# Development and Application of speleothem-based proxies for Past Climate Change

**Robert A. Owen**

Thesis submitted to the University of Oxford

for the degree of Doctor of Philosophy




Department of Earth Sciences

and

University of Oxford

Trinity 2017

Supervised by Dr. Chris C. Day & Prof. Gideon M. Henderson

# Abstract

This thesis develops and applies new techniques for reconstructing past environments from the speleothem archive. Chapter 3 introduces Ca isotopes as a novel palaeo-aridity proxy. Recent dripwater, farmed calcite and bedrock data show that, in the modern environment, 36 % of initial dripwater Ca is removed by prior calcite precipitation at the HS4 dripsite (Heshang Cave, Central China). A speleothem record spanning the 8.2kyr event suggests this value increased to 60 % during the event, in response to a decrease in effective rainfall of approximately a third.

Modern Ca isotope systematics, and the cycling of other Group II metals, are studied further in Chapter 4. Data from Heshang Cave monitoring samples explore the temporal and spatial controls on these proxy systems. Results highlight the importance of secondary calcite as a potential source of dripwater metals, as well as the significant contribution of particulate calcite to bulk dripwater chemistry. At the HS4 dripsite, over a third of bulk dripwater Ca is present as particulate calcite. This influences bulk dripwater Ca concentrations and isotopes, as well as potentially impacting speleothem growth and chemistry.

Chapters 5 introduces CaveCalc, a new numerical model for dripwater and speleothem chemistry and isotopes based on PHREEQC. Key strengths of CaveCalc include its ability to model multiple proxy systems in a single framework, the ability to quantify the extent of open-system dissolution, and the extensible nature of its design.

Chapter 6 applies CaveCalc to disentangling the controls on dripwater and speleothem $\delta^{13}C$ values. Model results are compared with speleothem data. At Heshang Cave, anomalously high $\delta^{13}C$ is quantitatively explained as the result of $CO_2$ degassing and prior calcite precipitation, an interpretation made possible by the availability of coupled $\delta^{13}C$, $a^{14}C$ and Ca isotope data. This approach provides a powerful tool to help researchers better interpret dripwater and speleothem $\delta^{13}C$ data.

# Extended Abstract

This thesis develops and applies new techniques for reconstructing past climates and environments from the speleothem archive. These include laboratory analyses, where my work has focused on the development of Ca isotopes as a novel proxy for PCP and aridity. The results of this work are discussed in Chapters 3 and 4, which use Heshang Cave (Central China) as the study site.

The second part of this thesis focuses on numerical modelling of speleothem and dripwater chemistry. Complex multi-proxy datasets are increasingly available, both as time-series of speleothem analyses, and also from cave monitoring programmes designed to better understand the modern environment. To aid in interpretation of such datasets, I have developed a new numerical model for cave geochemistry ('CaveCalc'). The design and capabilities of the model are discussed in Chapter 5, and the model is applied to modelling speleothem and dripwater $\delta^{13}C$ data in Chapter 6.

Chapter 3 presents the first systematic study of Ca isotopes in a cave system. Dripwaters at Heshang Cave (Central China; 30°27′ N, 110°25′ E) are enriched in Ca isotopes relative to bedrock due to prior calcite precipitation (PCP) occurring earlier in the drip flow path. Coeval measurements of Ca isotopes from dripwater and calcite formed on glass plates indicate a site-specific Ca isotope fractionation factor, $\Delta^{44/42}Ca$, of $-0.63$ ‰, which agrees well with other determinations of the fractionation between calcite and water. A simple Rayleigh fractionation model indicates that ∼36 % of the initial dripwater Ca is removed by PCP in the modern environment prior to the site of stalagmite formation.

In a stalagmite growing at the same site as this modern study, Ca isotopes show a positive excursion during the aridity of the 8.2kyr event, correlating well with positive excursions in established PCP proxies (Mg/Ca, Sr/Ca and Ba/Ca). This record is used to quantify PCP across the 8.2kyr event, and shows that the extent of PCP increased from the modern-like value of 40 % to 60 % Ca removal. This increase in PCP suggests, for a simple one-box model of the aquifer system, a reduction in mean annual rainfall in this region of Central China by a third during the 8.2kyr event.

Ca isotopes are also used to identify non-PCP controls on speleothem trace element chemistry. In particular, the Mg/Ca response during the 8.2kyr event may be affected by changing source composition or temperature. This has important implications for Mg/Ca-based PCP reconstructions.

This initial study demonstrates the potential of Ca isotope records in quantifying PCP from speleothem data — a key step towards rainfall reconstructions. Ca isotopes are less susceptible to variable source composition and detrital contamination than trace element

proxies, making them particularly useful on long timescales or in 'dirty' speleothems where trace element chemistry is influenced by non-carbonate phases.

Chapter 4 extends the Ca isotope monitoring record of Chapter 3 with data from the period 2011–2013 and from additional dripsites within Heshang Cave. Ca isotope measurements are supplemented by cation analyses to allow a more complete discussion of speleothem metal sourcing. Additionally, dripwater filtration experiments were performed to investigate the contribution of any particulate phase to Heshang dripwater metal concentrations. Together, these analyses provide a more complete description of metal sourcing at Heshang Cave, and may help improve understanding of the processes occurring in the unsaturated zone. Such an understanding is directly relevant to the interpretation of speleothem metal isotope and trace element data, as well as comparisons between speleothems within a cave (e.g. to make composite records).

Key findings relating to metal cycling at Heshang Cave are as follows:

1. Unfiltered dripwater $\delta^{44/42}\mathrm{Ca} = 0.40\,‰$ and glass plate $\delta^{44/42}\mathrm{Ca} = 0.03\,‰$ across the interval 2011–2013 at the HS4 drip-site. There is no evidence for seasonal controls on the Ca isotopic composition of dripwaters or glass plate calcite. This gives a revised fractionation factor of $\Delta^{44/42}\mathrm{Ca}_{calcite-water} = -0.36\,‰$.

2. Dripwater trace elements at the HS4 drip-site suggest sources other than the dolomite bedrock contribute significantly to dripwater metal chemistry. In particular, lower than expected Mg/Ca ratios are best explained by dissolution of calcium carbonate, either primary or secondary deposits, as a significant source of dripwater Ca. Secondary calcite deposits are the most likely candidate, and would predict an initial dripwater Ca isotopic composition lighter than expected from quantitative dolomite dissolution alone.

3. Ca isotopes, Mg/Ca, Sr/Ca and Ba/Ca vary greatly between drip-sites within Heshang Cave. X/Ca ratios and Ca concentrations suggest that sites deeper within the cave undergo more extensive $CO_2$ degassing and PCP due to the greater thickness of overlying rock and hence longer drip flow paths, however Ca isotopes show a more complex pattern.

4. Filtering of dripwaters reveals a significant particulate contribution to dripwater metal chemistry at the HS4 drip site, but not at other sites in Heshang Cave. These particles are likely calcite and represent 36 % of total dripwater Ca, significantly affecting bulk solution Ca concentration and isotopes. Their impact on speleothem growth and chemistry remain unknown and require further study.

This work highlights two previously unconsidered controls on the Ca isotope PCP proxy: secondary calcite in dolomite cave systems, and carbonate particulates in dripwater. Future work should take note of these factors and ensure they are well quantified for the study site in question. Dripwater filtration is recommended prior to Ca isotope analysis for PCP reconstruction as only the dissolved Ca pool is expected to fractionate during PCP.

Chapter 5 introduces and describes CaveCalc, a new numerical model for dripwater chemistry and isotopes. The model uses PHREEQC to perform various equilibrium chemical and isotopic calculations between solutions, minerals and gases of interest to understanding the geochemistry of cave systems. This model may be used to aid interpretation of multi-proxy datasets from dripwater monitoring programs or speleothem time-series.

CaveCalc can model a set of chemical reactions that, when linked together, simulate various aspects of the soil-karst-cave system. CaveCalc may be expanded to include new reactions and isotope systems as needed. Key chemical reactions available are: $CO_2$ equilibration between DIC and $CO_{2(g)}$, calcite dissolution, dolomite dissolution, pyrite oxidation, calcite precipitation, equilibrium $CO_2$ degassing and kinetic $CO_2$ degassing.

These basic reactions may be combined or run in sequence to describe and simulate a variety of soil and karst processes of interest to speleothem researchers including equilibration of soil gas and soil water, $CO_2$-driven bedrock dissolution (in the presence of a gas phase), pyrite-oxidation-driven bedrock dissolution, incongruent dolomite dissolution, water-rock interactions, $CO_2$ degassing and PCP. Through all these reactions, full solution chemistry is tracked (including DIC speciation, pH & alkalinity), along with C, O and Ca isotopes. Mg, Sr and Ba partitioning into calcite are also included. Key strengths of CaveCalc include its ability to model multiple proxy systems in a single framework, the ability to quantify the extent of open-system dissolution, and the extensible nature of its PHREEQC-based design. Future work can expand and improve the model as new data become available and new reactions are identified as important in the soil-karst-cave system.

Stalagmite $\delta^{13}C$ data has not been as widely interpreted as $\delta^{18}O$ data due difficulty in separating the many processes that control carbon isotopes in the karst environment. The increasing availability of speleothem radiocarbon data, and recent demonstration of the power of calcium isotopes, may enable the processes controlling $\delta^{13}C$ to be untangled. In Chapter 6, I use CaveCalc to quantify and separate the controls on carbon isotopes in cave carbonates and dripwaters.

Modelling bedrock dissolution under typical conditions suggests that, following closed-system dissolution, initial dripwater $\delta^{13}C$ falls in the range $-16$ to $-13\,‰$, dead carbon percent (DCP) in range 40–50 % and dripwater Ca concentration in the range 0.1–2.7 mmol kg$^{-1}$. Open-system dissolution predicts $\delta^{13}C$ in the range $-20$ to $-17\,‰$, DCP = 0 %, dripwater Ca in the range 0.8–4.5 mmol kg$^{-1}$. These values are highly dependent on the $pCO_2$ under which bedrock dissolves. Intermediate 'semi-open' dissolution conditions are also quantified. Dripwater $\delta^{13}C$ may subsequently be driven to higher values by $CO_2$ degassing and PCP, a process also modelled. Finally, Chapter 6 models the effects of pyrite oxidation on dripwater carbon isotope budgets, a process that may be important at some cave sites.

Model results for $\delta^{13}C$ and DCP are compared with a range of published Holocene speleothem data. Many sites show heavier $\delta^{13}C$ than can be explained by bedrock dissolution processes alone. At one site (Heshang Cave) this offset is the result of $CO_2$ degassing and prior calcite precipitation, an interpretation made possible by the availability of coupled C and Ca isotope data. At other sites offsets to higher $\delta^{13}C$ may be explained by a range of factors, including prior calcite precipitation and/or isotope exchange with atmospheric $CO_2$. The modelling approach developed here may be applied to other speleothem or dripwater datasets to improve understanding of the controls on multi-proxy records.

# Acknowledgements

Firstly I would like to thank my supervisors, Chris Day and Gideon Henderson, for their guidance and advice through all stages of this D.Phil. project. In addition, many other members of the department kindly gave their time to teach and guide me in the laboratory, including Phil Holdship, Yu-Te Hsieh, Jane Barling, Anton Vaks & Karine Wainer.

I thank other researchers for their particular scientific advice and suggestions directly relating to this work: Mike Bender, for his advice on the $Sr^{2+}$ correction during Ca isotope analysis, Philip Pogge Von Strandmann, for discussions about SrSpec resin, and Silvia Frisia, for her suggestion to include pyrite oxidation in the CaveCalc model. I also thank all others — too many to name — whose discussions at summer schools, conferences, coffee breaks and field trips have informed this work.

Special thanks are due to Chaoyong Hu and his research group at the China University of Geosciences. This research would not have been possible without their monitoring programme at Heshang Cave, which provided all samples used in Chapters 3 & 4. In addition, I would like to thank Chaoyong Hu, Yuhui Liu and colleagues for hosting Chris, Gideon and I during fieldwork in September 2015. The trip was as fantastic experience both scientifically and in general.

Finally, I would like to thank Sambuddha Misra for talking me into doing a D.Phil. in the first place, and Rebecca Morgan for her support throughout.

# Contents


Abstract . . . iii
Extended Abstract . . . v
Acknowledgements . . . ix

**Contents** **xi**
Contents . . . xiv
Abbreviations . . . xv

**List of Figures** **xvii**
List of Figures . . . xviii

**List of Tables** **xix**
List of Tables . . . xix

**1 Introduction** **1**
1.1 Introduction to Speleothems . . . 1
1.1.1 Speleothem Growth . . . 3
1.1.2 Speleothems as Palaeoclimate Archives . . . 5
1.2 Geochemical Proxies . . . 6
1.2.1 Stable Carbon Isotopes . . . 8
1.2.2 Radiocarbon . . . 9
1.2.3 Trace Metals . . . 10
1.3 Calcium Isotopes . . . 13
1.3.1 Stable Isotope Fractionation . . . 14
1.3.2 Ca Isotope Fractionation during Carbonate growth . . . 15
1.3.3 Applications to Speleothems . . . 18
1.4 Heshang Cave . . . 20

**2 Methods** **23**
2.1 Laboratory Setup . . . 23
2.1.1 PP Cleaning . . . 24
2.1.2 PTFE Cleaning . . . 25
2.2 Ca Isotope Analysis . . . 25
2.2.1 Overview of Analytical Methods . . . 25
2.2.2 Ca Isotope Notation . . . 27
2.2.3 Ca Isotope Standards . . . 28
2.2.4 Mass Spectrometer Setup . . . 29
2.2.5 Interference Testing . . . 31
2.2.6 Final Ca Isotope Routine . . . 35
2.2.7 Data Processing . . . 36
2.3 Ca Isotope Sample Preparation . . . 37

2.3.1 Why isolate Ca? . . . 37
2.3.2 Column Chemistry . . . 38
2.3.3 prepFAST-MC . . . 50
2.4 Trace Element Analysis . . . 55

**3 Ca Isotopes in Caves** **57**
3.1 Abstract . . . 57
3.2 Introduction . . . 58
3.3 Methods . . . 60
3.3.1 Study Site and Sample Selection . . . 60
3.3.2 Sample Preparation . . . 61
3.3.3 Ca Isotope Analysis . . . 61
3.3.4 Trace Element Analysis . . . 63
3.4 Results . . . 63
3.5 Discussion . . . 69
3.5.1 Interpreting Cave Ca Isotopes . . . 69
3.5.2 Modern Ca Isotopes . . . 70
3.5.3 8.2kyr Event . . . 72
3.6 Conclusions . . . 77

**4 Heshang Cave Monitoring 2011-2013** **79**
4.1 Introduction . . . 79
4.2 Samples & Methods . . . 80
4.2.1 Samples . . . 80
4.2.2 Preparation & Analysis . . . 81
4.2.3 Filtering . . . 82
4.3 Results . . . 82
4.4 Discussion . . . 90
4.4.1 HS4 Time-series analyses . . . 90
4.4.2 Drip Site Comparison . . . 93
4.4.3 Filtering Experiments . . . 95
4.4.4 Ca Sourcing at the HS4 dripsite . . . 97
4.5 Conclusions . . . 98

**5 CaveCalc Model Description** **99**
5.1 Introduction . . . 99
5.2 Model Description . . . 100
5.2.1 PHREEQC . . . 100
5.2.2 Structure & Implementation Details . . . 101
5.2.3 Supported Reactions . . . 104
5.3 Model Inputs . . . 107
5.3.1 Soil Gas & Water . . . 108
5.3.2 Soil & Aerosol Metals . . . 109
5.3.3 Bedrock Dissolution . . . 110
5.3.4 Cave Air . . . 111
5.3.5 General . . . 112
5.3.6 Run Modes . . . 113
5.4 Model Output . . . 115
5.5 An Example . . . 116
5.6 Comparison with other models . . . 118

5.6.1 Ca Isotopes . . . 119
5.6.2 Trace Metals . . . 119
5.6.3 C and O Isotopes . . . 120
5.6.4 PHREEQC-only Models . . . 121
5.7 Future Development . . . 122
5.8 Conclusions . . . 123

**6 Disentangling Controls on Cave Carbon Isotopes 125**
6.1 Introduction . . . 125
6.2 The Model . . . 127
6.3 Model Results . . . 129
6.3.1 Effect of gas volume and $pCO_2$ on bedrock dissolution . . . 129
6.3.2 Prior Calcite Precipitation . . . 132
6.3.3 Pyrite Oxidation . . . 135
6.4 Comparison with Other Models . . . 138
6.5 Comparison with Speleothem Data . . . 140
6.5.1 Soil Conditions . . . 141
6.5.2 Precipitation Mechanisms & Fractionation Factors . . . 142
6.5.3 $CO_2$ Degassing and PCP . . . 143
6.5.4 Atmospheric $CO_2$ . . . 143
6.6 Conclusions . . . 144

**Final Remarks 147**

**Bibliography 149**

**A CaveCalc: Installation Details & Code 161**
A.1 Installation . . . 161
A.1.1 Install Python 3 . . . 161
A.1.2 Install CaveCalc . . . 161
A.1.3 Install IPhreeqc COM (Windows only) . . . 162
A.2 Model Input . . . 162
A.2.1 Graphical User Interface (GUI) . . . 163
A.2.2 Python API . . . 164
A.3 Source Code . . . 165
A.3.1 cavecalc/__init__.py . . . 166
A.3.2 cavecalc/caves.py . . . 166
A.3.3 cavecalc/analyse.py . . . 178
A.3.4 cavecalc/forward_models.py . . . 182
A.3.5 cavecalc/setter.py . . . 184
A.3.6 cavecalc/util.py . . . 188
A.3.7 cavecalc/data/__init__.py . . . 193
A.3.8 cavecalc/data/defaults.py . . . 194
A.3.9 cavecalc/data/phreeqc_templates.py . . . 194
A.3.10 cavecalc/data/types_and_limits.py . . . 196
A.3.11 cavecalc/data/oxotope.dat . . . 197
A.3.12 cavecalc/gui/__init__.py . . . 263
A.3.13 cavecalc/gui/gui.py . . . 263
A.3.14 cavecalc/gui/layout.py . . . 268
A.3.15 cavecalc/gui/mapping.py . . . 269

A.3.16 scripts/cc_input_gui.py . . . . . 269
A.3.17 scripts/cc_output_gui.py . . . . . 270
A.3.18 examples/example1.py . . . . . 270
A.3.19 examples/example2.py . . . . . 270
A.3.20 examples/example3.py . . . . . 271
A.3.21 examples/example4.py . . . . . 271

# Abbreviations

| Symbol | Definition |
|---|---|
| $\delta^{44/42}$Ca | $\left[(^{44}\text{Ca}/^{42}\text{Ca})_{sample}/(^{44}\text{Ca}/^{42}\text{Ca})_{standard} - 1\right] * 1000$ |
| $\delta^{43/42}$Ca | $\left[(^{43}\text{Ca}/^{42}\text{Ca})_{sample}/(^{43}\text{Ca}/^{42}\text{Ca})_{standard} - 1\right] * 1000$ |
| $\delta^{44/40}$Ca | $\left[(^{44}\text{Ca}/^{40}\text{Ca})_{sample}/(^{44}\text{Ca}/^{40}\text{Ca})_{standard} - 1\right] * 1000$ |
| $\delta^{13}$C | $\left[(^{13}\text{C}/^{12}\text{C})_{sample}/(^{13}\text{C}/^{12}\text{C})_{standard} - 1\right] * 1000$ |
| $\delta^{18}$O | $\left[(^{18}\text{O}/^{16}\text{O})_{sample}/(^{18}\text{O}/^{16}\text{O})_{standard} - 1\right] * 1000$ |
| $^{44/42}\alpha_{a-b}$ | $\left[(^{44}\text{Ca}/^{42}\text{Ca})_a\right]/\left[(^{44}\text{Ca}/^{42}\text{Ca})_b\right]$ |
| $^{44/40}\alpha_{a-b}$ | $\left[(^{44}\text{Ca}/^{40}\text{Ca})_a\right]/\left[(^{44}\text{Ca}/^{40}\text{Ca})_b\right]$ |
| $\Delta^{44/42}\text{Ca}_{a-b}$ | $\delta^{44/42}\text{Ca}_a - \delta^{44/42}\text{Ca}_b$ |
| $\Delta^{44/40}\text{Ca}_{a-b}$ | $\delta^{44/40}\text{Ca}_a - \delta^{44/40}\text{Ca}_b$ |
| a$^{14}$C | Age-corrected $^{14}$C activity — see Stuiver and Polach (1977) |
| pMC | Percent modern carbon — see Stuiver and Polach (1977) |
| DCP | Dead Carbon Proportion — see Section 1.2.2 |
| TIMS | Thermal Ionisation Mass Spectrometer |
| ICP-MS | Inductively-Coupled Plasma Mass Spectrometer |
| m/z | mass-to-charge ratio |
| PTFE | Polytetrafluoroethylene |
| PP | Polypropylene |
| MQ | Milli-Q $H_2O$ (18 MΩ) |
| PCP | Prior Calcite Precipitation |
| API | Application Programming Interface |
| GUI | Graphical User Interface |

# List of Figures


1.1 Chinese speleothems . . . 2
1.2 Stalactite cross-section . . . 3
1.3 Schematic of Ca cycling through caves . . . 4
1.4 Stalagmite cross-section . . . 5
1.5 Life of a Proxy . . . 7
1.6 Schematic of speleothem carbon sources . . . 8
1.7 Grotto di Ernesto PCP evidence . . . 11
1.8 An example speleothem PCP record . . . 12
1.9 $\Delta^{44/42}$Ca calibration . . . 16
1.10 Cave analogue laboratory setup . . . 18
1.11 Low-resolution HS4 Holocene record . . . 20
1.12 Map of Heshang Cave . . . 22

2.1 Ca triple-isotope plot example demonstrating Sr interferences . . . 30
2.2 Isotope ratio drift during ICP-MS analysis . . . 31
2.3 The effects of Mg/Ca ratio on Ca isotope analyses . . . 33
2.4 SrSpec column elution curves (linear) . . . 40
2.5 SrSpec column elution curves (logarithmic) . . . 40
2.6 Ca isotope fractionation and Sr breakthrough in SrSpec columns . . . 47
2.7 SrSpec yield experiment Ca triple-isotope plot . . . 49
2.8 The prepFAST-MC Ca-Sr separation method . . . 51

3.1 Ca triple-isotope plot of Heshang Cave Ca isotope analyses . . . 62
3.2 HS4 drip-site Ca isotope monitoring 2005-2007 . . . 64
3.3 HS4 8.2kyr event record: $\delta^{44/42}$Ca, Mg/Ca, Sr/Ca & Ba/Ca . . . 65
3.4 Schematic of Ca cycling through caves . . . 69
3.5 PCP reconstruction across the 8.2kyr event . . . 73
3.6 8.2kyr event PCP proxy cross-plots . . . 76

4.1 HS4 Ca isotope & trace metal monitoring 2011-2013 . . . 85
4.2 Heshang Cave drip site variability . . . 86
4.3 Heshang Cave dripwater filtration results . . . 89
4.4 PCP plots of Heshang dripwater data . . . 93

5.1 An example PHREEQC database . . . 102
5.2 An example PHREEQC input script and output . . . 103
5.3 Schematic of a simple CaveCalc model run . . . 113
5.4 A Python script to run CaveCalc . . . 115
5.5 The CaveCalc input GUI . . . 116
5.6 Example CaveCalc output . . . 118

6.1 Modelling the effects of soil $pCO_2$ and gas volume . . . . . 131
6.2 Modelling the effects of $CO_2$ degassing and PCP . . . . . 133
6.3 Modelling the effects of pyrite oxidation . . . . . 136
6.4 Comparison of CaveCalc with Fohlmeister et al. (2011) . . . . . 139
6.5 Comparison of CaveCalc with Holocene speleothem data . . . . . 142

A.1 CaveCalc output GUI . . . . . 164

# List of Tables


1.1 Ca isotope abundances . . . . . 14

2.1 P2 Faraday cups . . . . . 30
2.2 Ca isotope analysis routine . . . . . 35
2.3 AG50W-X12 column recipe . . . . . 39
2.4 SrSpec elution curve calibration data . . . . . 43
2.5 SrSpec column yields mock sample . . . . . 45
2.6 SrSpec column yields data . . . . . 45
2.7 SrSpec column yields (percentages) . . . . . 46
2.8 SrSpec column recipe . . . . . 50
2.9 prepFAST-MC Ca-Sr separation test . . . . . 53

3.1 HS4 monitoring data 2005-2007 . . . . . 64
3.2 HS4 8.2kyr event data . . . . . 68

4.1 Heshang Cave 2011–2013 monitoring data . . . . . 84
4.2 Heshang dripwater filtering results . . . . . 88

5.1 CaveCalc inputs: soil . . . . . 108
5.2 CaveCalc inputs: soil metals . . . . . 109
5.3 CaveCalc inputs: bedrock . . . . . 110
5.4 CaveCalc inputs: cave air . . . . . 111
5.5 CaveCalc inputs: General . . . . . 112

6.1 CaveCalc selected fractionation factors . . . . . 128
6.2 Speleothem carbon isotope compilation . . . . . 140

A.1 CaveCalc inputs: other . . . . . 162

# Chapter 1

# Introduction

This chapter provides an introduction to caves, speleothems and their use in palaeoclimate reconstructions. To begin, Section 1.1 summarises the processes that lead to speleothem formation, where speleothems are found and why they are useful natural archives of climate change. Section 1.2 discusses several geochemical proxies commonly applied to speleothems and summarises how they are used to reconstruct palaeoclimate. These sections provide broad background relevant to any discussion of speleothem palaeoclimate science, and introduce many concepts built upon in later chapters.

A large part of this thesis focuses on the measurement of Ca isotopes in speleothems and cave dripwaters. Section 1.3 introduces Ca isotope systematics and reviews Ca isotope fractionation between water and calcite. This provides specialist background relevant to Chapters 3 & 4, where Ca isotopes data are presented from speleothem carbonates and cave dripwaters.

## 1.1 Introduction to Speleothems

'Speleothem' is a broad term that covers all secondary mineral growth occurring within caves. Such mineral growth leads to many caves across the world becoming heavily decorated with speleothem formations such as stalactites, stalagmites and flowstones (e.g. Figures 1.1 & 1.2). Such caves are often spectacular in appearance and have long fasci-



Figure 1.1: Stalagmites and stalactites in Hubei Province, China. Stalagmites grow up from the floor, stalactites down from the cave roof. In this cave the tallest stalagmites have grown to join their corresponding stalactites, forming columns. The person for scale is Dr. Chris Day. I took this photo during fieldwork in September 2015.

nated people for a variety of cultural, religious and scientific reasons.

Speleothems form by precipitation of minerals from water dripping or flowing through a cave; such minerals include calcite, aragonite and, more rarely, gypsum. The work presented in this thesis largely relates to calcite speleothems — the most common form — although many similar considerations also apply to aragonite speleothems.

Calcite and aragonite speleothems are found almost exclusively in karst caves (caves located in limestone or dolomite bedrock). Such caves are distributed globally and may be found in most areas where limestone or dolomite outcrop at the surface.



Figure 1.2: A cross-section through a broken stalactite, showing clear structure, possibly annual lamination, within the calcite. Growth hiatuses are visible as darker (i.e. dirtier) bands. Photo taken during fieldwork in September 2015 (Hubei Province, China). The fresh surface is approximately 5 cm across.

### 1.1.1 Speleothem Growth

Most natural carbonate speleothems, be they stalactites, stalagmites or other geometries, grow via the same chemical process, illustrated schematically in Figure 1.3. Water entering a cave begins as rainfall onto an overlying soil layer, where it becomes acidified by dissolved carbon dioxide produced by respiration of organic matter (Equation 1.1). This acidified solution then dissolves carbonate bedrock. These two processes are summarised by Equations 1.2 and 1.3 for limestone and dolomite caves, respectively.

$$CH_2O + O_2 \longrightarrow CO_2 + H_2O \tag{1.1}$$

$$CO_2 + CaCO_3 + H_2O \longrightarrow Ca^{2+} + 2\,HCO_3^- \tag{1.2}$$

$$2\,CO_2 + CaMg(CO_3)_2 + 2\,H_2O \longrightarrow Ca^{2+} + Mg^{2+} + 4\,HCO_3^- \tag{1.3}$$

The resulting solution is in equilibrium with soil $CO_2$ levels, which are highly variable

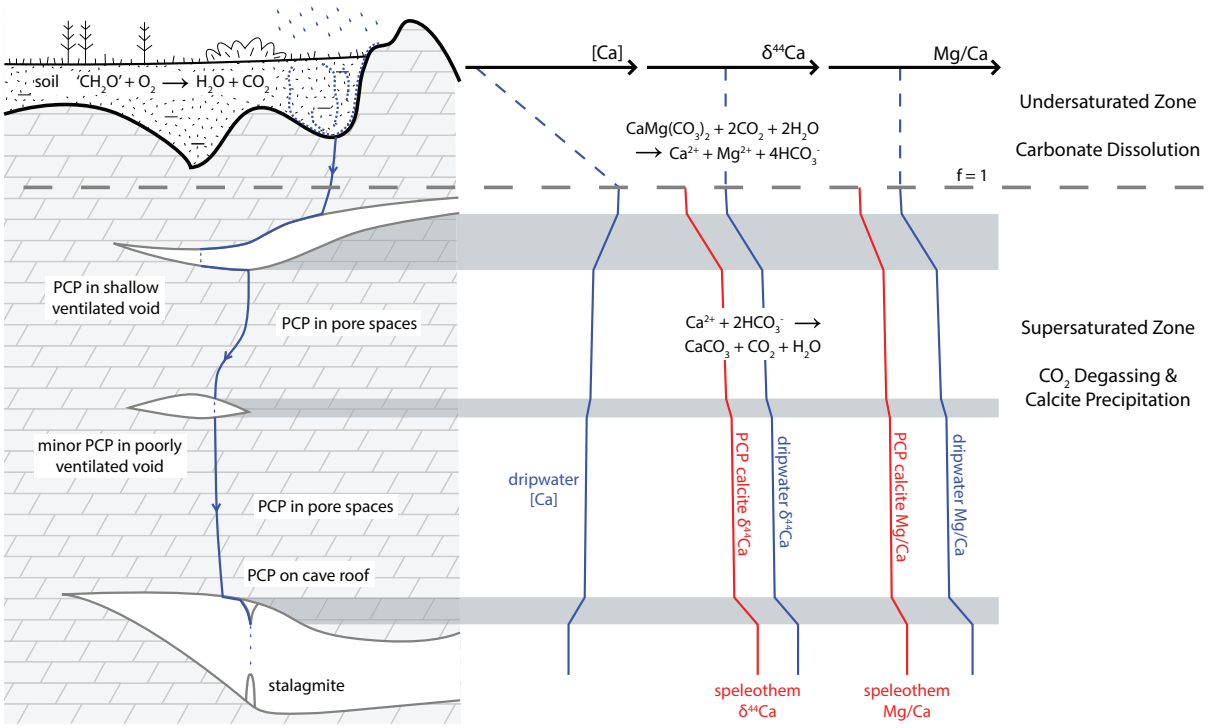



Figure 1.3: A schematic view of Ca cycling through a dolomite cave system showing evolution of dripwater Ca concentration ([Ca]), $\delta^{44/42}$Ca and Mg/Ca. Reproduced from Owen et al. (2016).

but frequently >5000 ppmv. On entering a cave the water is exposed to the cave air, which typically has a lower $CO_2$ concentration. This leads to degassing of $CO_2$ (Equation 1.4). $CO_2$ degassing increases the solution carbonate ion concentration and drives calcite or aragonite precipitation (Equation 1.5). These processes are frequently summed together, giving an approximate overall $CO_2$ degassing and $CaCO_3$ precipitation reaction (Equation 1.6) that is the reverse of Equation 1.2.

$$2\,HCO_3^- \longrightarrow CO_2 + {CO_3}^{2-} + H_2O \tag{1.4}$$

$$Ca^{2+} + {CO_3}^{2-} \longrightarrow CaCO_3 \tag{1.5}$$

$$Ca^{2+} + 2\,HCO_3^- \longrightarrow CO_2 + CaCO_3 + H_2O \tag{1.6}$$

This $CaCO_3$ precipitate may accumulate over time to form a speleothem, growing at rates typically measured in mm yr$^{-1}$. Speleothem are often visibly laminated in cross-section, clearly demonstrating their growth patterns (e.g. Figures 1.2 & 1.4). As stalagmites grow, their growth rate, structure and chemistry are all affected by local environ-



Figure 1.4: A cross-section through a speleothem, showing lamination, porosity and growth hiatuses. Note the relationship between depth and age, while monotonic, is not necessarily linear or continuous. Hiatuses are visible as darker bands and are often accompanied by subsequent changes in growth structure.

mental factors including, for example, temperature, drip rate, and soil $pCO_2$. The aim of speleothem palaeoclimate research is to understand these relationships in order to reconstruct environmental parameters from the evidence they leave in the speleothem archive. The reasons for doing this, and some methods used, are discussed in Sections 1.1.2 & 1.2, respectively.

### 1.1.2 Speleothems as Palaeoclimate Archives

Speleothems grow slowly, in rates typically measured in $\mathrm{mm\,yr^{-1}}$. Individual speleothems may grow for thousands of years if local soil, rainfall, cave ventilation and groundwater flow conditions remain suitable. This has made speleothems an interesting target for palaeoenvironmental reconstructions, particularly in the case of stalagmites, whose rela-

tively simple internal structure gives a monotonic relationship between axial depth and age, hiatuses notwithstanding (e.g. Figure 1.4).

The age of speleothem material can be accurately determined using radiometric dating techniques, in particular U-Th (Edwards et al., 1987; Wang et al., 2001). Palaeoclimate chronologies derived from U-series dating of speleothems are a major strength of the speleothem archive, providing more accurate constraints on absolute age than can be determined from ice core records and marine sediments. The development of such chronologies has improved the dating accuracy of many global climate events in the last 400 000 years (e.g. Genty et al., 2003; Xia et al., 2007; Cheng et al., 2009).

In addition to excellent age constraint, the speleothem archive has a wide geographical distribution. Speleothems may be found in karst regions globally and can provide continental palaeoclimate records at low-mid latitudes, areas traditionally lacking in reliable palaeoclimate archives on glacial-interglacial timescales. These areas are key to understanding glacial-interglacial changes in atmospheric circulation and rainfall patterns — topics of discussion that speleothems are well-placed to inform.

In addition to age-constraint and spatial distribution, a palaeoclimate archive must provide some information about local environmental or climate conditions to be of use. A brief overview of several important geochemical proxies widely used in speleothem research is given in Section 1.2.

## 1.2 Geochemical Proxies

A geochemical proxy is a chemical parameter that may be measured (e.g. an isotope ratio) in place of an unmeasurable environmental or climatic parameter (e.g. temperature) that we would like to know about. In the context of palaeoclimate reconstructions, the proxy is usually preserved in some resilient material that will survive for many years (e.g. speleothem calcite). A good proxy is one where the relationship between proxy and environmental control is well-understood and well-calibrated. Ideally a proxy will also have only a single control, with minimal interference from other environmental parameters. In

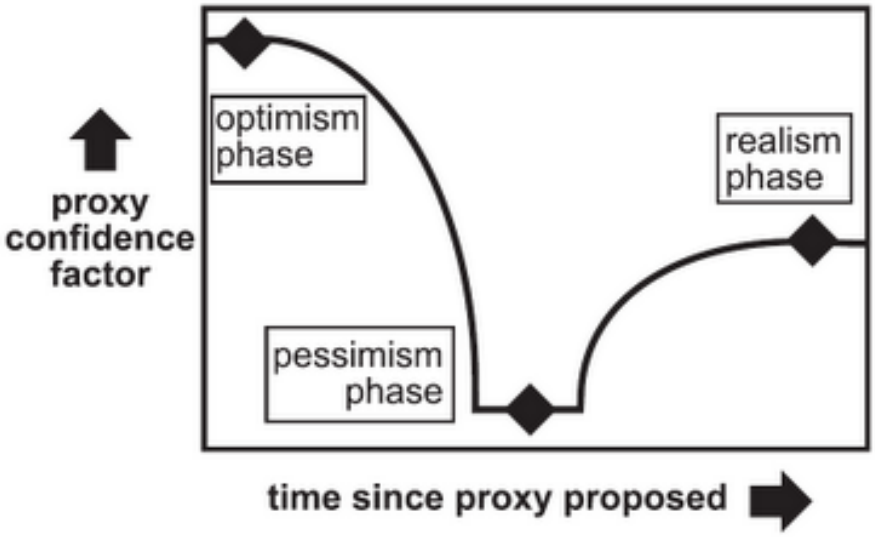



Figure 1.5: A humorous comment on the evolution of geochemical proxies, produced by Harry Elderfield.

reality these criteria are rarely met as detailed study of a promising proxy usually reveals additional potential controls that complicate interpretation, an observation humorously summarised in Harry Elderfield's "Life of a Proxy" (Figure 1.5).

A wide range of geochemical proxies have been applied to speleothem carbonates in an attempt to improve our knowledge of past climates. Such proxies include $\delta^{18}O$ and $\delta^{13}C$ analyses (e.g. Dykoski et al., 2005; Genty et al., 2006; Wang et al., 2008), radiocarbon (e.g. Genty et al., 2001; Noronha et al., 2014), trace metals (e.g. Cruz et al., 2007; Liu et al., 2013; Johnson et al., 2006) and metal isotope ratios (e.g. Goede et al., 1998; Owen et al., 2016; Riechelmann et al., 2012).

The development and interpretation of speleothem-based proxies is the main focus of this thesis. In this section I summarise a few of the proxies relevant to the work presented in later chapters. This is by no means intended as a systematic overview of all existing speleothem proxy work; such a task is well beyond the scope of a single thesis and is not attempted here. Excellent overviews are available elsewhere (e.g. Fairchild and Treble, 2009; Fairchild and Baker, 2012) if the reader is interested.

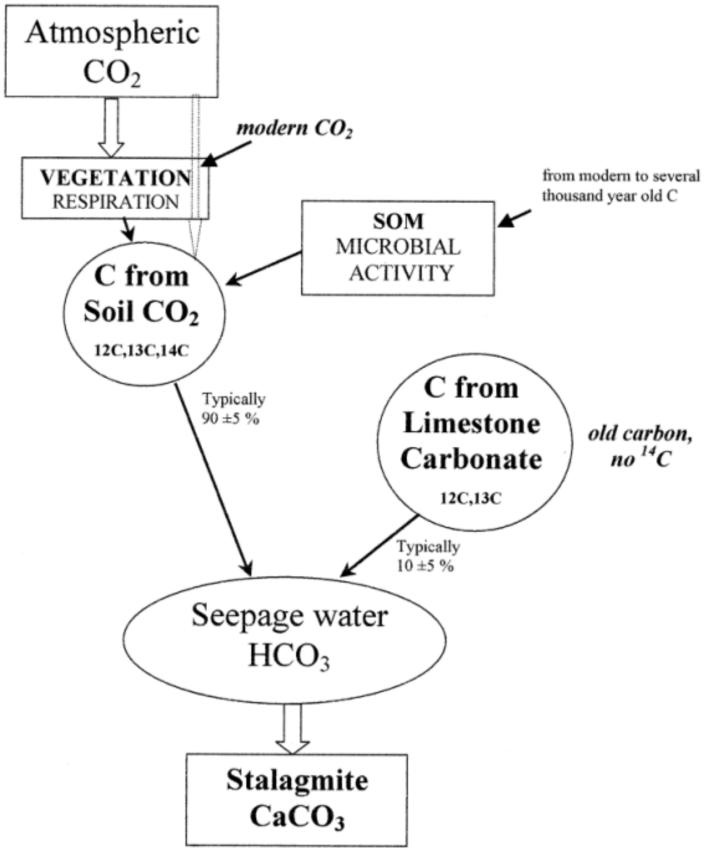



Figure 1.6: A schematic view of the various sources of speleothem carbon, and their isotopic signatures. Source: Genty et al. (2001).

### 1.2.1 Stable Carbon Isotopes

Carbon isotope ($\delta^{13}$C) analyses are, along with oxygen isotopes, the longest-established proxies applied to speleothem material. However, despite a wealth of accumulated data, the interpretation of $\delta^{13}$C records from speleothems remains difficult due to the myriad processes that may affect dripwater and speleothem carbonate chemistry and isotopes.

Early theoretical work identified mixing between soil organic-derived carbon ($\delta^{13}$C $\sim -25\,‰$) and bedrock-derived carbon ($\delta^{13}$C $\sim 0\,‰$) as an important control on speleothem $\delta^{13}$C budgets (Hendy, 1971). The relative contributions of these reservoirs are determined by the conditions under which bedrock is dissolved. Under closed-system conditions soil $CO_2$ and bedrock carbonate are added to dripwater in approximately equal amounts (Equations 1.2 & 1.3). In contrast, under open-system conditions, bedrock-derived carbon may exchange with the much larger reservoir of soil gas $CO_2$, diluting its contribution to the dripwater carbon isotope budget and leading to lighter $\delta^{13}$C values in the resulting dripwater and speleothem (Hendy, 1971).

A somewhat more complex view of this system is given in Figure 1.6. Figure 1.6 illustrates that soil $CO_2$ is comprised of several source reservoirs and may include atmospheric $CO_2$, $CO_2$ derived from respiration of plant matter and microbial $CO_2$, all of which may have distinct isotope signatures. As soil conditions change over time at a site, the relative contributions of these reservoirs may change, impacting speleothem isotope ratios.

Dripwater carbon isotopes are also fractionated as dripwater undergoes $CO_2$ degassing and calcite precipitation (Equations 1.4 & 1.5). These processes drive $CO_2$ to heavier $\delta^{13}C$ values as isotopically light carbon is preferentially removed into the gas phase (Frisia et al., 2011).

Due to this diversity of controls, speleothem $\delta^{13}C$ records have been interpreted as responding to all manner of processes, or in many cases not interpreted at all. Published interpretations include changes in soil $pCO_2$ (Hellstrom et al., 1998), changes in the extent of $CO_2$ degassing and calcite precipitation (Johnson et al., 2006; Frisia et al., 2011), changes atmospheric $pCO_2$ (Breecker, 2017), vegetation transitioning between C3 and C4 plants (e.g. Talma and Vogel, 1992), ageing of soil carbon (Genty et al., 2001), and changes between open- and closed-system dissolution (Hendy, 1971; Fohlmeister et al., 2011). Regardless of precise interpretation, speleothem $\delta^{13}C$ records are remarkably sensitive to changes in the local carbon and hydrological cycles and may be used to tightly constrain the timing of significant climate events (Genty et al., 2006). This local control provides a useful contrast with speleothem $\delta^{18}O$ records, where speleothem signals may be controlled by atmospheric processes thousands of kilometres away (e.g. Pausata et al., 2011).

### 1.2.2 Radiocarbon

Due to the contribution of radiocarbon-dead bedrock-derived carbon to dripwaters (Figure 1.6), radiocarbon measurements on speleothems cannot usually be used for geochronological purposes. Instead, radiocarbon data may be corrected for decay if the age of the speleothem is known (typically via U-Th dating), and the age-corrected values used to investigate carbon sourcing (e.g. Genty and Massault, 1997; Genty et al., 1998; Noronha et al., 2014). Age-corrected radiocarbon data may be used to calculate the dead carbon

proportion (DCP), as defined by Genty and Massault (1997):

$$\mathrm{DCP} = \left(1 - \frac{a^{14}\mathrm{C}_{sample}}{a^{14}\mathrm{C}_{atmo.}}\right) * 100\% \tag{1.7}$$

where $a^{14}\mathrm{C}_{sample}$ is the age-corrected sample radiocarbon activity, and $a^{14}\mathrm{C}_{atmo.}$ is the radiocarbon activity of the atmosphere at the time of speleothem growth (following Stuiver and Polach, 1977). DCP estimates the fraction of carbon derived from radiocarbon-dead sources, the most obvious of which is the host bedrock. This allows DCP to be used as a proxy for open-vs-closed system with closed-system dissolution giving DCP values near 50 % and open-system dissolution resulting in DCP values approaching 0 % as the bedrock-derived carbon is diluted by exchange with a large soil gas reservoir. In reality, speleothem data fall between these two end-members, indicating neither purely open- nor closed-system bedrock dissolution.

Care must be taken when interpreting DCP as purely the result of open- vs closed-system dissolution. Data from several cave sites show that ageing of soil organic matter may result in higher than expected DCP values (Genty et al., 2001; Noronha et al., 2015). Furthermore, Genty and Massault (1997) note that while primary limestone or dolomite bedrock is expected to contain no radiocarbon, this may not be the case for recent secondary carbonate deposits that may dissolve instead of bedrock.

### 1.2.3 Trace Metals

Minor or trace cations present in dripwaters may be incorporated into speleothem calcite through a variety of mechanisms. Some ions may become incorporated directly into the calcite lattice, while other elements (e.g. P, transition metals) are more frequently transported with organic or other colloidal material (Fairchild and Treble, 2009).

For ions that substitute into the calcite lattice (e.g. $Mg^{2+}$, $Sr^{2+}$, $Ba^{2+}$, $U^{2+}$), the relationship between calcite and solution composition may be related by a simple ther-

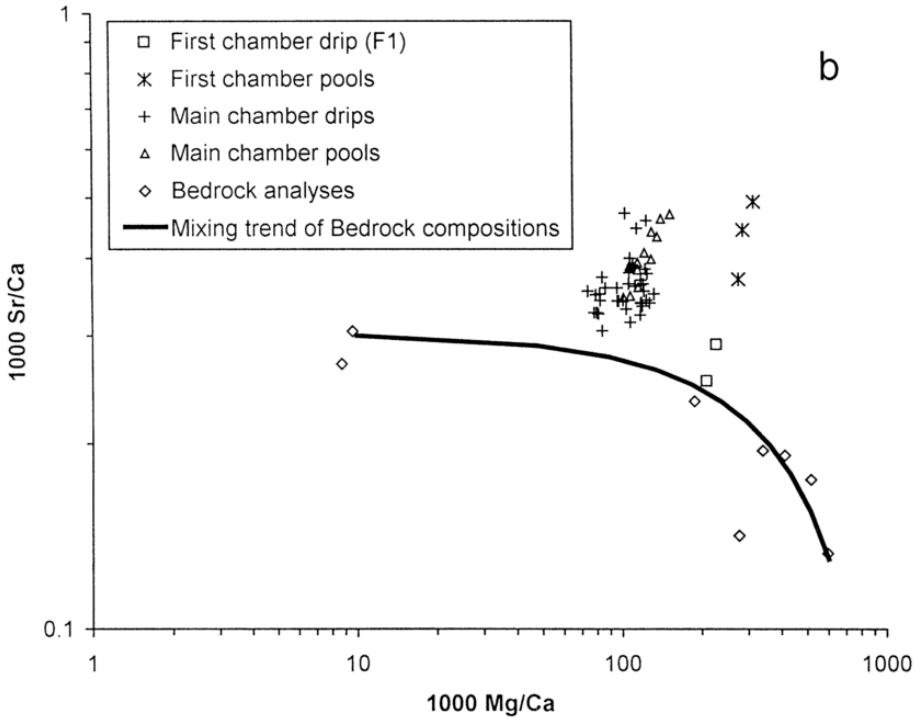



Figure 1.7: Dripwaters from Grotto di Ernesto (Italy) are shown to have elevated Mg/Ca and Sr/Ca ratios relative to the mixing line between dolomite and calcite bedrock, suggesting removal of Ca from the dripwater. Reproduced from Fairchild et al. (2000).

modynamic partitioning model:

$$D_x(T) = \frac{X/Ca_{calcite}}{X/Ca_{solution}} \tag{1.8}$$

Where $X$ is the element of interest and $D_X(T)$ is the temperature-dependent partition coefficient for element $X$, usually determined via laboratory calibration experiments (e.g. Day and Henderson, 2013).

Early studies of speleothem trace element chemistry largely focused on using variations in speleothem trace element contents (particularly Mg) over time to reconstruct temperature change (e.g. Gascoyne, 1983; Goede and Vogel, 1991; Roberts et al., 1998). This method assumes dripwater X/Ca ratios are approximately constant, and hence variations are attributed to temperature-dependent changes in $D_X$ (Equation 1.8). Other studies found that changes in trace element sourcing accounted for variations through time (e.g. Goede et al., 1998).

More recent research has highlighted the importance of karst processes that may mod-

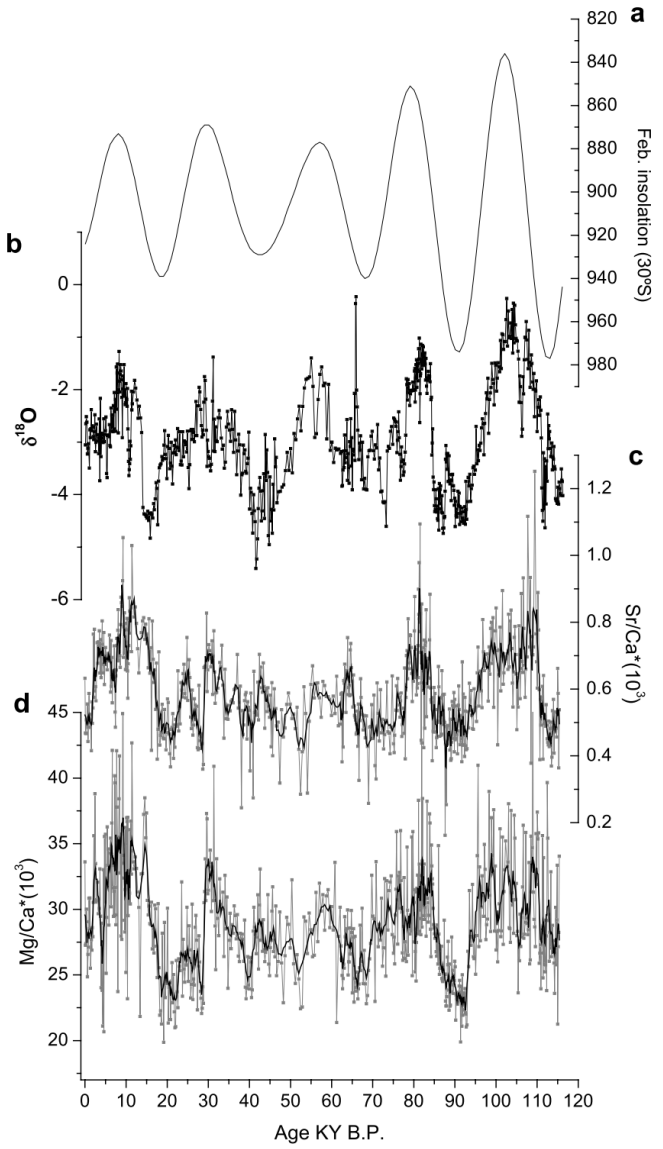



Figure 1.8: A speleothem time-series record of $\delta^{18}$O, Sr/Ca and Mg/Ca from Botuverá Cave, Brazil. Multi-millenial co-variation between Mg/Ca and Sr/Ca is clear. This is likely caused by changes in PCP, providing clear evidence for changes in local rainfall on this timescale. $\delta^{18}$O also correlates well with the trace element proxies, however interpretation of $\delta^{18}$O data is more complex and cannot easily be exclusively attributed to a local control. Reproduced from Cruz et al. (2007).

ify dripwater X/Ca ratios significantly from their initial compositions. Fairchild et al. (2000) first noted elevated dripwater Mg/Ca and Sr/Ca ratios relative to the ratios expected from quantitative bedrock dissolution (Figure 1.7). They suggested that calcite precipitation prior to dripwater arriving at the speleothem apex could explain the elevated ratio; they termed this process prior calcite precipitation (PCP). PCP preferentially removes Ca from dripwater, increasing X/Ca ratios. The extent of PCP is itself determined by effective rainfall above the cave, with periods of aridity corresponding to increased PCP and hence higher X/Ca ratios (Tooth and Fairchild, 2003; Tremaine and Froelich, 2013). PCP may also be influenced by variations in cave air $pCO_2$ (Mattey et al., 2010). In addition to PCP, several other karst/cave processes may influence speleothem X/Ca ratios. These include calcite recrystallisation reactions (e.g. Wong et al., 2011) and and

drip rate variations (e.g. Stoll et al., 2012).

Of these controls, rainfall-driven PCP remains the most common interpretation for variations in speleothem Mg/Ca and Sr/Ca time-series. This interpretation has been applied to speleothem trace element datasets from a wide range of cave sites over a variety of timescales to produce records of relative change in local aridity (e.g. Johnson et al., 2006; Cruz et al., 2007; Liu et al., 2013). An example of a coupled Mg/Ca and Sr/Ca dataset from southern Brazil, published by Cruz et al. (2007), is shown in Figure 1.8. Their record demonstrates Mg/Ca and Sr/Ca co-variation on multi-millenial timescales, and they suggest a PCP control responding to insolation-driven rainfall changes.

The concept of PCP will be revisited and developed in Chapters 3 & 4.

## 1.3 Calcium Isotopes

Calcium is an element abundant in many natural materials. It has six naturally-occurring isotopes, the atomic masses and bulk earth abundances of which are given in Table 1.1. Due to its abundance and diversity of isotopes, Ca has long been identified as an obvious target for isotope analysis, both for radiogenic and stable isotope signals (DePaolo, 2004). Ca isotope analyses have been applied to a range of fields including geochronology (e.g. Marshall and DePaolo, 1982; Shih et al., 1994), cosmochemistry (e.g. Jungck et al., 1984; Niederer and Papanastassiou, 1984), palaeoceangraphy (e.g. Farkaš et al., 2007a; Blättler et al., 2012) and biology (e.g. Skulan and DePaolo, 1999; Chu et al., 2006). Despite wide usage, the potential applications of Ca isotopes to cave carbonates have not previously been explored in the literature.

This section contains a brief introduction to stable isotope fractionation (Section 1.3.1), followed by a more detailed review of Ca isotope fractionation between water and calcite (Section 1.3.2). This provides background information for discussion of Ca isotopes in caves (Section 1.3.3), a topic developed further in Chapters 3 & 4. Details of Ca isotope analytical techniques are discussed in Chapter 2.

Table 1.1: Naturally-occurring isotopes of calcium (data from Boulyga, 2010).

| Isotope | Bulk Earth Abundance (%) | Half-life (yr) |
|---|---|---|
| $^{40}$Ca | 96.941 | stable |
| $^{42}$Ca | 0.647 | stable |
| $^{43}$Ca | 0.135 | stable |
| $^{44}$Ca | 2.086 | stable |
| $^{46}$Ca | 0.004 | stable |
| $^{48}$Ca | 0.187 | $6 \times 10^{18}$ |

### 1.3.1 Stable Isotope Fractionation

Isotope fractionation describes the enrichment or depletion of heavy isotopes relative to light isotopes of an element between two materials. Isotopes of any element may be fractionated by a variety of chemical and physical processes that discriminate based on atomic or nuclear mass. The driving force behind isotope fractionation may be thermodynamic or kinetic. In brief, thermodynamic (equilibrium) fractionations are driven differences in atomic orbital energies between isotopes. This often results in heavy isotopes forming more stable (i.e. lower energy) bonds, preferentially enriching more condensed phases in heavy isotopes when thermodynamic equilibrium is reached.

However, thermodynamic equilibrium is not always reached in low-temperature natural settings, and kinetic fractionations may play an important role. Kinetic fractionations are driven by differences in reaction rate between various isotopes. Light isotopes, owing to their lower mass, typically vibrate or diffuse faster within a given medium and may also have lower activation energies for chemical processes. During mineral precipitation reactions, this may allow light isotopes to desolvate and reach reaction sites faster, preferentially incorporating them into a growing crystal (DePaolo, 2011).

The distinction between thermodynamic and kinetic fractionations is important as the magnitude of a thermodynamic fractionation theoretically depends only on temperature, while kinetic fractionations may vary in response to a wide range of chemical and physical controls. Typically thermodynamic (equilibrium) fractionation processes are thought to dominate at high temperatures and with long equilibration times, while kinetic factors become increasingly important during faster processes and/or at lower temperatures.

### 1.3.2 Ca Isotope Fractionation during Carbonate growth

While many natural processes are expected to fractionate Ca isotopes, early studies of Ca isotope fractionation were limited by available analytical techniques (e.g. Russell et al., 1978; Zhang et al., 1988). Terrestrial variations in $\delta^{44/40}$Ca are small, typically <1 ‰, and were barely resolvable until further advances in TIMS and ICP-MS analytical techniques (DePaolo, 2004; Boulyga, 2010). Modern analytical techniques can typically achieve external reproducibility of 0.10–0.15 ‰ and such precision has led to the application of Ca isotope analysis to a wide range of fields, including carbonate geochemistry.

The Ca isotope fractionation occurring as calcite precipitates from water is of particular geochemical interest. Carbonate minerals are widespread in the geological record and knowledge of the isotope fractionation factor allows researchers to reconstruct source water Ca isotope ratios from measured calcite data.

Initial studies clearly showed that calcium incorporated into carbonate minerals is isotopically lighter than the precipitating solution (Gussone et al., 2003; Lemarchand et al., 2004; Marriott, 2004). The magnitude of this fractionation was found to vary with temperature (Gussone et al., 2005; Marriott, 2004) and also calculated growth rate (Lemarchand et al., 2004). The growth rate control suggests the importance of kinetic fractionation. Perhaps the most thorough laboratory calibration to date is that of Tang et al. (2008). Their results quantify the temperature and growth-rate dependence of the Ca isotope fractionation factor at growth rates perhaps more representative of natural carbonate formation, giving a total range of $-1.50 < \Delta^{44/40}\text{Ca} < -0.46$ (Figure 1.9). On an interesting tangent, Tang et al. (2008) also note a clear relationship between Ca isotope fractionation and Sr partitioning into calcite, suggesting similar incorporation mechanisms.

While these laboratory studies demonstrate a kinetic isotope effect, they are unable to give a reliable estimate of equilibrium fractionation factor as isotopic equilibration requires growth rates and/or equilibration times that are unfeasible on laboratory timescales. Instead, several researchers have attempted to directly measure the equilibrium fractionation factor in natural systems where carbonate and water have had time to reach isotopic

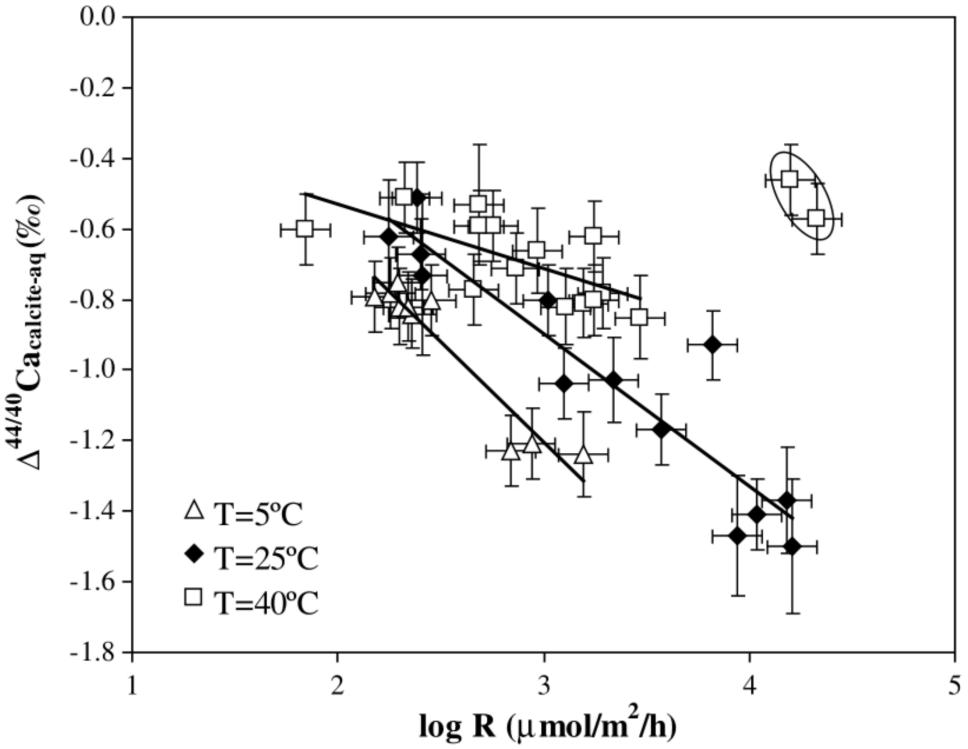



Figure 1.9: Laboratory calibration data for the Ca isotope fractionation factor between calcite and water as a function of temperature and precipitation rate. The magnitude of $\Delta^{44/40}$Ca is shown to increase with growth rate and decrease with temperature. The circled points are identified by Tang et al. (2008) as anomalous and are discounted. Reproduced from Tang et al. (2008).

equilibrium. Inorganic marine calcite cements (Steuber and Buhl, 2006), recrystallized nanofossil oozes (Fantle and DePaolo, 2007) and a carbonate aquifer (Jacobson and Holmden, 2008) all suggest a near-zero equilibrium isotope fractionation factor. Given this near-zero equilibrium fractionation factor, all observed fractionations can be ascribed to kinetic effects, controlled primarily by growth rate and temperature.

#### 1.3.2.1 Modelling Ca Isotope Fractionation

Theoretical models have been developed to help understand the controls on kinetic Ca isotope fractionation (Watson, 2004; Tang et al., 2008; DePaolo, 2011; Nielsen et al., 2012). Here the model of DePaolo (2011) is briefly discussed to illustrate various controls on Ca isotope fractionation.

DePaolo (2011) describes a system with forward (precipitation) and backward (dissolution) reactions occurring at the mineral surface (reaction rates $R_f$ and $R_b$, respectively),

the imbalance between which is the net precipitation rate. Due to reaction kinetics, both $R_f$ and $R_b$ are slightly faster for light isotopes. Under these assumptions alone, far-from-equilibrium crystal growth ($R_f >> R_b$) results in isotopically light calcite growth, while at dynamic equilibrium ($R_f = R_b$) total isotope fractionation is zero.

In addition, the model allows a solution boundary layer to evolve separately from the bulk solution, with diffusive exchange between the two reservoirs. As the crystal grows it removes Ca from the boundary layer, preferentially incorporating isotopically light Ca. If the precipitation rate is slow relative to the rate of Ca diffusion in the solution, the enrichment of the boundary layer is buffered by diffusive exchange with the bulk solution. However, if the precipitation rate is fast relative to the rate of diffusion, the isotopic composition of the interface region may become isotopically heavy relative to the bulk solution. An isotopically enriched boundary layer will lead to isotopically enriched calcite. Further development of this model by Nielsen et al. (2012) also suggests the importance of solution $Ca:CO_3$ ratio in determining the extent to which an isotopically distinct boundary layer may develop.

Quantitative application of these models to data from natural environments is tricky, as instantaneous reaction kinetics and surface areas are notoriously difficult to determine outside of a laboratory. Nevertheless, such models highlight important controls on the magnitude of the fractionation factor between solution and calcite precipitate. Such concepts can help us better interpret data from various settings. Several key controls are:

- Calcite growth rate ($R_f - R_b$). Increasing growth rate will increase fractionation as kinetic factors become more important, however this trend may reverse as growth rates become very high and an isotopically distinct boundary layer develops.
- Solution Mixing. Static solutions potentially allow for the development of isotopically distinct boundary layers, while vigorously mixed solutions do not.
- Solution $Ca:CO_3$ ratio. Increasing solution $Ca:CO_3$ ratio at constant calcite supersaturation decreases isotope fractionation.

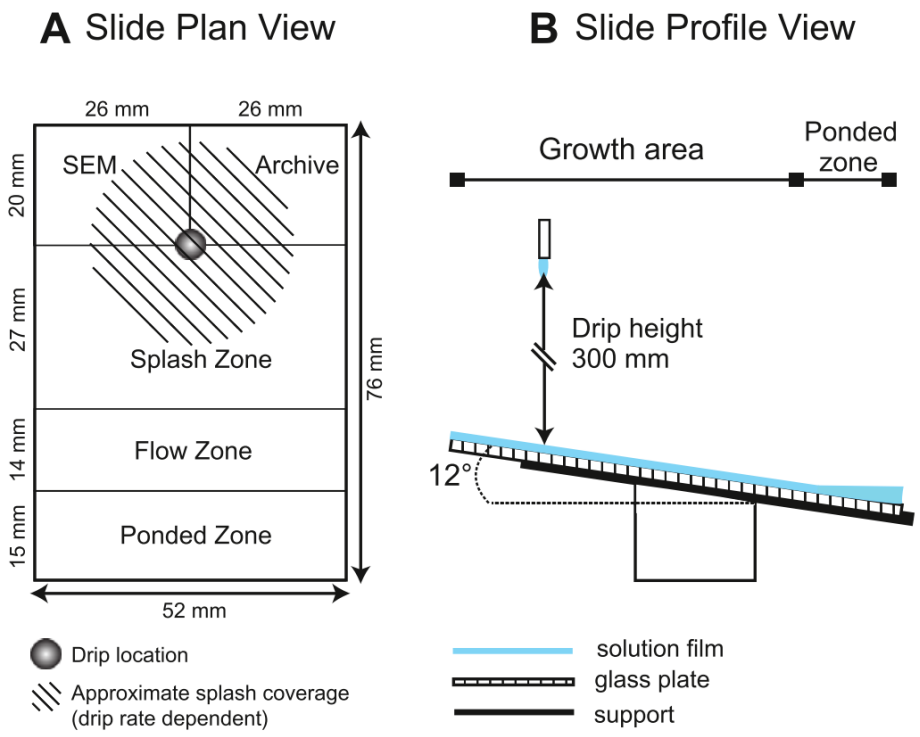



Figure 1.10: The laboratory setup used by Reynard et al. (2011) to simulate speleothem growth. Synthetic dripwater is dripped onto a sloping glass plate where calcite precipitation occurs, driven by $CO_2$ degassing. This is a better model for speleothem growth than beaker-type precipitation experiments, as it more closely simulates the kinetic environment found on the apex of a stalagmite. Reproduced from Reynard et al. (2011).

### 1.3.3 Applications to Speleothems

Most studies of calcium isotopes in natural carbonates have focused on palaeoceanographic reconstructions (e.g. De La Rocha and DePaolo, 2000; Heuser et al., 2005; Fantle and DePaolo, 2005; Farkaš et al., 2007a,b; Griffith et al., 2008, 2011; Blättler et al., 2011) and calibration of these marine proxies using both biogenic carbonates (Gussone et al., 2003, 2006; Gussone and Filipsson, 2010) and laboratory inorganic precipitation experiments (Lemarchand et al., 2004; Marriott, 2004; Tang et al., 2008). Very little attention has been paid to carbonate precipitation from thin-film solutions driven by $CO_2$ degassing, for example during speleothem growth. Modelling of reaction kinetics (Section 1.3.2.1) highlights the importance of factors such as precipitating solution volume and flow rates, as well as growth rate and $Ca:CO_3$ ratio, in determining the total fractionation occurring between bulk solution and carbonate. As such it is not clear whether the fractionation factors discussed in Section 1.3.2 are applicable to cave settings.

Prior to the research presented in this thesis, the only published study of calcium isotope fractionation directly relevant to speleothems is that of Reynard et al. (2011). They describe the results of a laboratory experiment simulating speleothem growth by dripping artificial dripwater onto glass plates, where $CO_2$ degassing drives $CaCO_3$ precipitation (Figure 1.10). Their results give an average $\Delta^{44/42}Ca_{CaCO_3\text{-}H2O} = -1.0\,‰$, a fractionation approximately double that measured in the beaker-type experiments with similar growth rates (Section 1.3.2). In addition to reporting larger absolute fractionations than other studies, data from Reynard et al. (2011) show no significant temperature dependence and only a weak correlation with saturation state.

The origin of this larger fractionation remains unclear. The authors interpret their results using the model of DePaolo (2011). They suggest that constant disruption of solution boundary layer, caused by thin-film flow and continued dripping, prevents the development of an isotopically enriched solution boundary layer and hence allows for greater isotope fractionation relative to the precipitating solution. This interpretation implies that the comparison data sets of Lemarchand et al. (2004), Marriott (2004) and Tang et al. (2008) are all significantly affected by transport limitation, contrary to the interpretation of DePaolo (2011). Alternatively, the larger fractionation could be related to differences in $Ca:CO_3$ ratios between experiments. It is difficult to precisely quantify the $Ca:CO_3$ ratios during $CaCO_3$ precipitation in the experiments of Reynard et al. (2011) due to the effects of degassing and mixing, however initial solution chemistry gives $Ca:CO_3$ ratios lower than those of Tang et al. (2008) by a factor of 2–10, which may result in the experiments of Reynard et al. (2011) exhibiting a greater degree of kinetic fractionation (following the model of Nielsen et al., 2012).

Regardless of mechanism, the results of Reynard et al. (2011) indicate that speleothem Ca is isotopically light relative to the dripwater composition, with only minimal sensitivity to temperature or dripwater supersaturation across the range of conditions tested. The application of Ca isotope measurements to speleothems, and interpretation of the resulting datasets, forms the major topic of Chapter 3, and is developed further in Chapter 4.

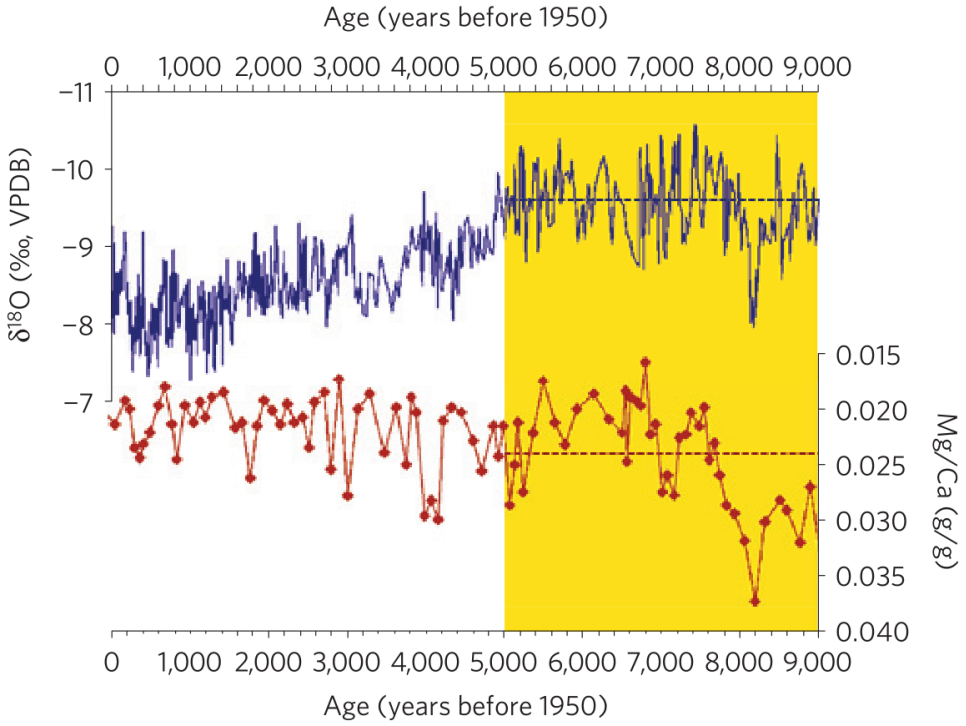



Figure 1.11: Low-resolution $\delta^{18}$O and Mg/Ca Holocene records from HS4, showing a long-term trend in $\delta^{18}$O and relative stability in Mg/Ca ratios. Note that the 8.2kyr event is clearly resolved in both proxies. Reproduced from Liu et al. (2013).

## 1.4 Heshang Cave

Much of the work in this thesis focuses on Heshang Cave and the stalagmite HS4. This Section provides some background to this study site.

Heshang Cave is a dolomite cave system located in Hubei Province, China (30°27′ N, 110°25′ E; 294 m). The cave is linear, approximately 250 m long and extends sub-horizontally into a steep mountainside rising ∼600 m above the cave entrance (Figure 1.12). The mountaintop is overlain by a well-vegetated soil horizon ∼30 cm thick (Hu et al., 2005, 2008a). Heshang Cave is open and well-ventilated, with a strong seasonal temperature cycle (12–26 °C), and relatively low cave air $pCO_2$ in the range 400–500 ppmv (Hu et al., 2008a). Ventilation is primarily through the large cave entrance, although airflow through the cave suggests a second entrance behind a rockfall at the southern end of the cave. Local climate is dominated by the East Asian Monsoon, with peak rainfall through June-July-August reaching values of ∼10 mm/day (Hu et al., 2008a).

HS4 is a 2.5 m tall stalagmite removed from the cave in 2001. The stalagmite has been

studied extensively for its continuous and high-resolution archive of Holocene palaeoclimate (Figure 1.11; Hu et al., 2005; Johnson et al., 2006; Hu et al., 2008b; Li et al., 2014). Notably, HS4 provides a well-resolved record of the 8.2kyr event (Liu et al., 2013); this event is discussed further in Chapter 3. The location of Heshang Cave, combined with the sub-annual resolution of HS4, renders this an excellent site for use in reconstructing changes in the East Asian Monsoon over multi-millennial timescales (e.g. Hu et al., 2008b).

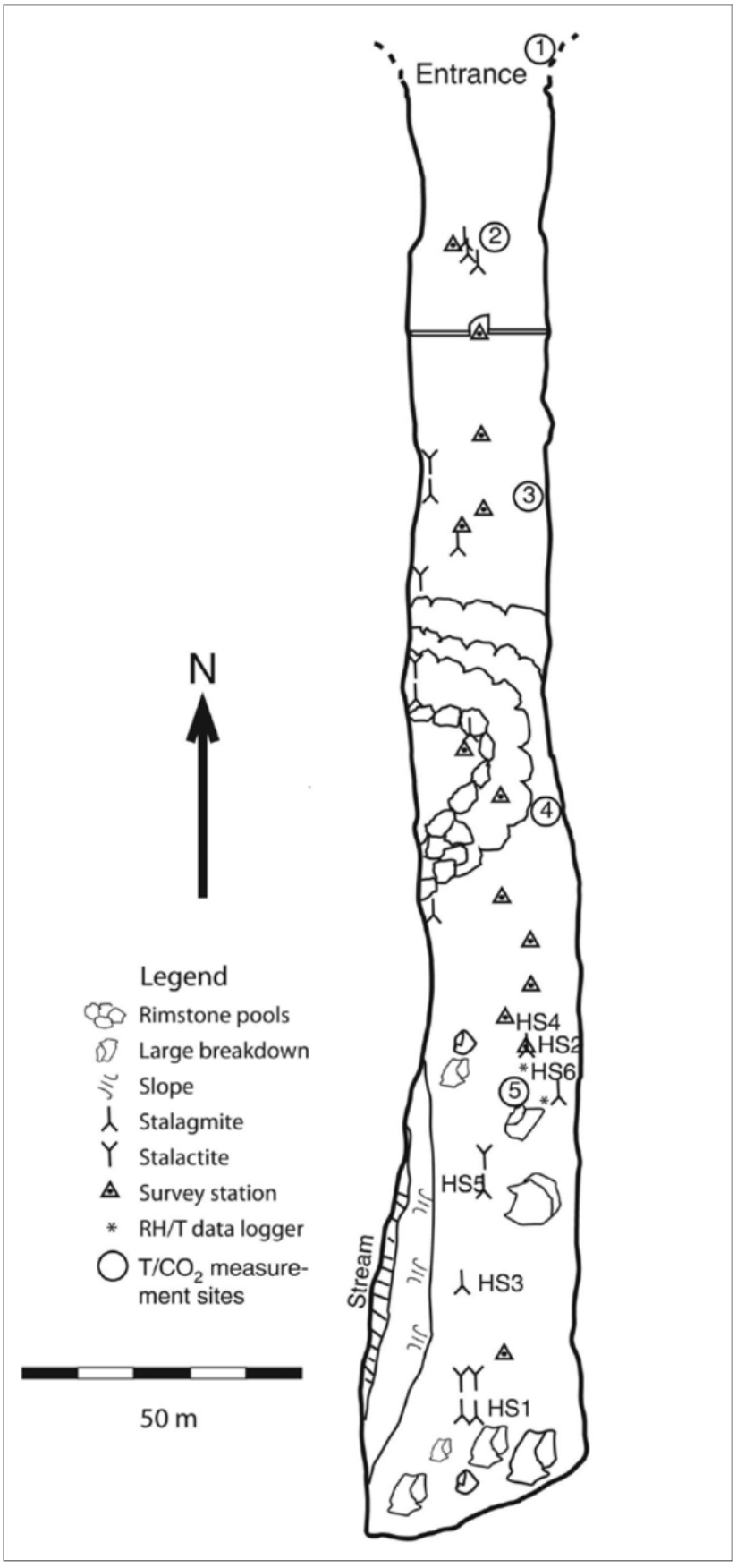



Figure 1.12: Heshang Cave map. Access is via the entrance at the north end of the cave, which sits ∼30 m above the Qingjiang river. The cave is gently sloped, with the cave entrance at the lower end, and very open: at the HS4 site, the cave roof is 18 m high. The south end of the cave is blocked by a rockfall and likely continues further. Reproduced from Hu et al. (2008a).

# Chapter 2

# Methods

In this chapter I discuss the analytical methods developed, tested and used in the course of this research. It is hoped that this material will be of use to those seeking to set up similar analytical procedures, or improve upon them either in Oxford or other geochemistry laboratories.

## 2.1 Laboratory Setup

The analytical work in this thesis focuses on metal abundance and isotope analyses of cave dripwaters and speleothems. Many elements or isotopes of interest in these samples are present in only trace quantities and as such it is necessary to prepare and analyse samples under carefully controlled conditions to minimise any potential sample contamination. Unless otherwise stated, all analytical work was conducted at the Department of Earth Sciences, University of Oxford. All clean laboratory chemistry work (e.g. sample preparation, column chemistry) was performed under class 1000 clean laboratory conditions.

All containers and equipment coming into contact with samples or reagents were precleaned to minimise contamination. Where possible, all samples and standards were handled and stored in clean polypropylene bottles or vials. For procedures that require drying of samples on a hot-plate, clean Teflon beakers were used instead, as the melt-

ing point of PTFE is higher than that of PP (327 °C versus 160 °C). For general use, polypropylene apparatus was used in preference to Teflon for two reasons:

- PP is typically more transparent than PTFE, allowing easier viewing of the contents of bottles.
- PTFE accumulates static electricity much more readily than PP. Accumulation of static charge on the surface of the plastic may cause erratic solution behaviour (e.g. droplets jumping out of a vial) or attract any dust particles present in the environment, a potential source of contamination. Both effects are undesirable.

Plastics used for clean laboratory apparatus (e.g. PP, PTFE) frequently contain elements that may contaminate elemental or isotopic analyses (e.g. Ca, Mg). Such contaminants may be integral impurities from the manufacturing procedure, or adsorbed material from prior laboratory use if containers are being re-used. When exposed to acidic solutions, metal ions may leach from these plastics into samples or reagents, thereby contaminating them (Laxen and Harrison, 1981). To avoid such problems, all plasticware used to handle reagents or samples was first cleaned through a series of rinses and acid leaches to remove any adsorbed or absorbed contaminants that could potentially mobilise into analytical solutions or reagents. The details of these cleaning routines are given in Sections 2.1.1 and 2.1.2.

Unless otherwise stated, all acids used ($HNO_3$ and HCl) were prepared from quartz-distilled stock acids prepared in-house. Dilutions were made using 18 MΩ MQ water.

### 2.1.1 PP Cleaning

All polypropylene equipment coming into direct contact with samples or analytical reagents (e.g. bottles, pipette tips and Eppendorf tubes) was cleaned using the following procedure to leach out contaminant metal ions and oxidise organic material adhered to plastic surfaces.

1. Rinse with MQ water. This step removes any dust or other large particles.

2. Soak in 10 % $HNO_3$ (reagent-grade) for 48 hours, inverting after 24 hours. This step leaches metals and oxidises organics adhered to the plastic surface.
3. Rinse with MQ water. This washes out any remaining acid.
4. Soak in MQ water for 48 hours, inverting after 24 hours. This step leaches out any acid absorbed into the plastic.
5. Rinse with MQ water.

### 2.1.2 PTFE Cleaning

All PTFE beakers used were cleaned using the following routine:

1. Wipe out with ethanol and paper to remove any residual material from previous use.
2. Rinse with MQ water. Repeat three times.
3. Add a small amount amount of 16 M reagent-grade $HNO_3$ to each beaker. Seal lids and reflux on a hotplate for 24 hours at 130 °C.
4. Rinse with MQ water. Repeat three times.
5. Soak beakers (internal and external surfaces) in 8 M reagent-grade $HNO_3$ and leave on a hotplate for 24 hours at 130 °C.
6. Rinse with MQ water. Repeat three times.
7. Add a small amount amount of 10 M HCl to each beaker. Seal lids and reflux on a hotplate for 24 hours at 80 °C.
8. Rinse with MQ water. Repeat three times.

## 2.2 Ca Isotope Analysis

### 2.2.1 Overview of Analytical Methods

Modern analyses of Ca isotope ratios are universally conducted via mass spectrometry. Both Thermal Ionisation Mass Spectrometry (TIMS) and Inductively-Coupled Plasma Mass Spectrometry (ICP-MS) have been widely applied to natural materials, with each

method providing unique advantage and drawbacks. Both approaches are capable of resolving variations in Ca isotope ratios of less than one part per thousand, sufficient to resolve signals in many natural materials (Russell et al., 1978). A brief comparison of these two methods is provided here; the reader is referred to Boulyga (2010) for a thorough review of published analytical techniques.

ICP-MS and TIMS instruments differ in their sample introduction methods. TIMS analysis requires that samples are dried onto a filament. The filament is loaded into the mass spectrometer and heated to vaporise and atomise the sample. In contrast, ICP-MS instruments take a dissolved sample which is vaporise as it passes through a nebuliser and subsequently atomised and ionised as the vapour flows into an argon plasma torch. Both instrument types share a common back end — ionised atoms are accelerated across a large voltage, focussed into a coherent beam, and split according to m/z ratio as the beam passes through a magnetic field. Finally, the intensity at each m/z ratio may be determined by a detector, usually a Faraday cup. The current detected by the Faraday cup is proportional to the current delivered by the beam, which may be calibrated to determine ion intensity.

In general, TIMS methods give greater precision, but are slow due to the laborious process of loading samples onto filaments, as well as the relatively long analytical time required for a single sample. ICP-MS techniques were developed as a higher-throughput alternative to TIMS, however ICP-MS analysis suffers from its own drawbacks. Due to the use an argon plasma torch, the ion beam is heavily contaminated by ionised argon (dominated by $^{40}Ar^{+}$), as well as any other argon-, solution- or atmosphere-derived contaminants. This has three important consequences which impact analytical precision and accuracy:

1. The ion beam is denser than during TIMS analysis, leading to more significant space-charge effects (Section 2.2.5) and hence greater instrumental fractionation. This must be corrected for.

2. There is the potential for formation of many undesired, and potentially interfering, polyatomic ions within the plasma. For example, $ArH_2^+$ may form an isobaric inter-

ference on $^{42}Ca^{+}$. Such interferences are less significant in TIMS analysis as samples are ionised under a vacuum.

3. The dominant plasma-derived ion ($^{40}Ar^{+}$) is an isobaric interference on $^{40}Ca^{+}$. As such, measurement of $^{40}Ca$, the major isotope of Ca, is generally impossible via ICP-MS techniques. For this reason, ICP-MS acquired data are usually reported as $\delta^{44/42}Ca$, rather than the $\delta^{44/40}Ca$ notation widely used for TIMS-derived data.

Despite these complications, recent advances in ICP-MS techniques have achieved long-term precision on $\delta^{44/42}Ca$ similar to that acquired by TIMS methods on $\delta^{44/40}Ca$ (0.10–0.15 ‰; Boulyga, 2010). If converted to the same isotope notation, this corresponds to uncertainty on ICP-MS data being approximately double that of TIMS data.

At the onset of the research presented in this thesis, it was not known what magnitude of Ca isotope variation could be expected in speleothem time-series, nor whether the two-fold increased precision of TIMS analysis would be required. Both TIMS (e.g. Russell et al., 1978; Skulan et al., 1997; Skulan and DePaolo, 1999) and ICP-MS (e.g. Halicz et al., 1999; Chu et al., 2006; Griffith et al., 2011; Blättler et al., 2011) methods have been successfully applied to Ca isotope analysis of a range of other natural materials. As such, I began analyses using the ICP-MS method originally established at the University of Oxford by Halicz et al. (1999). The development and application of this method to cave materials is discussed in Section 2.2.4.

### 2.2.2 Ca Isotope Notation

To measure mass-dependent isotope fractionation, measurement of two isotopes is required. As Ca has six different naturally-occurring isotopes (Table 1.1), there are many ways to achieve this and the selection of isotopes to measure usually depends on the analytical technique used (see Section 2). This thesis reports Ca isotope fractionation using the $\delta$-notation given in Equations 2.1 and 2.2.

$$\delta^{44/42}\text{Ca}\ (‰) = \left( \frac{(^{44}\text{Ca}/^{42}\text{Ca})_{sample}}{(^{44}\text{Ca}/^{42}\text{Ca})_{standard}} - 1 \right) * 1000 \tag{2.1}$$

$$\delta^{43/42}\text{Ca}\ (‰) = \left( \frac{(^{43}\text{Ca}/^{42}\text{Ca})_{sample}}{(^{43}\text{Ca}/^{42}\text{Ca})_{standard}} - 1 \right) * 1000 \tag{2.2}$$

Of the two, $\delta^{44/42}$Ca is used as the primary isotope measurement and $\delta^{43/42}$Ca is a check on data quality. To convert between $\delta^{44/42}$Ca and $\delta^{44/40}$Ca data reported elsewhere in the literature, data may simply be multiplied or divided by two, as appropriate, provided they are normalised to the same standard. The different Ca isotope standards in use are discussed in Section 2.2.3.

The isotope fractionation between two materials may be expressed in several ways, including:

$$^{44/42}\alpha_{a-b} = \frac{(^{44}\text{Ca}/^{42}\text{Ca})_a}{(^{44}\text{Ca}/^{42}\text{Ca})_b} \tag{2.3}$$

$$\Delta^{44/42}\text{Ca}_{a-b} = \delta^{44/42}\text{Ca}_a - \delta^{44/42}\text{Ca}_b \tag{2.4}$$

where $a$ and $b$ are two Ca-bearing materials. $\alpha$ (Equation 2.3) is usually used to express an isotope fractionation factor associated with some chemical or physical process, while $\Delta$Ca simply expresses a difference in $\delta$Ca values. The two are approximately related according to Equation 2.5:

$$\Delta\text{Ca}_{a-b} = 1000 * ln(\alpha_{a-b}) \tag{2.5}$$

### 2.2.3 Ca Isotope Standards

Traditionally Ca isotope data have been reported relative to NIST SRM 915a, a relatively pure and isotopically homogeneous calcium carbonate-derived solution. However, due to limited supply of this standard, many laboratories have switched to alternative primary

standards, including NIST SRM 915b, seawater, and various in-house standards.

NIST SRM 915b (the replacement to NIST SRM 915a) is, unfortunately, not as clean as its predecessor. SRM 915b contains significant strontium ($Sr/Ca = 0.17\,mmol\,mol^{-1}$), a contaminant that may directly interfere with Ca isotope analysis by ICP-MS (Section 2.2.5.2). It is also isotopically distinct from SRM 915a, with published values of $\delta^{44/40}Ca = 0.72 \pm 0.05\,‰$ (Wombacher et al., 2009) and $\delta^{44/40}Ca = 0.72 \pm 0.04\,‰$ (Heuser and Eisenhauer, 2008).

The work presented in this thesis uses NIST SRM 915a as the primary standard, and all $\delta$-values are reported relative to it. NIST SRM 915b was routinely run as a secondary standard to monitor instrument accuracy and precision. Taking the average of all analyses made during the course of this thesis, I report NIST SRM 915b $\delta^{44/42}Ca = 0.36 \pm 0.11\,‰$ (2s, n = 85) and $\delta^{44/42}Ca = 0.19 \pm 0.15\,‰$ (2s, n = 85). These data are reported after Sr-correction, as discussed in Section 2.2.5.2, and agree well with the published values of Wombacher et al. (2009) and Heuser and Eisenhauer (2008).

In addition, the in-house Ca isotope standard 'HPS Ca' was occasionally used as a secondary standard, giving $\delta^{44/42}Ca = 0.27 \pm 0.06\,‰$ (2s, n = 34) and $\delta^{43/42}Ca = 0.13 \pm 0.09\,‰$ (2s, n = 34).

### 2.2.4 Mass Spectrometer Setup

Ca isotopes have been measured extensively at the Department of Earth Sciences, University of Oxford, via ICP-MS using the sample-standard bracketing method (Halicz et al., 1999; Chu et al., 2006; Blättler et al., 2011; Reynard et al., 2011). These studies were conducted using the Nu Instruments Plasma MC-ICP-MS 'P2' at the Department of Earth Sciences, and I used the same basic setup for all analyses discussed in this thesis. The alignment of Faraday collectors is given in Table 2.1. This allows simultaneous measurement of $^{42}Ca^{+}$, $^{43}Ca^{+}$ and $^{44}Ca^{+}$. Furthermore, $^{87}Sr^{2+}$ is monitored to quantify and correct for isobaric $Sr^{2+}$ interferences.

Simultaneous measurement of three Ca isotopes allows discrimination between mass-

Table 2.1: Faraday cup arrangement for Ca isotope analysis (Halicz et al., 1999). Collector 6 is the axial detector. The instrument also features three ion counter detectors, not shown here. Masses are in amu.

| Collector: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Mass: | 45.5 | 45.0 | 44.5 | 44.25 | 44.0 | 43.75 | 43.5 | 43.25 | 43.0 | 42.5 | 42.0 | 41.5 |

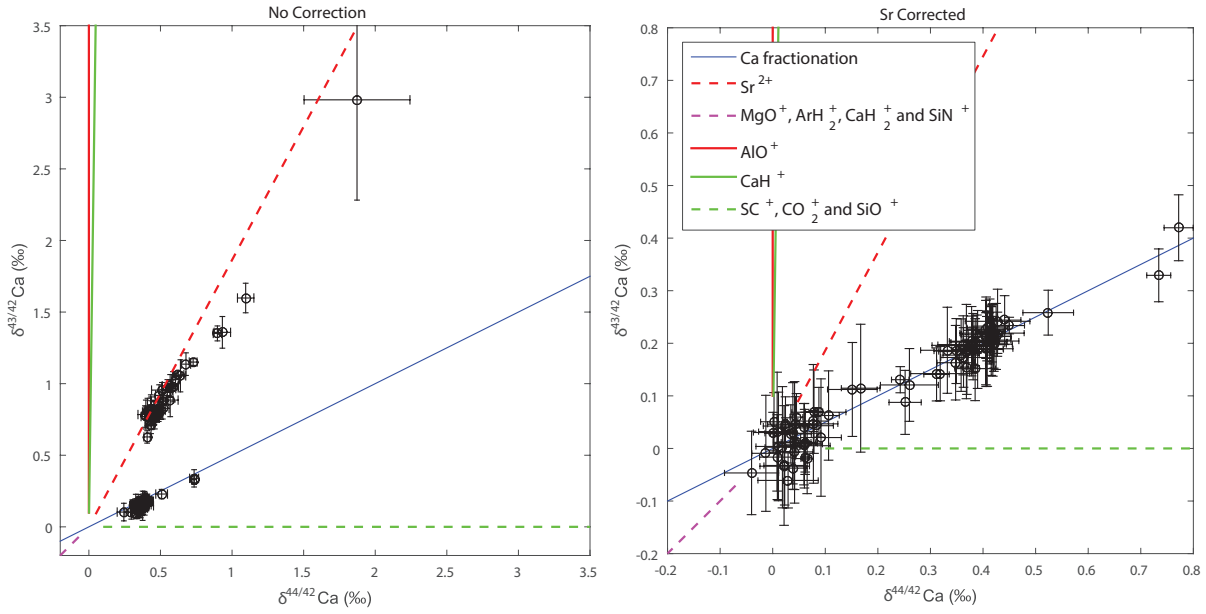



Figure 2.1: An example Ca triple-isotope plot, demonstrating the effects of mass-dependent Ca isotope fractionation (blue line) as well as offset vectors expected from various isobaric interferences. The example dataset plotted shows both mass-dependent fractionation and $Sr^{2+}$ interferences. The left-hand panel plots data processed without the Sr correction offset parallel to the $Sr^{2+}$ interference vector. The same dataset processed with the Sr correction (see Section 2.2.5.2) is shown on the right-hand panel, where all data lie on the mass-dependent fractionation line.

dependent isotope fractionation and contamination effects. This may be visualized on a Ca triple-isotope plot, a technique that is invaluable for method tests and data quality control. An example of this is shown in Figure 2.1. Interference-free analyses are expected to fall along the mass-dependent fractionation line while isobaric interferences offset measurements from this line. Figure 2.1 includes Sr-contaminated samples demonstrating a clear offset along the expected $Sr^{2+}$ interference vector. The measurement of $^{87}Sr^{2+}$ is important as it allows for correction of Sr interferences on the Ca isotopes. The details of this correction are discussed in Section 2.2.5.2.

I undertook method development work to optimize the existing Ca isotope workflow, developed primarily for marine carbonates, for Ca isotope analysis of cave carbonates. Importantly, speleothem carbonates and dripwaters are typically far 'cleaner' (i.e. lower

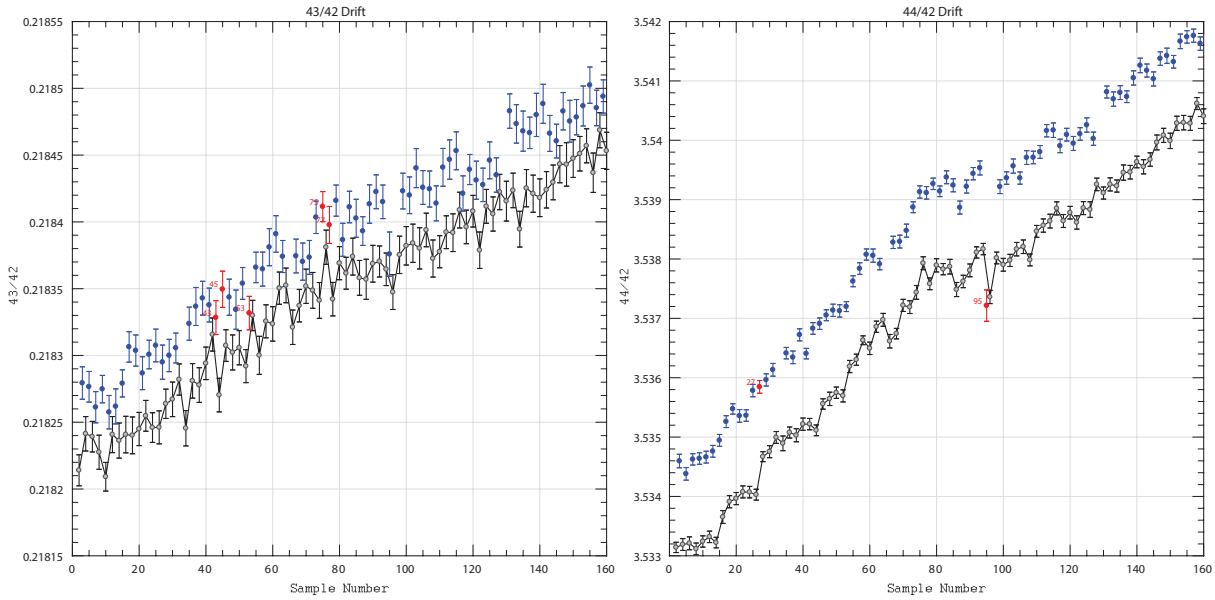



Figure 2.2: Drift in raw isotope ratios ($^{44}Ca/^{42}Ca$ and $^{43}Ca/^{42}Ca$) observed through a typical ICP-MS analysis session. This demonstrates the importance of closely-bracketed analyses. Bracketing standard measurements are shown in black, samples in blue, or red if they have been flagged as potentially erroneous — see Section 2.2.7. $\delta$-values may be visualised as the vertical offset between a sample measurement and the bracketing standard isotope ratio interpolated to that time (i.e. the black line).

non-Ca metal content) than the marine carbonates and seawater samples studied by previous workers, perhaps suggesting that a simpler and quicker sample preparation method would suffice, allowing for increased sample throughput. To determine the required performance of any revised Ca isotope separation method, trace element doping experiments were performed (Section 2.2.5). Following these tests, column calibration work and other method tests were performed to develop an optimized sample preparation method (Section 2.3).

### 2.2.5 Interference Testing

During ICP-MS analysis, there are two main instrumental signals that must be taken into consideration when interpreting raw output: space-charge effects (the preferential removal of light isotopes from the ion beam due to electrostatic repulsion between ions) and isobaric interferences (interferences on the detector due to contaminant ions having approximately the same m/z ratio as target ions). To determine accurate isotope ratios, both effects must be either removed or corrected for.

Space-charge effects are unavoidable, but may be quantified by running known standards through the mass spectrometer (a technique known as sample-standard bracketing), or by using a double-spike method (Section 2.2.1). All work presented in this thesis uses the sample-standard bracketing method. The magnitude of space-charge effects depends on the sample matrix and plasma conditions, both of which must be kept as constant as possible to obtain accurate results. Changes in plasma conditions over the course of an analysis session do occur, but do so gradually and do not typically affect closely-bracketed analyses (Figure 2.2). For Ca isotope analysis of carbonates, the sample matrix is dominated by the $^{40}Ca^{+}$ ion. It is therefore sufficient to concentration-match samples and standards to ensure matrix effects are kept constant. Samples were concentration-matched to within $\pm 10\,\%$ of standards to ensure no significant difference in matrix effect contributions to space-charge effects.

Potential isobaric interferences, and their predicted effects on a Ca triple-isotope plot, are shown in Figure 2.1. Of the potential sample-derived interferences, only $Sr^{2+}$ and $MgO^{+}$ are derived from metals present in significant quantities in speleothem material. If the potential impacts of these samples-derived interferences can be quantified or corrected for, criteria can be set to determine whether a given sample may be analysed without any separation chemistry, or whether a Ca purification step is necessary.

#### 2.2.5.1 $MgO^{+}$ Interferences

To investigate the effects of potential magnesium contamination in samples on Ca isotope analysis, clean calcium isotope standards were doped with variable amounts of magnesium and analysed for Ca isotopes. $MgO^{+}$ formation within the plasma, if occurring, is expected to offset doped samples to negative $\delta^{44/42}$Ca and $\delta^{43/42}$Ca with increasing Mg/Ca ratio (Figure 2.1).

I ran two experiments. The first experiment doped a Ca solution, derived from dissolution of 99.999 % $CaCO_3$ powder, with Mg to give Mg/Ca ratios in the range 0–40 mmol mol$^{-1}$, covering the typical range of speleothem Mg/Ca ratios. The second experiment doped NIST SRM 915a with Mg to give Mg/Ca ratios in the range 0–118 mmol mol$^{-1}$

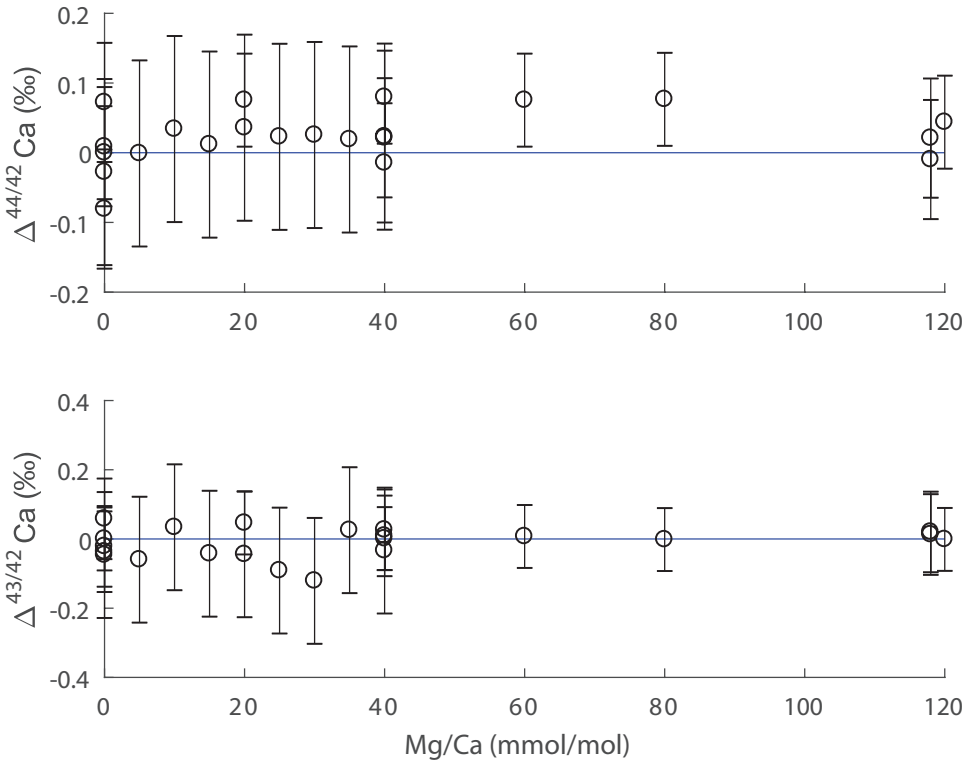



Figure 2.3: The effect of increasing sample Mg/Ca on Ca isotope ratios. Uncertainty is plotted as the 95 % confidence interval (t-distribution) on 3-5 repeat analyses.

to test for $MgO^+$ interferences at higher Mg/Ca ratios. Each sample was analysed three times at 10 ppm Ca using the sample-standard bracketing routine of Reynard et al. (2011). The results of these experiments are shown in Figure 2.3, normalised to their measured Ca isotope composition with no Mg contamination.

These results show that increasing Mg/Ca has no impact on measured $\delta^{44/42}$Ca or $\delta^{43/42}$Ca up to sample Mg/Ca ratios of 118 mmol mol$^{-1}$ with this analytical setup. Therefore, for natural samples with Mg/Ca ratios less than 118 mmol mol$^{-1}$, it is not necessary to separate Mg from Ca prior to Ca isotope analysis via ICP-MS.

#### 2.2.5.2 $Sr^{2+}$ Interference

I first investigated the effect of $Sr^{2+}$ interferences on Ca isotope analysis as part of the SrSpec column calibration (see Section 2.3.2.3). The experiment involved passing NIST SRM 915a, doped with Sr, through SrSpec columns filled with variable amounts of resin to quantify the minimum resin volume necessary to achieve separation. This resulted in aliquots of SRM 915a with variable Sr concentrations. Data from this experiment

are shown in Figure 2.7. The left panel shows data processed without an Sr correction. The right hand panel shows data processed with an Sr correction. NIST SRM 915b was analysed as a secondary standard and is also shown.

Results show data clearly offset along the predicted $Sr^{2+}$ interference vector. Correcting for $Sr^{2+}$ interferences using an assumed crustal isotope ratio of $Sr^{2+}$ ions brings all SrSpec column eluents to $\delta^{44/42}$Ca and $\delta^{43/42}$Ca values within uncertainty of zero. This strongly supports the conclusions that:

1. The columns did not fractionate Ca isotopes significantly (see Section 2.3.2.3).
2. The Sr-interference correction was able to fully correct the effects of Sr contamination up to Sr/Ca ratios of $\sim$35 mmol mol$^{-1}$ (the approximate Sr/Ca ratio of the most Sr contaminated sample)

These conclusions are further supported by the secondary standard (NIST SRM 915b). SRM 915b is known to contain Sr (Sr/Ca = 0.17 mmol mol$^{-1}$; Section 2.2.3), explaining its offset along the Sr interference vector. Once corrected, SRM 915b falls along the Ca mass fractionation line and gives a $\delta^{44/42}$Ca value of 0.36 ‰, which compares well to published values (Section 2.2.3). Following this result, all subsequent analytical runs include SRM 915b as a check on the continuing accuracy of the Sr correction.

Overall the investigation of $Sr^{2+}$ interferences shows that Sr causes a significant offset to higher $\delta^{44/42}$Ca and $\delta^{43/42}$Ca, however this offset may be accurately corrected for using the m/z = 43.5 beam to compute, and correct for, the expected interferences by $^{84}Sr^{2+}$ on $^{42}Ca^{+}$, $^{86}Sr^{2+}$ on $^{43}Ca^{+}$ and $^{88}Sr^{2+}$ on $^{44}Ca^{+}$. The magnitude of the Sr interference depends strongly on the $Sr^{2+}$/Sr ionization ratio in the plasma (Reynard et al., 2011). This ratio may vary significantly over the course of a sample analysis due to fluctuations in plasma conditions. As such, it is advisable to calculate and apply the Sr-correction at the smallest possible time-step (one second voltage integrations in this setup) for the greatest accuracy during data processing. The accuracy of this correction may be further improved by analysis of a pure Sr standard to calibrate the $Sr^{2+}$ isotope ratios, rather than assuming crustal ratios (pers. comm. M. Bender).

Table 2.2: The Ca isotope analysis routine used here. The method takes up to fifteen samples (labelled 1-15 here), several of which may be replaced by secondary standards. The tabulated routine is repeated five times, giving a total running time of 18 hours, 21 minutes, 14 seconds.

| Analysis | Time Elapsed | Sample | Analysis | Time Elapsed | Sample |
|---|---|---|---|---|---|
| 1 | 00:00:00 | Sr Standard | 17 | 01:50:08 | 8 |
| 2 | 00:06:53 | SRM 915a | 18 | 01:57:01 | SRM 915a |
| 3 | 00:13:46 | 1 | 19 | 02:03:54 | 9 |
| 4 | 00:20:39 | SRM 915a | 20 | 02:10:47 | SRM 915a |
| 5 | 00:27:32 | 2 | 21 | 02:17:39 | 10 |
| 6 | 00:34:25 | SRM 915a | 22 | 02:24:32 | SRM 915a |
| 7 | 00:41:18 | 3 | 23 | 02:31:25 | 11 |
| 8 | 00:48:11 | SRM 915a | 24 | 02:38:18 | SRM 915a |
| 9 | 00:55:04 | 4 | 25 | 02:45:11 | 12 |
| 10 | 01:01:57 | SRM 915a | 26 | 02:52:04 | SRM 915a |
| 11 | 01:08:50 | 5 | 27 | 02:58:57 | 13 |
| 12 | 01:15:43 | SRM 915a | 28 | 03:05:50 | SRM 915a |
| 13 | 01:22:36 | 6 | 29 | 03:12:43 | 14 |
| 14 | 01:29:29 | SRM 915a | 30 | 03:19:36 | SRM 915a |
| 15 | 01:36:22 | 7 | 31 | 03:26:29 | 15 |
| 16 | 01:43:15 | SRM 915a | 32 | 03:33:22 | SRM 915a |

### 2.2.6 Final Ca Isotope Routine

Following the testing and calibration work described above, I devised a modified standard analysis sequence for Ca isotopes. The run sequence is shown in Table 2.2. The aims of this setup are to:

- Provide five repeat measurements of each sample, allowing reasonable sample statistics even if one analysis is rendered invalid by, for example, plasma instability.
- Provide closely-bracketed analyses to minimise uncertainty when interpolating the standard isotope ratio between measurements.
- Include measurement of a pure Sr solution to determine $Sr^{2+}$ isotopic ratios in the plasma, improving the accuracy of the Sr correction.
- Analyse the maximum possible number of samples in a run time of ∼18 hours. This allows time for instrument tuning, concentration checks, sample dilution, reagent preparation and a full analysis sequence in a 24-hour session, convenient for running many analyses over the course of a week, for example.

These criteria were chosen to provide a standard overnight run sequence that provides

excellent data quality with built-in support for a calibrated Sr interference correction.

The mass spectrometer was tuned for maximum sensitivity and stability on the m/z = 44 beam ($^{44}Ca^{+}$). Typically sensitivities achieved were $\sim$0.8 V ppm$^{-1}$, although this varied greatly depending on the condition of the sampler and skimmer cones, and the nebuliser used. Analyses were made at either 5 ppm or 10 ppm Ca, depending on instrument sensitivity on the day. Samples were typically diluted to 95 % of the standard concentration — this pre-empts the over-concentration that typically occurs to samples during an overnight run due to evaporation. Sampler and skimmer cones were cleaned before multi-day analytical sessions to prevent occlusion of the aperture and retain high sensitivity.

This sequence supports fifteen samples. Typically I assign secondary standards to at least one of these positions to quantify accuracy and precision. NIST SRM 915b is an excellent candidate for this as it also provides a test of the $Sr^{2+}$ correction. The Sr standard is a dilute ($\sim$80 ppb Sr) Sr solution prepared from a commercially-available ICP-MS elemental standard. $Sr^{2+}$ signals detected during analysis of the Sr standard allow calibration of the Sr interference correction applied to Ca-bearing samples. Repeated analysis of the Sr standard throughout the run provides a check on the stability of the plasma Sr isotope ionisation ratios over the course of the run — typically they remain constant over the course of a single analytical session.

Unless otherwise stated, Ca isotope data reported in this thesis were analysed by this method and are reported after Sr-correction. Data processing and quality control was largely automated using the data processing software developed and briefly described in Section 2.2.7.

### 2.2.7 Data Processing

Data processing for all Ca isotope analyses was performed offline using MATLAB. A set of scripts (nicknamed 'cadr') were written to automate processing of raw voltage data into $\delta^{44/42}$Ca and $\delta^{43/42}$Ca values. This code handles Sr correction, background correction, isotope ratio calculation, drift analysis, $\delta$-value standardisation, and $\delta$-value summation

for repeat measurements. These scripts perform error propagation through all calculation steps and flag samples measured during periods of drift instability as potentially inaccurate.

These scripts provide a standardised processing workflow for all analyses. They also produce routine plots to allow immediate visualisation of both instrument performance and data quality. Examples of these automatically generated plots include Figures 2.2 & 2.1.

## 2.3 Ca Isotope Sample Preparation

### 2.3.1 Why isolate Ca?

It is common in isotope geochemistry to isolate the element of interest from its matrix (that is, other elements/isotopes that are not of analytical interest) prior to analysis via ICP-MS. This is because matrix impurities may offset measured isotope ratios relative to the true sample ratio due to isobaric inferences and/or space-charge effects occurring within the mass spectrometer (Section 2.2.5).

Testing the effects of Mg and Sr impurities on measured Ca isotopes ratios has quantified acceptable tolerances for these impurities in speleothem samples (Section 2.2.5). Mg contamination was shown to have no effect on measured Ca isotope ratios up to at least Mg/Ca ratios of 118 $\mathrm{mmol\,mol^{-1}}$, and Sr contamination may be corrected for accurately up to Sr/Ca ratios of at least 35 $\mathrm{mmol\,mol^{-1}}$. These limits are more than generous enough to allow most speleothem carbonates to be analysed without separation chemistry. However, analysis of other materials (e.g. carbonate bedrock, dripwaters) may still require separation chemistry to isolate the sample's Ca content.

In this section I discuss development work on two Ca separation methods: a traditional column chemistry procedure (Section 2.3.2) and an automated method using an ESI prepFAST-MC system (Section 2.3.3). For each method I discuss testing and development work undertaken, following by a description of the finalised procedures used elsewhere in

this thesis. The relative merits of the column and prepFAST-MC methods are considered briefly in Section 2.3.3.3.

### 2.3.2 Column Chemistry

Ion-exchange resins have long been used for elemental separation and purification in a wide range of chemical fields. Broadly, column separation methods load a dissolved sample onto a chromatographic column holding an ion-exchange resin. Due to eluent-specific binding of specific ions to the resin, different elements may then be eluted off the column using different volumes of eluent and/or different eluents.

The method used here to isolate Ca from its matrix elements is based on the procedure developed by Chu et al. (2006). The procedure involves two steps; firstly, Ca and Sr are isolated from other matrix elements using AG50W-X12 resin, and then, with a second column, Ca is isolated from Sr using Eichrom SrSpec resin (Section 2.3.2.1).

Due to purchase of a new batch of SrSpec resin, a comparative study was necessary to cross-calibrate this with older resin stock. This calibration work is presented in Section 2.3.2.2. Further tests (Section 2.3.2.3) were conducted to precisely determine column yields for the SrSpec column, due to concerns from other researchers about potential for incomplete yields and isotope fractionation (personal communications; C. Day & P. Pogge Von Strandmann). No further calibration work was necessary for the AG50W-X12 column step, which uses the procedure of Chu et al. (2006), given in Table 2.3.

#### 2.3.2.1 SrSpec Resin

SrSpec resin was originally designed to facilitate isolation of Sr from a variety of matrix elements. The particular motivation of this was to facilitate accurate quantification of $^{90}$Sr (a radionuclide produced during nuclear fission) in environmental samples (Horwitz et al., 1992). The resin is known to remove Ca from Sr samples extremely efficiently, and has been used widely in Sr isotope studies (e.g. Pin et al., 1994; Galy et al., 1999; Fortunato et al., 2004). However, Sr yield is not quantitative, with $\sim$5 % of loaded Sr

Table 2.3: Column recipe for Ca & Sr separation, modified from Chu et al. (2006). This process may be used to remove other matrix elements from samples prior to Ca-Sr separation using SrSpec columns.

| Step | Reagent | Volume (mL) |
|---|---|---|
| Condition | 6 M HCl | 6.0 |
| Condition | 2 M HCl | 6.0 |
| Load Ca | Sample (in 2 M HCl) | 0.5 |
| Elute matrix | 2 M HCl | 0.5 |
| Elute matrix | 2 M HCl | 1.0 |
| Elute matrix | 2 M HCl | 10.0 |
| Collect Ca & Sr | 6 M HCl | 2.0 |
| Collect Ca & Sr | 6 M HCl | 10.0 |
| Clean | 6 M HCl | 5.0 |
| Clean | $H_2O$ | 10.0 |

retained on the column (Pin and Bassin, 1992).

The use of SrSpec resin to purify Ca samples is not as well validated as is the case for Sr. While several studies have used the resin for this purpose (e.g. Chu et al., 2006; Tipper et al., 2006), questions remain concerning its performance. A major motivation of the calibration work presented in Sections 2.3.2.2 & 2.3.2.3 was to more accurately characterise the use of this resin for Ca isolation from Sr, and in particular address the following questions:

- What Ca yield can be achieved?
- If the maximum Ca yield is less than 100 %, is there any significant isotope fractionation associated with retention of Ca on the resin?
- How much Sr remains in the Ca cut after column chemistry?

#### 2.3.2.2 SrSpec Column Calibration - Elution Curves

A re-determination of the elution curves of the SrSpec resin was necessary to compare two resin batches (FSRS110202 and FSRS120411) for use in the laboratory. A secondary purpose of this experiment was to test whether the existing column procedure adequately captured the elution peaks of Ca and Sr.

This experiment used two custom-formed columns (resin volume ~250 µL, resin height

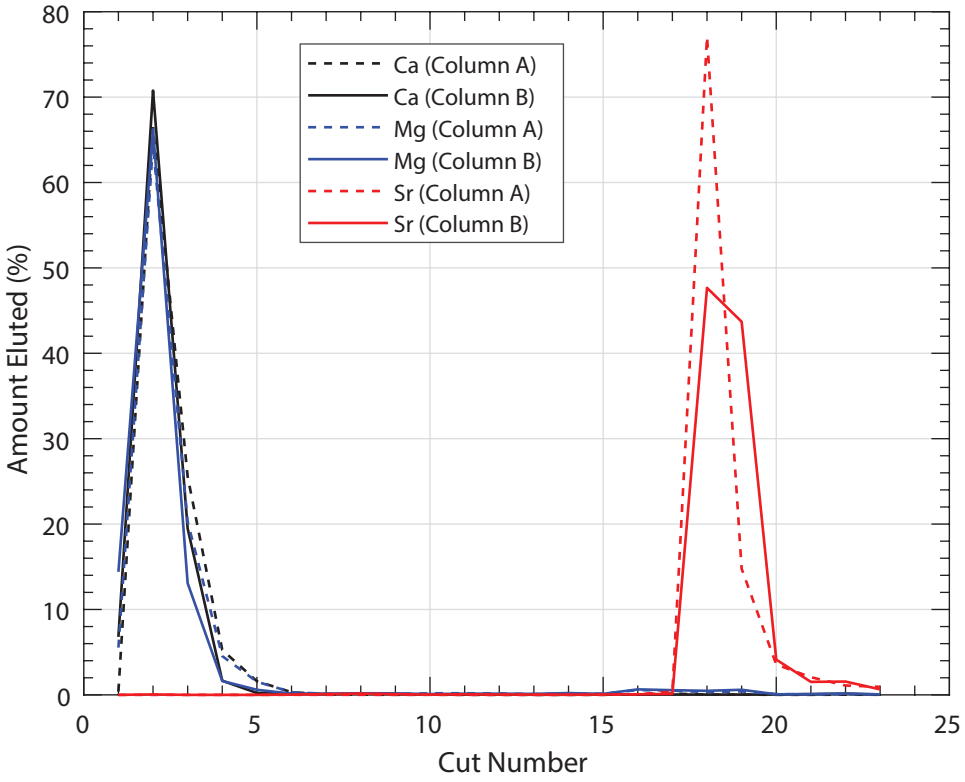



Figure 2.4: SrSpec Column calibration. Each cut was 250 µL. Cuts 1-15 eluted 3 M $HNO_3$, while cuts 16-23 eluted in 0.01 M $HNO_3$. Dashed lines show elution curves for Column A (resin batch FSRS110202). Solid lines show elution curves for Column B (resin batch FSRS120411).

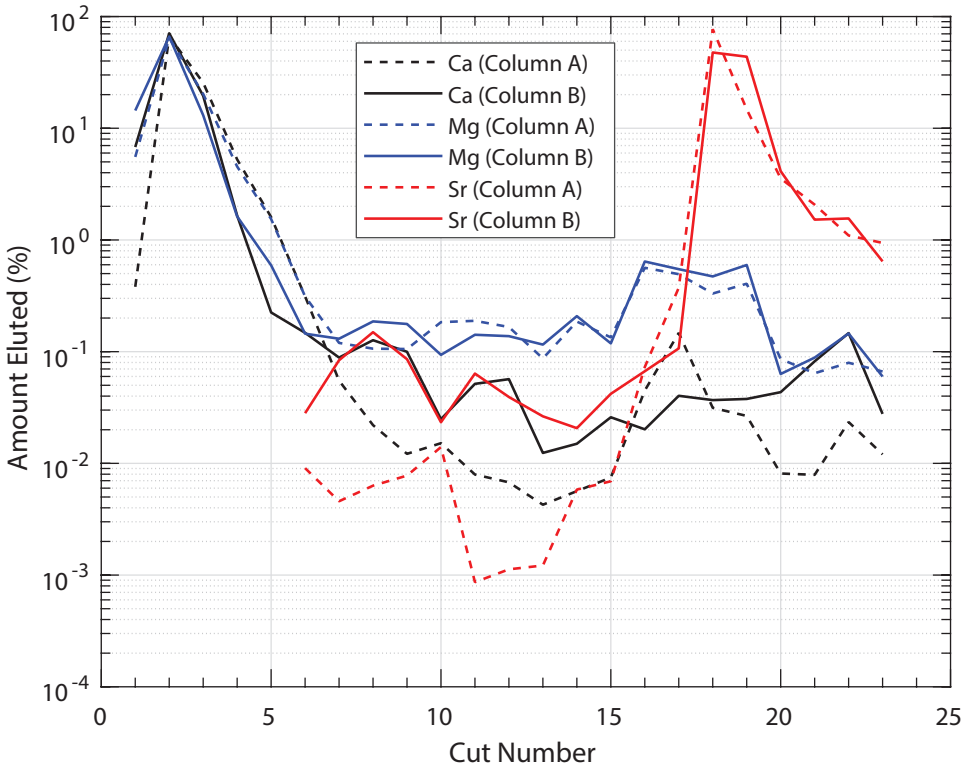



Figure 2.5: SrSpec Column calibration, plotting on a logarithmic y-axis to better show peak tailing. Data is identical to that shown in Figure 2.4. Dashed lines show elution curves for Column A (resin batch FSRS110202). Solid lines show elution curves for Column B (resin batch FSRS120411).

~1.5 cm, reservoir volume ~2 mL, each loaded with SrSpec resin to capacity (approximately 2 cm resin height). Column A used SrSpec resin batch FSRS110202 and column B used SrSpec resin batch FSRS120411. This purpose of this calibration was to determine the elution curves of Ca, Mg and Sr through the columns and to compare the two resin batches. Prior to loading onto columns, resin was batch cleaned to remove fine particles. This was done by suspending the resin in MQ water and allowing partial settling before removing the supernatant. The process was repeated five times for each batch. After loading onto the columns the resins were chemically cleaned with 1000 µL 6 M HCl, 1000 µL MQ and 2000 µL 0.01 M $HNO_3$ and conditioned with 2400 µL 3 M $HNO_3$, as described in Table 2.8.

