Letter Report: EQ3/6 Version 8: Differences from Version 7

Thomas Wolery

September 29, 1994

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This report has been reproduced
directly from the best available copy.

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information
P.O. Box 62, Oak Ridge, TN 37831
Prices available from (423) 576-8401

Available to the public from the
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd.,
Springfield, VA 22161

LLNL Yucca Mountain Project
Near-Field Environment Characterization Technical Area

Letter Report:
EQ3/6 Version 8: Differences From Version 7

Thomas J. Wolery

September 29, 1994

University of California

**Lawrence Livermore
National Laboratory**

EQ3/6 Version 8: Differences From Version 7

Summary

EQ3/6 is a software package for geochemical modeling of aqueous systems, such as water/rock or waste/water rock. It is being developed for a variety of applications in geochemical studies for the Yucca Mountain Site Characterization Project. The software has been extensively rewritten for Version 8. The source code has been extensively modernized. The software is now written in Fortran 77 with the most common extensions that are part of the new Fortran 90 standard. The architecture of the software has been improved for better performance and to allow the incorporation of new functional capabilities in Version 8 and planned subsequent versions. In particular, the structure of the major data arrays has been significantly altered and extended. Three new major functional capabilities have been incorporated in Version 8. The first of these allows the treatment of redox disequilibrium in reaction-path modeling. This is a natural extension of the long-running capability of providing for such disequilibrium in static speciation-solubility calculations. Such a capability is important, for example, when dealing with systems containing organic species and certain dissolved gas species. The user defines (and sets the controls for) the components in disequilibrium. Such corrections can now be made if the requisite data are present on a supporting data file. At present, this capability is supported only by the SHV data file, which is based on SUPCRT92. Equilibrium constants and other thermodynamic quantities are corrected for pressures which lie off a standard curve, which is defined on the supporting data file and ordinarily corresponds to 1.013 bar up to 100C, and the steam/liquid water equilibrium pressure up to 300C. The third new major capability is generic ion exchange option previously developed in prototype in a branch Version 7 level version of EQ3/6 by Brian Viani, Bill Bourcier, and Carol Bruton. This option has been modified to fit into the Version 8 data structure, which allows avoidance of some problems that occurred with the prototype. This capability allows the user to define exchange phases with multiple sites, specifying the exchange reactions and relevant thermodynamic data. Some minor improvements have also been made. The major option switch arrays for EQ3NR and EQ6 (**iopt**, **iopr**, **iodb**) have been made identical for both codes (though some elements may only pertain to one code or the other. EQ3NR now accepts alkalinity in units of mg/L equivalent CaCO_3 , in accord with the usual method of presenting data for groundwater chemistry. A previous treatment of alkalinity had been removed from the Version 7 series. Supporting data files may now utilize temperature-pressure grids with arbitrary numbers of temperature ranges and temperature points per range, instead of the old EQ3/6 standard of two temperature ranges with four and five points (one point being common to both temperature ranges). Version 8 of EQ3/6 also includes utility codes to convert EQ3NR and EQ6 input files from version levels 7 and 7.2 level to version level 8.

EQ/6 Version 8 Letter Report, Sept. 29, 1994

1.0 Introduction

This document is intended to describe Version 8 of EQ3/6. Version 7 and its usage has been extensively described both in formal code documentation (Wolery, 1992ab; Daveler and Wolery, 1992; Wolery and Daveler, 1992) and in more concise technical review (Wolery et al., 1990). Many other useful references may be found in these documents. EQ3/6 is being developed to model geochemical processes of relevance to the Yucca Mountain Site Characterization Project. The range of applications is quite broad, including studies pertaining to ambient groundwater geochemistry, waste form dissolution, and rock/water interactions in the altered zone that would develop around a nuclear waste repository. In some of this work the software is used to design or analyze experiments, in others, to predict processes that would occur in or about a repository. The software is also being applied in some natural analog studies.

EQ3/6 is a package which contains various code elements. EQ3NR is a speciation-solubility code. It is used for example to compute the speciation of ions and solutes in groundwaters, and the saturation indices which define the saturation states of such waters with respect to various minerals. EQ6 is a reaction-path code. It allows one to compute simulations of interactions as among groundwaters and minerals. EQPT is a data file preprocessor. XCON3 and XCON6 are input file reformatters for EQ3NR and EQ6, respectively. EQLIB is a supporting library. For Version 8, some parts of this have been spun off into two new libraries, EQLIBU and EQLIBG.

Due to the importance of the work of the Yucca Mountain Site Characterization Project involving the application of this software, it is essential to develop QA-certified versions of the software for use in quality-affecting work. The first version of EQ3/6 that was so certified was Version 7.2a, which was released on August 17, 1994. A certified form of Version 8 is expected later in FY95, after the completion of a required independent V & V (verification and validation) activity.

