
<td></td>
<td>188</td>
<td>DINGBAT CIRCLED DIGIT FIVE</td>
</tr>
<tr>
<td></td>
<td>189</td>
<td>DINGBAT CIRCLED DIGIT SIX</td>
</tr>
<tr>
<td></td>
<td>190</td>
<td>DINGBAT CIRCLED DIGIT SEVEN</td>
</tr>
<tr>
<td></td>
<td>191</td>
<td>DINGBAT CIRCLED DIGIT EIGHT</td>
</tr>
<tr>
<td></td>
<td>192</td>
<td>DINGBAT CIRCLED DIGIT NINE</td>
</tr>
<tr>
<td></td>
<td>193</td>
<td>DINGBAT CIRCLED NUMBER TEN</td>
</tr>
<tr>
<td></td>
<td>194</td>
<td>DINGBAT NEGATIVE CIRCLED DIGIT ONE</td>
</tr>
<tr>
<td></td>
<td>195</td>
<td>DINGBAT NEGATIVE CIRCLED DIGIT TWO</td>
</tr>
<tr>
<td></td>
<td>196</td>
<td>DINGBAT NEGATIVE CIRCLED DIGIT THREE</td>
</tr>
<tr>
<td></td>
<td>197</td>
<td>DINGBAT NEGATIVE CIRCLED DIGIT FOUR</td>
</tr>
<tr>
<td></td>
<td>198</td>
<td>DINGBAT NEGATIVE CIRCLED DIGIT FIVE</td>
</tr>
<tr>
<td></td>
<td>199</td>
<td>DINGBAT NEGATIVE CIRCLED DIGIT SIX</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>DINGBAT NEGATIVE CIRCLED DIGIT SEVEN</td>
</tr>
<tr>
<td></td>
<td>201</td>
<td>DINGBAT NEGATIVE CIRCLED DIGIT EIGHT</td>
</tr>
<tr>
<td></td>
<td>202</td>
<td>DINGBAT NEGATIVE CIRCLED DIGIT NINE</td>
</tr>
<tr>
<td></td>
<td>203</td>
<td>DINGBAT CIRCLED SANS-SERIF DIGIT ONE</td>
</tr>
<tr>
<td></td>
<td>204</td>
<td>DINGBAT CIRCLED SANS-SERIF DIGIT TWO</td>
</tr>
<tr>
<td></td>
<td>205</td>
<td>DINGBAT CIRCLED SANS-SERIF DIGIT THREE</td>
</tr>
<tr>
<td></td>
<td>206</td>
<td>DINGBAT CIRCLED SANS-SERIF DIGIT FOUR</td>
</tr>
<tr>
<td></td>
<td>207</td>
<td>DINGBAT CIRCLED SANS-SERIF DIGIT FIVE</td>
</tr>
<tr>
<td></td>
<td>208</td>
<td>DINGBAT CIRCLED SANS-SERIF DIGIT SIX</td>
</tr>
<tr>
<td></td>
<td>209</td>
<td>DINGBAT CIRCLED SANS-SERIF DIGIT SEVEN</td>
</tr>
<tr>
<td></td>
<td>210</td>
<td>DINGBAT CIRCLED SANS-SERIF DIGIT EIGHT</td>
</tr>
<tr>
<td></td>
<td>211</td>
<td>DINGBAT CIRCLED SANS-SERIF DIGIT NINE</td>
</tr>
<tr>
<td>font</td>
<td>symbol code</td>
<td>symbol name</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------</td>
</tr>
<tr>
<td>Zapf Dingbat</td>
<td>311</td>
<td>DINGBAT CIRCLED SANS-SERIF NUMBER TEN</td>
</tr>
<tr>
<td></td>
<td>312</td>
<td>DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT ONE</td>
</tr>
<tr>
<td></td>
<td>313</td>
<td>DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT TWO</td>
</tr>
<tr>
<td></td>
<td>314</td>
<td>DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT THREE</td>
</tr>
<tr>
<td></td>
<td>315</td>
<td>DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT FOUR</td>
</tr>
<tr>
<td></td>
<td>316</td>
<td>DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT FIVE</td>
</tr>
<tr>
<td></td>
<td>317</td>
<td>DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT SIX</td>
</tr>
<tr>
<td></td>
<td>318</td>
<td>DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT SEVEN</td>
</tr>
<tr>
<td></td>
<td>319</td>
<td>DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT EIGHT</td>
</tr>
<tr>
<td></td>
<td>320</td>
<td>DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGIT NINE</td>
</tr>
<tr>
<td></td>
<td>321</td>
<td>DINGBAT NEGATIVE CIRCLED SANS-SERIF NUMBER TEN</td>
</tr>
<tr>
<td></td>
<td>322</td>
<td>HEAVY WIDE-HEADED RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>323</td>
<td>HEAVY SOUTH EAST ARROW</td>
</tr>
<tr>
<td></td>
<td>324</td>
<td>HEAVY NORTH EAST ARROW</td>
</tr>
<tr>
<td></td>
<td>325</td>
<td>HEAVY NORTH WEST ARROW</td>
</tr>