Each column was loaded with 250 µL of a gravimetrically prepared solution containing Ca (360 ppm), Mg (4.2 ppm) and Sr (296 ppb) in 3 M $HNO_3$. These concentrations were chosen to mimic a typical speleothem sample. Calcium and magnesium were eluted off the columns by incremental addition of a further fourteen 250 µL aliquots of 3M $HNO_3$. Each of the fifteen cuts was collected in a pre-weighed 2 mL Eppendorf tube. A further eight cuts were then taken on addition of eight 250 µL aliquots of 0.01 M $HNO_3$ to characterise the strontium elution curve. All cuts were weighed after collection to precisely determine the amounts eluted. Ca and Mg are expected to pass through the column rapidly, while Sr is bound to the resin in 3 M $HNO_3$ and only re-mobilised when eluted by 0.01 M $HNO_3$ (Horwitz et al., 1992).

| | | | Elemental Content | | | | | | Percentage of Total | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Cut | Eluent | Mass Eluted (mg) | Ca (ppb) | Mg (ppb) | Sr (ppb) | Ca (ng) | Mg (ng) | Sr (ng) | Ca (%) | Mg (%) | Sr (%) |
| A1 | 3 M $HNO_3$ | 288.44 | 1413 | 262 | bdl. | 408 | 75.6 | - | 0 | 6 | - |
| A2 | 3 M $HNO_3$ | 271.27 | 262200 | 3245 | bdl. | 71127 | 880 | - | 66 | 64 | - |
| A3 | 3 M $HNO_3$ | 257.90 | 106089 | 1073 | bdl. | 27360 | 277 | - | 26 | 20 | - |
| A4 | 3 M $HNO_3$ | 268.21 | 20982 | 231 | bdl. | 5628 | 62.0 | - | 5 | 5 | - |
| A5 | 3 M $HNO_3$ | 264.46 | 6482 | 79.4 | bdl. | 1714 | 21.0 | - | 2 | 2 | - |
| A6 | 3 M $HNO_3$ | 277.28 | 1207 | 15.4 | 0.026 | 335 | 4.26 | 0.007 | 0 | 0 | 0 |
| A7 | 3 M $HNO_3$ | 274.45 | 214.6 | 5.96 | 0.013 | 58.9 | 1.64 | 0.004 | 0 | 0 | 0 |
| A8 | 3 M $HNO_3$ | 261.15 | 90.1 | 5.56 | 0.019 | 23.5 | 1.45 | 0.005 | 0 | 0 | 0 |
| A9 | 3 M $HNO_3$ | 271.83 | 47.9 | 5.31 | 0.023 | 13.0 | 1.44 | 0.006 | 0 | 0 | 0 |
| A10 | 3 M $HNO_3$ | 280.83 | 57.8 | 8.94 | 0.040 | 16.2 | 2.51 | 0.011 | 0 | 0 | 0 |
| A11 | 3 M $HNO_3$ | 266.19 | 32.0 | 9.69 | 0.003 | 8.51 | 2.58 | 0.001 | 0 | 0 | 0 |
| A12 | 3 M $HNO_3$ | 280.23 | 25.8 | 8.10 | 0.003 | 7.24 | 2.27 | 0.001 | 0 | 0 | 0 |
| A13 | 3 M $HNO_3$ | 267.18 | 17.1 | 4.45 | 0.004 | 4.56 | 1.19 | 0.001 | 0 | 0 | 0 |
| A14 | 3 M $HNO_3$ | 272.06 | 22.2 | 9.33 | 0.017 | 6.04 | 2.54 | 0.005 | 0 | 0 | 0 |
| A15 | 3 M $HNO_3$ | 282.27 | 28.3 | 6.50 | 0.020 | 7.99 | 1.83 | 0.006 | 0 | 0 | 0 |
| A16 | 0.01 M $HNO_3$ | 210.20 | 229 | 36.6 | 0.277 | 48.1 | 7.70 | 0.058 | 0 | 1 | 0 |
| A17 | 0.01 M $HNO_3$ | 249.00 | 633 | 27.1 | 1.20 | 158 | 6.75 | 0.299 | 0 | 0 | 0 |
| A18 | 0.01 M $HNO_3$ | 254.69 | 132 | 17.8 | 242 | 33.6 | 4.53 | 61.7 | 0 | 0 | 77 |
| A19 | 0.01 M $HNO_3$ | 245.59 | 116 | 22.6 | 48.1 | 28.5 | 5.54 | 11.8 | 0 | 0 | 15 |
| A20 | 0.01 M $HNO_3$ | 244.84 | 35.5 | 4.79 | 11.6 | 8.68 | 1.17 | 2.84 | 0 | 0 | 4 |
| A21 | 0.01 M $HNO_3$ | 254.63 | 33.2 | 3.44 | 6.51 | 8.45 | 0.875 | 1.66 | 0 | 0 | 2 |
| A22 | 0.01 M $HNO_3$ | 236.45 | 106 | 4.60 | 3.71 | 25.1 | 1.09 | 0.878 | 0 | 0 | 1 |
| A23 | 0.01 M $HNO_3$ | 247.51 | 52.1 | 3.67 | 3.04 | 12.9 | 0.907 | 0.753 | 0 | 0 | 1 |
| B1 | 3 M $HNO_3$ | 285.22 | 25728 | 690 | bdl. | 7338 | 197 | - | 7 | 14 | - |
| B2 | 3 M $HNO_3$ | 283.30 | 270041 | 3196 | 0.146 | 76503 | 905 | 0.041 | 71 | 66 | 0 |
| B3 | 3 M $HNO_3$ | 270.53 | 77867 | 661 | bdl. | 21065 | 179 | - | 19 | 13 | - |
| B4 | 3 M $HNO_3$ | 265.24 | 6730 | 83.7 | bdl. | 1785 | 22.2 | - | 2 | 2 | - |
| B5 | 3 M $HNO_3$ | 261.44 | 928 | 31.0 | bdl. | 243 | 8.11 | - | 0 | 1 | - |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| B6 | 3 M $HNO_3$ | 274.63 | 581 | 7.21 | 0.083 | 159 | 1.98 | 0.023 | 0 | 0 | 0 |
| B7 | 3 M $HNO_3$ | 263.10 | 364 | 6.78 | 0.259 | 95.7 | 1.78 | 0.068 | 0 | 0 | 0 |
| B8 | 3 M $HNO_3$ | 257.39 | 532 | 9.91 | 0.471 | 137 | 2.55 | 0.121 | 0 | 0 | 0 |
| B9 | 3 M $HNO_3$ | 280.72 | 384 | 8.62 | 0.248 | 108 | 2.42 | 0.070 | 0 | 0 | 0 |
| B10 | 3 M $HNO_3$ | 234.50 | 115 | 5.46 | 0.081 | 26.9 | 1.28 | 0.019 | 0 | 0 | 0 |
| B11 | 3 M $HNO_3$ | 297.74 | 187 | 6.50 | 0.174 | 55.7 | 1.94 | 0.052 | 0 | 0 | 0 |
| B12 | 3 M $HNO_3$ | 261.83 | 234 | 7.19 | 0.122 | 61.3 | 1.88 | 0.032 | 0 | 0 | 0 |
| B13 | 3 M $HNO_3$ | 270.56 | 49.6 | 5.84 | 0.079 | 13.4 | 1.58 | 0.021 | 0 | 0 | 0 |
| B14 | 3 M $HNO_3$ | 258.75 | 62.6 | 11.0 | 0.065 | 16.2 | 2.84 | 0.017 | 0 | 0 | 0 |
| B15 | 3 M $HNO_3$ | 286.58 | 97.6 | 5.66 | 0.119 | 28.0 | 1.62 | 0.034 | 0 | 0 | 0 |
| B16 | 0.01 M $HNO_3$ | 200.52 | 109 | 43.7 | 0.271 | 21.8 | 8.77 | 0.054 | 0 | 1 | 0 |
| B17 | 0.01 M $HNO_3$ | 235.20 | 185 | 31.8 | 0.369 | 43.5 | 7.49 | 0.087 | 0 | 1 | 0 |
| B18 | 0.01 M $HNO_3$ | 249.55 | 160 | 25.8 | 155 | 39.8 | 6.44 | 38.7 | 0 | 0 | 48 |
| B19 | 0.01 M $HNO_3$ | 245.59 | 166 | 33.2 | 145 | 40.8 | 8.16 | 35.5 | 0 | 1 | 44 |
| B20 | 0.01 M $HNO_3$ | 248.88 | 189 | 3.47 | 13.5 | 46.9 | 0.863 | 3.35 | 0 | 0 | 4 |
| B21 | 0.01 M $HNO_3$ | 202.68 | 437 | 5.96 | 6.09 | 88.5 | 1.21 | 1.23 | 0 | 0 | 2 |
| B22 | 0.01 M $HNO_3$ | 281.06 | 564 | 7.07 | 4.50 | 159 | 1.99 | 1.27 | 0 | 0 | 2 |
| B23 | 0.01 M $HNO_3$ | 255.49 | 117 | 3.20 | 2.05 | 30.0 | 0.818 | 0.523 | 0 | 0 | 1 |

Table 2.4: Column calibration data for SrSpec columns. bdl = below detection limit. Percentages are calculated relative to the total eluted by each column.

All column cuts were analysed for Ca, Mg and Sr content using an Element 2 ICP-MS (Section 2.4). Results are given in Table 2.4 and elution curves are plotted in Figures 2.4 & 2.5. Figure 2.4 shows elution curves for Ca, Mg and Sr through both columns, while Figure 2.5 shows the same data plotted on logarithmic axes, allowing better visualisation of the elution peak tailing. Both Ca and Mg elute rapidly in 3 M $HNO_3$, and are fully eluted in the first seven cuts. As expected, Sr remains on the column while Mg and Ca are eluted and is only mobilised on addition of 0.01 M $HNO_3$, starting at cut 16. Figure 2.5 shows that tailing of the Sr elution peak is not fully characterised by this experiment — a larger elution volume would be necessary.

The Ca and Mg elution behaviours of the two resin batches appear to be identical. Sr elution is also similar between the columns; Column A (resin batch FSRS110202) shows a systematically lower Sr blank in the 3 M $HNO_3$ cuts, however the amount detected remains analytically insignificant. Interestingly, a small increase in Mg signal is seen in both columns through cuts 16-19. This may reflect a true elution signal however analytical difficulties during analysis of the Sr elution cuts could also contribute.

This initial column calibration confirms that the SrSpec resin batches FSRS110202 and FSRS120411 exhibit similar elution curves for Ca, Mg and Sr. Ca and Mg are washed through the column in 3 M $HNO_3$ while Sr is retained until washed off by a very weak acid (0.01 M $HNO_3$). Fifteen cuts of 250 µL 3 M $HNO_3$ appear more than adequate to collect the full Ca and Mg elution, with concentrations falling to background levels after the seventh cut (a total of 1750 µL 3 M $HNO_3$). Complete collection of the Sr peak was not achieved with the eight cuts of 250 µL 3 M used here. Clearly a larger volume would be needed for use cases where quantitative Sr recovery is of interest.

Due to minor sample loss during collection of small increments and accumulated blanks, this experiment is not suited to accurate quantification of the Ca and Sr yields through the column. Having determined the elution curves here, a second experiment was conducted to accurately quantify yields. That work is discussed in Section 2.3.2.3.

Table 2.5: Synthetic standards for column yield experiments (Section 2.3.2.3). Solution A was used as the mock sample loaded onto the columns. Sample B is Sr-free, and was used for sample drying and dissolution tests. Both solutions were prepared from NIST SRM 915a.

| | Solution A | Solution B |
|---|---|---|
| Ca (ppm) | 450 | 433 |
| Sr (ppb) | 489 | - |
| Sr/Ca $\mathrm{mmol\,mol^{-1}}$ | 0.50 | - |
| Ca (µg in 250 µL) | 112.50 | 122.25 |

Table 2.6: SrSpec column yields test: elemental yields (top) and Ca isotope data (bottom). Ca isotope data are reported twice, once with the Sr correction applied and once without. Note that the uncertainties reported on uncorrected data are likely underestimates as changes in plasma conditions (and hence $Sr^{2+}/Sr$ ratios) are not taken into consideration.

| Column | Resin Mass - Dry (mg) | Resin Volume (µL) | Ca Cut Ca (µg) | Ca Cut Sr (ng) | Sr Cut Ca (µg) | Sr Cut Sr (ng) |
|---|---|---|---|---|---|---|
| 1 | 67.16 | 203.5 | 106.7 | 2 | 0.2 | 118 |
| 2 | 41.18 | 124.8 | 103.3 | 2 | 1.6 | 116 |
| 3 | 19.81 | 60.0 | 106.7 | 14 | 0.4 | 101 |
| 5 | 10.53 | 31.9 | 107.7 | 60 | 0.5 | 52 |
| 6 | 8.20 | 24.8 | 106.4 | 80 | 1.4 | 37 |

| | Sr-Corrected | | | | Uncorrected | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Sample | $\delta^{44/42}$Ca (‰ 915a) | C.I. (±) | $\delta^{43/42}$Ca (‰ 915a) | C.I. (±) | $\delta^{44/42}$Ca (‰ 915a) | C.I. (±) | $\delta^{43/42}$Ca (‰ 915a) | C.I. (±) | n |
| Column 1 | 0.04 | 0.13 | -0.02 | 0.18 | 0.04 | 0.13 | 0.00 | 0.18 | 3 |
| Column 2 | 0.08 | 0.09 | 0.06 | 0.12 | 0.07 | 0.09 | 0.06 | 0.12 | 4 |
| Column 3 | 0.08 | 0.09 | 0.00 | 0.12 | 0.33 | 0.09 | 0.47 | 0.12 | 4 |
| Column 5 | 0.00 | 0.13 | -0.04 | 0.18 | 1.17 | 0.13 | 2.11 | 0.18 | 3 |
| Column 6 | -0.01 | 0.09 | 0.02 | 0.12 | 1.47 | 0.09 | 2.76 | 0.12 | 4 |
| SB-A | 0.07 | 0.09 | 0.05 | 0.12 | 0.03 | 0.09 | 0.01 | 0.12 | 4 |
| SB-B | 0.04 | 0.13 | 0.07 | 0.18 | 0.01 | 0.13 | 0.01 | 0.18 | 3 |
| SolnB | -0.03 | 0.13 | -0.03 | 0.18 | -0.03 | 0.13 | -0.01 | 0.18 | 3 |

#### 2.3.2.3 SrSpec Column Calibration - Yields

Following determination of Ca, Mg & Sr elution curves for the SrSpec resin (Section 2.3.2.2), I conducted a second calibration experiment to accurately determine the Ca and Sr yields through the SrSpec columns, and whether Ca isotopes are fractionated by the SrSpec column. In addition, calculations of the Sr capacity of SrSpec resin show that the resin utilisation during the elution experiments was only 0.15 %, far lower than the 10 % considered safe by Horwitz et al. (1992). As such it may also be possible to use a lower resin

Table 2.7: Calculated yield percentages for SrSpec columns. Raw data are given in Table 2.6.

| Column | Eluted in Ca cut | | Eluted in Sr cut | | Not Eluted | |
|---|---|---|---|---|---|---|
| | Ca (%) | Sr (%) | Ca (%) | Sr (%) | Ca (%) | Sr (%) |
| 1 | 95 | 1.5 | 0.2 | 97 | 4.9 | 1.9 |
| 2 | 92 | 1.4 | 1.5 | 95 | 6.6 | 3.4 |
| 3 | 95 | 12 | 0.3 | 82 | 4.8 | 5.9 |
| 5 | 96 | 49 | 0.5 | 43 | 3.8 | 8.7 |
| 6 | 95 | 66 | 1.2 | 30 | 4.2 | 4.1 |

volume, thereby decreasing elution times and laboratory expenses.

To investigate these effects, five column were prepared, loaded with resin volumes (200–25 µL) corresponding to resin heights of 2 cm down to a few millimetres. The 200 µL volume is comparable to the resin height used for the elution curve calibrations in Section 2.3.2.2. Each column was cleaned and pre-conditioned as described in Table 2.8 then loaded with 250 µL of a mixed Ca-Sr synthetic standard (Solution A), prepared gravimetrically from NIST SRM 915a and a commercially-available 1000 ppm ICP-MS Sr standard (Table 2.5).

For each column, Ca cuts were eluted first using 2 mL 3 M $HNO_3$, followed by Sr cuts in 3 mL 0.01 M $HNO_3$. All cuts were collected in Teflon beakers, and subsequently dried on a hotplate. Samples were dried to allow re-dissolution in a known volume of acid, as well as to help oxidise any organic material washed off the columns, which may otherwise interfere with ICP-MS analysis of the cuts. Dried samples were re-dissolved in 4 mL 2 % $HNO_3$ in preparation for chemical and isotopic analysis. In addition, two aliquots of Solution B (labelled SB-A and SB-B) were dried and re-dissolved in identical fashion to the column cuts. This was done to test for any Ca loss or isotope fractionation during the drying and re-dissolution processes.

Sr cuts, along with Ca cuts for columns 5 and 6 and an aliquot of Solution A, were analysed for Ca and Sr concentrations by quadrupole ICP-MS. All Ca cuts, SB-A, SB-B and an aliquot of Solution B were analysed for Ca concentration and isotopes following the method described in Section 2.2.6. Measurement of the 43.5 signal during isotope analysis, calibrated using the known Sr/Ca ratios of the Ca cuts of columns 5 and 6,

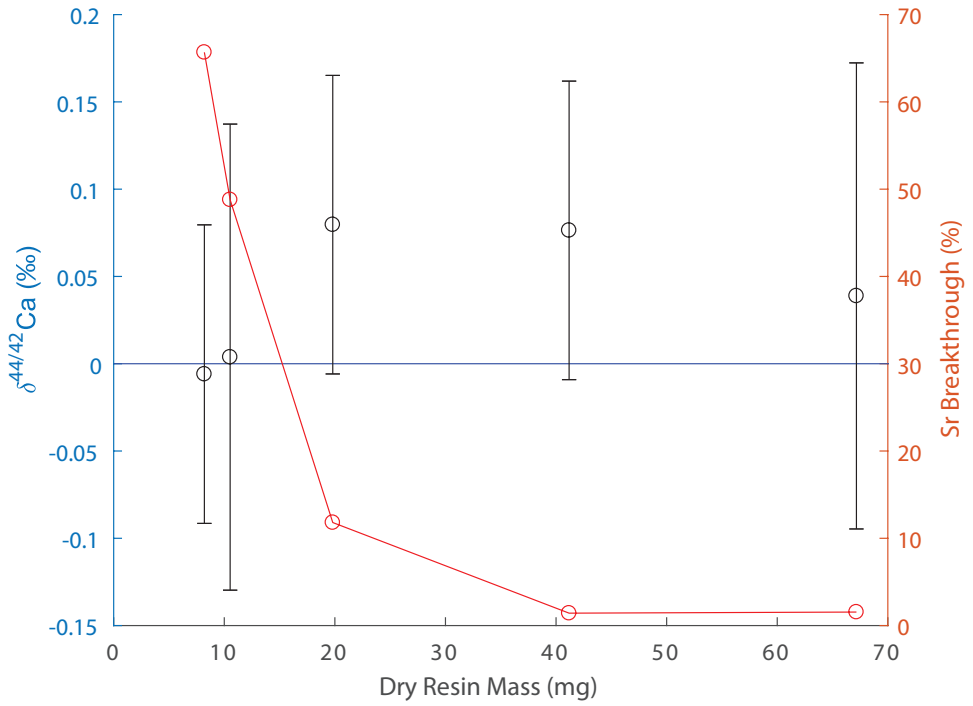



Figure 2.6: Ca isotope fractionation through SrSpec columns. Ca isotope data are plotted in black and Sr breakthrough percentages in red.

allowed calculation of the Sr/Ca ratios of those samples not analysed by quadruple ICP-MS. The synthesised dataset is given in Table 2.6. Elemental concentrations allow for yield calculations relative to the direct measurement of Solution A, the results of which are given in Table 2.7.

The Ca contents of SB-A and SB-B were found to be 108 µg and 110 µg, respectively. These values are, within uncertainty, equal to the Ca content of Solution B (250 µL; Table 2.5). In addition the Ca isotopic compositions of both SB-A and SB-B are indistinguishable from Solution B (Table 2.6). These data confirm that no Ca is lost or fractionated during the drying or re-dissolution steps.

Column Ca yield data (Table 2.7) shows Ca yields 92–96 % in the Ca cut for all columns, with a further 0.2–1.5 % of load Ca eluted into the Sr cut. There is no clear relationship between Ca yield and resin mass. The residual 3.8–6.6 % of Ca load is unaccounted for in either elution, and is presumably retained on the column.

Calcium isotope data for the Ca cuts are plotted in Figure 2.6. All data are within uncertainty of the value of NIST SRM 915a; however, Ca cuts from columns 1-3 all show a slight offset to positive $\delta^{44/42}$Ca, perhaps suggesting a slight isotope fractionation

associated with column processing. This offset, combined with the incomplete yields noted previously, suggests that perhaps a small amount of isotopically light Ca binds to the resin and is not eluted under the conditions used here. This interpretation would agree with experiments conducted by P. Pogge Von Strandmann, who found that at low Ca loads (20 µg Ca), $\delta^{44/42}$Ca could be elevated by as much as 0.35 ‰, with the offset decreasing as the Ca load increases (pers. comm. P. Pogge Von Strandmann). This suggests that, while the fractionation is within analytical uncertainty at the Ca loads used here, it may be significant if smaller samples were processed through SrSpec columns.

Sr yield data (Table 2.7) show a strong correlation with resin volume. With larger resin masses (Columns 1 and 2), >95 % of load Sr is eluted in the Sr cut, with ∼1.5 % eluting in the Ca cut. As resin volume is decreased significant amounts of Sr are eluted in the Ca cut, with 66 % of load Sr eluting in the Ca cut of Column 6 (Figure 2.6). This Sr breakthrough is perhaps expected — low resin masses result in a very small resin height, which is reduced even further as eluent is added to the column and suspends the top millimetre or two of resin. Furthermore the volumes of eluent used were not adjusted for resin volume and may have been inappropriate for the low resin volumes. Regardless of exact mechanism, this Sr breakthrough does not represent the resin reaching its theoretical Sr-bearing capacity, but rather physical disruption of the resin column and/or excessive eluent volumes for the lowest resin heights.

Interestingly, Sr breakthrough provides an excellent test of the Sr correction (Section 2.2.5.2), with Sr breakthrough in Column 6 (66 %) resulted in a maximum Ca isotope aliquot Sr/Ca ratio of ∼35 mmol mol$^{-1}$. Despite this extensive Sr contamination, Sr-corrected data for the Ca cuts fall on the mass-dependent fractionation line on a Ca triple isotope plot (Figure 2.7). This suggests that the Sr correction is accurate for these Sr/Ca ratios (Section 2.2.5.2).

#### 2.3.2.4 SrSpec Column Calibration - Summary

Re-calibration of the SrSpec column procedure has determined that Ca yields >90 % are achievable, however ∼5 % of load Ca is retained on the resin. Ca isotope data suggest

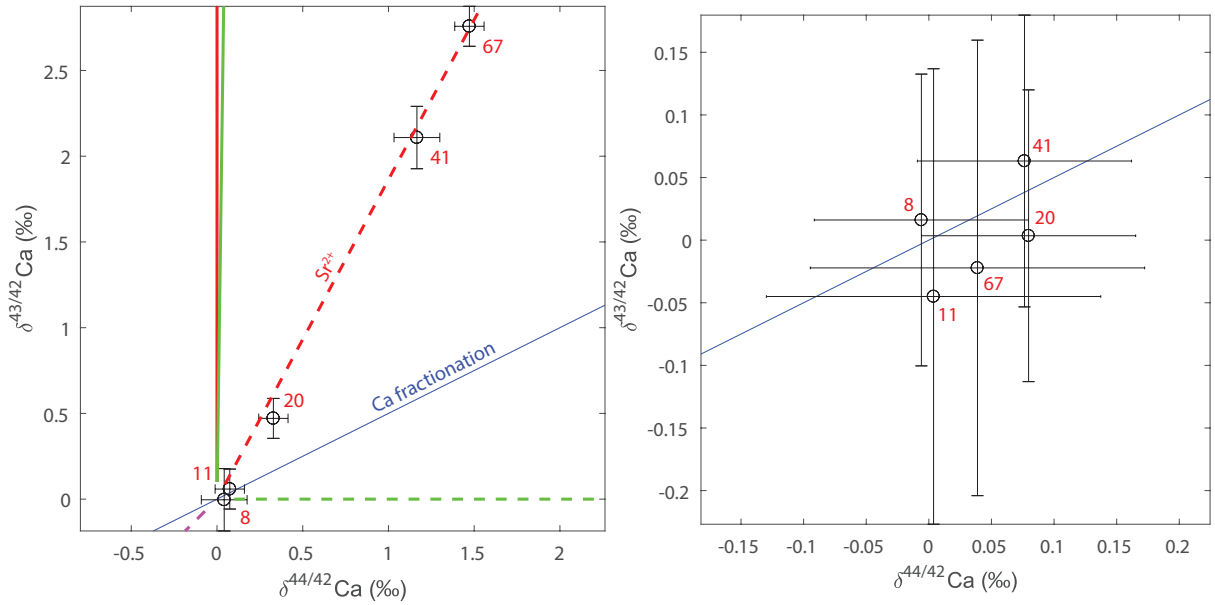



Figure 2.7: Ca triple isotope plots showing SrSpec yield test results (Table 2.6). Left: Data processed without the Sr correction, showing clear Sr interference in Ca cuts from column with low resin masses. Right: Data processed with the Sr correction. Data points are labelled with their respective Sr breakthrough percentages (Table 2.7).

that the column may preferentially retain isotopically light Ca, leading to fractionation of the eluted Ca to heavier ratios, although this fractionation is within analytical uncertainty under the conditions tested here. These tests were performed using a Ca load of approximately 100 µg; Ca fractionation is likely more noticeable with smaller Ca loads and care should be taken when processing any smaller samples through SrSpec columns. Equally, larger Ca loads may be desirable when sufficient sample material is available. Varying resin volume showed that a SrSpec column height of at least ∼1.2 cm is necessary to achieve good separation of Ca and Sr using the eluent volumes tested here.

The finalised SrSpec Ca-Sr separation method determined here is given in Table 2.8. This method uses the column cleaning and pre-conditioning steps of Chu et al. (2006). Eluent volumes are determined from the elution curves given in Figure 2.4. This method has been tested for Ca loads of approximately 100 µg, however the method should also be suitable for far larger samples if desired. Significant Ca isotope fractionation is likely to occur using smaller sample loads.

Column cuts were collected in Teflon beakers, dried down and re-dissolved in 2 % $HNO_3$ in preparation for analysis. Before use, all SrSpec resin was cleaned for fine particulates by repeatedly suspending the resin in MQ water, allowing the resin to settle and removing the

Table 2.8: SrSpec Ca-Sr separation procedure developed for 100 µg Ca loads. See Section 2.3.2 for more details.

| Step | Reagent | Volume (µL) |
|---|---|---|
| Clean | 6 M HCl | 1000 |
| Clean | $H_2O$ | 1000 |
| Clean | 0.01 M $HNO_3$ | 2000 |
| Condition | 3 M $HNO_3$ | 1500 |
| Condition | 3 M $HNO_3$ | 900 |
| Load Ca | Sample (in 3 M $HNO_3$) | 250 |
| Collect Ca | 3 M $HNO_3$ | 250 |
| Collect Ca | 3 M $HNO_3$ | 1500 |
| Collect Sr | 0.01 M $HNO_3$ | 1000 |
| Collect Sr | 0.01 M $HNO_3$ | 1000 |
| Collect Sr | 0.01 M $HNO_3$ | 1000 |

supernatant. Due to Ca & Sr retention, resin was discarded after a single use. This method is suitable for Ca-Sr separation on relatively clean carbonate samples. For samples rich in other elements (e.g. Mg), prior processing through AG50W-X12 resin is recommended to isolate the Ca and Sr content (Table 2.3).

### 2.3.3 prepFAST-MC

In Section 2.3.2 I described re-calibration work validating a two-step column procedure for separating Ca and Sr from speleothem samples. Column chemistry methods are robust and well-established, but are also time-intensive, expensive and require skilled work in a clean-lab environment.

During the course of this D.Phil. project, the Department of Earth Sciences obtained a Element Scientific prepFAST-MC — an automated system for chromatographic separation of various elements. This section describes testing and calibration work done to set up Ca isotope sample preparation on the prepFAST-MC. The resulting method provides a quick, automated method for Ca separation chemistry as an alternative to column chemistry.

The prepFAST-MC is an automated chromatography system produced by Element Scientific. The system is designed to automate elemental separation chemistry and cut collection according to programmable routines. The method used here is based on the

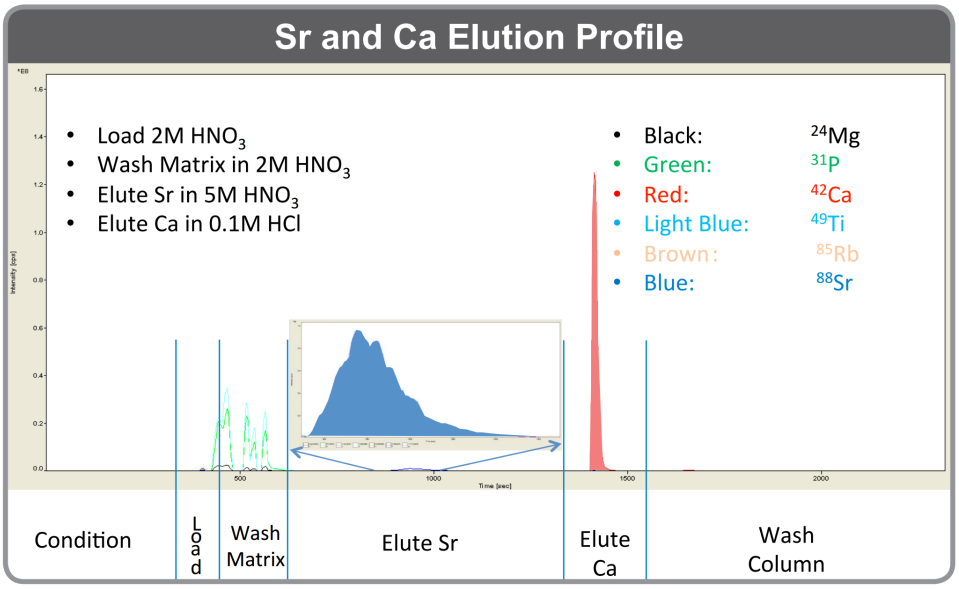



Figure 2.8: The prepFAST-MC Ca-Sr separation method (source: product brochure).

Ca-Sr separation method included in the prepFAST-MC software.

This section is not intended as a fully description of the device. Further information about the design and operation of the prepFAST-MC can be obtained from the manufacturer's website (`http://www.icpms.com/products/prepfast-MC.php`), and in Romaniello et al. (2015).

#### 2.3.3.1 Testing prepFAST-MC

The manufacturer-provided Ca-Sr separation method, described in Figure 2.8, loads a sample onto the Ca-Sr column, and elutes matrix elements, in 2 M $HNO_3$. Ca and Sr are retained on the column. Sr is then eluted in 5 M $HNO_3$ and finally Ca in 0.1 M HCl. This method has the potential to automate separation of Mg (and other matrix elements), Ca and Sr.

To validate this method I ran an experiment to test the system for blanks, Ca yields and isotope fractionation. Samples of NIST SRM 915b of 40 µg, 70 µg, 100 µg and 300 µg Ca were prepared in 2 M $HNO_3$ to act as mock samples and were processed through prepFAST-MC in sequence, interspersed with acid blank to test for Ca memory effects. The sample sequence, details and results are given in Table 2.9. All mock samples were processed twice; once using the standard elution volume of 5 mL, and once with an in-

creased elution volume of 10 mL. This was done to test for Ca tailing beyond the standard 5 mL elution.

Samples eluted in 5 mL 0.1 M HCl show Ca yields >97 %, while those eluted in 10 mL 0.1 M HCl show Ca yields >99 %. $\delta^{44/42}$Ca values of all eluted samples are indistinguishable from the value of unprocessed NIST SRM 915b ($\delta^{42/42}$Ca = 0.36 ‰; Section 2.2.3). There is no evidence for Ca isotope fractionation during prepFAST-MC processing. No processed samples show any significant Sr contamination, as determined by observation of the 43.5 m/z beam during Ca isotope analysis. This confirms the efficacy of the prepFAST-MC Ca-Sr separation at both 5 mL and 10 mL elution volumes.

Acid blanks processed through prepFAST elute a small amount of Ca, proportional to the mass of Ca in the previous sample. This demonstrates a small memory effect where 0.4–0.5 % of sample Ca is retained somewhere in the flow loop and eluted with the following sample. This may cause some cross-contamination of samples without intervening blanks, however the amount is small enough that the effect on Ca isotopes will be negligible for samples of similar concentrations, and can safely be ignored. The magnitude of the memory effect does not appear to change between 5 mL and 10 mL Ca elution volumes.

Table 2.9: Sample details and results for the prepFAST-MC Ca isotope test.

| Name | Description | Ca Load (µg) | Elution volume (mL) | Ca Eluted (µg) | $\delta^{44/42}$Ca (‰ 915a) | C.I. (±) | $\delta^{43/42}$Ca (‰ 915a) | C.I. (±) | n |
|---|---|---|---|---|---|---|---|---|---|
| B1 | Blank | 0 | 5 | 0.05 | - | - | - | - | - |
| B2 | Blank | 0 | 5 | 0.05 | - | - | - | - | - |
| B3 | Blank | 0 | 5 | 0.05 | - | - | - | - | - |
| A1 | SRM 915b | 40 | 5 | 40.3 | 0.32 | 0.07 | 0.13 | 0.09 | 5 |
| M1 | Blank | 0 | 5 | 0.24 | - | - | - | - | - |
| A2 | SRM 915b | 70 | 5 | 69.7 | 0.35 | 0.07 | 0.17 | 0.09 | 5 |
| M2 | Blank | 0 | 5 | 0.38 | - | - | - | - | - |
| A3 | SRM 915b | 100 | 5 | 99.1 | 0.35 | 0.07 | 0.15 | 0.09 | 5 |
| M3 | Blank | 0 | 5 | 0.49 | - | - | - | - | - |
| A4 | SRM 915b | 300 | 5 | 293.4 | 0.37 | 0.07 | 0.19 | 0.09 | 5 |
| M4 | Blank | 0 | 5 | 1.31 | - | - | - | - | - |
| A5 | SRM 915b | 40 | 10 | 40.9 | 0.37 | 0.09 | 0.24 | 0.12 | 4 |
| M5 | Blank | 0 | 10 | 0.28 | - | - | - | - | - |
| A6 | SRM 915b | 70 | 10 | 69.3 | 0.35 | 0.07 | 0.23 | 0.09 | 5 |
| M6 | Blank | 0 | 10 | 0.36 | - | - | - | - | - |
| A7 | SRM 915b | 100 | 10 | 99.6 | 0.37 | 0.07 | 0.19 | 0.09 | 5 |
| M7 | Blank | 0 | 10 | 0.48 | - | - | - | - | - |
| A8 | SRM 915b | 300 | 10 | 299.5 | 0.39 | 0.07 | 0.24 | 0.09 | 5 |
| M8 | Blank | 0 | 10 | 1.11 | - | - | - | - | - |
| M9 | Blank | 0 | 10 | 0.18 | - | - | - | - | - |

#### 2.3.3.2 Finalised prepFAST routine

Section 2.3.3.1 demonstrates that sample processing through prepFAST-MC can achieve excellent Ca yields and Ca-Sr separation with no measurable isotope fractionation. This is true for all sample masses tested (40–300 µg) and using both 5 mL and 10 mL Ca elution volumes. While 10 mL elution showed a marginally higher yield, isotope values were indistinguishable and as such elution of Ca in 5 mL 0.1 M HCl is sufficient.

Based on this experiment and manufacturer data concerning elemental separations (Figure 2.8) it is valid to apply the manufacturer-defined Ca-Sr separation routine for Ca isotope sample preparation for samples in the range 40–300 µg Ca. With further testing this method may be suitable for Mg-Sr-Ca separations, potentially facilitating coupled isotope measurements from all three systems.

#### 2.3.3.3 Comparison with Column Method

Both column chemistry (Section 2.3.2) and prepFAST-MC (Section 2.3.3) have been demonstrated as valid methods to separate Ca from various matrix elements, in particular Sr, and either method may be used to process samples with trace element contents that exceed the allowable limits determined in Section 2.2.5. Of the two methods, prepFAST-MC is preferable for the following reasons:

**Yields** prepFAST-MC achieved higher Ca yields than SrSpec column chemistry (>97 % versus >92 %. A higher yield reduces the possibility for isotope fractionation due to Ca loss during elemental separation.

**Isotope Fractionation** SrSpec columns likely retain some isotopically light Ca, causing fractionation of the eluted Ca. prepFAST-MC causes no noticeable isotope fractionation for the sample masses tested here.

**Sample Size** prepFAST-MC shows excellent results for tests using 40–300 µg Ca, validating the method for a wide range of sample sizes. In contrast, data from Philip Pogge Von Strandmann suggest that the isotope fractionation associated with Sr-Spec resin becomes more significant as sample sizes are reduced, thus limiting SrSpec

processing to relatively large samples.

**Ease of Use** The two-step column procedure used to isolate Ca is laborious, taking a minimum of three days to process a batch of ten samples. In contrast, prepFAST-MC may be automated to processed a large number of samples overnight, saving a great deal of time. Furthermore, the automated method reduces the potential for human error.

## 2.4 Trace Element Analysis

Trace element analyses were usually performed on the Element 2 single-collector ICP-MS at the Department of Earth Sciences, University of Oxford, with the technical assistance of Phil Holdship. Samples were analysed using the 'ratio' method, broadly following Rosenthal et al. (1999). Uncertainty is quoted as twice the root squared deviation (2 RSD) on repeat standard measurements.

Data processing was performed offline in MATLAB using a set of scripts produced as part of this project (nicknamed 'tedr'). These scripts automate evaluation of instrument stability and accuracy, perform calibration calculations via interpolation of standard measurements and calculate both absolute and Ca-normalised concentrations.

# Chapter 3

# Calcium isotopes in caves as a proxy for aridity: Modern calibration and application to the 8.2kyr event

The work presented in this Chapter has previously been published in Owen, Day, Hu, Liu, Pointing, Blättler, and Henderson (2016). The text has been modified to more clearly reflect the contributions of individual authors. Unless otherwise stated, all work was undertaken by the author of this thesis. Contributions from collaborators are listed below and noted where appropriate in the main text.

- Chaoyong Hu provided all speleothem, glass plate calcite and dripwater material.
- Yuhui Liu micro-milled the calcite powders used to construct the 8.2kyr event record.
- Sample preparation and Ca isotope analyses of glass plate calcite and dripwater samples were performed by Matt Pointing and Clara Blättler.

## 3.1 Abstract

This chapter presents the first study of Ca isotope cycling in a natural cave system, with measurements of bedrock, dripwater and recently formed carbonate, coupled to a first stalagmite time-series spanning the 8.2kyr event. Dripwaters at Heshang Cave (Central

China; 30°27′ N, 110°25′ E) are isotopically heavy relative to the dolomite bedrock, the result of PCP occurring earlier in the drip flow path. A simple Rayleigh fractionation model quantifies the extent of PCP in the modern environment at 36 % Ca removal. The observed in-situ calcium isotope fractionation factor between dripwater and carbonate is $\Delta^{44/42}\text{Ca} = -0.63 \pm 0.03$ ‰ and does not vary during the annual cycle.

Measurements of speleothem carbonate spanning the 8.2kyr event show the response of Ca isotopes to changing climate. $\delta^{44/42}$Ca increases by 0.35 ‰ at the onset of the event, coeval with changes in $\delta^{18}$O and Mg/Ca, and remains high for 80 years. This change is explained by decreased rainfall leading to increased PCP; an interpretation supported by established PCP proxies (Mg/Ca, Sr/Ca and Ba/Ca). Ca isotopes indicate that PCP increased to 60 % Ca removal during the event, which, from application of a simple box model, suggests mean annual rainfall decreased by approximately a third in Central China during the 8.2kyr event. The response of Ca isotopes across this event demonstrates their potential for the assessment of past conditions, including past dripwater flow rates and rainfall.

## 3.2 Introduction

Speleothem carbonates are continental palaeoclimate archives. Numerous speleothem-based climate proxies have been developed and applied including $\delta^{18}$O and $\delta^{13}$C analyses (e.g. Hendy, 1971; Wang et al., 2001), trace metal contents (e.g. Ayalon et al., 1999; Gascoyne, 1983), fluid inclusion chemistry (e.g. Schwarcz et al., 1976) and clumped isotope measurements (e.g. Affek et al., 2008). Of these, carbon and oxygen isotope analyses are the most widely used. $\delta^{18}$O records have shed light on past changes in precipitation patterns and atmospheric moisture transport (e.g. Bar-Matthews et al., 1999; Dykoski et al., 2005; Wang et al., 2001). These records are highly informative, but the controls on speleothem $\delta^{18}$O are complex and often non-local, complicating their interpretation (Dykoski et al., 2005; Pausata et al., 2011).

To address these uncertainties, speleothem $\delta^{18}$O records are increasingly being com-

plemented with minor and trace element data (Hellstrom and McCulloch, 2000; Moreno et al., 2010; Liu et al., 2013). The concentrations of certain trace elements (for example Mg, Sr and Ba) in dripwaters and calcite precipitates have been shown to vary in a range of settings, on seasonal and inter-annual timescales, in response to rainfall change (Fairchild et al., 2001; Treble et al., 2005; Tremaine and Froelich, 2013). This is due to precipitation of secondary carbonate minerals in the overlying aquifer and/or on the cave roof, a process termed prior calcite precipitation (PCP) (Fairchild et al., 2000). Preferential removal of Ca into calcite during PCP enriches dripwaters in Mg, Ba and Sr relative to Ca. The extent of PCP is related to local effective rainfall; during periods of aridity, decreased pressure head leads to increased water residence times and may increase the proportion of air-filled porosity in the aquifer — both factors that encourage PCP (Tooth and Fairchild, 2003; Cruz et al., 2007). The chemical signature of PCP preserved in speleothems has the potential to reconstruct past aridity and disentangle local from regional controls on $\delta^{18}$O records. However, while highly informative, trace element based PCP reconstructions have so far been limited to qualitative interpretation about rainfall change.

An additional geochemical proxy with potential to assess past PCP and local rainfall is the isotopic composition of calcium in speleothems. While speleothem carbon and oxygen isotope signals have been studied extensively, no systematic study has yet investigated isotope fractionation of Ca, the third major element of calcite, in the cave environment or speleothem archive. Speleothem Ca is primarily sourced from the host carbonate rock, dissolved by $CO_2$-charged waters in the soil and upper epikarst. PCP may occur as waters percolate through the unsaturated zone. Ca isotopes are fractionated during incorporation into calcite, with light isotopes preferentially incorporated into the solid phase (Lemarchand et al., 2004; Marriott, 2004; Gussone et al., 2005; Tang et al., 2008; Reynard et al., 2011). Therefore PCP is expected to isotopically enrich dripwater relative to its initial composition. Ca isotopes in speleothems may record this signal.

The 8.2kyr event is the largest natural climate perturbation in the Holocene. First identified in Greenland ice cores (Dansgaard, 1987), the event is known globally (Spooner et al., 2002; Ljung et al., 2008; Pross et al., 2009; Dixit et al., 2014). In China the event is

recorded as a period of increased aridity, interpreted as a weakening of the monsoon (Hu et al., 2008b; Cheng et al., 2009; Morrill et al., 2011; Liu et al., 2013). This perturbation to the hydrological cycle provides an excellent test case to assess the speleothem Ca isotope response to climate change.

## 3.3 Methods

### 3.3.1 Study Site and Sample Selection

The study site is Heshang Cave, China. Background for this cave site is given in (Section 1.4). Speleothem samples were taken from stalagmite HS4 — an extensively studied Holocene stalagmite previously described in the literature (Hu et al., 2005; Johnson et al., 2006; Hu et al., 2008b; Liu et al., 2013; Li et al., 2014). Liu et al. (2013) established a chronology for the 8.2kyr event in stalagmite HS4 and characterised the event based on an ultra-high-resolution $\delta^{18}O$ and Mg/Ca record. They identify the 8.2kyr event as a period of enhanced PCP at the site, suggesting a decrease in rainfall. Based on their record I selected 44 micromilled speleothem samples spanning the 8.2kyr event from the residual powders milled during the study of Liu et al. (2013). Each sample averages 4-5 years of stalagmite growth.

In addition, nineteen dripwater samples (acidified immediately after sampling) and twenty farmed glass-plate calcite samples were taken from the HS4 drip-site at monthly intervals over the course of two years (2005–2007) to understand seasonal controls on dripwater and speleothem Ca isotopes. A block of dolomite bedrock was powdered and subsampled to characterise the source of calcium.

All dripwater and glass-plate calcite samples were provided by Chaoyong Hu and speleothem samples were milled by Yuhui Liu.

### 3.3.2 Sample Preparation

Carbonate samples (200–300 µg $CaCO_3$ each) were dissolved in 2 % $HNO_3$. Samples to be processed through column chemistry were dried and re-dissolved in 2 M HCl. Dripwater and glass plate samples were processed through the two-step column procedure (Section 2.3.2) to isolate their Ca content. Briefly, BioRad AG50W-X12 is used to separate Ca and Sr from other matrix elements, notably Mg. This is followed by Eichrom SrSpec resin to separate Ca from Sr. Subsequent testing (described in Section 2.2.5) showed trace element contents were sufficiently low that only the first step of chemistry was necessary for the dolomite bedrock, and no processing was necessary for speleothem samples.

### 3.3.3 Ca Isotope Analysis

Calcium isotope ratios were analysed using a Nu Instruments MC-ICP-MS with Aridus desolvating nebuliser at the University of Oxford. The method is fully described in Section 2.2.6.

Samples were analysed for $^{42}$Ca, $^{43}$Ca and $^{44}$Ca allowing calculation of $\delta^{44/42}$Ca and $\delta^{43/42}$Ca. NIST SRM 915a was used as the bracketing standard, and samples were concentration matched to $10 \pm 1$ ppm Ca. Each sample was analysed a minimum of 5 times. Data have been corrected for Sr interferences (following Section 2.2.5.2), the effects of which are shown on the Ca isotope triple plot (Figure 3.1). All subsequent data in this chapter are reported after Sr correction.

For speleothem and bedrock samples, secondary standard NIST SRM 915b and an internal secondary standard (HPS Ca — a High Purity Standards Ca standard) were used to determine accuracy and external precision after Sr correction. Measured values for SRM 915b were $\delta^{44/42}\text{Ca} = 0.35 \pm 0.11$ ‰ and $\delta^{43/42}\text{Ca} = 0.19 \pm 0.13$ ‰ (2s, n = 40), and for HPS Ca were $\delta^{44/42}\text{Ca} = 0.27 \pm 0.07$ ‰ and $\delta^{43/42}\text{Ca} = 0.14 \pm 0.11$ ‰ (2s, n = 39). Analyses of dripwater and glass plate calcite samples were performed by Matt Pointing and Clara Blättler using a slightly different method. They used HPS Ca to determine external precision giving $\delta^{44/42}\text{Ca} = 0.34 \pm 0.14$ ‰ and $\delta^{43/42}\text{Ca} = 0.23 \pm 0.18$ ‰ (2s,

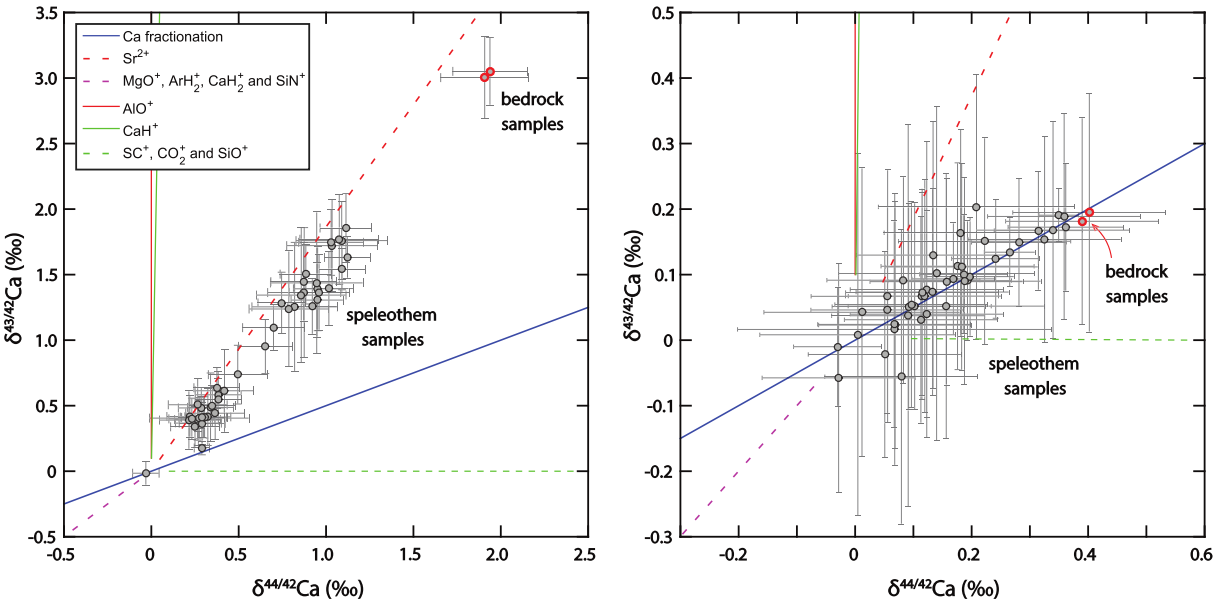



Figure 3.1: Heshang Ca isotope data on a triple-isotope plot showing the effects of mass-dependent Ca isotope fractionation and approximate offset vectors due to potential interferences. Left: Data processed without the Sr correction. Right: Data processed with the Sr correction. Unprocessed data are offset parallel to the $Sr^{2+}$ interference vector. Corrected data fall along the mass-dependent fractionation line without any obvious isobaric. Note the differences in scale. Sr correction details are given in Section 3.3.3).

n = 184). These accuracy and precision measurements are indistinguishable from the long-term values reported in Section 2.2.3.

Figure 3.1 shows data processed with and without the Sr correction on a triple-isotope plot. Without the Sr correction samples are offset parallel to the $Sr^{2+}$ interference vector; corrected data fall close to the mass-dependent fractionation line. Without the correction, reproducibility on Sr-bearing samples is poor due to variation in the $Sr^{2+}$/Sr ionisation ratio. Sr correction reduces the uncertainty (external reproducibility) on NIST SRM 915b $\delta^{44/42}$Ca by 77 %. In contrast, Sr correction increases the uncertainty (external reproducibility) on HPS Ca $\delta^{44/42}$Ca by 16 %.

Unless otherwise stated, uncertainty on Ca isotope data is quoted as the t-distribution-derived 95 % confidence interval on the mean of repeat measurements calculated using either the standard deviation on all repeat measurements in the run or the standard deviation on all secondary standard analyses, whichever is greater. SRM 915b was used as the secondary standard for bedrock and HS4 samples. HPS Ca was used for dripwater and glass plate calcite samples.

### 3.3.4 Trace Element Analysis

Dissolved HS4 samples were sub-sampled for trace element analysis using a Thermo Scientific Element-2 ICP-MS at the University of Oxford. Minor/trace element-to-Ca ratios were determined for Mg, Sr, Ba, Pb, Th, U, Na, P, S, Mn, Fe, Co, Zn and Rb following the method described in Section 2.4. Analyses were performed at 1 ppm Ca, concentration matched to within 10 %.

Reported here are Mg/Ca, Sr/Ca and Ba/Ca ratios with external precisions of 4.3 % on Mg/Ca, 3.8 % on Sr/Ca and 6.2 % on Ba/Ca. Uncertainty was quantified using a secondary standard interspersed repeatedly during sample analysis and is reported as 2x the relative standard deviation (RSD) on all analyses.

## 3.4 Results

Dripwater and glass plate calcite $\delta^{44/42}$Ca values did not vary significantly about their mean values over the two-year monitoring period (t-test, 95 % confidence level) despite large seasonal variations in cave air temperature and drip rate (Table 3.1; Figure 3.2). Mean dripwater $\delta^{44/42}\text{Ca} = 0.68 \pm 0.03$ ‰ (95 % CI, n = 17), and mean glass plate calcite $\delta^{44/42}\text{Ca} = 0.05 \pm 0.02$ ‰ (95 % CI, n = 19). Coeval measurements of dripwater and glass plate calcite allow calculation of the Ca isotope fractionation occurring at the HS4 drip-site: $\Delta^{44/42}\text{Ca}_{\text{calcite-water}} = -0.63 \pm 0.03$ ‰ (95 % CI, n = 15). This can be expressed as an effective fractionation factor of $\alpha^{44/42} = 0.99937 \pm 0.00003$. $\Delta^{44/42}\text{Ca}_{\text{calcite-water}}$ did not vary significantly about its mean value over the monitoring period (t-test, 95% confidence level). Two aliquots of Heshang Cave dolomite bedrock, analysed to characterise the source of dripwater Ca, gave indistinguishable $\delta^{44/42}$Ca values of $0.40 \pm 0.07$ ‰ (n = 5) and $0.39 \pm 0.07$ ‰ (n = 5).

Ca isotope, Mg/Ca, Sr/Ca and Ba/Ca records from stalagmite HS4 covering the 8.2kyr event (Table 3.2) are shown in Figure 3.3. The vertical grey shaded bars mark the 8.2kyr event as identified from the high resolution $\delta^{18}$O record of Liu et al. (2013). Prior to the 8.2kyr event, $\delta^{44/42}$Ca varies between approximately 0 ‰ and 0.15 ‰. A positive

Table 3.1: Modern Ca isotope data from the HS4 drip site from 2005 to 2007. All sample processing and analysis was performed by Matt Pointing and Clara Blättler.

| | Dripwater | | Glass Plate Calcite | | Fractionation | |
|---|---|---|---|---|---|---|
| Date | $\delta^{44/42}$Ca (‰ 915a) | 95% CI (±) | $\delta^{44/42}$Ca (‰ 915a) | 95% CI (±) | $\Delta^{44/42}$Ca (‰) | $\alpha$ |
| 29/01/2005 | 0.76 | 0.17 | 0.09 | 0.17 | -0.68 | 0.99933 |
| 28/03/2005 | 0.69 | 0.12 | -0.05 | 0.17 | -0.73 | 0.99927 |
| 16/05/2005 | - | 2.39 | 0.04 | 0.17 | - | - |
| 23/06/2005 | 0.60 | 0.21 | -0.04 | 0.17 | -0.64 | 0.99936 |
| 26/06/2005 | - | - | -0.10 | 0.17 | - | - |
| 24/07/2005 | 0.63 | 0.16 | 0.04 | 0.17 | -0.59 | 0.99941 |
| 26/08/2005 | 0.62 | 0.11 | 0.00 | 0.17 | -0.62 | 0.99938 |
| 05/10/2005 | 0.64 | 0.17 | -0.07 | 0.17 | -0.70 | 0.99930 |
| 14/11/2005 | 0.57 | 0.17 | -0.05 | 0.17 | -0.62 | 0.99939 |
| 16/12/2005 | - | - | -0.02 | 0.17 | - | - |
| 04/01/2006 | - | - | 0.10 | 0.20 | - | - |
| 03/02/2006 | 0.68 | 0.17 | 0.05 | 0.11 | -0.63 | 0.99937 |
| 14/03/2006 | 0.63 | 0.11 | 0.10 | 0.11 | -0.54 | 0.99946 |
| 27/04/2006 | 0.78 | 0.11 | 0.21 | 0.36 | -0.57 | 0.99944 |
| 12/06/2006 | 0.71 | 0.11 | - | - | - | - |
| 21/07/2006 | 0.78 | 0.17 | 0.11 | 0.11 | -0.67 | 0.99933 |
| 29/08/2006 | 0.71 | 0.11 | 0.12 | 0.18 | -0.59 | 0.99941 |
| 11/10/2006 | 0.70 | 0.17 | 0.09 | 0.17 | -0.61 | 0.99939 |
| 23/11/2006 | 0.68 | 0.11 | 0.06 | 0.11 | -0.61 | 0.99939 |
| 02/01/2007 | 0.68 | 0.11 | 0.07 | 0.11 | -0.62 | 0.99938 |
| 12/02/2007 | 0.64 | 0.11 | - | - | - | - |

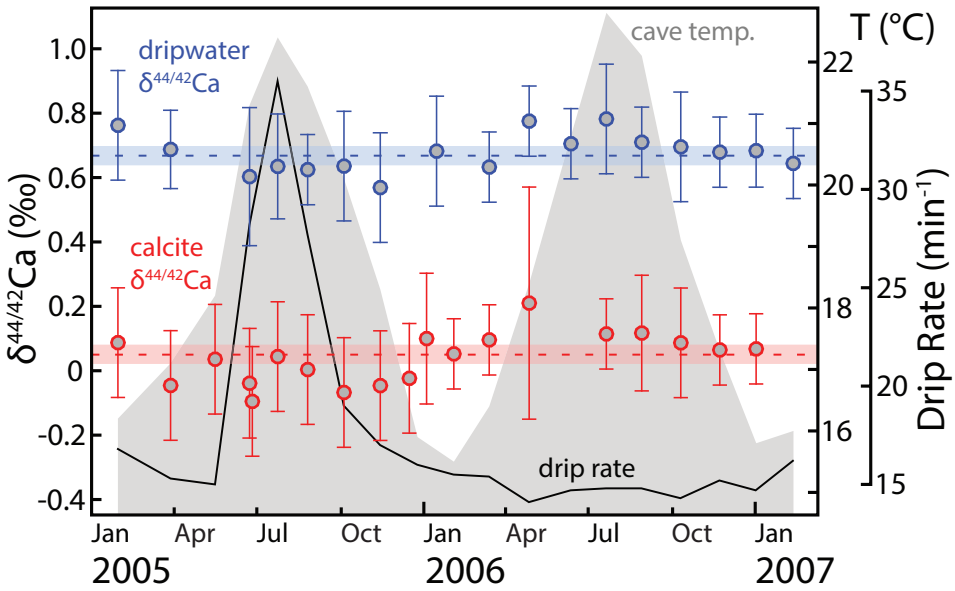



Figure 3.2: HS4 drip-site calcium isotope data spanning two years. Dripwater data are shown in blue and glass plate calcite data in red. Horizontal coloured bars show respective mean values and their 95 % confidence intervals. Ca isotope data are given in Table 3.1. Cave temperature and drip rate data from Hu et al. (2008a) are shown in grey and black, respectively.

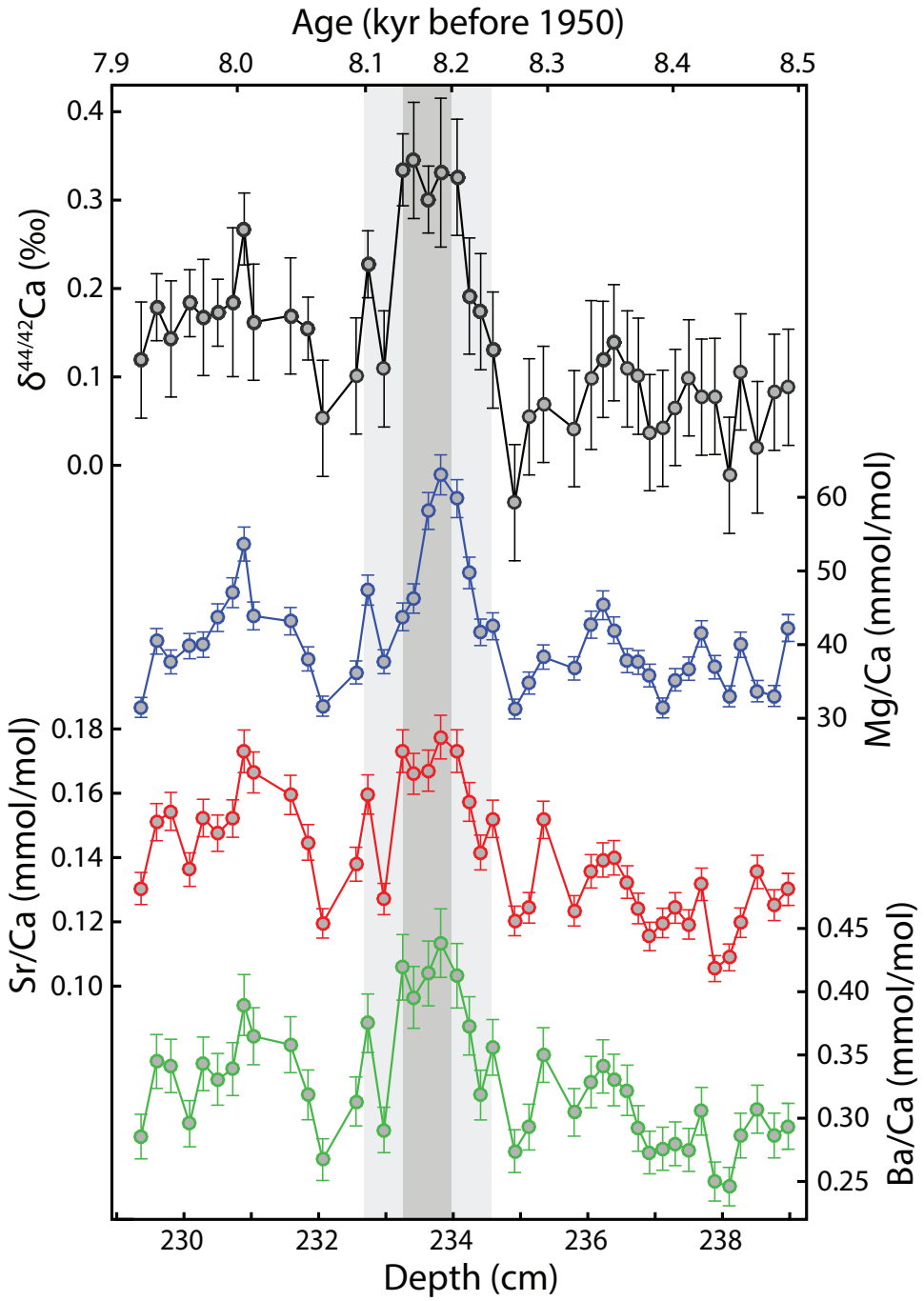



Figure 3.3: Ca isotope, Mg/Ca, Sr/Ca and Ba/Ca records across the 8.2kyr event. The grey shaded area shows the 8.2kyr event as identified by Liu et al. (2013) with a total duration of 150 years and a central extreme portion lasting 70 years. Depths given here are correct, and differ from those in Liu et al. (2013, Figure 3), which were incorrectly plotted. The age scale remains correct and unchanged.

excursion of 0.35 ‰ coincides with the 8.2kyr event. $\delta^{44/42}$Ca during the central part of the 8.2kyr event is stable at ~0.35 ‰ before returning to lower values after the event.

The 8.2kyr event in HS4 is further characterised by positive excursions in Mg/Ca, Sr/Ca and Ba/Ca. Ba/Ca and Sr/Ca correlate strongly with $\delta^{44/42}$Ca ($R^2$ = 0.77 and 0.75, respectively), suggesting shared controls on these proxies. Mg/Ca increases at the

onset of the event, coincident with increases in $\delta^{44/42}$Ca, Ba/Ca, and Sr/Ca, but records a significantly shorter excursion and returns to background values during the central portion of the 8.2kyr event.

| Sample | Depth (cm) | Age[1] (years before 1950) | $\delta^{44/42}$Ca (‰ 915a) | 95% CI ± | Mg/Ca (mmol mol$^{-1}$) | Sr/Ca (mmol mol$^{-1}$) | Ba/Ca (mmol mol$^{-1}$) |
|---|---|---|---|---|---|---|---|
| HS4-29 | 229.36 | 7923 | 0.13 | 0.07 | 31 | 0.13 | 0.29 |
| HS4-31 | 229.59 | 7936 | 0.19 | 0.04 | 40 | 0.15 | 0.35 |
| HS4-33 | 229.81 | 7947 | 0.16 | 0.07 | 38 | 0.15 | 0.34 |
| HS4-35 | 230.08 | 7962 | 0.20 | 0.04 | 40 | 0.14 | 0.30 |
| HS4-37 | 230.29 | 7973 | 0.18 | 0.07 | 40 | 0.15 | 0.34 |
| HS4-39 | 230.50 | 7984 | 0.19 | 0.04 | 44 | 0.15 | 0.33 |
| HS4-41 | 230.72 | 7996 | 0.20 | 0.08 | 47 | 0.15 | 0.34 |
| HS4-43 | 230.89 | 8005 | 0.28 | 0.04 | 54 | 0.17 | 0.39 |
| HS4-44 | 231.03 | 8013 | 0.18 | 0.07 | 44 | 0.17 | 0.37 |
| HS4-49 | 231.58 | 8042 | 0.18 | 0.07 | 43 | 0.16 | 0.36 |
| HS4-51 | 231.84 | 8056 | 0.17 | 0.04 | 38 | 0.14 | 0.32 |
| HS4-53 | 232.06 | 8068 | 0.07 | 0.07 | 32 | 0.12 | 0.27 |
| HS4-57 | 232.56 | 8087 | 0.12 | 0.07 | 36 | 0.14 | 0.31 |
| HS4-59 | 232.73 | 8103 | 0.24 | 0.04 | 47 | 0.16 | 0.38 |
| HS4-61 | 232.97 | 8117 | 0.12 | 0.07 | 38 | 0.13 | 0.29 |
| HS4-63 | 233.25 | 8129 | 0.35 | 0.04 | 44 | 0.17 | 0.42 |
| HS4-65 | 233.41 | 8147 | 0.36 | 0.07 | 46 | 0.17 | 0.40 |
| HS4-67 | 233.63 | 8167 | 0.31 | 0.04 | 58 | 0.17 | 0.41 |
| HS4-69 | 233.81 | 8183 | 0.34 | 0.08 | 63 | 0.18 | 0.44 |
| HS4-71 | 234.06 | 8208 | 0.34 | 0.07 | 60 | 0.17 | 0.41 |
| HS4-73 | 234.24 | 8221 | 0.21 | 0.07 | 50 | 0.16 | 0.37 |
| HS4-75 | 234.41 | 8234 | 0.19 | 0.07 | 42 | 0.14 | 0.32 |
| HS4-77 | 234.59 | 8249 | 0.14 | 0.07 | 42 | 0.15 | 0.36 |
| HS4-81 | 234.91 | 8273 | -0.03 | 0.07 | 31 | 0.12 | 0.27 |
| HS4-83 | 235.13 | 8285 | 0.07 | 0.07 | 35 | 0.12 | 0.29 |
| HS4-85 | 235.34 | 8296 | 0.08 | 0.07 | 38 | 0.15 | 0.35 |
| HS4-87 | 235.55 | 8308 | - | - | 28 | 0.14 | 0.34 |
| HS4-89 | 235.80 | 8321 | 0.06 | 0.07 | 37 | 0.12 | 0.30 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| HS4-91 | 236.05 | 8335 | 0.12 | 0.08 | 43 | 0.14 | 0.33 |
| HS4-93 | 236.22 | 8344 | 0.13 | 0.07 | 45 | 0.14 | 0.34 |
| HS4-95 | 236.39 | 8353 | 0.15 | 0.07 | 42 | 0.14 | 0.33 |
| HS4-97 | 236.59 | 8363 | 0.12 | 0.07 | 38 | 0.13 | 0.32 |
| HS4-99 | 236.75 | 8372 | 0.11 | 0.07 | 38 | 0.12 | 0.29 |
| HS4-101 | 236.92 | 8381 | 0.05 | 0.07 | 36 | 0.12 | 0.27 |
| HS4-103 | 237.11 | 8392 | 0.06 | 0.07 | 31 | 0.12 | 0.28 |
| HS4-105 | 237.30 | 8402 | 0.08 | 0.07 | 35 | 0.12 | 0.28 |
| HS4-107 | 237.50 | 8413 | 0.11 | 0.07 | 37 | 0.12 | 0.28 |
| HS4-109 | 237.69 | 8423 | 0.09 | 0.07 | 41 | 0.13 | 0.31 |
| HS4-111 | 237.89 | 8433 | 0.09 | 0.07 | 37 | 0.11 | 0.25 |
| HS4-113 | 238.10 | 8445 | 0.00 | 0.07 | 33 | 0.11 | 0.25 |
| HS4-115 | 238.27 | 8454 | 0.12 | 0.07 | 40 | 0.12 | 0.29 |
| HS4-117 | 238.52 | 8467 | 0.03 | 0.07 | 34 | 0.14 | 0.31 |
| HS4-119 | 238.77 | 8481 | 0.10 | 0.07 | 33 | 0.13 | 0.29 |
| HS4-121 | 238.98 | 8492 | 0.10 | 0.07 | 42 | 0.13 | 0.29 |

Table 3.2: Ca isotope and trace element data from stalagmite HS4. [1] — Chronology from Liu et al. (2013)

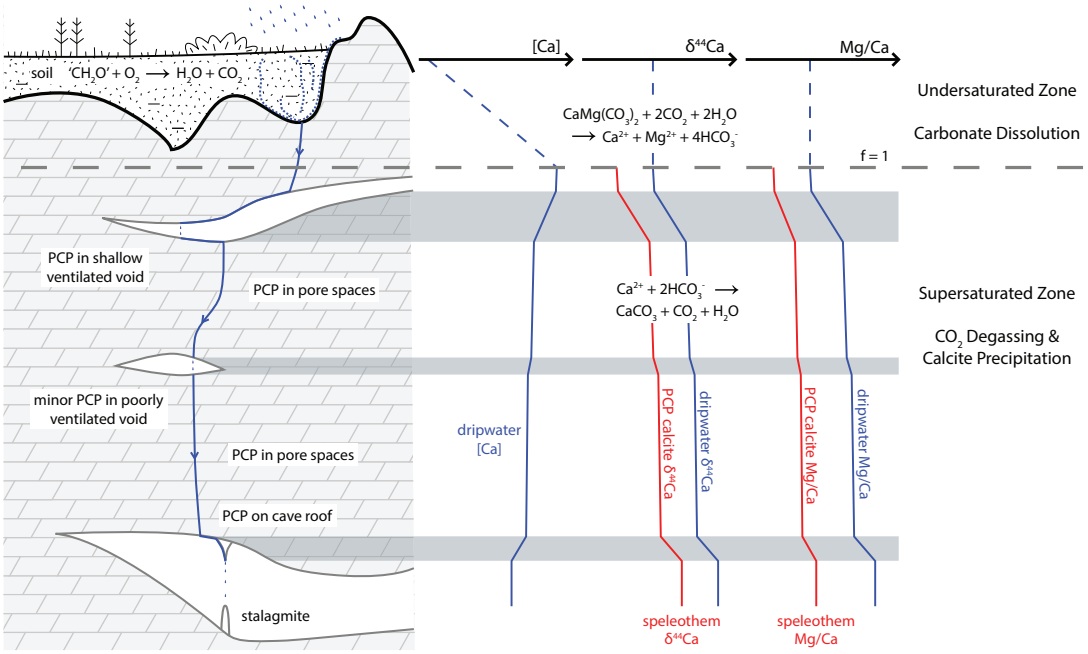



Figure 3.4: A schematic view of Ca cycling through a cave system showing evolution of dripwater Ca concentration ([Ca]), $\delta^{44/42}$Ca and Mg/Ca through PCP.

## 3.5 Discussion

### 3.5.1 Interpreting Cave Ca Isotopes

A schematic view of the cave cycling of Ca is shown in Figure 3.4. Ca is predominantly sourced from bedrock in the soil and upper epikarst, where carbonate dissolution is driven by $CO_2$ produced by soil respiration. Waters percolating through the aquifer encounter air pockets with $pCO_2$ lower than the soil waters with which they are equilibrated, leading to $CO_2$ degassing and subsequent PCP (Fairchild and Baker, 2012). The evolution of dripwater and precipitate Ca isotopes through this process may usefully be thought of in terms of Rayleigh distillation, in which the isotopic compositions of the solution and instantaneous precipitate are given by:

$$r_d = r_0 * f^{\alpha-1} \tag{3.1}$$

$$r_s = \alpha * r_0 * f^{\alpha-1} \tag{3.2}$$

$r_d$ is the Ca isotope ratio in the dripwater ($r_d = \delta_{drip}/1000 + 1$). $r_s$ is the Ca isotope ratio in the instantaneous precipitate from the dripwater ($r_s = \delta_{CaCO3}/1000 + 1$). $r_0$ is the initial dripwater Ca isotope ratio ($r_0 = \delta_{drip}^{f=1}/1000 + 1$). $\alpha$ is the Ca isotope fractionation factor between calcite and water. $f$ is the fraction of Ca remaining in the solution, a measure of PCP.

Equations 3.1 and 3.2 describe the evolution of dripwater and precipitate chemistry from an initial solution chemistry ($f = 1$) towards a hypothetical fully Ca-depleted solution ($f = 0$). As PCP occurs, $f$ decreases. Under this simple model a speleothem is formed from instantaneous calcite precipitates whose composition is given by Equation 3.2.

Equations 3.1 and 3.2 may be re-arranged to give explicit expressions for $f$, based on dripwater (Equation 3.3) or speleothem (Equation 3.4) data:

$$f = \left(\frac{r_d}{r_0}\right)^{\frac{1}{\alpha-1}} \tag{3.3}$$

$$f = \left(\frac{r_s}{\alpha * r_0}\right)^{\frac{1}{\alpha-1}} \tag{3.4}$$

From these equations, $f$ may be calculated either in the modern environment or from the speleothem archive using site-specific values for the Ca isotope fractionation factor ($\alpha$) and initial solution composition ($r_0$). This simple model provides a powerful tool for quantitative PCP reconstruction and is used in Sections 3.5.2 and 3.5.3 to interpret modern and 8.2kyr event Ca isotope data.

### 3.5.2 Modern Ca Isotopes

Modern HS4 dripwaters ($\delta^{44/42}$Ca= 0.68 ‰) are isotopically heavy relative to the dolomite bedrock source ($\delta^{44/42}$Ca= 0.40 ‰). This is best explained as the result of PCP. The invariance of dripwater and glass plate $\delta^{44/42}$Ca values over the monitoring period suggest that PCP at Heshang Cave is rather insensitive to seasonal changes in temperature and rainfall (Figure 3.2). This is likely the result of long (>1 year) groundwater residence

times in the aquifer above the cave.

I argue that the initial dripwater Ca isotopic composition is equal to that of the dolomite bedrock, giving $r_0 = 1.0004 \pm 0.00007$. Firstly, this requires that dolomite dissolution is congruent with respect to Ca isotopes. Incongruent dissolution (simultaneous dolomite dissolution and calcite precipitation), if occurring, is unlikely to modify dripwater Ca isotopes as any calcite precipitated via this process is expected to preferentially and rapidly re-dissolve due to the relative dissolution rates of calcite and dolomite (Chou et al., 1989). Secondly, the bedrock sample is assumed to be representative of the Ca source. The bedrock sample characterises a dolomite source; however, this does not rule out a calcite source somewhere in the epikarst. While the potential for such a source is acknowledged, it is not constrained by the dataset presented here and has not been observed in the field, and so is not considered further. For the purposes of this study any aeolian contribution to dripwater Ca is considered negligible.

Ca isotope fractionation between water and calcite, introduced in Section 1.3.2, has been studied in beaker-type $CaCO_3$ precipitation experiments (Lemarchand et al., 2004; Marriott, 2004; Gussone et al., 2005; Tang et al., 2008), under cave-analogue conditions in the laboratory (Reynard et al., 2011) and during travertine deposition in nature (Yan et al., 2016). Ca isotope fractionation measured at the HS4 drip-site ($\Delta^{44/42}$Ca $= -0.63$ ‰) is similar to that found by Yan et al. (2016) in travertine-depositing pools ($\Delta^{44/42}$Ca $= -0.6$ ‰) and falls within the range of values from laboratory calibrations (e.g. $-0.75 < \Delta^{44/42}\text{Ca} < -0.23$; Tang et al., 2008). The dominant control on the fractionation factor is calcite growth rate, itself controlled by saturation state (Tang et al., 2008). Growth rate may vary along the drip flow path as PCP occurs. However, calcite growth rates during PCP are poorly constrained, making it difficult to incorporate kinetic controls into this model. Instead I prefer to apply the drip-site fractionation factor to the entire flow path. Potential variations in the fractionation factor across the 8.2kyr event are discussed in Section 3.5.3.

Using the values for dripwater composition, initial solution isotopic composition and Ca fractionation factor presented here, Equation 3.3 gives a mean value of $f = 0.64 \pm 0.03$

(95 % CI, n = 15) over the monitoring period. This indicates that, under the current hydroclimatic regime, ∼36 % of initial dripwater Ca is precipitated on the drip flow path before dripwaters arrive at the site of stalagmite formation. This modern calibration provides the framework necessary for quantitative interpretation of past records captured in the stalagmite, including that spanning the 8.2kyr event.

### 3.5.3 8.2kyr Event

The 8.2kyr event in HS4 has previously been identified as a period of aridity based on oxygen isotopes (Hu et al., 2008b), Mg/Ca ratios (Liu et al., 2013) and radiocarbon content (Noronha et al., 2014). Liu et al. (2013) interpreted a Mg/Ca excursion at the event as a response to increased PCP. This section further develops this interpretation using Ca isotopes.

#### 3.5.3.1 PCP Reconstruction

The positive Ca isotope excursion at the 8.2kyr event (Figure 3.3) could be explained by changes in source composition, the Ca isotope fractionation factor, or PCP (Equation 3.2).

To explain the observed $\delta^{44/42}$Ca excursion by changing source composition would require a change in the $\delta^{44/42}$Ca of dissolving bedrock of ∼0.35 ‰. This is approximately equal in magnitude to the maximum change in seawater $\delta^{44/42}$Ca observed across the Phanerozoic (Farkaš et al., 2007a). Isotopic homogeneity (or otherwise) of individual marine carbonate formations is poorly constrained, but such large variation above Heshang Cave seems unlikely given the narrow range of seawater values through time. A source composition control on the observed $\delta^{44/42}$Ca signal would also not explain the correlations between $\delta^{44/42}$Ca, Mg/Ca, Sr/Ca and Ba/Ca.

The magnitude of the Ca isotope fractionation factor is affected by kinetic factors (DePaolo, 2011; Reynard et al., 2011) and varies with calcite growth rate, as controlled by saturation state (Lemarchand et al., 2004; Tang et al., 2008). To explain the $\delta^{44/42}$Ca excursion (Figure 3.3) as the result of changing fractionation factor would require a de-

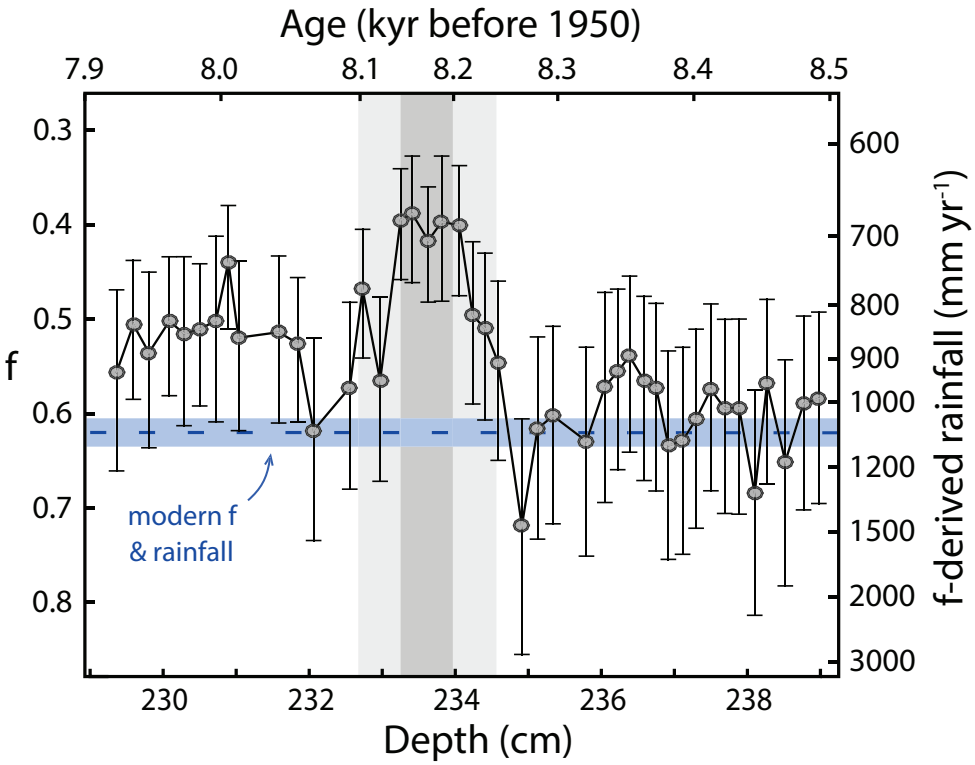



Figure 3.5: PCP reconstruction across the 8.2kyr event using Equation 3.4. Uncertainty on $f$ is calculated using a Monte-Carlo approach. The right-hand y-axis shows the $f$-derived rainfall reconstruction. The calculation of rainfall from $f$ is described in Section 3.5.3.2. Modern $f$ and rainfall values are shown in blue. The grey shaded area shows the 8.2kyr event as identified by Liu et al. (2013). Note that the rainfall axis is non-linear.

crease in $\Delta^{44/42}$Ca from $-0.63\,‰$ to $-0.08\,‰$. Tang et al. (2008) found that a ten-fold decrease in growth rate increases $\Delta^{44/42}$Ca by $0.22\,‰$ at $25\,°C$. If this sensitivity applies to Heshang Cave, a 360-fold decrease in growth rate is required to explain the Ca isotope excursion. This is inconsistent with annual layer thickness data which record little change in linear extension rate in HS4 at the onset of the 8.2kyr event (Liu et al., 2013). Although linear extension rate and growth rate are not entirely equivalent, it is very difficult to reconcile a huge change in one without significant change in the other.

Neither changes in source composition nor in fractionation factor can readily account for the observed Ca isotope excursion. Instead PCP is likely the primary control on the Ca isotope record. Changes in $f$ across the 8.2kyr record may therefore be calculated using Equation 3.4. The resulting $f$ record is shown in Figure 3.5. Prior to the 8.2kyr event $f$ varies about a mean of 0.60, indistinguishable from the modern value calculated from dripwater chemistry ($f_{modern} = 0.64$). $f$ decreases to a mean value of 0.40 during the central portion of the 8.2kyr event, suggesting a 50 % increase in the fraction of Ca being

removed by PCP. This provides strong evidence for decreased local rainfall at the onset of the event. PCP, and presumably rainfall, remain remarkably constant through the central, extreme portion of the event. After the 8.2kyr event PCP decreases, although remains slightly elevated relative to the pre-event baseline, hinting at prolonged local aridity after the event.

#### 3.5.3.2 Quantitative Rainfall Reconstruction

Quantitative PCP estimates provide an important step towards quantitative speleothem-based local rainfall reconstruction. The relationship between PCP and rainfall is likely complex and site-specific but can, to a first approximation, be based on a simple model for flow through the karst system above the cave. During periods of reduced rainfall decreased pressure head, saturation and hydraulic conductivity in the bedrock are expected; this leads to decreased flow rates and hence increased groundwater residence time above the cave (Schwartz and Zhang, 2002). It is widely assumed that this increase in residence time, potentially combined with reduced saturation, allows for more extensive $CO_2$ degassing and PCP (e.g. Cruz et al., 2007; Sinclair et al., 2012).

A one-box model of the aquifer above the cave, at steady state, gives the residence time of water above the cave as the ratio of aquifer volume to the flux of water added to the aquifer by rainwater infiltration. Here the infiltration flux is considered proportional to local rainfall, giving a residence time inversely proportional to local rainfall. The extent of PCP is assumed to be proportional to this residence time. Under these assumptions the extent of PCP, calculated from Ca isotopes, is inversely proportional to effective rainfall.

This relationship may be quantified using the modern calibration dataset, where mean annual rainfall of $1144\,\mathrm{mm\,yr^{-1}}$ results in $f = 0.63$. This tie-point between rainfall and $f$ allows calculation of $f$-derived mean annual rainfall across the 8.2kyr event. The results of this approach are marked on the right-hand axis of Figure 3.5. Ca isotopes suggest a decrease in rainfall of approximately a third at the onset of the 8.2kyr event, with rainfall dropping from modern levels to $\sim 700\,\mathrm{mm\,yr^{-1}}$ for $\sim$80 years.

This result differs from Hu et al. (2008b), who estimated Holocene rainfall change

in this region by differencing $\delta^{18}$O records from spatially separated caves. They report only a 7 % decrease in rainfall at the 8.2kyr event relative to modern. However, due to low-resolution data binning and uncertainty in correlating ages between the two sites, their method is unable to fully resolve the 8.2kyr event and the reported 7 % decrease is likely a lower bound. While highly simplified, the model used here provides a framework for quantitative local rainfall reconstruction based on a single stalagmite. This is used to fully resolve rainfall change during the 8.2kyr event at Heshang Cave.

#### 3.5.3.3 Comparison with Trace Element Records

A multi-proxy approach utilising trace elements and Ca isotopes may offer insight into controls on speleothem metal chemistry. Speleothem Mg/Ca, Sr/Ca and Ba/Ca ratios are well known proxies for PCP change (Huang et al., 2001; Cruz et al., 2007; Sinclair et al., 2012), and provide a useful test for the PCP control on Ca isotopes. In this dataset, Sr/Ca and Ba/Ca ratios mirror changes in $\delta^{44/42}$Ca and show identical structure within the 8.2kyr event (Figure 3.3). The Mg/Ca record, while broadly similar, shows a significantly shorter excursion during the 8.2kyr event.

The evolution of calcite Mg/Ca, Sr/Ca and Ba/Ca ratios as PCP occurs may be described by Rayleigh distillation, analogous to Ca isotope evolution:

$$X/Ca_{CaCO3} = D_X * X/Ca_0 * f^{D_X - 1} \tag{3.5}$$

Where $X$ is Mg, Sr or Ba. $D_x$ is the partition coefficient of $X$ between calcite and water and $X/Ca_0$ is the initial X/Ca ratio of the dripwater at $f = 1$. Equations 3.2 and 3.5 predict a relationship between X/Ca and $\delta^{44/42}$Ca as a function of PCP; these are plotted in Figure 3.6. Partition coefficients are taken from Day and Henderson (2013). X/Ca$_0$ ratios are generally unknown; here they are fitted to the dataset giving Mg/Ca$_0$ = 1.54 mol mol$^{-1}$, Sr/Ca$_0$ = 0.64 mmol mol$^{-1}$ and Ba/Ca$_0$ = 1.67 mmol mol$^{-1}$. Note that while X/Ca$_0$ ratios are fitted to the dataset, the gradients of model lines in Figure 3.6 are rather insensitive to these values and remain robust tests of PCP.

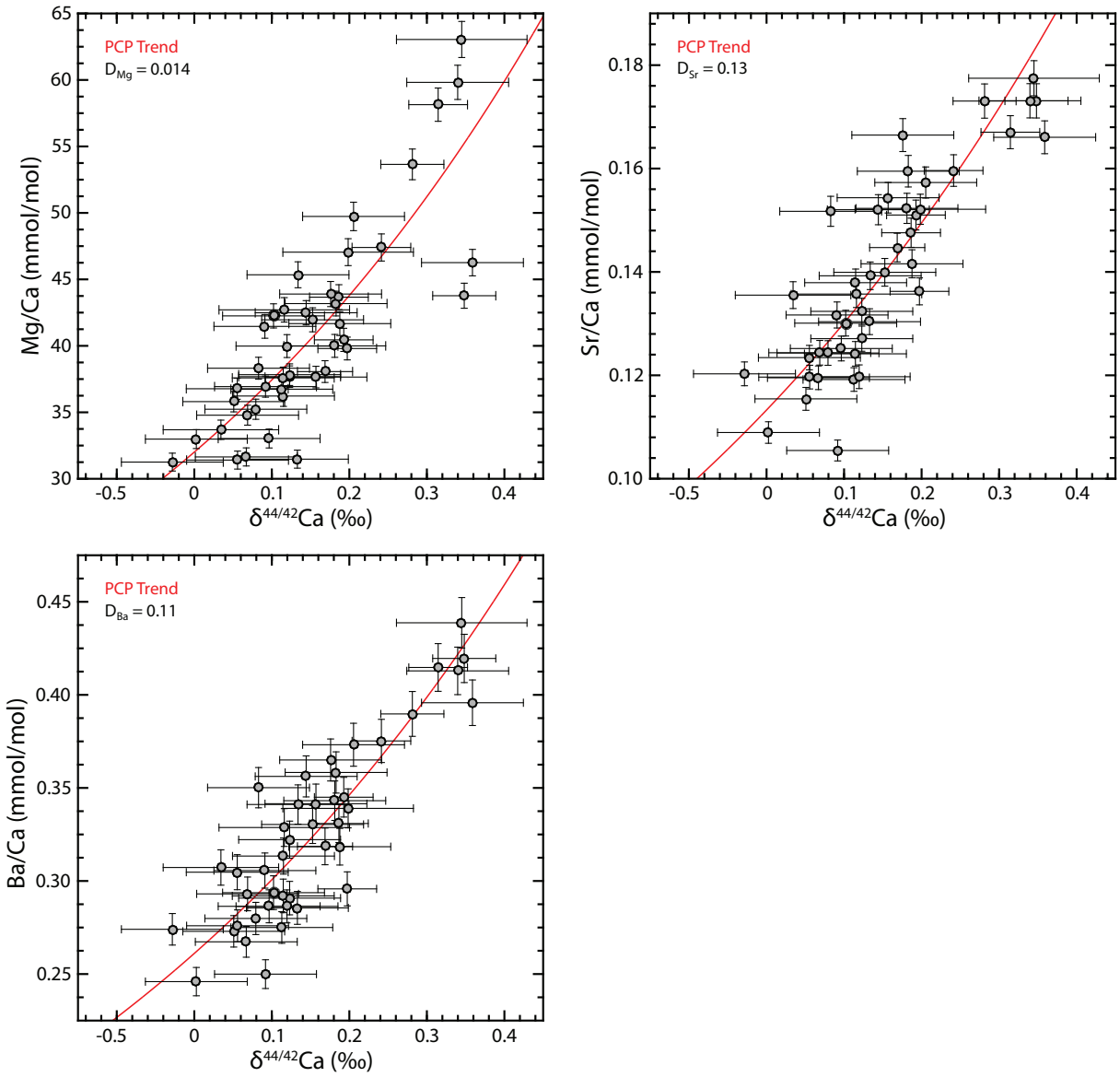



Figure 3.6: Plots of Mg/Ca, Sr/Ca and Ba/Ca against $\delta^{44/42}$Ca across the 8.2kyr event. Model PCP trends are shown in red. These vectors are calculated using Equations 3.2 and 3.5. Partition coefficients used in the model fit are from Day and Henderson (2013), calculated at 19 °C. Initial solution Mg/Ca, Sr/Ca and Ba/Ca ratios are fitted to the dataset: $Mg/Ca_0 = 1.54\,mol\,mol^{-1}$, $Sr/Ca_0 = 0.64\,mmol\,mol^{-1}$, $Ba/Ca_0 = 1.67\,mmol\,mol^{-1}$.

Figure 3.6 shows trace element and Ca isotope data largely fall on the expected PCP trends, supporting the interpretation based on Ca isotopes alone. Nevertheless, there are deviations from this trend, particularly in the Mg/Ca ratio. It has long been understood that factors other than PCP affect speleothem trace element chemistry. For example, partitioning of Mg between calcite and water is temperature-dependent (Nürnberg et al., 1996; Day and Henderson, 2013) and dripwater Mg/Ca ratios may vary due to variable contributions from calcite and dolomite end-members (Fairchild et al., 2000) or aerosol sources (Moreno et al., 2010). These factors likely contribute to decreasing Mg/Ca during

the central portion of the 8.2kyr event (Figure 3.3), where PCP remains constant (Figure 3.5). In isolation, Mg/Ca in this interval might be interpreted as responding to PCP; this would lead to an underestimate of the duration of the local dry event associated with the 8.2kyr event. This highlights the value of a multi-proxy approach incorporating Ca isotope analysis.

$X/Ca_0$ values fitted to the dataset are of potential palaeoclimate interest as recorders of trace element sources. However, values calculated by this method are highly sensitive to the partition coefficients assumed. Use of laboratory-derived $D_X$ values is therefore problematic as their extrapolation to natural settings is likely imprecise (Morse and Bender, 1990). For example, the $Mg/Ca_0$ ratio of $1.54\,mol\,mol^{-1}$, calculated using a $D_{Mg}$ value of 0.014, is unfeasibly high. Re-calculation using the $D_{Mg}$ value of 0.031 calculated by Huang et al. (2001) would instead yield a $Mg/Ca_0$ ratio of $720\,mmol\,mol^{-1}$. Site-specific partition coefficients are clearly necessary for any detailed discussion of $X/Ca_0$.

Quantitative agreement between Mg/Ca, Sr/Ca, Ba/Ca and Ca isotopes shows all four proxies are responding primarily to PCP. However, this may not be the case in other settings or on longer timescales. Changes in trace element source or partition coefficient may obscure or modify trace element PCP signals, as demonstrated by the Mg/Ca record. Furthermore, trace element analyses may be biased by non-carbonate phases in dirty samples. Ca isotopes are less susceptible to contamination, have a constant source composition through time, and the fractionation factor is unlikely to vary greatly under typical cave conditions. I argue that Ca isotopes are a more robust recorder of PCP signals and may be used in a multi-proxy approach to quantify PCP and identify other potential controls on speleothem metal chemistry.

## 3.6 Conclusions

This first systematic study of Ca isotopes in a cave system demonstrates that dripwater Ca isotopes are enriched relative to bedrock due to PCP occurring earlier in the drip flow path. Coeval measurements of Ca isotopes from dripwater and calcite formed on glass

plates indicate a site-specific Ca isotope fractionation factor ($\Delta^{44/42}$Ca) of $-0.63$ ‰, which agrees well with other determinations of the fractionation between calcite and water. A simple Rayleigh fractionation model indicates that ∼36 % of the initial dripwater Ca is removed by PCP in the modern environment prior to the site of stalagmite formation.

In a stalagmite growing at the same site as the modern study, Ca isotopes show a positive excursion during the aridity of the 8.2kyr event, correlating well with positive excursions in established PCP proxies (Mg/Ca, Sr/Ca and Ba/Ca). I use this record to quantify PCP across the 8.2kyr event, and show that it increased from the modern-like value of 40 % to 60 % Ca removal. This increase in PCP suggests, for a simple one-box model of the aquifer system, a reduction in mean annual rainfall in this region of Central China by a third during the 8.2kyr event.

Ca isotopes are also used to identify non-PCP controls on speleothem trace element chemistry. I show that the Mg/Ca response during the 8.2kyr event may be affected by changing source composition or partition coefficient. This has important implications for Mg/Ca-based PCP reconstructions.

This study demonstrates the potential of Ca isotope records in quantifying PCP from speleothem data — a key step towards rainfall reconstructions. Ca isotopes are less susceptible to variable source composition and detrital contamination than trace element proxies, making them particularly useful on long timescales or in 'dirty' speleothems where trace element chemistry is influenced by non-carbonate phases.

# Chapter 4

# Heshang Cave Monitoring 2011-2013

## 4.1 Introduction

This chapter discusses Ca isotope and trace element analyses made on a range of modern dripwaters and calcite precipitates from Heshang Cave, sampled in the two-year period 2011–2013. The purpose of this research was to better understand modern metal cycling through Heshang Cave with a view to more robust interpretation of published and unpublished data from speleothem time-series. This builds on the work presented in Chapter 3.

Chapter 3 used Ca isotope measurements from one Heshang cave drip site over a two-year period (2005–2007) to better understand Ca isotopes in a natural cave environment. The results showed temporal stability over this period and produced a robust fractionation factor ($\Delta^{44/42}$Ca) between calcite and dripwater. This value of $\Delta^{44/42}$Ca was instrumental in reconstructing absolute rainfall over hundreds of years (Chapter 3) from measurements of the HS4 speleothem.

Here I extend the Ca isotope monitoring record of Chapter 3 with data from the period 2011–2013 and from additional dripsites within Heshang Cave. Ca isotope measurements are supplemented by cation analyses to allow a more complete discussion of speleothem metal sourcing. Additionally, dripwater filtration experiments were performed to investigate the contribution of any particulate phase to Heshang dripwater metal concentrations. Together, these analyses provide a more complete description of metal sourcing at Heshang

Cave, and may help improve understanding of the processes occurring in the unsaturated zone. Such an understanding is directly relevant to the interpretation of speleothem metal isotope and trace element time-series, as well as comparisons between speleothems within a cave (e.g. to make composite records).

## 4.2 Samples & Methods

### 4.2.1 Samples

Samples used here are from the Heshang Cave monitoring programme, collected and provided by Chaoyong Hu & colleagues. Samples cover the period May 2011 to April 2013. Full monitoring details and collection procedures are detailed elsewhere (Hu et al., 2005; Johnson et al., 2006; Hu et al., 2008a). Samples were selected for three separate sub-studies: time-series analyses at the HS4 drip site, an inter-site comparison at Heshang cave and a filtration experiment to analyse any particulate content in Heshang Cave dripwaters.

Aliquots of acidified dripwater (2–3 mL) and farmed glass plate calcite (∼300 µg) from the HS4 dripsite (D3) provide monthly resolution through the monitoring period. Dripwater samples were collected on individual days during visits to the cave site. Glass plates were left between visits and each represents cumulative calcite growth over approximately one month.

Dripwater and glass plate samples from four additional sites were provided for the dripsite comparison experiment. These samples cover June-August 2012 from drip sites D1, D2, D4 and D5. Site D1 is located furthest from the cave entrance while D5 is nearest the entrance; other samples lie between in numerical order. Due to the geometry of Heshang cave, D1 is overlain by the greatest thickness of dolomite bedrock, and this thickness decreases towards the cave entrance (see map in Figure 1.12). No calcite sample exists for D1 as it is not actively precipitating calcite.

Large volume samples of unacidified dripwater samples were provided for four drip sites

at Heshang Cave (D2, D3, D5, W2) for filtration experiments. D2, D3 and D5 are natural, active speleothem-forming drips — the same sites present in the time-series dataset. W2 is from a drip site that re-activated during sampling of a stalactite. Dripwaters from W2 were are cloudy with particulate matter while all others appear clear to the eye. All samples were sealed in PP bottles immediately after sampling to minimise $CO_2$ degassing and calcite precipitation.

Finally, I include a sample of Heshang Cave dolomite bedrock for trace element analysis to better constrain trace element sources.

### 4.2.2 Preparation & Analysis

All time-series and dripsite comparison samples were analysed for trace metals and Ca isotopes. A brief summary follows; procedures used are fully described in Chapter 2.

Carbonate samples were dissolved in 2 mL 2 % $HNO_3$. Small aliquots of dissolved carbonates and acidified dripwater were taken for Ca and trace element concentration analysis on an Element 2 ICP-MS as described in Section 2.4. Analytical uncertainty was quantified as 2RSD on repeat standard measurements: Cd/Ca = 6.3 %, Ba/Ca = 5.4 %, U/Ca = 9.4 %, Na/Ca = 2.4 %, Mg/Ca = 2.8 %, P/Ca = 8.9 %, S/Ca = 10.3 % and Sr/Ca = 2.5 %.

Following determination of Ca concentrations during elemental analysis, stock samples were subsampled for Ca isotope analysis. Dripwater aliquots (∼50 µg Ca) were processed through prepFAST-MC to isolate their Ca content (method given in Section 2.3.3.1). Carbonate aliquots did not require separation chemistry due to their low trace element concentrations and could be analysed directly. Ca isotope samples were diluted to 5 ppm Ca in 1.8 mL 2 % $HNO_3$, drying and re-dissolving where necessary, and analysed via the standard procedure described in Section 2.2.6. Uncertainty is quoted as the 95 % confidence interval on the mean of 3–5 repeat analyses. Long-term accuracy and precision was measured using the secondary standard NIST SRM 915b, giving $\delta^{44/42}\text{Ca} = 0.37 \pm 0.11$ ‰ (2 s.d., n = 35), $\delta^{43/42}\text{Ca} = 0.18 \pm 0.18$ ‰ (2 s.d. n = 35). These values are in agreement with published work (Heuser and Eisenhauer, 2008; Schiller et al., 2012).

### 4.2.3 Filtering

Samples to be analysed for the effect of particulates (D2, D3, D5 and W2) were well shaken (to suspend any particulate matter) and two aliquots taken — one to acidify and analyse directly and one to filter. Filtering was performed using Minisart single use 200 nm filters attached to BD Plastipak syringes. This equipment was water-rinsed only — acid leaching was not thought necessary as dripwater samples had not yet been acidified. A water blank was filtered to verify this.

Approximately 2 mL aliquots of each sample were filtered into acid-cleaned 2 mL Eppendorf tubes. Unfiltered aliquots (also ∼2 mL) were pipetted directly into acid-cleaned 2 mL Eppendorf tubes. All samples were then acidified by addition of a few drops of concentrated distilled $HNO_3$, shaken well and left overnight to allow for dissolution of any particulate phases.

Samples were later processed for trace element analysis using the same method as the dripwater and glass plate samples described above. Ca isotope analysis was not conducted as part of the filtration experiment.

## 4.3 Results

| Date | Site | Type | $\delta^{44/42}$Ca (‰ 915a) | 95 ‰ CI (±) | Ba/Ca mmol mol$^{-1}$ | U/Ca µmol mol$^{-1}$ | Na/Ca mmol mol$^{-1}$ | Mg/Ca mmol mol$^{-1}$ | P/Ca mmol mol$^{-1}$ | S/Ca mmol mol$^{-1}$ | Sr/Ca mmol mol$^{-1}$ | Ca concentration µg g$^{-1}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 08/05/2011 | 3 | Calcite | 0.06 | 0.07 | 0.14 | 0.31 | bdl | 25.8 | 1.53 | 1.20 | 0.08 | - |
| 14/06/2011 | 3 | Calcite | 0.00 | 0.07 | 0.19 | 0.28 | 0.41 | 25.0 | 2.02 | 1.15 | 0.09 | - |
| 14/07/2011 | 3 | Calcite | 0.00 | 0.08 | 0.19 | 0.24 | 1.15 | 28.0 | 1.37 | 1.03 | 0.09 | - |
| 14/08/2011 | 3 | Calcite | 0.03 | 0.07 | 0.20 | 0.23 | 0.35 | 25.8 | 1.05 | 0.99 | 0.09 | - |
| 13/09/2011 | 3 | Calcite | 0.04 | 0.07 | 0.15 | 0.34 | 0.38 | 25.1 | 1.57 | 1.24 | 0.08 | - |
| 14/10/2011 | 3 | Calcite | 0.05 | 0.07 | 0.19 | 0.25 | 0.43 | 24.8 | 1.03 | 1.00 | 0.09 | - |
| 14/11/2011 | 3 | Calcite | 0.02 | 0.08 | 0.19 | 0.24 | 1.02 | 25.9 | 1.04 | 0.98 | 0.09 | - |
| 14/12/2011 | 3 | Calcite | 0.01 | 0.07 | 0.17 | 0.25 | 0.62 | 22.5 | 1.21 | 1.11 | 0.09 | - |
| 15/01/2012 | 3 | Calcite | 0.07 | 0.07 | 0.12 | 0.30 | 0.62 | 23.4 | 1.73 | 1.26 | 0.07 | - |
| 14/02/2012 | 3 | Calcite | 0.04 | 0.07 | 0.18 | 0.27 | 0.38 | 22.4 | 1.11 | 1.00 | 0.08 | - |
| 15/03/2012 | 3 | Calcite | -0.01 | 0.07 | 0.18 | 0.25 | 0.74 | 22.7 | 1.17 | 0.96 | 0.09 | - |
| 14/04/2012 | 3 | Calcite | 0.02 | 0.07 | 0.18 | 0.28 | 0.44 | 21.7 | 1.29 | 1.02 | 0.09 | - |
| 16/05/2012 | 3 | Calcite | 0.09 | 0.07 | 0.18 | 0.25 | 0.44 | 25.7 | 1.35 | 0.90 | 0.09 | - |
| 16/06/2012 | 3 | Calcite | 0.06 | 0.07 | 0.17 | 0.27 | 0.68 | 25.1 | 2.36 | 1.15 | 0.08 | - |
| 15/07/2012 | 3 | Calcite | 0.02 | 0.07 | 0.20 | 0.27 | 0.60 | 27.0 | 2.31 | 1.18 | 0.09 | - |
| 15/08/2012 | 3 | Calcite | 0.06 | 0.07 | 0.18 | 0.25 | 0.53 | 29.2 | 2.01 | 0.95 | 0.08 | - |
| 15/09/2012 | 3 | Calcite | 0.02 | 0.07 | 0.17 | 0.23 | 0.35 | 25.9 | 1.48 | 0.94 | 0.08 | - |
| 15/10/2012 | 3 | Calcite | -0.04 | 0.07 | 0.19 | 0.29 | 0.42 | 22.7 | 1.57 | 1.16 | 0.09 | - |
| 14/11/2012 | 3 | Calcite | 0.03 | 0.07 | 0.18 | 0.27 | 0.41 | 24.6 | 1.55 | 1.01 | 0.08 | - |
| 15/12/2012 | 3 | Calcite | 0.04 | 0.07 | 0.18 | 0.28 | bdl | 22.5 | 1.83 | 1.13 | 0.09 | - |
| 15/01/2013 | 3 | Calcite | 0.04 | 0.08 | 0.18 | 0.27 | 0.45 | 23.4 | 1.88 | 1.10 | 0.09 | - |
| 15/02/2013 | 3 | Calcite | 0.01 | 0.07 | 0.19 | 0.28 | bdl | 21.5 | 1.44 | 1.04 | 0.09 | - |
| 17/03/2013 | 3 | Calcite | 0.08 | 0.08 | 0.16 | 0.26 | 0.47 | 22.8 | 1.28 | 0.95 | 0.08 | - |
| 30/05/2011 | 3 | Dripwater | 0.45 | 0.06 | 0.75 | 1.26 | 35.37 | 819 | 0.46 | 125 | 0.51 | 80 |
| 29/06/2011 | 3 | Dripwater | 0.41 | 0.05 | 0.76 | 1.21 | 34.15 | 809 | 0.53 | 122 | 0.51 | 78 |
| 30/07/2011 | 3 | Dripwater | 0.38 | 0.05 | 0.80 | 1.21 | 51.08 | 811 | 0.49 | 137 | 0.52 | 80 |
| 29/08/2011 | 3 | Dripwater | 0.41 | 0.05 | 0.74 | 1.17 | 37.50 | 810 | 0.56 | 138 | 0.51 | 77 |
| 29/09/2011 | 3 | Dripwater | 0.38 | 0.05 | 0.74 | 1.18 | 37.13 | 799 | 0.66 | 128 | 0.51 | 79 |
| 30/10/2011 | 3 | Dripwater | 0.42 | 0.05 | 0.70 | 1.13 | 39.79 | 782 | 0.76 | 128 | 0.49 | 82 |
| 29/11/2011 | 3 | Dripwater | 0.42 | 0.05 | 0.75 | 1.30 | 33.74 | 811 | 0.50 | 127 | 0.50 | 77 |
| 30/12/2011 | 3 | Dripwater | 0.43 | 0.06 | 0.69 | 1.19 | 34.44 | 793 | 0.73 | 125 | 0.50 | 78 |
| 31/01/2012 | 3 | Dripwater | 0.41 | 0.05 | 0.68 | 1.24 | 33.05 | 811 | 0.46 | 127 | 0.49 | 84 |
| 29/02/2012 | 3 | Dripwater | 0.41 | 0.05 | 0.69 | 1.24 | 36.57 | 804 | 0.48 | 128 | 0.50 | 80 |
| 30/03/2012 | 3 | Dripwater | 0.38 | 0.05 | 0.68 | 1.29 | 37.74 | 810 | 0.86 | 124 | 0.50 | 77 |
| 30/04/2012 | 3 | Dripwater | 0.37 | 0.05 | 0.72 | 1.31 | 38.07 | 800 | 0.78 | 118 | 0.50 | 79 |
| 01/06/2012 | 3 | Dripwater | 0.41 | 0.05 | 0.71 | 1.25 | 34.47 | 805 | 1.20 | 126 | 0.50 | 79 |
| 01/07/2012 | 3 | Dripwater | 0.41 | 0.05 | 0.78 | 1.24 | 93.69 | 795 | 3.98 | 136 | 0.52 | 76 |
| 30/07/2012 | 3 | Dripwater | 0.41 | 0.07 | 0.83 | 1.31 | 51.77 | 832 | 1.22 | 130 | 0.55 | 75 |
| 01/09/2012 | 3 | Dripwater | 0.41 | 0.07 | 0.80 | 1.24 | 35.16 | 846 | 0.73 | 132 | 0.54 | 71 |
| 29/09/2012 | 3 | Dripwater | 0.38 | 0.08 | 0.74 | 1.20 | 36.18 | 807 | 0.86 | 127 | 0.51 | 75 |
| 31/10/2012 | 3 | Dripwater | 0.36 | 0.07 | 0.76 | 1.28 | 33.90 | 857 | 0.76 | 129 | 0.53 | 73 |
| 29/11/2012 | 3 | Dripwater | 0.38 | 0.07 | 0.74 | 1.27 | 36.07 | 840 | 0.85 | 142 | 0.52 | 70 |
| 31/12/2012 | 3 | Dripwater | 0.32 | 0.07 | 0.68 | 1.19 | 42.26 | 791 | 0.65 | 128 | 0.49 | 76 |
| 30/01/2013 | 3 | Dripwater | 0.39 | 0.07 | 0.75 | 1.33 | 41.58 | 853 | 0.91 | 140 | 0.53 | 71 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 03/03/2013 | 3 | Dripwater | 0.42 | 0.10 | 0.70 | 1.25 | 33.60 | 833 | 0.52 | 128 | 0.51 | 74 |
| 01/04/2013 | 3 | Dripwater | 0.36 | 0.07 | 0.71 | 1.29 | 38.83 | 848 | 0.51 | 131 | 0.51 | 72 |
| 16/05/2012 | 2 | Calcite | 0.06 | 0.07 | 0.24 | 0.31 | 0.57 | 25.2 | 1.11 | 0.69 | 0.09 | - |
| 16/06/2012 | 2 | Calcite | 0.08 | 0.07 | 0.23 | 0.45 | 0.47 | 27.4 | 0.91 | 0.71 | 0.09 | - |
| 15/07/2012 | 2 | Calcite | 0.09 | 0.07 | 0.24 | 0.27 | 0.73 | 27.4 | 0.89 | 0.76 | 0.09 | - |
| 16/05/2012 | 4 | Calcite | 0.01 | 0.07 | 0.09 | 0.25 | 0.55 | 23.5 | 1.23 | 1.36 | 0.07 | - |
| 16/06/2012 | 4 | Calcite | 0.02 | 0.07 | 0.11 | 0.21 | 0.57 | 23.5 | 1.19 | 1.43 | 0.08 | - |
| 15/07/2012 | 4 | Calcite | 0.08 | 0.07 | 0.09 | 0.23 | 4.79 | 25.9 | 1.24 | 1.40 | 0.08 | - |
| 16/05/2012 | 5 | Calcite | 0.04 | 0.07 | 0.07 | 0.18 | 0.38 | 23.3 | 0.39 | 1.18 | 0.08 | - |
| 16/06/2012 | 5 | Calcite | 0.06 | 0.08 | 0.07 | 0.17 | 0.81 | 26.2 | 0.36 | 1.15 | 0.08 | - |
| 15/07/2012 | 5 | Calcite | 0.15 | 0.07 | 0.07 | 0.17 | 0.48 | 28.9 | 0.35 | 1.10 | 0.08 | - |
| 01/06/2012 | 1 | Dripwater | 0.52 | 0.07 | 2.00 | 2.06 | 161.1 | 1157 | 10.91 | 209 | 0.86 | 56 |
| 01/07/2012 | 1 | Dripwater | 0.77 | 0.07 | 2.73 | 3.57 | 97.40 | 1879 | 0.79 | 353 | 1.21 | 34 |
| 30/07/2012 | 1 | Dripwater | 0.43 | 0.10 | 2.62 | 3.45 | 109.5 | 1930 | 1.58 | 370 | 1.25 | 33 |
| 01/06/2012 | 2 | Dripwater | 0.41 | 0.08 | 1.09 | 1.93 | 46.40 | 878 | 0.51 | 114 | 0.59 | 65 |
| 01/07/2012 | 2 | Dripwater | 0.42 | 0.07 | 1.11 | 1.86 | 43.09 | 872 | 0.51 | 117 | 0.60 | 66 |
| 30/07/2012 | 2 | Dripwater | 0.42 | 0.07 | 1.18 | 1.81 | 61.73 | 924 | 0.83 | 115 | 0.56 | 70 |
| 01/06/2012 | 4 | Dripwater | 0.36 | 0.07 | 0.35 | 1.11 | 28.98 | 817 | 0.36 | 142 | 0.44 | 92 |
| 01/07/2012 | 4 | Dripwater | 0.26 | 0.09 | 0.28 | 0.75 | 17.33 | 576 | 0.45 | 90 | 0.33 | 129 |
| 30/07/2012 | 4 | Dripwater | 0.39 | 0.07 | 0.37 | 0.99 | 26.14 | 814 | 0.42 | 131 | 0.44 | 93 |
| 01/06/2012 | 5 | Dripwater | 0.44 | 0.07 | 0.25 | 0.80 | 24.77 | 871 | 0.31 | 108 | 0.42 | 100 |
| 01/07/2012 | 5 | Dripwater | 0.42 | 0.07 | 0.26 | 0.78 | 24.77 | 844 | 0.38 | 104 | 0.42 | 100 |
| 30/07/2012 | 5 | Dripwater | 0.73 | 0.08 | 0.28 | 0.80 | 31.06 | 847 | 0.44 | 106 | 0.42 | 102 |
| - | - | Bedrock | 0.40* | 0.07* | 0.02 | 0.31 | 2.21 | 1016 | 0.29 | 0.63 | 0.32 | - |

Table 4.1: Heshang cave dripwater and glass plate data 2011–2013. Date gives the collection date of dripwaters, and the mid-point growth date for glass plate calcite samples. D3 is the HS4 dripsite. 'bdl' — below detection limit. Dolomite bedrock Ca isotope data are from Owen et al. (2016).

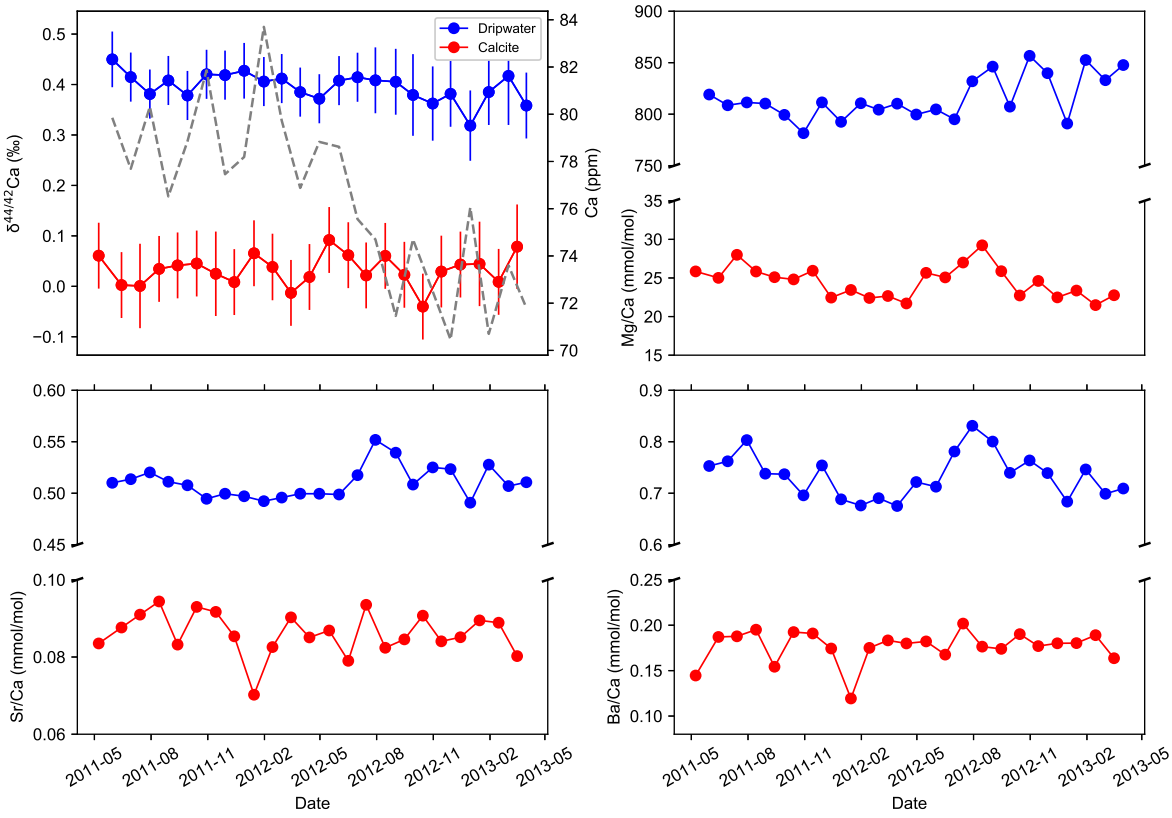



Figure 4.1: Dripwater and glass plate calcite Ca isotopes, Mg/Ca, Sr/Ca and Ba/Ca ratios from the HS4 dripsite (D3) from 2011 to 2013. In blue are shown dripwater values, in red glass plate calcite samples. Dates plotted represent collection dates for dripwaters, and growth mid-point dates for glass plates. Uncertainty on Ca isotope data (top left panel) is quantified as the 95 % confidence interval, as described in the main text. Uncertainty in elemental ratios is also described in the main text. The top left panel also shows dripwater Ca concentrations calculated during trace element analysis. Note the split Ba/Ca and Mg/Ca y-axes.

Figure 4.1 shows Ca concentration, Ca isotope and selected trace element (Mg, Sr, Ba) time-series from the HS4 drip-site (D3) from 2011 to 2013. All data are given in Table 4.1. Dripwater and glass plate calcite $\delta^{44/42}$Ca do not vary significantly over the time period, giving mean values of $\delta^{44/42}Ca_{dripwater} = 0.40 \pm 0.01$ ‰ (95 % CI, n = 23) and $\delta^{44/42}Ca_{calcite} = 0.03 \pm 0.01$ ‰ (95 % CI, n = 23). Across the record, dripwater Ca concentrations drop from approximately 80 ppm in 2011 to 73 ppm in 2013.

Trace element time-series show some significant variation, with dripwater Mg/Ca, Sr/Ca and Ba/Ca all broadly increasing in value and variance from Autumn 2012 without any corresponding increase in glass plate X/Ca records. Across the whole record mean dripwater Mg/Ca = 816 mmol mol$^{-1}$, Sr/Ca = 0.51 mmol mol$^{-1}$ and Ba/Ca = 0.73 mmol mol$^{-1}$. Mean glass plate Mg/Ca = 24.5 mmol mol$^{-1}$, Sr/Ca = 0.086 mmol mol$^{-1}$

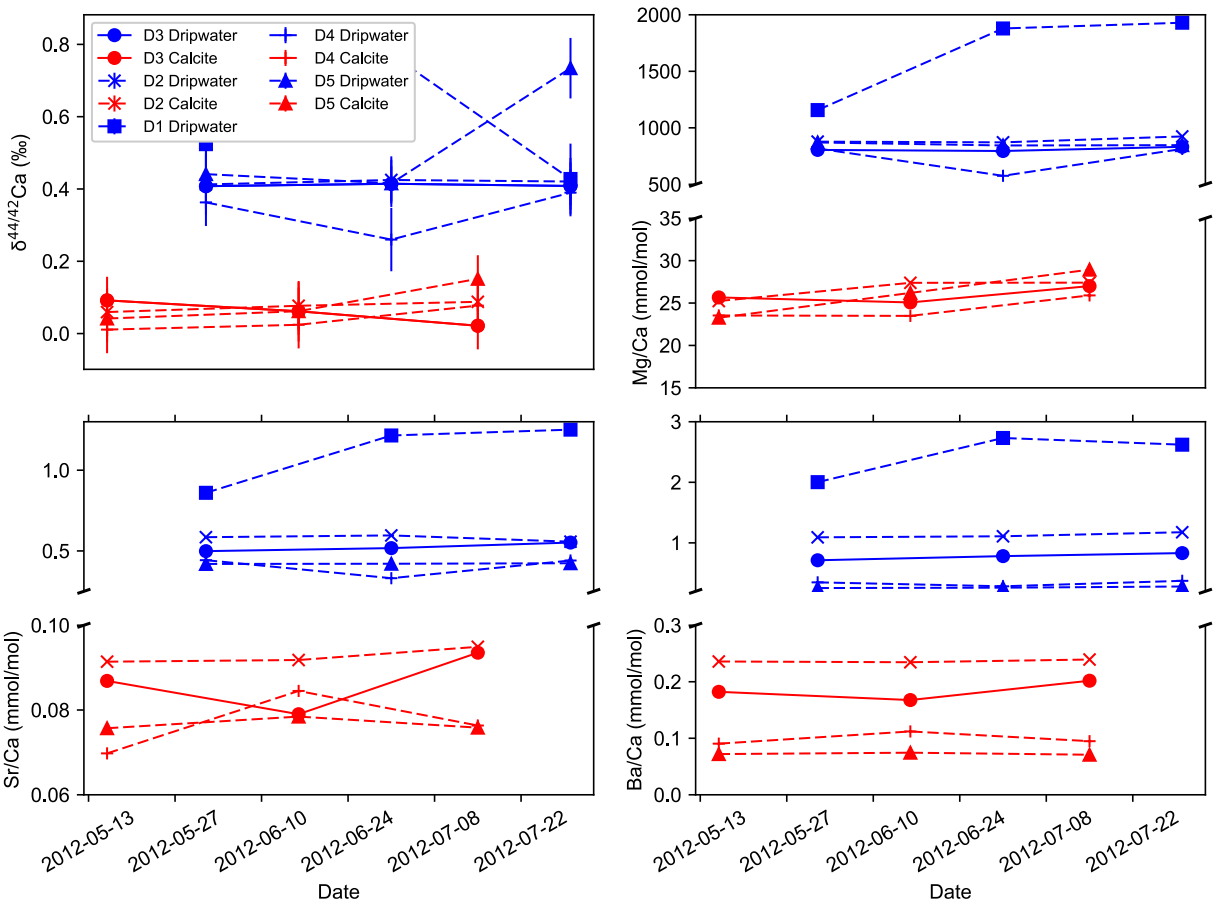



Figure 4.2: Dripwater and glass plate calcite Ca isotopes, Mg/Ca, Sr/Ca and Ba/Ca ratios from drip sites D1-D5 covering June-August 2012. In blue are shown dripwater values, in red glass plate calcite samples. Dates plotted represent collection dates for dripwaters, and growth mid-point dates for glass plates. Uncertainty on Ca isotope data (top left panel) is quantified as the 95 % confidence interval, as described in the main text. Uncertainty in elemental ratios is also described in the main text. Note that Mg/Ca, Sr/Ca and Ba/Ca y-axes are split.

and $\mathrm{Ba/Ca} = 0.18\,\mathrm{mmol\,mol^{-1}}$.

This dataset is used to calculate an effective in-situ Ca isotope fractionation factor between dripwater and glass plate calcite, averaging values across the two-year period. This gives $\Delta^{44/42}\mathrm{Ca}_{calcite-water} = -0.36 \pm 0.02\,‰$ (95 % CI, n = 23). Similarly, trace element data may be used to calculate in-situ partition coefficients of $\mathrm{D}_{Mg} = 0.030 \pm 0.001$, $\mathrm{D}_{Sr} = 0.170 \pm 0.004$, $\mathrm{D}_{Ba} = 0.24 \pm 0.01$, $\mathrm{D}_{U} = 0.22 \pm 0.01$ (95 % CI, n = 23).

Data from different drip-sites in Heshang Cave are compared in Figure 4.2. Three months sampling for each site covers June-August 2012. Note that site D1 is not actively precipitating calcite, as such no glass plate data are available for this site. The apparent range of dripwater $\delta^{44/42}$Ca is large. Sites D3 and D2 show constant and indistinguishable Ca isotope values across the three months ($\delta^{44/42}\mathrm{Ca} = 0.41\,‰$). Dripwater Ca isotopes

at sites D1 and D5 are highly variable, each with a single data point heavier than 0.7 ‰. Values for D4 fall between 0.3 and 0.4 ‰. In contrast, glass plate calcite values for all drip sites do not vary significantly and largely fall within analytical uncertainty of one another in the range 0–0.1 ‰.

Dripwater and glass plate calcite Mg/Ca, Sr/Ca and Ba/Ca data follow consistent trends between sites. For all three elements, dripwater X/Ca concentrations fall in the order $D1 > D2 >= D3 >= D4 >= D5$. This trend is reflected in the corresponding glass plate data for all three elements. D1 shows notably lower dripwater X/Ca concentrations (as well as higher Ca concentration) in June compared to July and August measurements, while at other sites dripwater trace element chemistry does not vary significantly between months.

Table 4.2: Data from Heshang cave dripwater filtering experiments. Top: Elemental concentrations in filtered and unfiltered aliquots. Bottom: Element-to-Ca ratios. The filtering technique used is described in Section 4.2.3.

| | Sample | Ca | Ba | U | Na | Mg | P | S | Sr |
|---|---|---|---|---|---|---|---|---|---|
| | | $\mu g\,g^{-1}$ | $ng\,g^{-1}$ | $ng\,g^{-1}$ | $\mu g\,g^{-1}$ | $\mu g\,g^{-1}$ | $ng\,g^{-1}$ | $\mu g\,g^{-1}$ | $ng\,g^{-1}$ |
| filtered | D2 | 51.2 | 216 | 0.58 | 1.44 | 32.9 | 5.65 | 6.17 | 75.7 |
| | D3 | 24.0 | 126 | 0.39 | 1.31 | 32.1 | 2.07 | 8.68 | 61.5 |
| | D5 | 55.3 | 70.1 | 0.39 | 1.23 | 49.2 | 4.23 | 10.0 | 75.5 |
| | W2 | 44.7 | 102 | 0.30 | 0.59 | 20.7 | 28.7 | 2.34 | 41.9 |
| unfiltered | D2 | 53.5 | 224 | 0.62 | 1.44 | 32.7 | 3.10 | 6.35 | 77.1 |
| | D3 | 37.6 | 149 | 0.45 | 1.30 | 35.3 | 6.45 | 9.27 | 70.4 |
| | D5 | 60.4 | 73.1 | 0.39 | 1.22 | 50.0 | 6.63 | 10.1 | 78.2 |
| | W2 | 118 | 565 | 1.54 | 3.81 | 82.7 | 9.30 | 16.3 | 197 |

| | Sample | Ba/Ca | U/Ca | Na/Ca | Mg/Ca | P/Ca | S/Ca | Sr/Ca |
|---|---|---|---|---|---|---|---|---|
| | | $mmol\,mol^{-1}$ | $\mu mol\,mol^{-1}$ | $mmol\,mol^{-1}$ | $mmol\,mol^{-1}$ | $\mu mol\,mol^{-1}$ | $mmol\,mol^{-1}$ | $mmol\,mol^{-1}$ |
| filtered | D2 | 1.23 | 1.92 | 48.9 | 1059 | 143 | 151 | 0.68 |
| | D3 | 1.53 | 2.76 | 95.3 | 2200 | 111 | 451 | 1.17 |
| | D5 | 0.37 | 1.18 | 38.9 | 1468 | 99 | 227 | 0.62 |
| | W2 | 0.66 | 1.14 | 23.1 | 762 | 830 | 65 | 0.43 |
| unfiltered | D2 | 1.22 | 1.95 | 47.0 | 1008 | 75 | 148 | 0.66 |
| | D3 | 1.16 | 2.02 | 60.2 | 1550 | 222 | 308 | 0.86 |
| | D5 | 0.35 | 1.09 | 35.3 | 1364 | 142 | 209 | 0.59 |
| | W2 | 1.40 | 2.20 | 56.5 | 1159 | 102 | 173 | 0.77 |

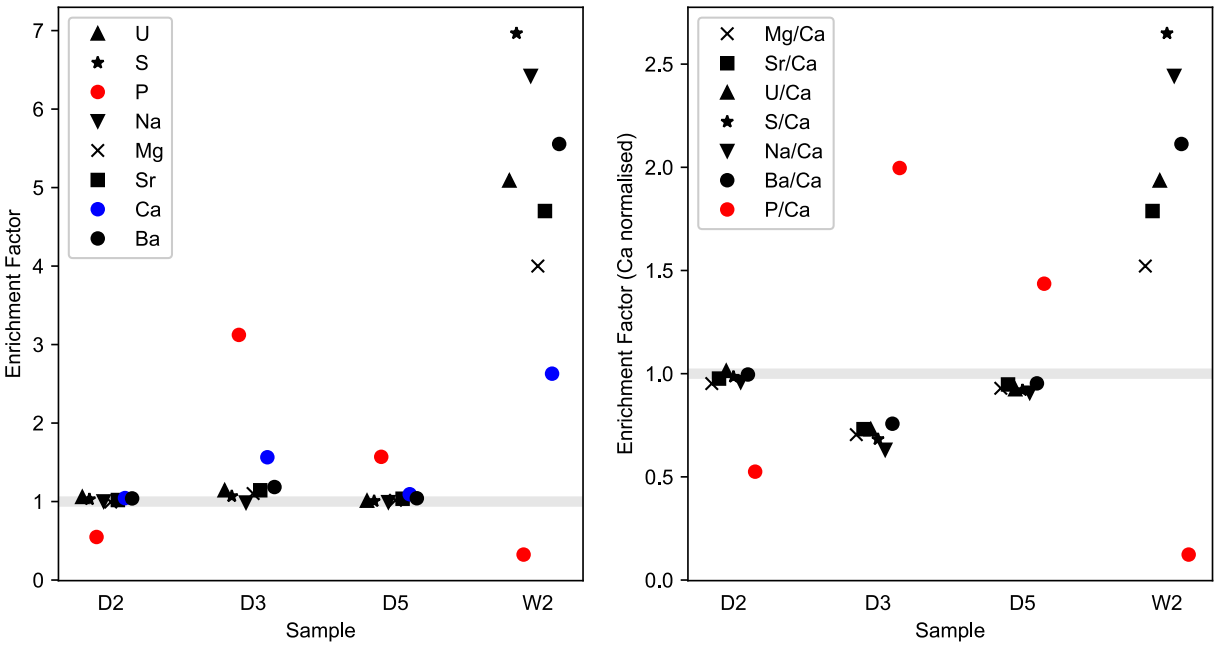



Figure 4.3: Results of dripwater filtration experiments. The left panel shows the concentration of various elements in unfiltered dripwaters normalised to filtered dripwaters (the enrichment factor). The right panel shows the same data normalised to Ca concentrations.

Results from the filtration experiments are given in Table 4.2. Figure 4.3 plots elemental concentrations and ratios in the unfiltered dripwater samples normalised to those in the corresponding filtered sample. This shows the enrichments in each element due to the presence of any particulate phase. Blanks measured from the filtration procedure were negligible, and have been corrected for nevertheless. The left panel of Figure 4.3 shows elemental enrichments in the unfiltered dripwater; for most elements analysed (S, Ca, Mg, Ba, Sr, U and Na), sites D2 and D5 show little-to-no enrichment in the unfiltered samples, suggesting no significant particulate phase affecting these elemental budgets. Site D3 shows a significant enrichment in Ca but not in other metals. W2 shows enrichments in all metals in the unfiltered sample. The right panel of Figure 4.3 shows these data normalised to their respective Ca concentrations, providing information about the X/Ca ratios of any particulates present.

# 4.4 Discussion

## 4.4.1 HS4 Time-series analyses

### 4.4.1.1 Ca Isotopes

The Ca isotope time-series in Figure 4.1 shows no significant deviation from mean values in dripwater or calcite Ca isotopes, suggesting little sensitivity to seasonal changes in the local environment. This is in agreement with Owen et al. (2016), who concluded that Ca isotopes were not sensitive to environmental changes at this drip-site through the years 2005–2007. The mean value for dripwater $\delta^{44/42}$Ca (0.40 ‰) is significantly lighter than that reported previously (0.68 ‰; Owen et al., 2016) while mean values for glass plate calcite, 0.03 ‰ and 0.05 ‰ respectively, are within analytical uncertainty. A change in dripwater Ca isotopes with no corresponding change in glass plate calcite Ca isotopes is unexpected and could be explained by:

1. A change in dripwater Ca isotopes, coupled with a change in effective fractionation factor, coincidentally resulting in no overall change in glass plate Ca isotopes.
2. A laboratory-derived isotope fractionation resulting from dripwater Ca separation chemistry.

Option 1 is a possibility — water samples from other drip-sites in Heshang cave intermittently show similarly high dripwater $\delta^{44/42}$Ca values with no change in calcite $\delta^{44/42}$Ca (Figure 4.2; discussed later). However, there is reason to suspect that analytical artefacts are responsible for the deviation between dripwater $\delta^{44/42}$Ca values reported here and by Owen et al. (2016). The work presented here isolated the Ca content of dripwaters using prepFAST-MC. This method has been tested extensively and shown to result in little-to-no isotope fractionation at similar Ca loads (see Section 2.3.3). In contrast, dripwater samples included in Owen et al. (2016) had their Ca content isolated using the two-step column procedure (Section 2.3.2). Later calibration experiments showed the second column (using SrSpec resin) retains light Ca isotopes, and may result in significant fractionation of the Ca cut, particularly at low Ca loads (Section 2.3.2). Laboratory notes

confirm that relatively small (<20 µg Ca; personal communication, C. Blättler) samples were used during column chemistry processing of the dripwater samples reported in Chapter 3. The fractionation expected from this process (∼0.30 ‰; personal communication, P. Pogge Von Strandmann) likely explains the offset to heavier isotope compositions seen in the 2005–2007 dataset. I conclude that 'true' modern bulk dripwater $\delta^{44/42}$Ca = 0.40 ‰, and there is no evidence for natural variation from this value in either the 2005–2007 or 2011–2013.

$\delta^{44/42}$Ca of unfiltered dripwaters is within analytical uncertainty of the dolomite bedrock composition (0.40 ‰; Table 4.1). As such, dripwater Ca isotopes could be explained as the result of quantitative dissolution of dolomite bedrock, without requiring significant prior calcite precipitation. However, dripwater Mg/Ca (Section 4.4.1.2) and solution particulate matter (Section 4.4.3) both suggest a significant non-dolomite Ca source and provide evidence for PCP, supporting the conclusion of Owen et al. (2016).

#### 4.4.1.2 Trace Elements

Dripwater trace element ratios have been widely used to determine metal sources (e.g. aerosol, soil, bedrock), as well as identifying relative changes in the extend of PCP (Section 1.2.3). Simple partitioning models (e.g. Huang and Fairchild, 2001; Johnson et al., 2006; Owen et al., 2016) have been used to model the incorporation of these signals into speleothem calcite, allowing speleothem-based reconstructions to be made (e.g. Cruz et al., 2007). The dataset presented here may be useful for calibrating these proxies.

Changes in dripwater and glass plate calcite X/Ca ratios (Figure 4.1) across this interval are small. Mean dripwater X/Ca ratios provide a fingerprint of trace element sourcing, potentially modified to higher values by prior calcite precipitation. Mean dripwater Mg/Ca (816 mmol mol$^{-1}$) falls below the 1 mol mol$^{-1}$ ratio expected from quantitative dissolution of the surrounding dolomite bedrock. This could be explained by:

- A non-dolomite dripwater Ca source. Potential candidates include non-bedrock soil sources (e.g. Ca-rich dust), or $CaCO_3$ present in the bedrock, which may represent primary or secondary deposits.

- Particulate Ca-rich phases present in the dripwater (e.g. $CaCO_3$, $CaSO_4$). Dripwaters used in this study were not routinely filtered prior to acidification, rendering this a possibility. Note that such phases would not affect bulk dripwater X/Ca ratios if they precipitated from the host dripwater in a closed system — an external source is required.

Regardless, a simple mixing model applied to dripwater Mg/Ca suggests a minimum of 18 % of dripwater Ca content is derived from Mg-free (i.e. non-dolomite) sources. If PCP is occurring this percentage will be higher. A non-dolomite Ca source may have a different Ca isotopic composition to the dolomite; in particular, a secondary calcite precipitate will be isotopically lighter. This has important implications for the interpretation of dripwater and speleothem Ca isotope ratios.

Mean dripwater Sr/Ca (0.51 mmol mol$^{-1}$) and Ba/Ca (0.73 mmol mol$^{-1}$) are both significantly higher than measured values for the dolomite bedrock (0.32 and 0.02 mmol mol$^{-1}$, respectively). This requires either prior calcite precipitation raising dripwater Sr/Ca and Ba/Ca ratios, or significant non-dolomite sources. In particular, quantitative dolomite dissolution can only account for 3 % of measured dripwater Ba — this requires an additional Ba source, perhaps in addition to PCP.

The in-situ trace element fractionation factors determined across this record are systematically higher than those determined in the cave-analogue laboratory experiments of Day and Henderson (2013). This suggests that more extensive (i.e. cumulative) calcite precipitation is occurring in the cave setting, compared to the near-instantaneous precipitates produced by Day and Henderson (2013). Analysis of a cumulative precipitate would result in a higher effective partition coefficient.

While variation in the trace elements time-series is small, some significant variation is present. All three dripwater X/Ca records increase slightly around August 2012 and remain elevated for several months. In addition, two poorly-resolved drops in glass plate calcite Sr/Ca and Ba/Ca occur during 2011. It is of interest that these variations in dripwater trace element chemistry are not mirrored in the coeval glass plate data, and vice-versa. This perhaps reflects a sampling bias where discreet dripwater samples are not

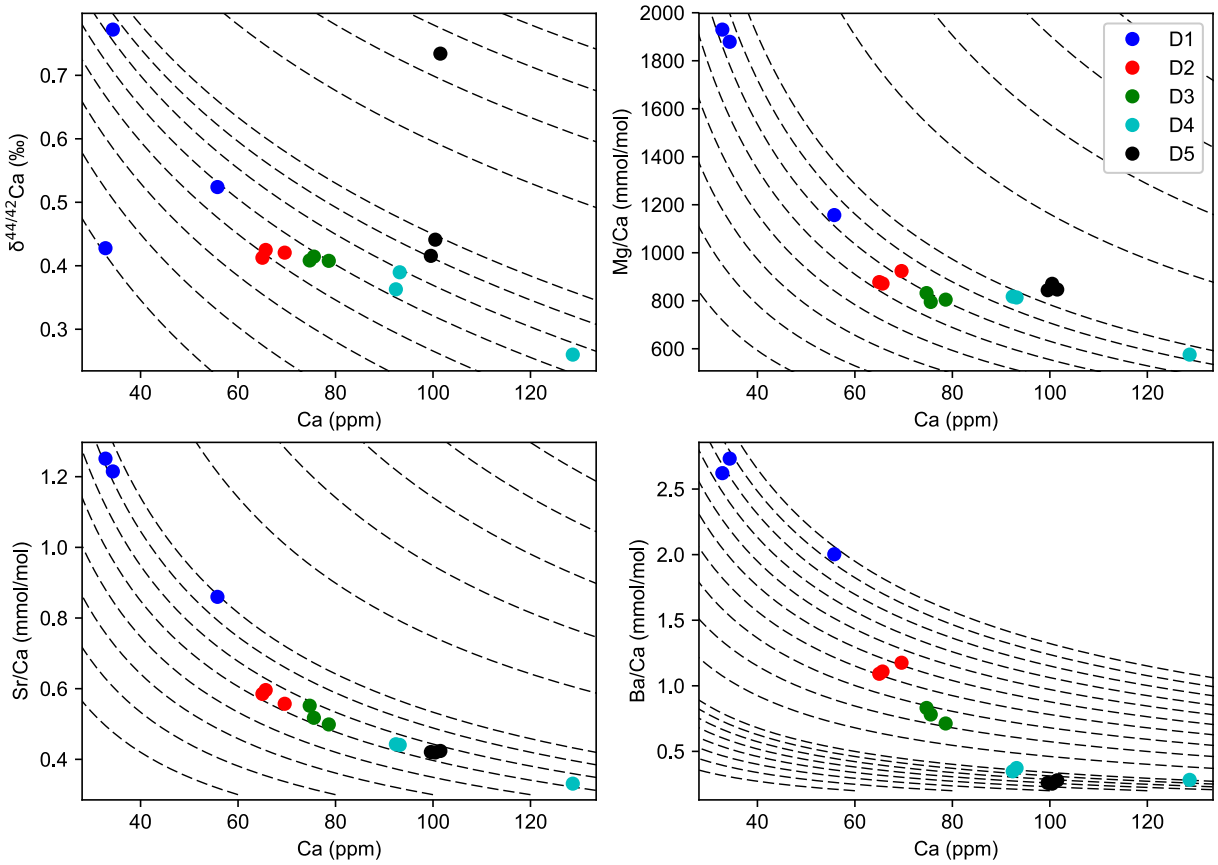



Figure 4.4: Dripwater monitoring data from Figure 4.2, replotted against Ca concentration. Also shown are dashed lines representing PCP solution evolution curves. Model lines are calculated as via Rayleigh distillation using the fractionation factors and partition coefficients given in the main text. Initial solution chemistry is unknown; each line represents a different initial solution composition.

representative of the entire month recorded by calcite growth. Alternatively, a particulate contribution to dripwater elemental budgets could complicate this simple partitioning model.

### 4.4.2 Drip Site Comparison

Studying multiple drip-sites within a cave allows us to better constrain local variations in hydrology and geochemistry. Such understanding aids in the selection of drip sites for palaeoclimate studies, and is particularly important when creating composite time-series from different stalagmites within a single cave.

Dripwater Mg/Ca, Sr/Ca and Ba/Ca data from D1-D5 show a consistent pattern whereby D1 has the highest ratio, followed by D2, D3, D4 then D5 (Figure 4.2). This trend is, within uncertainty, mirrored in the glass plate calcite trace element data. This

suggests a systematic trend as samples are taken from deeper within Heshang Cave: either there are increasingly dominant trace element sources (relative to Ca sourcing), or the dripwaters are undergoing more extensive prior calcite precipitation. Of the two, increasing PCP is perhaps more likely. Sites deeper within the cave require a longer flow path from soil to drip-site, a factor likely to encourage PCP. PCP would readily explain the co-variation between all three X/Ca ratios, and their negative correlations with dripwater Ca concentrations (Figure 4.4). Furthermore, the observations that site D1 is not actively precipitating calcite, has the lowest Ca concentrations of the drip-sites studied and exhibits the highest observed X/Ca ratios all suggest that it has undergone sufficient PCP to reach equilibrium with cave air $pCO_2$.

Figure 4.4 quantitatively tests these data against a simple Rayleigh distillation PCP model. Ca isotope and trace element data are plotted against Ca concentrations for each drip site. Grey dashed lines show modelled trends due to PCP. Model calculations use the partition coefficients determined here and assume an initial dripwater Ca concentration. Each line represents a calculation using a different initial dripwater isotopic composition or X/Ca ratio. Dripwater data points that lie along a given line may be explained as evolving from the same initial solution, varying only in the extent of PCP. Dripwater data points that do not lie along a common line may differ in initial solution composition.

Mg/Ca data broadly follow the expected PCP trend, but with minor variation between drip sites, likely due to differences initial solution Ca or Mg concentrations. Sr/Ca shows an excellent fit with the PCP models, with D2, D3, D4 and D5 falling on the same line. Data from D1 follow a PCP trend but are offset relative to other drip sites, suggesting a higher initial solution Sr/Ca ratio or Ca concentration. Finally, Ba/Ca correlates negatively with Ca concentration, but the correlation differs from that predicted by the PCP models. This is probably due to significant heterogeneities in Ba sourcing between drip sites.

Variations in dripwater Ca isotopes between drip sites do not follow the trends present in the trace element data (Figures 4.2 & 4.4). Many analyses fall close to 0.4 ‰, but significant offsets to heavier (∼0.8 ‰ at D1 and D5) and somewhat lighter (0.3 ‰ at D4)

compositions are also observed. Variations in dripwater Ca isotopes are not reflected in the corresponding glass plate data; this suggests that either that dripwater sampling (a spot measurement) is not representative of the entire month's dripwater, or that these variations in dripwater Ca isotopes are due to contributions from some Ca pool that is not available for fractionation into calcite precipitates. An obvious candidate would be a Ca-bearing particulate phase, a possibility discussed further in Section 4.4.3.

### 4.4.3 Filtering Experiments

Particulate matter transported by dripwater may affect speleothem growth kinetics by providing additional nucleation sites, and may become incorporated into speleothem material, affecting trace element and isotope chemistry. X/Ca ratios for site D3 show enrichment factors $< 1$ (Figure 4.3), suggesting a Ca-bearing solid phase containing little of these trace constituents. In contrast, W2 shows X/Ca enrichment factors $> 1$ for all elements other than P, indicative of significant trace metal contents in the particulate phase. P and P/Ca data show no systematic trend, with filtration enriching some samples while depleting others. This is likely an artefact of the filtration process and P data are not considered further.

Of the three speleothem-forming sites studied here (D2, D3 and D5), only D3 shows any geochemical evidence for a particulate phase. Unfiltered D3 dripwater is 56 % enriched in Ca relative to filtered dripwater with little-to-no enrichment in other metals, including Mg. This suggests a calcium carbonate particulate phase accounting for approximately 36 % of total dripwater Ca content.

It is possible that the observed carbonate particles in D3 dripwaters are the result of post-sampling $CO_2$ degassing and precipitation, however this is unlikely. D2, D3 and D5 (all speleothem-growing drip-sites) were collected in identical fashion and well-sealed against $CO_2$ degassing prior to filtration. If post-sampling calcite precipitation had occurred, it would be expected in all three samples. Instead, the particulate carbonate present in D3 dripwater is likely a natural signal.

This particulate carbonate may be sourced from the host rock (primary or secondary

calcite), or have precipitated directly from the dripwater. If originating from the host rock, particulate calcite could account for the non-dolomite bedrock source identified from Mg/Ca data (>18 % of dripwater Ca — Section 4.4.1.2). Such calcite could be entrained at any point along the drip flow path. Alternatively, the particulate calcite may be a precipitate from the host dripwater — the result of PCP. If entrained, such precipitates would not affect bulk dripwater chemistry. From the data presented here these two possibilities cannot be distinguished, although a host rock source would more readily explain the observed Mg/Ca ratios.

The impact of dripwater particulates on the bulk dripwater Ca isotope budget may be significant. Unfortunately, Ca isotope analyses were not performed as part of the filtration experiment. Nevertheless, a simple mass balance calculation allows us to estimate the composition of both aqueous and particulate Ca, assuming that the particulate Ca represents a secondary precipitate. Using bulk solution $\delta^{44/42}\mathrm{Ca} = 0.40\,‰$, a fractionation factor of $\Delta^{44/42}\mathrm{Ca}_{calcite-water} = -0.36\,‰$ and a particulate phase accounting for 36 % of total solution Ca, this calculation gives a dissolved $\delta^{44/42}\mathrm{Ca}$ of 0.53 ‰ and a particulate $\delta^{44/42}\mathrm{Ca}$ of 0.17 ‰. If correct, such an elevated aqueous $\delta^{44/42}\mathrm{Ca}$ would be clear evidence for PCP. Direct measurement of particulate or filtered aqueous $\delta^{44/42}\mathrm{Ca}$ would allow more robust determination of the true aqueous $\delta^{44/42}\mathrm{Ca}$.

This discussion highlights the importance of dripwater filtration prior to analysis of aqueous phases, particularly when determining solution Ca concentrations and isotopes. Coupled analysis of filtered and unfiltered dripwater may elucidate the nature of any dripwater particles and may provide information on processes occurring in the unsaturated zone.

The effects of a particulate contribution to speleothem chemistry are unclear. Carbonate particles in dripwater may act as additional nucleation points for carbonate precipitation, perhaps enhancing speleothem growth. Dripwater particles may also be incorporated during speleothem growth. Whether such particles could significantly influence bulk speleothem chemistry would depend on their composition and relative contribution to speleothem mass, factors that cannot be constrained with the available data for this

site.

### 4.4.4 Ca Sourcing at the HS4 dripsite

The similarity of Ca isotopes between 2011–2013 dripwater samples from the HS4 dripsite (D3) and the dolomite bedrock at first glance suggests that no PCP is occurring at HS4 drip-site. However, trace element and filtering results show this to be unlikely. Dripwater Mg/Ca results require a non-dolomite Ca source — most likely secondary calcite above the cave. Such a source would lead to dripwaters with initial $\delta^{44/42}$Ca lighter than the dolomite bedrock, requiring PCP to explain the observation of dripwaters at 0.40 ‰. Filtering of D3 dripwaters provides evidence for a significant calcium carbonate particulate contribution to the dripwater Ca budget, and calculations derived from this suggest an elevated dissolved Ca isotope composition relative to the bulk solution. These particles may represent entrained dripwater precipitates — direct evidence of PCP. Alternatively, they may sample a non-dolomite Ca source above the cave, supporting the Mg/Ca interpretation. Either case would be evidence for PCP occurring above the dripsite.

Chapter 3 concluded that PCP occurs in the modern environment based on data showing dripwater Ca isotopes to be elevated relative to dolomite bedrock. Dolomite bedrock was assumed to be the sole source of dripwater Ca. The work presented here expands upon Chapter 3, highlighting that the true Ca source represents a mixture between dolomite and calcite dissolution, rather than pure dolomite. Furthermore, this work demonstrates that calcite particulates contribute significantly to the dripwater Ca budget at HS4, impacting bulk Ca concentration and isotopes measurements on unfiltered dripwaters.

The mixture of calcite and dolomite sources is expected to have a lighter isotopic composition than dolomite bedrock alone, but cannot be quantified without Ca isotope analysis of the calcite source. This work supports the conclusion of Chapter 3 that PCP is occurring in the modern environment, and also highlights the importance of secondary Ca sources and particulate phases to dripwater and speleothem Ca isotopes. Further work is required to quantify these effects to improve upon the modern PCP reconstruction given in Chapter 3 and more accurately calibrate the Ca isotope PCP proxy.

## 4.5 Conclusions

Key findings relating to metal cycling at Heshang Cave are as follows:

1. Unfiltered dripwater $\delta^{44/42}\text{Ca} = 0.40\,‰$ and glass plate $\delta^{44/42}\text{Ca} = 0.03\,‰$ across the interval 2011–2013 at the HS4 drip-site. There is no evidence for seasonal controls on the Ca isotopic composition of dripwaters or glass plate calcite. This gives a revised fractionation factor of $\Delta^{44/42}\text{Ca}_{calcite-water} = -0.36\,‰$.

2. Dripwater trace elements at the HS4 drip-site suggest sources other than the dolomite bedrock contribute significantly to dripwater metal chemistry. In particular, lower-than-expected Mg/Ca ratios are best explained by dissolution of either primary or secondary calcite as a significant source of dripwater Ca. Secondary calcite deposits are the most likely candidate, and would predict an initial dripwater Ca isotopic composition lighter than expected from quantitative dolomite dissolution alone.

3. Ca isotopes, Mg/Ca, Sr/Ca and Ba/Ca vary greatly between drip-sites within Heshang Cave. X/Ca ratios and Ca concentrations suggest that sites deeper within the cave undergo more extensive $CO_2$ degassing and PCP due to the greater thickness of overlying rock and hence longer drip flow paths, however Ca isotopes show a more complex pattern.

4. Filtering of dripwaters reveals a significant particulate contribution to dripwater metal chemistry at the HS4 drip site (D3), but not at other sites in Heshang Cave. These particles are likely calcite and represent 36 % of total dripwater Ca, significantly affecting bulk solution Ca concentration and isotopes. Their impacts on speleothem growth and chemistry remain unknown and require further study.

More generally, this work highlights two previously unconsidered controls on the Ca isotope PCP proxy: secondary calcite in dolomite cave systems, and carbonate particulates in dripwater. Future work should take note of these factors and ensure they are well quantified for the study site in question. Dripwater filtration is recommended prior to Ca isotope analysis for PCP reconstruction as only the dissolved Ca pool is expected to fractionate during PCP.

# Chapter 5

# CaveCalc: A New Model for Speleothem Chemistry & Isotopes

## 5.1 Introduction

This chapter introduces a new geochemical model ('CaveCalc') developed to model dripwater and speleothem chemistry through a variety of soil, karst and cave processes. In particular, this work was motivated by the need for a single unified model for a variety of proxy systems (including $\delta^{18}O$, $\delta^{13}C$, Ca isotopes, radiocarbon and trace metals) to help interpret multi-proxy datasets, such as those discussed in Chapters 3 & 4.

This chapter focuses on the implementation details of CaveCalc, its capabilities and limitations (Section 5.2), as well as giving some example usage (Section 5.5). This provides background and documentation to other researchers interested in using CaveCalc. In Chapter 6, I apply CaveCalc to modelling the controls on dripwater and speleothem $\delta^{13}C$, a long-standing subject of interest in speleothem research.

CaveCalc is not alone in this field — several models have previously been developed to aid in speleothem proxy interpretation. Many focus on modelling single components of dripwater and/or speleothem chemistry, for example dripwater Ca concentrations (Baker et al., 2016), DIC stable isotope evolution (Hendy, 1971; Deines et al., 1974; Mühlinghaus et al., 2009; Scholz et al., 2009; Dreybrodt et al., 2016), radiocarbon content (Fohlmeister

et al., 2011; Noronha et al., 2015; Dreybrodt et al., 2016), trace element ratios (Fairchild et al., 2000; Sinclair, 2011; Stoll et al., 2012) and metal isotope ratios (Owen et al., 2016). This chapter unifies these different chemical systems into a single model framework to facilitate the use of multiple proxies to quantify past conditions. The relative strengths and weaknesses of CaveCalc compared to existing models are discussed in Section 5.6.

Installation instructions and model source code are given in Appendix A.

## 5.2 Model Description

The purpose of CaveCalc is to forward-model the evolution of soil water chemistry (and any calcite precipitated) as water moves downwards from the soil into the unsaturated zone overlying a cave, and finally emerges into a cave, potentially precipitating speleothems. Along this flow path many processes may modify dripwater chemistry, including $CO_2$ exchange with gas phases (e.g. atmosphere, soil gas, cave air), carbonate dissolution and carbonate precipitation reactions.

This section begins by introducing PHREEQC, the hydrochemical software used by CaveCalc to perform many calculations (Section 5.2.1). Section 5.2.2 then describes how CaveCalc is structured around, and makes use of, PHREEQC. Finally, Section 5.2.3 discusses the reactions available to the model user, and how these reactions can be used to describe a variety of processes of interest to speleothem scientists.

### 5.2.1 PHREEQC

PHREEQC is a computer program produced by the USGS for numerical modelling of aqueous chemistry (Parkhurst and Appelo, 1999). It is designed to calculate chemical equilibrium between solutions, minerals, gases and ion exchangers, and over time has been expanded to include processes such as kinetics, isotope fractionation and one-dimensional transport (Parkhurst and Appelo, 2012). CaveCalc uses PHREEQC to perform many isotope-enabled geochemical calculations including DIC speciation, calcite precipitation

and bedrock dissolution.

Use of PHREEQC requires two data sources. Firstly, PHREEQC must be provided with a database file that describes all the elements, ions, dissolved species, gases and minerals to be included in the model, in addition to all possible reactions between them. Furthermore, the database file includes all thermodynamic data necessary to calculate equilibrium. An example of a very simple database is given in Figure 5.1. Secondly, PHREEQC requires an input that describes one or more solutions, and a sequence of reactions to perform on the solutions. PHREEQC uses the data provided in the database file to calculate the equilibrium solution composition at each reaction step.

This input may be provided as a text file (see the example in Figure 5.2), or generated dynamically using another program (e.g. using IPhreeqc; Charlton and Parkhurst, 2011). This latter approach is employed by CaveCalc and enjoys two main advantages over directly writing PHREEQC input files. Firstly, it allows for conditional programming, permitting the reaction sequence to depend on the state of the solution at a given step. Secondly, external programming allows the user to perform certain reactions that are difficult or impossible to implement directly in PHREEQC. Examples of this are discussed in Section 5.2.3.

### 5.2.2 Structure & Implementation Details

CaveCalc is a numerical model written in Python that interacts with PHREEQC using the IPhreeqc interface (Charlton and Parkhurst, 2011). This approach leverages the computational power of PHREEQC, while providing a flexible, cross-platform wrapper with additional functionality tailored to the specific requirements of cave geochemical modelling. The source code is structured as a Python 3 module. The full code and installation instructions are given in Appendix A.

CaveCalc uses a custom PHREEQC database ('oxotope.dat') to define the reactions necessary to describe a cave system. This database file is a modified version of 'iso.dat', a standard isotope database distributed with PHREEQC 3.0. Modifications made include the addition of Ca isotope systematics for aqueous ions and calcite, and a complete update

Figure 5.1: A minimalist database for PHREEQC, including water and hydrochloric acid chemistry only. The SOLUTION_MASTER_SPECIES block defines the 'master species' present in the system — elements, oxidation states and electrons. The SOLUTION_SPECIES block defines all aqueous reactions between species, and all molecules formed by the combination of the master species. Finally, the PHASES data block defines gaseous oxygen, hydrogen and water, and describes their solubilities. The actual database used by CaveCalc ('oxotope.dat') is considerably more complex, and is given in Appendix A. The syntax used is fully described in Parkhurst and Appelo (2012). Data given are taken from the databases distributed with PHREEQC.

```
# example_database.dat

SOLUTION_MASTER_SPECIES
H         H+              -1.      H                1.008
H(0)      H2              0.0      H
H(1)      H+              -1.      0.0
O         H2O             0.0      O                16.00
O(0)      O2              0.0      O
O(-2)     H2O             0.0      0.0
O(-2)     H2O             0.0      0.0
E         e-              0.0      0.0              0.0
Cl        Cl-             0.0      Cl               35.453


SOLUTION_SPECIES

H+ = H+
        log_k             0.000
        -gamma            9.0000   0.0000

e- = e-
        log_k             0.000

H2O = H2O
        log_k             0.000

H2O = OH- + H+
        log_k             -14.000
        delta_h           13.362   kcal
        -analytic         -283.971   -0.05069842   13323.0   102.24447   -1119669
        -gamma            3.5000   0.0000

2 H2O = O2 + 4 H+ + 4 e-
        log_k             -86.08
        delta_h           134.79   kcal

2 H+ + 2 e- = H2
        log_k             -3.15
        delta_h           -1.759   kcal

Cl- = Cl-
        log_k             0.000
        -gamma            3.5000   0.0150

H+ + Cl- = HCl
     log_k                -0.710
     delta_h              -12.298   kJ


PHASES
O2(g)
        O2 = O2
        log_k             -2.8983
        -analytic         -7.50e+0   7.90e-3   0.00e+0   0.00e+0   2.00e+5

H2(g)
        H2 = H2
        log_k             -3.150
        delta_h           -1.759   kcal

H2O(g)
        H2O = H2O
        log_k             1.51
        delta_h           -44.03   kJ

HCl(g)
    HCl = H+ + Cl-
         log_k      6.290
         delta_h   -74.770     kJ
```

Figure 5.2: Top: A simple example of a PHREEQC input script using the example database (Figure 5.1). The SOLUTION keyword defines a new solution with no components other than water (by default, 1 kg). The REACTION keyword adds 0.1 mol of HCl molecules to the solution. Bottom: Part of the PHREEQC output file generated, showing the calculated solution chemistry after adding 0.1 mol HCl to 1 kg $H_2O$ at 20 °C.

```
# example_input.phr

DATABASE example_database.dat

SOLUTION Pure Water
    temp              20
    pH                7 charge

REACTION
    HCl     0.1
END
```

```
------------------------------Description of solution------------------------------

                                    pH  =   1.088      Charge balance
                                    pe  =  17.145      Adjusted to redox equilibrium
                       Activity of water  =   0.997
                  Ionic strength (mol/kgw)  =   9.872e-02
                        Mass of water (kg)  =   1.000e+00
                  Total alkalinity (eq/kg)  =  -1.000e-01
                     Total carbon (mol/kg)  =   0.000e+00
                        Total CO2 (mol/kg)  =   0.000e+00
                          Temperature (ºC)  =  20.00
                   Electrical balance (eq)  =   3.077e-16
 Percent error, 100*(Cat-|An|)/(Cat+|An|)  =   0.00
                                Iterations  =  14
                                   Total H  = 1.111124e+02
                                   Total O  = 5.550622e+01

----------------------------------Distribution of species----------------------------------

                                                 Log        Log        Log     mole V
   Species          Molality    Activity  Molality  Activity      Gamma   cmş/mol

   H+              9.872e-02   8.164e-02    -1.006    -1.088     -0.082      0.00
   OH-             1.085e-13   8.288e-14   -12.965   -13.082     -0.117       (0)
   H2O             5.551e+01   9.966e-01     1.744    -0.001      0.000     18.05
Cl              1.000e-01
   Cl-             9.872e-02   7.569e-02    -1.006    -1.121     -0.115       (0)
   HCl             1.282e-03   1.311e-03    -2.892    -2.882      0.010       (0)
H(0)            4.986e-40
   H2              2.493e-40   2.550e-40   -39.603   -39.593      0.010       (0)
O(0)            2.844e-15
   O2              1.422e-15   1.454e-15   -14.847   -14.837      0.010       (0)

-------------------------------Saturation indices-------------------------------

  Phase               SI** log IAP   log K(293 K,   1 atm)

  H2(g)            -36.47    -39.59    -3.13  H2
  H2O(g)            -1.64     -0.00     1.64  H2O
  HCl(g)            -8.72     -2.21     6.51  HCl
  O2(g)            -11.98    -14.84    -2.86  O2

**For a gas, SI = log10(fugacity). Fugacity = pressure * phi / 1 atm.
  For ideal gases, phi = 1.

-------------------
End of simulation.
-------------------
```

of the equilibrium C & O isotope fractionation factors associated with $CO_2$–DIC–calcite system, based on the calibration data of O'Neil and Adami (1969), Romanek et al. (1992), Zhang et al. (1995), Beck et al. (2005) and Coplen (2007). Oxotope.dat also includes reactions for the partitioning of Mg, Sr and Ba into calcite, using the temperature-dependent

partition coefficients of Day and Henderson (2013). The full details of all reactions implemented in oxotope.dat are too exhaustive to list here — the reader is referred to the database file itself for a complete description. Modifications made to the standard iso.dat database are clearly noted in Appendix A.

The database may be expanded as required to include new chemical species, reactions or isotope systems. It may even be replaced by another database entirely, provided that the replacement database includes the species and reactions used internally by CaveCalc. However, in general, the user does not need to directly access or modify the underlying PHREEQC database. CaveCalc provides a wrapper around PHREEQC, such that the user is not required to write PHREEQC input scripts (such as Figure 5.2). This is done by providing a set of pre-defined reactions and inputs, discussed in Sections 5.2.3 & 5.3.

CaveCalc may be run in two ways; either via Python scripts, or by using the built-in GUI. The GUI allows users to quickly run forward models simulating a variety of cave conditions and requires no programming knowledge to operate. Using CaveCalc via Python scripts exposes the full model functionality (the API) and allows a greater degree of flexibility as to which reactions are performed and in what order.

### 5.2.3 Supported Reactions

CaveCalc works by defining an initial solution chemistry, then modifying it through a series of chemical reactions until a desired end-point is reached. The order, type and nature of these reactions may be specified by the user to simulate a variety of soil, karst & cave processes. For example, a very simple dripwater evolution path could be described as follows:

$$\text{Rain Water} \xrightarrow[\text{Respiration}]{\text{Soil}} \text{Soil Water} \xrightarrow[\text{Dissolution}]{\text{Bedrock}} \text{Dripwater} \xrightarrow[\text{Precipitation}]{\text{Calcite}} \text{Speleothem}$$

where the labelled arrows represent reactions, and the names (e.g. Soil Water) represent states of the solution and/or calcite precipitate between reactions. This captures the essence of how CaveCalc works; an initial solution is defined and acted upon, in sequence, by a variety of chemical reactions. The state of the system is recorded after every reaction,

allowing users to visualise how dripwater and carbonate chemistry evolve as the model progresses.

CaveCalc can model a set of chemical reactions that, when linked together, simulate various aspects of the soil-karst-cave system. CaveCalc may be expanded to include new reactions and isotope systems as needed. The chemical reactions available are:

- $CO_2$ equilibration between DIC and $CO_{2(g)}$
- Calcite dissolution
- Dolomite dissolution
- Pyrite oxidation
- Calcite precipitation
- Equilibrium $CO_2$ degassing (equilibrium between DIC species, $H_2O$ and gaseous $CO_2$)
- Kinetic $CO_2$ degassing (without equilibrium between DIC species or $H_2O$)

Internally, each of these reactions generates a block of PHREEQC input that is executed by IPhreeqc. It is also possible for users to write their own PHREEQC input that specifies any possible reaction and use this as part of CaveCalc; however, this topic is beyond the scope of this chapter — interested readers should consult the source code and examples in Appendix A.

These basic reactions may be combined or run in sequence to simulate a variety of soil and karst processes of interest to speleothem researchers, including some examples given below. Through all these reactions, full solution chemistry is tracked (including DIC speciation, pH & alkalinity), along with C, O and Ca isotopes. Mg, Sr and Ba partitioning into calcite are also included. The full set of thermodynamic data used by CaveCalc is written into oxotope.dat (Appendix A.3.11).

**Equilibration of soil gas and soil water** Most measurements of soil conditions (e.g. $pCO_2$, $\delta^{13}C$, radiocarbon) relate to soil gas as it is easier to sample than any liquid phase present. However, it is soil water chemistry that is the relevant starting point for dripwater evolution. Given gas phase data, CaveCalc can calculate the chemical and isotopic composition of the solution at equilibrium with it. This is used as the

initial solution composition for most CaveCalc model runs.

**Bedrock dissolution (in the presence of a gas phase)** CaveCalc can calculate the product of dissolving a carbonate bedrock (whose chemistry and isotopes may be specified by the user) in any solution (e.g. soil water) using the calcite or dolomite dissolution reactions. This simulates closed-system dissolution. Furthermore, this may be coupled to a gas-DIC $CO_2$ equilibration reaction to simulate open-system dissolution. The volume, chemistry and isotopes of this gas may be specified by the user, allowing complete control over the degree of open-system dissolution.

**Pyrite-oxidation-driven bedrock dissolution** In some caves, bedrock dissolution may be driven by pyrite oxidation, rather than respiration of organic matter (e.g. Spötl et al., 2016). The CaveCalc database includes all aqueous Fe and S chemistry necessary to model this process. CaveCalc allows users to specify the amount (if any) of pyrite present in bedrock, and the oxygen concentration of soil water, to investigate the effects this may have on bedrock dissolution.

**Incongruent carbonate bedrock dissolution** Incongruent dissolution may be modelled by coupling a dissolution reaction (e.g. dolomite) with a synchronous precipitation reaction (e.g. calcite). CaveCalc will give the final, equilibrium solution chemistry. Such processes are often inferred in dolomite cave systems and may affect dripwater trace element signatures.

**Water-rock interactions** Similarly to incongruent dissolution, any water-rock interaction may be modelled by allowing a previously calculated solution to equilibrate with a user-specified calcite phase. This may be modelled as one-way (dissolution only), or allow simultaneous re-precipitation of an equilibrium phase.

**$CO_2$ degassing and PCP** $CO_2$ degassing and PCP are important processes that may modify dripwater chemistry and isotopes prior to calcite growth. CaveCalc can model $CO_2$ degassing from dripwater, and the subsequent $CaCO_3$ precipitation, in a number of ways. For example, the solution may be allowed to fully equilibrate with a specified gas phase (e.g. cave air) in a single model step. Alternatively, $CO_2$ may be incrementally removed from solution in a one-way kinetic process to more finely

resolve a degassing evolution pathway. The user may allow calcite precipitation to occur during or after $CO_2$ degassing, not at all, or only once a specified saturation index is reached. There are many ways to parametrise the progressive loss of $CO_2$ and $CaCO_3$ from dripwater — CaveCalc provides flexible options for doing so. Note that during calcite precipitation reactions, CaveCalc does not distinguish between PCP calcite and speleothem calcite as they are precipitated by the same chemical mechanism; the distinction is purely operational. It is up to the user to decide how the two are best divided, depending on the specific nature of the cave site they are interested in.

The model inputs used to fully define these reactions are discussed in Section 5.3, and model output is discussed in Section 5.4. Other operational details are given in Appendix A.

There are a number of processes that CaveCalc does not presently seek to mimic. These include water mass mixing and the effects on trace element and isotopes generated by the rate of mineral growth. CaveCalc may be expanded to include new reactions, or additional isotope systems, as needed in the future.

## 5.3 Model Inputs

Section 5.2.3 discussed how a CaveCalc model run is structured: an initial solution is defined, then modified through a series of chemical reactions until a desired end-point is reached. This section discusses the geochemical inputs available to determine how these reactions are performed, and in what sequence.

CaveCalc also has a range of input options that control model scripting and the handling of model output. These operational details are not of direct scientific interest — they are discussed briefly in Appendix A.

Most geochemical inputs are self-explanatory. Note that radiocarbon inputs ('Atmospheric R14C' etc.) are age- and $\delta^{13}C$- corrected values expressed in percent modern carbon (Section 1.2.2). CaveCalc radiocarbon outputs are expressed in the same units.

Table 5.1: CaveCalc inputs for specifying soil carbon & oxygen chemistry. 'Parameter' gives the user-friendly name of the input, while 'Model Name' gives the name used internally by CaveCalc. These inputs are discussed in Section 5.3.1.

| Parameter | Model Name | Default Value |
|---|---|---|
| Atmospheric Gas End-member | | |
| Atmospheric O2 (%) | atm_O2 | 0.21 |
| Atmospheric pCO2 (ppmv) | atm_pCO2 | 270 |
| Atmospheric d13C (per mil) | atm_d13C | -7 |
| Atmospheric R14C (pmc) | atm_R14C | 100 |
| Soil Gas End-member | | |
| Soil Gas O2 (%) | soil_O2 | 0 |
| Soil R14C (pmc) | soil_R14C | 100 |
| Soil Gas d13C (per mil) | soil_d13C | -25 |
| Soil Gas pCO2 (ppmv) | soil_pCO2 | 20000 |
| Initial Gas Chemistry | | |
| Atmospheric Gas Exchange (0-1) | atmo_exchange | 0 |
| Initial O2 (%) | init_O2 | 'mix' |
| Initial R14C (pmc) | init_R14C | 'mix' |
| Initial d13C (per mil) | init_d13C | 'mix' |
| Initial pCO2 (ppmv) | init_pCO2 | 'mix' |

### 5.3.1 Soil Gas & Water

CaveCalc models begin by calculating soil water chemistry. This is used as the starting point for a series of reactions defined either manually by the user, or specified by the Run Mode (discussed in Section 5.3.6). To calculate soil water chemistry, the user gives inputs describing the equilibrium gas. This method is preferable as soil gas is more readily measured than soil water. CaveCalc calculates the equilibrium aqueous solution. Inputs for defining soil gas chemistry are summarised in Table 5.1. CaveCalc provides two methods for defining soil gas:

1. Explicitly define the initial gas chemistry ($O_2$, $pCO_2$, $\delta^{13}C$, $a^{14}C$)
2. Express initial gas chemistry as a mixing between two end-members (e.g. atmospheric gas and soil gas)

For input Method 1, enter numeric values for the relevant initial gas chemistry inputs (e.g. 'Initial d13C' and 'Initial pCO2'). This will disregard the values entered for the soil gas and atmospheric end-members. For input Method 2, set these parameters to 'mix'

Table 5.2: CaveCalc inputs for specifying non-bedrock metal sources. 'Parameter' gives the user-friendly name of the input, while 'Model Name' gives the name used internally by CaveCalc. These inputs are discussed in Section 5.3.2.

| Parameter | Model Name | Default Value |
|---|---|---|
| Soil Ba (mmol/kgw) | soil_Ba | 0 |
| Soil Ca (mmol/kgw) | soil_Ca | 0 |
| Soil Mg (mmol/kgw) | soil_Mg | 0 |
| Soil Sr (mmol/kgw) | soil_Sr | 0 |
| Soil d44Ca (per mil) | soil_d44Ca | 0 |

(their default value), and use the 'Atmospheric Exchange' input to specify the volumetric fraction of atmospheric gas end-member present in the mixture — the remainder is defined by the soil gas end-member. This method is provided to allow users to easily simulate the co-variation in solution $pCO_2$, $\delta^{13}C$, radiocarbon and oxygen content expected due to mixing of two gases (e.g. a deep carbon-rich soil gas and the atmosphere). Note that the terms 'atmospheric gas end-member' and 'soil gas end-member' are simply convenient labels; the end-member compositions may be used to simulate mixing between any two gas reservoirs.

### 5.3.2 Soil & Aerosol Metals

While most metals present in cave dripwater likely originate from the host bedrock, some may have arrived in the overlying soil via other means. Metals may, for example, be transported to the cave site as aerosols, or be derived from biological materials or dissolution of silicates. To account for non-bedrock metal sources, CaveCalc allows users to define an initial soil water Ba, Ca and Mg contents (Table 5.2). Any metals specified in this way are treated as chloride salts for charge balance purposes. If Ca is used, the user may also specify its isotopic composition (Table 5.2).

The presence of chloride salts in soil water prior to bedrock dissolution may affect the extent of bedrock dissolution, and the resulting dripwater chemistry. It may also be of interest to use CaveCalc to calculate how aerosol signatures may become diluted as bedrock dissolves.

Table 5.3: CaveCalc inputs for specifying bedrock chemistry and the conditions under which it dissolves. 'Parameter' gives the user-friendly name of the input, while 'Model Name' gives the name used internally by Cavecalc. These inputs are discussed in Section 5.3.3.

| Parameter | Model Name | Default Value |
|---|---|---|
| Bedrock lithology | bedrock_mineral | 'Calcite' |
| Bedrock Mg/Ca (mmol/mol) | bedrock_MgCa | 0 |
| Bedrock Sr/Ca (mmol/mol) | bedrock_SrCa | 0 |
| Bedrock Ba/Ca (mmol/mol) | bedrock_BaCa | 0 |
| Bedrock d13C (per mil) | bedrock_d13C | 0 |
| Bedrock d18O (per mil) | bedrock_d18O | 0 |
| Bedrock d44Ca (per mil) | bedrock_d44Ca | 0 |
| Bedrock Pyrite (moles) | bedrock_pyrite | 0 |
| Gas Volume (L) | gas_volume | 10 |
| Allow Calcite Reprecipitation | reprecip | FALSE |

### 5.3.3 Bedrock Dissolution

Bedrock dissolution occurs when soil water (defined by the inputs described in Sections 5.3.1 & 5.3.2) interacts with a carbonate bedrock. The inputs available to describe the nature of this process are given in Table 5.3.

Carbonate bedrock may be specified as either calcite or dolomite ('Bedrock lithology'). In addition, the user may define bedrock Mg/Ca, Sr/Ca and Ba/Ca ratios, and the isotopic compositions of carbon, oxygen and calcium. It is assumed that bedrock contains no radiocarbon. Furthermore, the 'Bedrock Mg/Ca (mmol/mol)' input is ignored if 'Bedrock lithology' is set to 'Dolomite'.

CaveCalc performs bedrock dissolution calculations by dissolving the carbonate bedrock phase in soil water until the solution reaches saturation with respect to calcite. This process may be modified using three additional inputs:

**Bedrock Pyrite (moles)** This input specifies the moles of pyrite present during bedrock dissolution. Note that this does not necessarily mean the pyrite will dissolve — the extent of oxidative pyrite dissolution depends strongly on the oxygen concentration in soil water. The input only specifies the total pyrite available for dissolution.

**Gas Volume (L)** This input specifies the volume of soil gas present for $CO_2$ equilibration during bedrock dissolution. This allows users to specify the degree of open-system

Table 5.4: CaveCalc inputs for defining cave air chemistry. 'Parameter' gives the user-friendly name of the input, while 'Model Name' gives the name used internally by Cavecalc. These inputs are discussed in Section 5.3.4.

| Parameter | Model Name | Default Value |
|---|---|---|
| Cave Air O2 (%) | cave_O2 | 0.21 |
| Cave Air pCO2 (ppmv) | cave_pCO2 | 1000 |
| Cave Air d13C (per mil) | cave_d13C | -10 |
| Cave Air R14C (pmc) | cave_R14C | 100 |
| Cave Air d18O (per mil) | cave_d18O | 0 |
| Cave Air Volume (L) | cave_air_volume | 0 |

dissolution occurring. The input is specified in litres of gas, with zero reflecting fully closed-system conditions and large positive numbers reflecting fully open-system conditions. CaveCalc reactions assume a solution mass of 1 kg, so this input may be considered as a volumetric ratio between gas and solution.

**Allow Calcite Reprecipitation** This input, if set to TRUE, allows calcite to precipitate during bedrock dissolution in chemical and isotopic equilibrium with the solution. This may be used to simulate extensive incongruent dissolution in dolomite cave systems. However, finer control on this process may be achieved in other ways (see Example 4 in Appendix A).

### 5.3.4 Cave Air

Cave air chemistry is relevant to dripwater chemistry and speleothem growth for several reasons. Firstly, the difference in $pCO_2$ between dripwater emerging into a cave and the cave air itself determines the extent of $CO_2$ degassing, and hence the extent of calcite precipitation. Secondly, cave air $CO_2$ may exchange with dripwater DIC, typically driving dripwater $\delta^{13}C$ to heavier values (Frisia et al., 2011).

CaveCalc allows the user to specify cave air chemistry and isotopes using the inputs in Table 5.4. 'Cave Air pCO2' is used to determine the end-point of models in most Run Modes (Section 5.3.6); dripwater evolution is complete once it reaches equilibrium with cave air $pCO_2$. The full cave air chemistry ($pCO_2$, $O_2$, $\delta^{13}C$, $\delta^{18}O$ and $a^{14}C$) are used for any equilibration reactions between dripwater and cave air. This functionality

Table 5.5: Other CaveCalc inputs. 'Parameter' gives the user-friendly name of the input, while 'Model Name' gives the name used internally by Cavecalc. These inputs are discussed in Section 5.3.5.

| Parameter | Model Name | Default Value |
|---|---|---|
| Temperature (Degrees C) | temperature | 25 |
| Rainfall d18O (per mil) | atm_d18O | -10 |
| Run Mode | kinetics_mode | |
| CO2(g) removal per step (fraction) | co2_decrement | 0.5 |
| Calcite supersaturation limit (SI) | calcite_sat_limit | 5 |

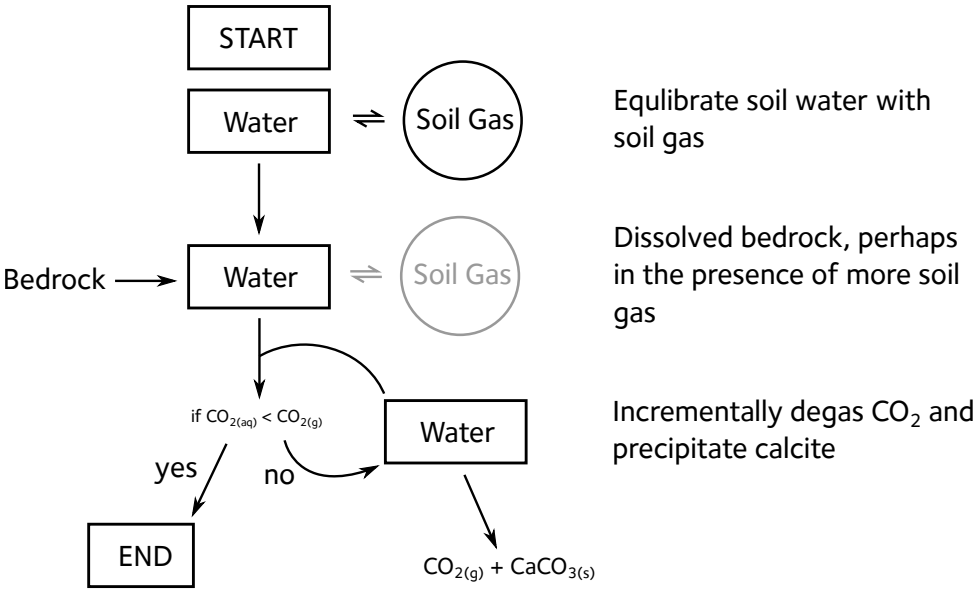
is used in the 'open_system_single_step' Run Mode, and may also be accessed in any user-written models. 'Cave Air Volume' sets the initial volume of cave air present during such reactions; this parametrises the extent of equilibration.

### 5.3.5 General

Model inputs unrelated to soil chemistry or bedrock dissolution are given in Table 5.5. 'Temperature (Degrees C)' specifies the temperature of the system; soil, bedrock and cave are considered to be at the same temperature. The affects equilibrium constants, partition coefficients and isotope fractionation factors as defined in the 'oxotope.dat' database file. Rainfall $\delta^{18}O$ specifies the oxygen isotope ratio of soil water.

The remaining inputs given in Table 5.5 are not geochemical, but relate to the scripting of model runs. The $CO_2$ decrement input specifies what fraction of aqueous $CO_2$ will degas when a kinetic degassing reaction step is performed, for example during a $CO_2$ degassing & PCP calculation. This is primarily used to control model resolution during these calculations; a large value (e.g. 0.5) will result in faster model computation and is suitable for most situations; smaller values will increase the resolution of $CO_2$ degassing and PCP calculations at the cost of increased model run time. 'Calcite supersaturation limit' is an input used only by the 'ss' Run Mode (see Section 5.3.6) to define the saturation index at which calcite may precipitate. 'Cave Air $pCO_2$' defines the model end-point where $CO_2$ degassing reactions will no longer occur.

Figure 5.3: A schematic of a typical CaveCalc model run. The order of reactions is controlled by the Run Mode (see Section 5.3.6), or may be written explicitly by the user. This example illustrates the 'closed_system_rayleigh' Run Mode.




### 5.3.6 Run Modes

The input 'Run Mode' determines the sequence of reactions used in the model and comes with several pre-defined options, described below. These options may simulate various aspects of the soil-karst-cave system. CaveCalc is not limited to these modes; the user may write their own Run Modes using any combination of the available reactions (Section 5.2.3), examples of which are given in Appendix A. However, use of a Run Mode is convenient for scripting simple model runs and is essential if operating CaveCalc using the GUI.

All provided Run Modes begin by calculating soil water chemistry. This is followed by bedrock dissolution. After this point, Run Modes diverge in the way they represent $CO_2$ degassing and calcite precipitation. This serves to provide simulations of different kinetic environments, and also providing less computationally-intensive options where only certain processes are of interest. Available Run Modes are described below.

**closed_system_rayleigh** This is the default mode. The dripwater chemistry resulting from bedrock dissolution repeatedly undergoes $CO_2$ degassing (kinetic) & calcite precipitation reactions until the solution reaches equilibrium with 'Cave pCO2' — effectively a Rayleigh distillation process. Through this degassing loop, equilibrium

is maintained with respect to calcite. This is shown schematically in Figure 5.3. This Run Mode does not allow $CO_2$ isotope equilibration between dripwater and cave air — degassing is assumed to be unidirectional. This mode represents rapid $CO_2$ degassing and calcite precipitation during PCP and speleothem growth.

**open_system_single_step** In this mode $CO_2$ degassing (equilibrium) and calcite precipitation are calculated in a single open-system step, where the solution is allowed to simultaneously reach equilibrium with cave air and precipitated calcite. The extent of equilibration with cave air is set by the 'Cave Air Volume' input (Table 5.4). This Mode is representative of slow precipitation and degassing environments, where near-equilibrium conditions are expected (e.g. cave pools).

**allow_supersaturation** This mode is similar to 'closed_system_rayleigh' with the exception that calcite is only allowed to precipitate once repeated $CO_2$ degassing (kinetic) reactions have driven the dripwater to a defined calcite saturation index. When precipitation occurs, calcite saturation is restored ($\Omega = 1$) and $CO_2$ degassing continues. This process repeats until dripwater reaches equilibrium with cave air $pCO_2$. The calcite saturation index at which precipitation will occur is given by the input 'Calcite Supersaturation Limit' (Table 5.5). This mode is useful for simulating situations where $CO_2$ degassing is rapid, and calcite nucleation and/or growth rates are too slow to maintain calcite saturation.

**allow_supersaturation_max** This model is identical to 'allow_supersaturation' with the exception that 'Calcite Supersaturation Limit' is not used. Instead $CO_2$ degasses until equilibrium is reached with cave $pCO_2$, at which point calcite precipitates. This process repeats until no more degassing occurs. Like 'allow_supersaturation' mode, this may be useful for modelling environments where calcite growth is limited by kinetic factors.

**degas_only** Following bedrock dissolution, dripwaters are only allowed to undergo $CO_2$ degassing — no calcite precipitation occurs. This simulates environments where calcite precipitation is impossible or extremely limited.

**diss_only** Using this mode, CaveCalc does not perform any further calculations af-

Figure 5.4: Python script to run the example models described in Section 5.5, and plot some model output. Note that only non-default settings need to be defined. By including both 'Calcite' and 'Dolomite' in the 'bedrock_mineral' input, CaveCalc will automatically run two models, one for each unique combination of input parameters.

```
from cavecalc.forward_models import ForwardModels
import cavecalc.analyse as cca

# define non-default settings to be used
s =  {  'bedrock_mineral' :      ['Calcite', 'Dolomite'],
        'cave_pCO2' :            500,
        'temperature'    :       15,             # degrees C
        'soil_pCO2' :            15000,          # ppmv gas equivalent
        'soil_d13C' :            -20,            # per mil, VPDB
        'bedrock_MgCa'   :       5,              # mmol/mol
        'kinetics_mode' :        'closed_system_rayleigh'
     }

# Run Models
p = ForwardModels(  settings = s )  # initialise the model
p.run_models()                      # run the model using the Run Mode given
p.save()                            # save model output to current directory

# Load Output Data and make plots
e = cca.Evaluate()       # initialise an 'Evaluate' object
e.load_data()            # load data from the current directory

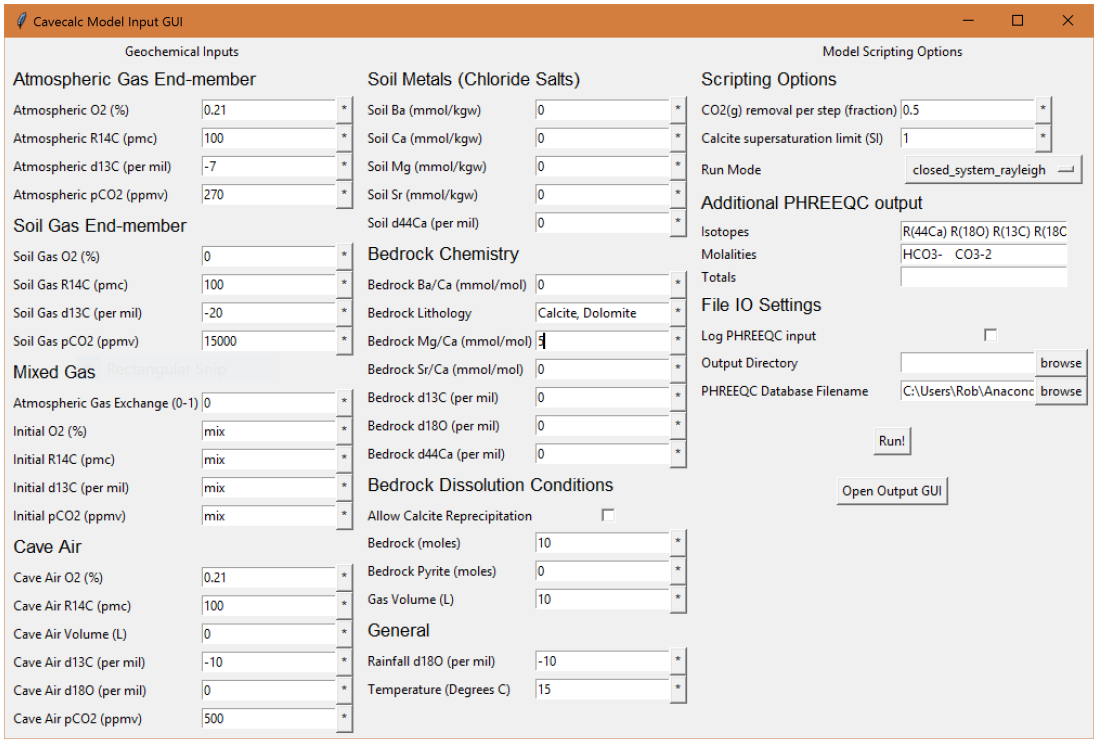
e2 = e.filter_by_index(0, n=True) # filter out the first model step
e2.plot_models(x_key='Ca(mol/kgw)', y_key='pH', label_with = 'bedrock_mineral')
e2.plot_models(x_key='Ca(mol/kgw)', y_key='d13C', label_with = 'bedrock_mineral')
cca.plt.show()
```

ter the bedrock dissolution reaction. This saves computation time when only the bedrock dissolution product is of interest (e.g. when modelling calcite growth at the speleothem apex where it is known that PCP does not occur).

## 5.4 Model Output

After each reaction step, the chemistry and isotopic composition of the resulting solution (and, if present, any calcite precipitated) are recorded. CaveCalc saves input settings and model results to file when a model run is complete. CaveCalc includes tools to quickly visualise model output, as well the facility to convert model outputs to formats readily useable with Python, MATLAB, or spreadsheet programmes (e.g. Excel). Use of these tools is demonstrated in Appendix A.

Figure 5.5: GUI input to run the example models described in Section 5.5. When opened, the GUI is automatically populated with default values. This window has been set with the same settings as the script shown in Figure 5.4 — both will produce the same output. As with the script example, including multiple values for a single input (in this case, 'Bedrock Lithology') causes CaveCalc to run a separate model for each unique combination of input parameters.




## 5.5 An Example

A schematic view of a CaveCalc model run is shown in Figure 5.3. Initially a soil water chemistry is defined in equilibrium with a soil gas phase. This step sets the initial solution carbon budget, isotope ratios and any non-bedrock-derived metals. In the second reaction step, the soil water dissolves a carbonate bedrock until saturation is reached with respect to calcite. This dissolution may take place in the presence of a variable volume of soil gas (determining the degree of open-system dissolution). The final step is an iterative loop where the solution incrementally degasses carbon dioxide and precipitates calcite until an ambient $pCO_2$ is reached (i.e. cave air $pCO_2$). This simulates PCP and speleothem growth. The schematic shown in Figure 5.3 corresponds to the 'closed_system_rayleigh' Run Mode (Section 5.3.6); other built-in modes are available and users may write their own (see Appendix A for more details).

To demonstrate working with CaveCalc this section walks through a simple example, comparing dripwater chemistry modelled for two sites that differ only in bedrock min-

eralogy (one calcite, one dolomite). This could, for example, represent two drip-sites in the same cave, one of which is overlain by a dolomite member in the host bedrock, while the other is overlain by limestone. The resulting difference in dripwater and speleothem Mg/Ca is obvious, however little attention has been paid to the impact of such a bedrock mineralogy change on other aspects of dripwater chemistry. This example shows how CaveCalc may be applied to a problem of this nature. Note that the purpose of this example is to briefly demonstrate CaveCalc usage, and the form of output provided; it is not intended to provide a realistic simulation of any particular cave system!

To run this example, all other input parameters (e.g. soil gas chemistry, bedrock isotopes, cave air) are kept constant. These models may be run either via the GUI or written into a simple Python script. These input methods are shown in Figures 5.5 and 5.6, respectively. The models are defined with some reasonable input parameters: cave $pCO_2 = 500\,\text{ppmv}$, temperature $= 15\,°\text{C}$, soil $pCO_2 = 15\,000\,\text{ppmv}$, soil $\delta^{13}C = -20\,‰$ and bedrock Mg/Ca $= 5\,\text{mmol}\,\text{mol}^{-1}$. The Run Mode (or 'kinetics_mode') is set to 'closed_system_rayleigh'. Bedrock mineral is set to both calcite and dolomite. When two or more different inputs are specified, CaveCalc will run a model for each unique combination of parameters. This feature is very useful when the user requires a suite of models that vary one or more parameters across a wide range — they may all be set up at once to run in sequence. All input parameters left unspecified are automatically set to their default values.

Both methods will give the same results. When run, CaveCalc will save two files in the specified output directory (the current directory, in this case): settings.pkl and results.pkl. These files store the settings used for the model runs and the results generated, respectively.

These results may be handled in a number of ways, including converted to .mat and .csv files (detailed in Appendix A). For this example, some plots are shown to visualise the modelled evolution of dripwater through $CO_2$ degassing and PCP, and how those results vary between calcite and dolomite bedrock (Figure 5.6). These plots were generated using the code in Figure 5.4, but could also have been produced using the GUI.

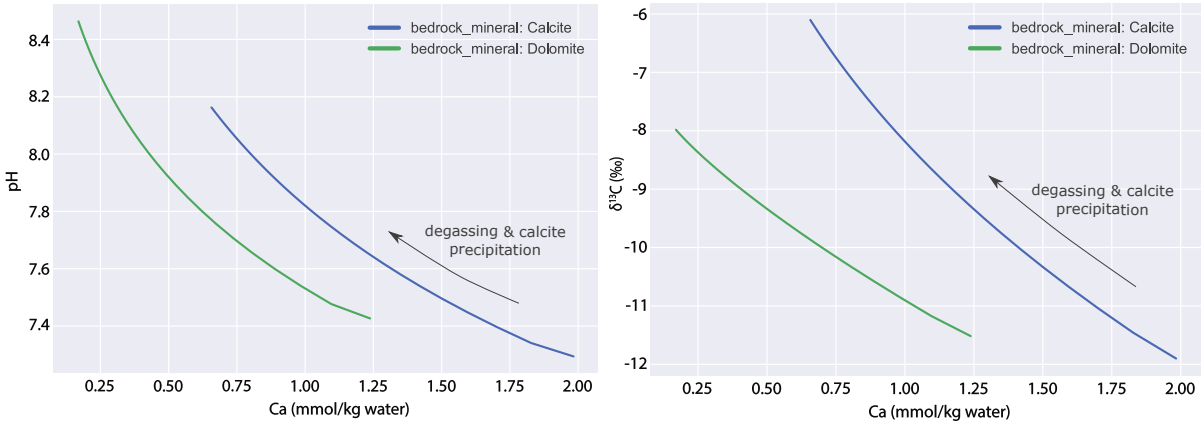



Figure 5.6: Example CaveCalc output (see Section 5.5). Left: pH evolution during $CO_2$ degassing and calcite precipitation. Right: Bulk solution $\delta^{13}C$ evolution during $CO_2$ degassing and calcite precipitation. In both cases, the highest Ca concentration represents the solution immediately following bedrock dissolution and the lowest Ca concentration represents the point at which the solution reaches equilibrium with cave air $pCO_2$ and degassing ceases. In blue are calculations using calcite bedrock, and in green using dolomite bedrock.

Figure 5.6 shows the co-evolution of solution Ca concentration, $\delta^{13}C$ and pH between these two models. Many other aspects of dripwater chemistry could be visualised — these were chosen simply for illustrative purposes. Notable features include the initial, post-bedrock dissolution chemistry of each dripwater, the progressive loss of Ca and movement to higher pH and $\delta^{13}C$ as $CO_2$ degassing and calcite precipitation occur, and the end-point at which dripwater reaches equilibrium with cave air $pCO_2$ (500 ppm in this case). It is interesting, although perhaps not surprising, that simply changing bedrock mineralogy affects all of these features in the model output.

## 5.6 Comparison with other models

The core aim in the development of CaveCalc was to model dripwater and speleothem metal isotopes (e.g. $\delta^{44/42}Ca$) and trace metal contents (e.g. Mg/Ca, Sr/Ca) in a framework that also included an isotope-enabled DIC system ($\delta^{18}O$, $\delta^{13}C$ and radiocarbon). In this regard, CaveCalc is a unique and useful tool that allows coupled modelling of many inorganic geochemical proxies, with the potential to expand to include many more. However, several other models have been published for individual proxy systems; this section discusses the capabilities and design of CaveCalc compared to these specific models.

### 5.6.1 Ca Isotopes

To date, the only published model for dripwater and speleothem Ca isotopes is that of Owen et al. (2016), introduced in Chapter 3. In that work, I used a simple Rayleigh distillation model to account for the incremental fractionation of Ca isotopes from an initial bedrock composition as progressive calcite precipitation occurs. The approach used by CaveCalc is similar, but more flexible. The iterative $CO_2$ degassing and $CaCO_3$ precipitation calculations used by CaveCalc to simulate PCP and stalagmite growth using the 'closed_system_rayleigh' Run Mode are a numerical approximation to the same Rayleigh distillation equation. Model results are correspondingly very similar. CaveCalc is more flexible as it allows the user to specify a non-bedrock Ca source, and may simulate the effects of recrystallisation reactions on Ca isotopes — neither feature could easily be added to the analytical model of Owen et al. (2016).

### 5.6.2 Trace Metals

Several models exist for Mg/Ca, Sr/Ca and Ba/Ca ratios in dripwaters and speleothems (e.g. Fairchild et al., 2000; Sinclair, 2011; Stoll et al., 2012; Owen et al., 2016), although none are coupled to a model of bulk solution chemistry. All model trace element partitioning between dripwater and calcite using a partition coefficient, but differ in the additional processes, and metal sources, considered. The models of Fairchild et al. (2000), Sinclair (2011) and Owen et al. (2016) use experimentally-derived partition coefficients to calculate increases in dripwater X/Ca ratio as calcite is incrementally removed from solution via calcite precipitation; Sinclair (2011) also calculated expected changes due to incongruent dissolution. These two processes, along with source composition (i.e. bedrock), are usually considered the primary controls on speleothem trace element chemistry. Such temperature-dependent calcite trace element partitioning calculations, be they for calcite precipitation, equilibration, or dissolution reactions, are readily performed by CaveCalc and may be structured flexibly by the user. By default, CaveCalc uses the temperature-dependent partition coefficients of Day and Henderson (2013); however the database may be modified to use any desired calibration.

Stoll et al. (2012) presented a more complex trace element model (I-STAL) for inversion of observed trace element data. In their model, initial dripwater chemistry (Ca, Mg, Sr and Ba concentrations) is user-defined and tracked through PCP and stalagmite growth. Unlike CaveCalc, I-STAL includes a time component and clearly differentiates between PCP and calcite precipitation on the speleothem surface. I-STAL parametrises chemical kinetics during $CO_2$ degassing and $CaCO_3$ precipitation, highlighting the control of cave air $pCO_2$ on the extent of PCP and the influence of drip rate on observed fractionation between dripwater and calcite.

The approach used by CaveCalc to simulate PCP and speleothem growth differs significantly from I-STAL. CaveCalc does not explicitly include reaction kinetics or drip rates; by default, PCP and speleothem growth are both considered part of a continuum of calcite precipitated as dripwater $pCO_2$ evolves towards equilibrium with cave air. This simpler approach has advantages over the kinetic treatment of I-STAL — it is difficult to reliably extrapolate kinetic parameters determined from modern cave monitoring into the past, and it is impossible to estimate kinetic parameters for now-inactive drip-sites. These factors make CaveCalc simpler to apply in general cases; however, the approach of I-STAL may provide a more thorough description of trace-element cycling where modern calibration data are available.

A final advantage of CaveCalc's trace element modelling, compared to other models mentioned here, is the inclusion of a model for bedrock dissolution (Section 5.3.3). This allows the user to calculate the initial dripwater chemistry from soil and bedrock parameters, rather than assuming a starting composition.

### 5.6.3 C and O Isotopes

Extensive modelling work has focused on speleothem and dripwater $\delta^{13}C$ and $\delta^{18}O$ isotope ratios. These models may broadly be divided into two categories: those that aim to model the full source-to-stalagmite evolution path of dripwater isotope signals (e.g. Hendy, 1971; Deines et al., 1974; Fohlmeister et al., 2011), and those that focus on modelling thin-film kinetic processes occurring on the stalagmite surfaces in order to fully understand the

observed fractionations between dripwater and calcite (Mühlinghaus et al., 2009; Scholz et al., 2009; Dreybrodt et al., 2016). The $\delta^{13}C$ and $\delta^{18}O$ modelling capabilities of CaveCalc fall into the former category.

CaveCalc is able to calculate mixing between sources (e.g. atmosphere, soil, bedrock) and track solution isotopes through $CO_2$ degassing and equilibrium $CaCO_3$ precipitation, but does not attempt to model any time-dependent kinetic processes. As mentioned in Section 5.6.2, the absence of many kinetic processes in CaveCalc is a design decision; CaveCalc is intended to model equilibrium processes.

### 5.6.4 PHREEQC-only Models

PHREEQC alone has the potential to be applied to many problems in speleothem science that do not require the additional functionality provided by CaveCalc. For example, Baker et al. (2016) used PHREEQC to investigate the controls on dripwater Ca concentrations globally by modelling bedrock dissolution under open- and closed- systems while varying soil temperature and $pCO_2$.

Studies such as Baker et al. (2016) are highly informative, and could be further improved by using CaveCalc. The key advantages of using CaveCalc over direct PHREEQC modelling are:

**Isotope-enabled Bedrock** CaveCalc allows the user to define the isotopic composition of carbonate bedrock minerals (calcite or dolomite), as well as their trace element content, for use during bedrock dissolution. When using PHREEQC directly, Calcite bedrock is assumed to be 'pure' $CaCO_3$, with no minor isotopes present. CaveCalc therefore allows isotope-enabled modelling of bedrock dissolution.

**Isotope-enabled Gases** Similarly to bedrock, CaveCalc allows the user to define the isotopic composition of $CO_2$-bearing gases. In pure PHREEQC this is difficult, and the user is generally limited to specifying only the concentration of $CO_2$. CaveCalc allows isotope-enabled modelling of solution-gas $CO_2$ exchange.

**Defined Volume Gases** In addition to being isotope-enabled, it is also easy to define

the initial volume of a gas phase in CaveCalc. This is primarily used during bedrock dissolution where the volume of gas able to exchange with the solution quantifies the extent of open-system dissolution (Hendy, 1971). In contrast, in pure PHREEQC the user is limited to either using an infinite volume (fully open-system) or zero volume (fully closed system) during mineral dissolution reactions.

**Incremental $CO_2$ Degassing** CaveCalc allows incremental, isotope-enabled $CO_2$ degassing under either equilibrium or kinetically-controlled conditions. This is essential for modelling progressive $CO_2$ degassing and calcite precipitation. Direct use of PHREEQC allows incremental removal of $CO_2$ from solution, but not in an isotope-enabled fashion.

This additional functionality allows fully isotope-enabled modelling of the soil-karst-cave system.

## 5.7 Future Development

It is hoped that future workers may add additional functionality to CaveCalc to expand its functionality and address any emerging issues. Such work would typically require knowledge of PHREEQC and/or Python. Possible improvements could include expansion of the oxotope.dat database to support additional chemistry, new isotope systems or improved thermodynamic data.

The list of core reactions supported by the CaveCalc API (Section 5.2.3) could be expanded. Additions of relevance to speleothem research include water mass mixing and evaporation reactions; both would be relatively simple to implement as they are fully supported by PHREEQC.

It was noted in Sections 5.6.2 & 5.6.3 that CaveCalc is an equilibrium chemistry model and does not model DIC speciation or calcite precipitation kinetics, as some stalagmite-surface models do (e.g. Stoll et al., 2012; Dreybrodt et al., 2016). This was a design decision; with the exception of the drip-site itself, the time-scales involved in various processes along the full dripwater flow path are very poorly constrained. As such, a full

treatment of chemical kinetics in CaveCalc would have been uninformative, and potentially confusing. Perhaps more importantly, the time required to develop such a model would have been beyond the scope of this D.Phil. research. Future work could couple CaveCalc with a detailed kinetic model of the drip-site and stalagmite growth. Such an approach would perhaps provide the best of both worlds: the soil-karst-cave model of CaveCalc and a detailed kinetic treatment of stalagmite growth at the end of the flow path.

## 5.8 Conclusions

This chapter has introduced CaveCalc, a new numerical model for dripwater chemistry and isotopes. The model uses PHREEQC to perform various equilibrium chemical and isotopic calculations of interest to understanding the geochemistry of cave systems and may be used to aid interpretation of multi-proxy datasets from dripwater monitoring programs or speleothem time-series.

Key strengths of CaveCalc include its ability to model multiple proxy systems in a single framework, the ability to quantify the extent of open-system dissolution (through the 'Gas Volume' input), and the extensible nature of its PHREEQC-based design. I hope that future work can expand and improve the model as new data become available and new reactions are identified as important in the soil-karst-cave system.

This chapter has provided an overview of the structure and capabilities of CaveCalc. The full source code, as well as installation and operational instructions, are given in Appendix A. In Chapter 6 CaveCalc is applied to tracing carbon isotopes through bedrock dissolution, $CO_2$ degassing and PCP, an important topic in speleothem research.

# Chapter 6

# Disentangling Controls on Cave Carbon Isotopes

## 6.1 Introduction

Cave carbonates (speleothems) are routinely analysed for a range of geochemical proxies to reconstruct local environment and hydrology on timescales ranging from seasonal (e.g. Johnson et al., 2006) to multi-millennial (e.g. Wang et al., 2008). Such reconstructions are made using a variety of proxies, including C and O stable isotope ratios (Dykoski et al., 2005), radiocarbon (e.g. Genty et al., 2001), trace metals (e.g. Cruz et al., 2007) and metal isotopes (e.g. Goede et al., 1998; Riechelmann et al., 2012; Owen et al., 2016). The diversity of environmental controls on these proxies reflects the complexity of the soil-karst-cave system. Any given proxy signal may be originally derived from rainwater chemistry, but is likely modified by the range of processes occurring in the soil, karst and cave as rainwater percolates into the cave to become speleothem-forming dripwaters. Such processes include respiration of organic matter in the soil, dissolution of silicate and carbonate minerals in the soil, karst aquifer processes (water mass mixing, $CO_2$ degassing, calcite precipitation), and $CO_2$ degassing and calcite precipitation within the cave to form speleothems.

Of these proxy systems, one of the most complex to interpret is the stable carbon iso-

tope composition ($\delta^{13}$C). Variations in speleothem and dripwater $\delta^{13}$C have been suggested to respond to changes in atmospheric $pCO_2$ (Breecker, 2017), soil $pCO_2$ (e.g. Hellstrom et al., 1998), changes between C3 and C4 vegetation in the overlying soil (e.g. Talma and Vogel, 1992), ageing of soil carbon (Genty et al., 2001), $CO_2$ degassing and calcite precipitation in the overlying karst or speleothem surface (Johnson et al., 2006; Frisia et al., 2011) and changes between open- and closed-system bedrock dissolution (Hendy, 1971; Fohlmeister et al., 2011). Speleothem $\delta^{13}$C records are clearly extremely sensitive to local environmental change, but quantitative interpretation of such records is complex and likely to be highly site-specific.

The conditions under which bedrock dissolves have long been considered an important control on dripwater carbon isotopes (Hendy, 1971). During bedrock dissolution, mixing of three isotopically distinct carbon sources may occur: bedrock carbonate, atmospheric $CO_2$ and respired soil $CO_2$. All three may contribute to the initial dripwater carbon isotope budget. Subsequently, $CO_2$ degassing and PCP may further modify dripwater to higher $\delta^{13}$C values. Finally, carbon isotopes are fractionated during precipitation of calcite to form a speleothem.

While the potential controls are numerous, the three that have the greatest potential to vary dripwater carbon isotopic compositions are:

- Variation in respired soil $CO_2$ isotopes
- Changes in bedrock dissolution conditions
- Changes in the extent of $CO_2$ degassing and PCP

Additional proxy systems may be used to separate these signals. Radiocarbon data allow calculation of the dead carbon proportion (DCP), providing a constraint on bedrock dissolution conditions (e.g. Genty et al., 2001). In addition, Ca isotope data have recently been used to quantify PCP (Owen et al., 2016), and such data could potentially be used to correct $\delta^{13}$C data for the effects of $CO_2$ degassing and PCP. A multi-proxy approach combining $\delta^{13}$C, a$^{14}$C and $\delta^{44/42}$Ca may therefore help disentangle the controls on speleothem and dripwater $\delta^{13}$C data, however the model framework required is currently lacking.

In this Chapter I develop a quantitative understanding of dripwater and speleothem

chemistry, in particular focusing on how bedrock dissolution conditions, $CO_2$ degassing and PCP affect dripwater and speleothem C and Ca isotopes. This is done through application of the CaveCalc model (Chapter 5). In particular, I focus on modelling open- and closed-system dissolution processes, including intermediate (semi-open) conditions that are more representative of natural cave systems. Quantitative modelling of these processes is a necessary step towards disentangling the controls on speleothem and dripwater carbon isotope data.

## 6.2 The Model

CaveCalc is a recently-developed numerical model for dripwater and speleothem chemistry; it is introduced and described more fully in Chapter 5. This section provides a brief overview of salient features of the model and discuss how it can be applied to the problem of understanding the Ca and C isotope budgets of the saturated dripwater entering the karst/cave system.

CaveCalc is a custom-built software package designed to mimic the karst and cave environment and is based on the geochemical modelling software PHREEQC. It is designed to perform equilibrium chemical reactions in sequence, tracing bulk solution chemistry, speciation and isotopes through each step. CaveCalc is able to model many processes in the soil-karst-cave system (e.g. soil water-gas equilibrium, bedrock dissolution and prior calcite precipitation); here I focus on the bedrock dissolution step. Key fractionation factors used by the model are given in Table 6.1. For full information on all reactions and thermodynamic data used by CaveCalc, refer to Chapter 5 and Appendix A.

Bedrock dissolution is modelled as follows. Firstly, a soil water chemistry is defined before any bedrock dissolution occurs — this sets the initial carbon budget, as well as any foreign (e.g. aerosol-derived) ions present. One kilogram of this solution then quantitatively dissolves carbonate bedrock until saturation is reached with respect to calcite. This dissolution process occurs while in equilibrium with a gas of user-defined volume, $pCO_2$, $O_2$ and isotopic composition. The volume of gas (specified in L/kg water) may be varied;

Table 6.1: Stable isotope fractionation factors used for key parts of the $CO_2$-DIC-Calcite system. Data are taken from the 'oxotope.dat' database used by CaveCalc (Chapter 5). $^{14}C$ fractionation factors are assumed double those given for $^{13}C$. Full details of these and all other fractionation factors and equilibrium constants used can be found in the database file used by CaveCalc.

| $^{13}C/^{12}C$ | Value at 25 °C ($1000ln(\alpha)$) | Source |
|---|---|---|
| $\alpha_{CO2(aq)\text{-}CO2(g)}$ | -1.43 | Zhang et al. (1995) |
| $\alpha_{HCO3\text{-}(aq)\text{-}CO2(g)}$ | 7.24 | Zhang et al. (1995) |
| $\alpha_{CO3\text{-}2(aq)\text{-}CO2(g)}$ | 5.91 | Zhang et al. (1995) |
| $\alpha_{Calcite\text{-}CO2(g)}$ | 8.95 | Romanek et al. (1992) |
| $^{18}O/^{16}O$ | | |
| $\alpha_{Calcite\text{-}H2O}$ | 29.76 | Coplen (2007) |
| $\alpha_{CO2(g)\text{-}H2O}$ | 40.0 | O'Neil and Adami (1969) |
| $^{44}Ca/^{40}Ca$ | | |
| $\alpha_{Calcite\text{-}H2O}$ | -1.27 | Owen et al. (2016) |

zero volume represents closed-system dissolution and an infinitely large volume represents open-system dissolution. Intermediate values provide a simple way to quantify semi-open conditions, in contrast to previous modelling studies that have expressed semi-open dissolution as open-system followed by closed-system reactions (e.g. Fohlmeister et al., 2011). CaveCalc results exploring the effects of varying this gas volume input and soil $pCO_2$ are discussed in Section 6.3.1. How these signals may be modified by subsequent $CO_2$ degassing and PCP is discussed in Section 6.3.2.

As well as soil carbon chemistry and open vs closed system dissolution, CaveCalc is able to simulate additional processes that modify the results of bedrock dissolution. Such processes include the presence of foreign metal ions in soil water, incongruent dissolution, water-rock interactions, and pyrite oxidation. It is not possible to cover all these processes here, however some consideration is given to modelling the effects of pyrite oxidation. The approach used and results are discussed in Section 6.3.3.

The approach to modelling semi-open bedrock dissolution used by CaveCalc differs significantly from the that used previously by Fohlmeister et al. (2011). They modelling semi-open dissolution as fully-open system dissolution followed by fully-closed system dissolution. These approaches are compared in Section 6.4. Finally, model output is compared with a compilation of Holocene speleothem data in Section 6.5.

## 6.3 Model Results

The results discussed in this section are intended to explore the model parameter-space to investigate the controls and co-variances of various modelled proxy systems. Section 6.3.1 considers the effects of varying soil $pCO_2$ and gas volume on dripwater C and Ca budgets. Section 6.3.2 demonstrates how these signals may be modified by $CO_2$ degassing and PCP. Section 6.3.3 discusses the potential effects of pyrite oxidation on dripwater Ca and C budgets.

These results are general and theoretical — they are intended to provide a overview of environment-proxy relationships across a range of cave conditions, rather than modelling any particular cave system in great detail. Readers interested in applying CaveCalc more specifically to their study site are encouraged to do so; details of the model operation are given in Chapter 5 and Appendix A.

### 6.3.1 Effect of gas volume and $pCO_2$ on bedrock dissolution

Two key controls on the Ca and C budgets of cave dripwaters are the extent of open-system dissolution (i.e. the volume of gas available for $CO_2$ exchange) and $pCO_2$ of that gas (Hendy, 1971). Together, these parameters define the total gaseous $CO_2$ available for exchange with water during bedrock dissolution. The extent of open-system dissolution is affected by hydrological conditions: closed-system conditions likely dominate during periods of high soil saturation while more open-system conditions are expected during periods of relative aridity as the ratio of gas-to-water in soil pore spaces increases. Soil conditions may also play a role determining the extend of open-system conditions under similar hydrological conditions. For example, extensive root systems may promote gas exchange, increasing the effective gas volume, while clay-rich soils likely inhibit exchange. Soil $pCO_2$ itself is related to soil respiration rates. Both parameters contain information about local soil and hydrological conditions.

A suite of CaveCalc models were run to investigate the effects of varying gas volume and soil $pCO_2$ on dripwater Ca concentrations, $\delta^{13}C$ and DCP. All calculations used a

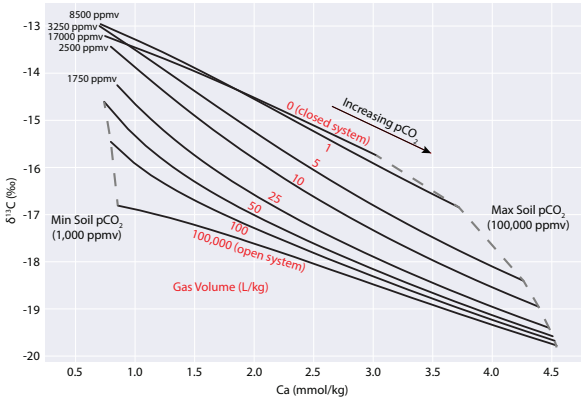
temperature of 15 °C, soil gas $\delta^{13}C$ of −25 ‰, soil radiocarbon activity ($a^{14}C$) of 100 pMC and bedrock $\delta^{13}C$ of 0 ‰. Gas volume was varied from 0 to 100 000 L kg$^{-1}$ to simulate the complete range of open-to-closed bedrock dissolution. 100 000 L kg$^{-1}$ was chosen as an arbitrarily large value to approximate fully open-system dissolution as the model does not support explicitly specifying an infinite volume. Soil gas $pCO_2$ was varied from 1000 to 100 000 ppmv to cover as full a range of conditions as possible. This upper bound on soil gas $pCO_2$ is set conservatively high. Results of these simulations are shown in Figure 6.1. Solid lines represent variation in $pCO_2$ at a constant gas volume. All model output displayed assumes a calcite bedrock. Model calculations using dolomite bedrock were also performed, but are not shown in Figure 6.1 for clarity. Dolomite-derived dripwaters exhibit significantly lower Ca concentrations than limestone-derived dripwaters, as expected due to the relative solubilities of the two minerals; however, their C isotope budgets are similar.

### Open-System Dissolution

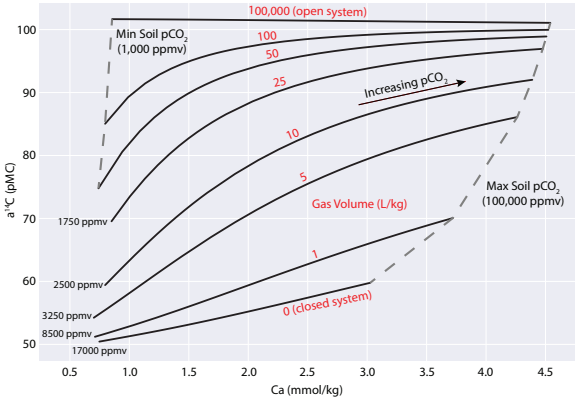
Under fully open-system conditions — simulated using a large gas volume of 100 000 L kg$^{-1}$ — dripwater radiocarbon is essentially constant with $a^{14}C$ = 100 pMC (Figure 6.1b), $\delta^{13}C$ is low (approximately −17 to −20 ‰; Figure 6.1c) and dripwater Ca concentrations are variable, depending on $pCO_2$. Increasing $pCO_2$ at open-system conditions leads to more negative $\delta^{13}C$ and higher Ca concentration (0.8–4.5 mmol kg$^{-1}$; Figure 6.1a). This is explained by increasing $pCO_2$ driving more extensive bedrock dissolution (and hence increasing Ca concentrations) while also providing a larger gaseous $CO_2$ pool for C isotope exchange, diluting the bedrock signal seen in dripwater $\delta^{13}C$ and radiocarbon.

### Closed-System Dissolution

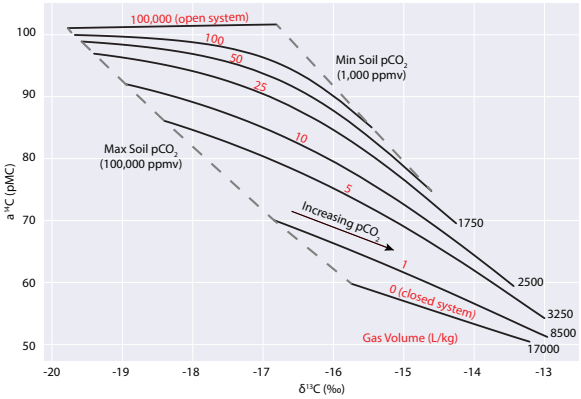
Closed-system conditions are simulated with a gas volume of zero. Compared to open-system dissolution, closed-system dissolution results in systematically lower $a^{14}C$ (50–60 pMC), higher $\delta^{13}C$ (−16 ‰ to −13 ‰) and somewhat lower Ca concentration (0.13–2.7 mmol kg$^{-1}$). Increasing $pCO_2$ under closed-system conditions leads to increased Ca concentration, increased $a^{14}C$ and more negative $\delta^{13}C$. Note that Figure 6.1 does not show




(a) $\delta^{13}$C vs Ca concentration.




(b) Radiocarbon vs Ca concentration.




(c) Radiocarbon activity vs $\delta^{13}$C.

Figure 6.1: Modelled dripwater chemistry immediately following limestone bedrock dissolution, showing expected ranges and co-variation between $\delta^{13}$C, Ca concentration and radiocarbon activity. Lines represent varying soil $pCO_2$ from 1000 to 100 000 ppmv under constant gas volume. The three panels plot different combinations of output parameters: $\delta^{13}$C is plotted against Ca concentration (top left), $a^{14}$C against Ca concentration (top right) and $a^{14}$C against $\delta^{13}$C (bottom left). Calculated dripwater chemistries that would not lead to $CO_2$ degassing and calcite precipitation under a 500 ppmv atmosphere are not shown — this typically occurs at a combination of low soil $pCO_2$ and low gas volume (e.g. top left corner; Figure 6.1a). In such cases the lowest $pCO_2$ that leads to $CO_2$ degassing and calcite precipitation is labelled.

model results with low gas volume and low $pCO_2$. Such solutions typically have $a^{14}$C values <50 pMC, $\delta^{13}$C values up to −6.5 ‰ and very low Ca concentrations (<0.5 mmol kg$^{-1}$). Waters of these compositions may well form in some karst settings, but are unlikely to lead to speleothem growth under reasonable cave air $pCO_2$ levels and hence are not shown here for clarity. The minimum soil $pCO_2$ needed for closed-system bedrock dissolution to produce speleothem-forming dripwater is approximately 17 000 ppmv, assuming a rather low cave air $pCO_2$ of 500 ppmv. Higher soil $pCO_2$ would be required for less well ventilated caves. The soil $pCO_2$ required to produce speleothem-forming dripwater decreases dramatically as gas volume is increased from zero.

**Intermediate Dissolution Conditions**

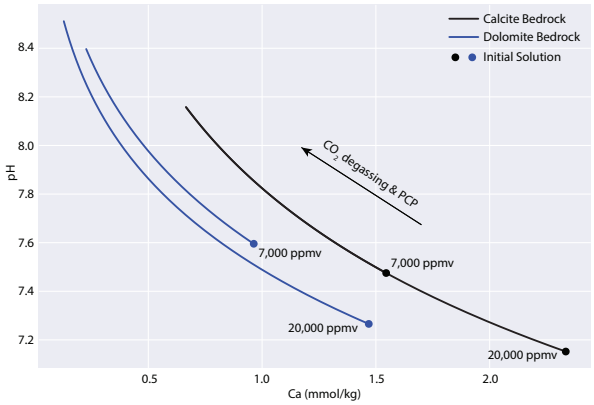
Dissolution calculations under intermediate gas volumes fall between fully-open and fully-closed conditions without exhibiting any more complex behaviour. In general, more open-system conditions lead to higher Ca concentration, higher $a^{14}C$ and more negative $\delta^{13}C$, however the exact values depend strongly on soil $pCO_2$ (Figure 6.1). The relationship between changing gas volume and resulting geochemical signal is highly non-linear; closed-systems are far more sensitive to gas volume changes than open ones. For example, at high $pCO_2$ increasing gas volume from 0 to $1\,L\,kg^{-1}$ increases $a^{14}C$ by $\sim$10 pMC. To achieve the same effect starting at a gas volume of $5\,L\,kg^{-1}$ requires an increase to approximately $25\,L\,kg^{-1}$. Finally, it is noteworthy that very high gas volumes ($>100\,L\,kg^{-1}$) are required to approach theoretical fully-open system dissolution (approximated here using a gas volume of $100\,000\,L\,kg^{-1}$).

Intermediate dissolution conditions are required to produce DCP values similar to those measured in natural cave systems (9–40 %; Table 6.2). However, DCP alone cannot be used to quantify the extent of open-system dissolution (gas volume), as there is a trade-off between the volume of $CO_2$-bearing gas present and the concentration of gaseous $CO_2$. Figure 6.1 demonstrates that by coupled use of dripwater $\delta^{13}C$ and DCP (calculated from $a^{14}C$ measurements — Section 1.2.2), these two controls may be separated.

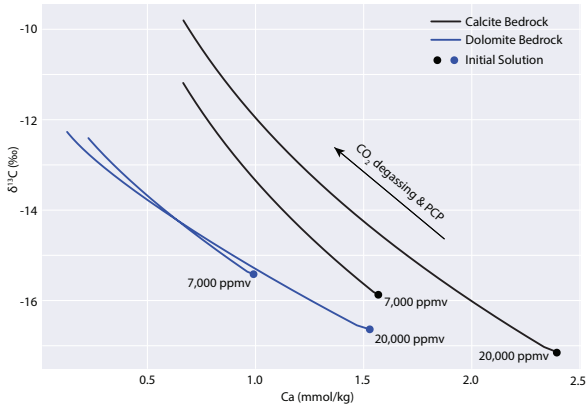
The model results presented in this section cover a wide range of soil and bedrock dissolution conditions, demonstrating the ranges of Ca concentration, $\delta^{13}C$ and $a^{14}C$ (and co-variation between them) that may be expected from changing soil $pCO_2$ and gas volume. This may provide a useful tool for quickly evaluating observed dripwater and/or speleothem chemistry against theoretical predictions. This forward-modelling approach may be applied to all chemical and isotopic systems tracked by CaveCalc.

### 6.3.2 Prior Calcite Precipitation

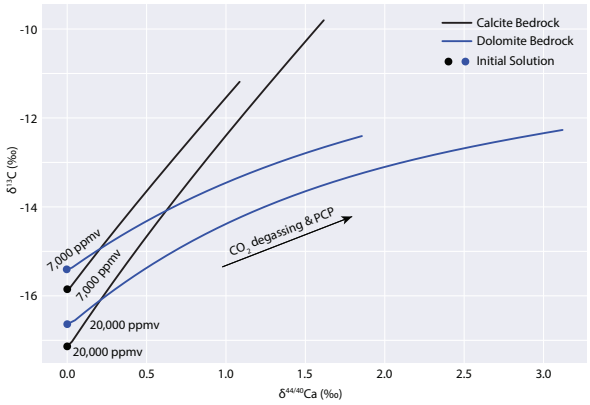
PCP, introduced in Section 1.2.3, may modify the initial dripwater compositions calculated in Section 6.3.1. Importantly, PCP is expected to increase dripwater $\delta^{13}C$ as




(a) pH evolution during PCP.




(b) $\delta^{13}$C evolution during PCP.




(c) $\delta^{13}$C & $\delta^{44/42}$Ca co-evolution.

Figure 6.2: Dripwater evolution as $CO_2$ degassing and PCP occur. Lines represent chemical and isotopic evolution of a single dripwater model as it undergoes $CO_2$ degassing and PCP while maintaining calcite saturation. PCP ends when the dripwater reaches an equivalent $CO_2$ concentration of 500 ppmv. Four model calculations are shown: two using limestone bedrock, and two with dolomite bedrock. For each bedrock mineral, one model uses a soil $pCO_2$ of 7000 ppmv and the other uses a soil $pCO_2$ of 20 000 ppmv. All models have a gas volume of 25 L kg$^{-1}$ and other settings are identical to those used in Section 6.3.1. Panel a (top left) and panel b (top right) show PCP moving dripwater chemistry to lower Ca concentrations while increasing pH and $\delta^{13}$C, respectively. Panel c (bottom left) shows dripwater chemistry evolving to higher $\delta^{44/40}$Ca, as isotopically light Ca is removed from solution.

isotopically-light $CO_2$ is degassed from solution. CaveCalc can model PCP by incremental $CO_2$-degassing and calcite precipitation, such that calcite remains at saturation (Run Mode 'closed_system_rayleigh'; Chapter 5). This is the simplest model for PCP; it is possible that $CO_2$-degassing occurs without concurrent calcite precipitation, or that calcite precipitation is driven by factors other than saturation equilibrium (e.g. microbial activity). Such processes may lead to different chemical and isotopic signatures. Nevertheless, here discussion is limited to the simple case where PCP is approximated via Equation 1.6, reproduced below:

$$Ca^{2+} + 2\,HCO_3^- \longrightarrow CO_2 + CaCO_3 + H_2O \tag{6.1}$$

CaveCalc can model progressive $CO_2$ degassing and calcite precipitation until a target $pCO_2$ value is reached (e.g. cave air $pCO_2$). This is useful for quantifying the maximum extent of dripwater chemical and isotopic evolution, as well as quantifying the total amount of calcite a solution can precipitate under a given $pCO_2$. This functionality can be used to make predictions about expected dripwater and calcite chemical evolution as PCP occurs. These calculations use starting compositions such as those calculated in Figure 6.1, and allow them to undergo PCP.

Figure 6.2 shows PCP evolution pathways for a selection of calculated initial dripwater compositions. Each line represents the chemical and isotopic evolution of a modelled dripwater chemistry as $CO_2$ degassing and calcite precipitation occur. As a solution undergoes degassing and PCP, Ca is removed from the solution and the solution evolves to higher pH (Figure 6.2a) and lighter $\delta^{13}C$ (Figure 6.2b). PCP ends when the solution reaches ambient cave air $pCO_2$; for this calculation a value of 500 ppmv is used. The limestone bedrock models used here are a subset of those shown in Figure 6.1. All models shown calculate bedrock dissolution at a gas volume of $25\,L\,kg^{-1}$, chosen as a representative semi-open-system value. The models shown differ in soil $pCO_2$ and bedrock mineralogy.

Figure 6.2a shows that the pH-Ca evolution of dripwater formed by calcite dissolution is essentially independent of soil $pCO_2$. Lower $pCO_2$ simply causes solutions to begin further along the dripwater evolution path (i.e. at lower Ca concentration). All such solutions stop undergoing PCP at the same point (pH $\sim$ 8.2). In contrast, the evolution of dripwaters formed by dolomite dissolution depends on soil $pCO_2$; dolomite-derived dripwaters formed under higher $pCO_2$ have systematically lower pH at all Ca concentrations. Such dripwaters also finish degassing at lower Ca concentrations and may evolve to a higher final pH than dolomite-derived dripwaters formed under lower $pCO_2$. Dolomite-derived dripwater models evolve to significantly higher pH and lower Ca concentration than calcite-derived dripwaters.

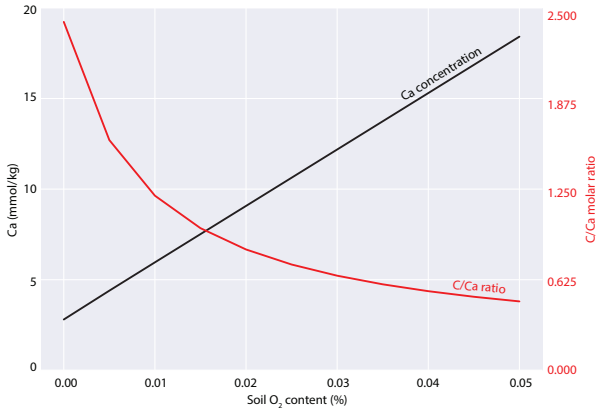
Similarly, Figure 6.2b shows $\delta^{13}$C-Ca evolution for the same models. Dripwaters derived from calcite dissolution evolve to much higher $\delta^{13}$C values (up to $-8\,‰$) than those derived from dolomite dissolution (up to $-12\,‰$). The lower C/Ca ratios of calcite-derived dripwaters relative to dolomite-derived dripwaters leads to greater depletion, and hence fractionation, of the DIC pool during PCP.

PCP evolution trends in certain proxies (e.g. Mg/Ca, $\delta^{13}$C and $\delta^{44/40}$Ca) are of interest as they may be preserved in speleothems and provide records of palaeo-aridity (Johnson et al., 2006; Cruz et al., 2007; Owen et al., 2016). The co-variation between $\delta^{13}$C and $\delta^{44/40}$Ca due to PCP is shown in Figure 6.2c. For calcite-derived dripwater, the correlation between calcite $\delta^{13}$C and $\delta^{44/40}$Ca is approximately linear with a gradient ($\sim 4$) determined by their respective fractionation factors. This linearity results from the C/Ca molar ratio of the initial solution being approximately two (Equation 1.2). PCP removes Ca and C from solution in the ratio 1:2, depleting both elements in equal proportions and hence fractionating both Ca and C in a consistent manner. This linearity does not exist with dolomite bedrock, where the initial solution C/Ca ratio is close to one (Equation 1.3); during PCP the solution is depleted of Ca faster than C, and hence Ca is fractionated to heavier values. These results have important implications for identifying PCP trends in multi-proxy datasets.

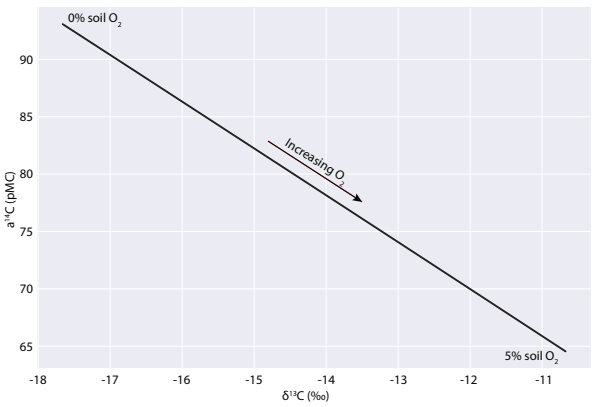
Ca isotopes have been interpreted as a proxy for PCP in speleothem and dripwater data (Chapter|3; Owen et al., 2016). It may be possible to use this proxy, in conjunction with the model output presented here, to correct speleothem carbon isotopes for the effects of PCP. This technique is discussed further and applied in Section 6.5.

### 6.3.3 Pyrite Oxidation

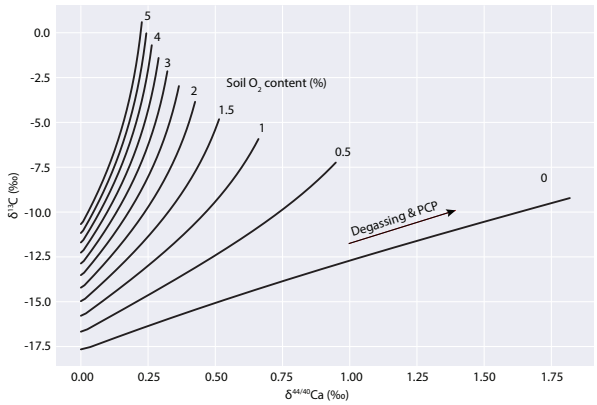
An advantage of CaveCalc over many existing models is its ability to include non-carbonate minerals and aqueous species. Here I use CaveCalc to investigate the effects of pyrite oxidation on initial dripwater chemistry. Pyrite may be present in some karst bedrocks and has been suggested in some regions as the primary driver of bedrock dissolution (e.g. Spötl et al., 2016). Aqueous pyrite oxidation in summarised by Equations 6.2, 6.3, 6.4 & 6.5:




(a) Initial dripwater Ca concentration and C/Ca ratio with variable soil $O_2$.




(b) Initial dripwater radiocarbon activity and $\delta^{13}C$.




(c) Dripwater $\delta^{13}C$ vs $\delta^{44/42}Ca$ PCP trends with variable pyrite oxidation.

Figure 6.3: The effects of pyrite oxidation on initial dripwater chemistry and PCP evolution pathways. This shows a suite of model runs, varying the soil gas $O_2$ concentration (from 0 to 5 %) with an excess of pyrite present during bedrock dissolution. Top left (a): Initial dripwater Ca concentration (black) and C/Ca ratio (red) as soil gas $O_2$ is increased. Top right (b): The effect of soil gas $O_2$ on carbon isotopes. Bottom left (c): The co-evolution of C and Ca isotopes as PCP occurs. Note that all dripwaters have initial $\delta^{44/42}Ca = 0$.

$$FeS_2 + 7\,H_2O + \frac{7}{2}\,O_2 \longrightarrow Fe^{2+} + 2\,SO_4{}^{2-} + 2\,H^+ \tag{6.2}$$

$$4\,Fe^{2+} + O_2 + 4\,H^+ \longrightarrow 4\,Fe^{3+} + 2\,H_2O \tag{6.3}$$

$$Fe^{3+} + 3\,H_2O \longrightarrow Fe(OH)_3 + 3\,H^+ \tag{6.4}$$

$$FeS_2 + 14\,Fe^{3+} + 8\,H_2O \longrightarrow 15\,Fe^{2+} + 2\,SO_4{}^{2-} + 16\,H^+ \tag{6.5}$$

Equations 6.2 describes dissolution of pyrite to form aqueous $Fe^{2+}$. Equation 6.3 describes the oxidation of $Fe^{2+}$ to $Fe^{3+}$. Equations 6.4 and 6.5 describe how $Fe^{3+}$ subse-

quently generates acidity via hydroxide formation and further pyrite dissolution, respectively. These reactions decrease solution pH and drives further carbonate dissolution.

A series of CaveCalc models were run to investigate the impact of pyrite oxidation on dripwater chemistry. These models simulate variable pyrite oxidation by assuming an excess of pyrite in the bedrock, and varying the oxygen content of soil water. Model runs with greater soil oxygen content are expected to undergo more extensive pyrite oxidation than those with anoxic soil waters (Equation 6.2). These models vary soil water $O_2$ content between 0 % and 5 %. The maximum of 5 % is somewhat arbitrary; higher, approaching atmospheric, values are well known at shallow soil depths (e.g. Kursar et al., 1995) however no data are available for soils above cave sites. The range 0–5 % is sufficient to illustrate the effects pyrite oxidation for the purposes of this discussion. Soil $O_2$ concentrations are affected by water saturation, soil porosity and depth. Sites with low soil water saturation, or times of low rainfall, are likely associated with increased soil water oxygen and increased pyrite oxidation, provided pyrite is present in the local bedrock.

Model results are shown in Figure 6.3. Figure 6.3a shows that increasing soil $O_2$ content in the presence of pyrite dramatically increases the extent of bedrock dissolution. Dripwater Ca concentrations increase from $\sim$2.5 mmol mol$^{-1}$ to 18 mmol mol$^{-1}$, a sevenfold increase. Pyrite-oxidation-driven dissolution significantly lowers the C/Ca ratio of the resulting solution (Figure 6.3b), and will impact the PCP evolution path of the dripwater (Figure 6.3c).

Figure 6.3b shows increasing $\delta^{13}$C and decreasing radiocarbon activity as the oxygen concentration of soil gas is increased. This is due to pyrite oxidation leading to increased bedrock dissolution. This increases $\delta^{13}$C by 7 ‰ and raises DCP from $\sim$5 % to 35 %. In terms of carbon isotopes alone, increasing pyrite oxidation has a virtually indistinguishable effect to decreasing soil $pCO_2$ (Figure 6.1). The two mechanisms may be distinguished by their impact on dripwater Ca concentrations; pyrite oxidation dramatically increases dripwater Ca through enhanced bedrock dissolution, while decreasing soil $pCO_2$ decreases dripwater Ca concentration (Section 6.3.1).

Pyrite oxidation affects the subsequent dripwater evolution as it undergoes $CO_2$ de-

gassing and calcite precipitation. Under high-$CO_2$, pyrite-free conditions, the C/Ca molar ratio of dripwaters is expected to be approximately two due $CO_2$-driven carbonate dissolution (Equation 1.2). In the case of pyrite oxidation, carbonate dissolution is driven by pyrite-derived $H^+$ rather than $CO_2$ (Equation 6.6). This adds C and Ca to the solution in the ratio 1:1 and may be followed by $CO_2$ degassing, lowering the dripwater C/Ca ratio further.

$$CaCO_3 + H^+ \longrightarrow Ca^{2+} + HCO_3^- \tag{6.6}$$

The effects of this are shown in Figure 6.3c. As solutions undergo $CO_2$ degassing and calcite precipitation, Ca and C are removed approximately in the ratio 2:1. This causes fractionation of both C and Ca isotopes. In the pyrite-free case this causes an approximately linear co-variance in both $\delta^{13}C$ and $\delta^{44/40}Ca$, as the elements are removed in similar proportions (as discussed in Section 6.3.2). In the model runs featuring pyrite oxidation, the initial solutions have C/Ca ratios less than two, and hence the smaller C isotope pool is fractionated to a much greater extent than the Ca isotopes. Such signals may be preserved in dripwaters and speleothems, particularly in caves with extensive $CO_2$ degassing and/or prior calcite precipitation. Pyrite oxidation followed by PCP may explain anomalously high $\delta^{13}C$ values.

## 6.4 Comparison with Other Models

The method used by CaveCalc to calculate semi-open dissolution conditions is fundamentally different to that employed by previous models. Fohlmeister et al. (2011) modelled semi-open dissolution as an open dissolution reaction followed by a closed system dissolution reaction, varying the proportions of each to control the degree of 'openness'. This contrasts with the work presented here, which quantifies the openness of the system as the volume of gas able to equilibrate during bedrock dissolution (L/kg water). It is therefore informative to compare the different methods.

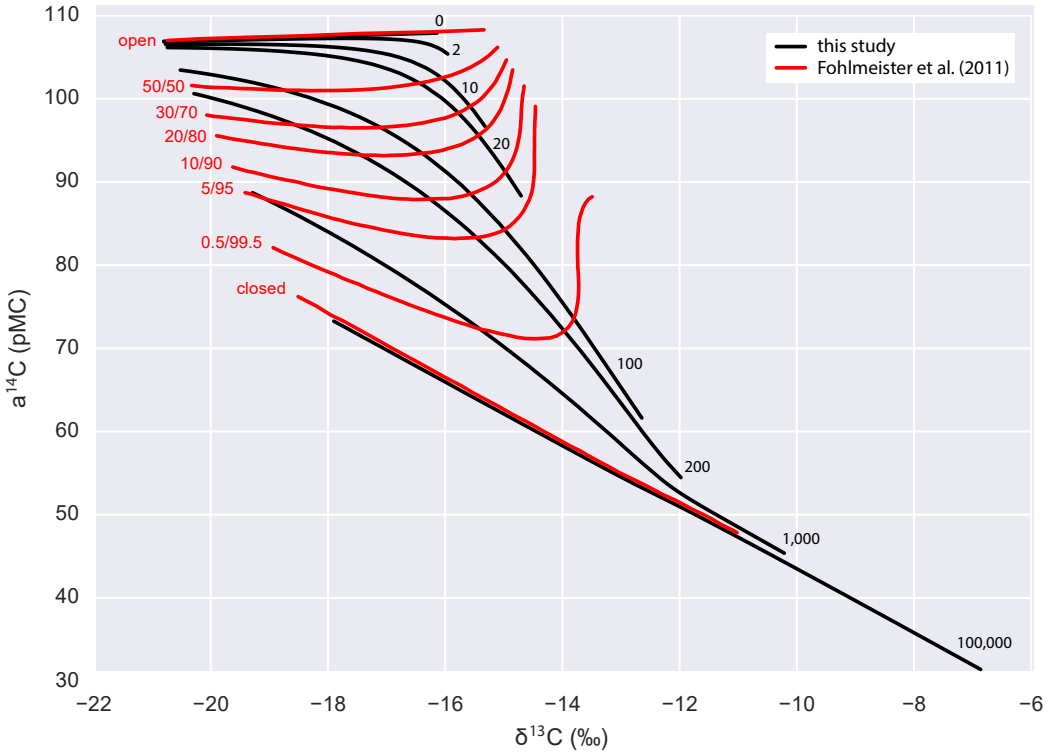



Figure 6.4: A comparison between CaveCalc model output (black) and that of Fohlmeister et al. (2011) (red). Model input parameters have been kept consistent where possible. In both cases lines represent varying soil $pCO_2$ under a constant 'degree of openness', as discussed in Section 6.4. Each black line is labelled with the gas volume used. Each red line is labelled with the proportion of dissolution occurring under open/closed system conditions. $pCO_2$ varied between 9000 and 250 000 ppmv, soil $a^{14}C$ = 106 pMC, temperature = 10 °C, soil gas $\delta^{13}C$ = −25 ‰.

Figure 6.4 shows a comparison of CaveCalc output with results from Fohlmeister et al. (2011). For these calculations, CaveCalc model inputs were set equal to the values used by Fohlmeister et al. (2011) to ensure as close a comparison as possible. Initial dripwater radiocarbon and $\delta^{13}C$ outputs are plotted under varying soil $pCO_2$ and dissolution conditions. In each case, solid lines represent varying model output when $pCO_2$ is changed under a constant degree of openness.