<tr>
<td></td>
<td>326</td>
<td>HEAVY WEST NORTH ARROW</td>
</tr>
<tr>
<td></td>
<td>327</td>
<td>HEAVY SOUTH EAST ARROW</td>
</tr>
<tr>
<td></td>
<td>328</td>
<td>HEAVY SOUTH WEST ARROW</td>
</tr>
<tr>
<td></td>
<td>329</td>
<td>HEAVY WEST SOUTH ARROW</td>
</tr>
<tr>
<td></td>
<td>330</td>
<td>DRAFTING POINT RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>331</td>
<td>HEAVY ROUND-TIPPED RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>332</td>
<td>TRIANGLE-HEADED RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>333</td>
<td>HEAVY TRIANGLE-HEADED RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>334</td>
<td>DASHED TRIANGLE-HEADED RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>335</td>
<td>HEAVY DASHED TRIANGLE-HEADED RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>336</td>
<td>BLACK RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>337</td>
<td>THREE-D TOP-LIGHTED RIGHTWARDS ARROWHEAD</td>
</tr>
<tr>
<td></td>
<td>338</td>
<td>THREE-D BOTTOM-LIGHTED RIGHTWARDS ARROWHEAD</td>
</tr>
<tr>
<td></td>
<td>339</td>
<td>BLACK RIGHTWARDS ARROW HEAD</td>
</tr>
<tr>
<td></td>
<td>340</td>
<td>HEAVY BLACK CURVED DOWNWARDS AND RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>341</td>
<td>HEAVY BLACK CURVED UPWARDS AND RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>342</td>
<td>HEAVY Black Curved Downwards and Rightwards Arrow</td>
</tr>
<tr>
<td></td>
<td>343</td>
<td>SOUAT BLACK RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>344</td>
<td>HEAVY CONCAVE-POINTED BLACK RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>345</td>
<td>RIGHT-SHADED WHITE RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>346</td>
<td>LEFT-SHADED WHITE RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>347</td>
<td>BACK-TILTED SHADOWED WHITE RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>348</td>
<td>FRONT-TILTED SHADOWED WHITE RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>349</td>
<td>HEAVY LOWER RIGHT-SHADED WHITE RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>350</td>
<td>HEAVY UPPER RIGHT-SHADED WHITE RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>351</td>
<td>NOTCHED LOWER RIGHT-SHADED WHITE RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>352</td>
<td>NOTCHED UPPER RIGHT-SHADED WHITE RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>353</td>
<td>CIRCLED HEAVY WHITE RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>354</td>
<td>WHITE-FEATHERED RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>355</td>
<td>BLACK-FEATHERED RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>356</td>
<td>BLACK-FEATHERED SOUTH EAST ARROW</td>
</tr>
<tr>
<td></td>
<td>357</td>
<td>BLACK-FEATHERED NORTH EAST ARROW</td>
</tr>
<tr>
<td></td>
<td>358</td>
<td>HEAVY BLACK-FEATHERED SOUTH EAST ARROW</td>
</tr>
<tr>
<td></td>
<td>359</td>
<td>HEAVY BLACK-FEATHERED NORTH EAST ARROW</td>
</tr>
<tr>
<td></td>
<td>360</td>
<td>TEARDROP-BARBED RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>361</td>
<td>HEAVY TEARDROP-SHANKED RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>362</td>
<td>WEDGE-TAILED RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>363</td>
<td>HEAVY WEDGE-TAILED RIGHTWARDS ARROW</td>
</tr>
<tr>
<td></td>
<td>364</td>
<td>OPEN-OUTLINED RIGHTWARDS ARROW</td>
</tr>
</tbody>
</table>
Appendix 4. The Standard and Latin-1 character sets