The two models give similar results for fully-open and fully-closed system conditions. This gives confidence that, under these end-member conditions, CaveCalc is in agreement with the equations of Hendy (1971), upon which the model of Fohlmeister et al. (2011) is based. Both models show that, under open-system conditions, increasing $pCO_2$ increases $\delta^{13}C$ while having no significant effect on $a^{14}C$. Under closed system conditions there is a linear relationship between $a^{14}C$ and $\delta^{13}C$ with varying $pCO_2$. Small offsets between

Table 6.2: Holocene speleothem data compilation. Sources: 1: Genty et al. (2001), 2: Liu et al. (2013), 3: Noronha et al. (2014).

| Speleothem | Location | Mean $\delta^{13}C$ (‰) | Mean DCP (%) | Source |
|---|---|---|---|---|
| Fau-stm14 | La Faurie (Dordogne, SW France) | -10.3 | 9 | 1 |
| Pos-stm4 | Postojna (Slovenia) | -9.6 | 11.6 | 1 |
| BFM-Boss | Brown's Folly Mine (UK) | -10 | 17.5 | 1 |
| Vil-stm1 | Villars (SW France) | -9.9 | 9.4 | 1 |
| Han-stm1 | Han-sur-Lesse (Belgium) | -6.8 | 17.5 | 1 |
| Han-stm5 | Han-sur-Lesse (Belgium) | -9.4 | 12.2 | 1 |
| SU-96-7 | Uamh-an-Tartair (Scotland, UK) | -10.7 | 36.7 | 1 |
| SU-2 | Uamh-an-Tartair (Scotland, UK) | -14.9 | 33.6 | 1 |
| SU-96-1 | Uamh-an-Tartair (Scotland, UK) | -12.7 | 24.5 | 1 |
| HS4 | Heshang Cave (China) | -11.0 | 9.7 | 2 & 3 |

the two models likely relate to different calibrations used for equilibrium constants and isotope fractionation factors.

CaveCalc results differ from those of Fohlmeister et al. (2011) for semi-open conditions. CaveCalc results show that if $pCO_2$ is increased while retaining the same degree of openness (gas volume), $a^{14}C$ decreases and $\delta^{13}C$ increases in a monotonic fashion. In contrast, the model of Fohlmeister et al. (2011) shows a minimum in $a^{14}C$ at some intermediate $pCO_2$ value for all semi-open conditions modelled. This discrepancy likely relates to the different methods used to simulate semi-open conditions. I suspect that the $a^{14}C$ inversion seen the model of Fohlmeister et al. (2011) is an artefact of the model implementation; it is surprising that dissolution under 99.5 % closed-system conditions should be the most non-linear result while the result at 100 % closed-system dissolution is linear. What result is expected for 99.95 %? Such non-linearity of response is not expected from this system, and is not reproduced by CaveCalc.

## 6.5 Comparison with Speleothem Data

In this section, CaveCalc model predictions are compared with published carbon isotope values for speleothems from a range of cave sites. The dataset used (Table 6.2) is based on the compilation of Genty et al. (2001), with some additions.

Figure 6.5 shows modelled DCP and $\delta^{13}C$ varying gas volume and soil $pCO_2$, similar to Figure 6.1. Also shown are calculated chemistries for initial precipitates (dashed lines) and dripwaters (solid lines). This is done to facilitate comparison with speleothem data. Black lines indicate model runs with typical values for temperature and soil gas $\delta^{13}C$ (15 °C and −25 ‰, respectively). Red lines represent model runs at 10 °C and −28 ‰, more representative of the conditions at Uamh an Tartair (Baker et al., 1999).

While some speleothem data lie near these model results (e.g. SU-2), the majority are offset to heavier $\delta^{13}C$ and/or lower DCP. These offsets could be due to a variety of factors, given below.

1. Soil conditions ($\delta^{13}C$, temperature, radiocarbon content) may differ from those used in the model.
2. Errors due to uncertainty in relevant precipitation mechanisms and fractionation factors.
3. The effects of $CO_2$ degassing and PCP.
4. Isotope exchange with atmospheric $CO_2$.

### 6.5.1 Soil Conditions

Soil conditions likely differ between sites, although attempts have been made to correct for this. Model output in Figure 6.5 use a soil $\delta^{13}C$ value of −28 ‰ to fit the Uamh-An-Tartair data, as this is more representation of the peat bogs overlying the cave (Genty et al., 2001). The soils overlying all other sites are known to have $\delta^{13}C$ values of approximately −25 ‰ (Genty et al., 2001; Noronha et al., 2015). Mean annual temperature varies between sites, however the effect of this on the resulting carbon isotope budget is small compared to the model-data offsets observed. For these reasons, differences in soil parameters, although likely, are not considered to be a significant uncertainty.

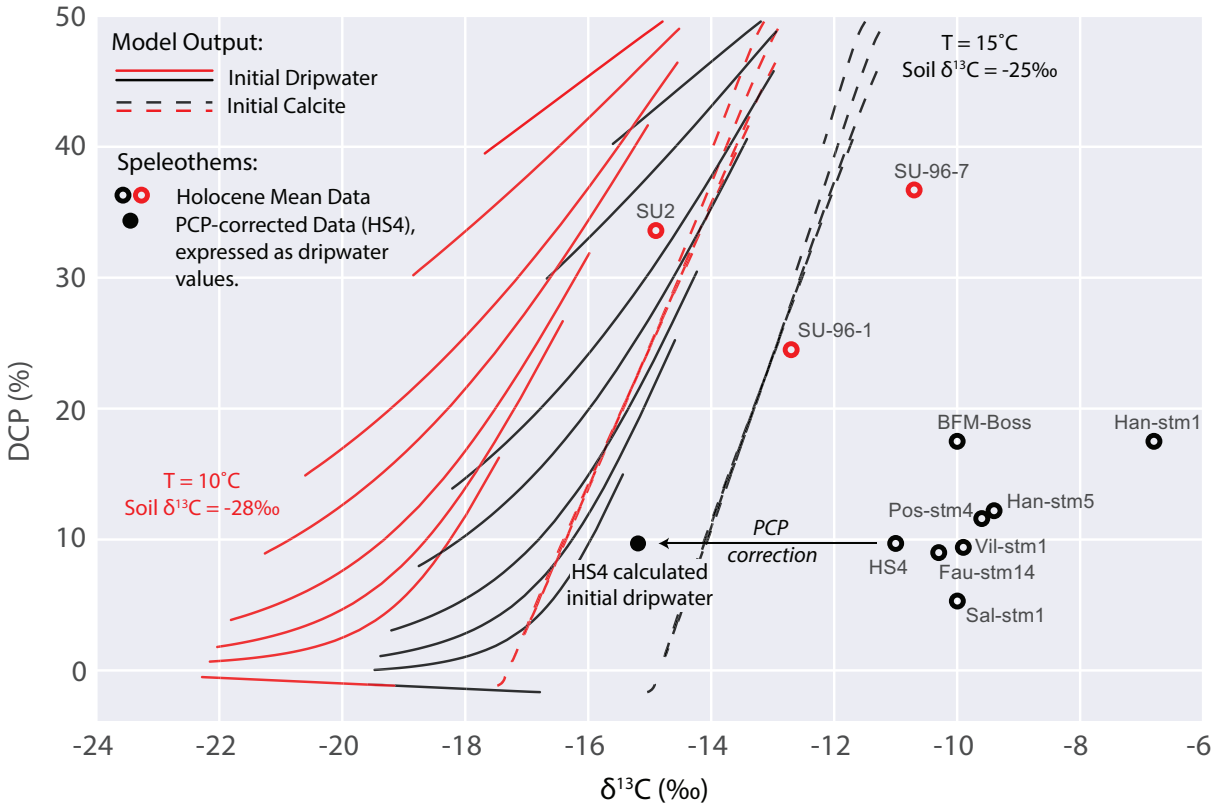



Figure 6.5: Model-data comparison. Each solid line shows calculated initial dripwater isotopes, varying soil $pCO_2$ under a constant gas volume. Gas volume varies between lines. Black represents soil $\delta^{13}C = -25\,‰$ and temperature $= 15\,°C$, representative for most sites shown. Red represents soil $\delta^{13}C = -28\,‰$ and temperature $= 10\,°C$, more representative of conditions for SU samples (Genty et al., 2001). Solid black lines are similar to those shown in Figure 6.1 (bottom left panel) with $a^{14}C$ re-plotted as DCP for ease of comparison with speleothem data. Dashed lines show the instantaneous calcite precipitate predicted for these models, in the same colours. Labelled points show the average speleothem DCP and $\delta^{13}C$ values as given in Table 6.2. The blue point shows HS4 data corrected for the effects of PCP, as discussed in the main text.

### 6.5.2 Precipitation Mechanisms & Fractionation Factors

Stable isotope fractionation between dripwater, gaseous $CO_2$ and speleothem calcite has been modelled extensively (Dreybrodt et al., 1997; Mühlinghaus et al., 2007; Dreybrodt and Scholz, 2011; Dreybrodt et al., 2016). Such studies suggest a diversity of controls on the fractionation of $\delta^{13}C$ between solution and carbonate including dripwater Ca concentration, $CO_2$ degassing kinetics, speleothem growth rate, drip rate and temperature.

With the exception of temperature, these controls are not included in the CaveCalc model. Such factors may reflect a discrepancy between data and model results, however few experimental data are available to quantify this. Furthermore, it is unclear whether detailed precipitation models developed for cave drip-sites are applicable to calcite precip-

itation occurring due to PCP earlier on the drip flow path. Such precipitation may occur in fractures and pore spaces under different — and unconstrained — kinetic regimes.

Given these uncertainties it is difficult to quantify the data-model offset that could be accounted for by variation in precipitation kinetics and uncertainty in fractionation factors. Instead we focus on mechanisms more readily quantified by our model framework and available data.

### 6.5.3 $CO_2$ Degassing and PCP

The effects of $CO_2$ degassing and PCP have been described in Section 6.3.2, and could explain the offset to higher $\delta^{13}C$ values. PCP is also expected to have signals in various trace elements and metal isotopes, allowing identification of this process in a given speleothem where such data are available.

Ca isotope analysis of HS4 suggests that, under regular Holocene conditions, PCP removes approximately 36 % of dripwater Ca before water arrives at the tip of the stalagmite (Chapter 3; Owen et al., 2016). This estimate of calculated PCP may be used with model results to correct speleothem $\delta^{13}C$ for the effects of $CO_2$ degassing and PCP, using model output as shown in Figure 6.2. This analysis gives a predicted $\delta^{13}C$ offset of $4.2 \pm 0.3$ ‰ between the initial dripwater $\delta^{13}C$ and the final, post-PCP calcite precipitate. This PCP-corrected initial dripwater $\delta^{13}C$ is shown on Figure 6.5. Interestingly, the corrected value lies very close to modelled initial dripwater isotopes (solid lines), suggesting that this PCP model can account for a large part of the offset between modelled and observed values at Heshang Cave. Further investigation of this coupled Ca and C isotope approach is necessary to test the method on additional cave sites where PCP is evident.

### 6.5.4 Atmospheric $CO_2$

So far this study has considered two carbon sources: bedrock ($\delta^{13}C \sim 0$ ‰) and soil organic matter ($\delta^{13}C \sim -25$ ‰). However, several speleothems show $\delta^{13}C$ values higher than predicted by this dissolution model (Figure 6.5). Section 6.5.3 discusses how PCP

may explain this offset; this section considers the potential impact of a third carbon source: the atmosphere.

Atmospheric $\delta^{13}C$ is relatively light ($\delta^{13}C \sim -8\,‰$) and could explain the offset to lighter $\delta^{13}C$ seen in the speleothem compilation at sites were PCP is not known to occur. Atmospheric $CO_2$ could be incorporated into dripwater DIC either through mixing of soil gas before or during bedrock dissolution, or by $CO_2$ exchange between dripwater and cave air. In the case of the former, a simple mixing model requires that an increased atmospheric signal in soil gas $\delta^{13}C$ should be accompanied by a significant decrease in soil $pCO_2$. Such a mechanism has implications for the extent of bedrock dissolution and the extent to which dripwaters may degas and precipitate calcite (Section 6.3.1). In the case of the latter mechanism, the potential impact would depend largely on the $CO_2$ and ventilation dynamics of the cave site in question.

A full model treatment of these mechanisms is beyond the scope of this chapter, however I suggest that an atmospheric $CO_2$ source may be an important consideration in cases where speleothem $\delta^{13}C$ is anomalously high and cannot be accounted for by PCP.

## 6.6 Conclusions

This chapter presents a new approach for modelling speleothem and dripwater geochemistry, based on the chemical model CaveCalc, developed in Chapter 5. This approach is capable of quantifying the effects of semi-open bedrock dissolution conditions on dripwater and speleothem chemistry and isotopes, and is particularly useful for quantifying controls on $\delta^{13}C$ signals.

Closed-system dissolution predicts initial dripwater $\delta^{13}C$ in the range $-16$ to $-13\,‰$, $a^{14}C$ in range 50–60 pMC, dripwater Ca in the range 0.1–2.7 $mmol\,kg^{-1}$. Open-system dissolution predicts $\delta^{13}C$ in range $-20$ to $-17\,‰$, $a^{14}C = 0$ pMC and dripwater Ca in the range 0.8–4.5 $mmol\,kg^{-1}$ for a wide range of soil $pCO_2$ values. Semi-open system conditions are modelled and fall between end-member open and closed system results. Model results show that increasing soil $pCO_2$ increases dripwater radiocarbon, decreases $\delta^{13}C$

and increases Ca concentration monotonically for open, closed and all intermediate dissolution conditions. This is in contrast with previous modelling efforts of Fohlmeister et al. (2011), who predicted more complex behaviour. The evolution of these initial dripwater chemistries during incremental $CO_2$ degassing and calcite precipitation is modelled and discussed.

Pyrite oxidation is modelled to have a large effect on dripwater chemistry, resulting in increased dripwater $\delta^{13}C$, increased DCP and raises the C/Ca ratio of dripwaters, modifying their expected chemical and isotopic evolution during degassing and calcite precipitation. The effects of pyrite dissolution on solution carbon isotope budgets are large and may readily overprint signals from other sources in pyrite-bearing regions.

We compare CaveCalc to speleothem carbon isotope data from a range of caves. Some data agree well with the model, however most show offsets to higher $\delta^{13}C$ than predicted. At Heshang Cave, where PCP has been quantified using Ca isotope data, it is shown that this offset may be largely explained by PCP. At other sites offsets to higher $\delta^{13}C$ may be explained by a range of factors, including prior calcite precipitation or isotope exchange with atmospheric $CO_2$.

The modelling approach developed here may be applied to other speleothem and/or dripwater datasets to improve understanding of the controls on multi-proxy records.

# Final Remarks

In this thesis I have developed new, and furthered existing, ways of using speleothem geochemistry to reconstruct past environments and climates. I have introduced Ca isotopes in speleothems as a novel proxy for palaeo-aridity, using Heshang Cave as a study site for both modern calibration work and a initial palaeoclimate record covering the 8.2kyr event. I have also tested and optimised laboratory procedures for conducting these analyses on cave materials, hopefully facilitating future work applying Ca isotope analysis to other cave sites and time periods.

The second half of this thesis has focused on modelling existing multi-proxy datasets, resulting in the development of Cavecalc. This new model is able to forward-model full dripwater chemistry in an isotope-enabled manner. CaveCalc allows simultaneous tracing of multiple proxies through a range soil-karst-cave processes, and may be expanded to include new proxy systems and processes as desired. CaveCalc represents a notable advancement over existing models, firstly for the wide range of proxy systems included, and secondly for its ability to quantitatively model bedrock dissolution under 'semi-open' conditions — conditions that likely exist at many natural cave sites.

Both aspects of this research lay the foundation for future projects. Chapter 3 has shown the power of Ca isotopes to reconstruct PCP, albeit with several potential complications discussed in Chapter 4. Analysis of speleothems for Ca isotopes is fairly straightforward, and could readily be applied to other cave sites, either to complement existing datasets or as part of new multi-proxy reconstructions. Using Ca isotopes to constrain PCP change, in addition to informing discussions of past rainfall change, may highlight other controls on proxy systems (e.g. $\delta^{13}C$, Mg/Ca), potentially providing additional in-

sight into past environments.

On the topic of rainfall reconstruction, the model used to relate PCP change to rainfall in Chapter 3, while a useful first step, is a notable oversimplification. Further observational and modelling studies of the relationship between effective rainfall and PCP could improve our ability to quantitatively reconstruct rainfall change from PCP proxies, such as Ca isotopes.

Finally, potential applications of CaveCalc are many and varied. As a forward-model, CaveCalc can be used to help understand the controls on modern dripwater data, interpret speleothem time-series, or simply as an aid to conducting thought-experiments. Future model development could expand CaveCalc in several directions: additional chemical functionality could be added (e.g. new isotope systems, kinetic processes), or new model features implemented (e.g. inverse modelling). CaveCalc may, and should, be kept up to date with modern calibration data.

I hope the methods, data, tools and ideas presented in this thesis are of interest to a wide range of geochemists and speleothem scientists, and prove useful in future research. Thank you for reading.

# Bibliography


H. P. Affek, M. Bar-Matthews, A. Ayalon, A. Matthews, and J. M. Eiler. Glacial/interglacial temperature variations in Soreq cave speleothems as recorded by 'clumped isotope' thermometry. *Geochimica et Cosmochimica Acta*, 72(22):5351–5360, nov 2008. doi:10.1016/j.gca.2008.06.031.

A. Ayalon, M. Bar-Matthews, and A. Kaufman. Petrography, strontium, barium and uranium concentrations, and strontium and uranium isotope ratios in speleothems as palaeoclimatic proxies: Soreq Cave, Israel. *The Holocene*, 9(6):715–722, nov 1999. doi:10.1191/095968399673664163.

A. Baker, C. J. Caseldine, M. A. Gilmour, D. Charman, C. J. Proctor, C. J. Hawkesworth, and N. Phillips. Stalagmite luminescence and peat humification records of palaeomoisture for the last 2500 years. *Earth and Planetary Science Letters*, 165(1):157–162, 1999. doi:10.1016/S0012-821X(98)00258-1.

A. Baker, I. Flemons, M. S. Andersen, K. Coleborn, and P. C. Treble. What determines the calcium concentration of speleothem-forming drip waters? *Global and Planetary Change*, 143:152–161, 2016. doi:10.1016/j.gloplacha.2016.06.001.

M. Bar-Matthews, A. Ayalon, A. Kaufman, and G. J. Wasserburg. The Eastern Mediterranean paleoclimate as a reflection of regional events: Soreq cave, Israel. *Earth and Planetary Science Letters*, 166(1-2):85–95, feb 1999. doi:10.1016/S0012-821X(98)00275-1.

W. C. Beck, E. L. Grossman, and J. W. Morse. Experimental studies of oxygen isotope fractionation in the carbonic acid system at 15C, 25C, and 40C. *Geochimica et Cosmochimica Acta*, 69(14):3493–3503, jul 2005. doi:10.1016/j.gca.2005.02.003.

C. L. Blättler, H. C. Jenkyns, L. M. Reynard, and G. M. Henderson. Significant increases in global weathering during Oceanic Anoxic Events 1a and 2 indicated by calcium isotopes. *Earth and Planetary Science Letters*, 309(1-2):77–88, 2011. doi:10.1016/j.epsl.2011.06.029.

C. L. Blättler, G. M. Henderson, and H. C. Jenkyns. Explaining the Phanerozoic Ca isotope history of seawater. *Geology*, 40(9):843–846, jul 2012. doi:10.1130/G33191.1.

S. F. Boulyga. Calcium isotope analysis by mass spectrometry. *Mass Spectrometry Reviews*, 29(5):685–716, sep 2010. doi:10.1002/mas.20244.

D. O. Breecker. Atmospheric pCO2 control on speleothem stable carbon isotope compositions. *Earth and Planetary Science Letters*, 458:58–68, 2017. doi:10.1016/j.epsl.2016.10.042.

S. R. Charlton and D. L. Parkhurst. Modules based on the geochemical model PHREEQC for use in scripting and programming languages. *Computers and Geosciences*, 37(10): 1653–1663, 2011. doi:10.1016/j.cageo.2011.02.005.

H. Cheng, D. Fleitmann, R. L. Edwards, X. Wang, F. W. Cruz, A. S. Auler, A. Mangini, Y. Wang, X. Kong, S. J. Burns, and A. Matter. Timing and structure of the 8.2 kyr B.P. event inferred from Âă18O records of stalagmites from China, Oman, and Brazil. *Geology*, 37(11):1007–1010, oct 2009. doi:10.1130/G30126A.1.

L. Chou, R. M. Garrels, and R. Wollast. Comparative study of the kinetics and mechanisms of dissolution of carbonate minerals. *Chemical Geology*, 78(3-4):269–282, dec 1989. doi:10.1016/0009-2541(89)90063-6.

N.-C. Chu, G. M. Henderson, N. S. Belshaw, and R. E. Hedges. Establishing the potential of Ca isotopes as proxy for consumption of dairy products. *Applied Geochemistry*, 21 (10):1656–1667, oct 2006. doi:10.1016/j.apgeochem.2006.07.003.

T. B. Coplen. Calibration of the calcite-water oxygen-isotope geothermometer at Devils Hole, Nevada, a natural laboratory. *Geochimica et Cosmochimica Acta*, 71(16):3948–3957, aug 2007. doi:10.1016/j.gca.2007.05.028.

F. W. Cruz, S. J. Burns, M. Jercinovic, I. Karmann, W. D. Sharp, and M. Vuille. Evidence of rainfall variations in Southern Brazil from trace element ratios (Mg/Ca and Sr/Ca) in a Late Pleistocene stalagmite. *Geochimica et Cosmochimica Acta*, 71(9):2250–2263, may 2007. doi:10.1016/j.gca.2007.02.005.

W. Dansgaard. Ice core evidence of abrupt climatic changes. In W. J. Berger and L. D. Labeyrie, editors, *Abrupt climatic change: Evidence and implications*, pages 223–233. Reidel, Dordrecht, The Netherlands, 1987.

C. C. Day and G. M. Henderson. Controls on trace-element partitioning in cave-analogue calcite. *Geochimica et Cosmochimica Acta*, 120:612–627, nov 2013. doi:10.1016/j.gca.2013.05.044.

C. L. De La Rocha and D. J. DePaolo. Isotopic Evidence for Variations in the Marine Calcium Cycle Over the Cenozoic. *Science*, 289(5482):1176–1178, aug 2000. doi:10.1126/science.289.5482.1176.

P. Deines, D. Langmuir, and R. S. Harmon. Stable carbon isotope ratios and the existence of a gas phase in the evolution of carbonate ground waters. *Geochimica et Cosmochimica Acta*, 38(7):1147–1164, jul 1974. doi:10.1016/0016-7037(74)90010-6.

D. J. DePaolo. Calcium Isotopic Variations Produced by Biological, Kinetic, Radiogenic and Nucleosynthetic Processes. *Reviews in Mineralogy and Geochemistry*, 55(1):255–288, jan 2004. doi:10.2138/gsrmg.55.1.255.

D. J. DePaolo. Surface kinetic model for isotopic and trace element fractionation during precipitation of calcite from aqueous solutions. *Geochimica et Cosmochimica Acta*, 75 (4):1039–1056, feb 2011. doi:10.1016/j.gca.2010.11.020.

Y. Dixit, D. A. Hodell, R. Sinha, and C. A. Petrie. Abrupt weakening of the Indian summer monsoon at 8.2kyr B.P. *Earth and Planetary Science Letters*, 391:16–23, apr 2014. doi:10.1016/j.epsl.2014.01.026.

W. Dreybrodt and D. Scholz. Climatic dependence of stable carbon and oxygen isotope signals recorded in speleothems: From soil water to speleothem calcite. *Geochimica et Cosmochimica Acta*, 75(3):734–752, feb 2011. doi:10.1016/j.gca.2010.11.002.

W. Dreybrodt, L. Eisenlohr, B. Madry, and S. Ringer. Precipitation kinetics of calcite in the system CaCO3 -> H2O+C02: The conversion to CO2 by the slow process H++HCO3âĹŠ -> CO2+H2O as a rate limiting step. *Geochimica et Cosmochimica Acta*, 61(18):3897–3904, sep 1997. doi:10.1016/S0016-7037(97)00201-9.

W. Dreybrodt, M. Hansen, and D. Scholz. Processes affecting the stable isotope composition of calcite during precipitation on the surface of stalagmites: Laboratory experiments investigating the isotope exchange between DIC in the solution layer on top of a speleothem and the CO2 of the cave at. *Geochimica et Cosmochimica Acta*, 174: 247–262, 2016. doi:10.1016/j.gca.2015.11.012.

C. Dykoski, R. Edwards, H. Cheng, D. Yuan, Y. Cai, M. Zhang, Y. Lin, J. Qing, Z. An, and J. Revenaugh. A high-resolution, absolute-dated Holocene and deglacial Asian monsoon record from Dongge Cave, China. *Earth and Planetary Science Letters*, 233 (1-2):71–86, apr 2005. doi:10.1016/j.epsl.2005.01.036.

L. R. Edwards, J. H. Chen, and G. J. Wasserburg. 238U234U230Th232Th systematics and the precise measurement of time over the past 500,000 years. *Earth and Planetary Science Letters*, 81(2-3):175–192, jan 1987. doi:10.1016/0012-821X(87)90154-3.

I. J. Fairchild and A. Baker. *Speleothem Science: from process to past environments.* Wiley-Blackwell, 2012. ISBN 978-1-4051-9620-8.

I. J. Fairchild and P. C. Treble. Trace elements in speleothems as recorders of environmental change. *Quaternary Science Reviews*, 28(5-6):449–468, mar 2009. doi:10.1016/j.quascirev.2008.11.007.

I. J. Fairchild, A. Borsato, A. F. Tooth, S. Frisia, C. J. Hawkesworth, Y. Huang, F. McDermott, and B. Spiro. Controls on trace element (Sr-Mg) compositions of carbonate cave waters: implications for speleothem climatic records. *Chemical Geology*, 166(3-4): 255–269, may 2000. doi:10.1016/S0009-2541(99)00216-8.

I. J. Fairchild, A. Baker, A. Borsato, S. Frisia, R. W. Hinton, F. McDermott, and A. F. Tooth. Annual to sub-annual resolution of multiple trace-element trends in speleothems. *Journal of the Geological Society*, 158(5):831–841, sep 2001. doi:10.1144/jgs.158.5.831.

M. S. Fantle and D. J. DePaolo. Variations in the marine Ca cycle over the past 20 million years. *Earth and Planetary Science Letters*, 237(1-2):102–117, 2005. doi:10.1016/j.epsl.2005.06.024.

M. S. Fantle and D. J. DePaolo. Ca isotopes in carbonate sediment and pore fluid from ODP Site 807A: The Ca2+(aq)-calcite equilibrium fractionation factor and calcite recrystallization rates in Pleistocene sediments. *Geochimica et Cosmochimica Acta*, 71 (10):2524–2546, may 2007. doi:10.1016/j.gca.2007.03.006.

J. Farkaš, F. Böhm, K. Wallmann, J. Blenkinsop, A. Eisenhauer, R. van Geldern, A. Munnecke, S. Voigt, and J. Veizer. Calcium isotope record of Phanerozoic oceans: Implications for chemical evolution of seawater and its causative mechanisms. *Geochimica et Cosmochimica Acta*, 71(21):5117–5134, nov 2007a. doi:10.1016/j.gca.2007.09.004.

J. Farkaš, D. Buhl, J. Blenkinsop, and J. Veizer. Evolution of the oceanic calcium cycle during the late Mesozoic: Evidence from $\delta$44/40Ca of marine skeletal carbonates. *Earth and Planetary Science Letters*, 253(1):96–111, 2007b. doi:10.1016/j.epsl.2006.10.015.

J. Fohlmeister, D. Scholz, B. Kromer, and A. Mangini. Modelling carbon isotopes of carbonates in cave drip water. *Geochimica et Cosmochimica Acta*, 75(18):5219–5228, 2011. doi:10.1016/j.gca.2011.06.023.

G. Fortunato, K. Mumic, S. Wunderli, L. Pillonel, J. O. Bosset, G. Gremaud, and G. Versini. Application of strontium isotope abundance ratios measured by MC-ICP-MS for food authentication. *Journal of Analytical Atomic Spectrometry*, 19(2):227, 2004. doi:10.1039/b307068a.

S. Frisia, I. J. Fairchild, J. Fohlmeister, R. Miorandi, C. Spötl, and A. Borsato. Carbon mass-balance modelling and carbon isotope exchange processes in dynamic caves. *Geochimica et Cosmochimica Acta*, 75(2):380–400, jan 2011. doi:10.1016/j.gca.2010.10.021.

A. Galy, C. France-Lanord, and L. A. Derry. The strontium isotopic budget of Himalayan rivers in Nepal and Bangladesh. *Geochimica et Cosmochimica Acta*, 63(13-14):1905–1925, jul 1999. doi:10.1016/S0016-7037(99)00081-2.

M. Gascoyne. Trace-element partition coefficients in the calcite-water system and their paleoclimatic significance in cave studies. *Journal of Hydrology*, 61(1-3):213–222, feb 1983. doi:10.1016/0022-1694(83)90249-4.

D. Genty and M. Massault. Bomb 14C recorded in laminated speleothems: calculation of dead carbon proportion. *Radiocarbon*, 39(1):33–48, 1997. doi:10.1017/S0033822200040881.

D. Genty, B. Vokal, B. Obelic, and M. Massault. Bomb 14C time history recorded in two modern stalagmites - importance for soil organic matter dynamics and bomb 14C distribution over continents. *Earth and Planetary Science Letters*, 160(3):795–809, 1998. doi:10.1016/S0012-821X(98)00128-9.

D. Genty, A. Baker, M. Massault, C. Proctor, M. Gilmour, E. Pons-Branchu, and B. Hamelin. Dead carbon in stalagmites: carbonate bedrock paleodissolution vs. ageing of soil organic matter. Implications for 13C variations in speleothems. *Geochimica et Cosmochimica Acta*, 65(20):3443–3457, oct 2001. doi:10.1016/S0016-7037(01)00697-4.

D. Genty, D. Blamart, R. Ouahdi, M. Gilmour, A. Baker, J. Jouzel, and S. Van-Exter. Precise dating of Dansgaard-Oeschger climate oscillations in western Europe from stalagmite data. *Nature*, 421(6925):833–7, feb 2003. doi:10.1038/nature01391.

D. Genty, D. Blamart, B. Ghaleb, V. Plagnes, C. Causse, M. Bakalowicz, K. Zouari, N. Chkir, J. Hellstrom, and K. Wainer. Timing and dynamics of the last deglaciation from European and North African $\delta$13C stalagmite profiles-comparison with Chinese and South Hemisphere stalagmites. *Quaternary Science Reviews*, 25(17-18):2118–2142, sep 2006. doi:10.1016/j.quascirev.2006.01.030.

A. Goede and J. Vogel. Trace element variations and dating of a Late Pleistocene Tasmanian speleothem. *Palaeogeography, Palaeoclimatology, Palaeoecology*, 88(1-2):121–131, nov 1991. doi:10.1016/0031-0182(91)90018-M.

A. Goede, M. McCulloch, F. McDermott, and C. Hawkesworth. Aeolian contribution to strontium and strontium isotope variations in a Tasmanian speleothem. *Chemical Geology*, 149(1-2):37–50, jul 1998. doi:10.1016/S0009-2541(98)00035-7.

E. M. Griffith, A. Paytan, K. Caldeira, T. D. Bullen, and E. Thomas. A dynamic marine calcium cycle during the past 28 million years. *Science*, 322(5908):1671–1674, 2008. doi:10.1126/science.1163614.

E. M. Griffith, A. Paytan, A. Eisenhauer, T. D. Bullen, and E. Thomas. Seawater calcium isotope ratios across the Eocene-Oligocene transition. *Geology*, 39(7):683–686, 2011. doi:10.1130/G31872.1.

N. Gussone and H. L. Filipsson. Calcium isotope ratios in calcitic tests of benthic foraminifers. *Earth and Planetary Science Letters*, 290(1-2):108–117, 2010. doi:10.1016/j.epsl.2009.12.010.

N. Gussone, A. Eisenhauer, A. Heuser, M. Dietzel, B. Bock, F. Böhm, H. J. Spero, D. W. Lea, J. Bijma, and T. F. Nägler. Model for kinetic effects on calcium isotope fractionation ($\delta$44Ca) in inorganic aragonite and cultured planktonic foraminifera. *Geochimica et Cosmochimica Acta*, 67(7):1375–1382, apr 2003. doi:10.1016/S0016-7037(02)01296-6.

N. Gussone, F. Böhm, A. Eisenhauer, M. Dietzel, A. Heuser, B. M. Teichert, J. Reitner, G. Wörheide, and W.-C. Dullo. Calcium isotope fractionation in calcite and aragonite. *Geochimica et Cosmochimica Acta*, 69(18):4485–4494, sep 2005. doi:10.1016/j.gca.2005.06.003.

N. Gussone, G. Langer, S. Thoms, G. Nehrke, A. Eisenhauer, U. Riebesell, and G. Wefer. Cellular calcium pathways and isotope fractionation in Emiliania huxleyi. *Geology*, 34 (8):625–628, 2006. doi:10.1130/G22733.1.

L. Halicz, A. Galy, N. S. Belshaw, and R. K. O'Nions. High-precision measurement of calcium isotopes in carbonates and related materials by multiple collector inductively coupled plasma mass spectrometry (MC-ICP-MS). *Journal of Analytical Atomic Spectrometry*, 14(12):1835–1838, 1999. doi:10.1039/a906422b.

J. Hellstrom and M. McCulloch. Multi-proxy constraints on the climatic significance of trace element records from a New Zealand speleothem. *Earth and Planetary Science Letters*, 179(2):287–297, jun 2000. doi:10.1016/S0012-821X(00)00115-1.

J. Hellstrom, M. McCulloch, and J. Stone. A Detailed 31,000-Year Record of Climate and Vegetation Change, from the Isotope Geochemistry of Two New Zealand Speleothems. *Quaternary Research*, 50(02):167–178, sep 1998. doi:10.1006/qres.1998.1991.

C. Hendy. The isotopic geochemistry of speleothems-I. The calculation of the effects of different modes of formation on the isotopic composition of speleothems and their applicability as palaeoclimatic indicators. *Geochimica et Cosmochimica Acta*, 35(8): 801–824, aug 1971. doi:10.1016/0016-7037(71)90127-X.

A. Heuser and A. Eisenhauer. The Calcium Isotope Composition ($\delta$ 44/40 Ca) of NIST SRM 915b and NIST SRM 1486. *Geostandards and Geoanalytical Research*, 32(3): 311–315, sep 2008. doi:10.1111/j.1751-908X.2008.00877.x.

A. Heuser, A. Eisenhauer, F. Böhm, K. Wallmann, N. Gussone, P. N. Pearson, T. F. Nägler, and W. C. Dullo. Calcium isotope ($\delta$44/40Ca) variations of Neogene planktonic foraminifera. *Paleoceanography*, 20(2):1–13, jun 2005. doi:10.1029/2004PA001048.

E. Horwitz, M. Dietz, and R. Chiarizia. The application of novel extraction chromatographic materials to the characterization of radioactive waste solutions. *Journal of radioanalytical and nuclear ...*, 161(2):575–583, aug 1992. doi:10.1007/BF02040504.

C. Hu, J. Huang, N. Fang, S. Xie, G. M. Henderson, and Y. Cai. Adsorbed silica in stalagmite carbonate and its relationship to past rainfall. *Geochimica et Cosmochimica Acta*, 69(9):2285–2292, may 2005. doi:10.1016/j.gca.2004.10.029.

C. Hu, G. Henderson, J. Huang, Z. Chen, and K. Johnson. Report of a three-year monitoring programme at Heshang Cave, Central China. *International Journal of Speleology*, 37(3):143–151, nov 2008a. doi:10.5038/1827-806X.37.3.1.

C. Hu, G. M. Henderson, J. Huang, S. Xie, Y. Sun, and K. R. Johnson. Quantification of Holocene Asian monsoon rainfall from spatially separated cave records. *Earth and Planetary Science Letters*, 266(3-4):221–232, feb 2008b. doi:10.1016/j.epsl.2007.10.015.

Y. Huang and I. J. Fairchild. Partitioning of Sr2+ and Mg2+ into calcite under karst-analogue experimental conditions. *Geochimica et Cosmochimica Acta*, 65(1):47–62, jan 2001. doi:10.1016/S0016-7037(00)00513-5.

Y. Huang, I. J. Fairchild, A. Borsato, S. Frisia, N. J. Cassidy, F. McDermott, and C. J. Hawkesworth. Seasonal variations in Sr, Mg and P in modern speleothems (Grotta di Ernesto, Italy). *Chemical Geology*, 175(3-4):429–448, jun 2001. doi:10.1016/S0009-2541(00)00337-5.

A. D. Jacobson and C. Holmden. $\delta$44Ca evolution in a carbonate aquifer and its bearing on the equilibrium isotope fractionation factor for calcite. *Earth and Planetary Science Letters*, 270(3-4):349–353, 2008. doi:10.1016/j.epsl.2008.03.039.

K. Johnson, C. Hu, N. Belshaw, and G. Henderson. Seasonal trace-element and stable-isotope variations in a Chinese speleothem: The potential for high-resolution paleomonsoon reconstruction. *Earth and Planetary Science Letters*, 244(1-2):394–407, apr 2006. doi:10.1016/j.epsl.2006.01.064.

M. Jungck, T. Shimamura, and G. Lugmair. Ca isotope variations in Allende. *Geochimica et Cosmochimica Acta*, 48(12):2651–2658, 1984. doi:10.1016/0016-7037(84)90313-2.

T. Kursar, S. J. Wright, and R. Radulovich. The Effects of the Rainy-Season and Irrigation on Soil-Water and Oxygen in a Seasonal Forest in Panama. *Journal of Tropical Ecology*, 11(May 2017):497–515, 1995.

D. Laxen and R. Harrison. Cleaning Methods for Polythene Containers Prior to the Determination of Trace Metals in Freshwater Samples. *Analytical Chemistry*, 53:345–350, 1981. doi:10.1021/ac00225a051.

D. Lemarchand, G. Wasserburg, and D. Papanastassiou. Rate-controlled calcium isotope fractionation in synthetic calcite. *Geochimica et Cosmochimica Acta*, 68(22):4665–4678, nov 2004. doi:10.1016/j.gca.2004.05.029.

X. Li, C. Hu, J. Huang, S. Xie, and A. Baker. A 9000-year carbon isotopic record of acid-soluble organic matter in a stalagmite from Heshang Cave, central China: Paleoclimate implications. *Chemical Geology*, 388:71–77, nov 2014. doi:10.1016/j.chemgeo.2014.08.029.

Y.-h. Liu, G. M. Henderson, C.-y. Hu, A. J. Mason, N. Charnley, K. R. Johnson, and S.-c. Xie. Links between the East Asian monsoon and North Atlantic climate during the 8,200 year event. *Nature Geoscience*, 6(2):117–120, jan 2013. doi:10.1038/ngeo1708.

K. Ljung, S. Björck, H. Renssen, and D. Hammarlund. South Atlantic island record reveals a South Atlantic response to the 8.2 kyr event. *Climate of the Past*, 4(1):35–45, feb 2008. doi:10.5194/cp-4-35-2008.

C. S. Marriott. *Lithium and Calcium Isotope Fractionation and Li/Ca Ratio Incorporation into Calcium Carbonate as Potential Geochemical Proxies*. PhD thesis, Oxford, 2004.

B. Marshall and D. DePaolo. Precise age determinations and petrogenetic studies using the K-Ca method. *Geochimica et Cosmochimica Acta*, 46(12):2537–2545, 1982. doi:10.1016/0016-7037(82)90376-3.

D. P. Mattey, I. J. Fairchild, T. C. Atkinson, J.-P. Latin, M. Ainsworth, and R. Durell. Seasonal microclimate control of calcite fabrics, stable isotopes and trace elements in modern speleothem from St Michaels Cave, Gibraltar. *Geological Society, London, Special Publications*, 336(1):323–344, jun 2010. doi:10.1144/SP336.17.

A. Moreno, H. Stoll, M. Jiménez-Sánchez, I. Cacho, B. Valero-Garcés, E. Ito, and R. L. Edwards. A speleothem record of glacial (25-11.6kyr BP) rapid climatic changes from northern Iberian Peninsula. *Global and Planetary Change*, 71(3-4):218–231, apr 2010. doi:10.1016/j.gloplacha.2009.10.002.

C. Morrill, A. Wagner, B. Otto-Bliesner, and N. Rosenbloom. Evidence for significant climate impacts in monsoonal Asia at 8.2 ka from multiple proxies and model simulations. *Journal of Earth Environment*, 2:426–441, 2011.

J. W. Morse and M. L. Bender. Partition coefficients in calcite: Examination of factors influencing the validity of experimental results and their application to natural systems. *Chemical Geology*, 82:265–277, jan 1990. doi:10.1016/0009-2541(90)90085-L.

C. Mühlinghaus, D. Scholz, and A. Mangini. Modelling stalagmite growth and $\delta$13C as a function of drip interval and temperature. *Geochimica et Cosmochimica Acta*, 71(11): 2780–2790, jun 2007. doi:10.1016/j.gca.2007.03.018.

C. Mühlinghaus, D. Scholz, and A. Mangini. Modelling fractionation of stable isotopes in stalagmites. *Geochimica et Cosmochimica Acta*, 73(24):7275–7289, dec 2009. doi:10.1016/j.gca.2009.09.010.

F. R. Niederer and D. Papanastassiou. Ca isotopes in refractory inclusions. *Geochimica et Cosmochimica Acta*, 48(6):1279–1293, 1984. doi:10.1016/0016-7037(84)90062-0.

L. C. Nielsen, D. J. DePaolo, and J. J. De Yoreo. Self-consistent ion-by-ion growth model for kinetic isotopic fractionation during calcite precipitation. *Geochimica et Cosmochimica Acta*, 86:166–181, 2012. doi:10.1016/j.gca.2012.02.009.

A. L. Noronha, K. R. Johnson, C. Hu, J. Ruan, J. R. Southon, and J. E. Ferguson. Assessing influences on speleothem dead carbon variability over the Holocene: Implications for speleothem-based radiocarbon calibration. *Earth and Planetary Science Letters*, 394:20–29, may 2014. doi:10.1016/j.epsl.2014.03.015.

A. L. Noronha, K. R. Johnson, J. R. Southon, C. Hu, J. Ruan, and S. McCabe-Glynn. Radiocarbon evidence for decomposition of aged organic matter in the vadose zone as the main source of speleothem carbon. *Quaternary Science Reviews*, 127:37–47, 2015. doi:10.1016/j.quascirev.2015.05.021.

D. Nürnberg, J. Bijma, and C. Hemleben. Assessing the reliability of magnesium in foraminiferal calcite as a proxy for water mass temperatures. *Geochimica et Cosmochimica Acta*, 60(5):803–814, mar 1996. doi:10.1016/0016-7037(95)00446-7.

J. R. O'Neil and L. H. Adami. Oxygen isotope partition function ratio of water and the structure of liquid water. *The Journal of Physical Chemistry*, 73(5):1553–1558, may 1969. doi:10.1021/j100725a062.

R. Owen, C. Day, C.-Y. Hu, Y.-H. Liu, M. Pointing, C. Blättler, and G. Henderson. Calcium isotopes in caves as a proxy for aridity: Modern calibration and application to the 8.2 kyr event. *Earth and Planetary Science Letters*, 443:129–138, jun 2016. doi:10.1016/j.epsl.2016.03.027.

D. L. Parkhurst and C. Appelo. User's guide to PHREEQC (Version 2)-A computer program for speciation, batch-reaction, one-dimensional transport, and inverse geochemical calculations. *Water-Resources Investigations Report*, 99(4259), 1999.

D. L. Parkhurst and C. Appelo. Description of Input and Examples for PHREEQC Version 3-A Computer Program for Speciation, Batch-Reaction, One-Dimensional Transport, and Inverse Geochemical Calculations. *Chapter 43 of Section 6, Groundwater Book 6, Modeling Techniques*, 2012.

F. S. R. Pausata, D. S. Battisti, K. H. Nisancioglu, and C. M. Bitz. Chinese stalagmite $\delta$18O controlled by changes in the Indian monsoon during a simulated Heinrich event. *Nature Geoscience*, 4(7):474–480, jun 2011. doi:10.1038/ngeo1169.

C. Pin and C. Bassin. Evaluation of a strontium-specific extraction chromatographic method for isotopic analysis in geological materials. *Analytica Chimica Acta*, 269(2): 249–255, nov 1992. doi:10.1016/0003-2670(92)85409-Y.

C. Pin, D. Briot, C. Bassin, and F. Poitrasson. Concomitant separation of strontium and samarium-neodymium for isotopic analysis in silicate samples, based on specific extraction chromatography. *Analytica Chimica Acta*, 298(2):209–217, nov 1994. doi:10.1016/0003-2670(94)00274-6.

J. Pross, U. Kotthoff, U. Muller, O. Peyron, I. Dormoy, G. Schmiedl, S. Kalaitzidis, and A. Smith. Massive perturbation in terrestrial ecosystems of the Eastern Mediterranean region associated with the 8.2 kyr B.P. climatic event. *Geology*, 37(10):887–890, oct 2009. doi:10.1130/G25739A.1.

L. Reynard, C. Day, and G. Henderson. Large fractionation of calcium isotopes during cave-analogue calcium carbonate growth. *Geochimica et Cosmochimica Acta*, 75(13): 3726–3740, jul 2011. doi:10.1016/j.gca.2011.04.010.

S. Riechelmann, D. Buhl, A. Schröder-Ritzrau, D. F. C. Riechelmann, D. K. Richter, H. B. Vonhof, J. a. Wassenburg, A. Geske, C. Spötl, and A. Immenhauser. The magnesium isotope record of cave carbonate archives. *Climate of the Past*, 8(6):1849–1867, nov 2012. doi:10.5194/cp-8-1849-2012.

M. S. Roberts, P. L. Smart, and A. Baker. Annual trace element variations in a Holocene speleothem. *Earth and Planetary Science Letters*, 154(1-4):237–246, jan 1998. doi:10.1016/S0012-821X(97)00116-7.

C. S. Romanek, E. L. Grossman, and J. W. Morse. Carbon isotopic fractionation in synthetic aragonite and calcite: Effects of temperature and precipitation rate. *Geochimica et Cosmochimica Acta*, 56(1):419–430, jan 1992. doi:10.1016/0016-7037(92)90142-6.

S. J. Romaniello, M. P. Field, H. B. Smith, G. W. Gordon, M. H. Kim, and A. D. Anbar. Fully automated chromatographic purification of Sr and Ca for isotopic analysis. *J. Anal. At. Spectrom.*, 30(9):1906–1912, 2015. doi:10.1039/C5JA00205B.

Y. Rosenthal, M. P. Field, and R. M. Sherrell. Precise determination of element/calcium ratios in calcareous samples using sector field inductively coupled plasma mass spectrometry. *Analytical chemistry*, 71(15):3248–53, aug 1999. doi:10.1021/ac981410x.

W. Russell, D. Papanastassiou, and T. Tombrello. Ca isotope fractionation on the Earth and other solar system materials. *Geochimica et Cosmochimica Acta*, 42(8):1075–1090, 1978. doi:10.1016/0016-7037(78)90105-9.

M. Schiller, C. Paton, and M. Bizzarro. Calcium isotope measurement by combined HR-MC-ICPMS and TIMS. *Journal of Analytical Atomic Spectrometry*, 27(1):38, jan 2012. doi:10.1039/c1ja10272a.

D. Scholz, C. Mühlinghaus, and A. Mangini. Modelling $\delta$13C and $\delta$18O in the solution layer on stalagmite surfaces. *Geochimica et Cosmochimica Acta*, 73(9):2592–2602, may 2009. doi:10.1016/j.gca.2009.02.015.

H. P. Schwarcz, R. S. Harmon, P. Thompson, and D. C. Ford. Stable isotope studies of fluid inclusions in speleothems and their paleoclimatic significance. *Geochimica et Cosmochimica Acta*, 40(6):657–665, jun 1976. doi:10.1016/0016-7037(76)90111-3.

F. Schwartz and H. Zhang. Theory of Ground-Water Flow in Unsaturated Zones and Fractured Media. In *Fundamentals of Groundwater*, chapter 6, page 592. Wiley, 2002. ISBN 978-0471137856.

C.-Y. Shih, L. Nyquist, D. Bogard, and H. Wiesmann. K-Ca and Rb-Sr dating of two lunar granites: Relative chronometer resetting. *Geochimica et Cosmochimica Acta*, 58 (14):3101–3116, 1994. doi:10.1016/0016-7037(94)90182-1.

D. J. Sinclair. Two mathematical models of Mg and Sr partitioning into solution during incongruent calcite dissolution. *Chemical Geology*, 283(3-4):119–133, apr 2011. doi:10.1016/j.chemgeo.2010.05.022.

D. J. Sinclair, J. L. Banner, F. W. Taylor, J. Partin, J. Jenson, J. Mylroie, E. Goddard, T. Quinn, J. Jocson, and B. Miklavič. Magnesium and strontium systematics in tropical speleothems from the Western Pacific. *Chemical Geology*, 294-295:1–17, feb 2012. doi:10.1016/j.chemgeo.2011.10.008.

J. Skulan and D. J. DePaolo. Calcium isotope fractionation between soft and mineralized tissues as a monitor of calcium use in vertebrates. *Proceedings of the National Academy of Sciences of the United States of America*, 96(24):13709–13713, 1999. doi:10.1073/pnas.96.24.13709.

J. Skulan, D. J. DePaolo, and T. L. Owens. Biological control of calcium isotopic abundances in the global calcium cycle. *Geochimica et Cosmochimica Acta*, 61(12):2505–2510, jun 1997. doi:10.1016/S0016-7037(97)00047-1.

I. Spooner, M. S. V. Douglas, and L. Terrusi. Multiproxy evidence of an early Holocene (8.2 kyr) climate oscillation in central Nova Scotia, Canada. *Journal of Quaternary Science*, 17(7):639–645, oct 2002. doi:10.1002/jqs.716.

C. Spötl, J. Fohlmeister, H. Cheng, and R. Boch. Modern aragonite formation at near-freezing conditions in an alpine cave, Carnic Alps, Austria. *Chemical Geology*, 435: 60–70, oct 2016. doi:10.1016/j.chemgeo.2016.04.017.

T. Steuber and D. Buhl. Calcium-isotope fractionation in selected modern and ancient marine carbonates. *Geochimica et Cosmochimica Acta*, 70(22):5507–5521, 2006. doi:10.1016/j.gca.2006.08.028.

H. M. Stoll, W. Müller, and M. Prieto. I-STAL, a model for interpretation of Mg/Ca, Sr/Ca and Ba/Ca variations in speleothems and its forward and inverse application on seasonal to millennial scales. *Geochemistry, Geophysics, Geosystems*, 13(1), sep 2012. doi:10.1029/2012GC004183.

M. Stuiver and H. A. Polach. Discussion: Reporting of 14C Data. *Radiocarbon*, 19(3): 355–363, 1977. doi:10.1021/ac100494m.

A. Talma and J. C. Vogel. Late Quaternary Paleotemperatures Derived from a Speleothem from Cango Caves, Cape Province, South Africa. *Quaternary Research*, 37(02):203–213, mar 1992. doi:10.1016/0033-5894(92)90082-T.

J. Tang, M. Dietzel, F. Böhm, S. J. Köhler, and A. Eisenhauer. Sr2+/Ca2+ and 44Ca/40Ca fractionation during inorganic calcite formation: II. Ca isotopes. *Geochimica et Cosmochimica Acta*, 72(15):3733–3745, aug 2008. doi:10.1016/j.gca.2008.05.033.

E. Tipper, A. Galy, and M. Bickle. Riverine evidence for a fractionated reservoir of Ca and Mg on the continents: Implications for the oceanic Ca cycle. *Earth and Planetary Science Letters*, 247(3-4):267–279, jul 2006. doi:10.1016/j.epsl.2006.04.033.

A. F. Tooth and I. J. Fairchild. Soil and karst aquifer hydrological controls on the geochemical evolution of speleothem-forming drip waters, Crag Cave, southwest Ireland. *Journal of Hydrology*, 273(1-4):51–68, mar 2003. doi:10.1016/S0022-1694(02)00349-9.

P. C. Treble, J. Chappell, and J. M. G. Shelley. Complex speleothem growth processes revealed by trace element mapping and scanning electron microscopy of annual layers. *Geochimica et Cosmochimica Acta*, 69(20):4855–4863, 2005. doi:10.1016/j.gca.2005.06.008.

D. M. Tremaine and P. N. Froelich. Speleothem trace element signatures: A hydrologic geochemical study of modern cave dripwaters and farmed calcite. *Geochimica et Cosmochimica Acta*, 121:522–545, nov 2013. doi:10.1016/j.gca.2013.07.026.

Y. Wang, H. Cheng, R. L. Edwards, X. Kong, X. Shao, S. Chen, J. Wu, X. Jiang, X. Wang, and Z. An. Millennial- and orbital-scale changes in the East Asian monsoon over the past 224,000 years. *Nature*, 451(7182):1090–1093, feb 2008. doi:10.1038/nature06692.

Y. J. Wang, H. Cheng, R. Edwards, Z. S. An, J. Wu, C.-C. Shen, and J. Dorale. A High-Resolution Absolute-Dated Late Pleistocene Monsoon Record from Hulu Cave, China. *Science*, 294(5550):2345–2348, dec 2001. doi:10.1126/science.1064618.

E. Watson. A conceptual model for near-surface kinetic controls on the trace-element and stable isotope composition of abiogenic calcite crystals. *Geochimica et Cosmochimica Acta*, 68(7):1473–1488, apr 2004. doi:10.1016/j.gca.2003.10.003.

F. Wombacher, A. Eisenhauer, A. Heuser, and S. Weyer. Separation of Mg, Ca and Fe from geological reference materials for stable isotope ratio analyses by MC-ICP-MS and double-spike TIMS. *Journal of Analytical Atomic Spectrometry*, 24(5):627, apr 2009. doi:10.1039/b820154d.

C. I. Wong, J. L. Banner, and M. Musgrove. Seasonal dripwater Mg/Ca and Sr/Ca variations driven by cave ventilation: Implications for and modeling of speleothem paleoclimate records. *Geochimica et Cosmochimica Acta*, 75(12):3514–3529, jun 2011. doi:10.1016/j.gca.2011.03.025.

Z. Xia, X. Kong, X. Jiang, and H. Cheng. Precise dating of East-Asian-Monsoon D/O events during 95-56 ka BP: Based on stalagmite data from Shanbao Cave at Shennongjia, China. *Science in China Series D: Earth Sciences*, 50(2):228–235, feb 2007. doi:10.1007/s11430-007-2029-x.

H. Yan, A.-D. Schmitt, Z. Liu, S. Gangloff, H. Sun, J. Chen, and F. Chabaux. Calcium isotopic fractionation during travertine deposition under different hydrodynamic conditions: Examples from Baishuitai (Yunnan, SW China). *Chemical Geology*, 426:60–70, may 2016. doi:10.1016/j.chemgeo.2016.02.002.

J. Zhang, P. Quay, and D. Wilbur. Carbon isotope fractionation during gas-water exchange and dissolution of CO2. *Geochimica et Cosmochimica Acta*, 59(1):107–114, jan 1995. doi:10.1016/0016-7037(95)91550-D.

R.-S. Zhang, C. P. Nash, and P. A. Rock. Thermodynamics of calcium-isotope-exchange reactions. 1. Exchange between isotopic calcium carbonates and aqueous calcium ions. *Journal of Physical Chemistry*, 92(13):3989–3993, 1988. doi:10.1021/j100324a060.

# Appendix A

# CaveCalc: Installation Details & Code

## A.1 Installation

CaveCalc is a Python module that may be installed on Windows, Mac OS X and Linux systems. Installation of CaveCalc requires users to execute some simple commands using the Terminal (Mac/Linux) or Command Prompt (Windows). To install CaveCalc:

1. Install Python 3
2. Install CaveCalc module
3. Install IPhreeqc COM (Windows only)

### A.1.1 Install Python 3

Python is the programming language used by CaveCalc. It must be installed as a pre-requisite. CaveCalc has been tested and developed using Python 3.5, and is likely compatible with more recent Python 3 versions. Python 2.7 is not supported.

It is advised that users download a scientific Python distribution (e.g. Anaconda), as this will come with various dependencies (e.g. numpy, scipy & matplotlib) pre-installed. Users more familiar with Python may install Python 3.5 as they wish, although a virtual environment is recommended.

To check that Python has been installed correctly, open the Terminal (Mac/Linux) or Command Prompt (Windows) and enter the command:

```
python -V
```

If successful, this command will return the Python version detected. On Mac/Linux systems, the command python3 may be needed instead as python refers to the system version (Python 2.7).

### A.1.2 Install CaveCalc

CaveCalc is distributed as a zip archive. Extract the files. This archive contains the source code for CaveCalc, scripts used to load the GUI, the PHREEQC database used for calculations and several pieces of example code. Run the following command from the extracted directory:

Table A.1: Additional CaveCalc input parameters. These inputs control file IO, modify model output and provide debugging options. * - R(13C) R(18O)_HCO3- R(13C)_HCO3- R(18O)_CO3-2 R(13C)_CO3-2 R(44Ca)_Calcite R(18O)_Calcite R(13C)_Calcite.

| Parameter | Model Name | Default Value |
|---|---|---|
| Log PHREEQC input | phreeqc_log_file | FALSE |
| Output Directory | out_dir | |
| PHREEQC Database Filename | database | oxotope.dat |
| PHREEQC Log Filename | phreeqc_log_file_name | log_{}.phr |
| Totals | totals | |
| Molalities | molalities | HCO3- CO3-2 |
| Isotopes | isotopes | R(44Ca) R(18O) ...* |

```
python setup.py install
```

This command will execute the CaveCalc installation script. Depending on which third party dependencies need to be installed, this may take a while.

### A.1.3 Install IPhreeqc COM (Windows only)

CaveCalc requires an interface with PHREEQC to perform geochemical calculations. On Windows systems, this is provided by the IPhreeqc COM server, available for download from the USGS website (`http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/phreeqc/`).

On Mac OSX (64-bit) and Linux (32-bit) systems, the interface with PHREEQC is handled by the phreeqpy module (installed with CaveCalc) and no further setup is required. Users on other systems (e.g. 64-bit Linux) are advised to follow the phreeqpy installation instructions (available at: `http://www.phreeqpy.com/`) to compile the system-specific .dll/.so file required by phreeqpy and copy this file to the phreeqpy installation directory. It is possible to run CaveCalc using phreeqpy on Windows systems (avoiding use of the IPhreeqc COM server), however it is not advised as the COM server is more memory efficient.

To test the installation, attempt to run one of the examples included in the CaveCalc zip archive. To do this, navigate to the location cavecalc/examples and run the following:

```
python ./example1.py
```

If successful, a plot will appear.

## A.2 Model Input

Chapter 5 describes all scientifically-relevant input options for Cavecalc. A few other options exist to handle file IO, manipulate model output and help with debugging. These are summarised in Table A.1.

'Log PHREEQC input' controls whether to log commands passed from CaveCalc to IPhreeqc. If set to TRUE, CaveCalc will write a log file of PHREEQC input for each

model calculation performed. The log files generated are valid PHREEQC input and may be run through PHREEQC to provide further information about a model run. This is particularly useful for debugging failed model runs, and may also help users better understand the inner workings of Cavecalc. 'PHREEQC Log Filename' determines the file naming convention for any log files created; {} is replaced with an arbitrary number when more than one model is run.

'PHREEQC Database Filename' gives the name of the PHREEQC database to be used. By default this is oxotope.dat (the database distributed with cavecalc). If another suitable database is available, its name may be given here. If the file is located outside the cavecalc/data directory, the full path should be given. 'Output Directory' specifies the location where model output files and any log files will be saved; if left blank the current directory is used.

'Totals', 'Molalities' and 'Isotopes' input parameters are used to customise the chemical data returned by Cavecalc. These inputs are passed to the SELECTED_OUTPUT PHREEQC input block (see PHREEQC manual).

### A.2.1 Graphical User Interface (GUI)

The CaveCalc GUI consists of two parts: the input GUI and output GUI. The scripts for these GUIs are included in the caveCalc archive (in cavecalc/scripts). If the Python scripts directory is on the system path, the GUIs may be run using the commands:

```
cc_input_gui.py
cc_output_gui.py
```

Alternatively they may be run directly from the cavecalc/scripts directory using the commands:

```
python ./cc_input_gui.py
python ./cc_output_gui.py
```

The input GUI provides an interface for all options described in Table A.1 and Chapter 5. For many inputs, clicking the asterisk allows users to quickly generate linearly spaced numerical arrays of multiple input values — useful when you want to run multiple models with systematic variations in one (or more) inputs. When all parameters are set as desired, click 'Run' to run the models. Runs may take some time, depending on the inputs chosen, the Run Mode and the number of models to be calculated. When complete, output files (settings.pkl and results.pkl) will be saved to the specified output directory. The 'Load Output GUI' button provides a shortcut to load the output GUI.

The output GUI (Figure A.1) provides a few simple tools to help visualise and convert the output data between formats. To use, click 'Load Model Output' and select a directory that contains model output (settings.pkl and results.pkl). Once loaded, the number of model runs loaded is displayed in the window; a single directory may contain multiple model runs. Multiple directories may be selected in sequence is you wish to combine their output or visualise them together. Loaded model output may be saved for use in MATLAB or spreadsheet programmes (e.g. Excel) by clicking the 'save as .mat' and 'save as .csv' buttons, respectively. If .csv files are requested, one file will be saved for each model run. If .mat is requested, a single file will be saved containing all settings and results data. Files are saved in the current directory.

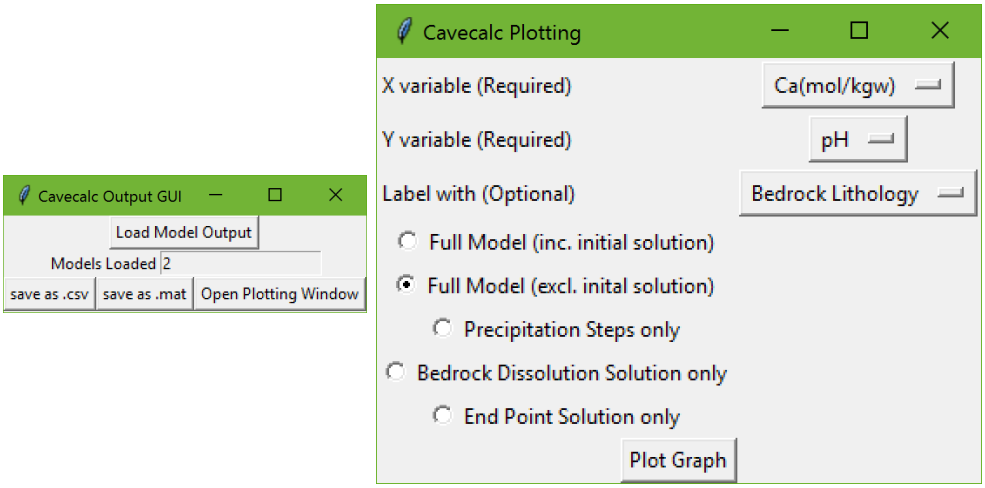



Figure A.1: CaveCalc output GUI. Left: main window. Right: plotting window.

The output GUI also contains options for plotting models. This tool is intended to allow quick visualisation of model output — it is not intended to provide a fully functional plotting programme! To use, once data have been loaded, click 'Open Plotting Window' (Figure A.1). Here you may select model output variables X and Y to be plotted, along with an optional third parameter to label them with. The radio buttons allow users to select which model steps should be plotted.

### A.2.2 Python API

All functionality exposed through the GUI, and more, is available via the Python API. Various examples of how to use CaveCalc within Python are given in the cavecalc/examples directory. In particular, Example 4 demonstrates how users may define their own series of reactions without relying on using a pre-defined Run Mode.

Users interested in fully understanding the workings and capabilities of CaveCalc are advised to read the documented source code (reproduced here in Section A.3). A brief description of each file and the code it contains is given here.

**caves.py** This is the main file. It includes all code for running a single model, including geochemical reactions and interaction with IPhreeqc. Notably, the Solution class provides the API for all available geochemical reactions used by the Run Modes. The Simulator class co-ordinates all interactions with IPhreeqc. Users who wish to understand the operation of CaveCalc are advised to being by reading the Simulator.run() method. The Simulator.run() method is the starting point for all model runs that use a Run Mode; it demonstrates the high level scripting of a CaveCalc model.

**forward_models.py** This file includes the ForwardModels class, which is used to handle the sequential running of multiple models and handle their output. This class provides a clean interface to caves.py that is often used in preference to interacting with caves.py directly; the main exception to this is when running models without a Run Mode (e.g. Example 4).

**analyse.py** This file contains all code for handling model output files, including plotting, filtering and converting file types. It is not an essential component of the model

but may be useful. Some useage of analyse.py is demonstrated in the examples provided.

**setter.py** This file contains code for handling of model input parameters: auto-population of default values, checking entries for type or value errors, and generating suites of model runs where applicable.

**util.py** This file contains utility code to help with a variety of tasks. This includes code for parsing the PHREEQC database, saving log files and performing common unit conversions.

**data/...** This subdirectory contains all static data used by Cavecalc. This includes the PHREEQC database, default and allowed values for model input parameters and various template strings used by caves.py to construct input strings for IPhreeqc.

**gui/...** This subdirectory includes all code for the GUI.

## A.3 Source Code

Directory structure:

```
cavecalc/
    __init__.py
    caves.py
    forward_models.py
    analyse.py
    setter.py
    util.py
    data/
        __init__.py
        defaults.py
        phreeqc_templates.py
        types_and_limits.py
        oxotope.dat
    gui/
        __init__.py
        gui.py
        layout.py
        mapping.py
scripts/
    cc_input_gui.py
    cc_output_gui.py
examples/
    example1.py
    example2.py
    example3.py
    example4.py
```

### A.3.1 cavecalc/__init__.py

```
"""The Cavecalc module provides a model for cave dripwater and carbonate
chemistry.

Author   :    Robert Owen
email    :    robert.owen@earth.ox.ac.uk

Package structure:
    /__init__.py
    /caves.py              - Core geochemical code & IPhreeqc IO
    /forward_models.py     - High-level scripting of Cavecalc models.
    /setter.py             - Utility code to generate model settings.
    /util.py               - Utility code largely relating to file IO.
    /analyse.py            - Analysis and processing of model output.
    /gui/
        /__init__.py
        /gui.py            - GUI Programming
        /layout.py         - Layout data for GUI
        /mapping.py        - Name mapping for gui.py and setter.py
    /data/
        /__init__.py
        /defaults.py               - default model input parameters
        /phreeqc_templates.py      - PHREEQC format strings for caves.py
        /types_and_limits.py       - Model parameter allowable values for setter.py
        /oxotope.dat               - PHREEQC database file
"""

import cavecalc.caves
import cavecalc.setter
import cavecalc.util
import cavecalc.forward_models
import cavecalc.analyse
```

### A.3.2 cavecalc/caves.py

```
"""Performs geochemical calculations and handles IO with IPhreeqc.

This module contains the core functionality of the Cavecalc model. All
geochemical calculations are defined here. Some are also calculated here -
others are delegated to IPhreeqc.

Contains classes:
    Solution    - solution chemistry declaration and reactions
    Gas         - gas phase declaration and calculations
    Carbonate   - carbonate phase definition and calculations
    Simulator   - handles IO with IPhreeqc
"""

import sys
import os
import math
import re
import numpy as np
from decimal import *
import cavecalc.util as ccu
from cavecalc.data.phreeqc_templates import *
from cavecalc.setter import SettingsObject
from copy import copy, deepcopy
import inflection

## phreeqpy mode (phreeqpy_dll, API_com or phreeqpy_com):
# mode may affect speed - author found phreeqpy_dll fastest but mileage may
# vary depending on system.
#   - API_com requires Windows and a full IPhreeqcCOM install
#   - phreeqpy_dll requires phreeqpy with a suitable IPhreeqc .dll/.so file
#   - phreeqpy_com requires Windows, a full IPhreeqcCOM install and phreeqpy
if sys.platform == 'win32':
    MODE = 'API_com'
else:
    MODE = 'phreeqpy_dll'

# edit MODE here to force behaviour

if MODE == 'phreeqpy_com':
    import phreeqpy.iphreeqc.phreeqc_com as phreeqc_mod
elif MODE == 'phreeqpy_dll':
    import phreeqpy.iphreeqc.phreeqc_dll as phreeqc_mod
elif MODE == 'API_com':
    from win32com.client import Dispatch
else:
    raise ValueError('Mode "%s" is not defined.' % MODE)

# Global parameters
EQ_STEP = True                      #   run initial isotope equilibration step
PHREEQC_TOLERANCE = 0.1             #   tolerance on phreeqc percent error
PRESSURE =  1.001                   #   atmospheric pressure (bar)
BEDROCK_PHASE_QZ_LEVEL = 1e-10      #   precision of bedrock phase declaration

# Class definitions
class Solution(object):
    """Performs calculations for aqueous reactions.

    Solution objects handle aqeuous chemistry reactions. Most methods return
    PHREEQC input strings that may be passed to IPhreeqc via a Simulator
    object. The Solution object does not store any data concerning the present
    solution state - that is handled by a Simulator object.

    This is usually instantiated by a Simulator object.

    Selected Methods (return reaction strings):
        Solution.set_soil_chem
        Solution.open_diss_rxn
        Solution.open_diss_rxn_with_reprecip
        Solution.wri_rxn
```

```
        Solution.full_degas_rxn
        Solution.degas_rxn
    """

    def __init__(self, Sim):
        """Initialise a solution object.

        Args:
            Sim: The Simulator object to provide input settings and handle
            IO with IPhreeqc.
        """

        self.s = Sim
        self.G = Gas(Sim=self.s)
        self.init_O = None # initial solution O concentration
        self.init_C = None # initial solution C concentration
        self.init_pH = None # initial solution pH (improved estimate)
        self.init_13c_aq = None # DIC d13c_aq

        self._get_init_co()
        self._get_init_d13c()

    @property
    def calcite_string(self):
        """Returns an empty calcite solid solution string for Iphreeqc."""

        if hasattr(self, '_calcite_string'):
            return "\nUSE SOLID_SOLUTION 1\n"
        else:
            # create solid solution template
            self._calcite_string = CALCITE_SS
            if self.s.settings['soil_Mg'] > 0 or \
               self.s.settings['bedrock_MgCa'] > 0 or \
               self.s.settings['bedrock_mineral'].capitalize() == 'Dolomite':
                self._calcite_string += SS_MG   # add Mg-bearing members
            if self.s.settings['soil_Sr'] > 0 or \
               self.s.settings['bedrock_SrCa'] > 0:
                self._calcite_string += SS_SR   # add Sr-bearing members
            if self.s.settings['soil_Ba'] > 0 or \
               self.s.settings['bedrock_BaCa'] > 0:
                self._calcite_string += SS_BA   # add Ba-bearing members
            return self._calcite_string

    @property
    def diss_gas_string(self):
        """Returns a IPhreeqc GAS_PHASE definition for the soil gas specified
        by user inputs. This is used for bedrock dissolution reactions.
        """

        if hasattr(self, '_diss_gas'):
            return self._diss_gas.get_gasphase_string()

        DissGas = Gas(Sim=self.s)
        DissGas.set_initial_gas()
        self._diss_gas = DissGas
        return self._diss_gas.get_gasphase_string()

    @property
    def degas_string(self):
        """Prepare PHREEQC input for degassing a small amount of CO2.

        Degassing is unidirectional and uses the REACTION keyword in its
        output. Isotope fractionation factors are taken from the database file.
        The fraction of aqueous CO2 removed is defined in the model settings.

        Returns:
            (pq_str, desc_str) for adding to the iphreeqc buffer.
        """


        co2_change = ( self.s.settings['co2_decrement'] *
                       self.s.get('m_CO2(mol/kgw)') *
                       self.s.get('mass_H2O') )

        a = self.G.calculate_eq_co2(co2_change)

        fmt = { 'molefrac_co2'   :   a[0],
                'molefrac_13C'   :   a[1],
                'molefrac_18O'   :   a[3],
                'molefrac_14C'   :   a[2],
                'gas_moles'      :   -1*co2_change}

        #degas_input = CO2_OUT.format(**fmt) # include radiocarbon (unnecessary)
        degas_input = CO2_OUT2.format(**fmt) # no radiocarbon (greater stability)
        return degas_input

    def _get_init_co(self):
        """Calculate initial solution pH, C and O concentrations.

        This sets up a separate Simulator object to to determine dissolved
        C and O concentrations in the initial solution. This step is
        performed before self._set_init_chem() as PHREEQC cannot determine
        gas-solution equilibrium at the same time as declaring solution isotope
        compositions.
        """

        # get settings
        fmt = { 'temp'   :   self.s.settings['temperature'],
                'ph'     :   self.s.settings['soil_pH'],
                'ca'     :   self.s.settings['soil_Ca'],
                'mg'     :   self.s.settings['soil_Mg'],
                'sr'     :   self.s.settings['soil_Sr'],
                'ba'     :   self.s.settings['soil_Ba'],
                'cl'     :   2*( self.s.settings['soil_Ca'] +
                                 self.s.settings['soil_Mg'] +
                                 self.s.settings['soil_Sr'] +
                                 self.s.settings['soil_Ba'] ),
                'co2_si':    math.log10(PRESSURE * 1e-6 * self.s.settings['init_pCO2']) }

        if self.s.settings['init_O2'] == 0:
            fmt['o2_si'] = -10
        else:
            fmt['o2_si'] = math.log10(PRESSURE * self.s.settings['init_O2'])

        # create a dummy PHREEQC solution in a new Simulator to calculate C and
        # O concentrations in initial solution
        dummy_chemistry = DUMMY_CHEM_CO.format(**fmt)
        dummy_settings = self.s.settings.copy()
        dummy_settings['phreeqc_log_file_name'] = 'log_{}_init.phr'
```

```
        dummy_simulator = Simulator(dummy_settings, self.s.id)
        dummy_simulator.ipq_buffer(dummy_chemistry, 'Calculate init C/O concs')
        dummy_simulator.ipq_exec()

        if self.s.settings['init_O2'] == 0:
            self.init_O = 0
        else:
            self.init_O = dummy_simulator.get('O(mol/kgw)')*1000 # initial [O]

        self.init_C = dummy_simulator.get('C(mol/kgw)')*1000 # initial [C]
        self.init_pH = dummy_simulator.get('pH')    # improved estimate of pH

    def _get_init_d13c(self):
        """Calculate the initial solution d13C.

        This requires it's own calculation step because init_d13C is given as
        a gas phase composition - we must calculate the initial solution ratio.

        To calculate the d13C of the solution based on init_d13C (a gas phase
        composition), we do the equilibration between a large gas volume with
        init_d13C and a small amount of solution, initially also at init_d13C.
        From this the 'correct' initial d13C may be calculated.
        """

        t = self.s.settings['temperature']
        fmt = { 'temp'   :   t,
                'ph'     :   self.init_pH,
                'ca'     :   self.s.settings['soil_Ca'],
                'mg'     :   self.s.settings['soil_Mg'],
                'sr'     :   self.s.settings['soil_Sr'],
                'ba'     :   self.s.settings['soil_Ba'],
                'cl'     :   2*( self.s.settings['soil_Ca'] +
                                 self.s.settings['soil_Mg'] +
                                 self.s.settings['soil_Sr'] +
                                 self.s.settings['soil_Ba'] ),
                'c'      :   self.init_C,
                'o'      :   self.init_O,
                'd18O'   :   self.s.settings['atm_d18O'],
                'd13C'   :   self.s.settings['init_d13C'], # guesstimate
                'R14C'   :   self.s.settings['init_R14C'],
                'd44Ca'  :   self.s.settings['soil_d44Ca']}

        dummy_chemistry = (DUMMY_CHEM_D13C.format(**fmt),
                           'soln with guesstimate d13C')

        self.G.set_initial_gas()
        self.G.volume = 1000 # arbitrary large volume
        self.G.name = 'Equilibration Gas'
        gas = self.G.get_gasphase_string()

        dummy_gas = (gas, 'equilibrate with high vol gas /w correct d13C')

        dummy_settings = self.s.settings.copy()
        dummy_settings['phreeqc_log_file_name'] = 'log_{}_d13c.phr'
        dummy_simulator = Simulator(dummy_settings, self.s.id)

        d_conv = [self.s.settings['init_d13C']]
        ph_conv = [self.init_pH]
        c_conv = [self.init_C]

        dummy_simulator.ipq_buffer([dummy_chemistry, dummy_gas])
        dummy_simulator.ipq_exec()

        # note output d18O will be slightly incorrect. Limitation of PHREEQC.
        self.init_13c_aq = dummy_simulator.get('I_R(13C)')

        # if a solution d13C has been explicitly stated, use that value
        if self.s.settings['init_solution_d13c'] != 'n/a':
            self.init_13c_aq = self.s.settings['init_solution_d13c']

    def set_soil_chem(self):
        """Add Initial Solution Chemistry to the phreeqc buffer. Initial
        Solution Chemistry is taken from model input parameters.

        This method normally requires _get_init_co() and _get_init_d13C to be
        run first to set the initial solution C, O and d13C values.

        Returns:
            (pq_str, desc_str) for adding to the iphreeqc buffer.
        """

        # get initial chemistry parameters (assume Cl- for charge balance)
        init_r14c = ccu.pmc_denormalise(    self.s.settings['init_R14C'],
                                            self.init_13c_aq     )

        fmt = { 'temp'   :   self.s.settings['temperature'],
                'ph'     :   self.init_pH,
                'ca'     :   self.s.settings['soil_Ca'],
                'mg'     :   self.s.settings['soil_Mg'],
                'sr'     :   self.s.settings['soil_Sr'],
                'ba'     :   self.s.settings['soil_Ba'],
                'cl'     :   2*( self.s.settings['soil_Ca'] +
                                 self.s.settings['soil_Mg'] +
                                 self.s.settings['soil_Sr'] +
                                 self.s.settings['soil_Ba'] ),
                'c'      :   self.init_C,
                'o'      :   self.init_O,
                'd18O'   :   self.s.settings['atm_d18O'],
                'd13C'   :   self.init_13c_aq,
                'R14C'   :   init_r14c,
                'd44Ca'  :   self.s.settings['soil_d44Ca']}

        self.initial_chemistry = INITIAL_CHEM.format(**fmt)

        if EQ_STEP:
            return self.initial_chemistry + WATER_EQUILIBRATE, 'initial_water'
        else:
            return self.initial_chemistry, 'initial_water'

    def open_diss_rxn(self, Carbonate):
        """Prepare PHREEQC input for open-system dissolution.

        Details of the bedrock and gas phase present are extracted from
        self.sim.settings. By varying gas_volume from 0 (closed system) up to
        <very big numbers>, open vs closed system dissolution may be
        quantified.

        open_diss_rxn with 0 gas volume is equivalent to closed system
        dissolution.
```

```
        Args:
            Carbonate: A Carbonate object defining the solid phase.
        Returns:
            (pq_str, desc_str) for adding to the iphreeqc buffer.
        """

        rxn_string = ""

        if not self.s._bedrock_set:
            rxn_string += Carbonate.phase_definition + '\n'
            self.s._bedrock_set = True

        rxn_string += Carbonate.bedrock_equilibrium1()
        rxn_string += '\n' + self.diss_gas_string
        return rxn_string, 'dissolve bedrock'

    def open_diss_rxn_with_reprecip(self, Carbonate):
        """Prepare PHREEQC input for dissolution with reprecipitation.

        Identical to open_diss_rxn but allows coeval precipitation of calcite
        while bedrock is dissolving. This simulates incongruent bedrock
        dissolution.

        Note that this allows extreme amounts of reprecipitation. For more
        controlled re-precipitation, use open_diss_rxn followed by wri (water-
        rock interactions).

        Args:
            Carbonate: A Carbonate object defining the solid phase.
        Returns:
            (pq_str, desc_str) for adding to the iphreeqc buffer.
        """

        rxn_string = ""

        if not self.s._bedrock_set:
            rxn_string += Carbonate.phase_definition + '\n'
            self.s._bedrock_set = True

        rxn_string += Carbonate.bedrock_equilibrium2()
        rxn_string += '\n' + self.diss_gas_string
        rxn_string += '\n' + self.calcite_string
        return rxn_string, 'dissolve bedrock'

    def wri_rxn(self, Carbonate, m_bedrock=0, m_pyrite=0):
        """Prepare phreeqc input for water-rock interactions.

        The resulting Iphreeqc command allows the current solution to dissolve
        a set amount of bedrock. No capacity is included for gas exchange or
        calcite precipitation - these reactions should be added separately.

        This differs from open_diss_rxn_with_reprecip, which allows
        re-precipitating in the presence of gas.

        Args:
            Carbonate: The Carbonate object to derive bedrock definitions from.
            m_bedrock: Moles of bedrock present. Default is 0.
            m_pyrite: Moles of pyrite present. Default is 0.
        Returns:
            (pq_str, desc_str) for adding to the iphreeqc buffer.
        """

        rxn_string = ""

        if not self.s._bedrock_set:
            rxn_string += Carbonate.phase_definition + '\n'
            self.s._bedrock_set = True

        rxn_string += Carbonate.bedrock_equilibrium2( m_bedrock=m_bedrock,
                                                      m_pyrite=m_pyrite)
        # rxn_string += '\n' + self.calcite_string
        # return rxn_string, 'water-rock equilibration & precipitation'
        return rxn_string, 'water-rock equilibration'

    def eq_degas_rxn(self, EqGas=None):
        """Prepare PHREEQC input for solution-gas CO2 equilibration.

        Args:
            EqGas: A Gas object to equilibrate the solution with. If none is
                   provided, cave air gas is used.
        Returns:
            (pq_str, desc_str) for adding to the iphreeqc buffer.
        """

        if Gas is None:
            EqGas = Gas(Sim=self.s)
            EqGas.set_cave_air()

        rxn_string = EqGas.get_gasphase_string()
        return rxn_string, 'Equilibrium Degassing'

    def kinetic_degas_rxn(self):
        """Prepare PHREEQC input for degassing a small amount of CO2.

        Degassing is unidirectional and uses the REACTION keyword in its
        output. Isotope fractionation factors are taken from the database file.
        The fraction of aqueous CO2 removed is defined in the model settings.

        Returns:
            (pq_str, desc_str) for adding to the iphreeqc buffer.
        """

        return self.degas_string, 'Kinetic Degassing'

    def will_it_degas(self, pCO2, printout=False):
        """Test whether the solution will degas at a given pCO2.

        Args:
            pCO2: gas CO2 concentration (ppmv)
            printout (logical): If true, prints the value of Solution pCO2
        Returns:
            True/False
        """

        temp          = self.s.get('temp(C)') + 273.15
        water         = self.s.get('mass_H2O')
        soln_CO2      = self.s.get('m_CO2(mol/kgw)')
        log_k         = self.s.reader.get_k_values(['CO2\(g\)'], temp)['analytic_value']
        eq_CO2        = soln_CO2 / 10**log_k
        gas_CO2       = pCO2 * 1e-6
```

```
        self.co2_change = self.s.settings['co2_decrement'] * soln_CO2 * water

        if printout:
            print("Water pCO2 (ppmv):\t%f"    %(eq_CO2 * 1e6))
            print("Gas phase pCO2 (ppmv):\t%f" % pCO2)

        if eq_CO2 < gas_CO2:
            if __name__ == '__main__':
                print("Water pCO2 (mol/kgw):\t%f" % soln_CO2)
                print("Water pCO2 (ppmv):\t%f"    %(eq_CO2 * 1e6))
                print("Gas phase pCO2 (ppmv):\t%f" % pCO2)
                print("Degassing complete.")
            return False
        else:
            return True

    def precipitate_rxn(self):
        """Prepare PHREEQC input for calcite precipitation.

        Precipitation will occur until the solution reaches saturation with
        respect to calcite. Radiocarbon fractionation is not included as the
        very small amounts present may cause numerical instability.


        Returns:
            (pq_str, desc_str) for adding to the iphreeqc buffer.
        """

        return self.calcite_string, "CaCO3_precipitation"

class Carbonate(object):
    """Defines bedrock chemistry.

    Solid phase isotope assemblages are not easily expressed natively in
    PHREEQC. This class allows isotopically-distinct calcite and dolomite
    mineralogies to be defined as PHREEQC phases for the purposes of
    dissolution reactions.

    Carbonate objects take their geochemical definitions from the settings
    of a Simulator object.

    Selected Attributes:
        phase_definition - bedrock phase definition string

    Selected Methods:
        bedrock_equilibrium1 - bedrock dissolution string
        bedrock_equilibrium2 - bedrock dissolution string (alternative)
    """

    def __init__(self, Sim):
        """Constructs PHREEQC input for isotope-defined bedrock.

        Args:
            Sim: The Simulator object to draw input from.

        Returns:
            Nothing
        """

        self.s = Sim

        if self.s.settings['bedrock_mineral'].capitalize() == 'Calcite':
            # self._define_bedrock('Calcite')
            self._get_calcite_phase()
        elif self.s.settings['bedrock_mineral'].capitalize() == 'Dolomite':
            self._get_dolomite_phase()
        else:
            raise TypeError("Bedrock mineralogy not recognised: %s" %
                                self.s.settings['bedrock_mineral'])

        # self.check_charge_balance(self.diss_reaction)

    @property
    def phase_definition(self):
        if hasattr(self, '_pd'):
            return self._pd

        self._pd = BEDROCK_PHASES.format( self.pq_phase_entry )
        return self._pd

    def bedrock_equilibrium1(self, m_bedrock=None, m_pyrite=None):
        """Returns a PHREEQC input string for bedrock dissolution.

        This method should not be used for incongruent dissolution reactions as
        the name 'Calcite' may not be used twice in the same PHREEQC step. In
        such cases use bedrock_equilibrium2.

        Args:
            m_bedrock: Moles of bedrock present during dissolution. Default
                       uses model input parameters.
            m_pyrite: Moles of pyrite present during dissolution. Default
                      uses model input parameters.
        Returns:
            A PHREEQC input string specifying a bedrock dissolution reaction.
        """

        if m_bedrock is None:
            m_bedrock = self.s.settings['bedrock']
        if m_pyrite is None:
            m_pyrite = self.s.settings['bedrock_pyrite']

        eq_string = BEDROCK_DISS1.format( moles_bedrock = m_bedrock,
                                          moles_pyrite = m_pyrite )
        return eq_string

    def bedrock_equilibrium2(self, m_bedrock=None, m_pyrite=None):
        """Returns a PHREEQC input string for bedrock dissolution.

        Alternative to bedrock_equilibrium1 when Calcite is present during the
        same reaction step (most commonly during incongruent dissolution
        calculations).

        Args:
            m_bedrock: Moles of bedrock present during dissolution. Default
                       uses model input parameters.
            m_pyrite: Moles of pyrite present during dissolution. Default
                      uses model input parameters.
        Returns:
            A PHREEQC input string specifying a bedrock dissolution reaction.
        """

        if m_bedrock is None:
```

```
            m_bedrock = self.s.settings['bedrock']
        if m_pyrite is None:
            m_pyrite = self.s.settings['bedrock_pyrite']

        eq_string = BEDROCK_DISS2.format( moles_bedrock = m_bedrock,
                                          moles_pyrite = m_pyrite )
        return eq_string

    def _get_calcite_phase(self):

        def dcqz(decimal):
            d2 = decimal.quantize( Decimal(str(BEDROCK_PHASE_QZ_LEVEL)),
                                   rounding=ROUND_HALF_UP)
            return d2

        bed_44Ca = self.s.settings['bedrock_d44Ca']
        bed_13C = self.s.settings['bedrock_d13C']
        bed_18O = self.s.settings['bedrock_d18O']
        MgCa = Decimal(self.s.settings['bedrock_MgCa'] * 0.001)
        SrCa = Decimal(self.s.settings['bedrock_SrCa'] * 0.001)
        BaCa = Decimal(self.s.settings['bedrock_BaCa'] * 0.001)
        a = Decimal(str(self.s.stnd44Ca * (1 + 0.001*bed_44Ca))) # 44/40 ratio
        b = Decimal(str(self.s.stnd13C * (1 + 0.001*bed_13C))) # 13/12 ratio
        c = Decimal(str(self.s.stnd18O * (1 + 0.001*bed_18O))) # 18/16 ratio

        # metal totals
        tCa = 1 / (1+MgCa+SrCa+BaCa)
        tMg = dcqz(MgCa * tCa)
        tSr = dcqz(SrCa * tCa)
        tBa = dcqz(BaCa * tCa)

        # isotope totals
        Ca44 = dcqz(a/(1+a) * tCa)
        Ca40 = dcqz(tCa - Ca44)
        C13 = dcqz(b/(1+b))
        C12 = dcqz(1 - C13)
        O18 = dcqz(c/(1+c) * 3)
        O16 = dcqz(3 - O18)

        # construct phase definition
        metals = "Ca{Ca40}[44Ca]{Ca44}".format(Ca40=Ca40, Ca44=Ca44)
        d = "{}Ca+2 + {}[44Ca]+2 + {}[13C]O3-2 + {}CO2[18O]-2 + {}CO3-2"
        d = d.format(Ca40, Ca44, C13, O18, C12-O18)

        if MgCa > 0:
            metals += "Mg{Mg}".format(Mg=tMg)
            d += " + {}Mg+2".format(tMg)
        if SrCa > 0:
            metals += "Sr{Sr}".format(Sr=tSr)
            d += " + {}Sr+2".format(tSr)
        if BaCa > 0:
            metals += "Ba{Ba}".format(Ba=tBa)
            d += " + {}Ba+2".format(tBa)

        carbonate = "C{}[13C]{}O{}[18O]{}".format( C12, C13, O16, O18 )
        s = metals + carbonate

        v = self.s.reader.get_k_values([r'CaCO3'])

        "{}Mg+2 + {}Sr+2 + {}Ba+2"
        self.diss_reaction = s + ' = ' + d
        self.pq_phase_entry = self.diss_reaction + \
                              '\n\tlog_k\t' + str(v['log_k']) + \
                              '\n\tdelta_h\t' + str(v['delta_h']) + \
                              '\n\t' + str(v['analytic_line'])

    def _get_dolomite_phase(self):

        def dcqz(decimal):
            d2 = decimal.quantize( Decimal(str(BEDROCK_PHASE_QZ_LEVEL)),
                                   rounding=ROUND_HALF_UP)
            return d2

        bed_44Ca = self.s.settings['bedrock_d44Ca']
        bed_13C = self.s.settings['bedrock_d13C']
        bed_18O = self.s.settings['bedrock_d18O']
        SrCa = Decimal(self.s.settings['bedrock_SrCa'] * 0.001)
        BaCa = Decimal(self.s.settings['bedrock_BaCa'] * 0.001)
        a = Decimal(str(self.s.stnd44Ca * (1 + 0.001*bed_44Ca))) # 44/40 ratio
        b = Decimal(str(self.s.stnd13C * (1 + 0.001*bed_13C))) # 13/12 ratio
        c = Decimal(str(self.s.stnd18O * (1 + 0.001*bed_18O))) # 18/16 ratio

        # metal totals
        tCa = 2 / (2+SrCa+BaCa)
        tMg = dcqz(2 / (2+SrCa+BaCa))
        tSr = dcqz(SrCa * tCa)
        tBa = dcqz(BaCa * tCa)

        # isotope totals
        Ca44 = dcqz(a/(1+a) * tCa)
        Ca40 = dcqz(tCa - Ca44)
        C13 = dcqz(2*b/(1+b))
        C12 = dcqz(2 - 2*b/(1+b))
        O18 = dcqz(c/(1+c) * 6)
        O16 = dcqz(6 - c/(1+c) * 6)

        # construct phase definition
        metals = "Ca{Ca40}[44Ca]{Ca44}".format(Ca40=Ca40, Ca44=Ca44)
        metals += "Mg{}".format(tMg)

        d = "{}Ca+2 + {}[44Ca]+2 + {}Mg+2 + {}[13C]O3-2 + {}CO2[18O]-2 + {}CO3-2"
        d = d.format(Ca40, Ca44, tMg, C13, O18, C12-O18)

        if SrCa > 0:
            metals += "Sr{}".format(tSr)
            d += " + {}Sr+2".format(tSr)
        if BaCa > 0:
            metals += "Ba{}".format(tBa)
            d += " + {}Ba+2".format(tBa)

        carbonate = "C{C12}[13C]{C13}O{O16}[18O]{O18}".format(
                        C12=C12, C13=C13, O16=O16, O18=O18
                        )

        s = metals + carbonate


        v = self.s.reader.get_k_values([r"CaMg\(CO3\)2"])
```

```
        self.diss_reaction = s + ' = ' + d
        self.pq_phase_entry = self.diss_reaction + \
                              '\n\tlog_k\t' + str(v['log_k']) + \
                              '\n\tdelta_h\t' + str(v['delta_h'])

    @classmethod
    def check_charge_balance(reaction_string):
        """Checks a PHREEQC dissolution string for charge balance.

        The string checked must be of the form SOLID = A + B + C where A, B, C
        are aqueous ions and SOLID is a single solid phase. This form is common
        in PHASES definitions where a mineral dissolution reaction is
        specified.

        This method is primarly used as a debugging tool to check the strings
        created by _get_calcite_phase() and _get_dolomite_phase() for
        isotopically and trace-element distinct bedrock phases.

        Args:
            reaction_string (str) A phreeqc-valid mineral dissolution reaction
                string.
        """

        # Carbonate oxidation states
        E = {   'Ca'    :   2,
                '[44Ca]':   2,
                'C'     :   4,
                '[13C]' :   4,
                'O'     :   -2,
                '[18O]' :   -2,
                'Mg'    :   2,
                'Sr'    :   2,
                'Ba'    :   2
            }

        print("\n" + reaction_string)
        lhs, rhs = [s.strip() for s in reaction_string.split('=')]

        a = []
        b = []

        r = re.match(r"(\[?[0-9]*[A-Z][a-z]*?\]?)([0-9]\.[0-9]*)", lhs)
        while r:
            a.append(r.groups())
            lhs = lhs.replace(r.group(0),'')
            r = re.match(r"(\[?[0-9]*[A-Z][a-z]*?\]?)([0-9]\.[0-9]*)", lhs)

        for species in [s.strip() for s in rhs.split(' + ')]:
            r = re.compile(r"([0-9]\.[0-9]*)(.*?)([\+\-][0-9])")
            m = re.match(r, species)
            b.append(m.groups())

        lhs_charge = 0
        rhs_charge = 0

        for m, n in a:
            lhs_charge += E[m] * Decimal(n)

        for m, _, n in b:
            rhs_charge += Decimal(m) * Decimal(n)

        print(lhs_charge)
        print(rhs_charge)
        if lhs_charge == rhs_charge:
            print("Equation is charge balanced!")
        else:
            print("Equation is NOT charge balanced :(")

class Gas(object):
    def __init__(self, pCO2=0, O2=0, d13C=0, d18O=0, R14C=100,
                 temperature=None, volume=0, name='Gas', Sim=None):

        self.pCO2 = pCO2
        self.O2 = O2
        self.d13C = d13C
        self.d18O = d18O
        self.R14C = R14C
        self.volume = volume
        self.name = name
        self.s = Sim

        if temperature is None:
            self.temperature = self.s.settings['temperature']
        else:
            self.temperature = temperature

    def set_soil_gas(self):
        """Set self to hold soil gas end-member chemistry."""

        self.pCO2 = self.s.settings['soil_pCO2']
        self.d13C = self.s.settings['soil_d13C']
        self.O2 = self.s.settings['soil_O2']
        self.d18O = self.s.settings['atm_d18O']
        self.R14C = self.s.settings['soil_R14C']
        self.volume = self.s.settings['gas_volume']
        self.name = "Soil Gas End-member"

    def set_atmosphere(self):
        """Set self to hold atmospheric chemistry."""

        self.pCO2 = self.s.settings['atm_pCO2']
        self.d13C = self.s.settings['atm_d13C']
        self.O2 = self.s.settings['atm_O2']
        self.d18O = self.s.settings['atm_d18O']
        self.R14C = self.s.settings['atm_R14C']
        self.volume = 1000 # arbitrary large volume by default
        self.name = "Atmosphere End-member"

    def set_initial_gas(self):
        """Set self to hold initial (mixed) soil gas chemistry."""

        self.pCO2 = self.s.settings['init_pCO2']
        self.d13C = self.s.settings['init_d13C']
        self.O2 = self.s.settings['init_O2']
        self.d18O = self.s.settings['atm_d18O']
        self.R14C = self.s.settings['init_R14C']
        self.volume = self.s.settings['gas_volume']
        self.name = "Mixed Initial Gas"

    def set_cave_air(self):
```

```
        """Set self to hold cave air chemistry."""

        self.pCO2 = self.s.settings['cave_pCO2']
        self.d13C = self.s.settings['cave_d13C']
        self.O2 = self.s.settings['cave_O2']
        self.d18O = self.s.settings['cave_d18O']
        self.R14C = self.s.settings['cave_R14C']
        self.volume = self.s.settings['cave_air_volume']
        self.name = "Cave Air"

    def calculate_eq_co2(self, co2_out):
        """Calculate CO2(g) in equilibrium with the current CO2(aq).

        Calculate co2 isotopologue abundances in a gas exolved from a
        solution. Equilibrium is with the CO2(aq) pool only - DIC equilibration
        is not considered. Isotope ratios are calculated via mass balance using
        fractionation factors from the database file.

        Args:
            co2_out: moles of CO2 to be removed from the solution.
        Returns:
            (c12o_2, c13o_2, c14o_2, c12o18o) - tuple of relative abundances
        """

        CO2_aq  = self.s.get('m_CO2(mol/kgw)')
        water   = self.s.get('mass_H2O')
        CO2_13C = self.s.get('I_R(13C)_CO2(aq)')
        CO2_14C = self.s.get('I_R(14C)_CO2(aq)')
        CO2_18O = self.s.get('I_R(18O)_CO2(aq)')
        temp    = self.s.get('temp(C)') + 273.15
        n_co2   = CO2_aq * water

        assert n_co2 > co2_out

        # get CO2(g)-CO2(aq) fractionation factors
        frac13C = self.s.reader.get_alpha( '13C','CO2(g)/CO2(aq)', temp )
        frac18O = self.s.reader.get_alpha( '18O','CO2(g)/CO2(aq)', temp )

        # calculate normalised isotope ratios
        c13 = 1 + 0.001*CO2_13C
        o18 = 1 + 0.001*CO2_18O

        # calculate gas isotopes via mass balance
        c13_g = c13 * n_co2 / (co2_out + frac13C * (n_co2-co2_out))
        o18_g = o18 * n_co2 / (co2_out + frac18O * (n_co2-co2_out))
        d13C_gas = 1000*(c13_g - 1)
        d18O_gas = 1000*(o18_g - 1)

        # get absolute gas isotopologue abundances
        return self.co2_isotopologues(d13C=d13C_gas, d18O=d18O_gas,
                                      R14C=CO2_14C)

    def co2_isotopologues(self, d13C=None, d18O=None, R14C=None):
        """Calculate CO2 isotopologue relative abundances.

        Calculate relative abundances of CO2, [13C]O2, CO[18O] and [14C]O2
        based on delta/ratio values. The values sum to 1. This method uses
        the absolute isotopic composition of the standards from the phreeqc
        database and then solves the mass balance equations for each isotope.

        Args:
            d13C: CO2(g) d13C
            d18O: CO2(g) d18O
            R14C: CO2(g) R14C as a pmc value
        Returns:
            (c12o_2, c13o_2, c14o_2, c12o18o) - tuple of molar abundances
        """

        if d13C is None:
            d13C = self.d13C
        if d18O is None:
            d18O = self.d18O
        if R14C is None:
            R14C = self.R14C

        # calculate absolute isotope ratios (14C from Stuvier & Polach, 1977)
        s13C = self.s.stnd13C
        s18O = self.s.stnd18O
        # s14C = self.s.stnd14C

        r13 = s13C * (1 + 0.001 * d13C) # 13C/12C ratio
        r18 = s18O * (1 + 0.001 * d18O) # 18O/16O ratio
        r14 = ccu.pmc_2_c14(R14C, d13C, self.s.stnd14C)

        # r14 = s14C * 0.01 * R14C * math.pow((1+0.001*float(d13C))/0.975,2)
        # mole fraction 14C

        ## system of equations to solve:
        # r13C*CO2 - 13CO2 + 0*14CO2 + r13C*CO18O = 0 (13C mass balance)
        # r14C*CO2 + r14C*13CO2 + (r14C-1)*14CO2 + r14C*CO18O = 0 (14C)
        # 2r18O*CO2 + 2r18O*13CO2 + 2r18O*14CO2 + (r18O*-1)*CO18O = 0 (18O)
        # CO2 + 13CO2 + 14CO2 + CO18O = 1 (total CO2 mass balance)

        ## method 1: solve as matrix:
        # m =     np.array([  [r13,         -1,           0,          r13],
                            # [r14,        r14,       r14-1,          r14],
                            # [2*r18,   2*r18,        2*r18,      (r18-1)],
                            # [1,           1,            1,           1]])
        # m = m.astype('float_')
        # print(np.linalg.cond(m)) # check matrix condition number
        # n =     np.array([0, 0, 0, 1])
        # x =     np.linalg.solve(m, n)
        # c12o_2 = x[0]
        # c13o_2 = x[1]
        # c14o_2 = x[2]
        # c12o18o = x[3]

        ## method 2: simultaneous equation solution:
        c13o_2 = r13*(1-r14)/(r13+1)
        c14o_2 = r14
        c12o18o = 2*r18/(1+r18)
        c12o_2 = 1 - c13o_2-c14o_2-c12o18o

        # print("Matrix vs simultaneous:") # precision check
        # print("%g\t%g" % (mc12o_2, c12o_2))
        # print("%g\t%g" % (mc13o_2, c13o_2))
        # print("%g\t%g" % (mc14o_2, c14o_2))
        # print("%g\t%g\n" % (mc12o18o, c12o18o))

        return c12o_2, c13o_2, c14o_2, c12o18o
```

```
    def get_gasphase_string(self):
        """Return PHREEQC input defining the object as a CO2-N2 gas phase."""

        # calculate 18O/O ratios in O2(g) and CO2(g)
        t_k = self.temperature + 273.15
        a_o2_h2o =  self.s.reader.get_alpha('18O','O2(g)/H2O(l)', t_k)
        a_co2_h2o = self.s.reader.get_alpha('18O','CO2(g)/H2O(l)', t_k)

        r18o_h2o = self.s.stnd18O * (1 + 0.001 * self.d18O)
        r18o_o2 = r18o_h2o * a_o2_h2o
        r18o_co2 = r18o_h2o * a_co2_h2o
        d18o_co2 = ( r18o_co2/self.s.stnd18O - 1 ) * 1000

        # get CO2 isotopologues
        iso = self.co2_isotopologues( d18O=d18o_co2 )

        # get O2 isotopologues
        o18o16 = 2*r18o_o2 / (1 + r18o_o2)
        o16o16 = 1 - o18o16

        fmt = { 'name'        :   self.name,
                'pressure'    :   PRESSURE,
                'volume'      :   self.volume,
                'temperature' :   self.temperature,
                'pp_co2'      :   PRESSURE * iso[0] * self.pCO2 * 1e-6,
                'pp_C18O'     :   PRESSURE * iso[3] * self.pCO2 * 1e-6,
                'pp_C18O2'    :   0,
                'pp_13C'      :   PRESSURE * iso[1] * self.pCO2 * 1e-6,
                'pp_13C18O'   :   0,
                'pp_13C18O2'  :   0,
                'pp_14C'      :   PRESSURE * iso[2] * self.pCO2 * 1e-6,
                'pp_14C18O'   :   0,
                'pp_14C18O2'  :   0,
                'pp_O2'       :   PRESSURE * self.O2 * o16o16,
                'pp_O18O'     :   PRESSURE * self.O2 * o18o16,
                'pp_N2'       :   PRESSURE * (1 - self.pCO2 * 1e-6 - self.O2) }

        gas_string = CO2_GAS.format(**fmt)
        return gas_string

    def get_empty_gasphase_string(self):
        """Return an empty PHREEQC CO2-N2 gas phase string."""

        fmt = { 'name'        :   '',
                'pressure'    :   PRESSURE,
                'volume'      :   0,
                'temperature' :   self.s.settings['temperature'],
                'pp_co2'      :   0,
                'pp_C18O'     :   0,
                'pp_C18O2'    :   0,
                'pp_13C'      :   0,
                'pp_13C18O'   :   0,
                'pp_13C18O2'  :   0,
                'pp_14C'      :   0,
                'pp_14C18O'   :   0,
                'pp_14C18O2'  :   0,
                'pp_O2'       :   0,
                'pp_O18O'     :   0,
                'pp_N2'       :   0}

        empty_gasphase = CO2_GAS.format(**fmt)

        return empty_gasphase

    def mix_gases(self, Gas2, frac, **kwargs):
        """Mixes two CO2 gas phases and returns the resultant gas.

        Args:
            Gas2: A Gas object to be mixed with self.
            frac: The fraction (0-1) of Gas2 present in the final mixture.
            kwargs: passed to Gas constructor for new mixed Gas
        """

        a = self.co2_isotopologues()
        b = Gas2.co2_isotopologues()

        f1 = (1-frac)*self.pCO2 # total co2 from A
        f2 = frac*Gas2.pCO2     # total co2 from B

        mix_pco2 = f1 + f2
        mix_o2 = (1-frac)*self.O2 + frac*Gas2.O2
        m_12c = (f1*(a[0]+a[3]) + f2*(b[0]+b[3])) / mix_pco2
        m_13c = (f1*a[1] + f2*b[1]) / mix_pco2
        m_14c = (f1*a[2] + f2*b[2]) / mix_pco2

        mix_d13c, mix_r14c = ccu.c14_to_pmc( m_12c, m_13c, m_14c,
                                             self.s.stnd13C, self.s.stnd14C )

        mix = {     'pCO2'          :   mix_pco2,
                    'd13C'          :   mix_d13c,
                    'R14C'          :   mix_r14c,
                    'O2'            :   mix_o2,
                    'temperature'   :   self.temperature,
                    'Sim'           :   self.s,
                    'name'          :   'mixed_gas'
        }

        kwargs.update(mix)
        return Gas(**kwargs)

class Simulator(object):
    """Runs Cavecalc models.

    A Simulator object sits at the centre of a Cavecalc model. It initalises
    IPhreeqc, executes PHREEQC input returned from Solution, Gas and
    Carbonate objects, and handles and stores output from IPhreeqc.

    Simulator objects also control interaction with various utility classes
    defined in util.py.

    Selected Methods:

    """

    def __init__(self, settings, id=0):
        """Initialises an IPhreeqc session.

        Initalising a Simulator object sets up an IPhreeqc session, parses
        the settings provided and initialises various utility objects.
```

```
        Args:
            settings (dict): Contains model settings. This dict should be
                generated using setter.py.
            id (int): Id number for the model being run.
        """

        if type(settings) is not dict:
            try:
                settings = settings.dict()
            except:
                raise TypeError("Settings arg must be of type dict.")

        self.settings = deepcopy(settings)          # will be updated and used
        self.settings_archive = deepcopy(settings) # will not be changed
        self.id = id

        # set up input log if requested
        if self.settings['phreeqc_log_file']:
            l = self.settings['phreeqc_log_file_name']
            k = l.format(self.id)
            db = self.settings['database']
            self.input_log = ccu.PhreeqcInputLog(filename = k,
                                                 dbpath = db)

        # set up database reader to get thermodynamic data & reactions
        self.reader = ccu.DBReader(self.settings['database'])

        self.count = 0
        self._bedrock_set = False
        self.string_buffer = ""
        self.desc_buffer = ""
        self._define_selected_output()
        self._parse_settings()

    def _define_selected_output(self):
        """Defines PHREEQC SELECTED_OUTPUT block."""

        fmt = { 'totals'        :   self.settings['totals'],
                'molalities'    :   self.settings['molalities'],
                'isotopes'      :   self.settings['isotopes']}

        if (self.settings['kinetics_mode'] == 'degas_only' or
            self.settings['kinetics_mode'] == 'diss_only'):
            fmt['saturation_indices'] = ''
        else:
            fmt['saturation_indices'] = '\n\t-saturation_indices Calcite'

        self.selected_output     = SEL_OUTPUT_CORE.format(**fmt)

    def _parse_settings(self):
        """Performs any necessary settings pre-processing.

        1)  Calculate the initial gas (may be a mixing product)
        """

        ae = self.settings['atmo_exchange']
        G1 = Gas(Sim=self)
        G2 = Gas(Sim=self)

        G1.set_soil_gas()
        G2.set_atmosphere()
        G3 = G1.mix_gases(G2, ae)

        # Set soil water carbon & isotope budget, based on inputs
        for p in ['pCO2', 'd13C', 'R14C', 'O2']: # d13C is of CO2(g)
            if self.settings['init_'+p] == 'mix':
                self.settings['init_'+p] = getattr(G3, p)
            elif self.settings['init_'+p] == 'atm':
                self.settings['init_'+p] = getattr(G2, p)
            elif self.settings['init_'+p] == 'soil':
                self.settings['init_'+p] = getattr(G1, p)

    @property
    def stnd13C(self):
        if not hasattr(self, '_stnd13C'):
            self._stnd13C = self.reader.get_iso_stnd('13C')
        return self._stnd13C

    @property
    def stnd14C(self):
        if not hasattr(self, '_stnd14C'):
            self._stnd14C = self.reader.get_iso_stnd('14C')
        return self._stnd14C

    @property
    def stnd18O(self):
        if not hasattr(self, '_stnd18O'):
            self._stnd18O = self.reader.get_iso_stnd('18O')
        return self._stnd18O

    @property
    def stnd44Ca(self):
        if not hasattr(self, '_stnd44Ca'):
            self._stnd44Ca = self.reader.get_iso_stnd('44Ca')
        return self._stnd44Ca

    def _get_selected_output(self):
        """Get results from most recent Iphreeqc step.

        Model parameters returned depend on SELECTED_OUTPUT. Model-essential
        outputs are written into phreeqc_templates.py. Additional desired
        outputs should be specified in the model input parameters (total,
        molalities, isotopes).

        Returns:
            A dict of outputs, as specified by SELECTED_OUTPUT.
        """
        output = self._call_iphreeqc('get_selected_output_array')
        assert len(output) > 0, "GetSelectedOutputArray returned null."
        header = output[0]
        out = {}
        for head in header:
            out[head] = []
        for row in output[1:]:
            for col, head in enumerate(header):
                out[head].append(row[col])

        ## correct bulk solution R14C for isotope fractionation
        # Some studies use uncorrected a14C (e.g. Genty et al. (1997)), some
```

```
        # corrected (e.g. Noronha et al. (2014)). Usually it should be
        # corrected. Note this was disabled in thesis work for like-for-like
        # comparison with Fohlmeister et al. (2011).
        new = []
        for c13,c14 in zip(out['I_R(13C)'], out['I_R(14C)']):
            pmc = ccu.pmc_normalise(  R14C=c14,
                                      d13C=c13,
                                      stnd14C=self.stnd14C  )
            new.append( pmc )
        out['I_R(14C)'] = new
        return out

    def _output_add(self, last_only=True):
        """Add current model output to long-term storage."""

        if not hasattr(self, 'output'):
            self.output = {}

        for key in self.last_output:
            if key not in self.output.keys():
                self.output[key] = []
            if last_only:
                self.output[key].append(self.last_output[key][-1])
            else:
                for e in self.last_output[key]:
                    self.output[key].append(e)


    def _verify_last_output(self):
        """Check last phreeqc step ran as intended."""
        # Check phreeqc error percentage
        for value in self.last_output['pct_err']:
            if abs(value) > PHREEQC_TOLERANCE:
                raise ValueError('PHREEQC percentage error exceeds tolerance.')

    def _call_iphreeqc(self, command, arg=None):
        """Run a command through IPhreeqc.

        Also initialises IPhreeqc if necessary. This method should not be
        called directly for reaction calls as it does not store IPhreeqc
        output. Use ipq_buffer() and ipq_exec() instead."""

        ## Initialise IPhreeqc interface and database
        if not hasattr(self, 'phreeqc'):

            if 'phreeqpy' in MODE:
                phreeqpy_path = os.path.dirname(phreeqc_mod.__file__)
                if 'linux' in sys.platform: # linux
                    f_name = 'libiphreeqc.so.0.0.0'
                elif sys.platform == 'win32': # windows
                    f_name = 'IPhreeqc.dll'
                else: # mac (presumably)
                    f_name = 'libiphreeqc.0.dylib'
                try:
                    self.phreeqc = phreeqc_mod.IPhreeqc(dll_path =
                                   os.path.join(phreeqpy_path, f_name))
                except:
                    phreeqpy_path = os.path.dirname(phreeqc_mod.__file__)
                    e = "\nPhreeqpy failed to find the neccessary file " + \
                    "for your system (.dll/.dylib/.so). Follow instructions " + \
                    "on the phreeqpy website to compile an appropriate " + \
                    "library file and copy it to:\n%s" % os.path.join(
                    phreeqpy_path, f_name)
                    raise Exception("IPhreeqc failed to load." + e)

            else: # if using API_com mode
                try:
                    self.phreeqc = Dispatch("IPhreeqcCOM.Object")
                except:
                    raise Exception("IPhreeqc failed to load. Ensure " + \
                    "IPhreeqcCOM is installed correctly.")

            # IPhreeqc is now initialised. Load the database:
            self._call_iphreeqc('load_database',self.settings['database'])

        if 'phreeqpy' in MODE:
            c = inflection.underscore(command)
        else:
            c = inflection.camelize(command)
        f = getattr(self.phreeqc,c)
        if arg: return f(arg)
        else: return f()

    def ipq_buffer(self, pq_input, description=''):
        """Adds IPhreeqc input strings to the buffer.

        Input strings are added to the buffer and decorated with USE, SAVE and
        END statements. The buffer is used to temporarily store model input
        before it is executed by IPhreeqc. The buffer may be executed using
        the ipq_exec() method.

        Args:
            pq_input: A valid PHREEQC input string. Multiple strings may be
                      passed as a list of tuples [(pq_input, description), ..].
                      If so they will be buffered in order, with no END
                      statements between.
            description: A short string describing the operation performed by
                         pq_input. Not necessary if pq_input is passed as a
                         list.
        """

        # if multiple arguments are passed, reduce them to single strings
        if type(pq_input) is list:
            pq_statements = [a for (a,b) in pq_input]
            descriptions = [b for (a,b) in pq_input]
            pq_input = '\n'.join(pq_statements)
            description = ', '.join(descriptions)

        self.count += 1
        if self.count == 1:
            input = self.selected_output + pq_input + "SAVE solution 1\nEND\n"
        else:
            soln_use = "\nUSE solution %i\n" % (self.count - 1)
            soln_save = "\nSAVE solution %i\nEND\n" % self.count
            input = soln_use + pq_input + soln_save

        self.string_buffer += input
        if len(self.desc_buffer) == 0:
            self.desc_buffer = description
        else:
```

```
            self.desc_buffer += ' | ' + description

    def ipq_exec(self):
        """Run the current string buffer through IPhreeqc and log the output.

        Commands are added to the buffer using the ipq_buffer() method. Output
        from the most recent call to ipq_exec() is stored in self.last_output.
        Previous model steps are logged in self.output.
        """

        inp = self.string_buffer

        if self.settings['phreeqc_log_file']:
            self.input_log.add(inp)    #add input to log file
        self._call_iphreeqc('run_string',inp)

        # save outputs
        self.last_output = self._get_selected_output()
        self.last_output['step_desc'] = [self.desc_buffer]

        self._output_add()
        self._verify_last_output()

        self.desc_buffer = ""       # clear description buffer
        self.string_buffer = ""     # clear string buffer

    def get(self, key):
        """Look up a specified value in the current model state.

        Looks up the given header as a key in the self.last_output dict,
        returning the most recent value stored there. This is used to perform
        model-state-dependent calculations (e.g. isotope fractionation).

        Args:
            key (str): A key in self.last_output.
        Returns:
            The most recent value of this header.
        """
        data = self.last_output
        assert key in list(data.keys()), \
            ("Key %s does not exist in selected_output. Keys:\n%s" %
            ( key, data.keys()))

        return data[key][-1]        # return most recent value in that entry

    def run(self):
        """Run Cavecalc!

        Runs a cavecalc model according to the model input parameters passed
        to the Simulator constructor. This high-level method is typically
        called immediately after initialising a Simulator object.

        The scriping followed depends on the kinetics_mode specified.

        If you want to understand the Cavecalc source code or scripting,
        this is a good place to start.

        Returns:
            A dict containing step-by-step model results as specified by
            SELECTED_OUTPUT.
        """

        # initialise soil water and calculate it's initial state
        water = Solution(self)
        self.ipq_buffer( *water.set_soil_chem() )
        self.ipq_exec()

        # initialise bedrock & calculate bedrock dissolution reaction
        bedrock = Carbonate(self)
        if self.settings['reprecip']:
            self.ipq_buffer(*water.open_diss_rxn_with_reprecip( bedrock ))
        else:
            self.ipq_buffer(*water.open_diss_rxn( bedrock ))
        self.ipq_exec()

        ## From here, behaviour depends on the run mode / kinetics_mode
        # kinetics mode options:
        #    closed_system_rayleigh
        #    open_system_single_step
        #    allow_supersaturation
        #    allow_supersaturation_max
        #    degas_only
        #    dissolve_only

        # closed_system_rayleigh
        if self.settings['kinetics_mode'] == 'closed_system_rayleigh': # keep at SI_calcite =↩
              0
            while water.will_it_degas(self.settings['cave_pCO2']):
                self.ipq_buffer( [water.kinetic_degas_rxn(), water.precipitate_rxn()] )
                self.ipq_exec()

        # open_system_single_step
        elif self.settings['kinetics_mode'] == 'open_system_single_step':
            self.ipq_buffer( [water.eq_degas_rxn(), water.precipitate_rxn()] )
            self.ipq_exec()

        # allow_supersaturation
        elif self.settings['kinetics_mode'] == 'allow_supersaturation':
            while water.will_it_degas(self.settings['cave_pCO2']):
                while self.get('si_Calcite') < self.settings['calcite_sat_limit']:
                    self.ipq_buffer( *water.kinetic_degas_rxn() )
                    self.ipq_buffer( WATER_EQUILIBRATE, 'DIC equilibrate' )
                    self.ipq_exec()
                self.ipq_buffer( *water.precipitate_rxn() )
                self.ipq_exec()

        # allow_supersaturation_max
        elif self.settings['kinetics_mode'] == 'allow_supersaturation_max':
            while water.will_it_degas(self.settings['cave_pCO2']):
                while water.will_it_degas(self.settings['cave_pCO2']):
                    self.ipq_buffer( *water.kinetic_degas_rxn() )
                    self.ipq_buffer( WATER_EQUILIBRATE, 'DIC equilibrate' )
                    self.ipq_exec()
                self.ipq_buffer( *water.precipitate_rxn() )
                self.ipq_exec()

        # degas_only
        elif self.settings['kinetics_mode'] == 'degas_only':
            while water.will_it_degas(self.settings['cave_pCO2']):
```

```
                self.ipq_buffer( *water.kinetic_degas_rxn() )
                self.ipq_buffer( WATER_EQUILIBRATE, 'DIC equilibrate' )
                self.ipq_exec()

        # diss_only
        elif self.settings['kinetics_mode'] == 'diss_only': # dissolve bedrock only
                pass

        ccu.PostProcessor(self) # perform offline calculations and processing

        return self.output

    def save_results(self, format='.pkl', filename='output'):
        """Save model output to specified file and format.

        Files will be saved in the current directory.

        This method is provided for low-level access. Generally it is advisable
        to run Cavecalc via forward_models.py and use the more powerful saving
        functions of that module.

        Args:
            format (str): '.csv', '.pkl' or '.mat'
            directory (str): Save directory
            filename (str): Save file name without extension. Not used for pkl.
        """

        if 'csv' in format:
            ccu.save_csv(self.output, '%s.csv' % filename)
        elif 'pkl' in format:
            ccu.save_pkl([self.output], 'results.pkl')
            ccu.save_pkl([self.settings], 'settings.pkl')
        elif 'mat' in format:
            ccu.save_mat(self.output, '%s.mat' % filename)
        else:
            raise IOError("Unrecognised output format.")
```

### A.3.3 cavecalc/analyse.py

```
"""Contains the Evaluate object, which has methods to load data from Cavecalc
.pkl output files, display data and write it to other file formats.

Classes defined here:
    Evaluate
"""

import pickle
import os
import copy
import matplotlib
from sys import platform
if platform != 'win32':
    matplotlib.use('TkAgg') # necessary for mac
from matplotlib import pyplot as plt
import numpy as np
import cavecalc.util as ccu
import scipy.io as sio
import seaborn as sns

class Evaluate(object):
    """Processes of Cavecalc model output.

    Evaluate contains methods to load, process, format and display Cavecalc
    model data saved in .pkl files. It allows model output to be saved to .csv
    and .mat files for further processing.

    Evaluate objects may be used directly (see examples), and are also used
    under the hood by the output GUI.

    Methods:
        get_settings_report - Get a dict summarising model settings
        load_data           - Load .pkl files from a directory
        save_csvs           - Save all loaded model output to .csv files
        save_all_mat        - Save all loaded model output to a .mat file
        filter_out_noprecip - Filter out all model steps that don't
                              precipitate calcite.
        filter_by_index     - Filter model steps in/out based on a model step
                              index.
        filter_by_results   - Filter model steps in/out based on any output
                              parameter.
        filter_by_settings  - Filter whole models in/out based in any input
                              parameter.
        plot_models         - Plot all loaded models.
        plot_points         - Plot selected steps from all loaded models.
    """

    def __init__(self):
        """Initialise an Evaluate object.

        After initialisation, load_data must be called to read data from .pkl
        files.
        """

        self._models = []
        self._settings = []

    @property
    def model_settings(self):
        """Return a list of dicts containing model settings.

        Generates a list of dicts from from all loaded SettingsObjects by
        calling their .dict() method.

        Returns:
            A list of settings dicts.
        """

        if not self._settings:
            raise ValueError("Object %r has no models loaded." % self)

        o = [s.dict() for s in self._settings]
```

```
            for d in o:
                try:
                    d.pop('id')
                except KeyError:
                    pass

            if o:
                return copy.deepcopy(o)
            else:
                raise ValueError("Object %r has no models loaded." % self)

        @property
        def model_results(self):
            """Return a list of dicts containing model output.

            Returns:
                A list of results dicts.
            """

            if self._models:
                return copy.deepcopy(self._models)
            else:
                raise ValueError("Object %r has no models loaded." % self)

        def get_settings_report(self):
            """Get a summary of the range of model settings.

            Returns:
            A dict of model settings, with one entry for each unique value detected.
            """

            d = self.model_settings[0]
            o = dict.fromkeys(d.keys(),[])
            for s in self.model_settings:
                for k, v in s.items():
                    if v not in o[k]:
                        o[k].append(v)
            try:
                o.pop('id')
            except KeyError:
                pass
            return o

        def load_data(self, *args):
            """Load .pkl data into the Evaluate object for processing.

            Data is loaded from a directory. The directory must contain
            settings.pkl and results.pkl. load_data may be called multiple times
            to merge different model suites for comparison.

            Args:
                *args: The directories to load data from.
            """

            ret_dir = os.getcwd()

            if len(args) == 0:
                args = (ret_dir,)

            for d in args:
                os.chdir(d)
                try:
                    print("Attempting to load data from %s..." % d, end="")
                    with open('settings.pkl', 'rb') as f:
                        self._settings.extend(pickle.load(f))

                    with open('results.pkl', 'rb') as f:
                        r = pickle.load(f)
                        self._models.extend([a for (a,b) in r])
                    print(" Done")
                finally:
                    os.chdir(ret_dir)

        def save_csvs(self, directory=None):
            """Save model output to .csv files.

            One file is saved for each model loaded. Note that only model output
            can be saved to csv, not the settings used.

            csv files may be easily read in any spreadsheet program.

            Args:
                directory (str): The directory to save output to.
            """

            if not directory:
                directory = os.getcwd()

            for (id, model) in enumerate(self._models):
                f = os.path.join(directory, "out_%i.csv" % id)
                ccu.save_csv(model, os.path.join(f))

        def save_all_mat(self, file):
            """Save all loaded data to a .mat file.

            Data is saved as two matlab structs, reflecting the data structures
            inside settings.pkl and results.pkl respectively.

            Args:
                file: Filename to save to (.mat will be auto-appended)
            """

            s = dict()

            for i, S0 in enumerate(self._settings):             # for each model
                set = S0.dict()          # settings dict
                res = self._models[i]     # results dict

                # remove any 'None' values from output (savemat can't handle them)
                # replace with -999 value, like PHREEQC
                n_res = dict()
                for k,v in res.items():
                    nv = []
                    for e in v:
                        if e is None:
                            nv.append(-999)
                        else:
                            nv.append(e)
                    n_res[k] = nv

                o = {k:(v if type(v) is list else [v]) for k,v in set.items()}
```

```
            a = ccu.numpify(o)                          # settings
            b = ccu.numpify(n_res)                      # output

            c = { 'settings' : a,
                  'results'  : b   }

            name = "m" + str(i)
            s[name] = c

        sio.savemat(file, s)

    def filter_out_noprecip(self):
        """Returns a filtered copy of the Evalulate object.

        Models are filtered out of they do not include any precipitation
        reactions. This is useful for 'eq' mode analyses to remove
        non-precipitating solutions.
        """

        A = Evaluate()

        for i,m in enumerate(self._models):
            a = False
            for s in m['step_desc']:
                if 'precip' in s:
                    a = True
                    break

            if a:
                A._models.append(copy.deepcopy(m))
                A._settings.append(copy.deepcopy(self._settings[i]))

        return A

    def filter_by_index(self, ind, n=False):
        """Return a filtered copy of the Evaluate object.

        Filter the model output data contained in the object. Data is filtered
        based on list index position - this corresponds to the calculation step
        in the model. This method is useful for subsetting data in preparation
        for plotting. It works similarly to filter_by_results.

        Example:
            e = Evaluate()
            e.load_data('./my_data/')
            f = e.filter_by_index(-1) # extracts the final dripwater chemistry

        Args:
            ind: An integer index to filter by. This corresponds to a model
                step number. E.g. index 0 is the first PHREEQC calculation
                (initial water chemistry), index 1 is the bedrock dissolution
                product, index -1 is the final solution chemistry.
            n: Optional boolean argument. If True, the filter is inverted.
                Default False.

        Returns:
            A modified copy of the object. The copy only contains model output
            that meet the filter criteria.
        """
        A = Evaluate()
        A._settings = copy.deepcopy(self._settings)

        for m in self._models:
            if ind < 0:
                explicitIndex = len(m['step_desc']) + ind
            else:
                explicitIndex = ind

            if n is False:
                fil = {k : [a[explicitIndex]] for k,a in m.items()}
            else:
                fil = {k : [v for i,v in enumerate(a) if i != explicitIndex] for k,a in m.↩
                    items()}
            A._models.append(fil)


        rem = []
        for i,r in enumerate(A._models):
            if max([len(v) for k,v in r.items()]) == 0:
                rem.append(i)

        [A._models.pop(i) for i in rem]
        [A._settings.pop(i) for i in rem]

        return copy.deepcopy(A)

    def filter_by_results(self, key, value, n=False):
        """Return a filtered copy of the Evaluate object.

        Filter the model output data contained in the object. Data is filtered
        based on a key, value combination. This method is useful for
        subsetting data in preparation for plotting. It works similarly to
        filter_by_index.

        Example:
            e = Evaluate()
            e.load_data('./my_data/')
            f = e.filter_by_settings('step_desc', 'degas')
            # f includes only data from the degassing steps

        Args:
            key: Key in model output dicts to filter by.
            value: Value to filter 'key' by. Accepts substrings for step_desc.
            n: Optional boolean argument. If True, the filter is inverted.
                Default False.

        Returns:
            A filtered copy of the Evaluate object.
        """

        A = Evaluate()
        A._models = []
        A._settings = self._settings

        # filter object
        for i, m in enumerate(self._models):
            fil = {}
            a = m[key]
            for j, v in m.items():
                if len(v) == len(a):
                    if n:
```

```
                        fil[j] = [v[k] for k in range(len(v)) if value not in a[k]]
                    else:
                        fil[j] = [v[k] for k in range(len(v)) if value in a[k]]
                else:
                    fil[j] = v
            A._models.append(fil)
        return copy.deepcopy(A)

    def filter_by_settings(self, setting, value, n=False):
        """Return a filtered copy of the Evaluate object.

        The returned Evaluate object contains a subset of the models in the
        original. Models are filtered based on the settings, value combination
        provided. Models that meet the critera have their data included in the
        copy.

        Args:
            setting (str): model parameter to filter by (e.g. 'gas_volume')
            value: value of 'setting' to include (e.g. 20).
            n: Optional boolean argument. If True, the filter is inverted.
                Default False.
        Returns:
            A filtered copy of the Evaluate object.
        """

        A = Evaluate()
        A._models = []
        A._settings = []

        for i, b in enumerate(self._settings):
            d = b.dict()
            if n:
                if isinstance(value, str):
                    if value not in d[setting]:
                        A._models.append(self._models[i])
                        A._settings.append(b)
                else:
                    if d[setting] != value:
                        A._models.append(self._models[i])
                        A._settings.append(b)
            else:
                if isinstance(value, str):
                    if value in d[setting]:
                        A._models.append(self._models[i])
                        A._settings.append(b)
                else:
                    if d[setting] == value:
                        A._models.append(self._models[i])
                        A._settings.append(b)

        return copy.deepcopy(A)

    def plot_models(self, *args, x_key=None, y_key=None,
                    label_with=None, ax=None, **kwargs):
        """Plot Model results with one series per model.

        Creates a simple matplotlib figure. Useful, for example, to quickly
        display the degassing evolution of a suite of models. May be combined
        with filter_by_settings, filter_by_results or filter_by_index to
        include / exclude certain parts of the dataset.

        Args:
            *args: Optional formatting parameters passed to pyplot.plot()
            x_key: Model output to plot on x-axis
            y_key: Model output to plot on y-axis
            label_with (optional): Model input parameter to annotate series
            ax (optional): Add data to a pre-existing matplotlib axis
            **kwargs (optional): kwargs to be passed to pyplot.plot()
        Returns:
            Axes object.
        """

        sns.set_style('darkgrid')
        if not ax:
            fig, ax = plt.subplots()
            ax.set_ylabel(y_key)
            ax.set_xlabel(x_key)
        for i, m in enumerate(self._models):
            if label_with:
                s = self._settings[i].get(label_with)
                a = "%s: %s" % (label_with, s)
                ax.plot(m[x_key], m[y_key], label = a, *args, **kwargs)
                ax.legend(prop={'size':6})
            else:
                ax.plot(m[x_key], m[y_key], *args, **kwargs)


        return ax

    def plot_points(self, *args, x_key=None, y_key=None, plot_index=1,
                    label_with=None, ax=None, **kwargs):
        """Plot Model results for a point-by-point inter-model comparison.

        Useful, for example, to show different bedrock dissolution products
        across a suite of models.

        Args:
            x_key: Model output or setting parameter to plot on x-axis
            y_key: Model output to plot on y-axis
            *args (optional): Formatting parameters passed to pyplot.plot()
            plot_index: Which point to plot. e.g. 0 (initial water), 1 (bedrock
            dissolution product), -1 (fully degassed solution)
            label_with (optional): Model input parameter to label points with
            ax (optional): Add data to a pre-existing plot
            **kwargs (optional): kwargs to be passed to pyplot.plot()
        Returns:
            Axes object.
        """
        sns.set_style('darkgrid')

        x_vals = []
        y_vals = []
        labels = []

        # look for x_key in results
        if x_key in list(self._models[0].keys()):
            for i, m in enumerate(self._models):
                try:
                    x_vals.append(m[x_key][plot_index])
                except IndexError:
```

```
                    pass

        # otherwise, find it in settings
        else:
            for i, s in enumerate(self._settings):
                x_vals.append(s.dict()[x_key])

        for i, m in enumerate(self._models):
            if label_with:
                s = self._settings[i].dict()
            try:
                y_vals.append(m[y_key][plot_index])
                if label_with:
                    labels.append(s[label_with])
            except IndexError:
                pass

        if not ax:
            fig, ax = plt.subplots()
            ax.set_ylabel(y_key)
            ax.set_xlabel(x_key)
        ax.plot(x_vals, y_vals, *args, **kwargs)
        if label_with:
            for lab, x, y in zip(labels, x_vals, y_vals):
                ax.annotate('%s=%s' % (label_with, lab),
                            xy=(x, y), fontsize=8)
        return ax
```

### A.3.4 cavecalc/forward_models.py

```
"""Allows multiple models to be run in sequence and handles model IO. For most
use-cases, cavecalc Models should be run using the ForwardModels class.

Classes defined here:
    ForwardModels

Functions defined here:
    run_a_model
    run_async
    run_linear
"""

# built-in modules
import os
import multiprocessing as mp
import pickle
# my modules
import cavecalc.caves as ccv
import cavecalc.util as ccu
from cavecalc.setter import SettingsObject, SettingsMaker
from copy import deepcopy

def run_a_model(SO):
    """Run a single cavecalc model.

    Args:
        SO (SettingsObject): A SettingsObject describing model input.
    Returns:
        (r, id) where r is the model results dict and id is the 'id' parameter
        in settings. If the model raises an error, (None, id) is returned.
    """

    i = SO.id
    sim = ccv.Simulator(SO.dict(), i)
    r = sim.run()
    print("Model %i complete." % i)
    return (r, i)

def run_async(SO_list):
    """Run multiple models in parallel.

    WARNING: Parallel processing may cause errors in some calculations. Use
    only when absolutely necessary. run_linear() is recommended instead.

    Args:
        SO_list: A list SettingsObjects, e.g. from a SettingsMaker
    Returns:
        A list of (r, id) tuples. r is the model results dict and id is the
        'id' parameter in the settings dict.
    """

    s = SO_list
    results = {}
    pool = mp.Pool()
    results = [pool.apply_async(run_a_model, (e.dict(), e.id)) for e in s]
    pool.close()
    pool.join()
    r = [res.get() for res in results]
    r2 = sorted(r, key=lambda tup: tup[1]) # sort by model id
    return r2

def run_linear(SO_list):
    """Runs multiple models in sequence.

    Args:
        SO_list: A list SettingsObjects, e.g. from a SettingsMaker
    Returns:
        A list of (r, id) tuples. r is the model results dict and id is the
        'id' parameter in the settings dict.
    """

    return [run_a_model(e) for e in SO_list]

class ForwardModels(object):
    """Runs Cavecalc models.

    Handles generation and checking of settings suites, and saving of bundled
    model output. This class provides a flexible interface to cavecalc.caves
    (the module which actually runs the models), and is the preferred method
    of running models.
```

```
    """

    def __init__(self, settings=None, output_dir=None):
        """Initialise the object and process settings.

        Args:
            settings (SO): Model input parameters. See defaults.txt for
                           options.
            settings_file: If no settings arg is given, settings may be loaded
                           from a file.
            output_dir: (Optional) Specify path to directory for saving files.
                        By default files are saved to the current directory.
        """

        self.done_input = []
        self.done_results = []

        # Get settings objects
        if settings:    self.input = SettingsMaker(**settings).settings()
        else:           self.input = SettingsMaker().settings()

        if output_dir:
            if not os.path.isdir(output_dir):
                os.mkdir(output_dir)
            self.output_dir = output_dir
        else: self.output_dir=os.getcwd()

    def _check_previous_saves(self, interactive=False, use_by_default=True):
        """Checks output directory for existing output and prompts the user
        to decide whether they want to use it or not."""


        def dict_find(SO, list_of_SOs):

            for i,s in enumerate(list_of_SOs):

                # check keys are equal
                if SO.dict().keys() != s.dict().keys():
                    continue

                # check all values are equal
                equal = True
                for k,v in SO.dict().items():
                    if v != s.dict()[k]:
                        equal = False
                if equal:
                    return i
            return None

        try:
            with open(os.path.join(self.output_dir, 'settings.pkl'), 'rb') as f:
                prev_input = pickle.load(f)

            with open(os.path.join(self.output_dir, 'results.pkl'), 'rb') as f:
                prev_results = pickle.load(f)
        except FileNotFoundError:
            return

        if (prev_input is None) or (prev_results is None):
            return

        new_input = []
        done_input = []
        done_results = []

        # for i,prev in enumerate(prev_inputs):
        for s in self.input:
            i = dict_find(s, prev_input)
            if i is None:
                new_input.append(s)
            else:
                done_input.append(prev_input[i])
                done_results.append(prev_results[i])

        print("Previous model output detected for selected input settings.")

        if interactive == True:
            a = ''
            while a == '':
                print( "%i out of %i models appear to be repeated." %
                       (len(done_input), len(self.input)))
                a = input("Re-use old output for these models (y/n)? ")
                if a.capitalize() == 'Y':
                    reuse = True
                elif a.capitalize() == 'N':
                    reuse = False
                else:
                    a = ''

        else:
            print("%i out of %i models are repeats." %
                       (len(done_input), len(self.input)))
            if use_by_default:
                print("Re-using old calculations where available.")
            else:
                print("Re-running all models")
            reuse = use_by_default

        if reuse:
            self.input = new_input
            self.done_input = done_input
            self.done_results = done_results

    def run_models(self, mode='Serial', **kwargs):
        """Run models for all parameter sets loaded into the object. Output is
        addded to self.results as a list of (r, id) tuples. See run_linear and
        run_async for method. Parallel mode is not currently functional (on
        Windows at least).

        Args:
            mode: 'Serial' (default) or 'Parallel'. Running models in parallel
                  is faster but not recommended - results may be inaccurate, as
                  Iphreeqc is not necessarily thread-safe in this
                  implementation.
        Returns:
            Nothing. Model results list is assigned to self.results. List
            indices in self.results correspond to indices in self.self.input.
        """

        self._check_previous_saves(**kwargs)
        ret_dir = os.getcwd()
```

```
        print("Models to run:\t%s" % len(self.input))
        try:
            os.chdir(self.output_dir)
            if mode.capitalize() == 'Serial': # Run models in order
                results = run_linear(self.input)
            elif mode.capitalize() == 'Parallel': # Run models multithreaded
                print("Warning: Parallel operation accuracy not guaranteed.")
                results = run_async(self.input)
            else:
                raise ValueError("Mode not recognised. Use serial / parallel")
            self.results = results


            # add re-used output, if any
            self.results.extend(self.done_results)
            self.input.extend(self.done_input)

        finally:
            os.chdir(ret_dir)

    def save(self):
        """Save results and settings data to .pkl files.

        The resulting files (results.pkl and settings.pkl) may be read using
        the cavecalc.analyse module.

        results.pkl contains a list of dicts. Each dict contains the output of
        a single model. settings.pkl contains a similar list of
        SettingsObjects, one for each model run.
        """

        if not os.path.exists(self.output_dir):
            os.makedirs(self.output_dir)
        with open(os.path.join(self.output_dir, 'settings.pkl'), 'wb') as f:
            pickle.dump(self.input, f)

        with open(os.path.join(self.output_dir, 'results.pkl'), 'wb') as f:
            pickle.dump(self.results, f)

    def _debug(self, i):
        """Runs the specified model and saves the pq_input_log file for
        debugging. The input log file may be run directly by PHREEQC.

        Args:
            i: Index position in self.input to re-run in debug mode.
        """
        s_debug = self.input[i]
        s_debug.set(phreeqc_log_file=True)

        dbug = ccv.Simulator(s_debug, id=i)
        dbug.run()
```

## A.3.5 cavecalc/setter.py

```
"""Generates settings for use in forward_models.py and caves.py for running
(suites of) models. SettingsObject instances are used for validating input
settings and passing them to caves.py to run.

Classes defined here:
    NameSwitcher
    SettingsObject
    SettingsMaker
"""

import os
import copy
import cavecalc.gui.mapping
import cavecalc.data.defaults as ccd
import cavecalc.data.types_and_limits as tal

cc_defaults = {k:v for k,v in vars(ccd).items() if '__' not in k}
cc_types = {k:v for k,v in vars(tal).items() if '__' not in k}

class NameSwitcher(object):
    """Handles switching parameter names between code and readable versions.

    Names and their readable equivalents are given in cavecalc.gui.mapping.

    Useage:
        ns = NameSwitcher()
        name2 = ns(name1)
        name3 = ns(name2)   # name3 = name1
    """

    def __init__(self):
        """Initialise a NameSwitcher object.

        Loads a name mapping ready for use.
        """

        self._name_map()

    def __call__(self, input):
        """Convert parameter names to their alternative formats.

        If names are passed as code names, readable versions are returned and
        vice versa.

        Args:
            input: Model name(s) to convert. May a string, list, set or dict
            where the keys represent the the parameter names.
        Returns:
            A copy of the original with the parameter names switched to their
            alternative forms.
        """

        return self._switch_names(input)

    def _name_map(self):
        """Load name mappings as two dicts."""
```

```
        d1 = vars(cavecalc.gui.mapping)
        for k in copy.copy(d1):
            if '__' in k:
                d1.pop(k)
        d2 = {v : k for (k, v) in d1.items()}
        self.m2g = d1
        self.g2m = d2

    def _switch_names(self, input):
        """Switches input entries (keys in the case of a dict) from model
        to gui parameter names, or vice versa.
        """

        s = input
        if type(s) is dict:
            try:
                output = { self.m2g[k] : v for (k,v) in s.items() }
            except KeyError:
                output = { self.g2m[k] : v for (k,v) in s.items() }

        elif type(s) is set:
            try:
                output = { self.m2g[k] for k in s }
            except KeyError:
                output = { self.g2m[k] for k in s }

        elif type(s) is list:
            try:
                output = [ self.m2g[k] for k in s ]
            except KeyError:
                output = [ self.g2m[k] for k in s ]

        elif type(s) is str:
            try:
                output = copy.copy(self.m2g[s])
            except KeyError:
                output = copy.copy(self.g2m[s])

        else:
            raise TypeError("Input of type %s unsupported" % type(s))

        return output

class SettingsObject(object):
    """Constructs settings for use in Cavecalc models.

    Cavecalc caves.Simulator objects treats input settings as a dict and it is
    possible to write these manually. However, use of SettingsObject is
    advised - it provides default values, parameter name switching and value
    checking.

    SettingsObjects are not normally instantiated directly - use SettingsMaker
    instead.
    """

    ns = NameSwitcher()

    def __init__(self, id=None, **kwargs):
        """Initialise settings object with default parameters.

        Default settings are stored in data/defaults.py.are loaded from
        defaults.py. Non-default values may be specifed in kwargs, or later
        using the set() method.

        Args:
            id: Model id number (optional)
            kwargs: Keyword arguments for model parameters
        """

        self.id = id

        # set defaults
        for k, v in cc_defaults.items():
            setattr(self, k, None)
            self.set(k, copy.copy(v))

        # set input parameters
        for k, v in kwargs.items():
            self.set(k, copy.copy(v))

    def __call__(self, *args):
        """Calls self.get() on *args."""

        return get(self, *args)

    def validate_entry(self, attribute, value='self'):
        """Check attribute, value pair is valid model input.

        Check entry against data/types_and_limits.py to confirm it is of
        the correct type and/or value.

        Args:
            attribute (str): A model input parameter value.
            value: Value to be tested for validity. If not specified, the
                value already set is used.
        """

        a = attribute

        if value == 'self':
            b = getattr(self, a)
        else:
            b = value
        assert a in cc_types.keys()
        r = cc_types[a]

        ## for simple string or boolean input
        def type_test(t):
            def inner():
                if type(r) is t:
                    if isinstance(b, t):
                        return True
                    else:
                        raise TypeError("%s is of incorrect type: %r" %
                            (b, type(b)))
            return inner

        if type_test(str)(): return True
        if type_test(bool)(): return True

        ## for tuple-defined acceptable values
```

```
        if type(r) is tuple and isinstance(b, (float, int)):
            if r[0] is not None and b < r[0]:
                raise ValueError("%s invalid value: %f is less than %f" %
                    (a, b, r[0]))
            if r[1] is not None and b > r[1]:
                raise ValueError("%s invalid value: %f is greater than %f" %
                    (a, b, r[1]))
            return True         # numeric value within allowed range

        if type(r) is tuple and isinstance(b, str):
            try:
                if b in r[2:]:
                    return True # allowable string value detected
                else:
                    raise ValueError("%s invalid value: %r" % (a, b))
            except IndexError:
                raise TypeError("%s unsupported type: %r" % (a, type(b)))

        if type(r) is tuple:
            raise TypeError("%s unsupported type: %r" % (a, type(b)))

        ## for list-defined acceptable values
        if type(r) is list:
            if b.capitalize() in [e.capitalize() for e in r]:
                return True
            else:
                raise ValueError("%s invalid value: %r" % (a, b))


        raise Exception("Failed to check input type.\nData:\t%r\t%r\t%r"
                % (a, b, r))

    def set(self, parameter=None, value=None, **kwargs):
        """Set model input parameter(s).

        Arguments are first checked with validate_entry(). If validated, the
        model settings are updated.

        Args:
            parameter (str): The name of the parameter to be set.
            value: The value to set.
            **kwargs: Keyword arguments specifying multiple parameter=value
                pairs.
        """

        if parameter:
            kwargs[parameter] = value

        for k, v in kwargs.items():
            if k not in self.ns.m2g.keys():
                if k in self.ns.g2m.keys():
                    k = self.ns(k)
                else:
                    raise AttributeError(
                        "SettingsObject has no attribute %r" % k )

            if k == 'database':
                db_path = os.path.dirname(cavecalc.data.__file__)
                v = os.path.join(db_path,v)
            self.validate_entry(k, v)
            setattr(self, k, v)

    def get(self, *args):
        """Gets specified parameters from the object.

        If no arguments are specified, self.dict() is returned.

        Args:
            *args: names of parameters to be queried.
        Returns:
            A tuple of values corresponding to the length of *args. If args are
            not provided, self.dict() is returned.
        """

        if len(args) > 0:
            o = []
            for k in args:
                try:
                    o.append(getattr(self, k))
                except AttributeError:
                    o.append(getattr(self, self.ns(k)))

            if len(o) == 1:
                return o[0]
            else:
                return tuple(o)
        else:
            return self.dict()

    def dict(self):
        """Return a dict of object settings.

        self.dict() is usually used to pass the model settings to a
        caves.Simulator() object to be run. This excludes the 'id' setting,
        which may be passed to Simulator separately.
        """

        a = copy.copy(vars(self))
        a.pop('id')
        return a

    def equals(self, SettingsObject):
        """Compare two SettingsObjects for equality.

        Args:
            SettingsObject: A second object to compare self to.
        Returns:
            True if all settings are equal. Otherwise False.
        """

        d1 = vars(self)
        d2 = vars(SettingsObject)

        mismatch = set(d1.items()) ^ set(d2.items())

        if len(mismatch) == 0:
            return True
        else:
            return False

class SettingsMaker(object):
```

```
    """Generates suites of SettingsObjects.

    Allows easy generation of multiple SettingsObjects covering a range of
    settings, usually to be run in series by cavecalc.forward_models."""

    def __init__(self, **kwargs):
        """Make a suite of SettingsObjects based on **kwargs.

        Args:
            **kwargs: keyword arguments of parameter=value for all non-default
            model input parameters. values may be single entries or lists of
            entries. If a lists are provided, a range of SettingsObjects are
            generated, covering all possible combinations of input parameters.

        SettingsObjects generated are stored in the list self.o.
        """

        self.inps = kwargs
        self.sv = []    # single values (non-default constants)
        self.mv = []    # multiple values (a list of non-defaults)
        self.n = 1
        self.o = []

        self._sort_inps()

        if len(self.mv) > 0:
            self._make_set()
        else:
            self._make_one()
        self._set_ids()

        # self.print_report()

    def _sort_inps(self):
        """Split inputs into constants and variables"""

        for k,v in self.inps.items():
            if type(v) is list:
                self.mv.append((k, v))
            else:
                self.sv.append((k, v))

    def _make_one(self):
        consts_dict = {a[0] : a[1] for a in self.sv}
        self.o.append(SettingsObject( **consts_dict ))

    def _make_set(self, constants=None, variables=None):
        """Create a set of unique SettingsObjects covering all possible
        combinations of inputs. constants and variables are lists of key,
        value tuples."""

        if not constants:
            constants = self.sv

        if not variables:
            variables = self.mv

        if len(variables) > 1:
            a_var = variables.pop()
            for e in a_var[1]:
                consts = copy.deepcopy(constants)
                consts.append((a_var[0], e))
                self._make_set( consts, copy.deepcopy(variables) )
            return

        if len(variables) == 1:
            for e in variables[0][1]:
                consts_dict = {a[0] : a[1] for a in constants}
                variables_dict = {variables[0][0] : e}
                self.o.append(SettingsObject( **consts_dict, **variables_dict ))

    def _remove_copies(self):
        """Filters self.o to remove identical SettingsObjects."""

        def all_diff(lst):
            for i,a in enumerate(lst):
                for j,b in enumerate(lst):
                    if a.equals(b) and i!=j:
                        return False
            return True

        def del_ifn_unique():
            c = copy.copy(self.o)
            for i,a in enumerate(c):
                for j,b in enumerate(c):
                    if a.equals(b) and i!=j:
                        self.o.pop(i)
                        return

        while not all_diff(self.o):
            for i,m in enumerate(self.o):
                del_ifn_unique()

    def _set_ids(self):
        for i,a in enumerate(self.o):
            a.id = i

    def print_report(self):
        """Debug function for viewing SettingsObjects in self.o."""

        def list_cmp(l):
            for i,e in enumerate(l):
                try:
                    if e != l[i+1]:
                        return True
                except:
                    return False

        d = self.o[0].dict()
        for k in d.keys():
            v = []
            for m in self.o:
                v.append(m.get(k))
            if list_cmp(v):
                print(k,end='\t\t')
                for e in v:
                    print(e,end='\t')
                print('')

    def settings(self):
        """Return the list of generated SettingsObjects."""
```

```
        return copy.deepcopy(self.o)

if __name__ == '__main__': # testing code
    # a = SettingsObject()
    # b = SettingsObject()
    # print("Equal?" + '\t' + str(a.equals(a)))
    # print("Equal?" + '\t' + str(a.equals(b)))
    # a.set(temperature=35)
    # print("Equal?" + '\t' + str(a.equals(b)))
    # print(a.get('temperature'))

    # n = NameSwitcher()
    # print(n('temperature'))

    a = SettingsMaker(temperature=[15, 20], soil_pCO2=[2222, 3333])
    a.print_report()
    a = SettingsMaker()
```

## A.3.6 cavecalc/util.py

```
"""Cavecalc module for utility functions and classes.

Contains code to perform file IO, model output post-processing and various
other helpful bits.
"""

import os
import datetime
import math
import pickle
import csv
import re
from collections import OrderedDict
from copy import copy
import numpy as np
import scipy.io as sio
import cavecalc.data

class DBReader(object):
    """Reads data from the PHREEQC database file.

    DBReader objects are used to extract out database information (e.g.
    partition coefficients and fractionation factors) that may be needed in
    geochemical calculations coded in caves.py. Thermodynamic parameters are
    not hard-coded into Cavecalc python files.

    DBReader objects cache use regular expressions to parse the database file
    and cache data extracted to improve performance on repeated requests.

    Selected Methods:
        get_k_values - get phase definition thermodynamic data
        get_1000lnalpha - get an isotope fractionation factor
        get_alpha - get an isotope fractionation factor
        get_iso_stnd - get the absolute isotope ratio in a standard
    """

    def __init__(self, database):
        """Initialise the object.

        Args:
            database (str): location of phreeqc database filename.
        """

        db_dir = os.path.dirname(cavecalc.data.__file__)
        self.db = os.path.join(db_dir,database)

        self.alphas = {}
        self.thermo = {}

    def _cc(self, line):
        return line.split('#')[0]

    def _database_eval(self, equation, temperature):
        """Evaluate a database expression for a fractionation factor or
        partition coefficent.

        Args:
            equation (list): A list of coefficents for a PHREEQC thermodynamic
                             parameter.
            temperature: Temperature (degrees K) at which to evaluate the
                             equation.
        Returns:
            Value of the expression at the given temperature
        """
        T = temperature
        a = equation

        coeffs = [1, T, 1/T, math.log10(T), 1/(T*T), T*T]
        evaluation = 0
        for i in range(len(a)):
            evaluation += float(a[i])*float(coeffs[i])   # compute 1000ln_alpha
        return evaluation

    def _phase_lookup(self, reactants_list):

        r = r' \+ '.join(reactants_list)
        rc1 = re.compile(r"\s*?{}\s*=".format(r))

        out = []

        with iter(open(self.db, 'r')) as f:
            for line in f:
                if re.match(rc1, line):
                    line = next(f)
                    while line.strip() not in ('', '#'):
                        a = self._cc(line.strip()).split()
                        out.append(copy(a))
                        line = next(f)
                    return [o for o in out if o]
```

```
            raise Exception("No PHASES entry matched: %s" % r)

        def _ne_lookup(self, isotope, species):

            def escape_brackets(string):
                s = string.replace(r'(', r'\(')
                s = s.replace(r')',r'\)')
                s = s.replace(r'\\)',r'\)')
                s = s.replace(r'\\(',r'\(')
                s = s.replace(r'\[',r'[')
                s = s.replace(r'\]',r']')
                return s

            isotope = escape_brackets(isotope)
            species = escape_brackets(species)

            r1 = r"\s*Log_alpha_{}_{}\s*".format(isotope, species)
            rc1 = re.compile(r1)
            out = []

            with iter(open(self.db, 'r')) as f:
                for line in f:
                    if re.match(rc1, line):
                        line = next(f)
                        while line.strip() not in ('', '#'):
                            a = self._cc(line.strip()).split()
                            out.append(copy(a))
                            line = next(f)
                        return [o for o in out if o]
            raise Exception("No NAMED EXPRESSIONS entry matched: %s" % r1)

        def get_k_values(self, reactants_list, temperature=298.15):
            """Get thermodynamic data for a specified phase.

            Looks up the reaction specified by 'reactants_list' in the database
            and returns all thermodynamic data found.

            Args:
                reactants_list: A list of strings. Each string defines a species
                    such that the species listed, all reacted together, specify a
                    dissolution reaction (i.e. PHASES defintiion) in the database
                    file.
                temperature: The temperature to evaluate any analytical expressions
                    at.

            Returns:
                A dict containing data found:
                log_k: log_k for the reaction, as defined in the database.
                delta_h: delta_h for the reaction, as defined in the database.
                gamma: gamma for the reaction, as defined in the database.
                analytic_line: string containing analytical K expression constants
                analytic: The evaluation of analytic_line at the specified
                    temperature.

                Entries not present in the database are returned as None.
            """

            if temperature < 50:
                print("Warning! Temperature (K) given as %i. Converting to %i" %
                      (temperature, temperature+273.15))
                temperature = temperature + 273.15

            # If possible return cached values
            chk = reactants_list.copy()
            chk.append(str(temperature))
            chk = ' '.join(chk)
            if chk in self.thermo:
                return self.thermo[chk]

            thermo = {}
            data = self._phase_lookup(reactants_list)
            for a in data:
                if a[0] == 'log_k':
                    thermo[a[0]] = a[1]
                elif a[0] == 'delta_h':
                    thermo[a[0]] = a[1]
                elif a[0] == '-gamma':
                    thermo[a[0]] = a[1]
                elif a[0] == '-analytic':
                    thermo['analytic_line'] = ' '.join(a)
                    thermo['analytic_value'] = self._database_eval( a[1:],
                                                                    temperature )

            self.thermo[chk] = thermo
            return thermo

        def get_1000lnalpha(self, isotope, species, temperature=298.15):
            """Get an isotope fractionation factor from the database.

            Returns 1000lnalpha for the specified reaction.

            Args:
                isotope (str): the isotope fractionating (e.g. '13C')
                species (str): the species pair involved in the fractionation
                    (eg. r'CO2(g)/CO2(aq)')
                temperature: The temperature (degrees K) to calculate 1000lnalpha
                    at.
            Returns:
                1000lnalpha for the reaction.
            """

            if temperature < 50:
                print("Warning! Temperature (K) given as %i. Converting to %i" %
                      (temperature, temperature+273.15))
                temperature = temperature + 273.15

            # Check for cached value
            chk = isotope + species + str(temperature)
            if chk in self.alphas:
                return self.alphas[chk]


            value = 0
            tmp = self._ne_lookup(isotope, species)

            for a in tmp:
                if a[0] == '-add_logk':
                    b = a[1].split('_')
                    value += self.get_1000lnalpha(b[2], b[3], temperature)*int(a[2])
                elif a[0] == '-ln_alpha1000':
```

```
                value += self._database_eval(a[1:], temperature)
        self.alphas[chk] = value
        return value

    def get_alpha(self, isotope, species, temperature=298.15):
        """Get an isotope fractionation factor from the database.

        Returns alpha for the specified reaction. Arguments are the same as
        get_1000lnalpha.

        Args:
            isotope (str): the isotope fractionating (e.g. '13C')
            species (str): the species pair involved in the fractionation
                (eg. r'CO2(g)/CO2(aq)')
            temperature: The temperature (degrees K) to calculate 1000lnalpha
                at.
        Returns:
            alpha for the reaction.
        """

        ln1000a = self.get_1000lnalpha(isotope, species, temperature=298.15)
        return math.exp(0.001*ln1000a)

    def get_iso_stnd(self, isotope):
        """Get the absolute isotope ratio in the standard.

        Looks up the absolute mole fraction of the specified isotope present
        in the standard. The standard used, and it's composition, are defined
        in the phreeqc database.

        Args:
            isotope (str): The isotope desired (e.g. '13C')
        Returns:
            The mole fraction of 'isotope' present in the standard.
        """

        a = None
        pat = "([ \t]*?)-isotope([ \t]*?)\[{!s}\][ \t]"
        c = re.compile(pat.format(isotope))
        with open(self.db, 'r') as f:
            for line in f:
                if re.match(c, line):
                    a = line

        if a is None:
            raise Exception("No isotope standard found.")

        b = a.split()
        return float(b[3])

class PhreeqcInputLog(object):
    """Logs IPhreeqc input to a text file.

    PhreeqcInputLog is a utility class used by caves.Simulator objects to log
    iphreeqc input strings to a .phr text file. The resulting file is useful
    for understanding how the code runs and debugging failed models. The log
    file is also valid phreeqc input and can be run directly as a phreeqc
    script.
    """

    def __init__(self, filename, dbpath):
        """Initialise the object and create a log file.

        Creates a .phr log file at the specified location. The first few lines
        are initialised with the date, time and full path to the database used.

        Args:
            filename (str): Location to write the log file.
            dbpath (str): Path to the database file. Perferably full path.
        """

        self.filename = filename
        self.pq_input = open(self.filename,'w')
        self.pq_input.truncate()
        self._preamble()

        self.pq_input.write("\nDATABASE %s" % dbpath)
        self.pq_input.close()

    def _preamble(self):
        """Write log file preamble."""

        now = datetime.datetime.now()
        line1 = "#\tIPhreeqc input log.\n"
        line2 = "#\tDate:\t%i-%i-%i\n" % \
            (now.year, now.month, now.day)
        line3 = "#\tTime:\t%i:%i:%i\n" % \
            (now.hour, now.minute, now.second)

        self.pq_input.write(line1 + '\n')
        self.pq_input.write(line2)
        self.pq_input.write(line3)

    def _buffer(self):
        """Write break into log file for readability."""

        self.pq_input.write("\n\n" + '#' + '-'*20 + "\n\n")

    def add(self, string):
        """Write a string to log file.

        Args:
            string (str): PHREEQC input text to be added to log file.
        """
        self.pq_input = open(self.filename,'a')
        self._buffer()
        self.pq_input.write(string)
        self.pq_input.close()

class PostProcessor(object):
    """Performs offline calculations and formatting of model results.

    Post-processes model output from a caves.Simulator object, adding
    parameters calculated offline and making the output more readable.
    """

    def __init__(self, Simulator):
        """Run the post-processor.

        Does the following:
            - Adds f_ca (fraction of Ca remaining) and f_c (fraction of c
```

```
            remaining) parameters to the model output.
        - Calculates X/Ca ratios from mole outputs (Mg, Sr, Ba)
        - Rename some outputs to be more readable
        - Remove nonsense radiocarbon calculations from the degassing loop
            results.
        - Formats some PHREEQC-output parameter names to make them more
            readable.

    Args:
        Simulator: The Simulator object to operate on.
    """

    self.s = Simulator
    self.calculate_f()
    self.calculate_XCa()
    self.tidy()
    self.calculate_radiocarbon()
    self.set_none()

def calculate_f(self):
    """Calculate the fractions of carbon and calcium remaining at each
    model step.

    Calculates f_c (fraction of C) and f_ca (fraction of Ca) remaining in
    the solution at each model step relative to the amount present
    immediately following bedrock dissolution. Calculated values are added
    to the Simulator output dict.
    """

    init = output_filter(self.s.output, 'step_desc', 'dissolve')
    init_ca = init['Ca(mol/kgw)'][0]
    init_c = init['C(mol/kgw)'][0]
    ca = self.s.output['Ca(mol/kgw)']
    c = self.s.output['C(mol/kgw)']
    self.s.output['f_ca'] = [x / init_ca for x in ca]
    self.s.output['f_c'] = [x / init_c for x in c]

def calculate_XCa(self):
    """Calculate X/Ca ratios of solution and precipitates.

    X/Ca is calculated at each time step using a Rayleigh distillation
    model. X/Ca ratios for solution and any precipitate are added to the
    Simulator output dict.

    X includes Mg, Sr and Ba.
    """

    a = self.s.output

    trace_elements = ['Ba', 'Sr', 'Mg']
    dissolved_ratios = {k : [] for k in trace_elements}
    precipitate_ratios = {k : [] for k in trace_elements}

    for i, desc in enumerate(a['step_desc']):
        w = a['mass_H2O']
        ca = a['Ca(mol/kgw)']
        dissolved_ca = ca[i] * w[i]

        if 'CaCO3_precipitation' in desc:
            solid_ca = dissolved_ca - ca[i-1] * w[i-1]

        for x in trace_elements:
            x_dissolved = a[x+'(mol/kgw)'][i] * w[i]
            try:
                dissolved_ratios[x].append( x_dissolved / dissolved_ca )
            except ZeroDivisionError:
                dissolved_ratios[x].append( 0 )

            # precipitated trace element = amount lost from solution
            if 'CaCO3_precipitation' in desc:
                x_precip = x_dissolved - a[x+'(mol/kgw)'][i-1] * w[i-1]
                try:
                    precipitate_ratios[x].append( x_precip / solid_ca )
                except ZeroDivisionError:
                    precipitate_ratios[x].append( 0 )
            else:
                    precipitate_ratios[x].append( 0 )

    for x in trace_elements:
        self.s.output[x+'/Ca(mol/mol)'] = dissolved_ratios[x]
        self.s.output[x+'/Ca(mol/mol)_Calcite'] = precipitate_ratios[x]

def calculate_radiocarbon(self):
    """Sets all post-dissolution R14C values to be constant.

    Radiocarbon distribution is fully calculated during the bedrock
    dissolution step, but is not included in subsequent degassing /
    precipitation steps for three reasons:
        - It increases computation time.
        - When model steps are very small, 14C may cause non-convergence.
        - d13C-corrected R14C is not expected to change during these
            processes.

    To avoid confusion, spurious post-dissolution R14C values reported by
    IPhreeqc are set equal to the post-dissolution value.

    This method also adds a new column ('DCP') to the output. Note that
    the DCP calculation assumes modern = 100 pMC.
    """

    for key in self.s.output.keys():
        if 'R14C' in key:
            pmc = None
            for (i, value) in enumerate(self.s.output[key]):
                if 'bedrock' in self.s.output['step_desc'][i]:
                    pmc = self.s.output[key][i]
                elif pmc is not None:
                    self.s.output[key][i] = pmc

    atmo_a14c_init = 100 # placeholder value (pMC)
    self.s.output['DCP'] = [ (1-v/atmo_a14c_init)*100 for v in
                             self.s.output['R14C'] ]

def tidy(self):
    """Rename PHREEQC variable names with more reader-friendly syntax.

    Also removes a couple of unwanted outputs.
    """
    o = self.s.output
    rep_keys = {}
```

```
        # remove unwanted outputs (not of interest to most users)
        o.pop('soln')
        o.pop('mass_H2O')
        o.pop('pct_err')
        o.pop('temp(C)')
        o.pop('I_R(14C)_CO2(aq)')

        # rename some outputs for readability
        for key, data in o.items():
            if key[0:3] == 'I_R': # rename isotopes
                iso = key[key.find('(')+1:key.find(')')]
                if iso == '14C':
                    new_key = 'R' + iso + key[key.find(')')+1:]
                else:
                    new_key = 'd' + iso + key[key.find(')')+1:]
                rep_keys[key] = new_key
            elif key[0:2] == 'm_': # rename molalities
                new_key = key[2:]
                rep_keys[key] = new_key
            elif key[0:2] == 's_':
                new_key = 'moles_' + key[2:]
                rep_keys[key] = new_key

        for k1, k2 in rep_keys.items():
            o[k2] = o.pop(k1)

    def set_none(self):
        """Set default paramter returns (e.g. d18O = -999) to None."""

        for k, lst in self.s.output.items():
            new_lst = []
            for e in lst:
                if not(isinstance(e, str)) and e <= -999:
                    new_lst.append(None)
                else:
                    new_lst.append(e)
            self.s.output[k] = new_lst

# Begin radiocarbon calculation functions
def pmc(C14, d13C, stnd14C=1.175887709e-12):
    """Converts 14C/C absolute ratio to a d13C-corrected pMC value.

    Args:
        C14 : 14C/C ratio (absolute)
        d13C : V-PDB d13C values
        stnd14C : The 14C/C ratio in the standard
    Returns:
        the standard-normalised radiocarbon value in pMC.
    """

    return C14 / stnd14C * 100 * pow(0.975/(1+0.001*d13C),2)

def pmc_2_c14(R14C, d13C, stnd14C=1.175887709e-12):
    """Converts a d13C-corrected pMC value to a 14C/C absolute ratio.

    Conversion follows Stuvier & Pollach (1977).

    Args:
        R14C: A radiocarbon value in d13C corrected percent modern carbon
              (pmc).
        d13C: The corresponding d13C value.
        stnd14C: The standard 14C/C molar ratio. Default is
                 1.175887709e-12.
    Returns:
        14C/C ratio
    """

    return stnd14C * 0.01 * R14C * pow((1+0.001*d13C)/0.975,2)

def c14_to_pmc(C12, C13, C14, stnd13C=0.0111802, stnd14C=1.175887709e-12):
    """Returns isotope ratios given relative c isotope abundance data.

    Args:
        C12: Mole fraction 12C
        C13: Mole fraction 13C
        C14: Mole fraction 14C
        stnd13C: 13C/12C ratio in the standard. Default is 0.0111802 (VPDB)
        stnd13C: 14C/C ratio in the standard. Default is 1.175887709e-12

    Returns:
        d13C, R14C (pMC, d13C-corrected)
    """

    d13C = ((C13/C12)/stnd13C - 1) * 1000
    R14C = pmc(C14, d13C, stnd14C)
    return d13C, R14C

def pmc_normalise(R14C, d13C, stnd14C=1.175887709e-12):
    """Convert a 14C/C pmc-normalised ratio (returned by PHREEQC) input a
    'proper' d13C corrected pMC ratiocarbon ratio."""

    true_ratio = 0.01 * R14C * stnd14C
    return pmc(true_ratio, d13C, stnd14C)

def pmc_denormalise(pMC, d13C, stnd14C=1.175887709e-12):
    """Convert a d13C-corrected R14C (pMC) to a normalised 14C/C ratio, as used
    by PHREEQC."""

    c14_ratio = pmc_2_c14(pMC, d13C, stnd14C)
    return 100 * c14_ratio / stnd14C

# begin helper function definitions
def output_filter(src, key, value):
    """Filter lists contained in a dict by the value in one of the lists.

    The function takes a dict of lists, where all lists are of equal length. It
    returns a new list of dicts where the lists are shortened: only list
    entries at indices that meet specified criteria are copied.

    Args:
        src: Dictionary to filter. Each entry should be a list of equal length.
        key: The key in src to filter the lists by
        value: value in src[key] to include in output.
    Returns:
        A filtered version of src.
    """

    if isinstance(value, str):
        f = lambda v : value in v
    else:
```

```
        f = lambda v : value == v

    inds = [i for i,a in enumerate(src[key]) if f(a)]

    o = {}
    for k, v in src.items():
        o[k] = [a for i,a in enumerate(v) if i in inds]
    return o

def matlab_header_parse(dictionary):
    """Remove illegal characters from dictionary keys.

    To transfer data to matlab, certain characters are not allowed in field /
    array names (e.g. brackets, hypens). This function removes these characters
    in preparation for writing dicts (of numpy arrays) to a .mat file.

    Args:
        dictionary: A dict
    Returns:
        A dict with modified key names
    """

    b = {}
    for k in dictionary:
        new_k = k.replace(')', '') # because matlab is fussy
        new_k = new_k.replace('(','_') # about variable names
        new_k = new_k.replace('-','_') # and does not allow these
        new_k = new_k.replace('/','') # characters
        new_k = new_k.replace('[','')
        new_k = new_k.replace(']','')

        b[new_k] = dictionary[k]
    return b

def numpify(dictionary):
    """Prepare a dict of lists for writing to a .mat file.

    Convert the dict of lists to a dict of numpy arrays. Dict keys are edited
    if they contain matlab-illegal characters. Lists in the dict are converted
    to numpy arrays.

    Args:
        dictionary: A dict of lists. Each list should be composed of a single
            type.
    Returns:
        A modified dict, ready for writing to a .mat file.
    """

    a = matlab_header_parse(dictionary)
    b = {}
    for k in a:
        if k == "step_desc":
            b[k] = np.asarray( a[k], order='F' )
        else:
            b[k] = np.asarray( a[k], order='F' )
    return b

def save_mat(dict_of_lists, filename):
    """Save data to a .mat file for use with Matlab/Octave.

    Takes a dict of lists (e.g. model results) and saves them to a .mat file.
    Data are prepared for saving using the numpify() function.

    Args:
        dict_of_lists: Data to be saved for Matlab use.
    """

    out = numpify(dict_of_lists)
    sio.savemat( filename, out )

def save_pkl(data, filename):
    """Save data to a .pkl file for use with Python.

    Args:
        data: a pickle-able Python object.
        filename: Output file name/location.
    """

    with open(filename,'wb') as f:
        pickle.dump( data, f)

def save_csv(dictionary, filename):
    """Save data to a .csv file.

    For saving data to view in for use with other programs (e.g. Excel). Data
    should be provided as a dict of lists. Dict keys give column headers and
    lists give column values. All lists must be of equal length.

    In the resulting .csv, columnns are arranged alphabetically by header.

    Args:
        dictionary: The dictionary to write to file.
    """

    r = OrderedDict(sorted(dictionary.items()))
    a, b = zip(*[(k,v) for (k,v) in r.items()])
    c = zip(*b)

    with open(filename,'w', newline='') as f:
        writer = csv.writer(f,dialect='excel')
        writer.writerow(a)
        for row in c:
            writer.writerow(row)
```

### A.3.7 cavecalc/data/__init__.py

```
```

### A.3.8 cavecalc/data/defaults.py

```
"""Default Cavecalc Model Input Parameters"""

# Atmospheric gas end-member
atm_O2    =       0.21 # Atmospheric O2 (%)
atm_d18O  =       -10  # Rainfall d18O (per mil VSMOW)
atm_pCO2  =       270  # Atmospheric pCO2 (ppmv)
atm_d13C  =       -7   # Atmospheric CO2 d13C (per mil VPDB)
atm_R14C  =       100  # Atmospheric CO2 R14C (pmc)

# Soil gas end-member
soil_O2   =             0         # end-member soil gas O2 (%)
soil_R14C =             100       # end-member soil gas R14C (pmc)
soil_d13C =             -25       # end-member soil gas d13C (per mil VPDB)
soil_pCO2 =             20000     # end-member soil gas pCO2 (ppmv)

# Soil Gas Mixing
atmo_exchange =     0    # Atmospheric contribution to soil gas CO2 (0-1)
init_O2   =       'mix'    # actual soil gas O2 content (%)
init_R14C =       'mix'    # actual soil gas R14C* (pmc)
init_d13C =       'mix'    # actual soil gas d13C* (per mil VPDB)
init_pCO2 =       'mix'    # actual soil gas pCO2* (ppmv). 'atm' to equilibrate with atmospheric↩
     , 'soil' to equilibrate with soil_pCO2 (after atmospheric mixing)
init_solution_d13c = 'n/a'

# Soil Metals
soil_Ba =               0         # soil water Ba concentration (mmol/kg water)
soil_Ca =               0         # soil water Ca concentration (mmol/kg water)
soil_Mg =               0         # soil water Mg concentration (mmol/kg water)
soil_Sr =               0         # soil water Sr concentration (mmol/kg water)
soil_pH =               4         # initial soil pH estimate - not important
soil_d44Ca =            0         # soil water Ca d44Ca (per mil 915a)

# Bedrock Chemistry
bedrock_BaCa =          0             # bedrock Ba/Ca (mmol/mol)
bedrock_MgCa =          0             # bedrock Mg/Ca (mmol/mol)
bedrock_SrCa =          0             # bedrock Sr/Ca (mmol/mol)
bedrock_d13C =          0             # bedrock d13C (per mil VPDB)
bedrock_d18O =          0             # bedrock d18O (per mil VSMOW)
bedrock_d44Ca =         0             # Bedrock d44Ca (per mil SRM 915a)
bedrock_mineral =       'Calcite'     # Bedrock mineralogy ('Calcite' or 'Dolomite')

# Bedrock Dissolution Conditions
bedrock =               10    # moles bedrock (defaults to excess)
bedrock_pyrite =        0             # moles of pyrite present during bedrock dissolution
gas_volume =            10    # Volume of soil gas present during bedrock dissolution (L/kg ↩
     water)
reprecip =              False     # Allow calcite re-precipitation during bedrock dissolution

# Cave Air
cave_O2 =               0.21
cave_pCO2 =             1000
cave_d13C =             -10
cave_R14C =             100
cave_d18O =             0
cave_air_volume =       0

# General
temperature =           20      # Temperature in degrees celsius
kinetics_mode =         'closed_system_rayleigh'     # Specifies how to run the model (see ↩
     types_and_limits.py for options)

# Scripting Options
co2_decrement =         0.5     # Fraction of CO2(aq) removed on each degassing step
calcite_sat_limit =     1       # Only used if kinetics_mode = 'ss'. CaCO3 only precipitates ↩
     when saturation index exceeds this value.

# Additional PHREEQC output
# SELECTED_OUTPUT (model-essential outputs are set in phreeqc_templates.py)
isotopes =    """R(44Ca) R(18O) R(13C) R(18O)_HCO3 R(13C)_HCO3 R(18O)_CO3-2
                 R(13C)_CO3-2 R(44Ca)_Calcite R(18O)_Calcite R(13C)_Calcite"""
molalities =    "HCO3-   CO3-2"
totals =        ""

# File IO Settings
phreeqc_log_file =      False           # If selected, a .phr file will be included in the ↩
     output.
out_dir =               ""              # Save location
database =              'oxotope.dat'   # database file
phreeqc_log_file_name = 'log_{}.phr'    # The .phr file name (if saved). {} will be replaced ↩
     by the model number.
```

### A.3.9 cavecalc/data/phreeqc_templates.py

```
"""PHREEQC format strings - used by caves.py"""

SEL_OUTPUT_CORE = """
SELECTED_OUTPUT
    -high_precision      true
    -reset               false
    -simulation          false
    -state               false
    -solution            true
    -distance            false
    -time                false
    -step                false
    -ph                  true
    -pe                  false
    -reaction            false
    -temperature         true
    -alkalinity          false
    -ionic_strength      false
    -water               true
```

```
    -charge_balance      false
    -percent_error       true{saturation_indices!s}
    -totals              Ca Mg Sr Ba C O{totals!s}
    -molalities          CO2{molalities!s}
    -isotopes            R(13C)_CO2(aq)  R(18O)_CO2(aq) R(14C)_CO2(aq) R(14C)
                         {isotopes!s}
    -solid_solutions     Calcite
"""

INITIAL_CHEM = """
SOLUTION 1 Soilwater
    temp     {temp:12G}
    units    mmol/kgw
    pH       {ph:12G}     charge
    Ca       {ca:12G}
    Mg       {mg:12G}
    Sr       {sr:12G}
    Ba       {ba:12G}
    Cl       {cl:12G}
    C        {c:12G}     # calculate using _get_init_co()
    O(0)     {o:12G}     # calculate using _get_init_co()
    [18O]    {d18O:12G}    # permil
    [13C]    {d13C:12G}    # permil
    [14C]    {R14C:12G}    # pmc
    [44Ca]   {d44Ca:12G}    # permil
    -water   1     # kg
"""

DUMMY_CHEM_CO = """
SOLUTION 0 Determine C and O concentrations
    temp     {temp:12G}
    units    mmol/kgw
    pH       {ph:12G}     charge
    Ca       {ca:12G}
    Mg       {mg:12G}
    Sr       {sr:12G}
    Ba       {ba:12G}
    Cl       {cl:12G}
    C        1     CO2(g)     {co2_si:12G}
    O(0)     1     O2(g)      {o2_si:12G}
    -water   1     # kg
"""

DUMMY_CHEM_D13C = """
SOLUTION 0 Determine solution d13C from gas
    temp     {temp:12G}
    units    mmol/kgw
    pH       {ph:12G}     charge
    Ca       {ca:12G}
    Mg       {mg:12G}
    Sr       {sr:12G}
    Ba       {ba:12G}
    Cl       {cl:12G}
    C        {c:12G}     # calculate using _get_init_co()
    O(0)     {o:12G}     # calculate using _get_init_co()
    [18O]    {d18O:12G}    # permil
    [13C]    {d13C:12G}    # permil
    [14C]    {R14C:12G}    # pmc
    [44Ca]   {d44Ca:12G}    # permil
    -water   1     # kg
"""

WATER_EQUILIBRATE = """
REACTION             # force isotopic equilibration
    H2O(g)    0
"""


BEDROCK_PHASES = """
PHASES
Bedrock
    {!s}
"""

# BEDROCK_DISS 1 uses Calcite saturation (more geochemically realistic)
# for use with quantitative dissolutions
BEDROCK_DISS1 = """
EQUILIBRIUM_PHASES
    Pyrite  0    {moles_pyrite}   dissolve_only
    Calcite 0    Bedrock {moles_bedrock} dissolve_only
        -force_equality


"""

# BEDROCK_DISS 2 uses bedrock dissolution parameters
# for use with incongruent dissolution (i.e. where the name 'Calcite' is used
# for a precipitating reaction, and hence can't be used here).
BEDROCK_DISS2 = """
EQUILIBRIUM_PHASES
    Bedrock 0    {moles_bedrock} dissolve_only
    Pyrite  0    {moles_pyrite}   dissolve_only
        -force_equality


"""

CO2_OUT = """
REACTION
    CO2          {molefrac_co2:12G}
    [13C]O2      {molefrac_13C:12G}
    CO[18O]      {molefrac_18O:12G}
    [14C]O2      {molefrac_14C:12G}
    {gas_moles:12G}
"""

CO2_OUT2 = """
REACTION
    CO2          {molefrac_co2:12G}
    [13C]O2      {molefrac_13C:12G}
    CO[18O]      {molefrac_18O:12G}
    {gas_moles:12G}
"""

CO2_GAS = """
GAS_PHASE {name}
    -fixed_pressure
    -pressure        {pressure:12G}
```

```
    -volume             {volume:12G}
    -temperature        {temperature:12G}
    CO2(g)              {pp_co2:12G}
    CO[18O](g)          {pp_C18O:12G}
    C[18O]2(g)          {pp_C18O2:12G}
    [13C]O2(g)          {pp_13C:12G}
    [13C]O[18O](g)      {pp_13C18O:12G}
    [13C][18O]2(g)      {pp_13C18O2:12G}
    [14C]O2(g)          {pp_14C:12G}
    [14C]O[18O](g)      {pp_14C18O:12G}
    [14C][18O]2(g)      {pp_14C18O2:12G}
    O2(g)               {pp_O2:12G}
    O[18O](g)           {pp_O18O:12G}
    Q2(g)               {pp_N2:12G}

"""

CALCITE_SS = """
SOLID_SOLUTION 1  Calcite
    Calcite
    -comp  Calcite 0
    -comp CaCO2[18O](s) 0
    -comp CaCO[18O]2(s) 0
    -comp CaC[18O]3(s) 0
    -comp Ca[13C]O3(s) 0
    -comp Ca[13C]O2[18O](s) 0
    -comp Ca[13C]O[18O]2(s) 0
    -comp Ca[13C][18O]3(s) 0
    -comp [44Ca]CO3(s) 0
    -comp [44Ca]CO2[18O](s) 0
    -comp [44Ca]CO[18O]2(s) 0
    -comp [44Ca]C[18O]3(s) 0
    -comp [44Ca][13C]O3(s) 0
    -comp [44Ca][13C]O2[18O](s) 0
    -comp [44Ca][13C]O[18O]2(s) 0
    -comp [44Ca][13C][18O]3(s) 0
    # -comp Ca[14C]O3(s) 0   # radiocarbon may cause non-convergence
    # -comp Ca[14C]O2[18O](s) 0
    # -comp Ca[14C]O[18O]2(s) 0
    # -comp Ca[14C][18O]3(s) 0
    # -comp [44Ca][14C]O3(s) 0
    # -comp [44Ca][14C]O2[18O](s) 0
    # -comp [44Ca][14C]O[18O]2(s) 0
    # -comp [44Ca][14C][18O]3(s) 0
"""

SS_MG = """   # Mg components:          -
    -comp MgCO3(s) 0
    -comp MgCO2[18O](s) 0
    -comp MgCO[18O]2(s) 0
    -comp MgC[18O]3(s) 0
    -comp Mg[13C]O3(s) 0
    -comp Mg[13C]O2[18O](s) 0
    -comp Mg[13C]O[18O]2(s) 0
    -comp Mg[13C][18O]3(s) 0
    # -comp Mg[14C]O3(s) 0
    # -comp Mg[14C]O2[18O](s) 0
    # -comp Mg[14C]O[18O]2(s) 0
    # -comp Mg[14C][18O]3(s) 0

"""

SS_SR = """   # Sr components:
     -comp SrCO3(s) 0
    -comp SrCO2[18O](s) 0
    -comp SrCO[18O]2(s) 0
    -comp SrC[18O]3(s) 0
    -comp Sr[13C]O3(s) 0
    -comp Sr[13C]O2[18O](s) 0
    -comp Sr[13C]O[18O]2(s) 0
    -comp Sr[13C][18O]3(s) 0
    # -comp Sr[14C]O3(s) 0
    # -comp Sr[14C]O2[18O](s) 0
    # -comp Sr[14C]O[18O]2(s) 0
    # -comp Sr[14C][18O]3(s) 0
"""

SS_BA = """   # Ba components:
     -comp BaCO3(s) 0
    -comp BaCO2[18O](s) 0
    -comp BaCO[18O]2(s) 0
    -comp BaC[18O]3(s) 0
    -comp Ba[13C]O3(s) 0
    -comp Ba[13C]O2[18O](s) 0
    -comp Ba[13C]O[18O]2(s) 0
    -comp Ba[13C][18O]3(s) 0
    # -comp Ba[14C]O3(s) 0
    # -comp Ba[14C]O2[18O](s) 0
    # -comp Ba[14C]O[18O]2(s) 0
    # -comp Ba[14C][18O]3(s) 0
"""
```

## A.3.10 cavecalc/data/types_and_limits.py

```
"""Encodes accepted values and ranges for model input parameters.

This data is used by the setter.SettingsObject.validate_entry() method. To
verify input parameter types and values.

      'str'       :   x must be a string
      (a, b)      :   x must be numeric, with x > a and x < b. a or b == None
                      means there is no lower or upper bound, respectively.
      (a, b, ...):    As above. If x is not numeric it must match an entry in ...
      True        :   x must be boolean
      [...]       :   x must match a entry in the list
"""
```

```
database =          'str'
phreeqc_log_file =      True
phreeqc_log_file_name = 'str'
out_dir =               'str'
temperature =           (-273.15, None)
co2_decrement =         (0, 1)
calcite_sat_limit =     (0, None)
bedrock =               (0, None)
bedrock_mineral =       ['Calcite', 'Dolomite']
bedrock_pyrite =        (0, None)
bedrock_d44Ca =         (None, None)
bedrock_d13C =          (None, None)
bedrock_d18O =          (None, None)
bedrock_MgCa =          (0, None)
bedrock_SrCa =          (0, None)
bedrock_BaCa =          (0, None)
atmo_exchange =     (0, 1)
gas_volume =        (0, None)
atm_O2 =        (0, 100)
atm_pCO2 =      (0, None)
atm_d13C =      (None, None)
atm_R14C =      (0, None)
atm_d18O =      (None, None)
cave_O2 =       (0, 100)
cave_pCO2 =     (0, None)
cave_d13C =     (None, None)
cave_R14C =     (0, None)
cave_d18O =     (None, None)
cave_air_volume = (0, None)
init_O2 =       (0, 100, 'soil', 'atm', 'mix')
init_pCO2 =     (0, None, 'soil', 'atm', 'mix')
init_d13C =     (None, None, 'soil', 'atm', 'mix')
init_R14C =     (0, None, 'soil', 'atm', 'mix')
init_solution_d13c = (None, None, 'n/a')
soil_O2 =       (0, 100)
soil_pH =       (0, 14)
soil_pCO2 =     (0, None)
soil_Ca =       (0, None)
soil_Mg =       (0, None)
soil_Sr =       (0, None)
soil_Ba =       (0, None)
soil_d13C =     (None, None)
soil_R14C =     (0, None)
soil_d44Ca =    (None, None)
kinetics_mode = ['closed_system_rayleigh', 'open_system_single_step', 'allow_supersaturation'↩
    , 'allow_supersaturation_max', 'degas_only', 'diss_only']
reprecip =      True
totals =        'str'
molalities =    'str'
isotopes =      'str'
```