The following tables show the characters, and their decimal and octal codings, that are available with the Latin-1 (ISO-8859-1) (default in PhreePlot) and ‘Standard’ encodings. The Latin-1 encoding is often named ‘ANSI’ (c.f. ASCII) although it is not actually a proper ANSI standard. These character sets are enabled with the font keyword either

font Standard
or
font Latin-1

The font can also be changed at the same time, e.g.

font Roman Latin-1

The following tables show the character sets available on a ‘typical’ PC (mine!). It is possible to reproduce the tables to show your particular setup by setting plotTitle to ‘character set’ and using ‘A4’ or ‘letter’ size paper and using the font keyword to select the encoding. The following input should produce a character table for Latin-1:

calculationType                      custom
plotTitle                            "character set" # special case
labelsize                            2 # sets text size in mm
font                                 Latin-1 # or Standard

See “Accented and other ‘foreign’ characters - the Latin-1 encoding”, p. 84 for how to enter characters not present on your keyboard.

The ASCII encoding points to the 7-bit ASCII character set and consists of only the first two columns in the tables below (decimal codes 0-127). This was the character set used in earlier versions of PhreePlot.

The extended characters with decimal codes 129-137 are used internally by PhreePlot to code subscripts, superscripts etc and should not be used in text strings.

Greek characters

There are a couple of Greek characters in the Latin-1 character set but a full set can be found in the symbols font. These are entered singly with the backslash-character format, e.g. \p for \pi, or using the Greek pair of tags, <g>....</g>, for one or more characters. The codes used to specify the Greek characters are given below.

\alpha \beta \chi \delta \phi \eta \iota \kappa \lambda \mu \nu \pi \rho \sigma \tau \upsilon \omega \psi \varsigma

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

\Alpha \Beta \Chi \Delta \Epsilon \Eta \Iota \Kappa \Lambda \Mu \Nu \Pi \Rho \Sigma \Tau \Upsilon \Omega \Psi \Varepsilon
`Standard' encoding