## A.3.11 cavecalc/data/oxotope.dat

```
SOLUTION_MASTER_SPECIES
E               e-              0       0               0.0
H               H3O+            -1      H               1.008
H(0)            H2              0       H
H(1)            H3O+            -1      H
O               H2O             0       O               16.00
O(0)            O2              0       O
O(-2)           H2O             0       O
Ca              Ca+2            0       Ca              40.08
Mg              Mg+2            0       Mg              24.312
Na              Na+             0       Na              22.9898
K               K+              0       K               39.102
Fe              Fe+2            0.0     Fe              55.847
Fe(+2)          Fe+2            0.0     Fe
Fe(+3)          Fe+3            -2.0    Fe
Al              Al+3            0.0     Al              26.9815
Si              H4SiO4          0.0     SiO2            28.0843
Cl              Cl-             0       Cl              35.453
C               CO2             0       HCO3            12.0111
C(4)            CO2             0       HCO3
C(-4)           CH4             0       CH4
S               SO4-2           0       S               31.972
S(6)            SO4-2           0       SO4
S(-2)           HS-             1       S
N               NO3-            0       N               14.0067
N(+5)           NO3-            0       N
N(+3)           NO2-            0       N
N(0)            N2              0       N
N(-3)           NH4+            0       N
P               PO4-3           2.0     P               30.9738
F               F-              0.0     F               18.9984
Br              Br-             0.0     Br              79.904
Q               Q2              0       Q               14.0067 # cavecalc gas placeholder (N2)
Sr              Sr+2            0       Sr              87.62   # cavecalc addition
Ba              Ba+2            0       Ba              137.327 # cavecalc addition

SOLUTION_SPECIES

Q2 = Q2
     log_k              0.000

Sr+2 = Sr+2
     log_k              0.000

Ba+2 = Ba+2
     log_k              0.000

H3O+ = H3O+
     log_k              0.000
     -gamma             9.0000     0.0000

e- = e-
     log_k              0.000

H2O = H2O
     log_k              0.000
```

```
Ca+2 = Ca+2
     log_k             0.000
     -gamma            5.0000    0.1650

Mg+2 = Mg+2
     log_k             0.000
     -gamma            5.5000    0.2000

Na+ = Na+
     log_k             0.000
     -gamma            4.0000    0.0750

K+ = K+
     log_k             0.000
     -gamma            3.5000    0.0150

Fe+2 = Fe+2
     log_k             0.000
     -gamma            6.0000    0.0000

Al+3 = Al+3
     log_k             0.000
     -gamma            9.0000    0.0000

H4SiO4 = H4SiO4
        log_k             0.000

Cl- = Cl-
     log_k             0.000
     -gamma            3.5000    0.0150

SO4-2 = SO4-2
     log_k             0.000
     -gamma            5.0000   -0.0400

NO3- = NO3-
     log_k             0.000
     -gamma            3.0000    0.0000

PO4-3 = PO4-3
        log_k             0.000
        -gamma    4.0000    0.0000

F- = F-
        log_k             0.000
        -gamma    3.5000    0.0000

Br- = Br-
        log_k             0.000
        -gamma    3.0000    0.0000

2H2O = OH- + H3O+
     log_k            -14.000
     delta_h          13.362 kcal
     -analytic        -283.971        -0.05069842   13323.0     102.24447       -1119669.0
     -gamma           3.5000    0.0000

6 H2O = O2 + 4 H3O+ + 4 e-
     log_k            -86.08
     delta_h          134.79 kcal

2 H3O+ + 2 e- = H2 + 2H2O
     log_k            -3.15
     delta_h          -1.759 kcal

CO2 + 2H2O = HCO3- + H3O+
    # Plummer et al. (1982)
     log_k            -6.352
     delta_h          2.177 kcal
     -analytic        -356.3094        -0.06092         21834.37         126.8339         -1684915

CO2 + 3H2O = CO3-2 + 2H3O+
    # Plummer et al. (1982)
     log_k            -16.681
     delta_h          5.738 kcal
     -analytic        -464.1965        -0.09344813   26986.16     165.75951       -2248628.9

CO3-2 + 10 H3O+ + 8 e- = CH4 + 13 H2O
     log_k            41.071
     delta_h          -61.039 kcal

SO4-2 + H3O+ = HSO4- + H2O
     log_k            1.988
     delta_h          3.85 kcal
     -analytic        -56.889         0.006473     2307.9         19.8858          0.0

SO4-2 + 9 H3O+ + 8 e- = HS- + 13 H2O
     log_k            33.65
     delta_h          -60.140 kcal
     -gamma           3.5000    0.0000

HS- + H2O = S-2 + H3O+
     log_k            -12.918
     delta_h          12.1 kcal
     -gamma           5.0000    0.0000

HS- + H3O+ = H2S + H2O
     log_k            6.994
     delta_h          -5.300 kcal
     -analytic        -11.17     0.02386        3279.0

NO3- + 2 H3O+ + 2 e- = NO2- + 3H2O
     log_k            28.570
     delta_h          -43.760 kcal
     -gamma           3.0000    0.0000

2 NO3- + 12 H3O+ + 10e- = N2 + 18 H2O
     log_k            207.080
     delta_h          -312.130 kcal

NH4+ + H2O = NH3 + H3O+
     log_k            -9.252
     delta_h          12.48 kcal
     -analytic        0.6322     -0.001225   -2835.76

NO3- + 10 H3O+ + 8 e- = NH4+ + 13 H2O
     log_k            119.077
     delta_h          -187.055 kcal
     -gamma           2.5000    0.0000

NH4+ + SO4-2 = NH4SO4-
     log_k            1.11

PO4-3 + H3O+ = HPO4-2 + H2O
        log_k             12.346
```

```
        delta_h  -3.530   kcal
        -gamma     4.0000     0.0000

PO4-3 + 2 H3O+ = H2PO4- + 2H2O
        log_k              19.553
        delta_h  -4.520   kcal
        -gamma     4.5000     0.0000

H3O+ + F- = HF + H2O
        log_k              3.18
        delta_h  3.18     kcal
        -analytic          -2.033          0.012645       429.01

H3O+ + 2 F- = HF2- + H2O
        log_k              3.760
        delta_h  4.550    kcal

Ca+2 + OH- = CaOH+
     log_k              -12.780

Ca+2 + CO3-2 = CaCO3
     log_k              3.224
     delta_h            3.545  kcal
     -analytic          -1228.732      -0.299440     35512.75       485.818

#Ca+2 + HCO3- = CaHCO3+
Ca+2 + CO3-2 + H3O+ = CaHCO3+ + H2O
     log_k              11.435
     delta_h            -0.871  kcal
     -analytic          1317.0071      0.34546894    -39916.84      -517.70761      563713.9
     -gamma             5.4000     0.0000

Ca+2 + SO4-2 = CaSO4
     log_k              2.300
     delta_h            1.650  kcal

Ca+2 + HSO4- = CaHSO4+
     log_k              1.08

Ca+2 + PO4-3 = CaPO4-
     log_k              6.459
     delta_h            3.100  kcal

Ca+2 + HPO4-2 = CaHPO4
     log_k              2.739
     delta_h            3.3  kcal

Ca+2 + H2PO4- = CaH2PO4+
     log_k              1.408
     delta_h            3.4  kcal

Ca+2 + F- = CaF+
        log_k              0.940
        delta_h  4.120    kcal

Mg+2 + OH- = MgOH+
     log_k              -11.440
     delta_h            15.952  kcal

Mg+2 + CO3-2 = MgCO3
     log_k              2.98
     delta_h            2.713  kcal
     -analytic          0.9910         0.00667

Mg+2 + H3O+ + CO3-2 = MgHCO3+ + H2O
     log_k              11.399
     delta_h            -2.771  kcal
     -analytic          48.6721        0.03252849    -2614.335      -18.00263       563713.9

Mg+2 + SO4-2 = MgSO4
     log_k              2.370
     delta_h            4.550  kcal

Mg+2 + PO4-3 = MgPO4-
        log_k              6.589
        delta_h  3.100    kcal

Mg+2 + HPO4-2 = MgHPO4
        log_k              2.87
        delta_h  3.3  kcal

Mg+2 + H2PO4- = MgH2PO4+
        log_k              1.513
        delta_h  3.4  kcal

Mg+2 + F- = MgF+
        log_k              1.820
        delta_h  3.200    kcal

Na+ + OH- = NaOH
     log_k              -14.180

Na+ + CO3-2 = NaCO3-
     log_k              1.270
     delta_h            8.910  kcal

Na+ + HCO3- = NaHCO3
     log_k              -0.25

Na+ + SO4-2 = NaSO4-
     log_k              0.700
     delta_h            1.120  kcal

Na+ + HPO4-2 = NaHPO4-
        log_k              0.29

Na+ + F- = NaF
        log_k              -0.240

K+ + OH- = KOH
     log_k              -14.460

K+ + SO4-2 = KSO4-
     log_k              0.850
     delta_h            2.250  kcal
     -analytic          3.106   0.0    -673.6

K+ + HPO4-2 = KHPO4-
        log_k              0.29

Fe+2 + 2H2O = FeOH+ + H3O+
     log_k              -9.500
     delta_h            13.200  kcal

Fe+2 + Cl- = FeCl+
     log_k              0.140
```

```
Fe+2 + CO3-2 = FeCO3
     log_k             4.380

Fe+2 + HCO3- = FeHCO3+
     log_k             2.0

Fe+2 + SO4-2 = FeSO4
     log_k             2.250
     delta_h           3.230  kcal

Fe+2 + HSO4- = FeHSO4+
     log_k             1.08

Fe+2 + 2HS- = Fe(HS)2
     log_k             8.95

Fe+2 + 3HS- = Fe(HS)3-
     log_k             10.987

Fe+2 + HPO4-2 = FeHPO4
       log_k             3.6

Fe+2 + H2PO4- = FeH2PO4+
       log_k             2.7

Fe+2 + F- = FeF+
       log_k             1.000

Fe+2 = Fe+3 + e-
       log_k             -13.020
       delta_h  9.680    kcal
       -gamma    9.0000     0.0000

Fe+3 + 2H2O = FeOH+2 + H3O+
       log_k             -2.19
       delta_h  10.4     kcal

Fe+3 + 4H2O = Fe(OH)2+ + 2 H3O+
       log_k             -5.67
       delta_h  17.1     kcal

Fe+3 + 6 H2O = Fe(OH)3 + 3 H3O+
       log_k             -12.56
       delta_h  24.8     kcal

Fe+3 + 8 H2O = Fe(OH)4- + 4 H3O+
       log_k             -21.6
       delta_h  31.9     kcal

2 Fe+3 + 4 H2O = Fe2(OH)2+4 + 2 H3O+
       log_k             -2.95
       delta_h  13.5     kcal

3 Fe+3 + 8 H2O = Fe3(OH)4+5 + 4 H3O+
       log_k             -6.3
       delta_h  14.3     kcal

Fe+3 + Cl- = FeCl+2
       log_k             1.48
       delta_h  5.6      kcal

Fe+3 + 2 Cl- = FeCl2+
       log_k             2.13

Fe+3 + 3 Cl- = FeCl3
       log_k             1.13

Fe+3 + SO4-2 = FeSO4+
       log_k             4.04
       delta_h  3.91     kcal

Fe+3 + HSO4- = FeHSO4+2
       log_k             2.48

Fe+3 + 2 SO4-2 = Fe(SO4)2-
       log_k             5.38
       delta_h  4.60             kcal

Fe+3 + HPO4-2 = FeHPO4+
       log_k             5.43
       delta_h  5.76             kcal

Fe+3 + H2PO4- = FeH2PO4+2
       log_k             5.43

Fe+3 + F- = FeF+2
       log_k             6.2
       delta_h  2.7              kcal

Fe+3 + 2 F- = FeF2+
       log_k             10.8
       delta_h  4.8              kcal

Fe+3 + 3 F- = FeF3
       log_k             14.0
       delta_h  5.4              kcal

Al+3 + 2H2O = AlOH+2 + H3O+
       log_k             -5.00
       delta_h  11.49    kcal
       -analytic         -38.253        0.0          -656.27        14.327

Al+3 + 4 H2O = Al(OH)2+ + 2 H3O+
       log_k             -10.1
       delta_h  26.90            kcal
       -analytic         88.500         0.0          -9391.6        -27.121

Al+3 + 6 H2O = Al(OH)3 + 3 H3O+
       log_k             -16.9
       delta_h  39.89            kcal
       -analytic         226.374        0.0          -18247.8       -73.597

Al+3 + 8 H2O = Al(OH)4- + 4 H3O+
       log_k             -22.7
       delta_h  42.30            kcal
       -analytic         51.578         0.0          -11168.9       -14.865

Al+3 + SO4-2 = AlSO4+
       log_k             3.5
       delta_h  2.29  kcal

Al+3 + 2SO4-2 = Al(SO4)2-
       log_k             5.0
       delta_h  3.11  kcal
```

```
Al+3 + HSO4- = AlHSO4+2
        log_k           0.46

Al+3 + F- = AlF+2
        log_k           7.000
        delta_h 1.060   kcal

Al+3 + 2 F- = AlF2+
        log_k           12.700
        delta_h 1.980   kcal

Al+3 + 3 F- = AlF3
        log_k           16.800
        delta_h 2.160   kcal

Al+3 + 4 F- = AlF4-
        log_k           19.400
        delta_h 2.200   kcal

Al+3 + 5 F- = AlF5-2
        log_k           20.600
        delta_h 1.840   kcal

Al+3 + 6 F- = AlF6-3
        log_k           20.600
        delta_h -1.670  kcal

H4SiO4 + H2O = H3SiO4- + H3O+
        log_k           -9.83
        delta_h 6.12            kcal
        -analytic       -302.3724       -0.050698       15669.69        108.18466       -1119669.0

H4SiO4 + 2H2O= H2SiO4-2 + 2 H3O+
        log_k           -23.0
        delta_h 17.6            kcal
        -analytic       -294.0184       -0.072650       11204.49        108.18466       -1119669.0

H4SiO4 + 4 H3O+ + 6 F- = SiF6-2 + 8 H2O
        log_k           30.180
        delta_h -16.260         kcal
PHASES
Calcite
    # Plummer et al. (1982)
     CaCO3 = CO3-2 + Ca+2
     log_k          -8.480
     delta_h        -2.297 kcal
     -analytic      -171.9065       -0.077993       2839.319        71.595

Dolomite
        CaMg(CO3)2 = Ca+2 + Mg+2 + 2 CO3-2
        log_k           -17.090
        delta_h -9.436 kcal

Siderite
        FeCO3 = Fe+2 + CO3-2
        log_k           -10.890
        delta_h -2.480 kcal

Gypsum
     CaSO4:2H2O = Ca+2 + SO4-2 + 2 H2O
     log_k          -4.580
     delta_h        -0.109 kcal
     -analytic      68.2401         0.0             -3221.51        -25.0627

Anhydrite
     CaSO4 = Ca+2 + SO4-2
     log_k          -4.360
     delta_h        -1.710 kcal
     -analytic      197.52          0.0             -8669.8         -69.835

Hydroxyapatite
        Ca5(PO4)3OH + 4 H3O+ = 5H2O + 3 HPO4-2 + 5 Ca+2
        log_k           -3.421
        delta_h -36.155 kcal

Fluorite
        CaF2 = Ca+2 + 2 F-
        log_k           -10.600
        delta_h 4.690 kcal
        -analytic       66.348          0.0             -4298.2         -25.271

SiO2(a)
        SiO2 + 2 H2O = H4SiO4
        log_k           -2.710
        delta_h 3.340 kcal
        -analytic       -0.26           0.0             -731.0

Chalcedony
        SiO2 + 2 H2O = H4SiO4
        log_k           -3.550
        delta_h 4.720 kcal
        -analytic       -0.09           0.0             -1032.0

Quartz
        SiO2 + 2 H2O = H4SiO4
        log_k           -3.980
        delta_h 5.990 kcal
        -analytic       0.41            0.0             -1309.0

Gibbsite
        Al(OH)3 + 3 H3O+ = Al+3 + 6 H2O
        log_k           8.110
        delta_h -22.800 kcal

Al(OH)3(a)
        Al(OH)3 + 3 H3O+ = Al+3 + 6 H2O
        log_k           10.800
        delta_h -26.500 kcal

Kaolinite
        Al2Si2O5(OH)4 + 6 H3O+ = 7H2O + 2 H4SiO4 + 2 Al+3
        log_k           7.435
        delta_h -35.300 kcal

Albite
        NaAlSi3O8 + 8 H2O = Na+ + Al(OH)4- + 3 H4SiO4
        log_k           -18.002
        delta_h 25.896 kcal

Anorthite
        CaAl2Si2O8 + 8 H2O = Ca+2 + 2 Al(OH)4- + 2 H4SiO4
        log_k           -19.714
        delta_h 11.580 kcal
```

```
K-feldspar
        KAlSi3O8 + 8 H2O = K+ + Al(OH)4- + 3 H4SiO4
        log_k           -20.573
        delta_h 30.820  kcal

K-mica
        KAl3Si3O10(OH)2 + 10 H3O+ = K+ + 3 Al+3 + 3 H4SiO4 + 10H2O
        log_k           12.703
        delta_h -59.376 kcal

Chlorite(14A)
        Mg5Al2Si3O10(OH)8 + 16 H3O+ = 5Mg+2 + 2Al+3 + 3H4SiO4 + 22H2O
        log_k           68.38
        delta_h -151.494 kcal

Ca-Montmorillonite
        Ca0.165Al2.33Si3.67O10(OH)2 + 14 H2O = 0.165Ca+2 + 2.33 Al(OH)4- + 3.67 H4SiO4 + 2 ↩
            H3O+
        log_k           -45.027
        delta_h 58.373  kcal

Talc
        Mg3Si4O10(OH)2 + 4 H2O + 6 H3O+ = 3 Mg+2 + 4 H4SiO4 + 6H2O
        log_k           21.399
        delta_h -46.352 kcal

Illite
        K0.6Mg0.25Al2.3Si3.5O10(OH)2 + 12.4H2O = 0.6K+ + 0.25Mg+2 + 2.3Al(OH)4- + 3.5H4SiO4 +↩
            1.2H3O+
        log_k           -40.267
        delta_h 54.684  kcal

Chrysotile
        Mg3Si2O5(OH)4 + 6 H3O+ = 7H2O + 2 H4SiO4 + 3 Mg+2
        log_k           32.200
        delta_h -46.800 kcal
        -analytic       13.248          0.0             10217.1         -6.1894

Sepiolite
        Mg2Si3O7.5OH:3H2O + 4 H3O+ + 0.5H2O = 2 Mg+2 + 3 H4SiO4 + 4H2O
        log_k           15.760
        delta_h -10.700 kcal

Sepiolite(d)
        Mg2Si3O7.5OH:3H2O + 4 H3O+ + 0.5H2O = 2 Mg+2 + 3 H4SiO4 + 4H2O
        log_k           18.660

Hematite
        Fe2O3 + 6 H3O+ = 2 Fe+3 + 9 H2O
        log_k           -4.008
        delta_h -30.845 kcal

Goethite
        FeOOH + 3 H3O+ = Fe+3 + 5 H2O
        log_k           -1.000
        delta_h         -14.48 kcal

Fe(OH)3(a)
        Fe(OH)3 + 3 H3O+ = Fe+3 + 6 H2O
        log_k           4.891

Pyrite
     FeS2 + 2H3O+ + 2e- = Fe+2 + 2HS- + 2H2O
     log_k              -18.479
     delta_h            11.300 kcal

FeS(ppt)
        FeS + H3O+ = Fe+2 + HS- + H2O
        log_k           -3.915

Mackinawite
        FeS + H3O+ = Fe+2 + HS- + H2O
        log_k           -4.648

Sulfur
        S + 2 H3O+ + 2e- = H2S + 2H2O
        log_k           4.882
        delta_h -9.5 kcal

Vivianite
        Fe3(PO4)2:8H2O = 3 Fe+2 + 2 PO4-3 + 8 H2O
        log_k           -36.000

Halite
        NaCl = Na+ + Cl-
        log_k           1.582
        delta_h 0.918 kcal

CO2(g)
     CO2(g) = CO2
     log_k              -1.468
     delta_h            -4.776 kcal
     -analytic          108.3865        0.01985076      -6919.53        -40.45154       669365.0

O2(g)
     O2 = O2
#    log_k       -2.960
#    delta_h -1.844  kcal
     # log K from llnl.dat  Dec 8, 2010
     log_k      -2.8983
     -analytic  -7.5001e+000 7.8981e-003 0.0000e+000 0.0000e+000 2.0027e+005

H2(g)
     H2 = H2
     log_k              -3.150
     delta_h            -1.759 kcal

H2O(g)
     H2O = H2O
     log_k              1.51
     delta_h            -44.03 kJ

N2(g)
     N2 = N2
     log_k              -3.260
     delta_h            -1.358 kcal

Q2(g)
     Q2 = Q2
     log_k              -100.000

H2S(g)
     H2S = H2S
     log_k              -0.997
```

```
      delta_h          -4.570 kcal

CH4(g)
      CH4 = CH4
      log_k            -2.860
      delta_h          -3.373 kcal

NH3(g)
      NH3 = NH3
      log_k            1.770
      delta_h          -8.170 kcal

Melanterite
        FeSO4:7H2O = 7 H2O + Fe+2 + SO4-2
        log_k            -2.209
        delta_h 4.910              kcal
        -analytic        1.447         -0.004153       0.0             0.0             -214949.0

Alunite
        KAl3(SO4)2(OH)6 + 6 H3O+ = K+ + 3 Al+3 + 2 SO4-2 + 12H2O
        log_k            -1.400
        delta_h -50.250 kcal

Jarosite-K
        KFe3(SO4)2(OH)6 + 6 H3O+ = 3 Fe+3 + 12 H2O + K+ + 2 SO4-2
        log_k            -9.210
        delta_h -31.280 kcal

EXCHANGE_MASTER_SPECIES
        X       X-
EXCHANGE_SPECIES
        X- = X-
        log_k            0.0

        Na+ + X- = NaX
        log_k    0.0
        -gamma   4.0      0.075

        K+ + X- = KX
        log_k    0.7
        -gamma   3.5      0.015
        delta_h  -4.3     # Jardine & Sparks, 1984

        NH4+ + X- = NH4X
        log_k    0.6
        -gamma   2.5      0.0
        delta_h  -2.4     # Laudelout et al., 1968

        Ca+2 + 2X- = CaX2
        log_k    0.8
        -gamma   5.0      0.165
        delta_h  7.2      # Van Bladel & Gheyl, 1980

        Mg+2 + 2X- = MgX2
        log_k    0.6
        -gamma   5.5      0.2
        delta_h  7.4      # Laudelout et al., 1968

        Fe+2 + 2X- = FeX2
        log_k    0.44
        -gamma   6.0      0.0

        Al+3 + 3X- = AlX3
        log_k    0.41
        -gamma   9.0      0.0

        AlOH+2 + 2X- = AlOHX2
        log_k    0.89
        -gamma   0.0      0.0

SURFACE_MASTER_SPECIES
        Hfo_s   Hfo_sOH
        Hfo_w   Hfo_wOH

SURFACE_SPECIES
#    All surface data from
#    Dzombak and Morel, 1990
#
#
#    Acid-base data from table 5.7
#
#    strong binding site--Hfo_s,

        Hfo_sOH = Hfo_sOH
        log_k   0.0

        Hfo_sOH  + H3O+ = Hfo_sOH2+ + H2O
        log_k   7.29     # = pKa1,int

        Hfo_sOH + H2O = Hfo_sO- + H3O+
        log_k   -8.93    # = -pKa2,int

#    weak binding site--Hfo_w

        Hfo_wOH = Hfo_wOH
        log_k   0.0

        Hfo_wOH  + H3O+ = Hfo_wOH2+ + H2O
        log_k   7.29     # = pKa1,int

        Hfo_wOH + H2O = Hfo_wO- + H3O+
        log_k   -8.93    # = -pKa2,int

#####################################################
#                   CATIONS                         #
#####################################################
#
#    Cations from table 10.1 or 10.5
#
#    Calcium
        Hfo_sOH + Ca+2 = Hfo_sOHCa+2
        log_k   4.97

        Hfo_wOH + Ca+2 + H2O = Hfo_wOCa+ + H3O+
        log_k  -5.85
#
#    Derived constants table 10.5
#
#    Magnesium
        Hfo_wOH + Mg+2 + H2O = Hfo_wOMg+ + H3O+
        log_k  -4.6
#  Iron
#         Hfo_sOH + Fe+2 + H2O = Hfo_sOFe+ + H3O+
#         log_k    0.7      # LFER using table 10.5
```

```
#         Hfo_wOH + Fe+2 + H2O = Hfo_wOFe+ + H3O+
#         log_k  -2.5          # LFER using table 10.5

# Iron, strong site: Appelo, Van der Weiden, Tournassat & Charlet, subm.
          Hfo_sOH + Fe+2 + H2O = Hfo_sOFe+ + H3O+
          log_k    -0.95
# Iron, weak site: Liger et al., GCA 63, 2939, re-optimized for D&M
          Hfo_wOH + Fe+2 + H2O = Hfo_wOFe+ + H3O+
          log_k  -2.98

          Hfo_wOH + Fe+2 + 3H2O = Hfo_wOFeOH + 2H3O+
          log_k  -11.55

###############################################
#               ANIONS                        #
###############################################
#
#    Anions from table 10.6
#
#    Phosphate
          Hfo_wOH + PO4-3 + 3 H3O+ = Hfo_wH2PO4 + 4H2O
          log_k    31.29

          Hfo_wOH + PO4-3 + 2 H3O+ = Hfo_wHPO4- + 3H2O
          log_k    25.39

          Hfo_wOH + PO4-3 + H3O+ = Hfo_wPO4-2 + 2H2O
          log_k    17.72
#
#    Anions from table 10.7
#
#    Sulfate
          Hfo_wOH + SO4-2 + H3O+ = Hfo_wSO4- + 2H2O
          log_k    7.78

          Hfo_wOH + SO4-2 = Hfo_wOHSO4-2
          log_k    0.79
#
#    Derived constants table 10.10
#
          Hfo_wOH + F- + H3O+ = Hfo_wF + 2H2O
          log_k    8.7

          Hfo_wOH + F- = Hfo_wOHF-
          log_k    1.6
#
# Carbonate: Van Geen et al., 1994 reoptimized for HFO
# 0.15 g HFO/L has 0.344 mM sites == 2 g of Van Geen's Goethite/L
#
#         Hfo_wOH + CO3-2 + H3O+ = Hfo_wCO3- + 2H2O
#         log_k    12.56
#
#         Hfo_wOH + CO3-2 + 2 H3O+= Hfo_wHCO3 + 3 H2O
#         log_k    20.62

# 9/19/96
#         Added analytical expression for H2S, NH3, KSO4.
#         Added species CaHSO4+.
#         Added delta H for Goethite.

#
#  Begin Isotope definitions
#
#↩
     ##########################################################################################################↩
SOLUTION_MASTER_SPECIES
#↩
     ##########################################################################################################↩

D               D2O             0       D                   2
D(1)            D2O             0       D
D(0)            HD              0       D
T               HTO             0       T                   3
T(1)            HTO             0       T
T(0)            HT              0       T
[18O]           H2[18O]         0       [18O]               18
[18O](-2)       H2[18O]         0       [18O]               18
[18O](0)        O[18O]          0       [18O]               18
[13C]           [13C]O2         0       [13C]               13
[13C](4)        [13C]O2         0       [13C]
[13C](-4)       [13C]H4         0       [13C]
[14C]           [14C]O2         0       [14C]               14
[14C](4)        [14C]O2         0       [14C]
[14C](-4)       [14C]H4         0       [14C]
[34S]           [34S]O4-2       0       [34S]               33.967
[34S](6)        [34S]O4-2       0       [34S]
[34S](-2)       H[34S]-         0       [34S]
[15N]           [15N]O3-        0       [15N]               15.000108973l2
[15N](5)        [15N]O3-        0       [15N]
[15N](3)        [15N]O2-        0       [15N]
[15N](0)        N[15N]          0       [15N]
[15N](-3)       [15N]H4+        0       [15N]
#↩
     ##########################################################################################################↩

ISOTOPES
#↩
     ##########################################################################################################↩

H
     -isotope         D       permil  155.76e-6               # VSMOW (Clark and Fritz, 1997)
     -isotope         T       TU      1e-18                   # Solomon and Cook, in eds, Cook↩
          and Herczeg, 2000
H(0)
     -isotope         D(0)    permil  155.76e-6               # VSMOW (Clark and Fritz, 1997)
     -isotope         T(0)    TU      1e-18                   # Solomon and Cook, in eds, Cook↩
          and Herczeg, 2000
                                                             # 1 THO in 10^18 H2O
#     -isotope         T       pCi/L   3.125e-18               #
#1e-18/3.2 = T/mol H2O
C
     -isotope         [13C]   permil  0.0111802               # VPDB, Vienna Pee Dee Belemnite
                                                             # Chang and Li, 1990, Chinese ↩
                                                                Science Bulletin
     -isotope         [13C](4)        permil  0.0111802       # VPDB, Vienna Pee Dee Belemnite
                                                             # Chang and Li, 1990, Chinese ↩
                                                                Science Bulletin
     -isotope         [13C](-4)       permil  0.0111802       # VPDB, Vienna Pee Dee Belemnite
     -isotope         [14C]           pmc     1.175887709e-12 # Mole fraction of 14C in Modern↩
          Carbon
```

```
      -isotope          [14C](4)          pmc      1.175887709e-12 # Mole fraction of 14C in Modern↩
            Carbon
                                                                     # 13.56 Modern Carbon dpm (Kalin↩
                                                                         , in eds, Cook and Herczeg, ↩
                                                                         2000)
      -isotope          [14C](-4)         pmc      1.175887709e-12 # Mole fraction of 14C in Modern↩
            Carbon
C(4)
      -isotope          [13C](4)          permil   0.0111802       # VPDB, Vienna Pee Dee Belemnite
                                                                     # Chang and Li, 1990, Chinese ↩
                                                                         Science Bulletin
      -isotope          [14C](4)          pmc      1.175887709e-12 # Mole fraction of 14C in Modern↩
            Carbon
                                                                     # 13.56 Modern Carbon dpm (Kalin↩
                                                                         , in eds, Cook and Herczeg, ↩
                                                                         2000)
                                                                     # 13.56 Modern Carbon dpm (Kalin↩
                                                                         , in eds, Cook and Herczeg, ↩
                                                                         2000)
C(-4)
      -isotope          [13C](-4)         permil   0.0111802       # VPDB, Vienna Pee Dee Belemnite
      -isotope          [14C](-4)         pmc      1.175887709e-12 # Mole fraction of 14C in Modern↩
            Carbon
      # 14C calculation
      #
      # lambda = ln(2)/(5730 yrs * 3.15576e7 sec/yr)
      # mole/g carbon = -(dn/dt)/lambda = 0.226 dps / 3.8332476e-12 / 6.022136736e23
      # mole C/g C NBS Oxalic Acid with 13C = -19.3:  0.08325783313
      # mole 14C/mol Modern Carbon mol/g carbon/ (mole C/g C) = 1.175887709e-12
      #
O
      -isotope          [18O]             permil   2005.2e-6       # VSMOW (Clark and Fritz, 1997)
      -isotope          [18O](0)          permil   2005.2e-6       # VSMOW (Clark and Fritz, 1997)
O(0)
      -isotope          [18O](0)          permil   2005.2e-6       # VSMOW (Clark and Fritz, 1997)

S
                                                                     # Coplen and others, 2002
      -isotope          [34S]             permil   0.04416264      # VCDT, Vienna Canyon Diablo ↩
            Troilite
      -isotope          [34S](6)          permil   0.04416264      # VCDT
      -isotope          [34S](-2)         permil   0.04416264      # VCDT
S(6)
      -isotope          [34S](6)          permil   0.04416264      # VCDT
S(-2)
      -isotope          [34S](-2)         permil   0.04416264      # VCDT
N
                                                                     # Coplen and others, 2002
      -isotope          [15N]             permil   0.003676867     # Air
      -isotope          [15N](5)          permil   0.003676867     # Air
      -isotope          [15N](3)          permil   0.003676867     # Air
      -isotope          [15N](0)          permil   0.003676867     # Air
      -isotope          [15N](-3)         permil   0.003676867     # Air
N(5)
      -isotope          [15N](5)          permil   0.003676867     # Air
N(3)
      -isotope          [15N](3)          permil   0.003676867     # Air
N(0)
      -isotope          [15N](0)          permil   0.003676867     # Air
N(-3)
      -isotope          [15N](-3)         permil   0.003676867     # Air
#↩
    ##################################################################################################↩

ISOTOPE_RATIOS
#↩
    ##################################################################################################↩

#
#   Total aqueous ratios
#
        R(D)                    D
        R(T)                    T
        R(18O)                  [18O]
        R(13C)                  [13C]
        R(14C)                  [14C]
        R(15N)                  [15N]
        R(34S)                  [34S]
# H2O(l) ratios
        R(D)_H2O(l)             D
        R(T)_H2O(l)             T
        R(18O)_H2O(l)           [18O]
# OH- ratios
        R(D)_OH-                D
        R(T)_OH-                T
        R(18O)_OH-              [18O]
# H3O+ ratios
        R(D)_H3O+               D
        R(T)_H3O+               T
        R(18O)_H3O+             [18O]
# O2(aq) ratios
        R(18O)_O2(aq)           [18O]
# H2(aq) ratios
        R(D)_H2(aq)             D
        R(T)_H2(aq)             T
# CO2(aq) ratios
        R(13C)_CO2(aq)          [13C]
        R(14C)_CO2(aq)          [14C]
        R(18O)_CO2(aq)          [18O]
# HCO3- ratios
        R(D)_HCO3-              D
        R(T)_HCO3-              T
        R(18O)_HCO3-            [18O]
        R(13C)_HCO3-            [13C]
        R(14C)_HCO3-            [14C]
# CO3-2 ratios
        R(18O)_CO3-2            [18O]
        R(13C)_CO3-2            [13C]
        R(14C)_CO3-2            [14C]
# CH4(aq) ratios
        R(D)_CH4(aq)            D
        R(T)_CH4(aq)            T
        R(13C)_CH4(aq)          [13C]
        R(14C)_CH4(aq)          [14C]
# SO4-2 ratios
        R(34S)_SO4-2            [34S]
# HSO4- ratios
        R(D)_HSO4-              D
```

```
        R(T)_HSO4-              T
        R(34S)_HSO4-            [34S]
# S-2 ratios
        R(34S)_S-2              [34S]
# HS- ratios
        R(D)_HS-                D
        R(T)_HS-                T
        R(34S)_HS-              [34S]
# H2S ratios
        R(D)_H2S(aq)            D
        R(T)_H2S(aq)            T
        R(34S)_H2S(aq)          [34S]
# NO3- ratios
        R(15N)_NO3-             [15N]
# NO2- ratios
        R(15N)_NO2-             [15N]
# N2(aq) ratios
        R(15N)_N2(aq)           [15N]
# NH3(aq) ratios
        R(D)_NH3(aq)            D
        R(T)_NH3(aq)            T
        R(15N)_NH3(aq)          [15N]
# NH4+ ratios
        R(D)_NH4+               D
        R(T)_NH4+               T
        R(15N)_NH4+             [15N]
#
#  Ratios for minerals and gases
#
# H2O(g) ratios
        R(D)_H2O(g)             D
        R(T)_H2O(g)             T
        R(18O)_H2O(g)           [18O]
# O2(g) ratios
        R(18O)_O2(g)            [18O]
# H2(g) ratios
        R(D)_H2(g)              D
        R(T)_H2(g)              T
# CO2(g) ratios
        R(18O)_CO2(g)           [18O]
        R(13C)_CO2(g)           [13C]
        R(14C)_CO2(g)           [14C]
# Calcite ratios
        R(18O)_Calcite          [18O]
        R(13C)_Calcite          [13C]
        R(14C)_Calcite          [14C]
# Pyrite ratios
        R(34S)_Pyrite           [34S]
# CH4(g) ratios
        R(D)_CH4(g)             D
        R(T)_CH4(g)             T
        R(13C)_CH4(g)           [13C]
        R(14C)_CH4(g)           [14C]
# H2S(g) ratios
        R(D)_H2S(g)             D
        R(T)_H2S(g)             T
        R(34S)_H2S(g)           [34S]
# Gypsum ratios
        R(34S)_Gypsum           [34S]
# Anhydrite ratios
        R(34S)_Anhydrite        [34S]
# N2(g) ratios
        R(15N)_N2(g)            [15N]
# NH3(g) ratios
        R(D)_NH3(g)             D
        R(T)_NH3(g)             T
        R(15N)_NH3(g)           [15N]

ISOTOPE_ALPHAS
# OH-
        Alpha_D_OH-/H2O(l)                  Log_alpha_D_OH-/H2O(l)
        Alpha_T_OH-/H2O(l)                  Log_alpha_T_OH-/H2O(l)
        Alpha_18O_OH-/H2O(l)                Log_alpha_18O_OH-/H2O(l)
# H3O+
        Alpha_D_H3O+/H2O(l)                 Log_alpha_D_H3O+/H2O(l)
        Alpha_T_H3O+/H2O(l)                 Log_alpha_T_H3O+/H2O(l)
        Alpha_18O_H3O+/H2O(l)               Log_alpha_18O_H3O+/H2O(l)
# O2(aq)
        Alpha_18O_O2(aq)/H2O(l)             Log_alpha_18O_O2(aq)/H2O(l)
# H2(aq)
        Alpha_D_H2(aq)/H2O(l)               Log_alpha_D_H2(aq)/H2O(l)
        Alpha_T_H2(aq)/H2O(l)               Log_alpha_T_H2(aq)/H2O(l)
# CO2(aq)
        Alpha_18O_CO2(aq)/H2O(l)            Log_alpha_18O_CO2(aq)/H2O(l)
        Alpha_13C_CO2(aq)/CO2(g)            Log_alpha_13C_CO2(aq)/CO2(g)
        Alpha_14C_CO2(aq)/CO2(g)            Log_alpha_14C_CO2(aq)/CO2(g)
# HCO3-
        Alpha_D_HCO3-/H2O(l)                Log_alpha_D_HCO3-/H2O(l)
        Alpha_T_HCO3-/H2O(l)                Log_alpha_T_HCO3-/H2O(l)
        Alpha_18O_HCO3-/H2O(l)              Log_alpha_18O_HCO3-/H2O(l)
        Alpha_13C_HCO3-/CO2(aq)             Log_alpha_13C_HCO3-/CO2(aq)
        Alpha_14C_HCO3-/CO2(aq)             Log_alpha_14C_HCO3-/CO2(aq)
        Alpha_13C_HCO3-/CO2(g)              Log_alpha_13C_HCO3-/CO2(g)
        Alpha_14C_HCO3-/CO2(g)              Log_alpha_14C_HCO3-/CO2(g)
# CO3-2
        Alpha_18O_CO3-2/H2O(l)              Log_alpha_18O_CO3-2/H2O(l)
        Alpha_13C_CO3-2/CO2(aq)             Log_alpha_13C_CO3-2/CO2(aq)
        Alpha_14C_CO3-2/CO2(aq)             Log_alpha_14C_CO3-2/CO2(aq)
        Alpha_13C_CO3-2/CO2(g)              Log_alpha_13C_CO3-2/CO2(g)
        Alpha_14C_CO3-2/CO2(g)              Log_alpha_14C_CO3-2/CO2(g)
# CH4(aq)
        Alpha_D_CH4(aq)/H2O(l)              Log_alpha_D_CH4(aq)/H2O(l)
        Alpha_T_CH4(aq)/H2O(l)              Log_alpha_T_CH4(aq)/H2O(l)
        Alpha_13C_CH4(aq)/CO2(aq)           Log_alpha_13C_CH4(aq)/CO2(aq)
        Alpha_14C_CH4(aq)/CO2(aq)           Log_alpha_14C_CH4(aq)/CO2(aq)
# HSO4-
        Alpha_D_HSO4-/H2O(l)                Log_alpha_D_HSO4-/H2O(l)
        Alpha_T_HSO4-/H2O(l)                Log_alpha_T_HSO4-/H2O(l)
        Alpha_34S_HSO4-/SO4-2               Log_alpha_34S_HSO4-/SO4-2
# S-2
        Alpha_34S_S-2/HS-                   Log_alpha_34S_S-2/HS-
# HS-
        Alpha_D_HS-/H2O(l)                  Log_alpha_D_HS-/H2O(l)
        Alpha_T_HS-/H2O(l)                  Log_alpha_T_HS-/H2O(l)
        Alpha_34S_HS-/SO4-2                 Log_alpha_34S_HS-/SO4-2
# H2S
        Alpha_D_H2S(aq)/H2O(l)              Log_alpha_D_H2S(aq)/H2O(l)
```

```
        Alpha_T_H2S(aq)/H2O(l)                  Log_alpha_T_H2S(aq)/H2O(l)
        Alpha_34S_H2S(aq)/HS-                   Log_alpha_34S_H2S(aq)/HS-
# NO2-
        Alpha_15N_NO2-/NO3-                     Log_alpha_15N_NO2-/NO3-
# N2(aq)
        Alpha_15N_N2(aq)/NO3-                   Log_alpha_15N_N2(aq)/NO3-
# NH3(aq)
    Alpha_D_NH3(aq)/H2O(l)                  Log_alpha_D_NH3(aq)/H2O(l)
    Alpha_T_NH3(aq)/H2O(l)                  Log_alpha_T_NH3(aq)/H2O(l)
        Alpha_15N_NH3(aq)/NO3-                  Log_alpha_15N_NH3(aq)/NO3-
# NH4+
        Alpha_D_NH4+/H2O(l)                     Log_alpha_D_NH4+/H2O(l)
        Alpha_T_NH4+/H2O(l)                     Log_alpha_T_NH4+/H2O(l)
        Alpha_15N_NH4+/NH3(aq)                  Log_alpha_15N_NH4+/NH3(aq)
# H2O(g)
        Alpha_D_H2O(g)/H2O(l)                   Log_alpha_D_H2O(g)/H2O(l)
        Alpha_T_H2O(g)/H2O(l)                   Log_alpha_T_H2O(g)/H2O(l)
        Alpha_18O_H2O(g)/H2O(l)                 Log_alpha_18O_H2O(g)/H2O(l)
# O2(g)
        Alpha_18O_O2(g)/H2O(l)                  Log_alpha_18O_O2(g)/H2O(l) #?
# H2(g)
        Alpha_D_H2(g)/H2O(l)                    Log_alpha_D_H2(g)/H2O(l)   #?
        Alpha_T_H2(g)/H2O(l)                    Log_alpha_T_H2(g)/H2O(l)   #?
# CO2(g)
        Alpha_18O_CO2(g)/H2O(l)                 Log_alpha_18O_CO2(g)/H2O(l)
        Alpha_13C_CO2(g)/CO2(aq)                Log_alpha_13C_CO2(g)/CO2(aq)
        Alpha_14C_CO2(g)/CO2(aq)                Log_alpha_14C_CO2(g)/CO2(aq)
#  Calcite
        Alpha_18O_Calcite/H2O(l)                Log_alpha_18O_Calcite/H2O(l)
        Alpha_13C_Calcite/CO2(aq)               Log_alpha_13C_Calcite/CO2(aq)
        Alpha_13C_Calcite/CO2(g)                Log_alpha_13C_Calcite/CO2(g)
        Alpha_14C_Calcite/CO2(aq)               Log_alpha_14C_Calcite/CO2(aq)
        Alpha_14C_Calcite/CO2(g)                Log_alpha_14C_Calcite/CO2(g)
#  Pyrite
        Alpha_34S_Pyrite/HS-                    Log_alpha_34S_Pyrite/HS-
# CH4(g)
        Alpha_D_CH4(g)/H2O(l)                   Log_alpha_D_CH4(g)/H2O(l)
        Alpha_T_CH4(g)/H2O(l)                   Log_alpha_T_CH4(g)/H2O(l)
        Alpha_13C_CH4(g)/CO2(aq)                Log_alpha_13C_CH4(g)/CO2(aq) #?
        Alpha_14C_CH4(g)/CO2(aq)                Log_alpha_14C_CH4(g)/CO2(aq) #?
# H2S(g)
        Alpha_D_H2S(g)/H2S(aq)                  Log_alpha_D_H2S(g)/H2S(aq)
        Alpha_T_H2S(g)/H2S(aq)                  Log_alpha_T_H2S(g)/H2S(aq)
        Alpha_34S_H2S(g)/H2S(aq)                Log_alpha_34S_H2S(g)/H2S(aq)
# Gypsum
        Alpha_34S_Gypsum/SO4-2                  Log_alpha_34S_Gypsum/SO4-2
# Anhydrite
        Alpha_34S_Anhydrite/SO4-2               Log_alpha_34S_Anhydrite/SO4-2
# N2(g)
        Alpha_15N_N2(g)/N2(aq)                  Log_alpha_15N_N2(g)/N2(aq)
# NH3(g)
        Alpha_D_NH3(g)/H2O(l)                   Log_alpha_D_NH3(g)/H2O(l)
        #Alpha_T_NH3(g)/H2O(l)                  Log_alpha_T_NH3(g)/H2O(l)
        Alpha_15N_NH3(g)/NH3(aq)                Log_alpha_15N_NH3(g)/NH3(aq)

NAMED_EXPRESSIONS
#
# OH- fractionation factors
#
Log_alpha_D_OH-/H2O(l)                  # 1000ln(alpha(25C)) = -1435
    # 13.5 C
    -ln_alpha1000    -1435.0

Log_alpha_T_OH-/H2O(l)                  # 1000ln(alpha(25C)) = -2870
    # 13.5 C
    -ln_alpha1000    -2870.0

Log_alpha_18O_OH-/H2O(l)                # 1000ln(alpha(25C)) = -37.8
    # 25 C
    -ln_alpha1000    -37.777
#
# H3O+ fractionation factors
#
Log_alpha_D_H3O+/H2O(l)                 # 1000ln(alpha(25C)) = 40.8
    # 13.5 C
    -ln_alpha1000    40.82

Log_alpha_T_H3O+/H2O(l)                 # 1000ln(alpha(25C)) = 81.6
    # 13.5 C
    -ln_alpha1000    81.64

Log_alpha_18O_H3O+/H2O(l)               # 1000ln(alpha(25C)) = 22.9
    # 25 C
    -ln_alpha1000    22.86
#
# O2(aq) fractionation factors
#
Log_alpha_18O_O2(aq)/H2O(l)
    -ln_alpha1000    0.0
#
# H2(aq) fractionation factors
#
Log_alpha_D_H2(aq)/H2O(l)
    -ln_alpha1000    0

Log_alpha_T_H2(aq)/H2O(l)
    -ln_alpha1000    0
#
# CO2(aq) fractionation factors
# Updated 01/06/2016 (R. Owen)
#
Log_alpha_18O_CO2(aq)/H2O(l)            # 1000ln(alpha(25C)) = 40.5
    # Beck et al. (2005)
    -ln_alpha1000    12.12  0   0   0   2.52e6

Log_alpha_13C_CO2(aq)/CO2(g)              # 1000ln(alpha(25C))  -1.43
    # Zheng et al. (1995)
    -ln_alpha1000    0.02952  -0.00491

Log_alpha_14C_CO2(aq)/CO2(g)
    -add_logk        Log_alpha_13C_CO2(aq)/CO2(g)    2


#
# HCO3- fractionation factors
# Updated 01/06/2016 (R. Owen)
#
Log_alpha_D_HCO3-/H2O(l)
    -ln_alpha1000    0.0
```

```
|
| Log_alpha_T_HCO3-/H2O(l)
|     -ln_alpha1000    0.0
|
| Log_alpha_18O_HCO3-/H2O(l)               # 1000ln(alpha(25C)) = 31.0
|     # Beck et al. (2005)
|     -ln_alpha1000    1.89    0    0    0    2.59e6
|
| Log_alpha_13C_HCO3-/CO2(g)               # 1000ln(alpha(25C)) = 7.24
|     # Zheng et al. (1995)
|     -ln_alpha1000    48.98   -0.14
|
| Log_alpha_13C_HCO3-/CO2(aq)              # 1000ln(alpha(25C)) = 8.67
|     -add_logk        Log_alpha_13C_HCO3-/CO2(g)       1
|     -add_logk        Log_alpha_13C_CO2(aq)/CO2(g)    -1
|
| Log_alpha_14C_HCO3-/CO2(g)
|     -add_logk         Log_alpha_13C_HCO3-/CO2(g)      2
|
| Log_alpha_14C_HCO3-/CO2(aq)
|     -add_logk        Log_alpha_14C_HCO3-/CO2(g)       1
|     -add_logk        Log_alpha_14C_CO2(aq)/CO2(g)    -1
| #
| # CO3-2 fractionation factors
| # Updated 01/06/2016 (R. Owen)
| #
| Log_alpha_18O_CO3-2/H2O(l)          # 1000ln(alpha(25C)) = 24.2
|     # Beck et al. (2005)
|     -ln_alpha1000    -2.7    0    0    0    2.39e6
|
| Log_alpha_13C_CO3-2/CO2(g)               # 1000ln(alpha(25C)) = 5.91
|     # Zheng et al. (1995)
|     -ln_alpha1000    21.32   -0.0517
|
| Log_alpha_13C_CO3-2/CO2(aq)
|     -add_logk        Log_alpha_13C_CO3-2/CO2(g)       1
|     -add_logk        Log_alpha_13C_CO2(aq)/CO2(g)    -1
|
| Log_alpha_14C_CO3-2/CO2(g)
|     -add_logk         Log_alpha_13C_CO3-2/CO2(g)      2
|
| Log_alpha_14C_CO3-2/CO2(aq)
|     -add_logk        Log_alpha_14C_CO3-2/CO2(g)       1
|     -add_logk        Log_alpha_14C_CO2(aq)/CO2(g)    -1
| #
| # CH4(aq) fractionation factors
| #
| Log_alpha_D_CH4(aq)/H2O(l)
|     -ln_alpha1000    0
|
| Log_alpha_T_CH4(aq)/H2O(l)
|     -ln_alpha1000    0
|
| Log_alpha_13C_CH4(aq)/CO2(aq)
|     -ln_alpha1000    0
|
| Log_alpha_14C_CH4(aq)/CO2(aq)
|     -ln_alpha1000    0
| #
| # HSO4- fractionation factors
| #
| Log_alpha_D_HSO4-/H2O(l)
|     -ln_alpha1000    0
|
| Log_alpha_T_HSO4-/H2O(l)
|     -ln_alpha1000    0
|
| Log_alpha_34S_HSO4-/SO4-2
|     -ln_alpha1000    0.0
| #
| # S-2 fractionation factors
| #
| Log_alpha_34S_S-2/HS-
|     -ln_alpha1000    0.0
| #
| # HS- fractionation factors
| #
| Log_alpha_D_HS-/H2O(l)
|     -ln_alpha1000    0
|
| Log_alpha_T_HS-/H2O(l)
|     -ln_alpha1000    0
|
| Log_alpha_34S_HS-/SO4-2
|     -ln_alpha1000    0.0
| #
| # H2S fractionation factors
| #
| Log_alpha_D_H2S(aq)/H2O(l)
|     -ln_alpha1000    0
|
| Log_alpha_T_H2S(aq)/H2O(l)
|     -ln_alpha1000    0
|
| Log_alpha_34S_H2S(aq)/HS-
|     -ln_alpha1000    0.0
| #
| # NO2- fractionation factors
| #
| Log_alpha_15N_NO2-/NO3-
|     -ln_alpha1000    0.0
| #
| # N2(aq) fractionation factors
| #
| Log_alpha_15N_N2(aq)/NO3-
|     -ln_alpha1000    0.0
| #
| # NH3(aq) fractionation factors
| #
| Log_alpha_D_NH3(aq)/H2O(l)
|     -ln_alpha1000    0
|
| Log_alpha_T_NH3(aq)/H2O(l)
|     -ln_alpha1000    0
|
| Log_alpha_15N_NH3(aq)/NO3-
|     -ln_alpha1000    0
| #
| # NH4+ fractionation factors
| #
| Log_alpha_D_NH4+/H2O(l)
|
```

```
    -ln_alpha1000   0

Log_alpha_T_NH4+/H2O(l)
    -ln_alpha1000   0

Log_alpha_15N_NH4+/NH3(aq)
    -ln_alpha1000   0
#
# H2O(g) fractionation factors
#
Log_alpha_D_H2O(g)/H2O(l)               # 1000ln(alpha(25C)) = -76.4
    # 0-100 C
    -ln_alpha1000   -52.612  0.0     76.248e3       0.0      -24.844e6

Log_alpha_T_H2O(g)/H2O(l)               # 1000ln(alpha(25C)) = -152.7
    # 0-100 C
    -ln_alpha1000   -105.224 0.0     152.496e3      0.0      -49.688e6

Log_alpha_18O_H2O(g)/H2O(l)             # 1000ln(alpha(25C)) = -9.3
    # 0-100 C
    -ln_alpha1000   2.0667   0.0     0.4156e3       0.0      -1.137e6
#
# O2(g) fractionaton factors
#
Log_alpha_18O_O2(g)/H2O(l)
    -ln_alpha1000   0.0

Log_alpha_18O_O2(g)/O2(aq)
    -add_logk       Log_alpha_18O_O2(g)/H2O(l)      1
    -add_logk       Log_alpha_18O_O2(aq)/H2O(l)    -1



#
# H2(g) fractionaton factors
#
Log_alpha_D_H2(g)/H2O(l)
    -ln_alpha1000   0

Log_alpha_T_H2(g)/H2O(l)
    -ln_alpha1000   0
#
# CO2(g) fractionaton factors
# Updated 01/06/2016 (R. Owen)
#
Log_alpha_18O_CO2(g)/H2O(l)             # 1000ln(alpha(25C)) = 40.0
    # O'Neil and Adami (1969)
    -ln_alpha1000   -15.69  0   1.66e4

Log_alpha_18O_CO2(g)/CO2(aq)
    -add_logk       Log_alpha_18O_CO2(g)/H2O(l)      1
    -add_logk       Log_alpha_18O_CO2(aq)/H2O(l)    -1

Log_alpha_13C_CO2(g)/CO2(aq)
    -add_logk       Log_alpha_13C_CO2(aq)/CO2(g)    -1

Log_alpha_14C_CO2(g)/CO2(aq)
    -add_logk       Log_alpha_13C_CO2(aq)/CO2(g)    -2

#
# CO2-Calcite fractionation factors
# Updated 01/06/2016 (R. Owen)
#
Log_alpha_18O_Calcite/H2O(l)            # 1000ln(alpha(25C)) = 29.76
    # Coplen et al. (2007)
    -ln_alpha1000    -28.6     0   17.4e3

Log_alpha_18O_Calcite/CO2(aq)
    -add_logk    Log_alpha_18O_Calcite/H2O(l)   1
    -add_logk    Log_alpha_18O_CO2(aq)/H2O(l)   -1

Log_alpha_13C_Calcite/CO2(g)            # 1000ln(alpha(25C)) = 8.95
    # Romanek et al. (1992)
    -ln_alpha1000   45.65    -0.121   -186.9

Log_alpha_13C_Calcite/CO2(aq)           # 1000ln(alpha(25C)) = 10.38
    -add_logk       Log_alpha_13C_Calcite/CO2(g)     1
    -add_logk       Log_alpha_13C_CO2(aq)/CO2(g)    -1

Log_alpha_14C_Calcite/CO2(g)            # 1000ln(alpha(25C)) = 17.89
    -add_logk       Log_alpha_13C_Calcite/CO2(g)     2

Log_alpha_14C_Calcite/CO2(aq)           # 1000ln(alpha(25C)) = 20.76
    -add_logk       Log_alpha_14C_Calcite/CO2(g)     1
    -add_logk       Log_alpha_14C_CO2(aq)/CO2(g)    -1
#
# Pyrite fractionation factors
#
Log_alpha_34S_Pyrite/HS-
    -ln_alpha1000   0
#
# CH4(g) fractionation factors
#
Log_alpha_D_CH4(g)/H2O(l) #?
    -ln_alpha1000   0.0

Log_alpha_T_CH4(g)/H2O(l) #?
    -ln_alpha1000   0.0

Log_alpha_13C_CH4(g)/CO2(aq) #?
    -ln_alpha1000   0.0

Log_alpha_14C_CH4(g)/CO2(aq) #?
    -ln_alpha1000   0.0
#
# H2S(g) fractionation factors
#
Log_alpha_D_H2S(g)/H2S(aq)
    -ln_alpha1000   0

Log_alpha_T_H2S(g)/H2S(aq)
    -ln_alpha1000   0

Log_alpha_34S_H2S(g)/H2S(aq)
    -ln_alpha1000   0
#
# Gypsum fractionation factors
#
Log_alpha_34S_Gypsum/SO4-2
    -ln_alpha1000   0
#
```

```
# Anhydrite fractionation factors
#
Log_alpha_34S_Anhydrite/SO4-2
     -ln_alpha1000   0
#
# N2(g) fractionation factors
#
Log_alpha_15N_N2(g)/N2(aq)
     -ln_alpha1000   0
#
# NH3(g) fractionation factors
#
Log_alpha_D_NH3(g)/H2O(l) #?
     -ln_alpha1000   0

Log_alpha_T_NH3(g)/H2O(l) #?
     -ln_alpha1000   0

Log_alpha_15N_NH3(g)/NH3(aq)
     -ln_alpha1000   0

#↩
     ##########################################################################################################↩

CALCULATE_VALUES
#↩
     ##########################################################################################################↩

#
#  Ratios
#
R(D)
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 100
30 total_D = TOT("D")
40 total_H = TOT("H")
50 ratio = total_D/total_H
100 save ratio
     -end

R(T)  ??
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 100
30 total_T = TOT("T")
40 total_water = MOL("H2O")
50 ratio = total_T/total_water
100 save ratio
     -end

R(18O)
     -start
10 ratio = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 100
30 total_18O = TOT("[18O]")
40 total_16O = TOT("O")
50 ratio = total_18O/total_16O
100 save ratio
     -end

R(13C)
     -start
10 ratio = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 100
30 total_13C = TOT("[13C]")
40 total_12C = TOT("C")
50 ratio = total_13C/total_12C
100 save ratio
     -end

R(14C)
     -start
10 ratio = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 100
30 total_14C = TOT("[14C]")
40 total_C = TOT("C") + TOT("[13C]") + TOT("[14C]")
50 ratio = total_14C/total_C
100 save ratio
     -end

R(15N)
     -start
10 ratio = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 100
30 total_15N = TOT("[15N]")
40 total_14N = TOT("N")
50 ratio = total_15N/total_14N
100 save ratio
     -end

R(34S)
     -start
10 ratio = -9999.999
20 if (TOT("[34S]") <= 0) THEN GOTO 100
30 total_34S = TOT("[34S]")
40 total_32S = TOT("S")
50 ratio = total_34S/total_32S
100 save ratio
     -end
#
# H2O(l) ratios
#
R(D)_H2O(l)
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 100
30 total_D = sum_species("{H,D,T}2{O,[18O]}","D")
40 total_H = sum_species("{H,D,T}2{O,[18O]}","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_D/total_H
100 save ratio
     -end

R(T)_H2O(l)
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 100
30 total_T = sum_species("{H,D,T}2{O,[18O]}","T")
40 total_H = sum_species("{H,D,T}2{O,[18O]}","H")
50 if (total_H <= 0) THEN GOTO 100
```

```
60 ratio = total_T/total_H
100 save ratio
     -end

R(18O)_H2O(l)
     -start
10 ratio = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 100
30 total_18O = sum_species("{H,D,T}2[18O]","[18O]")
40 total_16O = sum_species("{H,D,T}2O","O")
50 if (total_16O <= 0) THEN GOTO 100
60 ratio = total_18O/total_16O
100 save ratio
     -end
#
# OH- ratios
#
R(D)_OH-
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 100
30 total_D = sum_species("*{O,[18O]}D*","D")
40 total_H = sum_species("*{O,[18O]}H*","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_D/total_H
100 save ratio
     -end

R(T)_OH-
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 100
30 total_T = sum_species("*{O,[18O]}T*","T")
40 total_H = sum_species("*{O,[18O]}H*","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_T/total_H
100 save ratio
     -end

R(18O)_OH-
     -start
10 ratio = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 100
30 total_18O = sum_species("*[18O]{H,D,T}*","[18O]")
40 total_16O = sum_species("*O{H,D,T}*","O")
50 ratio = total_18O/total_16O
100 save ratio
     -end

#
# H3O+ ratios
#
R(D)_H3O+
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 100
30 total_D = sum_species("{H,D,T}3{O,[18O]}+","D")
40 total_H = sum_species("{H,D,T}3{O,[18O]}+","H")
50 ratio = total_D/total_H
100 save ratio
     -end

R(T)_H3O+
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 100
30 total_T = sum_species("{H,D,T}3{O,[18O]}+","T")
40 total_H = sum_species("{H,D,T}3{O,[18O]}+","H")
50 ratio = total_T/total_H
100 save ratio
     -end

R(18O)_H3O+
     -start
10 ratio = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 100
30 total_18O = sum_species("{H,D,T}3{O,[18O]}+","[18O]")
40 total_16O = sum_species("{H,D,T}3{O,[18O]}+","O")
50 ratio = total_18O/total_16O
100 save ratio
     -end

#
# O2(aq) ratios
#
R(18O)_O2(aq)
     -start
10 ratio = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 100
30 total_18O = sum_species("{O,[18O]}2","[18O]")
40 total_O = sum_species("{O,[18O]}2","O")
50 if (total_O <= 0) THEN GOTO 100
60 ratio = total_18O/total_O
100 save ratio
     -end
#
# H2(aq) ratios
#
R(D)_H2(aq)
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 100
30 total_D = sum_species("{H,D,T}2","D")
40 total_H = sum_species("{H,D,T}2","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_D/total_H
100 save ratio
     -end

R(T)_H2(aq)
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 100
30 total_T = sum_species("{H,D,T}2","T")
40 total_H = sum_species("{H,D,T}2","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_T/total_H
100 save ratio
     -end
#
# CO2(aq) ratios
```

```
#
R(13C)_CO2(aq)
     -start
10 ratio = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 100
30 total_13C = sum_species("[13C]{O,[18O]}2","[13C]")
40 total_12C = sum_species("C{O,[18O]}2","C")
50 if (total_12C <= 0) THEN GOTO 100
60 ratio = total_13C/total_12C
100 save ratio
     -end

R(14C)_CO2(aq)
     -start
10 ratio = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 100
30 total_14C = sum_species("[14C]{O,[18O]}2","[14C]")
40 total_12C = sum_species("C{O,[18O]}2","C")
41 total_13C = sum_species("[13C]{O,[18O]}2","[13C]")
50 if (total_12C <= 0) THEN GOTO 100
60 ratio = total_14C/(total_12C+total_13C+total_14C)
100 save ratio
     -end

R(18O)_CO2(aq)
     -start
10 ratio = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 100
30 if (TOT("C") <= 0) THEN GOTO 100
40 total_18O = sum_species("{C,[13C],[14C]}{O,[18O]}2","[18O]")
50 total_16O = sum_species("{C,[13C],[14C]}{O,[18O]}2","O")
60 if (total_16O <= 0) THEN GOTO 100
70 ratio = total_18O/total_16O
100 save ratio
     -end
#
# HCO3- ratios
#
R(D)_HCO3-
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 100
30 if (TOT("C") <= 0) THEN GOTO 100
40 total_D = sum_species("*D{C,[13C],[14C]}{O,[18O]}3*","D")
50 total_H = sum_species("*H{C,[13C],[14C]}{O,[18O]}3*","H")
60 if (total_H <= 0) THEN GOTO 100
70 ratio = total_D/total_H
100 save ratio
     -end

R(T)_HCO3-
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 100
30 if (TOT("C") <= 0) THEN GOTO 100
40 total_T = sum_species("*T{C,[13C],[14C]}{O,[18O]}3*","T")
50 total_H = sum_species("*H{C,[13C],[14C]}{O,[18O]}3*","H")
60 if (total_H <= 0) THEN GOTO 100
70 ratio = total_T/total_H
100 save ratio
     -end

R(18O)_HCO3-
     -start
10 ratio = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 100
30 if (TOT("C") <= 0) THEN GOTO 100
40 total_18O = sum_species("*{H,D,T}{C,[13C],[14C]}{O,[18O]}3*","[18O]")
50 total_16O = sum_species("*{H,D,T}{C,[13C],[14C]}{O,[18O]}3*","O")
60 if (total_16O <= 0) THEN GOTO 100
70 ratio = total_18O/total_16O
100 save ratio
     -end

R(13C)_HCO3-
     -start
10 ratio = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 100
30 total_13C = sum_species("*{H,D,T}[13C]{O,[18O]}3*","[13C]")
40 total_12C = sum_species("*{H,D,T}C{O,[18O]}3*","C")
50 if (total_12C <= 0) THEN GOTO 100
60 ratio = total_13C/total_12C
100 save ratio
     -end

R(14C)_HCO3-
     -start
10 ratio = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 100
30 total_14C = sum_species("*{H,D,T}[14C]{O,[18O]}3*","[14C]")
40 total_12C = sum_species("*{H,D,T}C{O,[18O]}3*","C")
41 total_13C = sum_species("*{H,D,T}[13C]{O,[18O]}3*","[13C]")
50 if (total_12C <= 0) THEN GOTO 100
60 ratio = total_14C/(total_12C+total_13C+total_14C)
100 save ratio
     -end
#
# CO3-2 ratios
#
R(18O)_CO3-2
     -start
10 ratio = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 100
30 if (TOT("C") <= 0) THEN GOTO 100
40 total_18O = sum_species("*{C,[13C],[14C]}{O,[18O]}3*","[18O]") - sum_species("*{H,D,T}{C↩
     ,[13C],[14C]}{O,[18O]}3*","[18O]")
50 total_16O = sum_species("*{C,[13C],[14C]}{O,[18O]}3*","O") - sum_species("*{H,D,T}{C,[13C↩
     ],[14C]}{O,[18O]}3*","O")
60 if (total_16O <= 0) THEN GOTO 100
70 ratio = total_18O/total_16O
100 save ratio
     -end

R(13C)_CO3-2
     -start
10 ratio = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 100
30 total_13C = sum_species("*[13C]{O,[18O]}3*","[13C]") - sum_species("*{H,D,T}[13C]{O,[18O↩
     ]}3*","[13C]")
40 total_12C = sum_species("*C{O,[18O]}3*","C") - sum_species("*{H,D,T}C{O,[18O]}3*","C")
```

```
50 if (total_12C <= 0) THEN GOTO 100
60 ratio = total_13C/total_12C
100 save ratio
     -end

R(14C)_CO3-2
     -start
10 ratio = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 100
30 total_14C = sum_species("*[14C]{O,[18O]}3*","[14C]") - sum_species("*{H,D,T}[14C]{O,[18O↩
     ]}3*","[14C]")
40 total_12C = sum_species("*C{O,[18O]}3*","C") - sum_species("*{H,D,T}C{O,[18O]}3*","C")
41 total_13C = sum_species("*[13C]{O,[18O]}3*","[13C]") - sum_species("*{H,D,T}[13C]{O,[18O↩
     ]}3*","[13C]")
50 if (total_12C <= 0) THEN GOTO 100
60 ratio = total_14C/(total_12C+total_13C+total_14C)
100 save ratio
     -end
#
# CH4(aq) ratios
#
R(D)_CH4(aq)
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 100
30 total_D = sum_species("{C,[13C],[14C]}{H,D,T}4","D")
40 total_H = sum_species("{C,[13C],[14C]}{H,D,T}4","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_D/total_H
100 save ratio
     -end

R(T)_CH4(aq)
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 100
30 total_T = sum_species("{C,[13C],[14C]}{H,D,T}4","T")
40 total_H = sum_species("{C,[13C],[14C]}{H,D,T}4","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_T/total_H
100 save ratio
     -end

R(13C)_CH4(aq)
     -start
10 ratio = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 100
30 total_13C = sum_species("[13C]{H,D,T}4","[13C]")
40 total_C = sum_species("C{H,D,T}4","C")
50 if (total_C <= 0) THEN GOTO 100
60 ratio = total_13C/total_C
100 save ratio
     -end

R(14C)_CH4(aq)
     -start
10 ratio = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 100
30 total_14C = sum_species("[14C]{H,D,T}4","[14C]")
40 total_C = sum_species("C{H,D,T}4","C")
41 total_13C = sum_species("[13C]{H,D,T}4","[13C]")
50 if (total_C <= 0) THEN GOTO 100
60 ratio = total_14C/(total_C+total_13C+total_14C)
100 save ratio
     -end
#
# SO4-2 ratios
#
R(34S)_SO4-2
     -start
10 ratio = -9999.999
20 if (TOT("[34S]") <= 0) THEN GOTO 100
30 total_34S = sum_species("*[34S]O4*","[34S]") - sum_species("*{H,D,T}[34S]O4*","[34S]")
40 total_S = sum_species("*SO4*","S") - sum_species("*{H,D,T}SO4*","S")
50 if (total_S <= 0) THEN GOTO 100
60 ratio = total_34S/total_S
100 save ratio
     -end
#
# HSO4- ratios
#
R(D)_HSO4-
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 100
30 total_D = sum_species("*D{S,[34S]}O4*","D")
40 total_H = sum_species("*H{S,[34S]}O4*","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_D/total_H
100 save ratio
     -end

R(T)_HSO4-
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 100
30 total_T = sum_species("*T{S,[34S]}O4*","T")
40 total_H = sum_species("*H{S,[34S]}O4*","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_T/total_H
100 save ratio
     -end

R(34S)_HSO4-
     -start
10 ratio = -9999.999
20 if (TOT("[34S]") <= 0) THEN GOTO 100
30 total_34S = sum_species("*{H,D,T}[34S]O4*","[34S]")
40 total_S = sum_species("*{H,D,T}SO4*","S")
50 if (total_S <= 0) THEN GOTO 100
60 ratio = total_34S/total_S
100 save ratio
     -end
#
# S-2 ratios
#
R(34S)_S-2
     -start
10 ratio = -9999.999
20 if (TOT("[34S](-2)") <= 0) THEN GOTO 100
```

```
30 total_34S = MOL("[34S]-2")
40 total_S = MOL("S-2")
50 if (total_S <= 0) THEN GOTO 100
60 ratio = total_34S/total_S
100 save ratio
     -end
#
# HS- ratios
#
R(D)_HS-
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 100
30 total_D = sum_species("*D{S,[34S]}*","D") - sum_species("*D{S,[34S]}O4*","D")
40 total_H = sum_species("*H{S,[34S]}*","H") - sum_species("*H{S,[34S]}O4*","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_D/total_H
100 save ratio
     -end

R(T)_HS-
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 100
30 total_T = sum_species("*T{S,[34S]}*","T") - sum_species("*T{S,[34S]}O4*","T")
40 total_H = sum_species("*H{S,[34S]}*","H") - sum_species("*H{S,[34S]}O4*","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_T/total_H
100 save ratio
     -end

R(34S)_HS-
     -start
10 ratio = -9999.999
20 if (TOT("[34S]") <= 0) THEN GOTO 100
30 total_34S = sum_species("*{H,D,T}[34S]*","[34S]") - sum_species("*{H,D,T}[34S]O4*","[34S]"↩
     )
40 total_S = sum_species("*{H,D,T}S*","S") - sum_species("*{H,D,T}SO4*","S")
50 if (total_S <= 0) THEN GOTO 100
60 ratio = total_34S/total_S
100 save ratio
     -end
#
# H2S ratios
#
R(D)_H2S(aq)
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 100
30 total_D = sum_species("*{H,D,T}2{S,[34S]}*","D") - sum_species("*{H,D,T}2{S,[34S]}O4*","D"↩
     )
40 total_S = sum_species("*{H,D,T}2{S,[34S]}*","H") - sum_species("*{H,D,T}2{S,[34]}SO4*","H"↩
     )
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_D/total_H
100 save ratio
     -end

R(T)_H2S(aq)
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 100
30 total_T = sum_species("{H,D,T}2{S,[34S]}","T")
40 total_H = sum_species("{H,D,T}2{S,[34S]}","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_T/total_H
100 save ratio
     -end

R(34S)_H2S(aq)
     -start
10 ratio = -9999.999
20 if (TOT("[34S]") <= 0) THEN GOTO 100
30 total_34S = sum_species("{H,D,T}2[34S]","[34S]")
40 total_S = sum_species("{H,D,T}2S","S")
50 if (total_S <= 0) THEN GOTO 100
60 ratio = total_34S/total_S
100 save ratio
     -end
#
# NO3- ratios
#
R(15N)_NO3-
     -start
10 ratio = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 100
30 total_15N = TOT("[15N](5)")
40 total_N = TOT("N(5)")
50 if (total_N <= 0) THEN GOTO 100
60 ratio = total_15N/total_N
100 save ratio
     -end
#
# NO2- ratios
#
R(15N)_NO2-
     -start
10 ratio = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 100
30 total_15N = TOT("[15N](3)")
40 total_N = TOT("N(3)")
50 if (total_N <= 0) THEN GOTO 100
60 ratio = total_15N/total_N
100 save ratio
     -end
#
# N2(aq) ratios
#
R(15N)_N2(aq)
     -start
10 ratio = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 100
30 total_15N = TOT("[15N](0)")
40 total_N = TOT("N(0)")
50 if (total_N <= 0) THEN GOTO 100
60 ratio = total_15N/total_N
100 save ratio
     -end
#
# NH3(aq) ratios
```

```
#
R(D)_NH3(aq)
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 100
30 total_D = sum_species("{[15N],N}{H,D,T}3","D")
40 total_H = sum_species("{[15N],N}{H,D,T}3","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_D/total_H
100 save ratio
     -end

R(T)_NH3(aq)
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 100
30 total_T = sum_species("{[15N],N}{H,D,T}3","T")
40 total_H = sum_species("{[15N],N}{H,D,T}3","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_T/total_H
100 save ratio
     -end

R(15N)_NH3(aq)
     -start
10 ratio = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 100
30 total_15N = sum_species("[15N]{H,D,T}3","[15N]")
40 total_N = sum_species("N{H,D,T}3","N")
50 if (total_N <= 0) THEN GOTO 100
60 ratio = total_15N/total_N
100 save ratio
     -end
#
# NH4+ ratios
#
R(D)_NH4+
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 100
30 total_D = sum_species("*{[15N],N}{H,D,T}4*","D")
40 total_H = sum_species("*{[15N],N}{H,D,T}4*","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_D/total_H
100 save ratio
     -end

R(T)_NH4+
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 100
30 total_T = sum_species("*{[15N],N}{H,D,T}4*","T")
40 total_H = sum_species("*{[15N],N}{H,D,T}4*","H")
50 if (total_H <= 0) THEN GOTO 100
60 ratio = total_T/total_H
100 save ratio
     -end

R(15N)_NH4+
     -start
10 ratio = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 100
30 total_15N = sum_species("*[15N]{H,D,T}4*","[15N]")
40 total_N = sum_species("*N{H,D,T}4*","N")
50 if (total_N <= 0) THEN GOTO 100
60 ratio = total_15N/total_N
100 save ratio
     -end
#
#   Ratios for minerals and gases
#
#
# H2O(g) ratios
#
R(D)_H2O(g)
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (GAS("H2O(g)") <= 0) THEN GOTO 1000
40 total_D = SUM_GAS("{H,D,T}2{O,[18O]}", "D")
50 total_H = SUM_GAS("{H,D,T}2{O,[18O]}", "H")
60 if (total_h <= 0) THEN GOTO 1000
70 ratio = total_D/total_H
1000 save ratio
     -end

R(T)_H2O(g)
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (GAS("H2O(g)") <= 0) THEN GOTO 1000
40 total_T = SUM_GAS("{H,D,T}2{O,[18O]}", "T")
50 total_H = SUM_GAS("{H,D,T}2{O,[18O]}", "H")
60 if (total_h <= 0) THEN GOTO 1000
70 ratio = total_T/total_H
1000 save ratio
     -end

R(18O)_H2O(g)
     -start
10 ratio = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (GAS("H2O(g)") <= 0) THEN GOTO 1000
40 total_18O = SUM_GAS("{H,D,T}2[18O]", "[18O]")
50 total_O = SUM_GAS("{H,D,T}2O", "O")
60 if (total_O <= 0) THEN GOTO 1000
70 ratio = total_18O/total_O
1000 save ratio
     -end
#
# O2(g) ratios
#
R(18O)_O2(g)
     -start
10 ratio = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (GAS("O2(g)") <= 0) THEN GOTO 1000
40 total_18O = SUM_GAS("{O,[18O]}2", "[18O]")
50 total_O = SUM_GAS("{O,[18O]}2", "O")
60 if (total_O <= 0) THEN GOTO 1000
```

```
70 ratio = total_18O/total_O
1000 save ratio
     -end
#
# H2(g) ratios
#
R(D)_H2(g)
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (GAS("H2(g)") <= 0) THEN GOTO 1000
40 total_D = SUM_GAS("{H,D,T}2", "D")
50 total_H = SUM_GAS("{H,D,T}2", "H")
60 if (total_H <= 0) THEN GOTO 1000
70 ratio = total_D/total_H
1000 save ratio
     -end

R(T)_H2(g)
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (GAS("H2(g)") <= 0) THEN GOTO 1000
40 total_T = SUM_GAS("{H,D,T}2", "T")
50 total_H = SUM_GAS("{H,D,T}2", "H")
60 if (total_H <= 0) THEN GOTO 1000
70 ratio = total_T/total_H
1000 save ratio
     -end
#
# CO2(g) ratios
#
R(18O)_CO2(g)
     -start
10 ratio = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (GAS("CO2(g)") <= 0) THEN GOTO 1000
40 total_18O = SUM_GAS("{C,[13C],[14C]}{O,[18O]}2", "[18O]")
50 total_O = SUM_GAS("{C,[13C],[14C]}{O,[18O]}2", "O")
60 if (total_O <= 0) THEN GOTO 1000
70 ratio = total_18O/total_O
1000 save ratio
     -end

R(13C)_CO2(g)
     -start
10 ratio = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (GAS("CO2(g)") <= 0) THEN GOTO 1000
40 if (GAS("[13C]O2(g)") <= 0) THEN GOTO 1000
50 total_13C = SUM_GAS("[13C]{O,[18O]}2", "[13C]")
60 total_C = SUM_GAS("C{O,[18O]}2", "C")
70 if (total_C <= 0) THEN GOTO 1000
80 ratio = total_13C/total_C
1000 save ratio
     -end

R(14C)_CO2(g)
     -start
10 ratio = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (GAS("CO2(g)") <= 0) THEN GOTO 1000
40 if (GAS("[14C]O2(g)") <= 0) THEN GOTO 1000
50 total_14C = SUM_GAS("[14C]{O,[18O]}2", "[14C]")
60 total_C = SUM_GAS("C{O,[18O]}2", "C")
61 total_13C = SUM_GAS("[13C]{O,[18O]}2", "[13C]")
70 if (total_C <= 0) THEN GOTO 1000
80 ratio = total_14C/(total_C+total_13C+total_14C)
1000 save ratio
     -end
#
# Calcite ratios
#
R(18O)_Calcite
     -start
10 ratio = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (S_S("Calcite") <= 0) THEN GOTO 1000
40 total_18O = SUM_S_S("Calcite", "[18O]")
50 total_O = SUM_S_S("Calcite", "O")
60 if (total_O <= 0) THEN GOTO 1000
70 ratio = total_18O/total_O
1000 save ratio
     -end

R(13C)_Calcite
     -start
10 ratio = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (S_S("Calcite") <= 0) THEN GOTO 1000
40 total_13C = SUM_S_S("Calcite", "[13C]")
50 total_C = SUM_S_S("Calcite", "C")
60 if (total_C <= 0) THEN GOTO 1000
70 ratio = total_13C/total_C
1000 save ratio
     -end

R(14C)_Calcite
     -start
10 ratio = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (S_S("Calcite") <= 0) THEN GOTO 1000
40 total_14C = SUM_S_S("Calcite", "[14C]")
50 total_C = SUM_S_S("Calcite", "C")
51 total_13C = SUM_S_S("Calcite", "[13C]")
60 if (total_C <= 0) THEN GOTO 1000
70 ratio = total_14C/(total_C+total_13C+total_14C)
1000 save ratio
     -end
#
# Pyrite ratios
#
R(34S)_Pyrite
     -start
10 ratio = -9999.999
20 if (TOT("[34S]") <= 0) THEN GOTO 1000
30 if (S_S("Pyrite") <= 0) THEN GOTO 1000
40 total_34S = SUM_S_S("Pyrite", "[34S]")
50 total_S = SUM_S_S("Pyrite", "S")
60 if (total_S <= 0) THEN GOTO 1000
```

```
70 ratio = total_34S/total_S
1000 save ratio
     -end
#
# CH4(g) ratios
#
R(D)_CH4(g)
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (GAS("CH4(g)") <= 0) THEN GOTO 1000
40 total_D = SUM_GAS("{C,[13C],[14C]}{H,D,T}4", "D")
50 total_H = SUM_GAS("{C,[13C],[14C]}{H,D,T}4", "H")
60 if (total_H <= 0) THEN GOTO 1000
70 ratio = total_D/total_H
1000 save ratio
     -end

R(T)_CH4(g)
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (GAS("CH4(g)") <= 0) THEN GOTO 1000
40 total_T = SUM_GAS("{C,[13C],[14C]}{H,D,T}4", "T")
50 total_H = SUM_GAS("{C,[13C],[14C]}{H,D,T}4", "H")
60 if (total_H <= 0) THEN GOTO 1000
70 ratio = total_T/total_H
1000 save ratio
     -end

R(13C)_CH4(g)
     -start
10 ratio = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (GAS("CH4(g)") <= 0) THEN GOTO 1000
40 total_13C = SUM_GAS("[13C]{H,D,T}4", "[13C]")
50 total_C = SUM_GAS("C{H,D,T}4", "C")
60 if (total_C <= 0) THEN GOTO 1000
70 ratio = total_13C/total_C
1000 save ratio
     -end

R(14C)_CH4(g)
     -start
10 ratio = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (GAS("CH4(g)") <= 0) THEN GOTO 1000
40 total_14C = SUM_GAS("[14C]{H,D,T}4", "[14C]")
50 total_C = SUM_GAS("C{H,D,T}4", "C")
51 total_13C = SUM_GAS("[13C]{H,D,T}4", "[13C]")
60 if (total_C <= 0) THEN GOTO 1000
70 ratio = total_14C/(total_C+total_13C+total_14C)
1000 save ratio
     -end
#
# H2S(g) ratios
#
R(D)_H2S(g)
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (GAS("H2S(g)") <= 0) THEN GOTO 1000
40 total_D = SUM_GAS("{H,D,T}2{S,[34S]}", "D")
50 total_H = SUM_GAS("{H,D,T}2{S,[34S]}", "H")
60 if (total_H <= 0) THEN GOTO 1000
70 ratio = total_D/total_H
1000 save ratio
     -end

R(T)_H2S(g)
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (GAS("H2S(g)") <= 0) THEN GOTO 1000
40 total_T = SUM_GAS("{H,D,T}2{S,[34S]}", "T")
50 total_H = SUM_GAS("{H,D,T}2{S,[34S]}", "H")
60 if (total_H <= 0) THEN GOTO 1000
70 ratio = total_T/total_H
1000 save ratio
     -end

R(34S)_H2S(g)
     -start
10 ratio = -9999.999
20 if (TOT("[34S]") <= 0) THEN GOTO 1000
30 if (GAS("H2S(g)") <= 0) THEN GOTO 1000
40 total_34S = SUM_GAS("{H,D,T}2[34S]", "[34S]")
50 total_S = SUM_GAS("{H,D,T}2S", "S")
60 if (total_S <= 0) THEN GOTO 1000
70 ratio = total_34S/total_S
1000 save ratio
     -end
#
# Gypsum ratios
#
R(34S)_Gypsum
     -start
10 ratio = -9999.999
20 if (TOT("[34S]") <= 0) THEN GOTO 1000
30 if (S_S("Gypsum") <= 0) THEN GOTO 1000
40 total_34S = SUM_S_S("Gypsum", "[34S]")
50 total_S = SUM_S_S("Gypsum", "S")
60 if (total_S <= 0) THEN GOTO 1000
70 ratio = total_34S/total_S
1000 save ratio
     -end
#
# Anhydrite ratios
#
R(34S)_Anhydrite
     -start
10 ratio = -9999.999
20 if (TOT("[34S]") <= 0) THEN GOTO 1000
30 if (S_S("Anhydrite") <= 0) THEN GOTO 1000
40 total_34S = SUM_S_S("Anhydrite", "[34S]")
50 total_S = SUM_S_S("Anhydrite", "S")
60 if (total_S <= 0) THEN GOTO 1000
70 ratio = total_34S/total_S
1000 save ratio
     -end
```

```
#
# N2(g) ratios
#
R(15N)_N2(g)
     -start
10 ratio = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 1000
30 if (GAS("N2(g)") <= 0) THEN GOTO 1000
40 total_15N = SUM_GAS("{[15N],N}2", "[15N]")
50 total_N = SUM_GAS("{[15N],N}2", "N")
60 if (total_N <= 0) THEN GOTO 1000
70 ratio = total_15N/total_N
1000 save ratio
     -end
#
# NH3(g) ratios
#
R(D)_NH3(g)
     -start
10 ratio = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (GAS("NH3(g)") <= 0) THEN GOTO 1000
40 total_D = SUM_GAS("{[15N],N}{H,D,T}3", "D")
50 total_H = SUM_GAS("{[15N],N}{H,D,T}3", "H")
60 if (total_H <= 0) THEN GOTO 1000
70 ratio = total_D/total_H
1000 save ratio
     -end

R(T)_NH3(g)
     -start
10 ratio = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (GAS("NH3(g)") <= 0) THEN GOTO 1000
40 total_T = SUM_GAS("{[15N],N}{H,D,T}3", "T")
50 total_H = SUM_GAS("{[15N],N}{H,D,T}3", "H")
60 if (total_H <= 0) THEN GOTO 1000
70 ratio = total_T/total_H
1000 save ratio
     -end

R(15N)_NH3(g)
     -start
10 ratio = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 1000
30 if (GAS("NH3(g)") <= 0) THEN GOTO 1000
40 total_15N = SUM_GAS("[15N]{H,D,T}3", "[15N]")
50 total_N = SUM_GAS("N{H,D,T}3", "N")
60 if (total_N <= 0) THEN GOTO 1000
70 ratio = total_15N/total_N
1000 save ratio
     -end
#↩
     ##########################################################################################################↩

CALCULATE_VALUES
#↩
     ##########################################################################################################↩

#
#  Alphas
#
#
# H2O(l) ratios
#
#
# OH- ratios
#
Alpha_D_OH-/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 r1 = CALC_VALUE("R(D)_OH-")
40 r2 = CALC_VALUE("R(D)_H2O(l)")
50 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_T_OH-/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 r1 = CALC_VALUE("R(T)_OH-")
40 r2 = CALC_VALUE("R(T)_H2O(l)")
50 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_18O_OH-/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 r1 = CALC_VALUE("R(18O)_OH-")
40 r2 = CALC_VALUE("R(18O)_H2O(l)")
50 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# H3O+ ratios
#
Alpha_D_H3O+/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 r1 = CALC_VALUE("R(D)_H3O+")
40 r2 = CALC_VALUE("R(D)_H2O(l)")
50 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_T_H3O+/H2O(l)
     -start
```

```
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 r1 = CALC_VALUE("R(T)_H3O+")
40 r2 = CALC_VALUE("R(T)_H2O(l)")
50 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_18O_H3O+/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 r1 = CALC_VALUE("R(18O)_H3O+")
40 r2 = CALC_VALUE("R(18O)_H2O(l)")
50 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# O2(aq) ratios
#
Alpha_18O_O2(aq)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(18O)_O2(aq)")
60 r2 = CALC_VALUE("R(18O)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# H2(aq) ratios
#
Alpha_D_H2(aq)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(D)_H2(aq)")
60 r2 = CALC_VALUE("R(D)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_T_H2(aq)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(T)_H2(aq)")
60 r2 = CALC_VALUE("R(T)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# CO2(aq) ratios
#
Alpha_18O_CO2(aq)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(18O)_CO2(aq)")
50 r2 = CALC_VALUE("R(18O)_H2O(l)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_13C_CO2(aq)/CO2(g)
     -start
10 alpha = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(13C)_CO2(aq)")
50 r2 = CALC_VALUE("R(13C)_CO2(g)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
      -end

Alpha_14C_CO2(aq)/CO2(g)
     -start
10 alpha = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(14C)_CO2(aq)")
50 r2 = CALC_VALUE("R(14C)_CO2(g)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
      -end
#
# HCO3- ratios
#
Alpha_D_HCO3-/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(D)_HCO3-")
50 r2 = CALC_VALUE("R(D)_H2O(l)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_T_HCO3-/H2O(l)
```

```
     -start
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(T)_HCO3-")
50 r2 = CALC_VALUE("R(T)_H2O(l)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_18O_HCO3-/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(18O)_HCO3-")
50 r2 = CALC_VALUE("R(18O)_H2O(l)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_18O_HCO3-/CO2(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(18O)_HCO3-")
50 r2 = CALC_VALUE("R(18O)_CO2(aq)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_13C_HCO3-/CO2(g)
     -start
10 alpha = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (GAS("CO2(g)") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(13C)_HCO3-")
50 r2 = CALC_VALUE("R(13C)_CO2(g)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_13C_HCO3-/CO2(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(13C)_HCO3-")
50 r2 = CALC_VALUE("R(13C)_CO2(aq)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_14C_HCO3-/CO2(g)
     -start
10 alpha = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (GAS("CO2(g)") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(14C)_HCO3-")
50 r2 = CALC_VALUE("R(14C)_CO2(g)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_14C_HCO3-/CO2(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(14C)_HCO3-")
50 r2 = CALC_VALUE("R(14C)_CO2(aq)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# CO3-2 ratios
#
Alpha_18O_CO3-2/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(18O)_CO3-2")
50 r2 = CALC_VALUE("R(18O)_H2O(l)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_18O_CO3-2/CO2(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(18O)_CO3-2")
50 r2 = CALC_VALUE("R(18O)_CO2(aq)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_13C_CO3-2/CO2(g)
```

```
     -start
10 alpha = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (GAS("CO2(g)") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(13C)_CO3-2")
50 r2 = CALC_VALUE("R(13C)_CO2(g)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_13C_CO3-2/CO2(aq) #?
     -start
10 alpha = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(13C)_CO3-2")
50 r2 = CALC_VALUE("R(13C)_CO2(aq)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_14C_CO3-2/CO2(g)
     -start
10 alpha = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (GAS("CO2(g)") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(14C)_CO3-2")
50 r2 = CALC_VALUE("R(14C)_CO2(g)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_14C_CO3-2/CO2(aq) #?
     -start
10 alpha = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(14C)_CO3-2")
50 r2 = CALC_VALUE("R(14C)_CO2(aq)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# CH4(aq) ratios
#
Alpha_D_CH4(aq)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(D)_CH4(aq)")
60 r2 = CALC_VALUE("R(D)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_T_CH4(aq)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(T)_CH4(aq)")
60 r2 = CALC_VALUE("R(T)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_13C_CH4(aq)/CO2(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(13C)_CH4(aq)")
60 r2 = CALC_VALUE("R(13C)_CO2(aq)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_14C_CH4(aq)/CO2(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (TOT("C") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(14C)_CH4(aq)")
60 r2 = CALC_VALUE("R(14C)_CO2(aq)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# HSO4-2 ratios
#
Alpha_D_HSO4-/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (TOT("S(6)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(D)_HSO4-")
60 r2 = CALC_VALUE("R(D)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
```

```
|
| Alpha_T_HSO4-/H2O(l)
|      -start
| 10 alpha = -9999.999
| 20 if (TOT("T") <= 0) THEN GOTO 1000
| 30 if (TOT("S(6)") <= 0) THEN GOTO 1000
| 50 r1 = CALC_VALUE("R(T)_HSO4-")
| 60 r2 = CALC_VALUE("R(T)_H2O(l)")
| 70 if (r2 <= 0) THEN GOTO 1000
| 80 if (r1 <= -9999) THEN GOTO 1000
| 90 alpha = r1/r2
| 1000 save alpha
|      -end
|
| Alpha_34S_HSO4-/SO4-2
|      -start
| 10 alpha = -9999.999
| 20 if (TOT("[34S]") <= 0) THEN GOTO 1000
| 30 if (TOT("S(6)") <= 0) THEN GOTO 1000
| 50 r1 = CALC_VALUE("R(34S)_HSO4-")
| 60 r2 = CALC_VALUE("R(34S)_SO4-2")
| 70 if (r2 <= 0) THEN GOTO 1000
| 80 if (r1 <= -9999) THEN GOTO 1000
| 90 alpha = r1/r2
| 1000 save alpha
|      -end
| #
| # S-2 ratios
| #
| Alpha_34S_S-2/HS-
|      -start
| 10 alpha = -9999.999
| 20 if (TOT("[34S]") <= 0) THEN GOTO 1000
| 30 if (TOT("S(-2)") <= 0) THEN GOTO 1000
| 50 r1 = CALC_VALUE("R(34S)_S-2")
| 60 r2 = CALC_VALUE("R(34S)_HS-")
| 70 if (r2 <= 0) THEN GOTO 1000
| 80 if (r1 <= -9999) THEN GOTO 1000
| 90 alpha = r1/r2
| 1000 save alpha
|      -end
|
| #
| # HS- ratios
| #
| Alpha_D_HS-/H2O(l)
|      -start
| 10 alpha = -9999.999
| 20 if (TOT("D") <= 0) THEN GOTO 1000
| 30 if (TOT("S(-2)") <= 0) THEN GOTO 1000
| 50 r1 = CALC_VALUE("R(D)_HS-")
| 60 r2 = CALC_VALUE("R(D)_H2O(l)")
| 70 if (r2 <= 0) THEN GOTO 1000
| 80 if (r1 <= -9999) THEN GOTO 1000
| 90 alpha = r1/r2
| 1000 save alpha
|      -end
|
| Alpha_T_HS-/H2O(l)
|      -start
| 10 alpha = -9999.999
| 20 if (TOT("T") <= 0) THEN GOTO 1000
| 30 if (TOT("S(-2)") <= 0) THEN GOTO 1000
| 50 r1 = CALC_VALUE("R(T)_HS-")
| 60 r2 = CALC_VALUE("R(T)_H2O(l)")
| 70 if (r2 <= 0) THEN GOTO 1000
| 80 if (r1 <= -9999) THEN GOTO 1000
| 90 alpha = r1/r2
| 1000 save alpha
|      -end
|
| Alpha_34S_HS-/SO4-2
|      -start
| 10 alpha = -9999.999
| 20 if (TOT("[34S]") <= 0) THEN GOTO 1000
| 30 if (TOT("S(6)") <= 0) THEN GOTO 1000
| 40 if (TOT("S(-2)") <= 0) THEN GOTO 1000
| 50 r1 = CALC_VALUE("R(34S)_HS-")
| 60 r2 = CALC_VALUE("R(34S)_SO4-2")
| 70 if (r2 <= 0) THEN GOTO 1000
| 80 if (r1 <= -9999) THEN GOTO 1000
| 90 alpha = r1/r2
| 1000 save alpha
|      -end
| #
| # H2S ratios
| #
| Alpha_D_H2S(aq)/H2O(l)
|      -start
| 10 alpha = -9999.999
| 20 if (TOT("D") <= 0) THEN GOTO 1000
| 30 if (TOT("S(-2)") <= 0) THEN GOTO 1000
| 50 r1 = CALC_VALUE("R(D)_H2S(aq)")
| 60 r2 = CALC_VALUE("R(D)_H2O(l)")
| 70 if (r2 <= 0) THEN GOTO 1000
| 80 if (r1 <= -9999) THEN GOTO 1000
| 90 alpha = r1/r2
| 1000 save alpha
|      -end
|
| Alpha_D_H2S(aq)/HS-
|      -start
| 10 alpha = -9999.999
| 20 if (TOT("D") <= 0) THEN GOTO 1000
| 30 if (TOT("S(-2)") <= 0) THEN GOTO 1000
| 50 r1 = CALC_VALUE("R(D)_H2S(aq)")
| 60 r2 = CALC_VALUE("R(D)_HS-")
| 70 if (r2 <= 0) THEN GOTO 1000
| 80 if (r1 <= -9999) THEN GOTO 1000
| 90 alpha = r1/r2
| 1000 save alpha
|      -end
|
| Alpha_T_H2S(aq)/H2O(l)
|      -start
| 10 alpha = -9999.999
| 20 if (TOT("T") <= 0) THEN GOTO 1000
| 30 if (TOT("S(-2)") <= 0) THEN GOTO 1000
| 50 r1 = CALC_VALUE("R(T)_H2S(aq)")
|
```

```
60 r2 = CALC_VALUE("R(T)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_T_H2S(aq)/HS-
     -start
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (TOT("S(-2)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(T)_H2S(aq)")
60 r2 = CALC_VALUE("R(T)_HS-")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_34S_H2S(aq)/HS-
     -start
10 alpha = -9999.999
20 if (TOT("[34S]") <= 0) THEN GOTO 1000
30 if (TOT("S(-2)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(34S)_H2S(aq)")
60 r2 = CALC_VALUE("R(34S)_HS-")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# NO3- ratios
#
Alpha_15N_NO2-/NO3-
     -start
10 alpha = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 1000
30 if (TOT("N(5)") <= 0) THEN GOTO 1000
40 if (TOT("N(3)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(15N)_NO2-")
60 r2 = CALC_VALUE("R(15N)_NO3-")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# N2(aq) ratios
#
Alpha_15N_N2(aq)/NO3-
     -start
10 alpha = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 1000
30 if (TOT("N(5)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(15N)_N2(aq)")
60 r2 = CALC_VALUE("R(15N)_NO3-")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# NH3(aq) ratios
#
Alpha_D_NH3(aq)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (TOT("N(-3)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(D)_NH3(aq)")
60 r2 = CALC_VALUE("R(D)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_T_NH3(aq)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (TOT("N(-3)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(T)_NH3(aq)")
60 r2 = CALC_VALUE("R(T)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_15N_NH3(aq)/NO3-
     -start
10 alpha = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 1000
30 if (TOT("N(-3)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(15N)_NH3(aq)")
60 r2 = CALC_VALUE("R(15)_NO3-")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# NH4+ ratios
#
Alpha_D_NH4+/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (TOT("N(-3)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(D)_NH4+")
60 r2 = CALC_VALUE("R(D)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
```

```
Alpha_T_NH4+/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (TOT("N(-3)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(T)_NH4+")
60 r2 = CALC_VALUE("R(T)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_15N_NH4+/NO3-
     -start
10 alpha = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 1000
30 if (TOT("N(5)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(15N)_NH4+")
60 r2 = CALC_VALUE("R(15N)_NO3-")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_15N_NH4+/NH3(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 1000
30 if (TOT("N(5)") <= 0) THEN GOTO 1000
40 if (TOT("N(-3)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(15N)_NH4+")
60 r2 = CALC_VALUE("R(15N)_NH3(aq)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
#  Ratios for minerals and gases
#
#
# H2O(g) ratios
#
Alpha_D_H2O(g)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (GAS("H2O(g)") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(D)_H2O(g)")
50 r2 = CALC_VALUE("R(D)_H2O(l)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_T_H2O(g)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (GAS("H2O(g)") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(T)_H2O(g)")
50 r2 = CALC_VALUE("R(T)_H2O(l)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_18O_H2O(g)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (GAS("H2O(g)") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(18O)_H2O(g)")
50 r2 = CALC_VALUE("R(18O)_H2O(l)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# O2(g) ratios
#
Alpha_18O_O2(aq)/O2(g)
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (GAS("O2(g)") <= 0) THEN GOTO 1000
40 if (GAS("O[18O](g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(18O)_O2(aq)")
60 r2 = CALC_VALUE("R(18O)_O2(g)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_18O_O2(g)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (GAS("O2(g)") <= 0) THEN GOTO 1000
40 if (GAS("O[18O](g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(18O)_O2(g)")
60 r2 = CALC_VALUE("R(18O)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

#
# H2(g) ratios
#
Alpha_D_H2(g)/H2O(l)
```

```
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (GAS("H2(g)") <= 0) THEN GOTO 1000
40 if (GAS("HD(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(D)_H2(g)")
60 r2 = CALC_VALUE("R(D)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_T_H2(g)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (GAS("H2(g)") <= 0) THEN GOTO 1000
40 if (GAS("HT(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(T)_H2(g)")
60 r2 = CALC_VALUE("R(T)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_D_H2(aq)/H2(g)
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (GAS("H2(g)") <= 0) THEN GOTO 1000
40 if (GAS("HD(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(D)_H2(aq)")
60 r2 = CALC_VALUE("R(D)_H2(g)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_T_H2(aq)/H2(g)
     -start
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (GAS("H2(g)") <= 0) THEN GOTO 1000
40 if (GAS("HT(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(T)_H2(aq)")
60 r2 = CALC_VALUE("R(T)_H2(g)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# CO2(g) ratios
#
Alpha_18O_CO2(g)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (GAS("CO2(g)") <= 0) THEN GOTO 1000
40 if (GAS("CO[18O](g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(18O)_CO2(g)")
60 r2 = CALC_VALUE("R(18O)_H2O(l)")
70 if (r1 <= 0) THEN GOTO 1000
80 if (r2 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_18O_CO2(g)/CO2(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (GAS("CO2(g)") <= 0) THEN GOTO 1000
40 if (GAS("CO[18O](g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(18O)_CO2(g)")
60 r2 = CALC_VALUE("R(18O)_CO2(aq)")
70 if (r1 <= 0) THEN GOTO 1000
80 if (r2 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_13C_CO2(g)/CO2(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (GAS("CO2(g)") <= 0) THEN GOTO 1000
40 if (GAS("[13C]O2(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(13C)_CO2(g)")
60 r2 = CALC_VALUE("R(13C)_CO2(aq)")
70 if (r1 <= 0) THEN GOTO 1000
80 if (r2 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_14C_CO2(g)/CO2(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (GAS("CO2(g)") <= 0) THEN GOTO 1000
40 if (GAS("[134]O2(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(14C)_CO2(g)")
60 r2 = CALC_VALUE("R(14C)_CO2(aq)")
70 if (r1 <= 0) THEN GOTO 1000
80 if (r2 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# Calcite ratios
#
Alpha_18O_Calcite/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
```

```
30 if (S_S("Calcite") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(18O)_Calcite")
50 r2 = CALC_VALUE("R(18O)_H2O(l)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_18O_Calcite/CO3-2
     -start
10 alpha = -9999.999
20 if (TOT("[18O]") <= 0) THEN GOTO 1000
30 if (S_S("Calcite") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(18O)_Calcite")
50 r2 = CALC_VALUE("R(18O)_CO3-2")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_13C_Calcite/CO2(g)
     -start
10 alpha = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (S_S("Calcite") <= 0) THEN GOTO 1000
35 if (GAS("CO2(g)") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(13C)_Calcite")
50 r2 = CALC_VALUE("R(13C)_CO2(g)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_13C_Calcite/CO2(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (S_S("Calcite") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(13C)_Calcite")
50 r2 = CALC_VALUE("R(13C)_CO2(aq)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_13C_Calcite/CO3-2
     -start
10 alpha = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (S_S("Calcite") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(13C)_Calcite")
50 r2 = CALC_VALUE("R(13C)_CO3-2")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_14C_Calcite/CO2(g)
     -start
10 alpha = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (S_S("Calcite") <= 0) THEN GOTO 1000
35 if (GAS("CO2(g)") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(14C)_Calcite")
50 r2 = CALC_VALUE("R(14C)_CO2(g)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_14C_Calcite/CO2(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (S_S("Calcite") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(14C)_Calcite")
50 r2 = CALC_VALUE("R(14C)_CO2(aq)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_14C_Calcite/CO3-2
     -start
10 alpha = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (S_S("Calcite") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(14C)_Calcite")
50 r2 = CALC_VALUE("R(14C)_CO3-2")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# Pyrite ratios
#
Alpha_34S_Pyrite/HS-
     -start
10 alpha = -9999.999
20 if (TOT("[34S]") <= 0) THEN GOTO 1000
30 if (S_S("Pyrite") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(34S)_Pyrite")
50 r2 = CALC_VALUE("R(34S)_HS-")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# CH4(g) ratios
#
```

```
Alpha_D_CH4(aq)/CH4(g)
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (GAS("CH4(g)") <= 0) THEN GOTO 1000
40 if (GAS("CH3D(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(D)_CH4(aq)")
60 r2 = CALC_VALUE("R(D)_CH4(g)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_D_CH4(g)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (GAS("CH4(g)") <= 0) THEN GOTO 1000
40 if (GAS("CH3D(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(D)_CH4(g)")
60 r2 = CALC_VALUE("R(D)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_T_CH4(aq)/CH4(g)
     -start
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (GAS("CH4(g)") <= 0) THEN GOTO 1000
40 if (GAS("CH3T(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(T)_CH4(aq)")
60 r2 = CALC_VALUE("R(T)_CH4(g)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_T_CH4(g)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (GAS("CH4(g)") <= 0) THEN GOTO 1000
40 if (GAS("CH3T(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(T)_CH4(g)")
60 r2 = CALC_VALUE("R(T)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_13C_CH4(g)/CO2(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (GAS("CH4(g)") <= 0) THEN GOTO 1000
40 if (GAS("[13C]H4(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(13C)_CH4(g)")
60 r2 = CALC_VALUE("R(13C)_CO2(aq)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_13C_CH4(aq)/CH4(g)
     -start
10 alpha = -9999.999
20 if (TOT("[13C]") <= 0) THEN GOTO 1000
30 if (GAS("CH4(g)") <= 0) THEN GOTO 1000
40 if (GAS("[13C]H4(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(13C)_CH4(aq)")
60 r2 = CALC_VALUE("R(13C)_CH4(g)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_14C_CH4(g)/CO2(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (GAS("CH4(g)") <= 0) THEN GOTO 1000
40 if (GAS("[14C]H4(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(14C)_CH4(g)")
60 r2 = CALC_VALUE("R(14C)_CO2(aq)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_14C_CH4(aq)/CH4(g)
     -start
10 alpha = -9999.999
20 if (TOT("[14C]") <= 0) THEN GOTO 1000
30 if (GAS("CH4(g)") <= 0) THEN GOTO 1000
40 if (GAS("[14C]H4(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(14C)_CH4(aq)")
60 r2 = CALC_VALUE("R(14C)_CH4(g)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# H2S(g) ratios
#
Alpha_D_H2S(g)/H2S(aq)
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (TOT("S(-2)") <= 0) THEN GOTO 1000
```

```
50 r1 = CALC_VALUE("R(D)_H2S(g)")
60 r2 = CALC_VALUE("R(D)_H2S(aq)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_T_H2S(g)/H2S(aq)
     -start
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (TOT("S(-2)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(T)_H2S(g)")
60 r2 = CALC_VALUE("R(T)_H2S(aq)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_34S_H2S(g)/H2S(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[34S]") <= 0) THEN GOTO 1000
30 if (TOT("S(6)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(34S)_H2S(g)")
60 r2 = CALC_VALUE("R(34S)_H2S(aq)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# Gypsum ratios
#
Alpha_34S_Gypsum/SO4-2
     -start
10 alpha = -9999.999
20 if (TOT("[34S]") <= 0) THEN GOTO 1000
30 if (TOT("S(6)") <= 0) THEN GOTO 1000
40 if (S_S("Gypsum") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(34S)_Gypsum")
60 r2 = CALC_VALUE("R(34S)_SO4-2")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# Anhydrite ratios
#
Alpha_34S_Anhydrite/SO4-2
     -start
10 alpha = -9999.999
20 if (TOT("[34S]") <= 0) THEN GOTO 1000
30 if (TOT("S(6)") <= 0) THEN GOTO 1000
40 if (S_S("Anhydrite") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(34S)_Anhydrite")
60 r2 = CALC_VALUE("R(34S)_SO4-2")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# N2(g) ratios
#
Alpha_15N_N2(g)/N2(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 1000
30 if (TOT("N(0)") <= 0) THEN GOTO 1000
40 if (GAS("N2(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(15N)_N2(g)")
60 r2 = CALC_VALUE("R(15N)_N2(aq)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#
# NH3(g) ratios
#
Alpha_D_NH3(g)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("D") <= 0) THEN GOTO 1000
30 if (TOT("N(-3)") <= 0) THEN GOTO 1000
40 if (GAS("NH3(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(D)_NH3(g)")
60 r2 = CALC_VALUE("R(D)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_T_NH3(g)/H2O(l)
     -start
10 alpha = -9999.999
20 if (TOT("T") <= 0) THEN GOTO 1000
30 if (TOT("N(-3)") <= 0) THEN GOTO 1000
40 if (GAS("NH3(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(T)_NH3(g)")
60 r2 = CALC_VALUE("R(T)_H2O(l)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

Alpha_15N_NH3(g)/NH3(aq)
     -start
10 alpha = -9999.999
20 if (TOT("[15N]") <= 0) THEN GOTO 1000
30 if (TOT("N(-3)") <= 0) THEN GOTO 1000
40 if (GAS("NH3(g)") <= 0) THEN GOTO 1000
50 r1 = CALC_VALUE("R(15N)_NH3(g)")
```

```
60 r2 = CALC_VALUE("R(15N)_NH3(aq)")
70 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end
#↩
     ##########################################################################################################↩

SOLUTION_SPECIES
#↩
     ##########################################################################################################↩

D2O = D2O
      -activity_water
      log_k           0

HTO = HTO
      -activity_water
      log_k           0

H2[18O] = H2[18O]
      -activity_water
      log_k           0

CO2 = CO2
      log_k           0

[13C]O2 = [13C]O2
      log_k           0

[14C]O2 = [14C]O2
      log_k           0

[34S]O4-2 = [34S]O4-2
      log_k           0.0
      -gamma          5.0000    -0.0400

[15N]O3- = [15N]O3-
      log_k           0.0
      -gamma          3.0000     0.0000

#
# H2O reactions
# last update July 31, 2006
# checked September 19, 2006
#
0.5H2O + 0.5D2O = HDO
      log_k           0.301029995663                          # log10(2)
      -activity_water

HDO + HTO = DTO + H2O
      -mole_balance   DTO
      log_k           -0.301029995663                         # log10(1/2)
      -activity_water

2HTO = T2O + H2O
      -mole_balance   T2O
      log_k           -0.6020599913279623960                  # log10(1/4)
      -activity_water

H2[18O] + HDO = HD[18O] + H2O
      -mole_balance   HD[18O]
      log_k           0.0
      -activity_water

H2[18O] + HTO = HT[18O] + H2O
      -mole_balance   HT[18O]
      log_k           0.0
      -activity_water

DTO  + H2[18O] = DT[18O] + H2O
      -mole_balance   DT[18O]
      log_k           0.0
      -activity_water

D2O + H2[18O] = D2[18O] + H2O
      -mole_balance   D2[18O]
      log_k           0.0
      -activity_water

H2[18O] + T2O = T2[18O] + H2O
      -mole_balance   T2[18O]
      log_k           0.0
      -activity_water
#
# OH- reactions
# last update July 31, 2006
# checked September 19, 2006
#
OH- + HDO = OD- + H2O
      -mole_balance   OD
      log_k           -0.301029995663                         # -log10(2)
      -add_logk       Log_alpha_D_OH-/H2O(l)                  1.0
      -gamma          3.5000     0.0000

OH- + HTO = OT- + H2O
      -mole_balance   OT
      log_k           -0.301029995663                         # -log10(2)
      -add_logk       Log_alpha_T_OH-/H2O(l)                  1.0
      -gamma          3.5000     0.0000

OH- + H2[18O] = [18O]H- + H2O
      -mole_balance   [18O]H
      -add_logk       Log_alpha_18O_OH-/H2O(l)                1.0
      -gamma          3.5000     0.0000

[18O]H- + OD- = [18O]D- + OH-
      -mole_balance   [18O]D
      log_k           0.0
      -gamma          3.5000     0.0000

[18O]H- + OT- = [18O]T- + OH-
      -mole_balance   [18O]T
      log_k           0.0
      -gamma          3.5000     0.0000
#
# H3O+ reactions
# last update July 31, 2006
# checked September 19, 2006
#
H3O+ + HDO = H2DO+ + H2O
      -mole_balance   H2DO
```

```
      log_k             0.176091259055                          # log10(1.5)
      -add_logk         Log_alpha_D_H3O+/H2O(l)                  1.0
      -gamma            9.0000    0.0000

2H2DO+ = HD2O+ + H3O+
      -mole_balance     HD2O
      log_k             -0.477121254719662                      # log10(1/3)
      -gamma            9.0000    0.0000

H2DO+ + HD2O+ = D3O+ + H3O+
      -mole_balance     D3O
      log_k             -0.954242509439324                     # log10(1/9)
      -gamma            9.0000    0.0000

H3O+ + HTO = H2TO+ + H2O
      -mole_balance     H2TO
      log_k             0.176091259055                          # log10(1.5)
      -add_logk         Log_alpha_T_H3O+/H2O(l)                  1.0
      -gamma            9.0000    0.0000

2H2TO+ = HT2O+ + H3O+
      -mole_balance     HT2O
      log_k             -0.477121254719662                      # log10(1/3)
      -gamma            9.0000    0.0000

H2TO+ + HT2O+ = T3O+ + H3O+
      -mole_balance     T3O
      log_k             -0.954242509439324                     # log10(1/9)
      -gamma            9.0000    0.0000

H2TO+ + H2DO+ = HDTO+ + H3O+
      -mole_balance     HDTO
      log_k             -0.477121254719662                      # log10(1/3)
      -gamma            9.0000    0.0000

H2TO+ + HD2O+ = D2TO+ + H3O+
      -mole_balance     D2TO
      log_k             -0.477121254719662                      # log10(1/3)
      -gamma            9.0000    0.0000

H2DO+ + HT2O+ = DT2O+ + H3O+
      -mole_balance     DT2O
      log_k             -0.477121254719662                      # log10(1/3)
      -gamma            9.0000    0.0000

H3O+ + H2[18O] = H3[18O]+ + H2O
      -mole_balance     H3[18O]
      -add_logk         Log_alpha_18O_H3O+/H2O(l)                1.0
      -gamma            9.0000    0.0000

H3[18O]+ + H2DO+ = H2D[18O]+ + H3O+
      -mole_balance     H2D[18O]
      log_k             0.0
      -gamma            9.0000    0.0000

H3[18O]+ + HD2O+ = HD2[18O]+ + H3O+
      -mole_balance     HD2[18O]
      log_k             0.0
      -gamma            9.0000    0.0000

H3[18O]+ + D3O+ = D3[18O]+ + H3O+
      -mole_balance     D3[18O]
      log_k             0.0
      -gamma            9.0000    0.0000

H3[18O]+ + H2TO+ = H2T[18O]+ + H3O+
      -mole_balance     H2T[18O]
      log_k             0.0
      -gamma            9.0000    0.0000

H3[18O]+ + HT2O+ = HT2[18O]+ + H3O+
      -mole_balance     HT2[18O]
      log_k             0.0
      -gamma            9.0000    0.0000

H3[18O]+ + T3O+ = T3[18O]+ + H3O+
      -mole_balance     T3[18O]
      log_k             0.0
      -gamma            9.0000    0.0000

H3[18O]+ + HDTO+ = HDT[18O]+ + H3O+
      -mole_balance     HDT[18O]
      log_k             0.0
      -gamma            9.0000    0.0000

H3[18O]+ + D2TO+ = D2T[18O]+ + H3O+
      -mole_balance     D2T[18O]
      log_k             0.0
      -gamma            9.0000    0.0000

H3[18O]+ + DT2O+ = DT2[18O]+ + H3O+
      -mole_balance     DT2[18O]
      log_k             0.0
      -gamma            9.0000    0.0000

#
# O2 reactions
# last update July 31, 2006
# checked September 19, 2006
#


O2 + H2[18O] = O[18O] + H2O
      log_k             0.301029995663                          # log10(2)
      -add_logk         Log_alpha_18O_O2(aq)/H2O(l)              1.0
      -mole_balance     O(0)[18O](0)

2O[18O] = [18O]2 + O2
      log_k             -0.602059991327962396                   # -log10(4)
      -mole_balance     [18O](0)2
#
# H2 reactions
# last update July 31, 2006
# checked September 19, 2006
#
H2 + HDO = HD + H2O
      -mole_balance     H(0)D(0)
      -add_logk         Log_alpha_D_H2(aq)/H2O(l)                1.0

2HD = D2 + H2
      -mole_balance     D(0)2
      log_k             -0.602059991327962396                   # -log10(4)
```

```
H2 + HTO = HT + H2O
     -mole_balance   H(0)T(0)
     -add_logk       Log_alpha_T_H2(aq)/H2O(l)                  1.0

2HT = T2 + H2
     -mole_balance   T(0)2
     log_k           -0.602059991327962396                      # -log10(4)

HT + HD = DT + H2
     -mole_balance   D(0)T(0)
     log_k           -0.301029995663                            # -log10(2)
#
# CO2 reactions
# last update July 31, 2006
# Checked September 19, 2006
#
CO2 + H2[18O] = CO[18O] + H2O
     log_k           0.301029995663                             # log10(2)
     -add_logk       Log_alpha_18O_CO2(aq)/H2O(l)               1.0

2CO[18O] = C[18O]2 + CO2
     log_k           -0.6020599913279623960                     # log10(1/4)

[13C]O2 + CO[18O] = [13C]O[18O] + CO2
     log_k           0

[13C]O2 + C[18O]2 = [13C][18O]2 + CO2
     log_k           0

[14C]O2 + CO[18O] = [14C]O[18O] + CO2
     log_k           0

[14C]O2 + C[18O]2 = [14C][18O]2 + CO2
     log_k           0
#
# HCO3- reactions
# last update July 31, 2006
# Checked September 19, 2006
#
HCO3- + H2[18O] = HCO2[18O]- + H2O
     -add_logk       Log_alpha_18O_HCO3-/H2O(l)                   1.0

HCO2[18O]- = HCO[18O]O-
HCO2[18O]- = HC[18O]O2-
2HCO2[18O]- = HCO[18O]2- + HCO3-
HCO[18O]2- = HC[18O]O[18O]-
HCO[18O]2- = HC[18O]2O-
3HCO2[18O]- = HC[18O]3- + 2HCO3-

HDO + HCO3- = DCO3- + H2O
     log_k           -0.301029995663                            # -log10(2)
     -add_logk       Log_alpha_D_HCO3-/H2O(l)                   1.0

DCO3- + HCO2[18O]- = DCO2[18O]- + HCO3-
DCO3- + HCO[18O]O- = DCO[18O]O- + HCO3-
DCO3- + HC[18O]O2- = DC[18O]O2- + HCO3-
DCO3- + HCO[18O]2- = DCO[18O]2- + HCO3-
DCO3- + HC[18O]O[18O]- = DC[18O]O[18O]- + HCO3-
DCO3- + HC[18O]2O- = DC[18O]2O- + HCO3-
DCO3- + HC[18O]3- = DC[18O]3- + HCO3-

HTO + HCO3- = TCO3- + H2O
     log_k           -0.301029995663                            # -log10(2)
     -add_logk       Log_alpha_T_HCO3-/H2O(l)                   1.0

TCO3- + HCO2[18O]- = TCO2[18O]- + HCO3-
TCO3- + HCO[18O]O- = TCO[18O]O- + HCO3-
TCO3- + HC[18O]O2- = TC[18O]O2- + HCO3-
TCO3- + HCO[18O]2- = TCO[18O]2- + HCO3-
TCO3- + HC[18O]O[18O]- = TC[18O]O[18O]- + HCO3-
TCO3- + HC[18O]2O- = TC[18O]2O- + HCO3-
TCO3- + HC[18O]3- = TC[18O]3- + HCO3-

HCO3- + [13C]O2 = H[13C]O3- + CO2
     -add_logk       Log_alpha_13C_HCO3-/CO2(aq)                1.0

H[13C]O3- + HCO2[18O]- = H[13C]O2[18O]- + HCO3-
H[13C]O3- + HCO[18O]O- = H[13C]O[18O]O- + HCO3-
H[13C]O3- + HC[18O]O2- = H[13C][18O]O2- + HCO3-
H[13C]O3- + HCO[18O]2- = H[13C]O[18O]2- + HCO3-
H[13C]O3- + HC[18O]O[18O]- = H[13C][18O]O[18O]- + HCO3-
H[13C]O3- + HC[18O]2O- = H[13C][18O]2O- + HCO3-
H[13C]O3- + HC[18O]3- = H[13C][18O]3- + HCO3-
DCO3- + H[13C]O3- = D[13C]O3- + HCO3-
H[13C]O3- + DCO2[18O]- = D[13C]O2[18O]- + HCO3-
H[13C]O3- + DCO[18O]O- = D[13C]O[18O]O- + HCO3-
H[13C]O3- + DC[18O]O2- = D[13C][18O]O2- + HCO3-
H[13C]O3- + DCO[18O]2- = D[13C]O[18O]2- + HCO3-
H[13C]O3- + DC[18O]O[18O]- = D[13C][18O]O[18O]- + HCO3-
H[13C]O3- + DC[18O]2O- = D[13C][18O]2O- + HCO3-
H[13C]O3- + DC[18O]3- = D[13C][18O]3- + HCO3-
TCO3- + H[13C]O3- = T[13C]O3- + HCO3-
H[13C]O3- + TCO2[18O]- = T[13C]O2[18O]- + HCO3-
H[13C]O3- + TCO[18O]O- = T[13C]O[18O]O- + HCO3-
H[13C]O3- + TC[18O]O2- = T[13C][18O]O2- + HCO3-
H[13C]O3- + TCO[18O]2- = T[13C]O[18O]2- + HCO3-
H[13C]O3- + TC[18O]O[18O]- = T[13C][18O]O[18O]- + HCO3-
H[13C]O3- + TC[18O]2O- = T[13C][18O]2O- + HCO3-
H[13C]O3- + TC[18O]3- = T[13C][18O]3- + HCO3-

HCO3- + [14C]O2 = H[14C]O3- + CO2
     -add_logk       Log_alpha_14C_HCO3-/CO2(aq)                1.0

H[14C]O3- + HCO2[18O]- = H[14C]O2[18O]- + HCO3-
H[14C]O3- + HCO[18O]O- = H[14C]O[18O]O- + HCO3-
H[14C]O3- + HC[18O]O2- = H[14C][18O]O2- + HCO3-
H[14C]O3- + HCO[18O]2- = H[14C]O[18O]2- + HCO3-
H[14C]O3- + HC[18O]O[18O]- = H[14C][18O]O[18O]- + HCO3-
H[14C]O3- + HC[18O]2O- = H[14C][18O]2O- + HCO3-
H[14C]O3- + HC[18O]3- = H[14C][18O]3- + HCO3-
DCO3- + H[14C]O3- = D[14C]O3- + HCO3-
H[14C]O3- + DCO2[18O]- = D[14C]O2[18O]- + HCO3-
H[14C]O3- + DCO[18O]O- = D[14C]O[18O]O- + HCO3-
H[14C]O3- + DC[18O]O2- = D[14C][18O]O2- + HCO3-
H[14C]O3- + DCO[18O]2- = D[14C]O[18O]2- + HCO3-
H[14C]O3- + DC[18O]O[18O]- = D[14C][18O]O[18O]- + HCO3-
H[14C]O3- + DC[18O]2O- = D[14C][18O]2O- + HCO3-
H[14C]O3- + DC[18O]3- = D[14C][18O]3- + HCO3-
TCO3- + H[14C]O3- = T[14C]O3- + HCO3-
```

```
H[14C]O3- + TCO2[18O]- = T[14C]O2[18O]- + HCO3-
H[14C]O3- + TCO[18O]O- = T[14C]O[18O]O- + HCO3-
H[14C]O3- + TC[18O]O2- = T[14C][18O]O2- + HCO3-
H[14C]O3- + TCO[18O]2- = T[14C]O[18O]2- + HCO3-
H[14C]O3- + TC[18O]O[18O]- = T[14C][18O]O[18O]- + HCO3-
H[14C]O3- + TC[18O]2O- = T[14C][18O]2O- + HCO3-
H[14C]O3- + TC[18O]3- = T[14C][18O]3- + HCO3-
#
# CO3-2 reactions
# last update July 31, 2006
# Checked September 19, 2006
#
CO3-2 + H2[18O] = CO2[18O]-2 + H2O
      log_k             0.477121254719                  # log10(3)
      -add_logk         Log_alpha_18O_CO3-2/H2O(l)      1.0

2CO2[18O]-2 = CO[18O]2-2 + CO3-2
      log_k             -0.477121254719                 # -log10(3)

CO2[18O]-2 + CO[18O]2-2 = C[18O]3-2 + CO3-2
      log_k             -0.954242509439324              # log10(1/9)

CO3-2 + [13C]O2 = [13C]O3-2 + CO2
      -add_logk         Log_alpha_13C_CO3-2/CO2(aq)     1.0

[13C]O3-2 + CO2[18O]-2 = [13C]O2[18O]-2 + CO3-2
      log_k             0.0

[13C]O3-2 + CO[18O]2-2 = [13C]O[18O]2-2 + CO3-2
      log_k             0.0

[13C]O3-2 + C[18O]3-2 = [13C][18O]3-2 + CO3-2
      log_k             0.0

CO3-2 + [14C]O2 = [14C]O3-2 + CO2
      -add_logk         Log_alpha_14C_CO3-2/CO2(aq)     1.0

[14C]O3-2 + CO2[18O]-2 = [14C]O2[18O]-2 + CO3-2
      log_k             0.0

[14C]O3-2 + CO[18O]2-2 = [14C]O[18O]2-2 + CO3-2
      log_k             0.0

[14C]O3-2 + C[18O]3-2 = [14C][18O]3-2 + CO3-2
      log_k             0.0
#
# CH4 reactions
# Updated September 19, 2006
# Checked September 19, 2006
#
CH4 + HDO = CH3D + H2O
      log_k             0.301029995663981198            # log10(2)
      -add_logk         Log_alpha_D_CH4(aq)/H2O(l)      1.0

2CH3D = CH2D2 + CH4
      log_k             -0.42596873227228               # log10(3/8)

3CH3D = CHD3 + 2CH4
      log_k             -1.20411998265                  # log10(1/16)

4CH3D = CD4 + 3CH4
      log_k             -2.408239965311                 # log10(1/256)

CH4 + HTO = CH3T + H2O
      log_k             0.301029995663981198            # log10(2)
      -add_logk         Log_alpha_T_CH4(aq)/H2O(l)      1.0

2CH3T = CH2T2 + CH4
      log_k             -0.42596873227228               # log10(3/8)

3CH3T = CHT3 + 2CH4
      log_k             -1.20411998265                  # log10(1/16)

4CH3T = CT4 + 3CH4
      log_k             -2.408239965311                 # log10(1/256)
#
#  Added mixed DT C methane species May 19, 2006
#
CH3T + CHD3 = CD3T + CH4
      log_k             -0.602059991327962396           # -log10(4)

CH2T2 + CH2D2 = CD2T2 + CH4
      log_k             -0.7781512503836                # -log10(6)

CHT3 + CH3D = CDT3 + CH4
      log_k             -0.602059991327962396           # -log10(4)

[13C]O2 + CH4 = [13C]H4 + CO2
      -add_logk         Log_alpha_13C_CH4(aq)/CO2(aq)   1.0

[13C]H4 + CH3D = [13C]H3D  + CH4
[13C]H4 + CH2D2 = [13C]H2D2 + CH4
[13C]H4 + CHD3 = [13C]HD3  + CH4
[13C]H4 + CD4 = [13C]D4   + CH4
[13C]H4 + CH3T = [13C]H3T  + CH4
[13C]H4 + CH2T2 = [13C]H2T2 + CH4
[13C]H4 + CHT3 = [13C]HT3  + CH4
[13C]H4 + CT4 = [13C]T4   + CH4
[13C]H4 + CD3T = [13C]D3T  + CH4
[13C]H4 + CD2T2 = [13C]D2T2 + CH4
[13C]H4 + CDT3 = [13C]DT3  + CH4

[14C]O2 + CH4 = [14C]H4 + CO2
      -add_logk         Log_alpha_14C_CH4(aq)/CO2(aq)   1.0
#
#  Added mixed DT 14C methane species May 19, 2006
#
[14C]H4 + CH3D = [14C]H3D  + CH4
[14C]H4 + CH2D2 = [14C]H2D2 + CH4
[14C]H4 + CHD3 = [14C]HD3  + CH4
[14C]H4 + CD4 = [14C]D4   + CH4
[14C]H4 + CH3T = [14C]H3T  + CH4
[14C]H4 + CH2T2 = [14C]H2T2 + CH4
[14C]H4 + CHT3 = [14C]HT3  + CH4
[14C]H4 + CT4 = [14C]T4   + CH4
[14C]H4 + CD3T = [14C]D3T  + CH4
[14C]H4 + CD2T2 = [14C]D2T2 + CH4
[14C]H4 + CDT3 = [14C]DT3  + CH4
#
#  HSO4- reactions
```

```
#  Updated September 28, 2006
#  Checked September 28, 2006
#
HSO4- + HDO = DSO4- + H2O
      log_k             -0.301029995663                  # -log10(2)
      -add_logk         Log_alpha_D_HSO4-/H2O(l)            1.0
      -mole_balance     DS(6)O4-

HSO4- + HTO = TSO4- + H2O
      log_k             -0.301029995663                  # -log10(2)
      -add_logk         Log_alpha_T_HSO4-/H2O(l)            1.0
      -mole_balance     TS(6)O4-

[34S]O4-2 + HSO4- = H[34S]O4- + SO4-2
      -add_logk         Log_alpha_34S_HSO4-/SO4-2        1.0
      -mole_balance     H[34S](6)O4-

H[34S]O4- + DSO4- = D[34S]O4- + HSO4-
      -mole_balance     D[34S](6)O4

H[34S]O4- + TSO4- = T[34S]O4- + HSO4-
      -mole_balance     T[34S](6)O4-
#
# S-2 reactions
#
S-2 + H[34S]- = [34S]-2 + HS-
      -add_logk         Log_alpha_34S_S-2/HS-            1.0
      -mole_balance     [34S](-2)
      -gamma            5.0000     0.0000
#
#  HS- reactions
#
HS- + HDO = DS- + H2O
      log_k             -0.301029995663                  # -log10(2)
      -add_logk         Log_alpha_D_HS-/H2O(l)              1.0
      -mole_balance     DS(-2)
      -gamma            3.5000     0.0000

HS- + HTO = TS- + H2O
      log_k             -0.301029995663                  # -log10(2)
      -add_logk         Log_alpha_T_HS-/H2O(l)              1.0
      -mole_balance     TS(-2)
      -gamma            3.5000     0.0000

HS- + [34S]O4-2 = H[34S]- + SO4-2
      -add_logk         Log_alpha_34S_HS-/SO4-2          1.0
      -mole_balance     H[34S](-2)
      -gamma            3.5000     0.0000

H[34S]- + DS- = D[34S]- + HS-
      -gamma            3.5000     0.0000
      -mole_balance     D[34S](-2)

H[34S]- + TS- = T[34S]- + HS-
      -gamma            3.5000     0.0000
      -mole_balance     T[34S](-2)
#
# H2S reactions
#
H2S + HDO = HDS + H2O
      -add_logk         Log_alpha_D_H2S(aq)/H2O(l)          1.0
      -mole_balance     HDS(-2)

2HDS = D2S + H2S
      log_k             -0.602059991327962396            # -log10(4)
      -mole_balance     D2S(-2)

H2S + HTO = HTS + H2O
      -add_logk         Log_alpha_T_H2S(aq)/H2O(l)          1.0
      -mole_balance     HTS(-2)

2HTS = T2S + H2S
      log_k             -0.602059991327962396            # -log10(4)
      -mole_balance     T2S(-2)

HDS + HTS = DTS + H2S
      log_k             -0.301029995663                          # log10(1/2)
      -mole_balance     DTS(-2)

H[34S]- + H2S = H2[34S] + HS-
      -add_logk         Log_alpha_34S_H2S(aq)/HS-          1.0
      -mole_balance     H2[34S](-2)

H2[34S] + HDS = HD[34S] + H2S
      -mole_balance     HD[34S](-2)

H2[34S] + D2S = D2[34S] + H2S
      -mole_balance     D2[34S](-2)

H2[34S] + HTS = HT[34S] + H2S
      -mole_balance     HT[34S](-2)

H2[34S] + T2S = T2[34S] + H2S
      -mole_balance     T2[34S](-2)

H2[34S] + DTS = DT[34S] + H2S
      -mole_balance     DT[34S](-2)
#
#   NO2- reactions
#   Updated March 20, 2006
#   Updated September 19, 2006
#   Checked September 19, 2006
#
[15N]O3- + NO2- = [15N]O2- + NO3-
      -add_logk         Log_alpha_15N_NO2-/NO3-             1.0
      -gamma            3.0000     0.0000
#
# N2 reactions
#
N2 + [15N]O3- = N[15N] + NO3-
      log_k             0.301029995663                   # log10(2)
      -add_logk         Log_alpha_15N_N2(aq)/NO3-           1.0
      -mole_balance     [15N](0)N(0)
2N[15N] = [15N]2 + N2
      log_k             -0.602059991327962396             # -log10(4)
      -mole_balance     [15N](0)2
#
# NH3 reactions
#
HDO + NH3 = NH2D + H2O
```

```
      log_k             0.176091259055                          # log10(1.5)
      -add_logk         Log_alpha_D_NH3(aq)/H2O(l)                  1.0

2NH2D = NHD2 + NH3
      log_k             -0.477121254719                           # -log10(3)

3NH2D = ND3 + 2NH3
      -logk             -1.431363764158                           # log10(1/27)

HTO + NH3 = NH2T + H2O
      log_k             0.176091259055                          # log10(1.5)
      -add_logk         Log_alpha_T_NH3(aq)/H2O(l)                  1.0

2NH2T = NHT2 + NH3
      log_k             -0.477121254719662                        # log10(1/3)

3NH2T = NT3 + 2NH3
      -logk             -1.431363764158                           # log10(1/27)

NHD2 + NH2T = ND2T + NH3
      log_k             -0.477121254719                           # -log10(3)

NH2D + NHT2 = NDT2 + NH3
      log_k             -0.477121254719                           # -log10(3)

# Checked September 19, 2006

NH3 + [15N]O3- = [15N]H3 + NO3-
      -add_logk         Log_alpha_15N_NH3(aq)/NO3-                    1.0
      -mole_balance     [15N](-3)H3

[15N]H3 + NH2D = [15N]H2D + NH3
[15N]H3 + NHD2 = [15N]HD2 + NH3
[15N]H3 + ND3 = [15N]D3  + NH3
[15N]H3 + NH2T = [15N]H2T + NH3
[15N]H3 + NHT2 = [15N]HT2 + NH3
[15N]H3 + NT3 = [15N]T3  + NH3
[15N]H3 + ND2T = [15N]D2T + NH3
[15N]H3 + NDT2 = [15N]DT2 + NH3
#
# NH4+ reactions
# Updated September 19, 2006
# Checked September 19, 2006
#
HDO + NH4+ = NH3D+ + H2O
      log_k             0.301029995663                            # log10(2)
      -add_logk         Log_alpha_D_NH4+/H2O(l)                     1.0
      -mole_balance     N(-3)H3D
      -gamma            2.5000    0.0000

2NH3D+ = NH2D2+ + NH4+
      log_k             -0.42596873227228                       # log10(3/8)
      -mole_balance     N(-3)H2D2
      -gamma            2.5000    0.0000

3NH3D+ = NHD3+ + 2NH4+
      log_k             -1.20411998265                            # log10(1/16)
      -mole_balance     N(-3)HD3
      -gamma            2.5000    0.0000

4NH3D+ = ND4+ + 3NH4+
      log_k             -2.408239965311                           # log10(1/256)
      -mole_balance     N(-3)D4
      -gamma            2.5000    0.0000

HTO + NH4+ = NH3T+ + H2O
      log_k             0.301029995663                            # log10(2)
      -add_logk         Log_alpha_T_NH4+/H2O(l)                     1.0
      -mole_balance     N(-3)H3T
      -gamma            2.5000    0.0000

2NH3T+ = NH2T2+ + NH4+
      log_k             -0.42596873227228                       # log10(3/8)
      -mole_balance     N(-3)H2T2
      -gamma            2.5000    0.0000

3NH3T+ = NHT3+ + 2NH4+
      log_k             -1.20411998265                            # log10(1/16)
      -mole_balance     N(-3)HT3
      -gamma            2.5000    0.0000

4NH3T+ = NT4+ + 3NH4+
      log_k             -2.408239965311                           # log10(1/256)
      -mole_balance     N(-3)T4
      -gamma            2.5000    0.0000
#
# Updated September 19, 2006
# Checked September 19, 2006
#
NHD3+ + NH3T+ = ND3T+ + NH4+
      log_k             -0.602059991327962396                     # -log10(4)
      -mole_balance     N(-3)D3T
      -gamma            2.5000    0.0000

NH2D2+ + NH2T2+ = ND2T2+ + NH4+
      log_k             -0.7781512503836                        # -log10(6)
      -mole_balance     N(-3)D2T2
      -gamma            2.5000    0.0000

NH3D+ + NHT3+ = NDT3+ + NH4+
      log_k             -0.602059991327962396                     # -log10(4)
      -mole_balance     N(-3)DT3
      -gamma            2.5000    0.0000

[15N]H3 + NH4+ = [15N]H4+ + NH3
      -add_logk         Log_alpha_15N_NH4+/NH3(aq)                  1.0
      -mole_balance     [15N](-3)H4
      -gamma            2.5000    0.0000

[15N]H4+ + NH3D+ = [15N]H3D+ + NH4+
      -mole_balance     [15N](-3)H3D
      -gamma            2.5000    0.0000

[15N]H4+ + NH2D2+ = [15N]H2D2+ + NH4+
      -mole_balance     [15N](-3)H2D2
      -gamma            2.5000    0.0000

[15N]H4+ + NHD3+ = [15N]HD3+ + NH4+
      -mole_balance     [15N](-3)HD3
      -gamma            2.5000    0.0000

[15N]H4+ + ND4+ = [15N]D4+ + NH4+
```

```
      -mole_balance    [15N](-3)D4+
      -gamma           2.5000     0.0000

[15N]H4+ + NH3T+ = [15N]H3T+ + NH4+
      -mole_balance    [15N](-3)H3T
      -gamma           2.5000     0.0000

[15N]H4+ + NH2T2+ = [15N]H2T2+ + NH4+
      -mole_balance    [15N](-3)H2T2
      -gamma           2.5000     0.0000

[15N]H4+ + NHT3+ = [15N]HT3+ + NH4+
      -mole_balance    [15N](-3)HT3
      -gamma           2.5000     0.0000

[15N]H4+ + NT4+ = [15N]T4+ + NH4+
      -mole_balance    [15N](-3)T4+
      -gamma           2.5000     0.0000

[15N]H4+ + ND3T+ = [15N]D3T+ + NH4+
      -mole_balance    [15N](-3)D3T
      -gamma           2.5000     0.0000

[15N]H4+ + ND2T2+ = [15N]D2T2+ + NH4+
      -mole_balance    [15N](-3)D2T2
      -gamma           2.5000     0.0000

[15N]H4+ + NDT3+ = [15N]DT3+ + NH4+
      -mole_balance    [15N](-3)DT3
      -gamma           2.5000     0.0000
#
# Missing NH4SO4- reactions
#
#
# Missing PO4-2 reactions
#
#
# Missing HPO4-2 reactions
#
#
# Missing H2PO4- reactions
#
#
# Added HF reactions 16Dec09
#
HF + HDO = DF + H2O
HF + HTO = TF + H2O
#
# Added HF2- reactions 16Dec09
#
HF2- + HDO = DF2- + H2O
HF2- + HTO = TF2- + H2O
#
# CaOH reactions
#
CaOH+ + [18O]H- = Ca[18O]H+ + OH-
      -mole_balance    Ca[18O]H
CaOH+ + OD- = CaOD+ + OH-
      -mole_balance    CaOD
CaOH+ + [18O]D- = Ca[18O]D+ + OH-
      -mole_balance    Ca[18O]D
CaOH+ + OT- = CaOT+ + OH-
      -mole_balance    CaOT+
CaOH+ + [18O]T- = Ca[18O]T+ + OH-
      -mole_balance    Ca[18O]T
#
# CaCO3 reactions
#
CaCO3 + CO2[18O]-2 = CaCO2[18O] + CO3-2
CaCO3 + CO[18O]2-2 = CaCO[18O]2 + CO3-2
CaCO3 + C[18O]3-2 = CaC[18O]3 + CO3-2
CaCO3 + [13C]O3-2 = Ca[13C]O3 + CO3-2
CaCO3 + [13C]O2[18O]-2 = Ca[13C]O2[18O] + CO3-2
CaCO3 + [13C]O[18O]2-2 = Ca[13C]O[18O]2 + CO3-2
CaCO3 + [13C][18O]3-2 = Ca[13C][18O]3 + CO3-2
CaCO3 + [14C]O3-2 = Ca[14C]O3 + CO3-2
CaCO3 + [14C]O2[18O]-2 = Ca[14C]O2[18O] + CO3-2
CaCO3 + [14C]O[18O]2-2 = Ca[14C]O[18O]2 + CO3-2
CaCO3 + [14C][18O]3-2 = Ca[14C][18O]3 + CO3-2
#
# CaHCO3+ reactions
#
CaHCO3+ + HCO2[18O]- = CaHCO2[18O]+ + HCO3-
      -gamma           5.4000     0.0000

CaHCO3+ + HCO[18O]O- = CaHCO[18O]O+ + HCO3-
      -gamma           5.4000     0.0000

CaHCO3+ + HC[18O]O2- = CaHC[18O]O2+ + HCO3-
      -gamma           5.4000     0.0000

CaHCO3+ + HCO[18O]2- = CaHCO[18O]2+ + HCO3-
      -gamma           5.4000     0.0000

CaHCO3+ + HC[18O]O[18O]- = CaHC[18O]O[18O]+ + HCO3-
      -gamma           5.4000     0.0000

CaHCO3+ + HC[18O]2O- = CaHC[18O]2O+ + HCO3-
      -gamma           5.4000     0.0000

CaHCO3+ + HC[18O]3- = CaHC[18O]3+ + HCO3-
      -gamma           5.4000     0.0000

CaHCO3+ + H[13C]O3- = CaH[13C]O3+ + HCO3-
      -gamma           5.4000     0.0000

CaHCO3+ + H[13C]O2[18O]- = CaH[13C]O2[18O]+ + HCO3-
      -gamma           5.4000     0.0000

CaHCO3+ + H[13C]O[18O]O- = CaH[13C]O[18O]O+ + HCO3-
      -gamma           5.4000     0.0000

CaHCO3+ + H[13C][18O]O2- = CaH[13C][18O]O2+ + HCO3-
      -gamma           5.4000     0.0000

CaHCO3+ + H[13C]O[18O]2- = CaH[13C]O[18O]2+ + HCO3-
      -gamma           5.4000     0.0000

CaHCO3+ + H[13C][18O]O[18O]- = CaH[13C][18O]O[18O]+ + HCO3-
      -gamma           5.4000     0.0000

CaHCO3+ + H[13C][18O]2O- = CaH[13C][18O]2O+ + HCO3-
```

```
    -gamma            5.4000    0.0000

CaHCO3+ + H[13C][18O]3- = CaH[13C][18O]3+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + H[14C]O3- = CaH[14C]O3+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + H[14C]O2[18O]- = CaH[14C]O2[18O]+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + H[14C]O[18O]O- = CaH[14C]O[18O]O+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + H[14C][18O]O2- = CaH[14C][18O]O2+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + H[14C]O[18O]2- = CaH[14C]O[18O]2+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + H[14C][18O]O[18O]- = CaH[14C][18O]O[18O]+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + H[14C][18O]2O- = CaH[14C][18O]2O+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + H[14C][18O]3- = CaH[14C][18O]3+ + HCO3-
    -gamma            5.4000    0.0000

# DCO3-
CaHCO3+ + DCO3- = CaDCO3+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + DCO2[18O]- = CaDCO2[18O]+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + DCO[18O]O- = CaDCO[18O]O+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + DC[18O]O2- = CaDC[18O]O2+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + DCO[18O]2- = CaDCO[18O]2+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + DC[18O]O[18O]- = CaDC[18O]O[18O]+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + DC[18O]2O- = CaDC[18O]2O+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + DC[18O]3- = CaDC[18O]3+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[13C]O3- = CaD[13C]O3+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[13C]O2[18O]- = CaD[13C]O2[18O]+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[13C]O[18O]O- = CaD[13C]O[18O]O+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[13C][18O]O2- = CaD[13C][18O]O2+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[13C]O[18O]2- = CaD[13C]O[18O]2+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[13C][18O]O[18O]- = CaD[13C][18O]O[18O]+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[13C][18O]2O- = CaD[13C][18O]2O+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[13C][18O]3- = CaD[13C][18O]3+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[14C]O3- = CaD[14C]O3+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[14C]O2[18O]- = CaD[14C]O2[18O]+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[14C]O[18O]O- = CaD[14C]O[18O]O+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[14C][18O]O2- = CaD[14C][18O]O2+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[14C]O[18O]2- = CaD[14C]O[18O]2+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[14C][18O]O[18O]- = CaD[14C][18O]O[18O]+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[14C][18O]2O- = CaD[14C][18O]2O+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + D[14C][18O]3- = CaD[14C][18O]3+ + HCO3-
    -gamma            5.4000    0.0000

# TCO3-
CaHCO3+ + TCO3- = CaTCO3+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + TCO2[18O]- = CaTCO2[18O]+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + TCO[18O]O- = CaTCO[18O]O+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + TC[18O]O2- = CaTC[18O]O2+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + TCO[18O]2- = CaTCO[18O]2+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + TC[18O]O[18O]- = CaTC[18O]O[18O]+ + HCO3-
    -gamma            5.4000    0.0000

CaHCO3+ + TC[18O]2O- = CaTC[18O]2O+ + HCO3-
    -gamma            5.4000    0.0000
```

```
CaHCO3+ + TC[18O]3- = CaTC[18O]3+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[13C]O3- = CaT[13C]O3+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[13C]O2[18O]- = CaT[13C]O2[18O]+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[13C]O[18O]O- = CaT[13C]O[18O]O+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[13C][18O]O2- = CaT[13C][18O]O2+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[13C]O[18O]2- = CaT[13C]O[18O]2+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[13C][18O]O[18O]- = CaT[13C][18O]O[18O]+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[13C][18O]2O- = CaT[13C][18O]2O+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[13C][18O]3- = CaT[13C][18O]3+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[14C]O3- = CaT[14C]O3+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[14C]O2[18O]- = CaT[14C]O2[18O]+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[14C]O[18O]O- = CaT[14C]O[18O]O+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[14C][18O]O2- = CaT[14C][18O]O2+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[14C]O[18O]2- = CaT[14C]O[18O]2+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[14C][18O]O[18O]- = CaT[14C][18O]O[18O]+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[14C][18O]2O- = CaT[14C][18O]2O+ + HCO3-
     -gamma            5.4000     0.0000

CaHCO3+ + T[14C][18O]3- = CaT[14C][18O]3+ + HCO3-
     -gamma            5.4000     0.0000
#
# CaSO4 reactions
#
# Updated May 17, 2006
CaSO4 + [34S]O4-2 = Ca[34S]O4 + SO4-2
#
# CaHSO4+ reactions
#
CaHSO4+ + H[34S]O4- = CaH[34S]O4+ + HSO4-
CaHSO4+ + D[34S]O4- = CaD[34S]O4+ + HSO4-
CaHSO4+ + T[34S]O4- = CaT[34S]O4+ + HSO4-
#
# Missing CaPO4- reactions
#
#
# Missing CaHPO4 reactions
#
#
# Missing CaH2PO4+ reactions
#
#
# MgOH+ reactions
#
MgOH+ + [18O]H- = Mg[18O]H+ + OH-
MgOH+ + OD- = MgOD+ + OH-
MgOH+ + [18O]D- = Mg[18O]D+ + OH-
MgOH+ + OT- = MgOT+ + OH-
MgOH+ + [18O]T- = Mg[18O]T+ + OH-
#
# MgCO3 reactions
#
MgCO3 + CO2[18O]-2 = MgCO2[18O] + CO3-2
MgCO3 + CO[18O]2-2 = MgCO[18O]2 + CO3-2
MgCO3 + C[18O]3-2 = MgC[18O]3 + CO3-2
MgCO3 + [13C]O3-2 = Mg[13C]O3 + CO3-2
MgCO3 + [13C]O2[18O]-2 = Mg[13C]O2[18O] + CO3-2
MgCO3 + [13C]O[18O]2-2 = Mg[13C]O[18O]2 + CO3-2
MgCO3 + [13C][18O]3-2 = Mg[13C][18O]3 + CO3-2
MgCO3 + [14C]O3-2 = Mg[14C]O3 + CO3-2
MgCO3 + [14C]O2[18O]-2 = Mg[14C]O2[18O] + CO3-2
MgCO3 + [14C]O[18O]2-2 = Mg[14C]O[18O]2 + CO3-2
MgCO3 + [14C][18O]3-2 = Mg[14C][18O]3 + CO3-2
#
# MgHCO3+ reactions
#
MgHCO3+ + HCO2[18O]- = MgHCO2[18O]+ + HCO3-
MgHCO3+ + HCO[18O]O- = MgHCO[18O]O+ + HCO3-
MgHCO3+ + HC[18O]O2- = MgHC[18O]O2+ + HCO3-
MgHCO3+ + HCO[18O]2- = MgHCO[18O]2+ + HCO3-
MgHCO3+ + HC[18O]O[18O]- = MgHC[18O]O[18O]+ + HCO3-
MgHCO3+ + HC[18O]2O- = MgHC[18O]2O+ + HCO3-
MgHCO3+ + HC[18O]3- = MgHC[18O]3+ + HCO3-
MgHCO3+ + H[13C]O3- = MgH[13C]O3+ + HCO3-
MgHCO3+ + H[13C]O2[18O]- = MgH[13C]O2[18O]+ + HCO3-
MgHCO3+ + H[13C]O[18O]O- = MgH[13C]O[18O]O+ + HCO3-
MgHCO3+ + H[13C][18O]O2- = MgH[13C][18O]O2+ + HCO3-
MgHCO3+ + H[13C]O[18O]2- = MgH[13C]O[18O]2+ + HCO3-
MgHCO3+ + H[13C][18O]O[18O]- = MgH[13C][18O]O[18O]+ + HCO3-
MgHCO3+ + H[13C][18O]2O- = MgH[13C][18O]2O+ + HCO3-
MgHCO3+ + H[13C][18O]3- = MgH[13C][18O]3+ + HCO3-
MgHCO3+ + H[14C]O3- = MgH[14C]O3+ + HCO3-
MgHCO3+ + H[14C]O2[18O]- = MgH[14C]O2[18O]+ + HCO3-
MgHCO3+ + H[14C]O[18O]O- = MgH[14C]O[18O]O+ + HCO3-
MgHCO3+ + H[14C][18O]O2- = MgH[14C][18O]O2+ + HCO3-
MgHCO3+ + H[14C]O[18O]2- = MgH[14C]O[18O]2+ + HCO3-
MgHCO3+ + H[14C][18O]O[18O]- = MgH[14C][18O]O[18O]+ + HCO3-
MgHCO3+ + H[14C][18O]2O- = MgH[14C][18O]2O+ + HCO3-
MgHCO3+ + H[14C][18O]3- = MgH[14C][18O]3+ + HCO3-

# DCO3-
MgHCO3+ + DCO3- = MgDCO3+ + HCO3-
```

```
| MgHCO3+ + DCO2[18O]- = MgDCO2[18O]+ + HCO3-                                    |
| MgHCO3+ + DCO[18O]O- = MgDCO[18O]O+ + HCO3-                                    |
| MgHCO3+ + DC[18O]O2- = MgDC[18O]O2+ + HCO3-                                    |
| MgHCO3+ + DCO[18O]2- = MgDCO[18O]2+ + HCO3-                                    |
| MgHCO3+ + DC[18O]O[18O]- = MgDC[18O]O[18O]+ + HCO3-                            |
| MgHCO3+ + DC[18O]2O- = MgDC[18O]2O+ + HCO3-                                    |
| MgHCO3+ + DC[18O]3- = MgDC[18O]3+ + HCO3-                                      |
| MgHCO3+ + D[13C]O3- = MgD[13C]O3+ + HCO3-                                      |
| MgHCO3+ + D[13C]O2[18O]- = MgD[13C]O2[18O]+ + HCO3-                            |
| MgHCO3+ + D[13C]O[18O]O- = MgD[13C]O[18O]O+ + HCO3-                            |
| MgHCO3+ + D[13C][18O]O2- = MgD[13C][18O]O2+ + HCO3-                            |
| MgHCO3+ + D[13C]O[18O]2- = MgD[13C]O[18O]2+ + HCO3-                            |
| MgHCO3+ + D[13C][18O]O[18O]- = MgD[13C][18O]O[18O]+ + HCO3-                    |
| MgHCO3+ + D[13C][18O]2O- = MgD[13C][18O]2O+ + HCO3-                            |
| MgHCO3+ + D[13C][18O]3- = MgD[13C][18O]3+ + HCO3-                              |
| MgHCO3+ + D[14C]O3- = MgD[14C]O3+ + HCO3-                                      |
| MgHCO3+ + D[14C]O2[18O]- = MgD[14C]O2[18O]+ + HCO3-                            |
| MgHCO3+ + D[14C]O[18O]O- = MgD[14C]O[18O]O+ + HCO3-                            |
| MgHCO3+ + D[14C][18O]O2- = MgD[14C][18O]O2+ + HCO3-                            |
| MgHCO3+ + D[14C]O[18O]2- = MgD[14C]O[18O]2+ + HCO3-                            |
| MgHCO3+ + D[14C][18O]O[18O]- = MgD[14C][18O]O[18O]+ + HCO3-                    |
| MgHCO3+ + D[14C][18O]2O- = MgD[14C][18O]2O+ + HCO3-                            |
| MgHCO3+ + D[14C][18O]3- = MgD[14C][18O]3+ + HCO3-                              |
|                                                                                |
| # TCO3-                                                                        |
| MgHCO3+ + TCO3- = MgTCO3+ + HCO3-                                              |
| MgHCO3+ + TCO2[18O]- = MgTCO2[18O]+ + HCO3-                                    |
| MgHCO3+ + TCO[18O]O- = MgTCO[18O]O+ + HCO3-                                    |
| MgHCO3+ + TC[18O]O2- = MgTC[18O]O2+ + HCO3-                                    |
| MgHCO3+ + TCO[18O]2- = MgTCO[18O]2+ + HCO3-                                    |
| MgHCO3+ + TC[18O]O[18O]- = MgTC[18O]O[18O]+ + HCO3-                            |
| MgHCO3+ + TC[18O]2O- = MgTC[18O]2O+ + HCO3-                                    |
| MgHCO3+ + TC[18O]3- = MgTC[18O]3+ + HCO3-                                      |
| MgHCO3+ + T[13C]O3- = MgT[13C]O3+ + HCO3-                                      |
| MgHCO3+ + T[13C]O2[18O]- = MgT[13C]O2[18O]+ + HCO3-                            |
| MgHCO3+ + T[13C]O[18O]O- = MgT[13C]O[18O]O+ + HCO3-                            |
| MgHCO3+ + T[13C][18O]O2- = MgT[13C][18O]O2+ + HCO3-                            |
| MgHCO3+ + T[13C]O[18O]2- = MgT[13C]O[18O]2+ + HCO3-                            |
| MgHCO3+ + T[13C][18O]O[18O]- = MgT[13C][18O]O[18O]+ + HCO3-                    |
| MgHCO3+ + T[13C][18O]2O- = MgT[13C][18O]2O+ + HCO3-                            |
| MgHCO3+ + T[13C][18O]3- = MgT[13C][18O]3+ + HCO3-                              |
| MgHCO3+ + T[14C]O3- = MgT[14C]O3+ + HCO3-                                      |
| MgHCO3+ + T[14C]O2[18O]- = MgT[14C]O2[18O]+ + HCO3-                            |
| MgHCO3+ + T[14C]O[18O]O- = MgT[14C]O[18O]O+ + HCO3-                            |
| MgHCO3+ + T[14C][18O]O2- = MgT[14C][18O]O2+ + HCO3-                            |
| MgHCO3+ + T[14C]O[18O]2- = MgT[14C]O[18O]2+ + HCO3-                            |
| MgHCO3+ + T[14C][18O]O[18O]- = MgT[14C][18O]O[18O]+ + HCO3-                    |
| MgHCO3+ + T[14C][18O]2O- = MgT[14C][18O]2O+ + HCO3-                            |
| MgHCO3+ + T[14C][18O]3- = MgT[14C][18O]3+ + HCO3-                              |
| #                                                                              |
| # MgSO4 reactions                                                              |
| #                                                                              |
| MgSO4 + [34S]O4-2 = Mg[34S]O4 + SO4-2                                          |
| #                                                                              |
| # Missing MgPO4- reactions                                                     |
| #                                                                              |
| #                                                                              |
| # Missing MgHPO4 reactions                                                     |
| #                                                                              |
| #                                                                              |
| # Missing Mg2PO4+ reactions                                                    |
| #                                                                              |
| #                                                                              |
| # NaOH reactions                                                               |
| #                                                                              |
| NaOH + [18O]H- = Na[18O]H + OH-                                                |
| NaOH + OD- = NaOD + OH-                                                        |
| NaOH + [18O]D- = Na[18O]D + OH-                                                |
| NaOH + OT- = NaOT + OH-                                                        |
| NaOH + [18O]T- = Na[18O]T + OH-                                                |
| #                                                                              |
| # NaCO3- reactions                                                             |
| #                                                                              |
| NaCO3- + CO2[18O]-2 = NaCO2[18O]- + CO3-2                                      |
| NaCO3- + CO[18O]2-2 = NaCO[18O]2- + CO3-2                                      |
| NaCO3- + C[18O]3-2 = NaC[18O]3- + CO3-2                                        |
| NaCO3- + [13C]O3-2 = Na[13C]O3- + CO3-2                                        |
| NaCO3- + [13C]O2[18O]-2 = Na[13C]O2[18O]- + CO3-2                              |
| NaCO3- + [13C]O[18O]2-2 = Na[13C]O[18O]2- + CO3-2                              |
| NaCO3- + [13C][18O]3-2 = Na[13C][18O]3- + CO3-2                                |
| NaCO3- + [14C]O3-2 = Na[14C]O3- + CO3-2                                        |
| NaCO3- + [14C]O2[18O]-2 = Na[14C]O2[18O]- + CO3-2                              |
| NaCO3- + [14C]O[18O]2-2 = Na[14C]O[18O]2- + CO3-2                              |
| NaCO3- + [14C][18O]3-2 = Na[14C][18O]3- + CO3-2                                |
| #                                                                              |
| # NaHCO3 reactions                                                             |
| #                                                                              |
| NaHCO3 + HCO2[18O]- = NaHCO2[18O] + HCO3-                                      |
| NaHCO3 + HCO[18O]O- = NaHCO[18O]O + HCO3-                                      |
| NaHCO3 + HC[18O]O2- = NaHC[18O]O2 + HCO3-                                      |
| NaHCO3 + HCO[18O]2- = NaHCO[18O]2 + HCO3-                                      |
| NaHCO3 + HC[18O]O[18O]- = NaHC[18O]O[18O] + HCO3-                              |
| NaHCO3 + HC[18O]2O- = NaHC[18O]2O + HCO3-                                      |
| NaHCO3 + HC[18O]3- = NaHC[18O]3 + HCO3-                                        |
| NaHCO3 + H[13C]O3- = NaH[13C]O3 + HCO3-                                        |
| NaHCO3 + H[13C]O2[18O]- = NaH[13C]O2[18O] + HCO3-                              |
| NaHCO3 + H[13C]O[18O]O- = NaH[13C]O[18O]O + HCO3-                              |
| NaHCO3 + H[13C][18O]O2- = NaH[13C][18O]O2 + HCO3-                              |
| NaHCO3 + H[13C]O[18O]2- = NaH[13C]O[18O]2 + HCO3-                              |
| NaHCO3 + H[13C][18O]O[18O]- = NaH[13C][18O]O[18O] + HCO3-                      |
| NaHCO3 + H[13C][18O]2O- = NaH[13C][18O]2O + HCO3-                              |
| NaHCO3 + H[13C][18O]3- = NaH[13C][18O]3 + HCO3-                                |
| NaHCO3 + H[14C]O3- = NaH[14C]O3 + HCO3-                                        |
| NaHCO3 + H[14C]O2[18O]- = NaH[14C]O2[18O] + HCO3-                              |
| NaHCO3 + H[14C]O[18O]O- = NaH[14C]O[18O]O + HCO3-                              |
| NaHCO3 + H[14C][18O]O2- = NaH[14C][18O]O2 + HCO3-                              |
| NaHCO3 + H[14C]O[18O]2- = NaH[14C]O[18O]2 + HCO3-                              |
| NaHCO3 + H[14C][18O]O[18O]- = NaH[14C][18O]O[18O] + HCO3-                      |
| NaHCO3 + H[14C][18O]2O- = NaH[14C][18O]2O + HCO3-                              |
| NaHCO3 + H[14C][18O]3- = NaH[14C][18O]3 + HCO3-                                |
|                                                                                |
| # DCO3-                                                                        |
| NaHCO3 + DCO3- = NaDCO3 + HCO3-                                                |
| NaHCO3 + DCO2[18O]- = NaDCO2[18O] + HCO3-                                      |
| NaHCO3 + DCO[18O]O- = NaDCO[18O]O + HCO3-                                      |
| NaHCO3 + DC[18O]O2- = NaDC[18O]O2 + HCO3-                                      |
| NaHCO3 + DCO[18O]2- = NaDCO[18O]2 + HCO3-                                      |
| NaHCO3 + DC[18O]O[18O]- = NaDC[18O]O[18O] + HCO3-                              |
```

```
NaHCO3 + DC[18O]2O- = NaDC[18O]2O + HCO3-
NaHCO3 + DC[18O]3- = NaDC[18O]3 + HCO3-
NaHCO3 + D[13C]O3- = NaD[13C]O3 + HCO3-
NaHCO3 + D[13C]O2[18O]- = NaD[13C]O2[18O] + HCO3-
NaHCO3 + D[13C]O[18O]O- = NaD[13C]O[18O]O + HCO3-
NaHCO3 + D[13C][18O]O2- = NaD[13C][18O]O2 + HCO3-
NaHCO3 + D[13C]O[18O]2- = NaD[13C]O[18O]2 + HCO3-
NaHCO3 + D[13C][18O]O[18O]- = NaD[13C][18O]O[18O] + HCO3-
NaHCO3 + D[13C][18O]2O- = NaD[13C][18O]2O + HCO3-
NaHCO3 + D[13C][18O]3- = NaD[13C][18O]3 + HCO3-
NaHCO3 + D[14C]O3- = NaD[14C]O3 + HCO3-
NaHCO3 + D[14C]O2[18O]- = NaD[14C]O2[18O] + HCO3-
NaHCO3 + D[14C]O[18O]O- = NaD[14C]O[18O]O + HCO3-
NaHCO3 + D[14C][18O]O2- = NaD[14C][18O]O2 + HCO3-
NaHCO3 + D[14C]O[18O]2- = NaD[14C]O[18O]2 + HCO3-
NaHCO3 + D[14C][18O]O[18O]- = NaD[14C][18O]O[18O] + HCO3-
NaHCO3 + D[14C][18O]2O- = NaD[14C][18O]2O + HCO3-
NaHCO3 + D[14C][18O]3- = NaD[14C][18O]3 + HCO3-

# TCO3-
NaHCO3 + TCO3- = NaTCO3 + HCO3-
NaHCO3 + TCO2[18O]- = NaTCO2[18O] + HCO3-
NaHCO3 + TCO[18O]O- = NaTCO[18O]O + HCO3-
NaHCO3 + TC[18O]O2- = NaTC[18O]O2 + HCO3-
NaHCO3 + TCO[18O]2- = NaTCO[18O]2 + HCO3-
NaHCO3 + TC[18O]O[18O]- = NaTC[18O]O[18O] + HCO3-
NaHCO3 + TC[18O]2O- = NaTC[18O]2O + HCO3-
NaHCO3 + TC[18O]3- = NaTC[18O]3 + HCO3-
NaHCO3 + T[13C]O3- = NaT[13C]O3 + HCO3-
NaHCO3 + T[13C]O2[18O]- = NaT[13C]O2[18O] + HCO3-
NaHCO3 + T[13C]O[18O]O- = NaT[13C]O[18O]O + HCO3-
NaHCO3 + T[13C][18O]O2- = NaT[13C][18O]O2 + HCO3-
NaHCO3 + T[13C]O[18O]2- = NaT[13C]O[18O]2 + HCO3-
NaHCO3 + T[13C][18O]O[18O]- = NaT[13C][18O]O[18O] + HCO3-
NaHCO3 + T[13C][18O]2O- = NaT[13C][18O]2O + HCO3-
NaHCO3 + T[13C][18O]3- = NaT[13C][18O]3 + HCO3-
NaHCO3 + T[14C]O3- = NaT[14C]O3 + HCO3-
NaHCO3 + T[14C]O2[18O]- = NaT[14C]O2[18O] + HCO3-
NaHCO3 + T[14C]O[18O]O- = NaT[14C]O[18O]O + HCO3-
NaHCO3 + T[14C][18O]O2- = NaT[14C][18O]O2 + HCO3-
NaHCO3 + T[14C]O[18O]2- = NaT[14C]O[18O]2 + HCO3-
NaHCO3 + T[14C][18O]O[18O]- = NaT[14C][18O]O[18O] + HCO3-
NaHCO3 + T[14C][18O]2O- = NaT[14C][18O]2O + HCO3-
NaHCO3 + T[14C][18O]3- = NaT[14C][18O]3 + HCO3-
#
# NaSO4- reactions
#
NaSO4- + [34S]O4-2 = Na[34S]O4- + SO4-2
#
# Missing NaHPO4- reactions
#
#
# KOH reactions
#
KOH + [18O]H- = K[18O]H + OH-
KOH + OD- = KOD + OH-
KOH + [18O]D- = K[18O]D + OH-
KOH + OT- = KOT + OH-
KOH + [18O]T- = K[18O]T + OH-
#
# KSO4- reactions
#
KSO4- + [34S]O4-2 = K[34S]O4- + SO4-2
#
# Added FeOH+ reactions 16Dec09
#
FeOH+ + OD- = FeOD+ + OH-
FeOH+ + OT- = FeOT+ + OH-
FeOH+ + [18O]H- = Fe[18O]H+ + OH-
FeOH+ + [18O]D- = Fe[18O]D+ + OH-
FeOH+ + [18O]T- = Fe[18O]T+ + OH-
#
# Added FeCO3 reactions 16Dec09
# Checked 17Dec09
#
FeCO3 + CO2[18O]-2 = FeCO2[18O] + CO3-2
FeCO3 + CO[18O]2-2 = FeCO[18O]2 + CO3-2
FeCO3 + C[18O]3-2 = FeC[18O]3 + CO3-2
FeCO3 + [13C]O3-2 = Fe[13C]O3 + CO3-2
FeCO3 + [13C]O2[18O]-2 = Fe[13C]O2[18O] + CO3-2
FeCO3 + [13C]O[18O]2-2 = Fe[13C]O[18O]2 + CO3-2
FeCO3 + [13C][18O]3-2 = Fe[13C][18O]3 + CO3-2
FeCO3 + [14C]O3-2 = Fe[14C]O3 + CO3-2
FeCO3 + [14C]O2[18O]-2 = Fe[14C]O2[18O] + CO3-2
FeCO3 + [14C]O[18O]2-2 = Fe[14C]O[18O]2 + CO3-2
FeCO3 + [14C][18O]3-2 = Fe[14C][18O]3 + CO3-2
#
# Added FeHCO3+ reactions 16Dec09
# Revised 17Dec09
#
FeHCO3+ + HCO2[18O]- = FeHCO2[18O]+ + HCO3-
FeHCO3+ + HCO[18O]O- = FeHCO[18O]O+ + HCO3-
FeHCO3+ + HCO[18O]2- = FeHCO[18O]2+ + HCO3-
FeHCO3+ + HC[18O]O2- = FeHC[18O]O2+ + HCO3-
FeHCO3+ + HC[18O]O[18O]- = FeHC[18O]O[18O]+ + HCO3-
FeHCO3+ + HC[18O]2O- = FeHC[18O]2O+ + HCO3-
FeHCO3+ + HC[18O]3- = FeHC[18O]3+ + HCO3-
FeHCO3+ + H[13C]O3- = FeH[13C]O3+ + HCO3-
FeHCO3+ + H[13C]O2[18O]- = FeH[13C]O2[18O]+ + HCO3-
FeHCO3+ + H[13C]O[18O]O- = FeH[13C]O[18O]O+ + HCO3-
FeHCO3+ + H[13C]O[18O]2- = FeH[13C]O[18O]2+ + HCO3-
FeHCO3+ + H[13C][18O]O2- = FeH[13C][18O]O2+ + HCO3-
FeHCO3+ + H[13C][18O]O[18O]- = FeH[13C][18O]O[18O]+ + HCO3-
FeHCO3+ + H[13C][18O]2O- = FeH[13C][18O]2O+ + HCO3-
FeHCO3+ + H[13C][18O]3- = FeH[13C][18O]3+ + HCO3-
FeHCO3+ + H[14C]O3- = FeH[14C]O3+ + HCO3-
FeHCO3+ + H[14C]O2[18O]- = FeH[14C]O2[18O]+ + HCO3-
FeHCO3+ + H[14C]O[18O]O- = FeH[14C]O[18O]O+ + HCO3-
FeHCO3+ + H[14C]O[18O]2- = FeH[14C]O[18O]2+ + HCO3-
FeHCO3+ + H[14C][18O]O2- = FeH[14C][18O]O2+ + HCO3-
FeHCO3+ + H[14C][18O]O[18O]- = FeH[14C][18O]O[18O]+ + HCO3-
FeHCO3+ + H[14C][18O]2O- = FeH[14C][18O]2O+ + HCO3-
FeHCO3+ + H[14C][18O]3- = FeH[14C][18O]3+ + HCO3-
FeHCO3+ + DCO3- = FeDCO3+ + HCO3-
FeHCO3+ + DCO2[18O]- = FeDCO2[18O]+ + HCO3-
FeHCO3+ + DCO[18O]O- = FeDCO[18O]O+ + HCO3-
FeHCO3+ + DCO[18O]2- = FeDCO[18O]2+ + HCO3-
FeHCO3+ + DC[18O]O2- = FeDC[18O]O2+ + HCO3-
```

```
FeHCO3+ + DC[18O]O[18O]- = FeDC[18O]O[18O]+ + HCO3-
FeHCO3+ + DC[18O]2O- = FeDC[18O]2O+ + HCO3-
FeHCO3+ + DC[18O]3- = FeDC[18O]3+ + HCO3-
FeHCO3+ + D[13C]O3- = FeD[13C]O3+ + HCO3-
FeHCO3+ + D[13C]O2[18O]- = FeD[13C]O2[18O]+ + HCO3-
FeHCO3+ + D[13C]O[18O]O- = FeD[13C]O[18O]O+ + HCO3-
FeHCO3+ + D[13C]O[18O]2- = FeD[13C]O[18O]2+ + HCO3-
FeHCO3+ + D[13C][18O]O2- = FeD[13C][18O]O2+ + HCO3-
FeHCO3+ + D[13C][18O]O[18O]- = FeD[13C][18O]O[18O]+ + HCO3-
FeHCO3+ + D[13C][18O]2O- = FeD[13C][18O]2O+ + HCO3-
FeHCO3+ + D[13C][18O]3- = FeD[13C][18O]3+ + HCO3-
FeHCO3+ + D[14C]O3- = FeD[14C]O3+ + HCO3-
FeHCO3+ + D[14C]O2[18O]- = FeD[14C]O2[18O]+ + HCO3-
FeHCO3+ + D[14C]O[18O]O- = FeD[14C]O[18O]O+ + HCO3-
FeHCO3+ + D[14C]O[18O]2- = FeD[14C]O[18O]2+ + HCO3-
FeHCO3+ + D[14C][18O]O2- = FeD[14C][18O]O2+ + HCO3-
FeHCO3+ + D[14C][18O]O[18O]- = FeD[14C][18O]O[18O]+ + HCO3-
FeHCO3+ + D[14C][18O]2O- = FeD[14C][18O]2O+ + HCO3-
FeHCO3+ + D[14C][18O]3- = FeD[14C][18O]3+ + HCO3-
FeHCO3+ + TCO3- = FeTCO3+ + HCO3-
FeHCO3+ + TCO2[18O]- = FeTCO2[18O]+ + HCO3-
FeHCO3+ + TCO[18O]O- = FeTCO[18O]O+ + HCO3-
FeHCO3+ + TCO[18O]2- = FeTCO[18O]2+ + HCO3-
FeHCO3+ + TC[18O]O2- = FeTC[18O]O2+ + HCO3-
FeHCO3+ + TC[18O]O[18O]- = FeTC[18O]O[18O]+ + HCO3-
FeHCO3+ + TC[18O]2O- = FeTC[18O]2O+ + HCO3-
FeHCO3+ + TC[18O]3- = FeTC[18O]3+ + HCO3-
FeHCO3+ + T[13C]O3- = FeT[13C]O3+ + HCO3-
FeHCO3+ + T[13C]O2[18O]- = FeT[13C]O2[18O]+ + HCO3-
FeHCO3+ + T[13C]O[18O]O- = FeT[13C]O[18O]O+ + HCO3-
FeHCO3+ + T[13C]O[18O]2- = FeT[13C]O[18O]2+ + HCO3-
FeHCO3+ + T[13C][18O]O2- = FeT[13C][18O]O2+ + HCO3-
FeHCO3+ + T[13C][18O]O[18O]- = FeT[13C][18O]O[18O]+ + HCO3-
FeHCO3+ + T[13C][18O]2O- = FeT[13C][18O]2O+ + HCO3-
FeHCO3+ + T[13C][18O]3- = FeT[13C][18O]3+ + HCO3-
FeHCO3+ + T[14C]O3- = FeT[14C]O3+ + HCO3-
FeHCO3+ + T[14C]O2[18O]- = FeT[14C]O2[18O]+ + HCO3-
FeHCO3+ + T[14C]O[18O]O- = FeT[14C]O[18O]O+ + HCO3-
FeHCO3+ + T[14C]O[18O]2- = FeT[14C]O[18O]2+ + HCO3-
FeHCO3+ + T[14C][18O]O2- = FeT[14C][18O]O2+ + HCO3-
FeHCO3+ + T[14C][18O]O[18O]- = FeT[14C][18O]O[18O]+ + HCO3-
FeHCO3+ + T[14C][18O]2O- = FeT[14C][18O]2O+ + HCO3-
FeHCO3+ + T[14C][18O]3- = FeT[14C][18O]3+ + HCO3-
#
# Added FeSO4 reactions 16Dec09
#
FeSO4 + [34S]O4-2 = Fe[34S]O4 + SO4-2
#
# Added FeHSO4+ reactions 16Dec09
#
FeHSO4+ + DSO4- = FeDSO4+ + HSO4-
FeHSO4+ + TSO4- = FeTSO4+ + HSO4-
FeHSO4+ + H[34S]O4- = FeH[34S]O4+ + HSO4-
FeHSO4+ + D[34S]O4- = FeD[34S]O4+ + HSO4-
FeHSO4+ + T[34S]O4- = FeT[34S]O4+ + HSO4-
#
# Added Fe(HS)2 reactions 16Dec09
# Revised 17Dec09, limited the number of species
#
Fe(HS)2 + HS- + DS- = FeHSDS + 2HS-
Fe(HS)2 + HS- + TS- = FeHSTS + 2HS-
Fe(HS)2 + HS- + H[34S]- = FeHSH[34S] + 2HS-
Fe(HS)2 + HS- + D[34S]- = FeHSD[34S] + 2HS-
Fe(HS)2 + HS- + T[34S]- = FeHST[34S] + 2HS-
Fe(HS)2 + DS- + HS- = FeDSHS + 2HS-
Fe(HS)2 + DS- + DS- = FeDSDS + 2HS-
Fe(HS)2 + DS- + TS- = FeDSTS + 2HS-
Fe(HS)2 + DS- + H[34S]- = FeDSH[34S] + 2HS-
Fe(HS)2 + DS- + D[34S]- = FeDSD[34S] + 2HS-
Fe(HS)2 + DS- + T[34S]- = FeDST[34S] + 2HS-
Fe(HS)2 + TS- + HS- = FeTSHS + 2HS-
Fe(HS)2 + TS- + DS- = FeTSDS + 2HS-
Fe(HS)2 + TS- + H[34S]- = FeTSH[34S] + 2HS-
Fe(HS)2 + TS- + D[34S]- = FeTSD[34S] + 2HS-
Fe(HS)2 + H[34S]- + HS- = FeH[34S]HS + 2HS-
Fe(HS)2 + H[34S]- + DS- = FeH[34S]DS + 2HS-
Fe(HS)2 + H[34S]- + TS- = FeH[34S]TS + 2HS-
Fe(HS)2 + H[34S]- + H[34S]- = FeH[34S]H[34S] + 2HS-
Fe(HS)2 + H[34S]- + D[34S]- = FeH[34S]D[34S] + 2HS-
Fe(HS)2 + H[34S]- + T[34S]- = FeH[34S]T[34S] + 2HS-
Fe(HS)2 + D[34S]- + HS- = FeD[34S]HS + 2HS-
Fe(HS)2 + D[34S]- + DS- = FeD[34S]DS + 2HS-
Fe(HS)2 + D[34S]- + TS- = FeD[34S]TS + 2HS-
Fe(HS)2 + D[34S]- + H[34S]- = FeD[34S]H[34S] + 2HS-
Fe(HS)2 + D[34S]- + D[34S]- = FeD[34S]D[34S] + 2HS-
Fe(HS)2 + D[34S]- + T[34S]- = FeD[34S]T[34S] + 2HS-
Fe(HS)2 + T[34S]- + HS- = FeT[34S]HS + 2HS-
Fe(HS)2 + T[34S]- + DS- = FeT[34S]DS + 2HS-
Fe(HS)2 + T[34S]- + H[34S]- = FeT[34S]H[34S] + 2HS-
Fe(HS)2 + T[34S]- + D[34S]- = FeT[34S]D[34S] + 2HS-
#
# Added Fe(HS)3- reactions 16Dec09
# Revised 17Dec09, limited the number of species
#
Fe(HS)3- + HS- + HS- + DS- = FeHSHSDS- + 3HS-
Fe(HS)3- + HS- + HS- + TS- = FeHSHSTS- + 3HS-
Fe(HS)3- + HS- + HS- + H[34S]- = FeHSHSH[34S]- + 3HS-
Fe(HS)3- + HS- + HS- + D[34S]- = FeHSHSD[34S]- + 3HS-
Fe(HS)3- + HS- + HS- + T[34S]- = FeHSHST[34S]- + 3HS-
Fe(HS)3- + HS- + DS- + HS- = FeHSDSHS- + 3HS-
Fe(HS)3- + HS- + DS- + DS- = FeHSDSDS- + 3HS-
Fe(HS)3- + HS- + DS- + TS- = FeHSDSTS- + 3HS-
Fe(HS)3- + HS- + DS- + H[34S]- = FeHSDSH[34S]- + 3HS-
Fe(HS)3- + HS- + DS- + D[34S]- = FeHSDSD[34S]- + 3HS-
Fe(HS)3- + HS- + DS- + T[34S]- = FeHSDST[34S]- + 3HS-
Fe(HS)3- + HS- + TS- + HS- = FeHSTSHS- + 3HS-
Fe(HS)3- + HS- + TS- + DS- = FeHSTSDS- + 3HS-
Fe(HS)3- + HS- + TS- + H[34S]- = FeHSTSH[34S]- + 3HS-
Fe(HS)3- + HS- + TS- + D[34S]- = FeHSTSD[34S]- + 3HS-
Fe(HS)3- + HS- + H[34S]- + HS- = FeHSH[34S]HS- + 3HS-
Fe(HS)3- + HS- + H[34S]- + DS- = FeHSH[34S]DS- + 3HS-
Fe(HS)3- + HS- + H[34S]- + TS- = FeHSH[34S]TS- + 3HS-
Fe(HS)3- + HS- + H[34S]- + H[34S]- = FeHSH[34S]H[34S]- + 3HS-
Fe(HS)3- + HS- + H[34S]- + D[34S]- = FeHSH[34S]D[34S]- + 3HS-
Fe(HS)3- + HS- + H[34S]- + T[34S]- = FeHSH[34S]T[34S]- + 3HS-
Fe(HS)3- + HS- + D[34S]- + HS- = FeHSD[34S]HS- + 3HS-
Fe(HS)3- + HS- + D[34S]- + DS- = FeHSD[34S]DS- + 3HS-
```

```
Fe(HS)3- + HS- + D[34S]- + TS- = FeHSD[34S]TS- + 3HS-
Fe(HS)3- + HS- + D[34S]- + H[34S]- = FeHSD[34S]H[34S]- + 3HS-
Fe(HS)3- + HS- + D[34S]- + D[34S]- = FeHSD[34S]D[34S]- + 3HS-
Fe(HS)3- + HS- + D[34S]- + T[34S]- = FeHSD[34S]T[34S]- + 3HS-
Fe(HS)3- + HS- + T[34S]- + HS- = FeHST[34S]HS- + 3HS-
Fe(HS)3- + HS- + T[34S]- + DS- = FeHST[34S]DS- + 3HS-
Fe(HS)3- + HS- + T[34S]- + H[34S]- = FeHST[34S]H[34S]- + 3HS-
Fe(HS)3- + HS- + T[34S]- + D[34S]- = FeHST[34S]D[34S]- + 3HS-
Fe(HS)3- + DS- + HS- + HS- = FeDSHSHS- + 3HS-
Fe(HS)3- + DS- + HS- + DS- = FeDSHSDS- + 3HS-
Fe(HS)3- + DS- + HS- + TS- = FeDSHSTS- + 3HS-
Fe(HS)3- + DS- + HS- + H[34S]- = FeDSHSH[34S]- + 3HS-
Fe(HS)3- + DS- + HS- + D[34S]- = FeDSHSD[34S]- + 3HS-
Fe(HS)3- + DS- + HS- + T[34S]- = FeDSHST[34S]- + 3HS-
Fe(HS)3- + DS- + DS- + HS- = FeDSDSHS- + 3HS-
Fe(HS)3- + DS- + DS- + TS- = FeDSDSTS- + 3HS-
Fe(HS)3- + DS- + DS- + H[34S]- = FeDSDSH[34S]- + 3HS-
Fe(HS)3- + DS- + DS- + T[34S]- = FeDSDST[34S]- + 3HS-
Fe(HS)3- + DS- + TS- + HS- = FeDSTSHS- + 3HS-
Fe(HS)3- + DS- + TS- + DS- = FeDSTSDS- + 3HS-
Fe(HS)3- + DS- + TS- + H[34S]- = FeDSTSH[34S]- + 3HS-
Fe(HS)3- + DS- + TS- + D[34S]- = FeDSTSD[34S]- + 3HS-
Fe(HS)3- + DS- + H[34S]- + HS- = FeDSH[34S]HS- + 3HS-
Fe(HS)3- + DS- + H[34S]- + DS- = FeDSH[34S]DS- + 3HS-
Fe(HS)3- + DS- + H[34S]- + TS- = FeDSH[34S]TS- + 3HS-
Fe(HS)3- + DS- + H[34S]- + H[34S]- = FeDSH[34S]H[34S]- + 3HS-
Fe(HS)3- + DS- + H[34S]- + D[34S]- = FeDSH[34S]D[34S]- + 3HS-
Fe(HS)3- + DS- + H[34S]- + T[34S]- = FeDSH[34S]T[34S]- + 3HS-
Fe(HS)3- + DS- + D[34S]- + HS- = FeDSD[34S]HS- + 3HS-
Fe(HS)3- + DS- + D[34S]- + TS- = FeDSD[34S]TS- + 3HS-
Fe(HS)3- + DS- + D[34S]- + H[34S]- = FeDSD[34S]H[34S]- + 3HS-
Fe(HS)3- + DS- + T[34S]- + HS- = FeDST[34S]HS- + 3HS-
Fe(HS)3- + DS- + T[34S]- + DS- = FeDST[34S]DS- + 3HS-
Fe(HS)3- + DS- + T[34S]- + H[34S]- = FeDST[34S]H[34S]- + 3HS-
Fe(HS)3- + TS- + HS- + HS- = FeTSHSHS- + 3HS-
Fe(HS)3- + TS- + HS- + DS- = FeTSHSDS- + 3HS-
Fe(HS)3- + TS- + HS- + H[34S]- = FeTSHSH[34S]- + 3HS-
Fe(HS)3- + TS- + HS- + D[34S]- = FeTSHSD[34S]- + 3HS-
Fe(HS)3- + TS- + DS- + HS- = FeTSDSHS- + 3HS-
Fe(HS)3- + TS- + DS- + DS- = FeTSDSDS- + 3HS-
Fe(HS)3- + TS- + DS- + H[34S]- = FeTSDSH[34S]- + 3HS-
Fe(HS)3- + TS- + DS- + D[34S]- = FeTSDSD[34S]- + 3HS-
Fe(HS)3- + TS- + H[34S]- + HS- = FeTSH[34S]HS- + 3HS-
Fe(HS)3- + TS- + H[34S]- + DS- = FeTSH[34S]DS- + 3HS-
Fe(HS)3- + TS- + H[34S]- + H[34S]- = FeTSH[34S]H[34S]- + 3HS-
Fe(HS)3- + TS- + H[34S]- + D[34S]- = FeTSH[34S]D[34S]- + 3HS-
Fe(HS)3- + TS- + D[34S]- + HS- = FeTSD[34S]HS- + 3HS-
Fe(HS)3- + TS- + D[34S]- + DS- = FeTSD[34S]DS- + 3HS-
Fe(HS)3- + TS- + D[34S]- + H[34S]- = FeTSD[34S]H[34S]- + 3HS-
Fe(HS)3- + H[34S]- + HS- + HS- = FeH[34S]HSHS- + 3HS-
Fe(HS)3- + H[34S]- + HS- + DS- = FeH[34S]HSDS- + 3HS-
Fe(HS)3- + H[34S]- + HS- + TS- = FeH[34S]HSTS- + 3HS-
Fe(HS)3- + H[34S]- + HS- + H[34S]- = FeH[34S]HSH[34S]- + 3HS-
Fe(HS)3- + H[34S]- + HS- + D[34S]- = FeH[34S]HSD[34S]- + 3HS-
Fe(HS)3- + H[34S]- + HS- + T[34S]- = FeH[34S]HST[34S]- + 3HS-
Fe(HS)3- + H[34S]- + DS- + HS- = FeH[34S]DSHS- + 3HS-
Fe(HS)3- + H[34S]- + DS- + DS- = FeH[34S]DSDS- + 3HS-
Fe(HS)3- + H[34S]- + DS- + TS- = FeH[34S]DSTS- + 3HS-
Fe(HS)3- + H[34S]- + DS- + H[34S]- = FeH[34S]DSH[34S]- + 3HS-
Fe(HS)3- + H[34S]- + DS- + D[34S]- = FeH[34S]DSD[34S]- + 3HS-
Fe(HS)3- + H[34S]- + DS- + T[34S]- = FeH[34S]DST[34S]- + 3HS-
Fe(HS)3- + H[34S]- + TS- + HS- = FeH[34S]TSHS- + 3HS-
Fe(HS)3- + H[34S]- + TS- + DS- = FeH[34S]TSDS- + 3HS-
Fe(HS)3- + H[34S]- + TS- + H[34S]- = FeH[34S]TSH[34S]- + 3HS-
Fe(HS)3- + H[34S]- + TS- + D[34S]- = FeH[34S]TSD[34S]- + 3HS-
Fe(HS)3- + H[34S]- + H[34S]- + HS- = FeH[34S]H[34S]HS- + 3HS-
Fe(HS)3- + H[34S]- + H[34S]- + DS- = FeH[34S]H[34S]DS- + 3HS-
Fe(HS)3- + H[34S]- + H[34S]- + TS- = FeH[34S]H[34S]TS- + 3HS-
Fe(HS)3- + H[34S]- + H[34S]- + H[34S]- = FeH[34S]H[34S]H[34S]- + 3HS-
Fe(HS)3- + H[34S]- + H[34S]- + D[34S]- = FeH[34S]H[34S]D[34S]- + 3HS-
Fe(HS)3- + H[34S]- + H[34S]- + T[34S]- = FeH[34S]H[34S]T[34S]- + 3HS-
Fe(HS)3- + H[34S]- + D[34S]- + HS- = FeH[34S]D[34S]HS- + 3HS-
Fe(HS)3- + H[34S]- + D[34S]- + DS- = FeH[34S]D[34S]DS- + 3HS-
Fe(HS)3- + H[34S]- + D[34S]- + TS- = FeH[34S]D[34S]TS- + 3HS-
Fe(HS)3- + H[34S]- + D[34S]- + H[34S]- = FeH[34S]D[34S]H[34S]- + 3HS-
Fe(HS)3- + H[34S]- + T[34S]- + HS- = FeH[34S]T[34S]HS- + 3HS-
Fe(HS)3- + H[34S]- + T[34S]- + DS- = FeH[34S]T[34S]DS- + 3HS-
Fe(HS)3- + H[34S]- + T[34S]- + H[34S]- = FeH[34S]T[34S]H[34S]- + 3HS-
Fe(HS)3- + D[34S]- + HS- + HS- = FeD[34S]HSHS- + 3HS-
Fe(HS)3- + D[34S]- + HS- + DS- = FeD[34S]HSDS- + 3HS-
Fe(HS)3- + D[34S]- + HS- + TS- = FeD[34S]HSTS- + 3HS-
Fe(HS)3- + D[34S]- + HS- + H[34S]- = FeD[34S]HSH[34S]- + 3HS-
Fe(HS)3- + D[34S]- + HS- + D[34S]- = FeD[34S]HSD[34S]- + 3HS-
Fe(HS)3- + D[34S]- + HS- + T[34S]- = FeD[34S]HST[34S]- + 3HS-
Fe(HS)3- + D[34S]- + DS- + HS- = FeD[34S]DSHS- + 3HS-
Fe(HS)3- + D[34S]- + DS- + TS- = FeD[34S]DSTS- + 3HS-
Fe(HS)3- + D[34S]- + DS- + H[34S]- = FeD[34S]DSH[34S]- + 3HS-
Fe(HS)3- + D[34S]- + TS- + HS- = FeD[34S]TSHS- + 3HS-
Fe(HS)3- + D[34S]- + TS- + DS- = FeD[34S]TSDS- + 3HS-
Fe(HS)3- + D[34S]- + TS- + H[34S]- = FeD[34S]TSH[34S]- + 3HS-
Fe(HS)3- + D[34S]- + H[34S]- + HS- = FeD[34S]H[34S]HS- + 3HS-
Fe(HS)3- + D[34S]- + H[34S]- + DS- = FeD[34S]H[34S]DS- + 3HS-
Fe(HS)3- + D[34S]- + H[34S]- + TS- = FeD[34S]H[34S]TS- + 3HS-
Fe(HS)3- + D[34S]- + H[34S]- + H[34S]- = FeD[34S]H[34S]H[34S]- + 3HS-
Fe(HS)3- + D[34S]- + D[34S]- + HS- = FeD[34S]D[34S]HS- + 3HS-
Fe(HS)3- + D[34S]- + T[34S]- + HS- = FeD[34S]T[34S]HS- + 3HS-
Fe(HS)3- + T[34S]- + HS- + HS- = FeT[34S]HSHS- + 3HS-
Fe(HS)3- + T[34S]- + HS- + DS- = FeT[34S]HSDS- + 3HS-
Fe(HS)3- + T[34S]- + HS- + H[34S]- = FeT[34S]HSH[34S]- + 3HS-
Fe(HS)3- + T[34S]- + HS- + D[34S]- = FeT[34S]HSD[34S]- + 3HS-
Fe(HS)3- + T[34S]- + DS- + HS- = FeT[34S]DSHS- + 3HS-
Fe(HS)3- + T[34S]- + DS- + DS- = FeT[34S]DSDS- + 3HS-
Fe(HS)3- + T[34S]- + DS- + H[34S]- = FeT[34S]DSH[34S]- + 3HS-
Fe(HS)3- + T[34S]- + H[34S]- + HS- = FeT[34S]H[34S]HS- + 3HS-
Fe(HS)3- + T[34S]- + H[34S]- + DS- = FeT[34S]H[34S]DS- + 3HS-
Fe(HS)3- + T[34S]- + H[34S]- + H[34S]- = FeT[34S]H[34S]H[34S]- + 3HS-
Fe(HS)3- + T[34S]- + D[34S]- + HS- = FeT[34S]D[34S]HS- + 3HS-
#
# Missing FeHPO4 reactions
#
#
# Missing FeH2PO4+ reactions
#
#
# Added FeOH+2 reactions 16Dec09
#
FeOH+2 + OD- = FeOD+2 + OH-
```

```
FeOH+2 + OT- = FeOT+2 + OH-
FeOH+2 + [18O]H- = Fe[18O]H+2 + OH-
FeOH+2 + [18O]D- = Fe[18O]D+2 + OH-
FeOH+2 + [18O]T- = Fe[18O]T+2 + OH-
#
# Added Fe(OH)2+ reactions 16Dec09
# Revised 17Dec09, limited the number of species
#
Fe(OH)2+ + OH- + OD- = FeOHOD+ + 2OH-
Fe(OH)2+ + OH- + OT- = FeOHOT+ + 2OH-
Fe(OH)2+ + OH- + [18O]H- = FeOH[18O]H+ + 2OH-
Fe(OH)2+ + OH- + [18O]D- = FeOH[18O]D+ + 2OH-
Fe(OH)2+ + OH- + [18O]T- = FeOH[18O]T+ + 2OH-
Fe(OH)2+ + OD- + OH- = FeODOH+ + 2OH-
Fe(OH)2+ + OD- + OD- = FeODOD+ + 2OH-
Fe(OH)2+ + OD- + OT- = FeODOT+ + 2OH-
Fe(OH)2+ + OD- + [18O]H- = FeOD[18O]H+ + 2OH-
Fe(OH)2+ + OD- + [18O]D- = FeOD[18O]D+ + 2OH-
Fe(OH)2+ + OD- + [18O]T- = FeOD[18O]T+ + 2OH-
Fe(OH)2+ + OT- + OH- = FeOTOH+ + 2OH-
Fe(OH)2+ + OT- + OD- = FeOTOD+ + 2OH-
Fe(OH)2+ + OT- + [18O]H- = FeOT[18O]H+ + 2OH-
Fe(OH)2+ + OT- + [18O]D- = FeOT[18O]D+ + 2OH-
Fe(OH)2+ + [18O]H- + OH- = Fe[18O]HOH+ + 2OH-
Fe(OH)2+ + [18O]H- + OD- = Fe[18O]HOD+ + 2OH-
Fe(OH)2+ + [18O]H- + OT- = Fe[18O]HOT+ + 2OH-
Fe(OH)2+ + [18O]H- + [18O]H- = Fe[18O]H[18O]H+ + 2OH-
Fe(OH)2+ + [18O]H- + [18O]D- = Fe[18O]H[18O]D+ + 2OH-
Fe(OH)2+ + [18O]H- + [18O]T- = Fe[18O]H[18O]T+ + 2OH-
Fe(OH)2+ + [18O]D- + OH- = Fe[18O]DOH+ + 2OH-
Fe(OH)2+ + [18O]D- + OD- = Fe[18O]DOD+ + 2OH-
Fe(OH)2+ + [18O]D- + OT- = Fe[18O]DOT+ + 2OH-
Fe(OH)2+ + [18O]D- + [18O]H- = Fe[18O]D[18O]H+ + 2OH-
Fe(OH)2+ + [18O]D- + [18O]D- = Fe[18O]D[18O]D+ + 2OH-
Fe(OH)2+ + [18O]D- + [18O]T- = Fe[18O]D[18O]T+ + 2OH-
Fe(OH)2+ + [18O]T- + OH- = Fe[18O]TOH+ + 2OH-
Fe(OH)2+ + [18O]T- + OD- = Fe[18O]TOD+ + 2OH-
Fe(OH)2+ + [18O]T- + [18O]H- = Fe[18O]T[18O]H+ + 2OH-
Fe(OH)2+ + [18O]T- + [18O]D- = Fe[18O]T[18O]D+ + 2OH-
#
# Added Fe(OH)3 reactions 16Dec09
# Revised 17Dec09, limited the number of species
#
Fe(OH)3 + OH- + OH- + OD- = FeOHOHOD + 3OH-
Fe(OH)3 + OH- + OH- + OT- = FeOHOHOT + 3OH-
Fe(OH)3 + OH- + OH- + [18O]H- = FeOHOH[18O]H + 3OH-
Fe(OH)3 + OH- + OH- + [18O]D- = FeOHOH[18O]D + 3OH-
Fe(OH)3 + OH- + OH- + [18O]T- = FeOHOH[18O]T + 3OH-
Fe(OH)3 + OH- + OD- + OH- = FeOHODOH + 3OH-
Fe(OH)3 + OH- + OD- + OD- = FeOHODOD + 3OH-
Fe(OH)3 + OH- + OD- + OT- = FeOHODOT + 3OH-
Fe(OH)3 + OH- + OD- + [18O]H- = FeOHOD[18O]H + 3OH-
Fe(OH)3 + OH- + OD- + [18O]D- = FeOHOD[18O]D + 3OH-
Fe(OH)3 + OH- + OD- + [18O]T- = FeOHOD[18O]T + 3OH-
Fe(OH)3 + OH- + OT- + OH- = FeOHOTOH + 3OH-
Fe(OH)3 + OH- + OT- + OD- = FeOHOTOD + 3OH-
Fe(OH)3 + OH- + OT- + [18O]H- = FeOHOT[18O]H + 3OH-
Fe(OH)3 + OH- + OT- + [18O]D- = FeOHOT[18O]D + 3OH-
Fe(OH)3 + OH- + [18O]H- + OH- = FeOH[18O]HOH + 3OH-
Fe(OH)3 + OH- + [18O]H- + OD- = FeOH[18O]HOD + 3OH-
Fe(OH)3 + OH- + [18O]H- + OT- = FeOH[18O]HOT + 3OH-
Fe(OH)3 + OH- + [18O]H- + [18O]H- = FeOH[18O]H[18O]H + 3OH-
Fe(OH)3 + OH- + [18O]H- + [18O]D- = FeOH[18O]H[18O]D + 3OH-
Fe(OH)3 + OH- + [18O]H- + [18O]T- = FeOH[18O]H[18O]T + 3OH-
Fe(OH)3 + OH- + [18O]D- + OH- = FeOH[18O]DOH + 3OH-
Fe(OH)3 + OH- + [18O]D- + OD- = FeOH[18O]DOD + 3OH-
Fe(OH)3 + OH- + [18O]D- + OT- = FeOH[18O]DOT + 3OH-
Fe(OH)3 + OH- + [18O]D- + [18O]H- = FeOH[18O]D[18O]H + 3OH-
Fe(OH)3 + OH- + [18O]D- + [18O]D- = FeOH[18O]D[18O]D + 3OH-
Fe(OH)3 + OH- + [18O]D- + [18O]T- = FeOH[18O]D[18O]T + 3OH-
Fe(OH)3 + OH- + [18O]T- + OH- = FeOH[18O]TOH + 3OH-
Fe(OH)3 + OH- + [18O]T- + OD- = FeOH[18O]TOD + 3OH-
Fe(OH)3 + OH- + [18O]T- + [18O]H- = FeOH[18O]T[18O]H + 3OH-
Fe(OH)3 + OH- + [18O]T- + [18O]D- = FeOH[18O]T[18O]D + 3OH-
Fe(OH)3 + OD- + OH- + OH- = FeODOHOH + 3OH-
Fe(OH)3 + OD- + OH- + OD- = FeODOHOD + 3OH-
Fe(OH)3 + OD- + OH- + OT- = FeODOHOT + 3OH-
Fe(OH)3 + OD- + OH- + [18O]H- = FeODOH[18O]H + 3OH-
Fe(OH)3 + OD- + OH- + [18O]D- = FeODOH[18O]D + 3OH-
Fe(OH)3 + OD- + OH- + [18O]T- = FeODOH[18O]T + 3OH-
Fe(OH)3 + OD- + OD- + OH- = FeODODOH + 3OH-
Fe(OH)3 + OD- + OD- + OT- = FeODODOT + 3OH-
Fe(OH)3 + OD- + OD- + [18O]H- = FeODOD[18O]H + 3OH-
Fe(OH)3 + OD- + OD- + [18O]T- = FeODOD[18O]T + 3OH-
Fe(OH)3 + OD- + OT- + OH- = FeODOTOH + 3OH-
Fe(OH)3 + OD- + OT- + OD- = FeODOTOD + 3OH-
Fe(OH)3 + OD- + OT- + [18O]H- = FeODOT[18O]H + 3OH-
Fe(OH)3 + OD- + OT- + [18O]D- = FeODOT[18O]D + 3OH-
Fe(OH)3 + OD- + [18O]H- + OH- = FeOD[18O]HOH + 3OH-
Fe(OH)3 + OD- + [18O]H- + OD- = FeOD[18O]HOD + 3OH-
Fe(OH)3 + OD- + [18O]H- + OT- = FeOD[18O]HOT + 3OH-
Fe(OH)3 + OD- + [18O]H- + [18O]H- = FeOD[18O]H[18O]H + 3OH-
Fe(OH)3 + OD- + [18O]H- + [18O]D- = FeOD[18O]H[18O]D + 3OH-
Fe(OH)3 + OD- + [18O]H- + [18O]T- = FeOD[18O]H[18O]T + 3OH-
Fe(OH)3 + OD- + [18O]D- + OH- = FeOD[18O]DOH + 3OH-
Fe(OH)3 + OD- + [18O]D- + OT- = FeOD[18O]DOT + 3OH-
Fe(OH)3 + OD- + [18O]D- + [18O]H- = FeOD[18O]D[18O]H + 3OH-
Fe(OH)3 + OD- + [18O]T- + OH- = FeOD[18O]TOH + 3OH-
Fe(OH)3 + OD- + [18O]T- + OD- = FeOD[18O]TOD + 3OH-
Fe(OH)3 + OD- + [18O]T- + [18O]H- = FeOD[18O]T[18O]H + 3OH-
Fe(OH)3 + OT- + OH- + OH- = FeOTOHOH + 3OH-
Fe(OH)3 + OT- + OH- + OD- = FeOTOHOD + 3OH-
Fe(OH)3 + OT- + OH- + [18O]H- = FeOTOH[18O]H + 3OH-
Fe(OH)3 + OT- + OH- + [18O]D- = FeOTOH[18O]D + 3OH-
Fe(OH)3 + OT- + OD- + OH- = FeOTODOH + 3OH-
Fe(OH)3 + OT- + OD- + OD- = FeOTODOD + 3OH-
Fe(OH)3 + OT- + OD- + [18O]H- = FeOTOD[18O]H + 3OH-
Fe(OH)3 + OT- + OD- + [18O]D- = FeOTOD[18O]D + 3OH-
Fe(OH)3 + OT- + [18O]H- + OH- = FeOT[18O]HOH + 3OH-
Fe(OH)3 + OT- + [18O]H- + OD- = FeOT[18O]HOD + 3OH-
Fe(OH)3 + OT- + [18O]H- + [18O]H- = FeOT[18O]H[18O]H + 3OH-
Fe(OH)3 + OT- + [18O]H- + [18O]D- = FeOT[18O]H[18O]D + 3OH-
Fe(OH)3 + OT- + [18O]D- + OH- = FeOT[18O]DOH + 3OH-
Fe(OH)3 + OT- + [18O]D- + OD- = FeOT[18O]DOD + 3OH-
Fe(OH)3 + OT- + [18O]D- + [18O]H- = FeOT[18O]D[18O]H + 3OH-
Fe(OH)3 + [18O]H- + OH- + OH- = Fe[18O]HOHOH + 3OH-
```

```
Fe(OH)3 + [18O]H- + OH- + OD- = Fe[18O]HOHOD + 3OH-
Fe(OH)3 + [18O]H- + OH- + OT- = Fe[18O]HOHOT + 3OH-
Fe(OH)3 + [18O]H- + OH- + [18O]H- = Fe[18O]HOH[18O]H + 3OH-
Fe(OH)3 + [18O]H- + OH- + [18O]D- = Fe[18O]HOH[18O]D + 3OH-
Fe(OH)3 + [18O]H- + OH- + [18O]T- = Fe[18O]HOH[18O]T + 3OH-
Fe(OH)3 + [18O]H- + OD- + OH- = Fe[18O]HODOH + 3OH-
Fe(OH)3 + [18O]H- + OD- + OD- = Fe[18O]HODOD + 3OH-
Fe(OH)3 + [18O]H- + OD- + OT- = Fe[18O]HODOT + 3OH-
Fe(OH)3 + [18O]H- + OD- + [18O]H- = Fe[18O]HOD[18O]H + 3OH-
Fe(OH)3 + [18O]H- + OD- + [18O]D- = Fe[18O]HOD[18O]D + 3OH-
Fe(OH)3 + [18O]H- + OD- + [18O]T- = Fe[18O]HOD[18O]T + 3OH-
Fe(OH)3 + [18O]H- + OT- + OH- = Fe[18O]HOTOH + 3OH-
Fe(OH)3 + [18O]H- + OT- + OD- = Fe[18O]HOTOD + 3OH-
Fe(OH)3 + [18O]H- + OT- + [18O]H- = Fe[18O]HOT[18O]H + 3OH-
Fe(OH)3 + [18O]H- + OT- + [18O]D- = Fe[18O]HOT[18O]D + 3OH-
Fe(OH)3 + [18O]H- + [18O]H- + OH- = Fe[18O]H[18O]HOH + 3OH-
Fe(OH)3 + [18O]H- + [18O]H- + OD- = Fe[18O]H[18O]HOD + 3OH-
Fe(OH)3 + [18O]H- + [18O]H- + OT- = Fe[18O]H[18O]HOT + 3OH-
Fe(OH)3 + [18O]H- + [18O]D- + OH- = Fe[18O]H[18O]DOH + 3OH-
Fe(OH)3 + [18O]H- + [18O]D- + OD- = Fe[18O]H[18O]DOD + 3OH-
Fe(OH)3 + [18O]H- + [18O]D- + OT- = Fe[18O]H[18O]DOT + 3OH-
Fe(OH)3 + [18O]H- + [18O]T- + OH- = Fe[18O]H[18O]TOH + 3OH-
Fe(OH)3 + [18O]H- + [18O]T- + OD- = Fe[18O]H[18O]TOD + 3OH-
Fe(OH)3 + [18O]D- + OH- + OH- = Fe[18O]DOHOH + 3OH-
Fe(OH)3 + [18O]D- + OH- + OD- = Fe[18O]DOHOD + 3OH-
Fe(OH)3 + [18O]D- + OH- + OT- = Fe[18O]DOHOT + 3OH-
Fe(OH)3 + [18O]D- + OH- + [18O]H- = Fe[18O]DOH[18O]H + 3OH-
Fe(OH)3 + [18O]D- + OH- + [18O]D- = Fe[18O]DOH[18O]D + 3OH-
Fe(OH)3 + [18O]D- + OH- + [18O]T- = Fe[18O]DOH[18O]T + 3OH-
Fe(OH)3 + [18O]D- + OD- + OH- = Fe[18O]DODOH + 3OH-
Fe(OH)3 + [18O]D- + OD- + OT- = Fe[18O]DODOT + 3OH-
Fe(OH)3 + [18O]D- + OD- + [18O]H- = Fe[18O]DOD[18O]H + 3OH-
Fe(OH)3 + [18O]D- + OT- + OH- = Fe[18O]DOTOH + 3OH-
Fe(OH)3 + [18O]D- + OT- + OD- = Fe[18O]DOTOD + 3OH-
Fe(OH)3 + [18O]D- + OT- + [18O]H- = Fe[18O]DOT[18O]H + 3OH-
Fe(OH)3 + [18O]D- + [18O]H- + OH- = Fe[18O]D[18O]HOH + 3OH-
Fe(OH)3 + [18O]D- + [18O]H- + OD- = Fe[18O]D[18O]HOD + 3OH-
Fe(OH)3 + [18O]D- + [18O]H- + OT- = Fe[18O]D[18O]HOT + 3OH-
Fe(OH)3 + [18O]D- + [18O]D- + OH- = Fe[18O]D[18O]DOH + 3OH-
Fe(OH)3 + [18O]D- + [18O]T- + OH- = Fe[18O]D[18O]TOH + 3OH-
Fe(OH)3 + [18O]T- + OH- + OH- = Fe[18O]TOHOH + 3OH-
Fe(OH)3 + [18O]T- + OH- + OD- = Fe[18O]TOHOD + 3OH-
Fe(OH)3 + [18O]T- + OH- + [18O]H- = Fe[18O]TOH[18O]H + 3OH-
Fe(OH)3 + [18O]T- + OH- + [18O]D- = Fe[18O]TOH[18O]D + 3OH-
Fe(OH)3 + [18O]T- + OD- + OH- = Fe[18O]TODOH + 3OH-
Fe(OH)3 + [18O]T- + OD- + OD- = Fe[18O]TODOD + 3OH-
Fe(OH)3 + [18O]T- + OD- + [18O]H- = Fe[18O]TOD[18O]H + 3OH-
Fe(OH)3 + [18O]T- + [18O]H- + OH- = Fe[18O]T[18O]HOH + 3OH-
Fe(OH)3 + [18O]T- + [18O]H- + OD- = Fe[18O]T[18O]HOD + 3OH-
Fe(OH)3 + [18O]T- + [18O]D- + OH- = Fe[18O]T[18O]DOH + 3OH-
#
# Added Fe(OH)4- reactions 16Dec09
# Revised 17Dec09, limited the number of species
#
Fe(OH)4- + OH- + OH- + OH- + OD- = FeOHOHOHOD- + 4OH-
Fe(OH)4- + OH- + OH- + OH- + OT- = FeOHOHOHOT- + 4OH-
Fe(OH)4- + OH- + OH- + OH- + [18O]H- = FeOHOHOH[18O]H- + 4OH-
Fe(OH)4- + OH- + OH- + OH- + [18O]D- = FeOHOHOH[18O]D- + 4OH-
Fe(OH)4- + OH- + OH- + OH- + [18O]T- = FeOHOHOH[18O]T- + 4OH-
Fe(OH)4- + OH- + OH- + OD- + OH- = FeOHOHODOH- + 4OH-
Fe(OH)4- + OH- + OH- + OD- + OD- = FeOHOHODOD- + 4OH-
Fe(OH)4- + OH- + OH- + OD- + OT- = FeOHOHODOT- + 4OH-
Fe(OH)4- + OH- + OH- + OD- + [18O]H- = FeOHOHOD[18O]H- + 4OH-
Fe(OH)4- + OH- + OH- + OD- + [18O]D- = FeOHOHOD[18O]D- + 4OH-
Fe(OH)4- + OH- + OH- + OD- + [18O]T- = FeOHOHOD[18O]T- + 4OH-
Fe(OH)4- + OH- + OH- + OT- + OH- = FeOHOHOTOH- + 4OH-
Fe(OH)4- + OH- + OH- + OT- + OD- = FeOHOHOTOD- + 4OH-
Fe(OH)4- + OH- + OH- + OT- + [18O]H- = FeOHOHOT[18O]H- + 4OH-
Fe(OH)4- + OH- + OH- + OT- + [18O]D- = FeOHOHOT[18O]D- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]H- + OH- = FeOHOH[18O]HOH- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]H- + OD- = FeOHOH[18O]HOD- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]H- + OT- = FeOHOH[18O]HOT- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]H- + [18O]H- = FeOHOH[18O]H[18O]H- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]H- + [18O]D- = FeOHOH[18O]H[18O]D- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]H- + [18O]T- = FeOHOH[18O]H[18O]T- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]D- + OH- = FeOHOH[18O]DOH- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]D- + OD- = FeOHOH[18O]DOD- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]D- + OT- = FeOHOH[18O]DOT- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]D- + [18O]H- = FeOHOH[18O]D[18O]H- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]D- + [18O]D- = FeOHOH[18O]D[18O]D- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]D- + [18O]T- = FeOHOH[18O]D[18O]T- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]T- + OH- = FeOHOH[18O]TOH- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]T- + OD- = FeOHOH[18O]TOD- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]T- + [18O]H- = FeOHOH[18O]T[18O]H- + 4OH-
Fe(OH)4- + OH- + OH- + [18O]T- + [18O]D- = FeOHOH[18O]T[18O]D- + 4OH-
Fe(OH)4- + OH- + OD- + OH- + OH- = FeOHODOHOH- + 4OH-
Fe(OH)4- + OH- + OD- + OH- + OD- = FeOHODOHOD- + 4OH-
Fe(OH)4- + OH- + OD- + OH- + OT- = FeOHODOHOT- + 4OH-
Fe(OH)4- + OH- + OD- + OH- + [18O]H- = FeOHODOH[18O]H- + 4OH-
Fe(OH)4- + OH- + OD- + OH- + [18O]D- = FeOHODOH[18O]D- + 4OH-
Fe(OH)4- + OH- + OD- + OH- + [18O]T- = FeOHODOH[18O]T- + 4OH-
Fe(OH)4- + OH- + OD- + OD- + OH- = FeOHODODOH- + 4OH-
Fe(OH)4- + OH- + OD- + OD- + OT- = FeOHODODOT- + 4OH-
Fe(OH)4- + OH- + OD- + OD- + [18O]H- = FeOHODOD[18O]H- + 4OH-
Fe(OH)4- + OH- + OD- + OD- + [18O]T- = FeOHODOD[18O]T- + 4OH-
Fe(OH)4- + OH- + OD- + OT- + OH- = FeOHODOTOH- + 4OH-
Fe(OH)4- + OH- + OD- + OT- + OD- = FeOHODOTOD- + 4OH-
Fe(OH)4- + OH- + OD- + OT- + [18O]H- = FeOHODOT[18O]H- + 4OH-
Fe(OH)4- + OH- + OD- + OT- + [18O]D- = FeOHODOT[18O]D- + 4OH-
Fe(OH)4- + OH- + OD- + [18O]H- + OH- = FeOHOD[18O]HOH- + 4OH-
Fe(OH)4- + OH- + OD- + [18O]H- + OD- = FeOHOD[18O]HOD- + 4OH-
Fe(OH)4- + OH- + OD- + [18O]H- + OT- = FeOHOD[18O]HOT- + 4OH-
Fe(OH)4- + OH- + OD- + [18O]H- + [18O]H- = FeOHOD[18O]H[18O]H- + 4OH-
Fe(OH)4- + OH- + OD- + [18O]H- + [18O]D- = FeOHOD[18O]H[18O]D- + 4OH-
Fe(OH)4- + OH- + OD- + [18O]H- + [18O]T- = FeOHOD[18O]H[18O]T- + 4OH-
Fe(OH)4- + OH- + OD- + [18O]D- + OH- = FeOHOD[18O]DOH- + 4OH-
Fe(OH)4- + OH- + OD- + [18O]D- + OT- = FeOHOD[18O]DOT- + 4OH-
Fe(OH)4- + OH- + OD- + [18O]D- + [18O]H- = FeOHOD[18O]D[18O]H- + 4OH-
Fe(OH)4- + OH- + OD- + [18O]T- + OH- = FeOHOD[18O]TOH- + 4OH-
Fe(OH)4- + OH- + OD- + [18O]T- + OD- = FeOHOD[18O]TOD- + 4OH-
Fe(OH)4- + OH- + OD- + [18O]T- + [18O]H- = FeOHOD[18O]T[18O]H- + 4OH-
Fe(OH)4- + OH- + OT- + OH- + OH- = FeOHOTOHOH- + 4OH-
Fe(OH)4- + OH- + OT- + OH- + OD- = FeOHOTOHOD- + 4OH-
Fe(OH)4- + OH- + OT- + OH- + [18O]H- = FeOHOTOH[18O]H- + 4OH-
Fe(OH)4- + OH- + OT- + OH- + [18O]D- = FeOHOTOH[18O]D- + 4OH-
Fe(OH)4- + OH- + OT- + OD- + OH- = FeOHOTODOH- + 4OH-
```

```
Fe(OH)4- + OH- + OT- + OD- + OD- = FeOHOTODOD- + 4OH-
Fe(OH)4- + OH- + OT- + OD- + [18O]H- = FeOHOTOD[18O]H- + 4OH-
Fe(OH)4- + OH- + OT- + OD- + [18O]D- = FeOHOTOD[18O]D- + 4OH-
Fe(OH)4- + OH- + OT- + [18O]H- + OH- = FeOHOT[18O]HOH- + 4OH-
Fe(OH)4- + OH- + OT- + [18O]H- + OD- = FeOHOT[18O]HOD- + 4OH-
Fe(OH)4- + OH- + OT- + [18O]H- + [18O]H- = FeOHOT[18O]H[18O]H- + 4OH-
Fe(OH)4- + OH- + OT- + [18O]H- + [18O]D- = FeOHOT[18O]H[18O]D- + 4OH-
Fe(OH)4- + OH- + OT- + [18O]D- + OH- = FeOHOT[18O]DOH- + 4OH-
Fe(OH)4- + OH- + OT- + [18O]D- + OD- = FeOHOT[18O]DOD- + 4OH-
Fe(OH)4- + OH- + OT- + [18O]D- + [18O]H- = FeOHOT[18O]D[18O]H- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OH- + OH- = FeOH[18O]HOHOH- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OH- + OD- = FeOH[18O]HOHOD- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OH- + OT- = FeOH[18O]HOHOT- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OH- + [18O]H- = FeOH[18O]HOH[18O]H- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OH- + [18O]D- = FeOH[18O]HOH[18O]D- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OH- + [18O]T- = FeOH[18O]HOH[18O]T- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OD- + OH- = FeOH[18O]HODOH- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OD- + OD- = FeOH[18O]HODOD- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OD- + OT- = FeOH[18O]HODOT- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OD- + [18O]H- = FeOH[18O]HOD[18O]H- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OD- + [18O]D- = FeOH[18O]HOD[18O]D- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OD- + [18O]T- = FeOH[18O]HOD[18O]T- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OT- + OH- = FeOH[18O]HOTOH- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OT- + OD- = FeOH[18O]HOTOD- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OT- + [18O]H- = FeOH[18O]HOT[18O]H- + 4OH-
Fe(OH)4- + OH- + [18O]H- + OT- + [18O]D- = FeOH[18O]HOT[18O]D- + 4OH-
Fe(OH)4- + OH- + [18O]H- + [18O]H- + OH- = FeOH[18O]H[18O]HOH- + 4OH-
Fe(OH)4- + OH- + [18O]H- + [18O]H- + OD- = FeOH[18O]H[18O]HOD- + 4OH-
Fe(OH)4- + OH- + [18O]H- + [18O]H- + OT- = FeOH[18O]H[18O]HOT- + 4OH-
Fe(OH)4- + OH- + [18O]H- + [18O]D- + OH- = FeOH[18O]H[18O]DOH- + 4OH-
Fe(OH)4- + OH- + [18O]H- + [18O]D- + OD- = FeOH[18O]H[18O]DOD- + 4OH-
Fe(OH)4- + OH- + [18O]H- + [18O]D- + OT- = FeOH[18O]H[18O]DOT- + 4OH-
Fe(OH)4- + OH- + [18O]H- + [18O]T- + OH- = FeOH[18O]H[18O]TOH- + 4OH-
Fe(OH)4- + OH- + [18O]H- + [18O]T- + OD- = FeOH[18O]H[18O]TOD- + 4OH-
Fe(OH)4- + OH- + [18O]D- + OH- + OH- = FeOH[18O]DOHOH- + 4OH-
Fe(OH)4- + OH- + [18O]D- + OH- + OD- = FeOH[18O]DOHOD- + 4OH-
Fe(OH)4- + OH- + [18O]D- + OH- + OT- = FeOH[18O]DOHOT- + 4OH-
Fe(OH)4- + OH- + [18O]D- + OH- + [18O]H- = FeOH[18O]DOH[18O]H- + 4OH-
Fe(OH)4- + OH- + [18O]D- + OH- + [18O]D- = FeOH[18O]DOH[18O]D- + 4OH-
Fe(OH)4- + OH- + [18O]D- + OH- + [18O]T- = FeOH[18O]DOH[18O]T- + 4OH-
Fe(OH)4- + OH- + [18O]D- + OD- + OH- = FeOH[18O]DODOH- + 4OH-
Fe(OH)4- + OH- + [18O]D- + OD- + OT- = FeOH[18O]DODOT- + 4OH-
Fe(OH)4- + OH- + [18O]D- + OD- + [18O]H- = FeOH[18O]DOD[18O]H- + 4OH-
Fe(OH)4- + OH- + [18O]D- + OT- + OH- = FeOH[18O]DOTOH- + 4OH-
Fe(OH)4- + OH- + [18O]D- + OT- + OD- = FeOH[18O]DOTOD- + 4OH-
Fe(OH)4- + OH- + [18O]D- + OT- + [18O]H- = FeOH[18O]DOT[18O]H- + 4OH-
Fe(OH)4- + OH- + [18O]D- + [18O]H- + OH- = FeOH[18O]D[18O]HOH- + 4OH-
Fe(OH)4- + OH- + [18O]D- + [18O]H- + OD- = FeOH[18O]D[18O]HOD- + 4OH-
Fe(OH)4- + OH- + [18O]D- + [18O]H- + OT- = FeOH[18O]D[18O]HOT- + 4OH-
Fe(OH)4- + OH- + [18O]D- + [18O]D- + OH- = FeOH[18O]D[18O]DOH- + 4OH-
Fe(OH)4- + OH- + [18O]D- + [18O]T- + OH- = FeOH[18O]D[18O]TOH- + 4OH-
Fe(OH)4- + OH- + [18O]T- + OH- + OH- = FeOH[18O]TOHOH- + 4OH-
Fe(OH)4- + OH- + [18O]T- + OH- + OD- = FeOH[18O]TOHOD- + 4OH-
Fe(OH)4- + OH- + [18O]T- + OH- + [18O]H- = FeOH[18O]TOH[18O]H- + 4OH-
Fe(OH)4- + OH- + [18O]T- + OH- + [18O]D- = FeOH[18O]TOH[18O]D- + 4OH-
Fe(OH)4- + OH- + [18O]T- + OD- + OH- = FeOH[18O]TODOH- + 4OH-
Fe(OH)4- + OH- + [18O]T- + OD- + OD- = FeOH[18O]TODOD- + 4OH-
Fe(OH)4- + OH- + [18O]T- + OD- + [18O]H- = FeOH[18O]TOD[18O]H- + 4OH-
Fe(OH)4- + OH- + [18O]T- + [18O]H- + OH- = FeOH[18O]T[18O]HOH- + 4OH-
Fe(OH)4- + OH- + [18O]T- + [18O]H- + OD- = FeOH[18O]T[18O]HOD- + 4OH-
Fe(OH)4- + OH- + [18O]T- + [18O]D- + OH- = FeOH[18O]T[18O]DOH- + 4OH-
Fe(OH)4- + OD- + OH- + OH- + OH- = FeODOHOHOH- + 4OH-
Fe(OH)4- + OD- + OH- + OH- + OD- = FeODOHOHOD- + 4OH-
Fe(OH)4- + OD- + OH- + OH- + OT- = FeODOHOHOT- + 4OH-
Fe(OH)4- + OD- + OH- + OH- + [18O]H- = FeODOHOH[18O]H- + 4OH-
Fe(OH)4- + OD- + OH- + OH- + [18O]D- = FeODOHOH[18O]D- + 4OH-
Fe(OH)4- + OD- + OH- + OH- + [18O]T- = FeODOHOH[18O]T- + 4OH-
Fe(OH)4- + OD- + OH- + OD- + OH- = FeODOHODOH- + 4OH-
Fe(OH)4- + OD- + OH- + OD- + OT- = FeODOHODOT- + 4OH-
Fe(OH)4- + OD- + OH- + OD- + [18O]H- = FeODOHOD[18O]H- + 4OH-
Fe(OH)4- + OD- + OH- + OD- + [18O]T- = FeODOHOD[18O]T- + 4OH-
Fe(OH)4- + OD- + OH- + OT- + OH- = FeODOHOTOH- + 4OH-
Fe(OH)4- + OD- + OH- + OT- + OD- = FeODOHOTOD- + 4OH-
Fe(OH)4- + OD- + OH- + OT- + [18O]H- = FeODOHOT[18O]H- + 4OH-
Fe(OH)4- + OD- + OH- + OT- + [18O]D- = FeODOHOT[18O]D- + 4OH-
Fe(OH)4- + OD- + OH- + [18O]H- + OH- = FeODOH[18O]HOH- + 4OH-
Fe(OH)4- + OD- + OH- + [18O]H- + OD- = FeODOH[18O]HOD- + 4OH-
Fe(OH)4- + OD- + OH- + [18O]H- + OT- = FeODOH[18O]HOT- + 4OH-
Fe(OH)4- + OD- + OH- + [18O]H- + [18O]H- = FeODOH[18O]H[18O]H- + 4OH-
Fe(OH)4- + OD- + OH- + [18O]H- + [18O]D- = FeODOH[18O]H[18O]D- + 4OH-
Fe(OH)4- + OD- + OH- + [18O]H- + [18O]T- = FeODOH[18O]H[18O]T- + 4OH-
Fe(OH)4- + OD- + OH- + [18O]D- + OH- = FeODOH[18O]DOH- + 4OH-
Fe(OH)4- + OD- + OH- + [18O]D- + OT- = FeODOH[18O]DOT- + 4OH-
Fe(OH)4- + OD- + OH- + [18O]D- + [18O]H- = FeODOH[18O]D[18O]H- + 4OH-
Fe(OH)4- + OD- + OH- + [18O]T- + OH- = FeODOH[18O]TOH- + 4OH-
Fe(OH)4- + OD- + OH- + [18O]T- + OD- = FeODOH[18O]TOD- + 4OH-
Fe(OH)4- + OD- + OH- + [18O]T- + [18O]H- = FeODOH[18O]T[18O]H- + 4OH-
Fe(OH)4- + OD- + OD- + OH- + OH- = FeODODOHOH- + 4OH-
Fe(OH)4- + OD- + OD- + OH- + OT- = FeODODOHOT- + 4OH-
Fe(OH)4- + OD- + OD- + OH- + [18O]H- = FeODODOH[18O]H- + 4OH-
Fe(OH)4- + OD- + OD- + OH- + [18O]T- = FeODODOH[18O]T- + 4OH-
Fe(OH)4- + OD- + OD- + OT- + OH- = FeODODOTOH- + 4OH-
Fe(OH)4- + OD- + OD- + OT- + [18O]H- = FeODODOT[18O]H- + 4OH-
Fe(OH)4- + OD- + OD- + [18O]H- + OH- = FeODOD[18O]HOH- + 4OH-
Fe(OH)4- + OD- + OD- + [18O]H- + OT- = FeODOD[18O]HOT- + 4OH-
Fe(OH)4- + OD- + OD- + [18O]H- + [18O]H- = FeODOD[18O]H[18O]H- + 4OH-
Fe(OH)4- + OD- + OD- + [18O]T- + OH- = FeODOD[18O]TOH- + 4OH-
Fe(OH)4- + OD- + OT- + OH- + OH- = FeODOTOHOH- + 4OH-
Fe(OH)4- + OD- + OT- + OH- + OD- = FeODOTOHOD- + 4OH-
Fe(OH)4- + OD- + OT- + OH- + [18O]H- = FeODOTOH[18O]H- + 4OH-
Fe(OH)4- + OD- + OT- + OH- + [18O]D- = FeODOTOH[18O]D- + 4OH-
Fe(OH)4- + OD- + OT- + OD- + OH- = FeODOTODOH- + 4OH-
Fe(OH)4- + OD- + OT- + OD- + [18O]H- = FeODOTOD[18O]H- + 4OH-
Fe(OH)4- + OD- + OT- + [18O]H- + OH- = FeODOT[18O]HOH- + 4OH-
Fe(OH)4- + OD- + OT- + [18O]H- + OD- = FeODOT[18O]HOD- + 4OH-
Fe(OH)4- + OD- + OT- + [18O]H- + [18O]H- = FeODOT[18O]H[18O]H- + 4OH-
Fe(OH)4- + OD- + OT- + [18O]D- + OH- = FeODOT[18O]DOH- + 4OH-
Fe(OH)4- + OD- + [18O]H- + OH- + OH- = FeOD[18O]HOHOH- + 4OH-
Fe(OH)4- + OD- + [18O]H- + OH- + OD- = FeOD[18O]HOHOD- + 4OH-
Fe(OH)4- + OD- + [18O]H- + OH- + OT- = FeOD[18O]HOHOT- + 4OH-
Fe(OH)4- + OD- + [18O]H- + OH- + [18O]H- = FeOD[18O]HOH[18O]H- + 4OH-
Fe(OH)4- + OD- + [18O]H- + OH- + [18O]D- = FeOD[18O]HOH[18O]D- + 4OH-
Fe(OH)4- + OD- + [18O]H- + OH- + [18O]T- = FeOD[18O]HOH[18O]T- + 4OH-
Fe(OH)4- + OD- + [18O]H- + OD- + OH- = FeOD[18O]HODOH- + 4OH-
Fe(OH)4- + OD- + [18O]H- + OD- + OT- = FeOD[18O]HODOT- + 4OH-
Fe(OH)4- + OD- + [18O]H- + OD- + [18O]H- = FeOD[18O]HOD[18O]H- + 4OH-
```

```
Fe(OH)4- + OD- + [18O]H- + OT- + OH- = FeOD[18O]HOTOH- + 4OH-
Fe(OH)4- + OD- + [18O]H- + OT- + OD- = FeOD[18O]HOTOD- + 4OH-
Fe(OH)4- + OD- + [18O]H- + OT- + [18O]H- = FeOD[18O]HOT[18O]H- + 4OH-
Fe(OH)4- + OD- + [18O]H- + [18O]H- + OH- = FeOD[18O]H[18O]HOH- + 4OH-
Fe(OH)4- + OD- + [18O]H- + [18O]H- + OD- = FeOD[18O]H[18O]HOD- + 4OH-
Fe(OH)4- + OD- + [18O]H- + [18O]H- + OT- = FeOD[18O]H[18O]HOT- + 4OH-
Fe(OH)4- + OD- + [18O]H- + [18O]D- + OH- = FeOD[18O]H[18O]DOH- + 4OH-
Fe(OH)4- + OD- + [18O]H- + [18O]T- + OH- = FeOD[18O]H[18O]TOH- + 4OH-
Fe(OH)4- + OD- + [18O]D- + OH- + OH- = FeOD[18O]DOHOH- + 4OH-
Fe(OH)4- + OD- + [18O]D- + OH- + OT- = FeOD[18O]DOHOT- + 4OH-
Fe(OH)4- + OD- + [18O]D- + OH- + [18O]H- = FeOD[18O]DOH[18O]H- + 4OH-
Fe(OH)4- + OD- + [18O]D- + OT- + OH- = FeOD[18O]DOTOH- + 4OH-
Fe(OH)4- + OD- + [18O]D- + [18O]H- + OH- = FeOD[18O]D[18O]HOH- + 4OH-
Fe(OH)4- + OD- + [18O]T- + OH- + OH- = FeOD[18O]TOHOH- + 4OH-
Fe(OH)4- + OD- + [18O]T- + OH- + OD- = FeOD[18O]TOHOD- + 4OH-
Fe(OH)4- + OD- + [18O]T- + OH- + [18O]H- = FeOD[18O]TOH[18O]H- + 4OH-
Fe(OH)4- + OD- + [18O]T- + OD- + OH- = FeOD[18O]TODOH- + 4OH-
Fe(OH)4- + OD- + [18O]T- + [18O]H- + OH- = FeOD[18O]T[18O]HOH- + 4OH-
Fe(OH)4- + OT- + OH- + OH- + OH- = FeOTOHOHOH- + 4OH-
Fe(OH)4- + OT- + OH- + OH- + OD- = FeOTOHOHOD- + 4OH-
Fe(OH)4- + OT- + OH- + OH- + [18O]H- = FeOTOHOH[18O]H- + 4OH-
Fe(OH)4- + OT- + OH- + OH- + [18O]D- = FeOTOHOH[18O]D- + 4OH-
Fe(OH)4- + OT- + OH- + OD- + OH- = FeOTOHODOH- + 4OH-
Fe(OH)4- + OT- + OH- + OD- + OD- = FeOTOHODOD- + 4OH-
Fe(OH)4- + OT- + OH- + OD- + [18O]H- = FeOTOHOD[18O]H- + 4OH-
Fe(OH)4- + OT- + OH- + OD- + [18O]D- = FeOTOHOD[18O]D- + 4OH-
Fe(OH)4- + OT- + OH- + [18O]H- + OH- = FeOTOH[18O]HOH- + 4OH-
Fe(OH)4- + OT- + OH- + [18O]H- + OD- = FeOTOH[18O]HOD- + 4OH-
Fe(OH)4- + OT- + OH- + [18O]H- + [18O]H- = FeOTOH[18O]H[18O]H- + 4OH-
Fe(OH)4- + OT- + OH- + [18O]H- + [18O]D- = FeOTOH[18O]H[18O]D- + 4OH-
Fe(OH)4- + OT- + OH- + [18O]D- + OH- = FeOTOH[18O]DOH- + 4OH-
Fe(OH)4- + OT- + OH- + [18O]D- + OD- = FeOTOH[18O]DOD- + 4OH-
Fe(OH)4- + OT- + OH- + [18O]D- + [18O]H- = FeOTOH[18O]D[18O]H- + 4OH-
Fe(OH)4- + OT- + OD- + OH- + OH- = FeOTODOHOH- + 4OH-
Fe(OH)4- + OT- + OD- + OH- + OD- = FeOTODOHOD- + 4OH-
Fe(OH)4- + OT- + OD- + OH- + [18O]H- = FeOTODOH[18O]H- + 4OH-
Fe(OH)4- + OT- + OD- + OH- + [18O]D- = FeOTODOH[18O]D- + 4OH-
Fe(OH)4- + OT- + OD- + OD- + OH- = FeOTODODOH- + 4OH-
Fe(OH)4- + OT- + OD- + OD- + [18O]H- = FeOTODOD[18O]H- + 4OH-
Fe(OH)4- + OT- + OD- + [18O]H- + OH- = FeOTOD[18O]HOH- + 4OH-
Fe(OH)4- + OT- + OD- + [18O]H- + OD- = FeOTOD[18O]HOD- + 4OH-
Fe(OH)4- + OT- + OD- + [18O]H- + [18O]H- = FeOTOD[18O]H[18O]H- + 4OH-
Fe(OH)4- + OT- + OD- + [18O]D- + OH- = FeOTOD[18O]DOH- + 4OH-
Fe(OH)4- + OT- + [18O]H- + OH- + OH- = FeOT[18O]HOHOH- + 4OH-
Fe(OH)4- + OT- + [18O]H- + OH- + OD- = FeOT[18O]HOHOD- + 4OH-
Fe(OH)4- + OT- + [18O]H- + OH- + [18O]H- = FeOT[18O]HOH[18O]H- + 4OH-
Fe(OH)4- + OT- + [18O]H- + OH- + [18O]D- = FeOT[18O]HOH[18O]D- + 4OH-
Fe(OH)4- + OT- + [18O]H- + OD- + OH- = FeOT[18O]HODOH- + 4OH-
Fe(OH)4- + OT- + [18O]H- + OD- + OD- = FeOT[18O]HODOD- + 4OH-
Fe(OH)4- + OT- + [18O]H- + OD- + [18O]H- = FeOT[18O]HOD[18O]H- + 4OH-
Fe(OH)4- + OT- + [18O]H- + [18O]H- + OH- = FeOT[18O]H[18O]HOH- + 4OH-
Fe(OH)4- + OT- + [18O]H- + [18O]H- + OD- = FeOT[18O]H[18O]HOD- + 4OH-
Fe(OH)4- + OT- + [18O]H- + [18O]D- + OH- = FeOT[18O]H[18O]DOH- + 4OH-
Fe(OH)4- + OT- + [18O]D- + OH- + OH- = FeOT[18O]DOHOH- + 4OH-
Fe(OH)4- + OT- + [18O]D- + OH- + OD- = FeOT[18O]DOHOD- + 4OH-
Fe(OH)4- + OT- + [18O]D- + OD- + OH- = FeOT[18O]DODOH- + 4OH-
Fe(OH)4- + OT- + [18O]D- + [18O]H- + OH- = FeOT[18O]D[18O]HOH- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OH- + OH- = Fe[18O]HOHOHOH- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OH- + OD- = Fe[18O]HOHOHOD- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OH- + OT- = Fe[18O]HOHOHOT- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OH- + [18O]H- = Fe[18O]HOHOH[18O]H- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OH- + [18O]D- = Fe[18O]HOHOH[18O]D- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OH- + [18O]T- = Fe[18O]HOHOH[18O]T- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OD- + OH- = Fe[18O]HOHODOH- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OD- + OD- = Fe[18O]HOHODOD- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OD- + OT- = Fe[18O]HOHODOT- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OD- + [18O]H- = Fe[18O]HOHOD[18O]H- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OD- + [18O]D- = Fe[18O]HOHOD[18O]D- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OD- + [18O]T- = Fe[18O]HOHOD[18O]T- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OT- + OH- = Fe[18O]HOHOTOH- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OT- + OD- = Fe[18O]HOHOTOD- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OT- + [18O]H- = Fe[18O]HOHOT[18O]H- + 4OH-
Fe(OH)4- + [18O]H- + OH- + OT- + [18O]D- = Fe[18O]HOHOT[18O]D- + 4OH-
Fe(OH)4- + [18O]H- + OH- + [18O]H- + OH- = Fe[18O]HOH[18O]HOH- + 4OH-
Fe(OH)4- + [18O]H- + OH- + [18O]H- + OD- = Fe[18O]HOH[18O]HOD- + 4OH-
Fe(OH)4- + [18O]H- + OH- + [18O]H- + OT- = Fe[18O]HOH[18O]HOT- + 4OH-
Fe(OH)4- + [18O]H- + OH- + [18O]D- + OH- = Fe[18O]HOH[18O]DOH- + 4OH-
Fe(OH)4- + [18O]H- + OH- + [18O]D- + OD- = Fe[18O]HOH[18O]DOD- + 4OH-
Fe(OH)4- + [18O]H- + OH- + [18O]D- + OT- = Fe[18O]HOH[18O]DOT- + 4OH-
Fe(OH)4- + [18O]H- + OH- + [18O]T- + OH- = Fe[18O]HOH[18O]TOH- + 4OH-
Fe(OH)4- + [18O]H- + OH- + [18O]T- + OD- = Fe[18O]HOH[18O]TOD- + 4OH-
Fe(OH)4- + [18O]H- + OD- + OH- + OH- = Fe[18O]HODOHOH- + 4OH-
Fe(OH)4- + [18O]H- + OD- + OH- + OD- = Fe[18O]HODOHOD- + 4OH-
Fe(OH)4- + [18O]H- + OD- + OH- + OT- = Fe[18O]HODOHOT- + 4OH-
Fe(OH)4- + [18O]H- + OD- + OH- + [18O]H- = Fe[18O]HODOH[18O]H- + 4OH-
Fe(OH)4- + [18O]H- + OD- + OH- + [18O]D- = Fe[18O]HODOH[18O]D- + 4OH-
Fe(OH)4- + [18O]H- + OD- + OH- + [18O]T- = Fe[18O]HODOH[18O]T- + 4OH-
Fe(OH)4- + [18O]H- + OD- + OD- + OH- = Fe[18O]HODODOH- + 4OH-
Fe(OH)4- + [18O]H- + OD- + OD- + OT- = Fe[18O]HODODOT- + 4OH-
Fe(OH)4- + [18O]H- + OD- + OD- + [18O]H- = Fe[18O]HODOD[18O]H- + 4OH-
Fe(OH)4- + [18O]H- + OD- + OT- + OH- = Fe[18O]HODOTOH- + 4OH-
Fe(OH)4- + [18O]H- + OD- + OT- + OD- = Fe[18O]HODOTOD- + 4OH-
Fe(OH)4- + [18O]H- + OD- + OT- + [18O]H- = Fe[18O]HODOT[18O]H- + 4OH-
Fe(OH)4- + [18O]H- + OD- + [18O]H- + OH- = Fe[18O]HOD[18O]HOH- + 4OH-
Fe(OH)4- + [18O]H- + OD- + [18O]H- + OD- = Fe[18O]HOD[18O]HOD- + 4OH-
Fe(OH)4- + [18O]H- + OD- + [18O]H- + OT- = Fe[18O]HOD[18O]HOT- + 4OH-
Fe(OH)4- + [18O]H- + OD- + [18O]D- + OH- = Fe[18O]HOD[18O]DOH- + 4OH-
Fe(OH)4- + [18O]H- + OD- + [18O]T- + OH- = Fe[18O]HOD[18O]TOH- + 4OH-
Fe(OH)4- + [18O]H- + OT- + OH- + OH- = Fe[18O]HOTOHOH- + 4OH-
Fe(OH)4- + [18O]H- + OT- + OH- + OD- = Fe[18O]HOTOHOD- + 4OH-
Fe(OH)4- + [18O]H- + OT- + OH- + [18O]H- = Fe[18O]HOTOH[18O]H- + 4OH-
Fe(OH)4- + [18O]H- + OT- + OH- + [18O]D- = Fe[18O]HOTOH[18O]D- + 4OH-
Fe(OH)4- + [18O]H- + OT- + OD- + OH- = Fe[18O]HOTODOH- + 4OH-
Fe(OH)4- + [18O]H- + OT- + OD- + OD- = Fe[18O]HOTODOD- + 4OH-
Fe(OH)4- + [18O]H- + OT- + OD- + [18O]H- = Fe[18O]HOTOD[18O]H- + 4OH-
Fe(OH)4- + [18O]H- + OT- + [18O]H- + OH- = Fe[18O]HOT[18O]HOH- + 4OH-
Fe(OH)4- + [18O]H- + OT- + [18O]H- + OD- = Fe[18O]HOT[18O]HOD- + 4OH-
Fe(OH)4- + [18O]H- + OT- + [18O]D- + OH- = Fe[18O]HOT[18O]DOH- + 4OH-
Fe(OH)4- + [18O]H- + [18O]H- + OH- + OH- = Fe[18O]H[18O]HOHOH- + 4OH-
Fe(OH)4- + [18O]H- + [18O]H- + OH- + OD- = Fe[18O]H[18O]HOHOD- + 4OH-
Fe(OH)4- + [18O]H- + [18O]H- + OH- + OT- = Fe[18O]H[18O]HOHOT- + 4OH-
Fe(OH)4- + [18O]H- + [18O]H- + OD- + OH- = Fe[18O]H[18O]HODOH- + 4OH-
Fe(OH)4- + [18O]H- + [18O]H- + OD- + OD- = Fe[18O]H[18O]HODOD- + 4OH-
Fe(OH)4- + [18O]H- + [18O]H- + OD- + OT- = Fe[18O]H[18O]HODOT- + 4OH-
Fe(OH)4- + [18O]H- + [18O]H- + OT- + OH- = Fe[18O]H[18O]HOTOH- + 4OH-
```

```
Fe(OH)4- + [18O]H- + [18O]H- + OT- + OD- = Fe[18O]H[18O]HOTOD- + 4OH-
Fe(OH)4- + [18O]H- + [18O]D- + OH- + OH- = Fe[18O]H[18O]DOHOH- + 4OH-
Fe(OH)4- + [18O]H- + [18O]D- + OH- + OD- = Fe[18O]H[18O]DOHOD- + 4OH-
Fe(OH)4- + [18O]H- + [18O]D- + OH- + OT- = Fe[18O]H[18O]DOHOT- + 4OH-
Fe(OH)4- + [18O]H- + [18O]D- + OD- + OH- = Fe[18O]H[18O]DODOH- + 4OH-
Fe(OH)4- + [18O]H- + [18O]D- + OT- + OH- = Fe[18O]H[18O]DOTOH- + 4OH-
Fe(OH)4- + [18O]H- + [18O]T- + OH- + OH- = Fe[18O]H[18O]TOHOH- + 4OH-
Fe(OH)4- + [18O]H- + [18O]T- + OH- + OD- = Fe[18O]H[18O]TOHOD- + 4OH-
Fe(OH)4- + [18O]H- + [18O]T- + OD- + OH- = Fe[18O]H[18O]TODOH- + 4OH-
Fe(OH)4- + [18O]D- + OH- + OH- + OH- = Fe[18O]DOHOHOH- + 4OH-
Fe(OH)4- + [18O]D- + OH- + OH- + OD- = Fe[18O]DOHOHOD- + 4OH-
Fe(OH)4- + [18O]D- + OH- + OH- + OT- = Fe[18O]DOHOHOT- + 4OH-
Fe(OH)4- + [18O]D- + OH- + OH- + [18O]H- = Fe[18O]DOHOH[18O]H- + 4OH-
Fe(OH)4- + [18O]D- + OH- + OH- + [18O]D- = Fe[18O]DOHOH[18O]D- + 4OH-
Fe(OH)4- + [18O]D- + OH- + OH- + [18O]T- = Fe[18O]DOHOH[18O]T- + 4OH-
Fe(OH)4- + [18O]D- + OH- + OD- + OH- = Fe[18O]DOHODOH- + 4OH-
Fe(OH)4- + [18O]D- + OH- + OD- + OT- = Fe[18O]DOHODOT- + 4OH-
Fe(OH)4- + [18O]D- + OH- + OD- + [18O]H- = Fe[18O]DOHOD[18O]H- + 4OH-
Fe(OH)4- + [18O]D- + OH- + OT- + OH- = Fe[18O]DOHOTOH- + 4OH-
Fe(OH)4- + [18O]D- + OH- + OT- + OD- = Fe[18O]DOHOTOD- + 4OH-
Fe(OH)4- + [18O]D- + OH- + OT- + [18O]H- = Fe[18O]DOHOT[18O]H- + 4OH-
Fe(OH)4- + [18O]D- + OH- + [18O]H- + OH- = Fe[18O]DOH[18O]HOH- + 4OH-
Fe(OH)4- + [18O]D- + OH- + [18O]H- + OD- = Fe[18O]DOH[18O]HOD- + 4OH-
Fe(OH)4- + [18O]D- + OH- + [18O]H- + OT- = Fe[18O]DOH[18O]HOT- + 4OH-
Fe(OH)4- + [18O]D- + OH- + [18O]D- + OH- = Fe[18O]DOH[18O]DOH- + 4OH-
Fe(OH)4- + [18O]D- + OH- + [18O]T- + OH- = Fe[18O]DOH[18O]TOH- + 4OH-
Fe(OH)4- + [18O]D- + OD- + OH- + OH- = Fe[18O]DODOHOH- + 4OH-
Fe(OH)4- + [18O]D- + OD- + OH- + OT- = Fe[18O]DODOHOT- + 4OH-
Fe(OH)4- + [18O]D- + OD- + OH- + [18O]H- = Fe[18O]DODOH[18O]H- + 4OH-
Fe(OH)4- + [18O]D- + OD- + OT- + OH- = Fe[18O]DODOTOH- + 4OH-
Fe(OH)4- + [18O]D- + OD- + [18O]H- + OH- = Fe[18O]DOD[18O]HOH- + 4OH-
Fe(OH)4- + [18O]D- + OT- + OH- + OH- = Fe[18O]DOTOHOH- + 4OH-
Fe(OH)4- + [18O]D- + OT- + OH- + OD- = Fe[18O]DOTOHOD- + 4OH-
Fe(OH)4- + [18O]D- + OT- + OH- + [18O]H- = Fe[18O]DOTOH[18O]H- + 4OH-
Fe(OH)4- + [18O]D- + OT- + OD- + OH- = Fe[18O]DOTODOH- + 4OH-
Fe(OH)4- + [18O]D- + OT- + [18O]H- + OH- = Fe[18O]DOT[18O]HOH- + 4OH-
Fe(OH)4- + [18O]D- + [18O]H- + OH- + OH- = Fe[18O]D[18O]HOHOH- + 4OH-
Fe(OH)4- + [18O]D- + [18O]H- + OH- + OD- = Fe[18O]D[18O]HOHOD- + 4OH-
Fe(OH)4- + [18O]D- + [18O]H- + OH- + OT- = Fe[18O]D[18O]HOHOT- + 4OH-
Fe(OH)4- + [18O]D- + [18O]H- + OD- + OH- = Fe[18O]D[18O]HODOH- + 4OH-
Fe(OH)4- + [18O]D- + [18O]H- + OT- + OH- = Fe[18O]D[18O]HOTOH- + 4OH-
Fe(OH)4- + [18O]D- + [18O]D- + OH- + OH- = Fe[18O]D[18O]DOHOH- + 4OH-
Fe(OH)4- + [18O]D- + [18O]T- + OH- + OH- = Fe[18O]D[18O]TOHOH- + 4OH-
Fe(OH)4- + [18O]T- + OH- + OH- + OH- = Fe[18O]TOHOHOH- + 4OH-
Fe(OH)4- + [18O]T- + OH- + OH- + OD- = Fe[18O]TOHOHOD- + 4OH-
Fe(OH)4- + [18O]T- + OH- + OH- + [18O]H- = Fe[18O]TOHOH[18O]H- + 4OH-
Fe(OH)4- + [18O]T- + OH- + OH- + [18O]D- = Fe[18O]TOHOH[18O]D- + 4OH-
Fe(OH)4- + [18O]T- + OH- + OD- + OH- = Fe[18O]TOHODOH- + 4OH-
Fe(OH)4- + [18O]T- + OH- + OD- + OD- = Fe[18O]TOHODOD- + 4OH-
Fe(OH)4- + [18O]T- + OH- + OD- + [18O]H- = Fe[18O]TOHOD[18O]H- + 4OH-
Fe(OH)4- + [18O]T- + OH- + [18O]H- + OH- = Fe[18O]TOH[18O]HOH- + 4OH-
Fe(OH)4- + [18O]T- + OH- + [18O]H- + OD- = Fe[18O]TOH[18O]HOD- + 4OH-
Fe(OH)4- + [18O]T- + OH- + [18O]D- + OH- = Fe[18O]TOH[18O]DOH- + 4OH-
Fe(OH)4- + [18O]T- + OD- + OH- + OH- = Fe[18O]TODOHOH- + 4OH-
Fe(OH)4- + [18O]T- + OD- + OH- + OD- = Fe[18O]TODOHOD- + 4OH-
Fe(OH)4- + [18O]T- + OD- + OD- + OH- = Fe[18O]TODODOH- + 4OH-
Fe(OH)4- + [18O]T- + OD- + [18O]H- + OH- = Fe[18O]TOD[18O]HOH- + 4OH-
Fe(OH)4- + [18O]T- + [18O]H- + OH- + OH- = Fe[18O]T[18O]HOHOH- + 4OH-
Fe(OH)4- + [18O]T- + [18O]H- + OH- + OD- = Fe[18O]T[18O]HOHOD- + 4OH-
Fe(OH)4- + [18O]T- + [18O]H- + OD- + OH- = Fe[18O]T[18O]HODOH- + 4OH-
Fe(OH)4- + [18O]T- + [18O]D- + OH- + OH- = Fe[18O]T[18O]DOHOH- + 4OH-
#
# Added Fe2(OH)2+4 reactions 16Dec09
# Revised 17Dec09, limited the number of species
#
Fe2(OH)2+4 + OH- + OD- = Fe2OHOD+4 + 2OH-
Fe2(OH)2+4 + OH- + OT- = Fe2OHOT+4 + 2OH-
Fe2(OH)2+4 + OH- + [18O]H- = Fe2OH[18O]H+4 + 2OH-
Fe2(OH)2+4 + OH- + [18O]D- = Fe2OH[18O]D+4 + 2OH-
Fe2(OH)2+4 + OH- + [18O]T- = Fe2OH[18O]T+4 + 2OH-
Fe2(OH)2+4 + OD- + OH- = Fe2ODOH+4 + 2OH-
Fe2(OH)2+4 + OD- + OD- = Fe2ODOD+4 + 2OH-
Fe2(OH)2+4 + OD- + OT- = Fe2ODOT+4 + 2OH-
Fe2(OH)2+4 + OD- + [18O]H- = Fe2OD[18O]H+4 + 2OH-
Fe2(OH)2+4 + OD- + [18O]D- = Fe2OD[18O]D+4 + 2OH-
Fe2(OH)2+4 + OD- + [18O]T- = Fe2OD[18O]T+4 + 2OH-
Fe2(OH)2+4 + OT- + OH- = Fe2OTOH+4 + 2OH-
Fe2(OH)2+4 + OT- + OD- = Fe2OTOD+4 + 2OH-
Fe2(OH)2+4 + OT- + [18O]H- = Fe2OT[18O]H+4 + 2OH-
Fe2(OH)2+4 + OT- + [18O]D- = Fe2OT[18O]D+4 + 2OH-
Fe2(OH)2+4 + [18O]H- + OH- = Fe2[18O]HOH+4 + 2OH-
Fe2(OH)2+4 + [18O]H- + OD- = Fe2[18O]HOD+4 + 2OH-
Fe2(OH)2+4 + [18O]H- + OT- = Fe2[18O]HOT+4 + 2OH-
Fe2(OH)2+4 + [18O]H- + [18O]H- = Fe2[18O]H[18O]H+4 + 2OH-
Fe2(OH)2+4 + [18O]H- + [18O]D- = Fe2[18O]H[18O]D+4 + 2OH-
Fe2(OH)2+4 + [18O]H- + [18O]T- = Fe2[18O]H[18O]T+4 + 2OH-
Fe2(OH)2+4 + [18O]D- + OH- = Fe2[18O]DOH+4 + 2OH-
Fe2(OH)2+4 + [18O]D- + OD- = Fe2[18O]DOD+4 + 2OH-
Fe2(OH)2+4 + [18O]D- + OT- = Fe2[18O]DOT+4 + 2OH-
Fe2(OH)2+4 + [18O]D- + [18O]H- = Fe2[18O]D[18O]H+4 + 2OH-
Fe2(OH)2+4 + [18O]D- + [18O]D- = Fe2[18O]D[18O]D+4 + 2OH-
Fe2(OH)2+4 + [18O]D- + [18O]T- = Fe2[18O]D[18O]T+4 + 2OH-
Fe2(OH)2+4 + [18O]T- + OH- = Fe2[18O]TOH+4 + 2OH-
Fe2(OH)2+4 + [18O]T- + OD- = Fe2[18O]TOD+4 + 2OH-
Fe2(OH)2+4 + [18O]T- + [18O]H- = Fe2[18O]T[18O]H+4 + 2OH-
Fe2(OH)2+4 + [18O]T- + [18O]D- = Fe2[18O]T[18O]D+4 + 2OH-
#
# Added Fe3(OH)4+5 reactions 16Dec09
# Revised 17Dec09, limited the number of species
#
Fe3(OH)4+5 + OH- + OH- + OH- + OD- = Fe3OHOHOHOD+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + OH- + OT- = Fe3OHOHOHOT+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + OH- + [18O]H- = Fe3OHOHOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + OH- + [18O]D- = Fe3OHOHOH[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + OH- + [18O]T- = Fe3OHOHOH[18O]T+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + OD- + OH- = Fe3OHOHODOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + OD- + OD- = Fe3OHOHODOD+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + OD- + OT- = Fe3OHOHODOT+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + OD- + [18O]H- = Fe3OHOHOD[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + OD- + [18O]D- = Fe3OHOHOD[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + OD- + [18O]T- = Fe3OHOHOD[18O]T+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + OT- + OH- = Fe3OHOHOTOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + OT- + OD- = Fe3OHOHOTOD+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + OT- + [18O]H- = Fe3OHOHOT[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + OT- + [18O]D- = Fe3OHOHOT[18O]D+5 + 4OH-
```

```
Fe3(OH)4+5 + OH- + OH- + [18O]H- + OH- = Fe3OHOH[18O]HOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]H- + OD- = Fe3OHOH[18O]HOD+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]H- + OT- = Fe3OHOH[18O]HOT+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]H- + [18O]H- = Fe3OHOH[18O]H[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]H- + [18O]D- = Fe3OHOH[18O]H[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]H- + [18O]T- = Fe3OHOH[18O]H[18O]T+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]D- + OH- = Fe3OHOH[18O]DOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]D- + OD- = Fe3OHOH[18O]DOD+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]D- + OT- = Fe3OHOH[18O]DOT+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]D- + [18O]H- = Fe3OHOH[18O]D[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]D- + [18O]D- = Fe3OHOH[18O]D[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]D- + [18O]T- = Fe3OHOH[18O]D[18O]T+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]T- + OH- = Fe3OHOH[18O]TOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]T- + OD- = Fe3OHOH[18O]TOD+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]T- + [18O]H- = Fe3OHOH[18O]T[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OH- + [18O]T- + [18O]D- = Fe3OHOH[18O]T[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + OH- + OH- = Fe3OHODOHOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + OH- + OD- = Fe3OHODOHOD+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + OH- + OT- = Fe3OHODOHOT+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + OH- + [18O]H- = Fe3OHODOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + OH- + [18O]D- = Fe3OHODOH[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + OH- + [18O]T- = Fe3OHODOH[18O]T+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + OD- + OH- = Fe3OHODODOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + OD- + OT- = Fe3OHODODOT+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + OD- + [18O]H- = Fe3OHODOD[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + OD- + [18O]T- = Fe3OHODOD[18O]T+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + OT- + OH- = Fe3OHODOTOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + OT- + OD- = Fe3OHODOTOD+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + OT- + [18O]H- = Fe3OHODOT[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + OT- + [18O]D- = Fe3OHODOT[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + [18O]H- + OH- = Fe3OHOD[18O]HOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + [18O]H- + OD- = Fe3OHOD[18O]HOD+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + [18O]H- + OT- = Fe3OHOD[18O]HOT+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + [18O]H- + [18O]H- = Fe3OHOD[18O]H[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + [18O]H- + [18O]D- = Fe3OHOD[18O]H[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + [18O]H- + [18O]T- = Fe3OHOD[18O]H[18O]T+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + [18O]D- + OH- = Fe3OHOD[18O]DOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + [18O]D- + OT- = Fe3OHOD[18O]DOT+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + [18O]D- + [18O]H- = Fe3OHOD[18O]D[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + [18O]T- + OH- = Fe3OHOD[18O]TOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + [18O]T- + OD- = Fe3OHOD[18O]TOD+5 + 4OH-
Fe3(OH)4+5 + OH- + OD- + [18O]T- + [18O]H- = Fe3OHOD[18O]T[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + OH- + OH- = Fe3OHOTOHOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + OH- + OD- = Fe3OHOTOHOD+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + OH- + [18O]H- = Fe3OHOTOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + OH- + [18O]D- = Fe3OHOTOH[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + OD- + OH- = Fe3OHOTODOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + OD- + OD- = Fe3OHOTODOD+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + OD- + [18O]H- = Fe3OHOTOD[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + OD- + [18O]D- = Fe3OHOTOD[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + [18O]H- + OH- = Fe3OHOT[18O]HOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + [18O]H- + OD- = Fe3OHOT[18O]HOD+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + [18O]H- + [18O]H- = Fe3OHOT[18O]H[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + [18O]H- + [18O]D- = Fe3OHOT[18O]H[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + [18O]D- + OH- = Fe3OHOT[18O]DOH+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + [18O]D- + OD- = Fe3OHOT[18O]DOD+5 + 4OH-
Fe3(OH)4+5 + OH- + OT- + [18O]D- + [18O]H- = Fe3OHOT[18O]D[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OH- + OH- = Fe3OH[18O]HOHOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OH- + OD- = Fe3OH[18O]HOHOD+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OH- + OT- = Fe3OH[18O]HOHOT+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OH- + [18O]H- = Fe3OH[18O]HOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OH- + [18O]D- = Fe3OH[18O]HOH[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OH- + [18O]T- = Fe3OH[18O]HOH[18O]T+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OD- + OH- = Fe3OH[18O]HODOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OD- + OD- = Fe3OH[18O]HODOD+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OD- + OT- = Fe3OH[18O]HODOT+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OD- + [18O]H- = Fe3OH[18O]HOD[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OD- + [18O]D- = Fe3OH[18O]HOD[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OD- + [18O]T- = Fe3OH[18O]HOD[18O]T+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OT- + OH- = Fe3OH[18O]HOTOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OT- + OD- = Fe3OH[18O]HOTOD+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OT- + [18O]H- = Fe3OH[18O]HOT[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + OT- + [18O]D- = Fe3OH[18O]HOT[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + [18O]H- + OH- = Fe3OH[18O]H[18O]HOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + [18O]H- + OD- = Fe3OH[18O]H[18O]HOD+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + [18O]H- + OT- = Fe3OH[18O]H[18O]HOT+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + [18O]D- + OH- = Fe3OH[18O]H[18O]DOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + [18O]D- + OD- = Fe3OH[18O]H[18O]DOD+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + [18O]D- + OT- = Fe3OH[18O]H[18O]DOT+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + [18O]T- + OH- = Fe3OH[18O]H[18O]TOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]H- + [18O]T- + OD- = Fe3OH[18O]H[18O]TOD+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + OH- + OH- = Fe3OH[18O]DOHOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + OH- + OD- = Fe3OH[18O]DOHOD+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + OH- + OT- = Fe3OH[18O]DOHOT+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + [18O]H- = Fe3OH[18O]DOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + [18O]D- = Fe3OH[18O]DOH[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + [18O]T- = Fe3OH[18O]DOH[18O]T+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + OD- + OH- = Fe3OH[18O]DODOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + OD- + OD- = Fe3OH[18O]DODOD+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + OD- + OT- = Fe3OH[18O]DODOT+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + OT- + OH- = Fe3OH[18O]DOTOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + OT- + OD- = Fe3OH[18O]DOTOD+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + OT- + [18O]H- = Fe3OH[18O]DOT[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + [18O]H- + OH- = Fe3OH[18O]D[18O]HOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + [18O]H- + OD- = Fe3OH[18O]D[18O]HOD+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + [18O]H- + OT- = Fe3OH[18O]D[18O]HOT+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + [18O]D- + OH- = Fe3OH[18O]D[18O]DOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]D- + [18O]T- + OH- = Fe3OH[18O]D[18O]TOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]T- + OH- + OH- = Fe3OH[18O]TOHOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]T- + OH- + OD- = Fe3OH[18O]TOHOD+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]T- + OH- + [18O]H- = Fe3OH[18O]TOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]T- + OH- + [18O]D- = Fe3OH[18O]TOH[18O]D+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]T- + OD- + OH- = Fe3OH[18O]TODOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]T- + OD- + OD- = Fe3OH[18O]TODOD+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]T- + OD- + [18O]H- = Fe3OH[18O]TOD[18O]H+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]T- + [18O]H- + OH- = Fe3OH[18O]T[18O]HOH+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]T- + [18O]H- + OD- = Fe3OH[18O]T[18O]HOD+5 + 4OH-
Fe3(OH)4+5 + OH- + [18O]T- + [18O]D- + OH- = Fe3OH[18O]T[18O]DOH+5 + 4OH-
Fe3(OH)4+5 + OD- + OH- + OH- + OH- = Fe3ODOHOHOH+5 + 4OH-
Fe3(OH)4+5 + OD- + OH- + OH- + OD- = Fe3ODOHOHOD+5 + 4OH-
Fe3(OH)4+5 + OD- + OH- + OH- + OT- = Fe3ODOHOHOT+5 + 4OH-
Fe3(OH)4+5 + OD- + OH- + OH- + [18O]H- = Fe3ODOHOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + OD- + OH- + OH- + [18O]D- = Fe3ODOHOH[18O]D+5 + 4OH-
Fe3(OH)4+5 + OD- + OH- + OH- + [18O]T- = Fe3ODOHOH[18O]T+5 + 4OH-
Fe3(OH)4+5 + OD- + OH- + OD- + OH- = Fe3ODOHODOH+5 + 4OH-
Fe3(OH)4+5 + OD- + OH- + OD- + OT- = Fe3ODOHODOT+5 + 4OH-
```

```
|Fe3(OH)4+5 + OD- + OH- + OD- + [18O]H- = Fe3ODOHOD[18O]H+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OH- + OD- + [18O]T- = Fe3ODOHOD[18O]T+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OH- + OT- + OH- = Fe3ODOHOTOH+5 + 4OH-                        |
|Fe3(OH)4+5 + OD- + OH- + OT- + OD- = Fe3ODOHOTOD+5 + 4OH-                        |
|Fe3(OH)4+5 + OD- + OH- + OT- + [18O]H- = Fe3ODOHOT[18O]H+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OH- + OT- + [18O]D- = Fe3ODOHOT[18O]D+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OH- + [18O]H- + OH- = Fe3ODOH[18O]HOH+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OH- + [18O]H- + OD- = Fe3ODOH[18O]HOD+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OH- + [18O]H- + OT- = Fe3ODOH[18O]HOT+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OH- + [18O]H- + [18O]H- = Fe3ODOH[18O]H[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + OH- + [18O]H- + [18O]D- = Fe3ODOH[18O]H[18O]D+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + OH- + [18O]H- + [18O]T- = Fe3ODOH[18O]H[18O]T+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + OH- + [18O]D- + OH- = Fe3ODOH[18O]DOH+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OH- + [18O]D- + OT- = Fe3ODOH[18O]DOT+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OH- + [18O]D- + [18O]H- = Fe3ODOH[18O]D[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + OH- + [18O]T- + OH- = Fe3ODOH[18O]TOH+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OH- + [18O]T- + OD- = Fe3ODOH[18O]TOD+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OH- + [18O]T- + [18O]H- = Fe3ODOH[18O]T[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + OD- + OH- + OH- = Fe3ODODOHOH+5 + 4OH-                        |
|Fe3(OH)4+5 + OD- + OD- + OH- + OT- = Fe3ODODOHOT+5 + 4OH-                        |
|Fe3(OH)4+5 + OD- + OD- + OH- + [18O]H- = Fe3ODODOH[18O]H+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OD- + OH- + [18O]T- = Fe3ODODOH[18O]T+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OD- + OT- + OH- = Fe3ODODOTOH+5 + 4OH-                        |
|Fe3(OH)4+5 + OD- + OD- + OT- + [18O]H- = Fe3ODODOT[18O]H+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OD- + [18O]H- + OH- = Fe3ODOD[18O]HOH+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OD- + [18O]H- + OT- = Fe3ODOD[18O]HOT+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OD- + [18O]H- + [18O]H- = Fe3ODOD[18O]H[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + OD- + [18O]T- + OH- = Fe3ODOD[18O]TOH+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OT- + OH- + OH- = Fe3ODOTOHOH+5 + 4OH-                        |
|Fe3(OH)4+5 + OD- + OT- + OH- + OD- = Fe3ODOTOHOD+5 + 4OH-                        |
|Fe3(OH)4+5 + OD- + OT- + OH- + [18O]H- = Fe3ODOTOH[18O]H+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OT- + OH- + [18O]D- = Fe3ODOTOH[18O]D+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OT- + OD- + OH- = Fe3ODOTODOH+5 + 4OH-                        |
|Fe3(OH)4+5 + OD- + OT- + OD- + [18O]H- = Fe3ODOTOD[18O]H+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OT- + [18O]H- + OH- = Fe3ODOT[18O]HOH+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OT- + [18O]H- + OD- = Fe3ODOT[18O]HOD+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + OT- + [18O]H- + [18O]H- = Fe3ODOT[18O]H[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + OT- + [18O]D- + OH- = Fe3ODOT[18O]DOH+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + [18O]H- + OH- + OH- = Fe3OD[18O]HOHOH+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + [18O]H- + OH- + OD- = Fe3OD[18O]HOHOD+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + [18O]H- + OH- + OT- = Fe3OD[18O]HOHOT+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + [18O]H- + OH- + [18O]H- = Fe3OD[18O]HOH[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + [18O]H- + OH- + [18O]D- = Fe3OD[18O]HOH[18O]D+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + [18O]H- + OH- + [18O]T- = Fe3OD[18O]HOH[18O]T+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + [18O]H- + OD- + OH- = Fe3OD[18O]HODOH+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + [18O]H- + OD- + OT- = Fe3OD[18O]HODOT+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + [18O]H- + OD- + [18O]H- = Fe3OD[18O]HOD[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + [18O]H- + OT- + OH- = Fe3OD[18O]HOTOH+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + [18O]H- + OT- + OD- = Fe3OD[18O]HOTOD+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + [18O]H- + OT- + [18O]H- = Fe3OD[18O]HOT[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + [18O]H- + [18O]H- + OH- = Fe3OD[18O]H[18O]HOH+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + [18O]H- + [18O]H- + OD- = Fe3OD[18O]H[18O]HOD+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + [18O]H- + [18O]H- + OT- = Fe3OD[18O]H[18O]HOT+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + [18O]H- + [18O]D- + OH- = Fe3OD[18O]H[18O]DOH+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + [18O]H- + [18O]T- + OH- = Fe3OD[18O]H[18O]TOH+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + [18O]D- + OH- + OH- = Fe3OD[18O]DOHOH+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + [18O]D- + OH- + OT- = Fe3OD[18O]DOHOT+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + [18O]D- + OH- + [18O]H- = Fe3OD[18O]DOH[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + [18O]D- + OT- + OH- = Fe3OD[18O]DOTOH+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + [18O]D- + [18O]H- + OH- = Fe3OD[18O]D[18O]HOH+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + [18O]T- + OH- + OH- = Fe3OD[18O]TOHOH+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + [18O]T- + OH- + OD- = Fe3OD[18O]TOHOD+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + [18O]T- + OH- + [18O]H- = Fe3OD[18O]TOH[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OD- + [18O]T- + OD- + OH- = Fe3OD[18O]TODOH+5 + 4OH-                |
|Fe3(OH)4+5 + OD- + [18O]T- + [18O]H- + OH- = Fe3OD[18O]T[18O]HOH+5 + 4OH-        |
|Fe3(OH)4+5 + OT- + OH- + OH- + OH- = Fe3OTOHOHOH+5 + 4OH-                        |
|Fe3(OH)4+5 + OT- + OH- + OH- + OD- = Fe3OTOHOHOD+5 + 4OH-                        |
|Fe3(OH)4+5 + OT- + OH- + OH- + [18O]H- = Fe3OTOHOH[18O]H+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + OH- + OH- + [18O]D- = Fe3OTOHOH[18O]D+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + OH- + OD- + OH- = Fe3OTOHODOH+5 + 4OH-                        |
|Fe3(OH)4+5 + OT- + OH- + OD- + OD- = Fe3OTOHODOD+5 + 4OH-                        |
|Fe3(OH)4+5 + OT- + OH- + OD- + [18O]H- = Fe3OTOHOD[18O]H+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + OH- + OD- + [18O]D- = Fe3OTOHOD[18O]D+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + OH- + [18O]H- + OH- = Fe3OTOH[18O]HOH+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + OH- + [18O]H- + OD- = Fe3OTOH[18O]HOD+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + OH- + [18O]H- + [18O]H- = Fe3OTOH[18O]H[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OT- + OH- + [18O]H- + [18O]D- = Fe3OTOH[18O]H[18O]D+5 + 4OH-        |
|Fe3(OH)4+5 + OT- + OH- + [18O]D- + OH- = Fe3OTOH[18O]DOH+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + OH- + [18O]D- + OD- = Fe3OTOH[18O]DOD+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + OH- + [18O]D- + [18O]H- = Fe3OTOH[18O]D[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OT- + OD- + OH- + OH- = Fe3OTODOHOH+5 + 4OH-                        |
|Fe3(OH)4+5 + OT- + OD- + OH- + OD- = Fe3OTODOHOD+5 + 4OH-                        |
|Fe3(OH)4+5 + OT- + OD- + OH- + [18O]H- = Fe3OTODOH[18O]H+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + OD- + OH- + [18O]D- = Fe3OTODOH[18O]D+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + OD- + OD- + OH- = Fe3OTODODOH+5 + 4OH-                        |
|Fe3(OH)4+5 + OT- + OD- + OD- + [18O]H- = Fe3OTODOD[18O]H+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + OD- + [18O]H- + OH- = Fe3OTOD[18O]HOH+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + OD- + [18O]H- + OD- = Fe3OTOD[18O]HOD+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + OD- + [18O]H- + [18O]H- = Fe3OTOD[18O]H[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OT- + OD- + [18O]D- + OH- = Fe3OTOD[18O]DOH+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + [18O]H- + OH- + OH- = Fe3OT[18O]HOHOH+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + [18O]H- + OH- + OD- = Fe3OT[18O]HOHOD+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + [18O]H- + OH- + [18O]H- = Fe3OT[18O]HOH[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OT- + [18O]H- + OH- + [18O]D- = Fe3OT[18O]HOH[18O]D+5 + 4OH-        |
|Fe3(OH)4+5 + OT- + [18O]H- + OD- + OH- = Fe3OT[18O]HODOH+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + [18O]H- + OD- + OD- = Fe3OT[18O]HODOD+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + [18O]H- + OD- + [18O]H- = Fe3OT[18O]HOD[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OT- + [18O]H- + [18O]H- + OH- = Fe3OT[18O]H[18O]HOH+5 + 4OH-        |
|Fe3(OH)4+5 + OT- + [18O]H- + [18O]H- + OD- = Fe3OT[18O]H[18O]HOD+5 + 4OH-        |
|Fe3(OH)4+5 + OT- + [18O]H- + [18O]D- + OH- = Fe3OT[18O]H[18O]DOH+5 + 4OH-        |
|Fe3(OH)4+5 + OT- + [18O]D- + OH- + OH- = Fe3OT[18O]DOHOH+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + [18O]D- + OH- + OD- = Fe3OT[18O]DOHOD+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + [18O]D- + OH- + [18O]H- = Fe3OT[18O]DOH[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + OT- + [18O]D- + OD- + OH- = Fe3OT[18O]DODOH+5 + 4OH-                |
|Fe3(OH)4+5 + OT- + [18O]D- + [18O]H- + OH- = Fe3OT[18O]D[18O]HOH+5 + 4OH-        |
|Fe3(OH)4+5 + [18O]H- + OH- + OH- + OH- = Fe3[18O]HOHOHOH+5 + 4OH-                |
|Fe3(OH)4+5 + [18O]H- + OH- + OH- + OD- = Fe3[18O]HOHOHOD+5 + 4OH-                |
|Fe3(OH)4+5 + [18O]H- + OH- + OH- + OT- = Fe3[18O]HOHOHOT+5 + 4OH-                |
|Fe3(OH)4+5 + [18O]H- + OH- + OH- + [18O]H- = Fe3[18O]HOHOH[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + [18O]H- + OH- + OH- + [18O]D- = Fe3[18O]HOHOH[18O]D+5 + 4OH-        |
|Fe3(OH)4+5 + [18O]H- + OH- + OH- + [18O]T- = Fe3[18O]HOHOH[18O]T+5 + 4OH-        |
|Fe3(OH)4+5 + [18O]H- + OH- + OD- + OH- = Fe3[18O]HOHODOH+5 + 4OH-                |
|Fe3(OH)4+5 + [18O]H- + OH- + OD- + OD- = Fe3[18O]HOHODOD+5 + 4OH-                |
|Fe3(OH)4+5 + [18O]H- + OH- + OD- + OT- = Fe3[18O]HOHODOT+5 + 4OH-                |
|Fe3(OH)4+5 + [18O]H- + OH- + OD- + [18O]H- = Fe3[18O]HOHOD[18O]H+5 + 4OH-        |
|Fe3(OH)4+5 + [18O]H- + OH- + OD- + [18O]D- = Fe3[18O]HOHOD[18O]D+5 + 4OH-        |
```

```
Fe3(OH)4+5 + [18O]H- + OH- + OD- + [18O]T- = Fe3[18O]HOHOD[18O]T+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OH- + OT- + OH- = Fe3[18O]HOHOTOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OH- + OT- + OD- = Fe3[18O]HOHOTOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OH- + OT- + [18O]H- = Fe3[18O]HOHOT[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OH- + OT- + [18O]D- = Fe3[18O]HOHOT[18O]D+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OH- + [18O]H- + OH- = Fe3[18O]HOH[18O]HOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OH- + [18O]H- + OD- = Fe3[18O]HOH[18O]HOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OH- + [18O]H- + OT- = Fe3[18O]HOH[18O]HOT+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OH- + [18O]D- + OH- = Fe3[18O]HOH[18O]DOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OH- + [18O]D- + OD- = Fe3[18O]HOH[18O]DOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OH- + [18O]D- + OT- = Fe3[18O]HOH[18O]DOT+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OH- + [18O]T- + OH- = Fe3[18O]HOH[18O]TOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OH- + [18O]T- + OD- = Fe3[18O]HOH[18O]TOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + OH- + OH- = Fe3[18O]HODOHOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + OH- + OD- = Fe3[18O]HODOHOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + OH- + OT- = Fe3[18O]HODOHOT+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + OH- + [18O]H- = Fe3[18O]HODOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + OH- + [18O]D- = Fe3[18O]HODOH[18O]D+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + OH- + [18O]T- = Fe3[18O]HODOH[18O]T+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + OD- + OH- = Fe3[18O]HODODOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + OD- + OT- = Fe3[18O]HODODOT+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + OD- + [18O]H- = Fe3[18O]HODOD[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + OT- + OH- = Fe3[18O]HODOTOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + OT- + OD- = Fe3[18O]HODOTOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + OT- + [18O]H- = Fe3[18O]HODOT[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + [18O]H- + OH- = Fe3[18O]HOD[18O]HOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + [18O]H- + OD- = Fe3[18O]HOD[18O]HOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + [18O]H- + OT- = Fe3[18O]HOD[18O]HOT+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + [18O]D- + OH- = Fe3[18O]HOD[18O]DOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OD- + [18O]T- + OH- = Fe3[18O]HOD[18O]TOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OT- + OH- + OH- = Fe3[18O]HOTOHOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OT- + OH- + OD- = Fe3[18O]HOTOHOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OT- + OH- + [18O]H- = Fe3[18O]HOTOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OT- + OH- + [18O]D- = Fe3[18O]HOTOH[18O]D+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OT- + OD- + OH- = Fe3[18O]HOTODOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OT- + OD- + OD- = Fe3[18O]HOTODOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OT- + OD- + [18O]H- = Fe3[18O]HOTOD[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OT- + [18O]H- + OH- = Fe3[18O]HOT[18O]HOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OT- + [18O]H- + OD- = Fe3[18O]HOT[18O]HOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + OT- + [18O]D- + OH- = Fe3[18O]HOT[18O]DOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]H- + OH- + OH- = Fe3[18O]H[18O]HOHOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]H- + OH- + OD- = Fe3[18O]H[18O]HOHOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]H- + OH- + OT- = Fe3[18O]H[18O]HOHOT+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]H- + OD- + OH- = Fe3[18O]H[18O]HODOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]H- + OD- + OD- = Fe3[18O]H[18O]HODOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]H- + OD- + OT- = Fe3[18O]H[18O]HODOT+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]H- + OT- + OH- = Fe3[18O]H[18O]HOTOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]H- + OT- + OD- = Fe3[18O]H[18O]HOTOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]D- + OH- + OH- = Fe3[18O]H[18O]DOHOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]D- + OH- + OD- = Fe3[18O]H[18O]DOHOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]D- + OH- + OT- = Fe3[18O]H[18O]DOHOT+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]D- + OD- + OH- = Fe3[18O]H[18O]DODOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]D- + OT- + OH- = Fe3[18O]H[18O]DOTOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]T- + OH- + OH- = Fe3[18O]H[18O]TOHOH+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]T- + OH- + OD- = Fe3[18O]H[18O]TOHOD+5 + 4OH-
Fe3(OH)4+5 + [18O]H- + [18O]T- + OD- + OH- = Fe3[18O]H[18O]TODOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + OH- + OH- = Fe3[18O]DOHOHOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + OH- + OD- = Fe3[18O]DOHOHOD+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + OH- + OT- = Fe3[18O]DOHOHOT+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + OH- + [18O]H- = Fe3[18O]DOHOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + OH- + [18O]D- = Fe3[18O]DOHOH[18O]D+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + OH- + [18O]T- = Fe3[18O]DOHOH[18O]T+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + OD- + OH- = Fe3[18O]DOHODOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + OD- + OT- = Fe3[18O]DOHODOT+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + OD- + [18O]H- = Fe3[18O]DOHOD[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + OT- + OH- = Fe3[18O]DOHOTOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + OT- + OD- = Fe3[18O]DOHOTOD+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + OT- + [18O]H- = Fe3[18O]DOHOT[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + [18O]H- + OH- = Fe3[18O]DOH[18O]HOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + [18O]H- + OD- = Fe3[18O]DOH[18O]HOD+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + [18O]H- + OT- = Fe3[18O]DOH[18O]HOT+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + [18O]D- + OH- = Fe3[18O]DOH[18O]DOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OH- + [18O]T- + OH- = Fe3[18O]DOH[18O]TOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OD- + OH- + OH- = Fe3[18O]DODOHOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OD- + OH- + OT- = Fe3[18O]DODOHOT+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OD- + OH- + [18O]H- = Fe3[18O]DODOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OD- + OT- + OH- = Fe3[18O]DODOTOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OD- + [18O]H- + OH- = Fe3[18O]DOD[18O]HOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OT- + OH- + OH- = Fe3[18O]DOTOHOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OT- + OH- + OD- = Fe3[18O]DOTOHOD+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OT- + OH- + [18O]H- = Fe3[18O]DOTOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OT- + OD- + OH- = Fe3[18O]DOTODOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + OT- + [18O]H- + OH- = Fe3[18O]DOT[18O]HOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + OH- + OH- = Fe3[18O]D[18O]HOHOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + OH- + OD- = Fe3[18O]D[18O]HOHOD+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + OH- + OT- = Fe3[18O]D[18O]HOHOT+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + OD- + OH- = Fe3[18O]D[18O]HODOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + OD- + OT- = Fe3[18O]D[18O]HODOT+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + OT- + OH- = Fe3[18O]D[18O]HOTOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + OT- + OD- = Fe3[18O]D[18O]HOTOD+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + [18O]H- + OH- = Fe3[18O]D[18O]HOHOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + [18O]H- + OD- = Fe3[18O]D[18O]HOHOD+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + [18O]H- + OT- = Fe3[18O]D[18O]HOHOT+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + [18O]D- + OH- = Fe3[18O]D[18O]HODOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + [18O]D- + OD- = Fe3[18O]D[18O]HODOD+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + [18O]D- + OT- = Fe3[18O]D[18O]HODOT+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + [18O]T- + OH- = Fe3[18O]D[18O]HOTOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + [18O]T- + OD- = Fe3[18O]D[18O]HOTOD+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + [18O]T- + [18O]H- = Fe3[18O]D[18O]HOTOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]H- + [18O]T- + [18O]D- = Fe3[18O]D[18O]HOTOH[18O]D+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]T- + OH- + OH- = Fe3[18O]D[18O]TOHOH+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]T- + OH- + OD- = Fe3[18O]D[18O]TOHOD+5 + 4OH-
Fe3(OH)4+5 + [18O]D- + [18O]T- + OD- + OH- = Fe3[18O]D[18O]TODOH+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OH- + OH- + OH- = Fe3[18O]TOHOHOH+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OH- + OH- + OD- = Fe3[18O]TOHOHOD+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OH- + OH- + [18O]H- = Fe3[18O]TOHOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OH- + OH- + [18O]D- = Fe3[18O]TOHOH[18O]D+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OH- + OD- + OH- = Fe3[18O]TOHODOH+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OH- + OD- + OD- = Fe3[18O]TOHODOD+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OH- + OD- + [18O]H- = Fe3[18O]TOHOD[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OH- + [18O]H- + OH- = Fe3[18O]TOH[18O]HOH+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OH- + [18O]H- + OD- = Fe3[18O]TOH[18O]HOD+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OH- + [18O]D- + OH- = Fe3[18O]TOH[18O]DOH+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OD- + OH- + OH- = Fe3[18O]TODOHOH+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OD- + OH- + OD- = Fe3[18O]TODOHOD+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OD- + OH- + [18O]H- = Fe3[18O]TODOH[18O]H+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OD- + OD- + OH- = Fe3[18O]TODODOH+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + OD- + [18O]H- + OH- = Fe3[18O]TOD[18O]HOH+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + [18O]H- + OH- + OH- = Fe3[18O]T[18O]HOHOH+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + [18O]H- + OH- + OD- = Fe3[18O]T[18O]HOHOD+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + [18O]H- + OD- + OH- = Fe3[18O]T[18O]HODOH+5 + 4OH-
Fe3(OH)4+5 + [18O]T- + [18O]D- + OH- + OH- = Fe3[18O]T[18O]DOHOH+5 + 4OH-
#
# Added FeSO4+ reactions 16Dec09
#
FeSO4+ + [34S]O4-2 = Fe[34S]O4+ + SO4-2
#
# Added FeHSO4+2 reactions 16Dec09
#
```

```
FeHSO4+2 + DSO4- = FeDSO4+2 + HSO4-
FeHSO4+2 + TSO4- = FeTSO4+2 + HSO4-
FeHSO4+2 + H[34S]O4- = FeH[34S]O4+2 + HSO4-
FeHSO4+2 + D[34S]O4- = FeD[34S]O4+2 + HSO4-
FeHSO4+2 + T[34S]O4- = FeT[34S]O4+2 + HSO4-
#
# Added Fe(SO4)2- reactions 16Dec09
# Revised 17Dec09
#
Fe(SO4)2- + SO4-2 + [34S]O4-2 = FeSO4[34S]O4- + 2SO4-2
Fe(SO4)2- + [34S]O4-2 + SO4-2 = Fe[34S]O4SO4- + 2SO4-2
Fe(SO4)2- + [34S]O4-2 + [34S]O4-2 = Fe[34S]O4[34S]O4- + 2SO4-2
#
# Missing FeHPO4+ reactions
#
#
# Missing FeH2PO4+2 reactions
#
#
# Added AlOH+2 reactions 16Dec09
#
AlOH+2 + OD- = AlOD+2 + OH-
AlOH+2 + OT- = AlOT+2 + OH-
AlOH+2 + [18O]H- = Al[18O]H+2 + OH-
AlOH+2 + [18O]D- = Al[18O]D+2 + OH-
AlOH+2 + [18O]T- = Al[18O]T+2 + OH-
#
# Added Al(OH)2+ reactions 16Dec09
# Revised 17Dec09, limited the number of species
#
Al(OH)2+ + OH- + OD- = AlOHOD+ + 2OH-
Al(OH)2+ + OH- + OT- = AlOHOT+ + 2OH-
Al(OH)2+ + OH- + [18O]H- = AlOH[18O]H+ + 2OH-
Al(OH)2+ + OH- + [18O]D- = AlOH[18O]D+ + 2OH-
Al(OH)2+ + OH- + [18O]T- = AlOH[18O]T+ + 2OH-
Al(OH)2+ + OD- + OH- = AlODOH+ + 2OH-
Al(OH)2+ + OD- + OD- = AlODOD+ + 2OH-
Al(OH)2+ + OD- + OT- = AlODOT+ + 2OH-
Al(OH)2+ + OD- + [18O]H- = AlOD[18O]H+ + 2OH-
Al(OH)2+ + OD- + [18O]D- = AlOD[18O]D+ + 2OH-
Al(OH)2+ + OD- + [18O]T- = AlOD[18O]T+ + 2OH-
Al(OH)2+ + OT- + OH- = AlOTOH+ + 2OH-
Al(OH)2+ + OT- + OD- = AlOTOD+ + 2OH-
Al(OH)2+ + OT- + [18O]H- = AlOT[18O]H+ + 2OH-
Al(OH)2+ + OT- + [18O]D- = AlOT[18O]D+ + 2OH-
Al(OH)2+ + [18O]H- + OH- = Al[18O]HOH+ + 2OH-
Al(OH)2+ + [18O]H- + OD- = Al[18O]HOD+ + 2OH-
Al(OH)2+ + [18O]H- + OT- = Al[18O]HOT+ + 2OH-
Al(OH)2+ + [18O]H- + [18O]H- = Al[18O]H[18O]H+ + 2OH-
Al(OH)2+ + [18O]H- + [18O]D- = Al[18O]H[18O]D+ + 2OH-
Al(OH)2+ + [18O]H- + [18O]T- = Al[18O]H[18O]T+ + 2OH-
Al(OH)2+ + [18O]D- + OH- = Al[18O]DOH+ + 2OH-
Al(OH)2+ + [18O]D- + OD- = Al[18O]DOD+ + 2OH-
Al(OH)2+ + [18O]D- + OT- = Al[18O]DOT+ + 2OH-
Al(OH)2+ + [18O]D- + [18O]H- = Al[18O]D[18O]H+ + 2OH-
Al(OH)2+ + [18O]D- + [18O]D- = Al[18O]D[18O]D+ + 2OH-
Al(OH)2+ + [18O]D- + [18O]T- = Al[18O]D[18O]T+ + 2OH-
Al(OH)2+ + [18O]T- + OH- = Al[18O]TOH+ + 2OH-
Al(OH)2+ + [18O]T- + OD- = Al[18O]TOD+ + 2OH-
Al(OH)2+ + [18O]T- + [18O]H- = Al[18O]T[18O]H+ + 2OH-
Al(OH)2+ + [18O]T- + [18O]D- = Al[18O]T[18O]D+ + 2OH-
#
# Added Al(OH)3 reactions 16Dec09
# Revised 17Dec09, limited the number of species
#
Al(OH)3 + OH- + OH- + OD- = AlOHOHOD + 3OH-
Al(OH)3 + OH- + OH- + OT- = AlOHOHOT + 3OH-
Al(OH)3 + OH- + OH- + [18O]H- = AlOHOH[18O]H + 3OH-
Al(OH)3 + OH- + OH- + [18O]D- = AlOHOH[18O]D + 3OH-
Al(OH)3 + OH- + OH- + [18O]T- = AlOHOH[18O]T + 3OH-
Al(OH)3 + OH- + OD- + OH- = AlOHODOH + 3OH-
Al(OH)3 + OH- + OD- + OD- = AlOHODOD + 3OH-
Al(OH)3 + OH- + OD- + OT- = AlOHODOT + 3OH-
Al(OH)3 + OH- + OD- + [18O]H- = AlOHOD[18O]H + 3OH-
Al(OH)3 + OH- + OD- + [18O]D- = AlOHOD[18O]D + 3OH-
Al(OH)3 + OH- + OD- + [18O]T- = AlOHOD[18O]T + 3OH-
Al(OH)3 + OH- + OT- + OH- = AlOHOTOH + 3OH-
Al(OH)3 + OH- + OT- + OD- = AlOHOTOD + 3OH-
Al(OH)3 + OH- + OT- + [18O]H- = AlOHOT[18O]H + 3OH-
Al(OH)3 + OH- + OT- + [18O]D- = AlOHOT[18O]D + 3OH-
Al(OH)3 + OH- + [18O]H- + OH- = AlOH[18O]HOH + 3OH-
Al(OH)3 + OH- + [18O]H- + OD- = AlOH[18O]HOD + 3OH-
Al(OH)3 + OH- + [18O]H- + OT- = AlOH[18O]HOT + 3OH-
Al(OH)3 + OH- + [18O]H- + [18O]H- = AlOH[18O]H[18O]H + 3OH-
Al(OH)3 + OH- + [18O]H- + [18O]D- = AlOH[18O]H[18O]D + 3OH-
Al(OH)3 + OH- + [18O]H- + [18O]T- = AlOH[18O]H[18O]T + 3OH-
Al(OH)3 + OH- + [18O]D- + OH- = AlOH[18O]DOH + 3OH-
Al(OH)3 + OH- + [18O]D- + OD- = AlOH[18O]DOD + 3OH-
Al(OH)3 + OH- + [18O]D- + OT- = AlOH[18O]DOT + 3OH-
Al(OH)3 + OH- + [18O]D- + [18O]H- = AlOH[18O]D[18O]H + 3OH-
Al(OH)3 + OH- + [18O]D- + [18O]D- = AlOH[18O]D[18O]D + 3OH-
Al(OH)3 + OH- + [18O]D- + [18O]T- = AlOH[18O]D[18O]T + 3OH-
Al(OH)3 + OH- + [18O]T- + OH- = AlOH[18O]TOH + 3OH-
Al(OH)3 + OH- + [18O]T- + OD- = AlOH[18O]TOD + 3OH-
Al(OH)3 + OH- + [18O]T- + [18O]H- = AlOH[18O]T[18O]H + 3OH-
Al(OH)3 + OH- + [18O]T- + [18O]D- = AlOH[18O]T[18O]D + 3OH-
Al(OH)3 + OD- + OH- + OH- = AlODOHOH + 3OH-
Al(OH)3 + OD- + OH- + OD- = AlODOHOD + 3OH-
Al(OH)3 + OD- + OH- + OT- = AlODOHOT + 3OH-
Al(OH)3 + OD- + OH- + [18O]H- = AlODOH[18O]H + 3OH-
Al(OH)3 + OD- + OH- + [18O]D- = AlODOH[18O]D + 3OH-
Al(OH)3 + OD- + OH- + [18O]T- = AlODOH[18O]T + 3OH-
Al(OH)3 + OD- + OD- + OH- = AlODODOH + 3OH-
Al(OH)3 + OD- + OD- + OT- = AlODODOT + 3OH-
Al(OH)3 + OD- + OD- + [18O]H- = AlODOD[18O]H + 3OH-
Al(OH)3 + OD- + OD- + [18O]T- = AlODOD[18O]T + 3OH-
Al(OH)3 + OD- + OT- + OH- = AlODOTOH + 3OH-
Al(OH)3 + OD- + OT- + OD- = AlODOTOD + 3OH-
Al(OH)3 + OD- + OT- + [18O]H- = AlODOT[18O]H + 3OH-
Al(OH)3 + OD- + OT- + [18O]D- = AlODOT[18O]D + 3OH-
Al(OH)3 + OD- + [18O]H- + OH- = AlOD[18O]HOH + 3OH-
Al(OH)3 + OD- + [18O]H- + OD- = AlOD[18O]HOD + 3OH-
Al(OH)3 + OD- + [18O]H- + OT- = AlOD[18O]HOT + 3OH-
Al(OH)3 + OD- + [18O]H- + [18O]H- = AlOD[18O]H[18O]H + 3OH-
Al(OH)3 + OD- + [18O]H- + [18O]D- = AlOD[18O]H[18O]D + 3OH-
Al(OH)3 + OD- + [18O]H- + [18O]T- = AlOD[18O]H[18O]T + 3OH-
Al(OH)3 + OD- + [18O]D- + OH- = AlOD[18O]DOH + 3OH-
```

```
Al(OH)3 + OD- + [18O]D- + OT- = AlOD[18O]DOT + 3OH-
Al(OH)3 + OD- + [18O]D- + [18O]H- = AlOD[18O]D[18O]H + 3OH-
Al(OH)3 + OD- + [18O]T- + OH- = AlOD[18O]TOH + 3OH-
Al(OH)3 + OD- + [18O]T- + OD- = AlOD[18O]TOD + 3OH-
Al(OH)3 + OD- + [18O]T- + [18O]H- = AlOD[18O]T[18O]H + 3OH-
Al(OH)3 + OT- + OH- + OH- = AlOTOHOH + 3OH-
Al(OH)3 + OT- + OH- + OD- = AlOTOHOD + 3OH-
Al(OH)3 + OT- + OH- + [18O]H- = AlOTOH[18O]H + 3OH-
Al(OH)3 + OT- + OH- + [18O]D- = AlOTOH[18O]D + 3OH-
Al(OH)3 + OT- + OD- + OH- = AlOTODOH + 3OH-
Al(OH)3 + OT- + OD- + OD- = AlOTODOD + 3OH-
Al(OH)3 + OT- + OD- + [18O]H- = AlOTOD[18O]H + 3OH-
Al(OH)3 + OT- + OD- + [18O]D- = AlOTOD[18O]D + 3OH-
Al(OH)3 + OT- + [18O]H- + OH- = AlOT[18O]HOH + 3OH-
Al(OH)3 + OT- + [18O]H- + OD- = AlOT[18O]HOD + 3OH-
Al(OH)3 + OT- + [18O]H- + [18O]H- = AlOT[18O]H[18O]H + 3OH-
Al(OH)3 + OT- + [18O]H- + [18O]D- = AlOT[18O]H[18O]D + 3OH-
Al(OH)3 + OT- + [18O]D- + OH- = AlOT[18O]DOH + 3OH-
Al(OH)3 + OT- + [18O]D- + OD- = AlOT[18O]DOD + 3OH-
Al(OH)3 + OT- + [18O]D- + [18O]H- = AlOT[18O]D[18O]H + 3OH-
Al(OH)3 + [18O]H- + OH- + OH- = Al[18O]HOHOH + 3OH-
Al(OH)3 + [18O]H- + OH- + OD- = Al[18O]HOHOD + 3OH-
Al(OH)3 + [18O]H- + OH- + OT- = Al[18O]HOHOT + 3OH-
Al(OH)3 + [18O]H- + OH- + [18O]H- = Al[18O]HOH[18O]H + 3OH-
Al(OH)3 + [18O]H- + OH- + [18O]D- = Al[18O]HOH[18O]D + 3OH-
Al(OH)3 + [18O]H- + OH- + [18O]T- = Al[18O]HOH[18O]T + 3OH-
Al(OH)3 + [18O]H- + OD- + OH- = Al[18O]HODOH + 3OH-
Al(OH)3 + [18O]H- + OD- + OD- = Al[18O]HODOD + 3OH-
Al(OH)3 + [18O]H- + OD- + OT- = Al[18O]HODOT + 3OH-
Al(OH)3 + [18O]H- + OD- + [18O]H- = Al[18O]HOD[18O]H + 3OH-
Al(OH)3 + [18O]H- + OD- + [18O]D- = Al[18O]HOD[18O]D + 3OH-
Al(OH)3 + [18O]H- + OD- + [18O]T- = Al[18O]HOD[18O]T + 3OH-
Al(OH)3 + [18O]H- + OT- + OH- = Al[18O]HOTOH + 3OH-
Al(OH)3 + [18O]H- + OT- + OD- = Al[18O]HOTOD + 3OH-
Al(OH)3 + [18O]H- + OT- + [18O]H- = Al[18O]HOT[18O]H + 3OH-
Al(OH)3 + [18O]H- + OT- + [18O]D- = Al[18O]HOT[18O]D + 3OH-
Al(OH)3 + [18O]H- + [18O]H- + OH- = Al[18O]H[18O]HOH + 3OH-
Al(OH)3 + [18O]H- + [18O]H- + OD- = Al[18O]H[18O]HOD + 3OH-
Al(OH)3 + [18O]H- + [18O]H- + OT- = Al[18O]H[18O]HOT + 3OH-
Al(OH)3 + [18O]H- + [18O]D- + OH- = Al[18O]H[18O]DOH + 3OH-
Al(OH)3 + [18O]H- + [18O]D- + OD- = Al[18O]H[18O]DOD + 3OH-
Al(OH)3 + [18O]H- + [18O]D- + OT- = Al[18O]H[18O]DOT + 3OH-
Al(OH)3 + [18O]H- + [18O]T- + OH- = Al[18O]H[18O]TOH + 3OH-
Al(OH)3 + [18O]H- + [18O]T- + OD- = Al[18O]H[18O]TOD + 3OH-
Al(OH)3 + [18O]D- + OH- + OH- = Al[18O]DOHOH + 3OH-
Al(OH)3 + [18O]D- + OH- + OD- = Al[18O]DOHOD + 3OH-
Al(OH)3 + [18O]D- + OH- + OT- = Al[18O]DOHOT + 3OH-
Al(OH)3 + [18O]D- + OH- + [18O]H- = Al[18O]DOH[18O]H + 3OH-
Al(OH)3 + [18O]D- + OH- + [18O]D- = Al[18O]DOH[18O]D + 3OH-
Al(OH)3 + [18O]D- + OH- + [18O]T- = Al[18O]DOH[18O]T + 3OH-
Al(OH)3 + [18O]D- + OD- + OH- = Al[18O]DODOH + 3OH-
Al(OH)3 + [18O]D- + OD- + OT- = Al[18O]DODOT + 3OH-
Al(OH)3 + [18O]D- + OD- + [18O]H- = Al[18O]DOD[18O]H + 3OH-
Al(OH)3 + [18O]D- + OT- + OH- = Al[18O]DOTOH + 3OH-
Al(OH)3 + [18O]D- + OT- + OD- = Al[18O]DOTOD + 3OH-
Al(OH)3 + [18O]D- + OT- + [18O]H- = Al[18O]DOT[18O]H + 3OH-
Al(OH)3 + [18O]D- + [18O]H- + OH- = Al[18O]D[18O]HOH + 3OH-
Al(OH)3 + [18O]D- + [18O]H- + OD- = Al[18O]D[18O]HOD + 3OH-
Al(OH)3 + [18O]D- + [18O]H- + OT- = Al[18O]D[18O]HOT + 3OH-
Al(OH)3 + [18O]D- + [18O]D- + OH- = Al[18O]D[18O]DOH + 3OH-
Al(OH)3 + [18O]D- + [18O]T- + OH- = Al[18O]D[18O]TOH + 3OH-
Al(OH)3 + [18O]T- + OH- + OH- = Al[18O]TOHOH + 3OH-
Al(OH)3 + [18O]T- + OH- + OD- = Al[18O]TOHOD + 3OH-
Al(OH)3 + [18O]T- + OH- + [18O]H- = Al[18O]TOH[18O]H + 3OH-
Al(OH)3 + [18O]T- + OH- + [18O]D- = Al[18O]TOH[18O]D + 3OH-
Al(OH)3 + [18O]T- + OD- + OH- = Al[18O]TODOH + 3OH-
Al(OH)3 + [18O]T- + OD- + OD- = Al[18O]TODOD + 3OH-
Al(OH)3 + [18O]T- + OD- + [18O]H- = Al[18O]TOD[18O]H + 3OH-
Al(OH)3 + [18O]T- + [18O]H- + OH- = Al[18O]T[18O]HOH + 3OH-
Al(OH)3 + [18O]T- + [18O]H- + OD- = Al[18O]T[18O]HOD + 3OH-
Al(OH)3 + [18O]T- + [18O]D- + OH- = Al[18O]T[18O]DOH + 3OH-
#
# Added Al(OH)4- reactions 16Dec09
# Revised 17Dec09, limited the number of species
#
Al(OH)4- + OH- + OH- + OH- + OD- = AlOHOHOHOD- + 4OH-
Al(OH)4- + OH- + OH- + OH- + OT- = AlOHOHOHOT- + 4OH-
Al(OH)4- + OH- + OH- + OH- + [18O]H- = AlOHOHOH[18O]H- + 4OH-
Al(OH)4- + OH- + OH- + OH- + [18O]D- = AlOHOHOH[18O]D- + 4OH-
Al(OH)4- + OH- + OH- + OH- + [18O]T- = AlOHOHOH[18O]T- + 4OH-
Al(OH)4- + OH- + OH- + OD- + OH- = AlOHOHODOH- + 4OH-
Al(OH)4- + OH- + OH- + OD- + OD- = AlOHOHODOD- + 4OH-
Al(OH)4- + OH- + OH- + OD- + OT- = AlOHOHODOT- + 4OH-
Al(OH)4- + OH- + OH- + OD- + [18O]H- = AlOHOHOD[18O]H- + 4OH-
Al(OH)4- + OH- + OH- + OD- + [18O]D- = AlOHOHOD[18O]D- + 4OH-
Al(OH)4- + OH- + OH- + OD- + [18O]T- = AlOHOHOD[18O]T- + 4OH-
Al(OH)4- + OH- + OH- + OT- + OH- = AlOHOHOTOH- + 4OH-
Al(OH)4- + OH- + OH- + OT- + OD- = AlOHOHOTOD- + 4OH-
Al(OH)4- + OH- + OH- + OT- + [18O]H- = AlOHOHOT[18O]H- + 4OH-
Al(OH)4- + OH- + OH- + OT- + [18O]D- = AlOHOHOT[18O]D- + 4OH-
Al(OH)4- + OH- + OH- + [18O]H- + OH- = AlOHOH[18O]HOH- + 4OH-
Al(OH)4- + OH- + OH- + [18O]H- + OD- = AlOHOH[18O]HOD- + 4OH-
Al(OH)4- + OH- + OH- + [18O]H- + OT- = AlOHOH[18O]HOT- + 4OH-
Al(OH)4- + OH- + OH- + [18O]H- + [18O]H- = AlOHOH[18O]H[18O]H- + 4OH-
Al(OH)4- + OH- + OH- + [18O]H- + [18O]D- = AlOHOH[18O]H[18O]D- + 4OH-
Al(OH)4- + OH- + OH- + [18O]H- + [18O]T- = AlOHOH[18O]H[18O]T- + 4OH-
Al(OH)4- + OH- + OH- + [18O]D- + OH- = AlOHOH[18O]DOH- + 4OH-
Al(OH)4- + OH- + OH- + [18O]D- + OD- = AlOHOH[18O]DOD- + 4OH-
Al(OH)4- + OH- + OH- + [18O]D- + OT- = AlOHOH[18O]DOT- + 4OH-
Al(OH)4- + OH- + OH- + [18O]D- + [18O]H- = AlOHOH[18O]D[18O]H- + 4OH-
Al(OH)4- + OH- + OH- + [18O]D- + [18O]D- = AlOHOH[18O]D[18O]D- + 4OH-
Al(OH)4- + OH- + OH- + [18O]D- + [18O]T- = AlOHOH[18O]D[18O]T- + 4OH-
Al(OH)4- + OH- + OH- + [18O]T- + OH- = AlOHOH[18O]TOH- + 4OH-
Al(OH)4- + OH- + OH- + [18O]T- + OD- = AlOHOH[18O]TOD- + 4OH-
Al(OH)4- + OH- + OH- + [18O]T- + [18O]H- = AlOHOH[18O]T[18O]H- + 4OH-
Al(OH)4- + OH- + OH- + [18O]T- + [18O]D- = AlOHOH[18O]T[18O]D- + 4OH-
Al(OH)4- + OH- + OH- + OD- + OH- = AlOHODOHOH- + 4OH-
Al(OH)4- + OH- + OD- + OH- + OD- = AlOHODOHOD- + 4OH-
Al(OH)4- + OH- + OD- + OH- + OT- = AlOHODOHOT- + 4OH-
Al(OH)4- + OH- + OD- + OH- + [18O]H- = AlOHODOH[18O]H- + 4OH-
Al(OH)4- + OH- + OD- + OH- + [18O]D- = AlOHODOH[18O]D- + 4OH-
Al(OH)4- + OH- + OD- + OH- + [18O]T- = AlOHODOH[18O]T- + 4OH-
Al(OH)4- + OH- + OD- + OD- + OH- = AlOHODODOH- + 4OH-
Al(OH)4- + OH- + OD- + OD- + OT- = AlOHODODOT- + 4OH-
Al(OH)4- + OH- + OD- + OD- + [18O]H- = AlOHODOD[18O]H- + 4OH-
Al(OH)4- + OH- + OD- + OD- + [18O]T- = AlOHODOD[18O]T- + 4OH-
```

```
Al(OH)4- + OH- + OD- + OT- + OH- = AlOHODOTOH- + 4OH-
Al(OH)4- + OH- + OD- + OT- + OD- = AlOHODOTOD- + 4OH-
Al(OH)4- + OH- + OD- + OT- + [18O]H- = AlOHODOT[18O]H- + 4OH-
Al(OH)4- + OH- + OD- + OT- + [18O]D- = AlOHODOT[18O]D- + 4OH-
Al(OH)4- + OH- + OD- + [18O]H- + OH- = AlOHOD[18O]HOH- + 4OH-
Al(OH)4- + OH- + OD- + [18O]H- + OD- = AlOHOD[18O]HOD- + 4OH-
Al(OH)4- + OH- + OD- + [18O]H- + OT- = AlOHOD[18O]HOT- + 4OH-
Al(OH)4- + OH- + OD- + [18O]H- + [18O]H- = AlOHOD[18O]H[18O]H- + 4OH-
Al(OH)4- + OH- + OD- + [18O]H- + [18O]D- = AlOHOD[18O]H[18O]D- + 4OH-
Al(OH)4- + OH- + OD- + [18O]H- + [18O]T- = AlOHOD[18O]H[18O]T- + 4OH-
Al(OH)4- + OH- + OD- + [18O]D- + OH- = AlOHOD[18O]DOH- + 4OH-
Al(OH)4- + OH- + OD- + [18O]D- + OT- = AlOHOD[18O]DOT- + 4OH-
Al(OH)4- + OH- + OD- + [18O]D- + [18O]H- = AlOHOD[18O]D[18O]H- + 4OH-
Al(OH)4- + OH- + OD- + [18O]T- + OH- = AlOHOD[18O]TOH- + 4OH-
Al(OH)4- + OH- + OD- + [18O]T- + OD- = AlOHOD[18O]TOD- + 4OH-
Al(OH)4- + OH- + OD- + [18O]T- + [18O]H- = AlOHOD[18O]T[18O]H- + 4OH-
Al(OH)4- + OH- + OT- + OH- + OH- = AlOHOTOHOH- + 4OH-
Al(OH)4- + OH- + OT- + OH- + OD- = AlOHOTOHOD- + 4OH-
Al(OH)4- + OH- + OT- + OH- + [18O]H- = AlOHOTOH[18O]H- + 4OH-
Al(OH)4- + OH- + OT- + OH- + [18O]D- = AlOHOTOH[18O]D- + 4OH-
Al(OH)4- + OH- + OT- + OD- + OH- = AlOHOTODOH- + 4OH-
Al(OH)4- + OH- + OT- + OD- + OD- = AlOHOTODOD- + 4OH-
Al(OH)4- + OH- + OT- + OD- + [18O]H- = AlOHOTOD[18O]H- + 4OH-
Al(OH)4- + OH- + OT- + OD- + [18O]D- = AlOHOTOD[18O]D- + 4OH-
Al(OH)4- + OH- + OT- + [18O]H- + OH- = AlOHOT[18O]HOH- + 4OH-
Al(OH)4- + OH- + OT- + [18O]H- + OD- = AlOHOT[18O]HOD- + 4OH-
Al(OH)4- + OH- + OT- + [18O]H- + [18O]H- = AlOHOT[18O]H[18O]H- + 4OH-
Al(OH)4- + OH- + OT- + [18O]H- + [18O]D- = AlOHOT[18O]H[18O]D- + 4OH-
Al(OH)4- + OH- + OT- + [18O]D- + OH- = AlOHOT[18O]DOH- + 4OH-
Al(OH)4- + OH- + OT- + [18O]D- + OD- = AlOHOT[18O]DOD- + 4OH-
Al(OH)4- + OH- + OT- + [18O]D- + [18O]H- = AlOHOT[18O]D[18O]H- + 4OH-
Al(OH)4- + OH- + [18O]H- + OH- + OH- = AlOH[18O]HOHOH- + 4OH-
Al(OH)4- + OH- + [18O]H- + OH- + OD- = AlOH[18O]HOHOD- + 4OH-
Al(OH)4- + OH- + [18O]H- + OH- + OT- = AlOH[18O]HOHOT- + 4OH-
Al(OH)4- + OH- + [18O]H- + OH- + [18O]H- = AlOH[18O]HOH[18O]H- + 4OH-
Al(OH)4- + OH- + [18O]H- + OH- + [18O]D- = AlOH[18O]HOH[18O]D- + 4OH-
Al(OH)4- + OH- + [18O]H- + OH- + [18O]T- = AlOH[18O]HOH[18O]T- + 4OH-
Al(OH)4- + OH- + [18O]H- + OD- + OH- = AlOH[18O]HODOH- + 4OH-
Al(OH)4- + OH- + [18O]H- + OD- + OD- = AlOH[18O]HODOD- + 4OH-
Al(OH)4- + OH- + [18O]H- + OD- + OT- = AlOH[18O]HODOT- + 4OH-
Al(OH)4- + OH- + [18O]H- + OD- + [18O]H- = AlOH[18O]HOD[18O]H- + 4OH-
Al(OH)4- + OH- + [18O]H- + OD- + [18O]D- = AlOH[18O]HOD[18O]D- + 4OH-
Al(OH)4- + OH- + [18O]H- + OD- + [18O]T- = AlOH[18O]HOD[18O]T- + 4OH-
Al(OH)4- + OH- + [18O]H- + OT- + OH- = AlOH[18O]HOTOH- + 4OH-
Al(OH)4- + OH- + [18O]H- + OT- + OD- = AlOH[18O]HOTOD- + 4OH-
Al(OH)4- + OH- + [18O]H- + OT- + [18O]H- = AlOH[18O]HOT[18O]H- + 4OH-
Al(OH)4- + OH- + [18O]H- + OT- + [18O]D- = AlOH[18O]HOT[18O]D- + 4OH-
Al(OH)4- + OH- + [18O]H- + [18O]H- + OH- = AlOH[18O]H[18O]HOH- + 4OH-
Al(OH)4- + OH- + [18O]H- + [18O]H- + OD- = AlOH[18O]H[18O]HOD- + 4OH-
Al(OH)4- + OH- + [18O]H- + [18O]H- + OT- = AlOH[18O]H[18O]HOT- + 4OH-
Al(OH)4- + OH- + [18O]H- + [18O]D- + OH- = AlOH[18O]H[18O]DOH- + 4OH-
Al(OH)4- + OH- + [18O]H- + [18O]D- + OD- = AlOH[18O]H[18O]DOD- + 4OH-
Al(OH)4- + OH- + [18O]H- + [18O]D- + OT- = AlOH[18O]H[18O]DOT- + 4OH-
Al(OH)4- + OH- + [18O]H- + [18O]T- + OH- = AlOH[18O]H[18O]TOH- + 4OH-
Al(OH)4- + OH- + [18O]H- + [18O]T- + OD- = AlOH[18O]H[18O]TOD- + 4OH-
Al(OH)4- + OH- + [18O]D- + OH- + OH- = AlOH[18O]DOHOH- + 4OH-
Al(OH)4- + OH- + [18O]D- + OH- + OD- = AlOH[18O]DOHOD- + 4OH-
Al(OH)4- + OH- + [18O]D- + OH- + OT- = AlOH[18O]DOHOT- + 4OH-
Al(OH)4- + OH- + [18O]D- + OH- + [18O]H- = AlOH[18O]DOH[18O]H- + 4OH-
Al(OH)4- + OH- + [18O]D- + OH- + [18O]D- = AlOH[18O]DOH[18O]D- + 4OH-
Al(OH)4- + OH- + [18O]D- + OH- + [18O]T- = AlOH[18O]DOH[18O]T- + 4OH-
Al(OH)4- + OH- + [18O]D- + OD- + OH- = AlOH[18O]DODOH- + 4OH-
Al(OH)4- + OH- + [18O]D- + OD- + OT- = AlOH[18O]DODOT- + 4OH-
Al(OH)4- + OH- + [18O]D- + OD- + [18O]H- = AlOH[18O]DOD[18O]H- + 4OH-
Al(OH)4- + OH- + [18O]D- + OT- + OH- = AlOH[18O]DOTOH- + 4OH-
Al(OH)4- + OH- + [18O]D- + OT- + OD- = AlOH[18O]DOTOD- + 4OH-
Al(OH)4- + OH- + [18O]D- + OT- + [18O]H- = AlOH[18O]DOT[18O]H- + 4OH-
Al(OH)4- + OH- + [18O]D- + [18O]H- + OH- = AlOH[18O]D[18O]HOH- + 4OH-
Al(OH)4- + OH- + [18O]D- + [18O]H- + OD- = AlOH[18O]D[18O]HOD- + 4OH-
Al(OH)4- + OH- + [18O]D- + [18O]H- + OT- = AlOH[18O]D[18O]HOT- + 4OH-
Al(OH)4- + OH- + [18O]D- + [18O]D- + OH- = AlOH[18O]D[18O]DOH- + 4OH-
Al(OH)4- + OH- + [18O]D- + [18O]T- + OH- = AlOH[18O]D[18O]TOH- + 4OH-
Al(OH)4- + OH- + [18O]T- + OH- + OH- = AlOH[18O]TOHOH- + 4OH-
Al(OH)4- + OH- + [18O]T- + OH- + OD- = AlOH[18O]TOHOD- + 4OH-
Al(OH)4- + OH- + [18O]T- + OH- + [18O]H- = AlOH[18O]TOH[18O]H- + 4OH-
Al(OH)4- + OH- + [18O]T- + OH- + [18O]D- = AlOH[18O]TOH[18O]D- + 4OH-
Al(OH)4- + OH- + [18O]T- + OD- + OH- = AlOH[18O]TODOH- + 4OH-
Al(OH)4- + OH- + [18O]T- + OD- + OD- = AlOH[18O]TODOD- + 4OH-
Al(OH)4- + OH- + [18O]T- + OD- + [18O]H- = AlOH[18O]TOD[18O]H- + 4OH-
Al(OH)4- + OH- + [18O]T- + [18O]H- + OH- = AlOH[18O]T[18O]HOH- + 4OH-
Al(OH)4- + OH- + [18O]T- + [18O]H- + OD- = AlOH[18O]T[18O]HOD- + 4OH-
Al(OH)4- + OH- + [18O]T- + [18O]D- + OH- = AlOH[18O]T[18O]DOH- + 4OH-
Al(OH)4- + OD- + OH- + OH- + OH- = AlODOHOHOH- + 4OH-
Al(OH)4- + OD- + OH- + OH- + OD- = AlODOHOHOD- + 4OH-
Al(OH)4- + OD- + OH- + OH- + OT- = AlODOHOHOT- + 4OH-
Al(OH)4- + OD- + OH- + OH- + [18O]H- = AlODOHOH[18O]H- + 4OH-
Al(OH)4- + OD- + OH- + OH- + [18O]D- = AlODOHOH[18O]D- + 4OH-
Al(OH)4- + OD- + OH- + OH- + [18O]T- = AlODOHOH[18O]T- + 4OH-
Al(OH)4- + OD- + OH- + OD- + OH- = AlODOHODOH- + 4OH-
Al(OH)4- + OD- + OH- + OD- + OT- = AlODOHODOT- + 4OH-
Al(OH)4- + OD- + OH- + OD- + [18O]H- = AlODOHOD[18O]H- + 4OH-
Al(OH)4- + OD- + OH- + OD- + [18O]T- = AlODOHOD[18O]T- + 4OH-
Al(OH)4- + OD- + OH- + OT- + OH- = AlODOHOTOH- + 4OH-
Al(OH)4- + OD- + OH- + OT- + OD- = AlODOHOTOD- + 4OH-
Al(OH)4- + OD- + OH- + OT- + [18O]H- = AlODOHOT[18O]H- + 4OH-
Al(OH)4- + OD- + OH- + OT- + [18O]D- = AlODOHOT[18O]D- + 4OH-
Al(OH)4- + OD- + OH- + [18O]H- + OH- = AlODOH[18O]HOH- + 4OH-
Al(OH)4- + OD- + OH- + [18O]H- + OD- = AlODOH[18O]HOD- + 4OH-
Al(OH)4- + OD- + OH- + [18O]H- + OT- = AlODOH[18O]HOT- + 4OH-
Al(OH)4- + OD- + OH- + [18O]H- + [18O]H- = AlODOH[18O]H[18O]H- + 4OH-
Al(OH)4- + OD- + OH- + [18O]H- + [18O]D- = AlODOH[18O]H[18O]D- + 4OH-
Al(OH)4- + OD- + OH- + [18O]H- + [18O]T- = AlODOH[18O]H[18O]T- + 4OH-
Al(OH)4- + OD- + OH- + [18O]D- + OH- = AlODOH[18O]DOH- + 4OH-
Al(OH)4- + OD- + OH- + [18O]D- + OT- = AlODOH[18O]DOT- + 4OH-
Al(OH)4- + OD- + OH- + [18O]D- + [18O]H- = AlODOH[18O]D[18O]H- + 4OH-
Al(OH)4- + OD- + OH- + [18O]T- + OH- = AlODOH[18O]TOH- + 4OH-
Al(OH)4- + OD- + OH- + [18O]T- + OD- = AlODOH[18O]TOD- + 4OH-
Al(OH)4- + OD- + OH- + [18O]T- + [18O]H- = AlODOH[18O]T[18O]H- + 4OH-
Al(OH)4- + OD- + OD- + OH- + OH- = AlODODOHOH- + 4OH-
Al(OH)4- + OD- + OD- + OH- + OT- = AlODODOHOT- + 4OH-
Al(OH)4- + OD- + OD- + OH- + [18O]H- = AlODODOH[18O]H- + 4OH-
Al(OH)4- + OD- + OD- + OH- + [18O]T- = AlODODOH[18O]T- + 4OH-
Al(OH)4- + OD- + OD- + OT- + OH- = AlODODOTOH- + 4OH-
Al(OH)4- + OD- + OD- + OT- + [18O]H- = AlODODOT[18O]H- + 4OH-
Al(OH)4- + OD- + OD- + [18O]H- + OH- = AlODOD[18O]HOH- + 4OH-
Al(OH)4- + OD- + OD- + [18O]H- + OT- = AlODOD[18O]HOT- + 4OH-
```

```
|Al(OH)4- + OD- + OD- + [18O]H- + [18O]H- = AlODOD[18O]H[18O]H- + 4OH-                |
|Al(OH)4- + OD- + OD- + [18O]T- + OH- = AlODOD[18O]TOH- + 4OH-                       |
|Al(OH)4- + OD- + OT- + OH- + OH- = AlODOTOHOH- + 4OH-                              |
|Al(OH)4- + OD- + OT- + OH- + OD- = AlODOTOHOD- + 4OH-                              |
|Al(OH)4- + OD- + OT- + OH- + [18O]H- = AlODOTOH[18O]H- + 4OH-                      |
|Al(OH)4- + OD- + OT- + OH- + [18O]D- = AlODOTOH[18O]D- + 4OH-                      |
|Al(OH)4- + OD- + OT- + OD- + OH- = AlODOTODOH- + 4OH-                              |
|Al(OH)4- + OD- + OT- + OD- + [18O]H- = AlODOTOD[18O]H- + 4OH-                      |
|Al(OH)4- + OD- + OT- + [18O]H- + OH- = AlODOT[18O]HOH- + 4OH-                      |
|Al(OH)4- + OD- + OT- + [18O]H- + OD- = AlODOT[18O]HOD- + 4OH-                      |
|Al(OH)4- + OD- + OT- + [18O]H- + [18O]H- = AlODOT[18O]H[18O]H- + 4OH-              |
|Al(OH)4- + OD- + OT- + [18O]D- + OH- = AlODOT[18O]DOH- + 4OH-                      |
|Al(OH)4- + OD- + [18O]H- + OH- + OH- = AlOD[18O]HOHOH- + 4OH-                      |
|Al(OH)4- + OD- + [18O]H- + OH- + OD- = AlOD[18O]HOHOD- + 4OH-                      |
|Al(OH)4- + OD- + [18O]H- + OH- + OT- = AlOD[18O]HOHOT- + 4OH-                      |
|Al(OH)4- + OD- + [18O]H- + OH- + [18O]H- = AlOD[18O]HOH[18O]H- + 4OH-              |
|Al(OH)4- + OD- + [18O]H- + OH- + [18O]D- = AlOD[18O]HOH[18O]D- + 4OH-              |
|Al(OH)4- + OD- + [18O]H- + OH- + [18O]T- = AlOD[18O]HOH[18O]T- + 4OH-              |
|Al(OH)4- + OD- + [18O]H- + OD- + OH- = AlOD[18O]HODOH- + 4OH-                      |
|Al(OH)4- + OD- + [18O]H- + OD- + OT- = AlOD[18O]HODOT- + 4OH-                      |
|Al(OH)4- + OD- + [18O]H- + OD- + [18O]H- = AlOD[18O]HOD[18O]H- + 4OH-              |
|Al(OH)4- + OD- + [18O]H- + OT- + OH- = AlOD[18O]HOTOH- + 4OH-                      |
|Al(OH)4- + OD- + [18O]H- + OT- + OD- = AlOD[18O]HOTOD- + 4OH-                      |
|Al(OH)4- + OD- + [18O]H- + OT- + [18O]H- = AlOD[18O]HOT[18O]H- + 4OH-              |
|Al(OH)4- + OD- + [18O]H- + [18O]H- + OH- = AlOD[18O]H[18O]HOH- + 4OH-              |
|Al(OH)4- + OD- + [18O]H- + [18O]H- + OD- = AlOD[18O]H[18O]HOD- + 4OH-              |
|Al(OH)4- + OD- + [18O]H- + [18O]H- + OT- = AlOD[18O]H[18O]HOT- + 4OH-              |
|Al(OH)4- + OD- + [18O]H- + [18O]D- + OH- = AlOD[18O]H[18O]DOH- + 4OH-              |
|Al(OH)4- + OD- + [18O]H- + [18O]T- + OH- = AlOD[18O]H[18O]TOH- + 4OH-              |
|Al(OH)4- + OD- + [18O]D- + OH- + OH- = AlOD[18O]DOHOH- + 4OH-                      |
|Al(OH)4- + OD- + [18O]D- + OH- + OT- = AlOD[18O]DOHOT- + 4OH-                      |
|Al(OH)4- + OD- + [18O]D- + OH- + [18O]H- = AlOD[18O]DOH[18O]H- + 4OH-              |
|Al(OH)4- + OD- + [18O]D- + OT- + OH- = AlOD[18O]DOTOH- + 4OH-                      |
|Al(OH)4- + OD- + [18O]D- + [18O]H- + OH- = AlOD[18O]D[18O]HOH- + 4OH-              |
|Al(OH)4- + OD- + [18O]T- + OH- + OH- = AlOD[18O]TOHOH- + 4OH-                      |
|Al(OH)4- + OD- + [18O]T- + OH- + OD- = AlOD[18O]TOHOD- + 4OH-                      |
|Al(OH)4- + OD- + [18O]T- + OH- + [18O]H- = AlOD[18O]TOH[18O]H- + 4OH-              |
|Al(OH)4- + OD- + [18O]T- + OD- + OH- = AlOD[18O]TODOH- + 4OH-                      |
|Al(OH)4- + OD- + [18O]T- + [18O]H- + OH- = AlOD[18O]T[18O]HOH- + 4OH-              |
|Al(OH)4- + OT- + OH- + OH- + OH- = AlOTOHOHOH- + 4OH-                              |
|Al(OH)4- + OT- + OH- + OH- + OD- = AlOTOHOHOD- + 4OH-                              |
|Al(OH)4- + OT- + OH- + OH- + [18O]H- = AlOTOHOH[18O]H- + 4OH-                      |
|Al(OH)4- + OT- + OH- + OH- + [18O]D- = AlOTOHOH[18O]D- + 4OH-                      |
|Al(OH)4- + OT- + OH- + OD- + OH- = AlOTOHODOH- + 4OH-                              |
|Al(OH)4- + OT- + OH- + OD- + OD- = AlOTOHODOD- + 4OH-                              |
|Al(OH)4- + OT- + OH- + OD- + [18O]H- = AlOTOHOD[18O]H- + 4OH-                      |
|Al(OH)4- + OT- + OH- + OD- + [18O]D- = AlOTOHOD[18O]D- + 4OH-                      |
|Al(OH)4- + OT- + OH- + [18O]H- + OH- = AlOTOH[18O]HOH- + 4OH-                      |
|Al(OH)4- + OT- + OH- + [18O]H- + OD- = AlOTOH[18O]HOD- + 4OH-                      |
|Al(OH)4- + OT- + OH- + [18O]H- + [18O]H- = AlOTOH[18O]H[18O]H- + 4OH-              |
|Al(OH)4- + OT- + OH- + [18O]H- + [18O]D- = AlOTOH[18O]H[18O]D- + 4OH-              |
|Al(OH)4- + OT- + OH- + [18O]D- + OH- = AlOTOH[18O]DOH- + 4OH-                      |
|Al(OH)4- + OT- + OH- + [18O]D- + OD- = AlOTOH[18O]DOD- + 4OH-                      |
|Al(OH)4- + OT- + OH- + [18O]D- + [18O]H- = AlOTOH[18O]D[18O]H- + 4OH-              |
|Al(OH)4- + OT- + OD- + OH- + OH- = AlOTODOHOH- + 4OH-                              |
|Al(OH)4- + OT- + OD- + OH- + OD- = AlOTODOHOD- + 4OH-                              |
|Al(OH)4- + OT- + OD- + OH- + [18O]H- = AlOTODOH[18O]H- + 4OH-                      |
|Al(OH)4- + OT- + OD- + OH- + [18O]D- = AlOTODOH[18O]D- + 4OH-                      |
|Al(OH)4- + OT- + OD- + OD- + OH- = AlOTODODOH- + 4OH-                              |
|Al(OH)4- + OT- + OD- + OD- + [18O]H- = AlOTODOD[18O]H- + 4OH-                      |
|Al(OH)4- + OT- + OD- + [18O]H- + OH- = AlOTOD[18O]HOH- + 4OH-                      |
|Al(OH)4- + OT- + OD- + [18O]H- + OD- = AlOTOD[18O]HOD- + 4OH-                      |
|Al(OH)4- + OT- + OD- + [18O]H- + [18O]H- = AlOTOD[18O]H[18O]H- + 4OH-              |
|Al(OH)4- + OT- + OD- + [18O]D- + OH- = AlOTOD[18O]DOH- + 4OH-                      |
|Al(OH)4- + OT- + [18O]H- + OH- + OH- = AlOT[18O]HOHOH- + 4OH-                      |
|Al(OH)4- + OT- + [18O]H- + OH- + OD- = AlOT[18O]HOHOD- + 4OH-                      |
|Al(OH)4- + OT- + [18O]H- + OH- + [18O]H- = AlOT[18O]HOH[18O]H- + 4OH-              |
|Al(OH)4- + OT- + [18O]H- + OH- + [18O]D- = AlOT[18O]HOH[18O]D- + 4OH-              |
|Al(OH)4- + OT- + [18O]H- + OD- + OH- = AlOT[18O]HODOH- + 4OH-                      |
|Al(OH)4- + OT- + [18O]H- + OD- + OD- = AlOT[18O]HODOD- + 4OH-                      |
|Al(OH)4- + OT- + [18O]H- + OD- + [18O]H- = AlOT[18O]HOD[18O]H- + 4OH-              |
|Al(OH)4- + OT- + [18O]H- + [18O]H- + OH- = AlOT[18O]H[18O]HOH- + 4OH-              |
|Al(OH)4- + OT- + [18O]H- + [18O]H- + OD- = AlOT[18O]H[18O]HOD- + 4OH-              |
|Al(OH)4- + OT- + [18O]H- + [18O]D- + OH- = AlOT[18O]H[18O]DOH- + 4OH-              |
|Al(OH)4- + OT- + [18O]D- + OH- + OH- = AlOT[18O]DOHOH- + 4OH-                      |
|Al(OH)4- + OT- + [18O]D- + OH- + OD- = AlOT[18O]DOHOD- + 4OH-                      |
|Al(OH)4- + OT- + [18O]D- + OH- + [18O]H- = AlOT[18O]DOH[18O]H- + 4OH-              |
|Al(OH)4- + OT- + [18O]D- + OD- + OH- = AlOT[18O]DODOH- + 4OH-                      |
|Al(OH)4- + OT- + [18O]D- + [18O]H- + OH- = AlOT[18O]D[18O]HOH- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + OH- + OH- = Al[18O]HOHOHOH- + 4OH-                      |
|Al(OH)4- + [18O]H- + OH- + OH- + OD- = Al[18O]HOHOHOD- + 4OH-                      |
|Al(OH)4- + [18O]H- + OH- + OH- + OT- = Al[18O]HOHOHOT- + 4OH-                      |
|Al(OH)4- + [18O]H- + OH- + OH- + [18O]H- = Al[18O]HOHOH[18O]H- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + OH- + [18O]D- = Al[18O]HOHOH[18O]D- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + OH- + [18O]T- = Al[18O]HOHOH[18O]T- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + OD- + OH- = Al[18O]HOHODOH- + 4OH-                      |
|Al(OH)4- + [18O]H- + OH- + OD- + OD- = Al[18O]HOHODOD- + 4OH-                      |
|Al(OH)4- + [18O]H- + OH- + OD- + OT- = Al[18O]HOHODOT- + 4OH-                      |
|Al(OH)4- + [18O]H- + OH- + OD- + [18O]H- = Al[18O]HOHOD[18O]H- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + OD- + [18O]D- = Al[18O]HOHOD[18O]D- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + OD- + [18O]T- = Al[18O]HOHOD[18O]T- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + OT- + OH- = Al[18O]HOHOTOH- + 4OH-                      |
|Al(OH)4- + [18O]H- + OH- + OT- + OD- = Al[18O]HOHOTOD- + 4OH-                      |
|Al(OH)4- + [18O]H- + OH- + OT- + [18O]H- = Al[18O]HOHOT[18O]H- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + OT- + [18O]D- = Al[18O]HOHOT[18O]D- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + [18O]H- + OH- = Al[18O]HOH[18O]HOH- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + [18O]H- + OD- = Al[18O]HOH[18O]HOD- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + [18O]H- + OT- = Al[18O]HOH[18O]HOT- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + [18O]D- + OH- = Al[18O]HOH[18O]DOH- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + [18O]D- + OD- = Al[18O]HOH[18O]DOD- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + [18O]D- + OT- = Al[18O]HOH[18O]DOT- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + [18O]T- + OH- = Al[18O]HOH[18O]TOH- + 4OH-              |
|Al(OH)4- + [18O]H- + OH- + [18O]T- + OD- = Al[18O]HOH[18O]TOD- + 4OH-              |
|Al(OH)4- + [18O]H- + OD- + OH- + OH- = Al[18O]HODOHOH- + 4OH-                      |
|Al(OH)4- + [18O]H- + OD- + OH- + OD- = Al[18O]HODOHOD- + 4OH-                      |
|Al(OH)4- + [18O]H- + OD- + OH- + OT- = Al[18O]HODOHOT- + 4OH-                      |
|Al(OH)4- + [18O]H- + OD- + OH- + [18O]H- = Al[18O]HODOH[18O]H- + 4OH-              |
|Al(OH)4- + [18O]H- + OD- + OH- + [18O]D- = Al[18O]HODOH[18O]D- + 4OH-              |
|Al(OH)4- + [18O]H- + OD- + OH- + [18O]T- = Al[18O]HODOH[18O]T- + 4OH-              |
|Al(OH)4- + [18O]H- + OD- + OD- + OH- = Al[18O]HODODOH- + 4OH-                      |
|Al(OH)4- + [18O]H- + OD- + OD- + OT- = Al[18O]HODODOT- + 4OH-                      |
|Al(OH)4- + [18O]H- + OD- + OD- + [18O]H- = Al[18O]HODOD[18O]H- + 4OH-              |
|Al(OH)4- + [18O]H- + OD- + OT- + OH- = Al[18O]HODOTOH- + 4OH-                      |
|Al(OH)4- + [18O]H- + OD- + OT- + OD- = Al[18O]HODOTOD- + 4OH-                      |
|Al(OH)4- + [18O]H- + OD- + OT- + [18O]H- = Al[18O]HODOT[18O]H- + 4OH-              |
|Al(OH)4- + [18O]H- + OD- + [18O]H- + OH- = Al[18O]HOD[18O]HOH- + 4OH-              |
```

```
Al(OH)4- + [18O]H- + OD- + [18O]H- + OD- = Al[18O]HOD[18O]HOD- + 4OH-
Al(OH)4- + [18O]H- + OD- + [18O]H- + OT- = Al[18O]HOD[18O]HOT- + 4OH-
Al(OH)4- + [18O]H- + OD- + [18O]D- + OH- = Al[18O]HOD[18O]DOH- + 4OH-
Al(OH)4- + [18O]H- + OD- + [18O]T- + OH- = Al[18O]HOD[18O]TOH- + 4OH-
Al(OH)4- + [18O]H- + OT- + OH- + OH- = Al[18O]HOTOHOH- + 4OH-
Al(OH)4- + [18O]H- + OT- + OH- + OD- = Al[18O]HOTOHOD- + 4OH-
Al(OH)4- + [18O]H- + OT- + OH- + [18O]H- = Al[18O]HOTOH[18O]H- + 4OH-
Al(OH)4- + [18O]H- + OT- + OH- + [18O]D- = Al[18O]HOTOH[18O]D- + 4OH-
Al(OH)4- + [18O]H- + OT- + OD- + OH- = Al[18O]HOTODOH- + 4OH-
Al(OH)4- + [18O]H- + OT- + OD- + OD- = Al[18O]HOTODOD- + 4OH-
Al(OH)4- + [18O]H- + OT- + OD- + [18O]H- = Al[18O]HOTOD[18O]H- + 4OH-
Al(OH)4- + [18O]H- + OT- + [18O]H- + OH- = Al[18O]HOT[18O]HOH- + 4OH-
Al(OH)4- + [18O]H- + OT- + [18O]H- + OD- = Al[18O]HOT[18O]HOD- + 4OH-
Al(OH)4- + [18O]H- + OT- + [18O]D- + OH- = Al[18O]HOT[18O]DOH- + 4OH-
Al(OH)4- + [18O]H- + [18O]H- + OH- + OH- = Al[18O]H[18O]HOHOH- + 4OH-
Al(OH)4- + [18O]H- + [18O]H- + OH- + OD- = Al[18O]H[18O]HOHOD- + 4OH-
Al(OH)4- + [18O]H- + [18O]H- + OH- + OT- = Al[18O]H[18O]HOHOT- + 4OH-
Al(OH)4- + [18O]H- + [18O]H- + OD- + OH- = Al[18O]H[18O]HODOH- + 4OH-
Al(OH)4- + [18O]H- + [18O]H- + OD- + OD- = Al[18O]H[18O]HODOD- + 4OH-
Al(OH)4- + [18O]H- + [18O]H- + OD- + OT- = Al[18O]H[18O]HODOT- + 4OH-
Al(OH)4- + [18O]H- + [18O]H- + OT- + OH- = Al[18O]H[18O]HOTOH- + 4OH-
Al(OH)4- + [18O]H- + [18O]H- + OT- + OD- = Al[18O]H[18O]HOTOD- + 4OH-
Al(OH)4- + [18O]H- + [18O]D- + OH- + OH- = Al[18O]H[18O]DOHOH- + 4OH-
Al(OH)4- + [18O]H- + [18O]D- + OH- + OD- = Al[18O]H[18O]DOHOD- + 4OH-
Al(OH)4- + [18O]H- + [18O]D- + OH- + OT- = Al[18O]H[18O]DOHOT- + 4OH-
Al(OH)4- + [18O]H- + [18O]D- + OD- + OH- = Al[18O]H[18O]DODOH- + 4OH-
Al(OH)4- + [18O]H- + [18O]D- + OT- + OH- = Al[18O]H[18O]DOTOH- + 4OH-
Al(OH)4- + [18O]H- + [18O]T- + OH- + OH- = Al[18O]H[18O]TOHOH- + 4OH-
Al(OH)4- + [18O]H- + [18O]T- + OH- + OD- = Al[18O]H[18O]TOHOD- + 4OH-
Al(OH)4- + [18O]H- + [18O]T- + OD- + OH- = Al[18O]H[18O]TODOH- + 4OH-
Al(OH)4- + [18O]D- + OH- + OH- + OH- = Al[18O]DOHOHOH- + 4OH-
Al(OH)4- + [18O]D- + OH- + OH- + OD- = Al[18O]DOHOHOD- + 4OH-
Al(OH)4- + [18O]D- + OH- + OH- + OT- = Al[18O]DOHOHOT- + 4OH-
Al(OH)4- + [18O]D- + OH- + OH- + [18O]H- = Al[18O]DOHOH[18O]H- + 4OH-
Al(OH)4- + [18O]D- + OH- + OH- + [18O]D- = Al[18O]DOHOH[18O]D- + 4OH-
Al(OH)4- + [18O]D- + OH- + OH- + [18O]T- = Al[18O]DOHOH[18O]T- + 4OH-
Al(OH)4- + [18O]D- + OH- + OD- + OH- = Al[18O]DOHODOH- + 4OH-
Al(OH)4- + [18O]D- + OH- + OD- + OT- = Al[18O]DOHODOT- + 4OH-
Al(OH)4- + [18O]D- + OH- + OD- + [18O]H- = Al[18O]DOHOD[18O]H- + 4OH-
Al(OH)4- + [18O]D- + OH- + OT- + OH- = Al[18O]DOHOTOH- + 4OH-
Al(OH)4- + [18O]D- + OH- + OT- + OD- = Al[18O]DOHOTOD- + 4OH-
Al(OH)4- + [18O]D- + OH- + OT- + [18O]H- = Al[18O]DOHOT[18O]H- + 4OH-
Al(OH)4- + [18O]D- + OH- + [18O]H- + OH- = Al[18O]DOH[18O]HOH- + 4OH-
Al(OH)4- + [18O]D- + OH- + [18O]H- + OD- = Al[18O]DOH[18O]HOD- + 4OH-
Al(OH)4- + [18O]D- + OH- + [18O]H- + OT- = Al[18O]DOH[18O]HOT- + 4OH-
Al(OH)4- + [18O]D- + OH- + [18O]D- + OH- = Al[18O]DOH[18O]DOH- + 4OH-
Al(OH)4- + [18O]D- + OH- + [18O]T- + OH- = Al[18O]DOH[18O]TOH- + 4OH-
Al(OH)4- + [18O]D- + OD- + OH- + OH- = Al[18O]DODOHOH- + 4OH-
Al(OH)4- + [18O]D- + OD- + OH- + OT- = Al[18O]DODOHOT- + 4OH-
Al(OH)4- + [18O]D- + OD- + OH- + [18O]H- = Al[18O]DODOH[18O]H- + 4OH-
Al(OH)4- + [18O]D- + OD- + OT- + OH- = Al[18O]DODOTOH- + 4OH-
Al(OH)4- + [18O]D- + OD- + [18O]H- + OH- = Al[18O]DOD[18O]HOH- + 4OH-
Al(OH)4- + [18O]D- + OT- + OH- + OH- = Al[18O]DOTOHOH- + 4OH-
Al(OH)4- + [18O]D- + OT- + OH- + OD- = Al[18O]DOTOHOD- + 4OH-
Al(OH)4- + [18O]D- + OT- + OH- + [18O]H- = Al[18O]DOTOH[18O]H- + 4OH-
Al(OH)4- + [18O]D- + OT- + OD- + OH- = Al[18O]DOTODOH- + 4OH-
Al(OH)4- + [18O]D- + OT- + [18O]H- + OH- = Al[18O]DOT[18O]HOH- + 4OH-
Al(OH)4- + [18O]D- + [18O]H- + OH- + OH- = Al[18O]D[18O]HOHOH- + 4OH-
Al(OH)4- + [18O]D- + [18O]H- + OH- + OD- = Al[18O]D[18O]HOHOD- + 4OH-
Al(OH)4- + [18O]D- + [18O]H- + OH- + OT- = Al[18O]D[18O]HOHOT- + 4OH-
Al(OH)4- + [18O]D- + [18O]H- + OD- + OH- = Al[18O]D[18O]HODOH- + 4OH-
Al(OH)4- + [18O]D- + [18O]H- + OT- + OH- = Al[18O]D[18O]HOTOH- + 4OH-
Al(OH)4- + [18O]D- + [18O]D- + OH- + OH- = Al[18O]D[18O]DOHOH- + 4OH-
Al(OH)4- + [18O]D- + [18O]T- + OH- + OH- = Al[18O]D[18O]TOHOH- + 4OH-
Al(OH)4- + [18O]T- + OH- + OH- + OH- = Al[18O]TOHOHOH- + 4OH-
Al(OH)4- + [18O]T- + OH- + OH- + OD- = Al[18O]TOHOHOD- + 4OH-
Al(OH)4- + [18O]T- + OH- + OH- + [18O]H- = Al[18O]TOHOH[18O]H- + 4OH-
Al(OH)4- + [18O]T- + OH- + OH- + [18O]D- = Al[18O]TOHOH[18O]D- + 4OH-
Al(OH)4- + [18O]T- + OH- + OD- + OH- = Al[18O]TOHODOH- + 4OH-
Al(OH)4- + [18O]T- + OH- + OD- + OD- = Al[18O]TOHODOD- + 4OH-
Al(OH)4- + [18O]T- + OH- + OD- + [18O]H- = Al[18O]TOHOD[18O]H- + 4OH-
Al(OH)4- + [18O]T- + OH- + [18O]H- + OH- = Al[18O]TOH[18O]HOH- + 4OH-
Al(OH)4- + [18O]T- + OH- + [18O]H- + OD- = Al[18O]TOH[18O]HOD- + 4OH-
Al(OH)4- + [18O]T- + OH- + [18O]D- + OH- = Al[18O]TOH[18O]DOH- + 4OH-
Al(OH)4- + [18O]T- + OD- + OH- + OH- = Al[18O]TODOHOH- + 4OH-
Al(OH)4- + [18O]T- + OD- + OH- + OD- = Al[18O]TODOHOD- + 4OH-
Al(OH)4- + [18O]T- + OD- + OH- + [18O]H- = Al[18O]TODOH[18O]H- + 4OH-
Al(OH)4- + [18O]T- + OD- + OD- + OH- = Al[18O]TODODOH- + 4OH-
Al(OH)4- + [18O]T- + OD- + [18O]H- + OH- = Al[18O]TOD[18O]HOH- + 4OH-
Al(OH)4- + [18O]T- + [18O]H- + OH- + OH- = Al[18O]T[18O]HOHOH- + 4OH-
Al(OH)4- + [18O]T- + [18O]H- + OH- + OD- = Al[18O]T[18O]HOHOD- + 4OH-
Al(OH)4- + [18O]T- + [18O]H- + OD- + OH- = Al[18O]T[18O]HODOH- + 4OH-
Al(OH)4- + [18O]T- + [18O]D- + OH- + OH- = Al[18O]T[18O]DOHOH- + 4OH-
#
# Added AlSO4+ reactions 16Dec09
#
AlSO4+ + [34S]O4-2 = Al[34S]O4+ + SO4-2
#
# Added Al(SO4)2- reactions 16Dec09
# Revised 17Dec09, limited the number of species
#
Al(SO4)2- + SO4-2 + [34S]O4-2 = AlSO4[34S]O4- + 2SO4-2
Al(SO4)2- + [34S]O4-2 + SO4-2 = Al[34S]O4SO4- + 2SO4-2
Al(SO4)2- + [34S]O4-2 + [34S]O4-2 = Al[34S]O4[34S]O4- + 2SO4-2
#
# Added AlHSO4+2 reactions 16Dec09
#
AlHSO4+2 + DSO4- = AlDSO4+2 + HSO4-
AlHSO4+2 + TSO4- = AlTSO4+2 + HSO4-
AlHSO4+2 + H[34S]O4- = AlH[34S]O4+2 + HSO4-
AlHSO4+2 + D[34S]O4- = AlD[34S]O4+2 + HSO4-
AlHSO4+2 + T[34S]O4- = AlT[34S]O4+2 + HSO4-
#
# Missing H4SiO4 reactions
#
#
# Missing H3SiO4- reactions
#
#
# Missing H2SiO4-2 reactions
#

#↩
     ##############################################################################################↩

PHASES
#↩
     ##############################################################################################↩
```

```
#
# Water vapor - Water
# last update March 20, 2006
# Updated September 19, 2006
# Checked September 19, 2006
#
HDO(g)
      HDO(g) + H2O(l) = H2O(g) + HDO(aq)
      -add_logk       Log_alpha_D_H2O(g)/H2O(l)              -1.0

HTO(g)
      HTO(g) + H2O(l) = H2O(g) + HTO(aq)
      -add_logk       Log_alpha_T_H2O(g)/H2O(l)              -1.0

DTO(g)
      DTO(g) + H2O(g) = HDO(g) + HTO(g)
      log_k           0.301029995663                          # log10(2)

D2O(g)
      D2O(g) + H2O(g) = 2HDO(g)
      log_k           0.602059991327962396                   # log10(4)

T2O(g)
      T2O(g) + H2O(g) = 2HTO(g)
      log_k           0.602059991327962396                   # log10(4)

H2[18O](g)
      H2[18O](g) + H2O(l) = H2O(g) + H2[18O](aq)
      -add_logk       Log_alpha_18O_H2O(g)/H2O(l)              -1.0

HD[18O](g)
      HD[18O](g) + H2O(g) = HDO(g) + H2[18O](g)
      log_k           0.0

HT[18O](g)
      HT[18O](g) + H2O(g) = HTO(g) + H2[18O](g)
      log_k           0.0

D2[18O](g)
      D2[18O](g) + H2O(g) = D2O(g) + H2[18O](g)
      log_k           0.0

T2[18O](g)
      T2[18O](g) + H2O(g) = T2O(g) + H2[18O](g)
      log_k           0.0

DT[18O](g)
      DT[18O](g) + H2O(g) = DTO(g) + H2[18O](g)
      log_k           0.0
#
# O2 gas - O2 aqueous
# last update March 20, 2006
# Updated September 19, 2006
# Checked September 19, 2006
#

O[18O](g)
      O[18O](g) + O2 = O2(g) + O[18O]
      -add_logk       Log_alpha_18O_O2(g)/O2(aq)             -1.0

[18O]2(g)
      [18O]2(g) + O2(g) = 2O[18O](g)
      log_k           0.602059991327962396                   # log10(4)

# H2 gas - H2 aqueous
# last update March 20, 2006
# Updated September 19, 2006
# Checked September 19, 2006
#
HD(g)
      HD(g) + H2O(l) = H2(g) + HDO(aq)
      -add_logk       Log_alpha_D_H2(g)/H2O(l)               -1.0

HT(g)
      HT(g) + H2O(l) = H2(g) + HTO(aq)
      -add_logk       Log_alpha_T_H2(g)/H2O(l)               -1.0

D2(g)
      D2(g) + H2(g) = 2HD(g)
      log_k           0.602059991327962396                   # log10(4)

T2(g)
      T2(g) + H2(g) = 2HT(g)
      log_k           0.602059991327962396                   # log10(4)

DT(g)
      DT(g) + H2(g) = HD(g) + HT(g)
      log_k           0.301029995663                          # log10(2)
#
# CO2 gas - CO2 aqueous
# last update March 20, 2006
# Updated September 19, 2006
# Checked September 19, 2006
#
C0[18O](g)
      C0[18O](g) + H2O(l) = H2[18O](aq) + CO2(g)
      log_k           -0.301029995663                         # -log10(2)
      -add_logk       Log_alpha_18O_CO2(g)/H2O(l)            -1.0

C[18O]2(g)
      C[18O]2(g) + CO2(g) = 2C0[18O](g)
      log_k           0.602059991327962396                   # log10(4)

[13C]O2(g)
      [13C]O2(g) + CO2(aq) = [13C]O2(aq) + CO2(g)
      -add_logk       Log_alpha_13C_CO2(g)/CO2(aq)             -1.0

[13C]O[18O](g)
      [13C]O[18O](g) + CO2(g) = [13C]O2(g) + C0[18O](g)
      log_k           0

[13C][18O]2(g)
      [13C][18O]2(g) + CO2(g) = C[18O]2(g) + [13C]O2(g)
      log_k           0

[14C]O2(g)
      [14C]O2(g) + CO2(aq) = [14C]O2(aq) + CO2(g)
      -add_logk       Log_alpha_14C_CO2(g)/CO2(aq)             -1.0
```

```
[14C]O[18O](g)
      [14C]O[18O](g) + CO2(g) = [14C]O2(g) + CO[18O](g)
      log_k             0

[14C][18O]2(g)
      [14C][18O]2(g) + CO2(g) = [14C]O2(g) + C[18O]2(g)
      log_k             0
#
# Calcite
# last update Feb 13, 2006
# Updated September 19, 2006
# Checked September 19, 2006
# CaCO2[18O](s) equation updated June 1, 2016 (R. Owen)
#
CaCO2[18O](s)
     CaCO2[18O](s) + CO2 = Calcite(s) + CO[18O]
     log_k             -0.176091259055                    # log10(2/3)
     -add_logk         Log_alpha_18O_Calcite/CO2(aq)       -1.0

CaCO[18O]2(s)
      CaCO[18O]2(s) + Calcite(s) = 2CaCO2[18O](s)
      log_k             0.477121254719                  # log10(3)

CaC[18O]3(s)
      CaC[18O]3(s) + 2Calcite(s) = 3CaCO2[18O](s)
      log_k             1.431363764158                   # log10(27)

Ca[13C]O3(s)
      Ca[13C]O3(s) + CO2 = [13C]O2 + Calcite(s)
      -add_logk         Log_alpha_13C_Calcite/CO2(aq)       -1.0

Ca[13C]O2[18O](s)
      Ca[13C]O2[18O](s) + Calcite(s) = Ca[13C]O3(s) + CaCO2[18O](s)
      log_k             0

Ca[13C]O[18O]2(s)
      Ca[13C]O[18O]2(s) + Calcite(s) = Ca[13C]O3(s) + CaCO[18O]2(s)
      log_k             0

Ca[13C][18O]3(s)
      Ca[13C][18O]3(s) + Calcite(s) = Ca[13C]O3(s) + CaC[18O]3(s)
      log_k             0

Ca[14C]O3(s)
      Ca[14C]O3(s) + CO2 = Calcite(s) + [14C]O2
      -add_logk         Log_alpha_14C_Calcite/CO2(aq)       -1.0

Ca[14C]O2[18O](s)
      Ca[14C]O2[18O](s) + Calcite(s) = Ca[14C]O3(s) + CaCO2[18O](s)
      log_k             0

Ca[14C]O[18O]2(s)
      Ca[14C]O[18O]2(s) + Calcite(s) = Ca[14C]O3(s) + CaCO[18O]2(s)
      log_k             0

Ca[14C][18O]3(s)
      Ca[14C][18O]3(s) + Calcite(s) = Ca[14C]O3(s) + CaC[18O]3(s)
      log_k             0
#
# Pyrite, March 20, 2006
# Update September 28, 2006
# Checked September 28, 2006
#
Pyrite_FeS[34S]
      FeS[34S](s) + HS- = Pyrite(s) + H[34S]-
      -add_logk         Log_alpha_34S_Pyrite/HS-          -1.0

Pyrite_Fe[34S]S
      Fe[34S]S(s) = Pyrite_FeS[34S](s)

Pyrite_Fe[34S]2
      Fe[34S]2(s) + Pyrite(s) = 2Pyrite_FeS[34S](s)
#
# CH4 reactions
# last update March 20, 2006
# Updated September 19, 2006
# Checked September 19, 2006
#
CH3D(g)
      CH3D(g) + H2O(l) = CH4(g) + HDO(aq)
      log_k             -0.301029995663                    # -log10(2)    ???
      -add_logk         Log_alpha_D_CH4(g)/H2O(l)      -1.0

CH2D2(g)
      CH2D2(g) + CH4(g) = 2CH3D(g)
      log_k             0.42596873227228                # log10(8/3)

CHD3(g)
      CHD3(g) + 2CH4(g) = 3CH3D(g)
      log_k             1.20411998265                    # log10(16)

CD4(g)
      CD4(g) + 3CH4(g) = 4CH3D(g)
      log_k             2.408239965311                  # log10(256)

CH3T(g)
      CH3T(g) + H2O(l) = CH4(g) + HTO(aq)
      log_k             -0.301029995663                    # -log10(2)    ???
      -add_logk         Log_alpha_T_CH4(g)/H2O(l)       -1.0

CH2T2(g)
      CH2T2(g) + CH4(g) = 2CH3T(g)
      log_k             0.42596873227228                # log10(8/3)

CHT3(g)
      CHT3(g) + 2CH4(g) = 3CH3T(g)
      log_k             1.20411998265                    # log10(16)

CT4(g)
      CT4(g) + 3CH4(g) = 4CH3T(g)
      log_k             2.408239965311                  # log10(256)

CD3T(g)
      CD3T(g) + CH4(g) = CHD3(g) + CH3T(g)
      log_k             0.6020599913279623               # log10(4)

CD2T2(g)
      CD2T2(g) + CH4(g) = CH2D2(g) + CH2T2(g)
      log_k             0.7781512503836                 # log10(6)
```

```
CDT3(g)
      CDT3(g) + CH4(g) = CHT3(g) + CH3D(g)
      log_k             0.602059991327962396                    # log10(4)

[13C]H4(g)
      [13C]H4(g) + CO2(aq) = [13C]O2(aq) + CH4(g)
      -add_logk         Log_alpha_13C_CH4(g)/CO2(aq)     -1.0

[13C]H3D(g)
      [13C]H3D(g) + CH4(g) = CH3D(g) + [13C]H4(g)

[13C]H2D2(g)
      [13C]H2D2(g) + CH4(g) = CH2D2(g) + [13C]H4(g)

[13C]HD3(g)
      [13C]HD3(g) + CH4(g) = CHD3(g) + [13C]H4(g)

[13C]D4(g)
      [13C]D4(g) + CH4(g) = CD4(g) + [13C]H4(g)

[13C]H3T(g)
      [13C]H3T(g) + CH4(g) = CH3T(g) + [13C]H4(g)

[13C]H2T2(g)
      [13C]H2T2(g) + CH4(g) = CH2T2(g) + [13C]H4(g)

[13C]HT3(g)
      [13C]HT3(g) + CH4(g) = CHT3(g) + [13C]H4(g)

[13C]T4(g)
      [13C]T4(g) + CH4(g) = CT4(g) + [13C]H4(g)

[13C]D3T(g)
      [13C]D3T(g) + CH4(g) = CD3T(g) + [13C]H4(g)

[13C]D2T2(g)
      [13C]D2T2(g) + CH4(g) = CD2T2(g) + [13C]H4(g)

[13C]DT3(g)
      [13C]DT3(g) + CH4(g) = CDT3(g) + [13C]H4(g)

[13C]T4(g)
      [13C]T4(g) + CH4(g) = CT4(g) + [13C]H4(g)

[14C]H4(g)
      [14C]H4(g) + CO2(aq) = [14C]O2(aq) + CH4(g)
      -add_logk         Log_alpha_14C_CH4(g)/CO2(aq)     -1.0

[14C]H3D(g)
      [14C]H3D(g) + CH4(g) = CH3D(g) + [14C]H4(g)

[14C]H2D2(g)
      [14C]H2D2(g) + CH4(g) = CH2D2(g) + [14C]H4(g)

[14C]HD3(g)
      [14C]HD3(g) + CH4(g) = CHD3(g) + [14C]H4(g)

[14C]D4(g)
      [14C]D4(g) + CH4(g) = CD4(g) + [14C]H4(g)

[14C]H3T(g)
      [14C]H3T(g) + CH4(g) = CH3T(g) + [14C]H4(g)

[14C]H2T2(g)
      [14C]H2T2(g) + CH4(g) = CH2T2(g) + [14C]H4(g)

[14C]HT3(g)
      [14C]HT3(g) + CH4(g) = CHT3(g) + [14C]H4(g)

[14C]T4(g)
      [14C]T4(g) + CH4(g) = CT4(g) + [14C]H4(g)

[14C]D3T(g)
      [14C]D3T(g) + CH4(g) = CD3T(g) + [14C]H4(g)

[14C]D2T2(g)
      [14C]D2T2(g) + CH4(g) = CD2T2(g) + [14C]H4(g)

[14C]DT3(g)
      [14C]DT3(g) + CH4(g) = CDT3(g) + [14C]H4(g)

[14C]T4(g)
      [14C]T4(g) + CH4(g) = CT4(g) + [14C]H4(g)
#
# updated March 20, 2006
# Updated September 28, 2006
# Checked September 28, 2006
#
HDS(g)
      HDS(g) + H2S(aq) = HDS(aq) + H2S(g)
      -add_logk         Log_alpha_D_H2S(g)/H2S(aq)       -1.0

D2S(g)
      D2S(g) + H2S(g) = 2HDS(g)
      log_k             0.602059991327962396              # log10(4)

HTS(g)
      HTS(g) + H2S(aq) = HTS(aq) + H2S(g)
      -add_logk         Log_alpha_T_H2S(g)/H2S(aq)       -1.0

T2S(g)
      T2S(g) + H2S(g) = 2HTS(g)
      log_k             0.602059991327962396              # log10(4)

DTS(g)
      DTS(g) + H2S(g) = HDS(g) + HTS(g)
      log_k             0.301029995663                  # log10(2)

H2[34S](g)
      H2[34S](g) + H2S(aq) = H2[34S](aq) + H2S(g)
      -add_logk         Log_alpha_34S_H2S(g)/H2S(aq)     -1.0

HD[34S](g)
      HD[34S](g) + H2S(g) = HDS(g) + H2[34S](g)

D2[34S](g)
      D2[34S](g) + H2S(g) = D2S(g) + H2[34S](g)

HT[34S](g)
```

```
      HT[34S](g) + H2S(g) = HTS(g) + H2[34S](g)

T2[34S](g)
      T2[34S](g) + H2S(g) = T2S(g) + H2[34S](g)

DT[34S](g)
      DT[34S](g) + H2S(g) = DTS(g) + H2[34S](g)

#
# Gypsum and anhydrite
#
Ca[34S]O4:2H2O
      Ca[34S]O4:2H2O + SO4-2 = [34S]O4-2 + Gypsum(s)
      -add_logk         Log_alpha_34S_Gypsum/SO4-2        -1.0

Ca[34S]O4
      Ca[34S]O4 + SO4-2 = [34S]O4-2 + Anhydrite(s)
      -add_logk         Log_alpha_34S_Anhydrite/SO4-2     -1.0
#
#  Nitrogen phases
#  Checked September 19, 2006
#
N[15N](g)
      N[15N](g) + N2(aq) = N[15N](aq) + N2(g)
      -add_logk         Log_alpha_15N_N2(g)/N2(aq)              -1.0

[15N]2(g)
      [15N]2(g) + N2(g) = 2N[15N](g)
      log_k             0.602059991327962396                    # log10(4)

NH2D(g)
      NH2D(g) + H2O(l) = NH3(g) + HDO(aq)
## symmetry K??
      -add_logk         Log_alpha_D_NH3(g)/H2O(l)               -1.0

NHD2(g)
      NHD2(g) + NH3(g) = 2NH2D(g)
      log_k             0.477121254719                          # log10(3)

ND3(g)
      ND3(g) + 2NH3(g) = 3NH2D(g)
          -logk             1.431363764158                          # log10(27)

NH2T(g)
      NH2T(g) + H2O(l) = NH3(g) + HTO(aq)
## symmetry K??
      -add_logk         Log_alpha_T_NH3(g)/H2O(l)               -1.0

NHT2(g)
      NHT2(g) + NH3(g) = 2NH2T(g)
      log_k             0.477121254719                          # log10(3)

NT3(g)
      NT3(g) + 2NH3(g) = 3NH2T(g)
          -logk             1.431363764158                          # log10(27)

ND2T(g)
      ND2T(g) + NH3(g) = NHD2(g) + NH2T(g)
      log_k             0.477121254719                          # log10(3)

NDT2(g)
      NDT2(g) + NH3(g) = NH2D(g) + NHT2(g)
      log_k             0.477121254719                          # log10(3)

[15N]H3(g)
      [15N]H3(g) + NH3(aq) = [15N]H3(aq) + NH3(g)
      -add_logk         Log_alpha_15N_NH3(g)/NH3(aq)            -1.0

[15N]H2D(g)
      [15N]H2D(g) + NH3(g) = NH2D(g) + [15N]H3(g)

[15N]HD2(g)
      [15N]HD2(g) + NH3(g) = NHD2(g) + [15N]H3(g)

[15N]D3(g)
      [15N]D3(g) + NH3(g) = ND3(g) + [15N]H3(g)

[15N]H2T(g)
      [15N]H2T(g) + NH3(g) = NH2T(g) + [15N]H3(g)

[15N]HT2(g)
      [15N]HT2(g) + NH3(g) = NHT2(g) + [15N]H3(g)

[15N]T3(g)
      [15N]T3(g) + NH3(g) = NT3(g) + [15N]H3(g)

[15N]D2T(g)
      [15N]D2T(g) + NH3(g) = ND2T(g) + [15N]H3(g)

[15N]DT2(g)
      [15N]DT2(g) + NH3(g) = NDT2(g) + [15N]H3(g)

#↩
      ##############################################################################################↩

#↩
      ##############################################################################################↩

#↩
      ##############################################################################################↩

#  Begin Ca isotope definitions
#
#↩
      ##############################################################################################↩

SOLUTION_MASTER_SPECIES

[44Ca]        [44Ca]+2     0    [44Ca]   43.9555
[44Ca](2)     [44Ca]+2     0    [44Ca]

#↩
      ##############################################################################################↩

ISOTOPES

# Abundance data from Coplen et al (2002), modified to give minor/major ratios
Ca
    -isotope     [44Ca]   permil   0.02152 # molar ratios to 40Ca
```

```
#↩
    ##############################################################################################↩

ISOTOPE_RATIOS #print isotope ratios calculated in CALCULATE_VALUES, defines units
    R(44Ca)          [44Ca]
    R(44Ca)_Calcite [44Ca]

#↩
    ##############################################################################################↩

ISOTOPE_ALPHAS
    Alpha_44Ca_Calcite/Ca(aq)        Log_alpha_44Ca_Calcite/Ca(aq)

#↩
    ##############################################################################################↩

CALCULATE_VALUES #calculate isotope ratios for display, and display fractionation factors
R(44Ca) #molality ratio in solution (modified from iso.dat 18O)
     -start
10 ratio = -9999
20 if (TOT("[44Ca]") <= 0) THEN GOTO 100
30 total_44Ca = TOT("[44Ca]")
40 total_40Ca = TOT("Ca")
50 ratio = total_44Ca/total_40Ca
100 save ratio
     -end

R(44Ca)_Calcite #molar isotope ratio in calcite (modified from iso.dat)
     -start
10 ratio = -9999
20 if (TOT("[44Ca]") <= 0) THEN GOTO 1000
30 if (S_S("Calcite") <= 0) THEN GOTO 1000
40 total_44Ca = SUM_S_S("Calcite", "[44Ca]")
50 total_Ca = SUM_S_S("Calcite", "Ca")
60 if (total_Ca <= 0) THEN GOTO 1000
70 ratio = total_44Ca/total_Ca
1000 save ratio
     -end

Alpha_44Ca_Calcite/Ca(aq)
     -start
10 alpha = -9999
20 if (TOT("[44Ca]") <= 0) THEN GOTO 1000
30 if (S_S("Calcite") <= 0) THEN GOTO 1000
40 r1 = CALC_VALUE("R(44Ca)_Calcite")
50 r2 = CALC_VALUE("R(44Ca)")
60 if (r2 <= 0) THEN GOTO 1000
80 if (r1 <= -9999) THEN GOTO 1000
90 alpha = r1/r2
1000 save alpha
     -end

NAMED_EXPRESSIONS #give fractionation factors

#
## Calcite Trace Element partition coefficients
Log_alpha_MgCa_Calcite/H2O(l)
    # Day and Henderson (2013)
    -ln_alpha1000     -10078.22052  20

Log_alpha_SrCa_Calcite/H2O(l)
    # Day and Henderson (2013)
    -ln_alpha1000     -1739.755829  -1.1

Log_alpha_BaCa_Calcite/H2O(l)
    # Day and Henderson (2013)
    -ln_alpha1000     -2507.739913   1.1

#
## Ca isotope fractionation factors
Log_alpha_44Ca_Calcite/Ca(aq)
    # Owen et al. (2016)
    -ln_alpha1000    -1.27    # constant in this version

#↩
    ##############################################################################################↩

SOLUTION_SPECIES # 44Ca exchange reactions

[44Ca]+2 = [44Ca]+2 #copied from iso.dat - defines 44Ca+2 as a dissolved ion
    log_k   0
    -gamma  5.0000  0.1650

# other dissolved Ca species exchange reactions
[44Ca]+2 + CaOH+ = [44Ca]OH+ + Ca+2
[44Ca]+2 + CaHCO3+ = [44Ca]HCO3+ + Ca+2
[44Ca]+2 + CaSO4 = [44Ca]SO4 + Ca+2
[44Ca]+2 + CaHSO4+ = [44Ca]HSO4+ + Ca+2
[44Ca]+2 + CaPO4- = [44Ca]PO4- + Ca+2
[44Ca]+2 + CaHPO4 = [44Ca]HPO4 + Ca+2
[44Ca]+2 + CaH2PO4+ = [44Ca]H2PO4+ + Ca+2
[44Ca]+2 + CaF+ = [44Ca]F+ + Ca+2

# carbonate species Ca exchange reactions
CaCO3 + [44Ca]+2 = [44Ca]CO3 + Ca+2
CaCO2[18O] + [44Ca]+2 = [44Ca]CO2[18O] + Ca+2
CaCO[18O]2 + [44Ca]+2 = [44Ca]CO[18O]2 + Ca+2
CaC[18O]3 + [44Ca]+2 = [44Ca]C[18O]3 + Ca+2
Ca[13C]O3 + [44Ca]+2 = [44Ca][13C]O3 + Ca+2
Ca[13C]O2[18O] + [44Ca]+2 = [44Ca][13C]O2[18O] + Ca+2
Ca[13C]O[18O]2 + [44Ca]+2 = [44Ca][13C]O[18O]2 + Ca+2
Ca[13C][18O]3 + [44Ca]+2 = [44Ca][13C][18O]3 + Ca+2
Ca[14C]O3 + [44Ca]+2 = [44Ca][14C]O3 + Ca+2
Ca[14C]O2[18O] + [44Ca]+2 = [44Ca][14C]O2[18O] + Ca+2
Ca[14C]O[18O]2 + [44Ca]+2 = [44Ca][14C]O[18O]2 + Ca+2
Ca[14C][18O]3 + [44Ca]+2 = [44Ca][14C][18O]3 + Ca+2

# HCO3- species Ca exchange reactions
CaHCO3+ + [44Ca]+2 = [44Ca]HCO3+ + Ca+2
     -gamma             5.4000     0.0000

CaHCO2[18O]+ + [44Ca]+2 = [44Ca]HCO2[18O]+ + Ca+2
     -gamma             5.4000     0.0000

CaHCO[18O]O+ + [44Ca]+2 = [44Ca]HCO[18O]O+ + Ca+2
     -gamma             5.4000     0.0000

CaHC[18O]O2+ + [44Ca]+2 = [44Ca]HC[18O]O2+ + Ca+2
     -gamma             5.4000     0.0000
```

```
CaHCO[18O]2+ + [44Ca]+2 = [44Ca]HCO[18O]2+ + Ca+2
     -gamma            5.4000    0.0000

CaHC[18O]O[18O]+ + [44Ca]+2 = [44Ca]HC[18O]O[18O]+ + Ca+2
     -gamma            5.4000    0.0000

CaHC[18O]2O+ + [44Ca]+2 = [44Ca]HC[18O]2O+ + Ca+2
     -gamma            5.4000    0.0000

CaHC[18O]3+ + [44Ca]+2 = [44Ca]HC[18O]3+ + Ca+2
     -gamma            5.4000    0.0000

CaH[13C]O3+ + [44Ca]+2 = [44Ca]H[13C]O3+ + Ca+2
     -gamma            5.4000    0.0000

CaH[13C]O2[18O]+ + [44Ca]+2 = [44Ca]H[13C]O2[18O]+ + Ca+2
     -gamma            5.4000    0.0000

CaH[13C]O[18O]O+ + [44Ca]+2 = [44Ca]H[13C]O[18O]O+ + Ca+2
     -gamma            5.4000    0.0000

CaH[13C][18O]O2+ + [44Ca]+2 = [44Ca]H[13C][18O]O2+ + Ca+2
     -gamma            5.4000    0.0000

CaH[13C]O[18O]2+ + [44Ca]+2 = [44Ca]H[13C]O[18O]2+ + Ca+2
     -gamma            5.4000    0.0000

CaH[13C][18O]O[18O]+ + [44Ca]+2 = [44Ca]H[13C][18O]O[18O]+ + Ca+2
     -gamma            5.4000    0.0000

CaH[13C][18O]2O+ + [44Ca]+2 = [44Ca]H[13C][18O]2O+ + Ca+2
     -gamma            5.4000    0.0000

CaH[13C][18O]3+ + [44Ca]+2 = [44Ca]H[13C][18O]3+ + Ca+2
     -gamma            5.4000    0.0000

CaH[14C]O3+ + [44Ca]+2 = [44Ca]H[14C]O3+ + Ca+2
     -gamma            5.4000    0.0000

CaH[14C]O2[18O]+ + [44Ca]+2 = [44Ca]H[14C]O2[18O]+ + Ca+2
     -gamma            5.4000    0.0000

CaH[14C]O[18O]O+ + [44Ca]+2 = [44Ca]H[14C]O[18O]O+ + Ca+2
     -gamma            5.4000    0.0000

CaH[14C][18O]O2+ + [44Ca]+2 = [44Ca]H[14C][18O]O2+ + Ca+2
     -gamma            5.4000    0.0000

CaH[14C]O[18O]2+ + [44Ca]+2 = [44Ca]H[14C]O[18O]2+ + Ca+2
     -gamma            5.4000    0.0000

CaH[14C][18O]O[18O]+ + [44Ca]+2 = [44Ca]H[14C][18O]O[18O]+ + Ca+2
     -gamma            5.4000    0.0000

CaH[14C][18O]2O+ + [44Ca]+2 = [44Ca]H[14C][18O]2O+ + Ca+2
     -gamma            5.4000    0.0000

CaH[14C][18O]3+ + [44Ca]+2 = [44Ca]H[14C][18O]3+ + Ca+2
     -gamma            5.4000    0.0000

# DCO3-
CaDCO3+ + [44Ca]+2 = [44Ca]DCO3+ + Ca+2
     -gamma            5.4000    0.0000

CaDCO2[18O]+ + [44Ca]+2 = [44Ca]DCO2[18O]+ + Ca+2
     -gamma            5.4000    0.0000

CaDCO[18O]O+ + [44Ca]+2 = [44Ca]DCO[18O]O+ + Ca+2
     -gamma            5.4000    0.0000

CaDC[18O]O2+ + [44Ca]+2 = [44Ca]DC[18O]O2+ + Ca+2
     -gamma            5.4000    0.0000

CaDCO[18O]2+ + [44Ca]+2 = [44Ca]DCO[18O]2+ + Ca+2
     -gamma            5.4000    0.0000

CaDC[18O]O[18O]+ + [44Ca]+2 = [44Ca]DC[18O]O[18O]+ + Ca+2
     -gamma            5.4000    0.0000

CaDC[18O]2O+ + [44Ca]+2 = [44Ca]DC[18O]2O+ + Ca+2
     -gamma            5.4000    0.0000

CaDC[18O]3+ + [44Ca]+2 = [44Ca]DC[18O]3+ + Ca+2
     -gamma            5.4000    0.0000

CaD[13C]O3+ + [44Ca]+2 = [44Ca]D[13C]O3+ + Ca+2
     -gamma            5.4000    0.0000

CaD[13C]O2[18O]+ + [44Ca]+2 = [44Ca]D[13C]O2[18O]+ + Ca+2
     -gamma            5.4000    0.0000

CaD[13C]O[18O]O+ + [44Ca]+2 = [44Ca]D[13C]O[18O]O+ + Ca+2
     -gamma            5.4000    0.0000

CaD[13C][18O]O2+ + [44Ca]+2 = [44Ca]D[13C][18O]O2+ + Ca+2
     -gamma            5.4000    0.0000

CaD[13C]O[18O]2+ + [44Ca]+2 = [44Ca]D[13C]O[18O]2+ + Ca+2
     -gamma            5.4000    0.0000

CaD[13C][18O]O[18O]+ + [44Ca]+2 = [44Ca]D[13C][18O]O[18O]+ + Ca+2
     -gamma            5.4000    0.0000

CaD[13C][18O]2O+ + [44Ca]+2 = [44Ca]D[13C][18O]2O+ + Ca+2
     -gamma            5.4000    0.0000

CaD[13C][18O]3+ + [44Ca]+2 = [44Ca]D[13C][18O]3+ + Ca+2
     -gamma            5.4000    0.0000

CaD[14C]O3+ + [44Ca]+2 = [44Ca]D[14C]O3+ + Ca+2
     -gamma            5.4000    0.0000

CaD[14C]O2[18O]+ + [44Ca]+2 = [44Ca]D[14C]O2[18O]+ + Ca+2
     -gamma            5.4000    0.0000

CaD[14C]O[18O]O+ + [44Ca]+2 = [44Ca]D[14C]O[18O]O+ + Ca+2
     -gamma            5.4000    0.0000

CaD[14C][18O]O2+ + [44Ca]+2 = [44Ca]D[14C][18O]O2+ + Ca+2
     -gamma            5.4000    0.0000

CaD[14C]O[18O]2+ + [44Ca]+2 = [44Ca]D[14C]O[18O]2+ + Ca+2
     -gamma            5.4000    0.0000
```

```
CaD[14C][18O]O[18O]+ + [44Ca]+2 = [44Ca]D[14C][18O]O[18O]+ + Ca+2
     -gamma              5.4000      0.0000

CaD[14C][18O]2O+ + [44Ca]+2 = [44Ca]D[14C][18O]2O+ + Ca+2
     -gamma              5.4000      0.0000

CaD[14C][18O]3+ + [44Ca]+2 = [44Ca]D[14C][18O]3+ + Ca+2
     -gamma              5.4000      0.0000

# TCO3-
CaTCO3+ + [44Ca]+2 = [44Ca]TCO3+ + Ca+2
     -gamma              5.4000      0.0000

CaTCO2[18O]+ + [44Ca]+2 = [44Ca]TCO2[18O]+ + Ca+2
     -gamma              5.4000      0.0000

CaTCO[18O]O+ + [44Ca]+2 = [44Ca]TCO[18O]O+ + Ca+2
     -gamma              5.4000      0.0000

CaTC[18O]O2+ + [44Ca]+2 = [44Ca]TC[18O]O2+ + Ca+2
     -gamma              5.4000      0.0000

CaTCO[18O]2+ + [44Ca]+2 = [44Ca]TCO[18O]2+ + Ca+2
     -gamma              5.4000      0.0000

CaTC[18O]O[18O]+ + [44Ca]+2 = [44Ca]TC[18O]O[18O]+ + Ca+2
     -gamma              5.4000      0.0000

CaTC[18O]2O+ + [44Ca]+2 = [44Ca]TC[18O]2O+ + Ca+2
     -gamma              5.4000      0.0000

CaTC[18O]3+ + [44Ca]+2 = [44Ca]TC[18O]3+ + Ca+2
     -gamma              5.4000      0.0000

CaT[13C]O3+ + [44Ca]+2 = [44Ca]T[13C]O3+ + Ca+2
     -gamma              5.4000      0.0000

CaT[13C]O2[18O]+ + [44Ca]+2 = [44Ca]T[13C]O2[18O]+ + Ca+2
     -gamma              5.4000      0.0000

CaT[13C]O[18O]O+ + [44Ca]+2 = [44Ca]T[13C]O[18O]O+ + Ca+2
     -gamma              5.4000      0.0000

CaT[13C][18O]O2+ + [44Ca]+2 = [44Ca]T[13C][18O]O2+ + Ca+2
     -gamma              5.4000      0.0000

CaT[13C]O[18O]2+ + [44Ca]+2 = [44Ca]T[13C]O[18O]2+ + Ca+2
     -gamma              5.4000      0.0000

CaT[13C][18O]O[18O]+ + [44Ca]+2 = [44Ca]T[13C][18O]O[18O]+ + Ca+2
     -gamma              5.4000      0.0000

CaT[13C][18O]2O+ + [44Ca]+2 = [44Ca]T[13C][18O]2O+ + Ca+2
     -gamma              5.4000      0.0000

CaT[13C][18O]3+ + [44Ca]+2 = [44Ca]T[13C][18O]3+ + Ca+2
     -gamma              5.4000      0.0000

CaT[14C]O3+ + [44Ca]+2 = [44Ca]T[14C]O3+ + Ca+2
     -gamma              5.4000      0.0000

CaT[14C]O2[18O]+ + [44Ca]+2 = [44Ca]T[14C]O2[18O]+ + Ca+2
     -gamma              5.4000      0.0000

CaT[14C]O[18O]O+ + [44Ca]+2 = [44Ca]T[14C]O[18O]O+ + Ca+2
     -gamma              5.4000      0.0000

CaT[14C][18O]O2+ + [44Ca]+2 = [44Ca]T[14C][18O]O2+ + Ca+2
     -gamma              5.4000      0.0000

CaT[14C]O[18O]2+ + [44Ca]+2 = [44Ca]T[14C]O[18O]2+ + Ca+2
     -gamma              5.4000      0.0000

CaT[14C][18O]O[18O]+ + [44Ca]+2 = [44Ca]T[14C][18O]O[18O]+ + Ca+2
     -gamma              5.4000      0.0000

CaT[14C][18O]2O+ + [44Ca]+2 = [44Ca]T[14C][18O]2O+ + Ca+2
     -gamma              5.4000      0.0000

CaT[14C][18O]3+ + [44Ca]+2 = [44Ca]T[14C][18O]3+ + Ca+2
     -gamma              5.4000      0.0000


#
# CaSO4+ reactions (no O isotopes) - iso.dat does not fully support this
#
# CaSO4 + [44Ca]+2 = [44Ca]SO4 + Ca+2
# Ca[34S]O4 + [44Ca]+2 = [44Ca][34S]O4 + Ca+2

#
# CaHSO4+ reactions (no O isotopes) - iso.dat does not fully support this
#
# CaHSO4+ + [44Ca]+2 = [44Ca]HSO4+ + Ca+2
# CaDSO4+ + [44Ca]+2 = [44Ca]DSO4+ + Ca+2
# CaTSO4+ + [44Ca]+2 = [44Ca]TSO4+ + Ca+2
# CaH[34S]O4+ + [44Ca]+2 = [44Ca]H[34S]O4+ + Ca+2
# CaD[34S]O4+ + [44Ca]+2 = [44Ca]D[34S]O4+ + Ca+2
# CaT[34S]O4+ + [44Ca]+2 = [44Ca]T[34S]O4+ + Ca+2

#
# Missing CaPO4- reactions
#
#
# Missing CaHPO4 reactions
#
#
# Missing CaH2PO4+ reactions

#↩
     ##############################################################################################↩

PHASES # list isotope exchanges into calcite (fractionation factors given in ↩
     NAMED EXPRESSIONS
########################

# Trace Metal Calcite Exchanges
# for use in solid solutions

MgCO3(s)
     MgCO3(s) + Ca+2 = Calcite(s) + Mg+2
     -add_logk Log_alpha_MgCa_Calcite/H2O(l)      -1.0

MgCO2[18O](s)
```

```
    MgCO2[18O](s) + Calcite(s) = MgCO3(s) + CaCO2[18O](s)

MgCO[18O]2(s)
    MgCO[18O]2(S) + Calcite(s) = MgCO3(s) + CaCO[18O]2(s)

MgC[18O]3(s)
    MgC[18O]3(s) + Calcite(s) = MgCO3(s) + CaC[18O]3(s)

Mg[13C]O3(s)
    Mg[13C]O3(s) + Calcite(s) = MgCO3(s) + Ca[13C]O3(s)(s)

Mg[13C]O2[18O](s)
    Mg[13C]O2[18O](s) + Calcite(s) = MgCO3(s) + Ca[13C]O2[18O](s)

Mg[13C]O[18O]2(s)
    Mg[13C]O[18O]2 + Calcite(s) = MgCO3(s) + Ca[13C]O[18O]2(s)

Mg[13C][18O]3(s)
    Mg[13C][18O]3(s) + Calcite(s) = MgCO3(s) + Ca[13C][18O]3(s)

Mg[14C]O3(s)
    Mg[14C]O3(s) + Calcite(s) = MgCO3(s) + Ca[14C]O3(s)

Mg[14C]O2[18O](s)
    MgCO2[18O](s) + Calcite(s) = MgCO3(s) + CaCO2[18O](s)

Mg[14C]O[18O]2(s)
    MgCO2[18O](s) + Calcite(s) = MgCO3(s) + CaCO2[18O](s)

Mg[14C][18O]3(s)
    MgCO2[18O](s) + Calcite(s) = MgCO3(s) + CaCO2[18O](s)


SrCO3(s)
    SrCO3(s) + Ca+2 = Calcite(s) + Sr+2
    -add_logk Log_alpha_SrCa_Calcite/H2O(l)      -1.0

SrCO2[18O](s)
    SrCO2[18O](s) + Calcite(s) = SrCO3(s) + CaCO2[18O](s)

SrCO[18O]2(s)
    SrCO[18O]2(S) + Calcite(s) = SrCO3(s) + CaCO[18O]2(s)

SrC[18O]3(s)
    SrC[18O]3(s) + Calcite(s) = SrCO3(s) + CaC[18O]3(s)

Sr[13C]O3(s)
    Sr[13C]O3(s) + Calcite(s) = SrCO3(s) + Ca[13C]O3(s)(s)

Sr[13C]O2[18O](s)
    Sr[13C]O2[18O](s) + Calcite(s) = SrCO3(s) + Ca[13C]O2[18O](s)

Sr[13C]O[18O]2(s)
    Sr[13C]O[18O]2 + Calcite(s) = SrCO3(s) + Ca[13C]O[18O]2(s)

Sr[13C][18O]3(s)
    Sr[13C][18O]3(s) + Calcite(s) = SrCO3(s) + Ca[13C][18O]3(s)

Sr[14C]O3(s)
    Sr[14C]O3(s) + Calcite(s) = SrCO3(s) + Ca[14C]O3(s)

Sr[14C]O2[18O](s)
    SrCO2[18O](s) + Calcite(s) = SrCO3(s) + CaCO2[18O](s)

Sr[14C]O[18O]2(s)
    SrCO2[18O](s) + Calcite(s) = SrCO3(s) + CaCO2[18O](s)

Sr[14C][18O]3(s)
    SrCO2[18O](s) + Calcite(s) = SrCO3(s) + CaCO2[18O](s)


BaCO3(s)
    BaCO3(s) + Ca+2 = Calcite(s) + Ba+2
    -add_logk Log_alpha_BaCa_Calcite/H2O(l)      -1.0

BaCO2[18O](s)
    BaCO2[18O](s) + Calcite(s) = BaCO3(s) + CaCO2[18O](s)

BaCO[18O]2(s)
    BaCO[18O]2(S) + Calcite(s) = BaCO3(s) + CaCO[18O]2(s)

BaC[18O]3(s)
    BaC[18O]3(s) + Calcite(s) = BaCO3(s) + CaC[18O]3(s)

Ba[13C]O3(s)
    Ba[13C]O3(s) + Calcite(s) = BaCO3(s) + Ca[13C]O3(s)(s)

Ba[13C]O2[18O](s)
    Ba[13C]O2[18O](s) + Calcite(s) = BaCO3(s) + Ca[13C]O2[18O](s)

Ba[13C]O[18O]2(s)
    Ba[13C]O[18O]2 + Calcite(s) = BaCO3(s) + Ca[13C]O[18O]2(s)

Ba[13C][18O]3(s)
    Ba[13C][18O]3(s) + Calcite(s) = BaCO3(s) + Ca[13C][18O]3(s)

Ba[14C]O3(s)
    Ba[14C]O3(s) + Calcite(s) = BaCO3(s) + Ca[14C]O3(s)

Ba[14C]O2[18O](s)
    BaCO2[18O](s) + Calcite(s) = BaCO3(s) + CaCO2[18O](s)

Ba[14C]O[18O]2(s)
    BaCO2[18O](s) + Calcite(s) = BaCO3(s) + CaCO2[18O](s)

Ba[14C][18O]3(s)
    BaCO2[18O](s) + Calcite(s) = BaCO3(s) + CaCO2[18O](s)

# 44Ca into Calcite
[44Ca]CO3(s)
     [44Ca]CO3(s) + Ca+2 = Calcite(s) + [44Ca]+2    #44Cainto calcite
     -add_logk  Log_alpha_44Ca_Calcite/Ca(aq)    -1.0

[44Ca]CO2[18O](s)
     [44Ca]CO2[18O](s) + Calcite(s) = [44Ca]CO3(s) + CaCO2[18O](s)

[44Ca]CO[18O]2(s)
     [44Ca]CO[18O]2(s) + Calcite(s) = [44Ca]CO3(s) + CaCO[18O]2(s)

[44Ca]C[18O]3(s)
```

```
      [44Ca]C[18O]3(s) + Calcite(s) = [44Ca]CO3(s) + CaC[18O]3(s)

[44Ca][13C]O3(s)
      [44Ca][13C]O3(s) + Calcite(s) = [44Ca]CO3(s) + Ca[13C]O3(s)

[44Ca][13C]O2[18O](s)
      [44Ca][13C]O2[18O](s) + Calcite(s) = [44Ca]CO3(s) + Ca[13C]O2[18O](s)

[44Ca][13C]O[18O]2(s)
      [44Ca][13C]O[18O]2(s) + Calcite(s) = [44Ca]CO3(s) + Ca[13C]O[18O]2(s)

[44Ca][13C][18O]3(s)
      [44Ca][13C][18O]3(s) + Calcite(s) = [44Ca]CO3(s) + Ca[13C][18O]3(s)

[44Ca][14C]O3(s)
      [44Ca][14C]O3(s) + Calcite(s) = [44Ca]CO3(s) + Ca[14C]O3(s)

[44Ca][14C]O2[18O](s)
      [44Ca][14C]O2[18O](s) + Calcite(s) = [44Ca]CO3(s) + Ca[14C]O2[18O](s)

[44Ca][14C]O[18O]2(s)
      [44Ca][14C]O[18O]2(s) + Calcite(s) = [44Ca]CO3(s) + Ca[14C]O[18O]2(s)

[44Ca][14C][18O]3(s)
      [44Ca][14C][18O]3(s) + Calcite(s) = [44Ca]CO3(s) + Ca[14C][18O]3(s)

SOLUTION_SPECIES # non-convergence aid (from phreeqc manual)
H2O + 0.01e- = H2O-0.01
log_k -9.0
```