<table>
<thead>
<tr>
<th>dec</th>
<th>oct</th>
<th>chr</th>
<th>dec</th>
<th>oct</th>
<th>chr</th>
<th>dec</th>
<th>oct</th>
<th>chr</th>
<th>dec</th>
<th>oct</th>
<th>chr</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000</td>
<td>@</td>
<td>0001</td>
<td>001</td>
<td>A</td>
<td>0002</td>
<td>002</td>
<td>B</td>
<td>0003</td>
<td>003</td>
<td>C</td>
</tr>
<tr>
<td>0004</td>
<td>004</td>
<td>D</td>
<td>0005</td>
<td>005</td>
<td>E</td>
<td>0006</td>
<td>006</td>
<td>F</td>
<td>0007</td>
<td>007</td>
<td>G</td>
</tr>
<tr>
<td>0008</td>
<td>008</td>
<td>H</td>
<td>0009</td>
<td>009</td>
<td>I</td>
<td>0010</td>
<td>010</td>
<td>J</td>
<td>0011</td>
<td>011</td>
<td>K</td>
</tr>
<tr>
<td>0012</td>
<td>012</td>
<td>L</td>
<td>0013</td>
<td>013</td>
<td>M</td>
<td>0014</td>
<td>014</td>
<td>N</td>
<td>0015</td>
<td>015</td>
<td>O</td>
</tr>
<tr>
<td>0016</td>
<td>016</td>
<td>P</td>
<td>0017</td>
<td>017</td>
<td>Q</td>
<td>0018</td>
<td>018</td>
<td>R</td>
<td>0019</td>
<td>019</td>
<td>S</td>
</tr>
<tr>
<td>0020</td>
<td>020</td>
<td>T</td>
<td>0021</td>
<td>021</td>
<td>U</td>
<td>0022</td>
<td>022</td>
<td>V</td>
<td>0023</td>
<td>023</td>
<td>W</td>
</tr>
<tr>
<td>0024</td>
<td>024</td>
<td>X</td>
<td>0025</td>
<td>025</td>
<td>Y</td>
<td>0026</td>
<td>026</td>
<td>Z</td>
<td>0027</td>
<td>027</td>
<td>a</td>
</tr>
<tr>
<td>0028</td>
<td>028</td>
<td>b</td>
<td>0029</td>
<td>029</td>
<td>c</td>
<td>0030</td>
<td>030</td>
<td>d</td>
<td>0031</td>
<td>031</td>
<td>e</td>
</tr>
<tr>
<td>0032</td>
<td>032</td>
<td>f</td>
<td>0033</td>
<td>033</td>
<td>g</td>
<td>0034</td>
<td>034</td>
<td>h</td>
<td>0035</td>
<td>035</td>
<td>i</td>
</tr>
<tr>
<td>0036</td>
<td>036</td>
<td>j</td>
<td>0037</td>
<td>037</td>
<td>k</td>
<td>0038</td>
<td>038</td>
<td>l</td>
<td>0039</td>
<td>039</td>
<td>m</td>
</tr>
<tr>
<td>0040</td>
<td>040</td>
<td>n</td>
<td>0041</td>
<td>041</td>
<td>o</td>
<td>0042</td>
<td>042</td>
<td>p</td>
<td>0043</td>
<td>043</td>
<td>q</td>
</tr>
<tr>
<td>0044</td>
<td>044</td>
<td>r</td>
<td>0045</td>
<td>045</td>
<td>s</td>
<td>0046</td>
<td>046</td>
<td>t</td>
<td>0047</td>
<td>047</td>
<td>u</td>
</tr>
<tr>
<td>0048</td>
<td>048</td>
<td>v</td>
<td>0049</td>
<td>049</td>
<td>w</td>
<td>0050</td>
<td>050</td>
<td>x</td>
<td>0051</td>
<td>051</td>
<td>y</td>
</tr>
<tr>
<td>0052</td>
<td>052</td>
<td>z</td>
<td>0053</td>
<td>053</td>
<td>{</td>
<td>0054</td>
<td>054</td>
<td></td>
<td></td>
<td>0055</td>
<td>055</td>
</tr>
<tr>
<td>0056</td>
<td>056</td>
<td>^</td>
<td>0057</td>
<td>057</td>
<td>_</td>
<td>0058</td>
<td>058</td>
<td>`</td>
<td>0059</td>
<td>059</td>
<td>a</td>
</tr>
<tr>
<td>0060</td>
<td>060</td>
<td>å</td>
<td>0061</td>
<td>061</td>
<td>Æ</td>
<td>0062</td>
<td>062</td>
<td>æ</td>
<td>0063</td>
<td>063</td>
<td>Æ</td>
</tr>
<tr>
<td>0064</td>
<td>064</td>
<td>Å</td>
<td>0065</td>
<td>065</td>
<td>Æ</td>
<td>0066</td>
<td>066</td>
<td>æ</td>
<td>0067</td>
<td>067</td>
<td>Æ</td>
</tr>
<tr>
<td>0068</td>
<td>068</td>
<td>²</td>
<td>0069</td>
<td>069</td>
<td>£</td>
<td>0070</td>
<td>070</td>
<td>²</td>
<td>0071</td>
<td>071</td>
<td>¢</td>
</tr>
<tr>
<td>0072</td>
<td>072</td>
<td>¥</td>
<td>0073</td>
<td>073</td>
<td>€</td>
<td>0074</td>
<td>074</td>
<td>¥</td>
<td>0075</td>
<td>075</td>
<td>£</td>
</tr>
<tr>
<td>0076</td>
<td>076</td>
<td>²</td>
<td>0077</td>
<td>077</td>
<td>¢</td>
<td>0078</td>
<td>078</td>
<td>¥</td>
<td>0079</td>
<td>079</td>
<td>€</td>
</tr>
</tbody>
</table>

Note: The table above lists some standard characters and their corresponding ASCII codes.