## A.3.12 cavecalc/gui/__init__.py

## A.3.13 cavecalc/gui/gui.py

```
"""Codes for the Cavecalc GUIs.

The GUIs provide a simple user-friendly interface for Cavecalc. They do not
expose the full model functionality but allow most common calculations to be
performed.

GUIs may be accessed by running the scripts:
    cc_input_gui.py
    cc_output_gui.py
"""

import os
import copy
from collections import OrderedDict
import operator
from numpy import linspace
import matplotlib
from sys import platform
if platform != 'win32':
    matplotlib.use('TkAgg') # necessary for mac
from matplotlib import pyplot as plt
from tkinter import *
from tkinter import filedialog
from cavecalc.analyse import Evaluate
import cavecalc.data.types_and_limits
import cavecalc.gui.mapping
import cavecalc.gui.layout
from cavecalc.setter import SettingsMaker, NameSwitcher, SettingsObject

# settings options hidden from plotting menus (useless clutter)
HIDE_OPS = True
HIDDEN_OPTS = [
    'totals', 'molalities', 'isotopes', 'out_dir',
    'phreeqc_log_file', 'phreeqc_log_file_name',
    'database'] # options not available for plotting

ns = NameSwitcher()

def od(dict):
    """Converts a dict to an ordered dict, sorted by key."""

    return OrderedDict(sorted(dict.items()))

def py2tk(dict):
    """Converts a dict to Tkinter types.

    Convert dictionary entries from doubles and strings to StringVar for use
    with tkinter. Returns a modified copy.

    Args:
        dict: A dict with entries that are simple data types.
    Returns:
        A modified dict.
    """

    out = copy.copy(dict)
    types = vars(cavecalc.data.types_and_limits).copy()

    for k in dict.keys():
        if type(types[k]) is bool:
            out[k] = BooleanVar()
            out[k].set(False)
        elif dict[k] is not None:
            out[k] = StringVar()
```

```
            out[k].set(dict[k])
        else:
            out[k] = None
    return out

def _parse_value_input(string, allow=[]):
    """
    Parses leftmost panel input to detect ranges of values or single values.
    Returns either a double or a list of doubles.
    """
    a = string

    # remove brackets
    rm = ['(', ')', '[', ']']
    for s in rm:
        if s not in allow:
            a = a.replace(s,'')

    # replace commas with space
    a = a.replace(',',' ')

    #split string on spaces
    a = a.split(' ')

    # attempt conversion to float (most data types are numeric)
    try:
        b = [float(v) for v in a if v != '']
    except ValueError:
        b = [str(v) for v in a if v!= '']

    # remove list structure from single entries
    if len(b) == 1:
        b = b[0]

    return b

def tk2py(dict, parse=False):
    """Inverse of py2tk.

    Convert a dict of tkinter StringVar types to a a dict understandable by
    the cavecalc setter module.

    Args:
        dict: A dict full of GUI inputs (e.g. StringVar types)
        parse (optional): If True, process numeric input to get a list of
            values if possible. Default False.
    Returns:
        A dictionary of booleans, strings, lists and floats.
    """

    a = dict.copy()
    for k in dict.keys():
        if dict[k] is None:
            a[k] = None
        elif type(dict[k].get()) is str:
            if parse:
                a[k] = _parse_value_input(dict[k].get())
            else:
                a[k] = dict[k].get()
        else:
            a[k] = dict[k].get()

    b = {k:v for k,v in a.items() if v is not None}
    return b

def gplot(  x_values, y_values, x_label, y_label,
            label_vals, label_name ):
    """Plot data in a new window.

    This plotting function is a simplified version of the plotting code used in
    the analyse module for use with the GUI. If x_values contains lists of
    length 1, a single series is plotted, connecting data points from all
    models together. If length > 1 then each model is plotted as a separate
    series.

    If more advanced plotting is required... use something else.

    Args:
        x_values: A list of lists containing the model output to be plotted on
            the x-axis.
        y_values: A list of lists containing the model output to be plotted on
            the y-axis.
        x_label (str): x-axis label.
        y_label (Str): y-axis label.
        label_vals = A list of values to label the series with.
        label_name (str) = The name of the data in label_vals.

    """

    fig, ax = plt.subplots()

    # if plotting... a single point from each model
    if all(len(x) == 1 for x in x_values):
        xs = [x[0] for x in x_values]
        ys = [y[0] for y in y_values]

        ax.plot(xs, ys, 'x--')

        if label_name:
            for label, x, y in zip(label_vals, xs, ys):
                ax.annotate(     "%s=%s" % (label_name, label),
                                 xy = (x,y), fontsize=8)
        plt.ylabel(y_label)
        plt.xlabel(x_label)

        plt.show(block=False)

    # else plot each model as it's own series
    else:
        for (i, xs) in enumerate(x_values):
            ys = y_values[i]
            if label_name:
                l_str = "%s: %s" % (label_name, label_vals[i])
                ax.plot(xs, ys, label = l_str)
                ax.legend(prop={'size':8})
            else:
                ax.plot(xs, ys)

        plt.ylabel(y_label)
        plt.xlabel(x_label)
```

```
        plt.show(block=False)

class FileFindWidget(Frame):
    """A Tkinter Widget for opening a file browser window."""

    def __init__(self, master=None, value=None, mode=None):
        super().__init__(master)
        self.master = master
        self.value = value

        self.entry = Entry(self, textvariable=value).grid(row=0, column=0)

        if mode.capitalize() == 'Load':
            self.button = Button(self, text='browse', command= self._openfilename)
        elif mode.capitalize() == 'Save':
            self.button = Button(self, text='browse', command= self._saveasfilename)
        elif mode.capitalize() == 'Dir':
            self.button = Button(self, text='browse', command= self._getdirectory)
        else:
            raise ValueError("Mode %s not recognised. Use save or load." % mode)
        self.button.grid(row=0, column=1)

    def _openfilename(self, event=None):
        file_path = filedialog.askopenfilename()
        if file_path:
            self.value.set(file_path)

    def _saveasfilename(self, event=None):
        file_path = filedialog.asksaveasfilename()
        if file_path:
            self.value.set(file_path)

    def _getdirectory(self, event=None):
        dir_path = filedialog.askdirectory()
        if dir_path:
            self.value.set(dir_path)

class InputsRangeWidget(Frame):
    """A compound widget for inputting a range of numeric values.

    The InputsRangeWidget contains an Entry field for inputting a single (or
    multiple values) manually. It also has a 'Set Range' button that opens
    a new window for generating an equally-spaced sequence.
    """

    def __init__(self, master=None, value=None):
        super().__init__(master)
        self.master = master
        self.value = value
        self.min = DoubleVar()
        self.max = DoubleVar()
        self.steps = DoubleVar()

        self.entry = Entry(self, textvariable=self.value)
        self.entry.grid(row=0, column=0)
        self.button = Button(self, text='*', command=self.select_range)
        self.button.grid(row=0, column=1)

    def select_range(self):
        """Creates new window to input range information."""

        def get_range():
            vals = linspace( self.min.get(), self.max.get(),
                             num = self.steps.get())
            self.value.set(vals.tolist())
            top.destroy()

        top = Toplevel(self)
        top.title("Select Range") # NOT WORKING / NOT SHOWING
        Label(top, text="Select Range").grid(row=0, columnspan=2)
        Label(top, text="Min").grid(row=1)
        Label(top, text="Max").grid(row=2)
        Label(top, text="Steps").grid(row=3)
        e1 = Entry(top, textvariable = self.min).grid(row=1,column=1)
        e2 = Entry(top, textvariable = self.max).grid(row=2,column=1)
        e3 = Entry(top, textvariable = self.steps).grid(row=3,column=1)
        Button(top, text="OK", command=get_range).grid(row=4)
        Button(top, text="Cancel", command=top.destroy).grid(row=4, column=1)

class OptsWidget(Frame):
    def __init__(self, master, parameter, value, options, row):
        super().__init__(master)
        self.r = row
        self.master = master
        self.parameter = parameter
        self.value = value
        self.options = options

        assert type(options) is list

        self.l = Label(master, text=self.parameter)
        self.l.grid(row=self.r, column=0, sticky=W)
        self.o = OptionMenu(master, self.value, *self.options)
        self.o.grid(row=self.r, column=1)

class InputsWidget(Frame):
    def __init__(self, master, parameter, value, row):
        super().__init__(master)
        self.r = row
        self.master = master
        self.parameter = parameter
        self.value = value

        self.l = Label(master, text=parameter)
        self.l.grid(row=self.r, column=0, sticky=W)
        self.o = Entry(master, textvariable=value)
        self.o.grid(row=self.r, column=1)

class CCInputGUI(object):
    """The Cavecalc input GUI window."""

    def __init__(self, master):
        """Initialise - Open the window."""

        self.master = master
        self.master.title('Cavecalc Model Input GUI')
        self._load_defaults()
        self.construct_inputs()

    def _loop_gen(self, layout_numbers):
```

```
        ln = layout_numbers
        if type(ln) is int:
            ln = [ln]
        elif type(ln) is not list:
            raise TypeError("layout_numbers must be int or list of ints.")

        g = [(k, self.layout[k][1]) for k in self.settings.keys()
             if self.layout[k][0] in ln]

        return sorted(g, key=lambda tup: ns(tup[0]))

    def get_ln(self, key):
        """Gets the layout index number of 'key'"""

        try:
            return self.layout[key][1]
        except KeyError:
            return self.layout[ns(key)][1]

    def _load_defaults(self):

        self.d = SettingsObject()

        self.units = vars(cavecalc.data.types_and_limits).copy()
        self.layout = vars(cavecalc.gui.layout).copy()

        settings = self.d.dict()
        self.settings = py2tk(settings)

    def construct_inputs(self):
        """Frame 1 contains the left-hand panel of the input GUI."""

        def add_things_to_frame(frame, layout_number, header_text, i):
            l = Label(frame, text=header_text, font="-size 13")
            l.grid(row=i,columnspan=2, sticky=SW, pady=3)
            i += 1
            for a, b in self._loop_gen(layout_number):
                if b == 'A': # text with range
                    Label(frame, text=ns(a)).grid(row=i, sticky=W)
                    x = InputsRangeWidget(frame, self.settings[a])
                    x.grid(row=i, column=1, sticky=W)
                elif b == 'B': # text without range
                    Label(frame, text=ns(a)).grid(row=i, sticky=W)
                    x = Entry(frame, textvariable=self.settings[a], width=25)
                    x.grid(row=i, column=1, columnspan=2, sticky=W)
                elif b == 'C': # options menu
                    x = OptsWidget( frame, ns(a), self.settings[a],
                                    self.units[a], row=i )
                    x.grid(row=i, column=0, sticky=W)
                elif b == 'D': # check button
                    l = Label(frame, text=ns(a)).grid(row=i, sticky=W)
                    r = Checkbutton( frame, variable=self.settings[a],
                                     onvalue=True, offvalue=False )
                    r.grid(row=i, column=1)
                elif b == 'E': # load button
                    l = Label(frame, text=ns(a)).grid(row=i, sticky=W)
                    f = FileFindWidget( frame, value=self.settings[a],
                                        mode='load')
                    f.grid(row=i, column=1)
                elif b == 'F': # save button
                    l = Label(frame, text=ns(a)).grid(row=i, sticky=W)
                    f = FileFindWidget( frame, value=self.settings[a],
                                        mode='dir')
                    f.grid(row=i, column=1)
                i += 1
            return i

        px = 5 # padding between frames
        py = 2 # padding between frames


        F1 = Frame(self.master)
        Label(F1, text='Geochemical Inputs').grid(row=0,columnspan=3)
        i = 1
        i = add_things_to_frame(F1, 10, 'Atmospheric Gas End-member', i)
        i = add_things_to_frame(F1, 11, 'Soil Gas End-member', i)
        i = add_things_to_frame(F1, 12, 'Mixed Gas', i)
        i = add_things_to_frame(F1, 17, 'Cave Air', i)
        F1.pack(side='left', fill=Y, anchor='n', padx=px, pady=py)

        F2 = Frame(self.master)
        Label(F2, text=' ').grid(row=0,columnspan=3)
        i = 1
        i = add_things_to_frame(F2, 16, 'Soil Metals (Chloride Salts)', i)
        i = add_things_to_frame(F2, 13, 'Bedrock Chemistry', i)
        i = add_things_to_frame(F2, 14, 'Bedrock Dissolution Conditions', i)
        i = add_things_to_frame(F2, 15, 'General', i)
        F2.pack(side='left', fill=Y, anchor='n', padx=px, pady=py)
        self.F2 = F2

        F3 = Frame(self.master)
        Label(F3, text='Model Scripting Options').grid(row=0,columnspan=3)
        i = 1
        i = add_things_to_frame(F3, 2, 'Scripting Options', i)
        i = add_things_to_frame(F3, 3, 'Additional PHREEQC output', i)
        i = add_things_to_frame(F3, 4, 'File IO Settings', i)

        # add run button
        RunButton = Button( F3, text="Run!", command= lambda : \
                            self._run_models())
        RunButton.grid(row=i, columnspan=3, sticky=S, pady=20)
        i = i + 1

        # add link to output GUI
        LinkButton = Button( F3, text="Open Output GUI", command= lambda : \
                             CCAnalyseGUI(Toplevel(self.master)) )
        LinkButton.grid(row=i, columnspan=3, sticky=S)

        F3.pack(side='left', anchor='n', padx=px, pady=py)
        self.F3 = F3

    def _run_models(self):

        s = self.settings.copy()
        out_dir = s.pop('out_dir').get()
        d = {}

        d1 = {k:v for (k,v) in s.items() if self.get_ln(k) != 'A'}
        d2 = {k:v for (k,v) in s.items() if self.get_ln(k) == 'A'}
```

```
        d1 = ns(tk2py(d1, parse=False))
        d2 = ns(tk2py(d2, parse=True))

        d = {**d1, **d2}

        p = cavecalc.forward_models.ForwardModels(settings=d,
                                                  output_dir=out_dir)
        p.run_models()
        p.save()
        print("Done.")

class CCAnalyseGUI(object):
    """The Cavecalc Output GUI window."""

    def __init__(self, master):
        self.master = master
        self.master.title('Cavecalc Output GUI')
        self.e = Evaluate()
        self.dir = StringVar()
        self.dir.set(os.getcwd())
        self.loaded_dirs = []
        self.dnum = IntVar() # total no of models loaded
        self.dnum.set(0)
        self.settings_report = {}

        self.load_outputs_frame()
        self.save_buttons_frame()

    def _csv_dir_save(self):
        d = filedialog.askdirectory()
        if d:
            self.e.save_csvs(d)

    def _mat_save(self):
        d = filedialog.asksaveasfilename()
        if d:
            self.e.save_all_mat(file=d)

    def _add_data(self):
        """Loads data from the currently selected directory."""
        d = filedialog.askdirectory()

        if d:
            if d not in self.loaded_dirs:
                self.e.load_data(d)
                self.dnum.set(len(self.e.model_results))
                self.loaded_dirs.append(d)

    def load_outputs_frame(self):
        F0 = Frame(self.master)

        b = Button(master=F0, text="Load Model Output",
                   command = lambda : self._add_data())
        b.grid(row=0, column=0, columnspan=2)

        Label(F0, text="Models Loaded").grid(row=1, column=0, sticky=W)

        t1 = Entry(master=F0, textvariable=self.dnum, state='readonly')
        t1.grid(row=1,column=1)
        F0.pack()

    def save_buttons_frame(self):
        F1 = Frame(self.master)

        b1 = Button(master=F1, text="save as .csv",
                    command = lambda : self._csv_dir_save())
        b1.grid(row=0, column=0)
        b2 = Button(master=F1, text="save as .mat",
                    command = lambda : self._mat_save())
        b2.grid(row=0, column=1)
        b3 = Button(master=F1, text='Open Plotting Window',
                    command = lambda : PlottingWindow(self))
        b3.grid(row=0, column=2)

        F1.pack()

class PlottingWindow(Toplevel):
    def __init__(self, CCAnalyseGUI):
        super().__init__(CCAnalyseGUI.master)
        self.title('Cavecalc Plotting')

        self.e = CCAnalyseGUI.e
        self.o = self.e.model_results[0]
        self.s = self.e.model_settings[0]
        self.report = self.e.get_settings_report()

        lx = Label(self, text = "X variable (Required)")
        ly = Label(self, text = "Y variable (Required)")
        ll = Label(self, text = "Label with (Optional)")
        X_Sel, self.x = self.OutputSelectWidget()
        Y_Sel, self.y = self.OutputSelectWidget()
        L_Sel, self.l = self.SettingSelectWidget()

        lx.grid(row=0, column=0, sticky=W)
        X_Sel.grid(row=0, column=1)
        ly.grid(row=1, column=0, sticky=W)
        Y_Sel.grid(row=1, column=1)
        ll.grid(row=2, column=0, sticky=W)
        L_Sel.grid(row=2, column=1)

        # Radiobutton selector for data filtering (for plot)
        self.v = IntVar()
        self.v.set(0)
        r0 = Radiobutton(   self, text="Full Model (inc. initial solution)",
                            variable=self.v, value=0)
        r1 = Radiobutton(   self, text="Full Model (excl. inital solution)",
                            variable=self.v, value=1)
        r2 = Radiobutton(   self, text="Bedrock Dissolution Solution only",
                            variable=self.v, value=2)
        r3 = Radiobutton(   self, text="End Point Solution only",
                            variable=self.v, value=3)
        r4 = Radiobutton(   self, text="Precipitation Steps only",
                            variable=self.v, value=4)


        r0.grid(row=3)
        r1.grid(row=4)
        r2.grid(row=6)
```

```
        r3.grid(row=7)
        r4.grid(row=5)

        b = self.PlotButton()
        b.grid(row=8,columnspan=2)

    def SettingSelectWidget(self):
        v = StringVar()
        opt = list(self.report.keys())
        if HIDDEN_OPTS:
            opt = [o for o in opt if o not in HIDDEN_OPTS]

        o = []
        for entry in opt:
            try:
                o.append(ns(entry))
            except KeyError:
                o.append(entry)
        return OptionMenu(self, v, *sorted(o)), v

    def OutputSelectWidget(self):
        v = StringVar()
        opt = list(self.o.keys())
        return OptionMenu(self, v, *sorted(opt)), v

    def PlotButton(self):

        b = Button(self, text='Plot Graph',
                   command = lambda : self.plot())
        return b

    def plot(self):

        if self.v.get() == 1:
            a = self.e.filter_by_index(ind=0, n=True)
        elif self.v.get() == 2:
            a = self.e.filter_by_index(ind=1)
        elif self.v.get() == 3:
            a = self.e.filter_by_index(ind=-1)
        elif self.v.get() == 4:
            a = self.e.filter_by_results('step_desc', 'precip')


        else:
            a = copy.deepcopy(self.e)

        f = lambda x : None if x == '' else x
        x_lab = f(self.x.get())
        y_lab = f(self.y.get())
        lab_name = f(self.l.get())

        x = []
        y = []
        for v in a.model_results:
            x.append(v[x_lab])
            y.append(v[y_lab])

        if lab_name:
            labs = [s[ns(lab_name)] for s in a.model_settings]
        else:
            labs = None

        print("Plotting...")
        gplot( x, y, x_lab, y_lab, labs, lab_name )

if __name__ == '__main__':
    CCInputGUI()
    # CCAnalyseGUI()
```

## A.3.14 cavecalc/gui/layout.py

```
"""Codes locations and type of each parameter in the GUI input panel.

    0: Hidden
    10: Geochem: Atmospheric End-Member
    11: Geochem: Soil End-Member
    12: Geochem: Soil gas mixing
    13: Geochem: Bedrock chemistry
    14: Geochem: Bedrock dissolution conditions
    15: Geochem: General (e.g. temp)
    16: Geochem: Soil Metals
    17: Geochem: Cave Air
    2: Scripting options
    3: Additional PHREEQC Output
    4: IO Settings

Type:
    A: Text-with-range
    B: Text without range
    C: Options Menu
    D: Check button
    E: File browser - load
    F: File browser - select directory
    """

database =          (4, 'E')
phreeqc_log_file =      (4, 'D')
phreeqc_log_file_name = (0, 'B') # no reason to change
temperature =           (15, 'A')
co2_decrement =         (2, 'A')
calcite_sat_limit =     (2, 'A')
bedrock =               (14, 'A')
bedrock_mineral =       (13, 'A')
bedrock_pyrite =        (14, 'A')
bedrock_d44Ca =         (13, 'A')
bedrock_d13C =          (13, 'A')
bedrock_d18O =          (13, 'A')
bedrock_MgCa =          (13, 'A')
bedrock_SrCa =          (13, 'A')
```

```
bedrock_BaCa =          (13, 'A')
atmo_exchange =     (12, 'A')
gas_volume =        (14, 'A')
atm_pCO2 =          (10, 'A')
atm_d13C =      (10, 'A')
atm_R14C =      (10, 'A')
atm_d18O =      (15, 'A')
init_pCO2 =     (12, 'A')
init_d13C =     (12, 'A')
init_R14C =     (12, 'A')
init_solution_d13c = (0, 'A')    # not exposed in GUI useage
soil_pH =       (0, 'A')        # pH is charge-balanced - initial estimate is unimportant
soil_pCO2 =     (11, 'A')
soil_Ca =       (16, 'A')
soil_Mg =       (16, 'A')
soil_Sr =       (16, 'A')
soil_Ba =       (16, 'A')
soil_d13C =     (11, 'A')
soil_R14C =     (11, 'A')
soil_d44Ca =    (16, 'A')
cave_O2 =       (17, 'A')
cave_pCO2 =     (17, 'A')
cave_d13C =     (17, 'A')
cave_R14C =     (17, 'A')
cave_d18O =     (17, 'A')
cave_air_volume = (17, 'A')
kinetics_mode = (2, 'C')
reprecip =      (14, 'D')
totals =        (3, 'B')
molalities =    (3, 'B')
isotopes =      (3, 'B')
out_dir =       (4, 'F')
soil_O2 =       (11, 'A')
atm_O2 =        (10, 'A')
init_O2 =       (12, 'A')
```

## A.3.15 cavecalc/gui/mapping.py

```
"""Maps between model names and more readable versions.

Model/code names (on the left) are used internally by Cavecalc. Readable names
are largely used in the GUI. Conversion between them is handled by the
setter.NameSwitcher class.
"""

database =          "PHREEQC Database Filename"
phreeqc_log_file =      "Log PHREEQC input"
phreeqc_log_file_name = "PHREEQC Log Filename"
out_dir =               "Output Directory"
temperature =           "Temperature (Degrees C)"
co2_decrement =         "CO2(g) removal per step (fraction)"
calcite_sat_limit =     "Calcite supersaturation limit (SI)"
bedrock =               "Bedrock (moles)"
bedrock_mineral =       "Bedrock Lithology"
bedrock_pyrite =        "Bedrock Pyrite (moles)"
bedrock_d44Ca =         "Bedrock d44Ca (per mil)"
bedrock_d13C =          "Bedrock d13C (per mil)"
bedrock_d18O =          "Bedrock d18O (per mil)"
bedrock_MgCa =          "Bedrock Mg/Ca (mmol/mol)"
bedrock_SrCa =          "Bedrock Sr/Ca (mmol/mol)"
bedrock_BaCa =          "Bedrock Ba/Ca (mmol/mol)"
atmo_exchange =     "Atmospheric Gas Exchange (0-1)"
gas_volume =        "Gas Volume (L)"
atm_O2 =        "Atmospheric O2 (%)"
atm_pCO2 =      "Atmospheric pCO2 (ppmv)"
atm_d13C =      "Atmospheric d13C (per mil)"
atm_R14C =      "Atmospheric R14C (pmc)"
atm_d18O =      "Rainfall d18O (per mil)"
init_O2 =       "Initial O2 (%)"
init_pCO2 =     "Initial pCO2 (ppmv)"
init_d13C =     "Initial d13C (per mil)"
init_R14C =     "Initial R14C (pmc)"
init_solution_d13c = "Initial Solution d13C (per mil) - OVERWRITE"
cave_O2 =       "Cave Air O2 (%)"
cave_pCO2 =     "Cave Air pCO2 (ppmv)"
cave_d13C =     "Cave Air d13C (per mil)"
cave_R14C =     "Cave Air R14C (pmc)"
cave_d18O =     "Cave Air d18O (per mil)"
cave_air_volume = "Cave Air Volume (L)"
soil_pH =       "Soil pH (approx.)"
soil_O2 =       "Soil Gas O2 (%)"
soil_pCO2 =     "Soil Gas pCO2 (ppmv)"
soil_Ca =       "Soil Ca (mmol/kgw)"
soil_Mg =       "Soil Mg (mmol/kgw)"
soil_Sr =       "Soil Sr (mmol/kgw)"
soil_Ba =       "Soil Ba (mmol/kgw)"
soil_d13C =     "Soil Gas d13C (per mil)"
soil_R14C =     "Soil Gas R14C (pmc)"
soil_d44Ca =    "Soil d44Ca (per mil)"
kinetics_mode =         "Run Mode"
reprecip =              "Allow Calcite Reprecipitation"
totals =        "Totals"
molalities =    "Molalities"
isotopes =      "Isotopes"
```

## A.3.16 scripts/cc_input_gui.py

```
from cavecalc.gui.gui import CCInputGUI
import tkinter as tk

if __name__ == '__main__':
    root = tk.Tk()
    app = CCInputGUI(root)
    root.mainloop()
```

### A.3.17 scripts/cc_output_gui.py

```
from cavecalc.gui.gui import CCAnalyseGUI
import tkinter as tk

if __name__ == '__main__':
    root = tk.Tk()
    app = CCAnalyseGUI(root)
    root.mainloop()
```

### A.3.18 examples/example1.py

```
"""
Runs Cavecalc with default settings and:
    1) Save a csv file with the results
    2) Make a plot
"""

from cavecalc.forward_models import ForwardModels
import cavecalc.analyse as cca

#
## STEP 1: Run the model
#
dir = './ex1/' # directory to save model output
p = ForwardModels( output_dir = dir )    # initialise the model with default inputs
p.run_models()                              # run the model
p.save()                                    # save the model output

#
## STEP 2: Load model output and do stuff with it
#
e = cca.Evaluate()      # initialise an 'Evaluate' object
e.load_data( dir )      # load data from dir
e.save_csvs( dir )      # save the model output to a .csv file

#
## STEP 3: Make a plot of pH against Ca concentration. Use red crosses.
#
e2 = e.filter_by_index(0, n=True) # filter out the first model step
e2.plot_models('-xr', x_key='Ca(mol/kgw)', y_key='d13C')
cca.plt.show()
```

### A.3.19 examples/example2.py

```
"""
Run two models, one with dolomite bedrock and one with calcite bedrock. All
other parameters are set to default values.

Also save phreeqc log file recording how the model ran.
"""

from cavecalc.forward_models import ForwardModels
import cavecalc.analyse as cca

#
## STEP 1: Define the non-default settings and run the model
#

s =  {  'bedrock_mineral' :       ['Calcite', 'Dolomite'],
        'phreeqc_log_file':       True,
        'cave_pCO2' :             1000
     }

dir = './ex2/' # directory to save model output
p = ForwardModels(  settings =      s,
                    output_dir  =   dir )   # initialise the model
p.run_models()                              # run the model
p.save()                                    # save the model output

#
## STEP 2: Load model output and save new data formats
#
e = cca.Evaluate()      # initialise an 'Evaluate' object
e.load_data( dir )      # load data from dir
e.save_all_mat( dir )   # save data to a .mat file
e.save_csvs( dir )      # save each model to a .csv file

#
```

```
## STEP 3: Plot pH vs Ca concentration.
#
e2 = e.filter_by_index(0, n=True) # filter out the first model step
e2.plot_models(x_key='Ca(mol/kgw)', y_key='pH', label_with = 'bedrock_mineral')
cca.plt.show()
```

## A.3.20 examples/example3.py

```
"""Run a suite of models investigating the effect of temperature on the bedrock
dissolution step under open system conditions. Do it for both Calcite and
Dolomite
"""

from cavecalc.forward_models import ForwardModels
import cavecalc.analyse as cca

#
## STEP 1: Define the non-default settings and run the model.
# use diss_only mode because we don't care about the degassing phase.

s =  {   'bedrock_mineral' :       ['Calcite', 'Dolomite'],
         'temperature'     :       [20, 22, 24, 26, 28, 30],
         'gas_volume'      :       20,
         'init_pCO2' :             15000,
         'init_d13C' :            -15,
         'kinetics_mode' :         'diss_only',
         'cave_pCO2' :             500
     }

dir = './ex3/' # directory to save model output
p = ForwardModels(  settings =       s,
                    output_dir  =    dir )   # initialise the model
p.run_models()                               # run the model
p.save()                                     # save the model output

#
## STEP 2: Load model output and do stuff with it
#
e = cca.Evaluate()       # initialise an 'Evaluate' object
e.load_data( dir )       # load data from dir

#
## STEP 3: Plot pH vs Ca concentration.
#

# plot_points plots a single point from each model
e1 = e.filter_by_settings('bedrock_mineral', 'Calcite')
e2 = e.filter_by_settings('bedrock_mineral', 'Dolomite')

e1.plot_points( x_key='Ca(mol/kgw)', y_key='pH',
                label_with = 'temperature',
                plot_index = 1 )

a1 = e1.plot_points( x_key='temperature', y_key='Ca(mol/kgw)',
                label_with = 'bedrock_mineral',
                plot_index = 1 )

e2.plot_points( x_key='temperature', y_key='Ca(mol/kgw)',
                label_with = 'bedrock_mineral',
                plot_index = 1, ax=a1 )
cca.plt.show()
```

## A.3.21 examples/example4.py

```
"""A more advanced API example - this is not possible using the GUI.

Here we undergo a degassing loop similar to running Cavecalc in 'eq' mode,
but also allow some amount of equilibration with the bedrock on each model
step. 3 models are run, each with a different amount of equilibration allowed.

In previous examples we use the ForwardModels object to handle the running of
the model. Here we skip that step and directly access the objects in caves.py
to allow us more flexibility.

Consult the documentation for caves.py for detailed info on all methods and
objects used.
"""

from cavecalc.caves import *
from cavecalc.setter import SettingsObject
import cavecalc.util as ccu
import matplotlib.pyplot as plt

s =  {   'bedrock_mineral' :       'Calcite',
         'temperature'     :       32,
         'gas_volume'      :       20,
         'init_pCO2' :             15000,
         'init_d13C' :            -15,
         'phreeqc_log_file' :      True,
         'cave_pCO2' :             500
     }

# we run three models, each with a different amount of bedrock equilibration
for m_bedrock in [0, 1e-5, 1e-3]: # three values for m (moles of bedrock)

    # A SettingsObject fills in s with default values for all other parameters
    # and checks our specified inputs are of valid types.
    S0 = SettingsObject(**s)
```

```
    # sim - a Simulator object - is the core of the model. It handles all IO
    # from IPhreeqc. We initialise one by providing it with a SettingsObject.
    sim = Simulator(S0)

    # initialise bedrock and water objects. These are associated with the
    # simulator 'sim' and get their initial settings from it.
    bedrock = Carbonate(sim)
    water = Solution(sim)

    # Here we do the first reaction. water.set_soil_chem() returns PHREEQC
    # input to define the initial soil chemistry. This is added to the Iphreeqc
    # buffer, to be executed later by sim.ipq_exec().
    sim.ipq_buffer( *water.set_soil_chem() )
    sim.ipq_exec()    #     Calculate initial soilwater chemistry

    # Next reaction: dissolve bedrock in the soilwater.
    sim.ipq_buffer(*water.open_diss_rxn( bedrock ))
    sim.ipq_exec()    #     Calculate bedrock dissolution product

    # Now do an interative degassing loop until 'degas_pCO2' is reached.
    while water.will_it_degas( sim.settings['degas_pCO2'] ):
        # on each loop, add 3 reactions to the buffer.
        # water.wri_rxn is the one not used in the standard 'eq' mode
        sim.ipq_buffer([ water.wri_rxn(bedrock, m_bedrock=m_bedrock),
                         water.degas_rxn(),
                         water.precipitate_rxn()
                       ])
        sim.ipq_exec()

    ccu.PostProcessor(sim) # perform offline calculations and processing

    #### look at output
    x = 'f_ca'
    y = 'dT3C'
    plt.figure()
    plt.plot(sim.output[x][1:], sim.output[y][1:])
    plt.xlabel(x)
    plt.ylabel(y)

plt.show()
```