EQ3/6 up to and including Version 7 has a long history, and incorporates many of the capabilities needed for use in various kinds of geochemical modeling studies (see Wolery, 1992a and Wolery et al., 1990). However, neither it nor any other geochemical modeling software offers all of the capabilities that are currently perceived as necessary or highly useful in the broad range of geochemical studies required in connection with the geologic disposal of nuclear waste (see Wolery, 1992a). Thus, new development is required. In some cases, the software must be co-developed with some of the requisite theory, mathematical description, and data for certain phenomena, such as waste form dissolution.

Through Version 7, EQ3/6 has been repeatedly extended by the addition of various new capabilities (e.g., precipitation kinetics, Delany et al., 1986). The software has also been moved through a succession of computer platforms. It was originally developed on a CDC 6400 computer; now, it is supported on UNIX platforms (e.g., Sun SPARCstations and SGI workstations) and 486/Pentium PCs. However, the basic structure of the software had never been changed. This was influenced by the limited range of problems that the software was originally designed to address. It was also limited by the characteristics of the computer hardware available when the software was first developed. Memory, for example, often posed a major limitation. The software was further influenced by the compilers and language standards that were originally

EQ/6 Version 8 Letter Report, Sept. 29, 1994

extant, although in this respect there has been a steady evolution of the software. Nevertheless, the architecture of the major data arrays in Version 7 of EQ3/6 had become a major obstacle to further development. It was time for a major re-write, which has been accomplished in Version 8.

In re-writing the software, it was recognized that it was necessary to preserve the value of the existing documentation and V & V history associated with Version 7. In order to facilitate regression testing of Version 8 against Version 7 (using the rather extensive EQ3/6 test case library), a conservative approach has been taken with regard to implementing various possible algorithmic changes, some of which are only now possible due to the new data array structure. For example, certain types of basis switching, as of a mineral for an aqueous species, have been temporarily put off until an accepted baseline version has been established for Version 8.

The architectural changes to Version 8 facilitated the introduction of the three major new code capabilities incorporated into this version: optional redox disequilibrium in reaction-path calculations (in the EQ6 code), thermodynamic pressure corrections, and a generic ion-exchange modeling capability. Aspects of the language and architecture changes are described in the following section (with additional details given in Appendix A). The sections that follow deal in turn with each of the new major capabilities, followed by one addressing new minor capabilities.

2.0 Language and Architecture Changes

Version 7 was technically written in Fortran 77 (ANSI, 1978). A few common extensions to that standard were used (e.g., use of INCLUDE files). A new Fortran 90 standard (ANSI, 1991) has appeared, in which many common Fortran 77 extensions (e.g., DO-ENDDO structures) have been adopted as standard. Unfortunately, few Fortran 90 compilers have yet appeared. Version 8 has been rewritten using a level of Fortran which is essentially Fortran 77 with many extensions that are standard under Fortran 90. It is thus in a "pre-F90" state. The new coding is much cleaner than the old, as there is little usage of statement labels (mostly due to the use of DO-ENDDO structures), and language constructs that are generally considered to be obsolete (e.g., arithmetic IF statements) have been purged from the code.

Several other coding improvements have been made. The number of COMMON blocks (and hence the number of INCLUDE files) has been greatly diminished. Most of the flow of data between subroutines is handled using calling sequences. All such data flow would be handled in this manner. However, Fortran 77 imposes some limits on how much data can be transferred in a calling sequence, so some COMMON block usage is still required. In general, COMMON blocks are only employed in main programs and large subroutines. When possible, the mechanism of data flow is shifted to calling statements, and once a variable is so transmitted, there is no further COMMON block reference to it down the calling chain. All variable names are now explicitly typed. The statement IMPLICIT NONE is present in every subroutine to enforce this. However, the previous EQ3/6 convention relating the first letter in a variable name to the type of the variable has been retained (starting letters **i**, **j**, **k**, and **n** denote type INTEGER, starting letter **q**, type LOGICAL, starting letter **u**, type CHARACTER, all others, type REAL*8).

EQ/6 Version 8 Letter Report, Sept. 29, 1994

COMMON block variables are now typed in the corresponding INCLUDE file. Calling sequence variables are typed within a subroutine in a separate block from local variables. The former begin with the header:

```
c      Calling sequence variable declarations.  
c
```

The latter begin with the header:

```
c      Local variable declarations.  
c
```

In Version 7, the error, warning, and note messages were expanded to be largely self-contained. This trend was continued in the development Version 8. All such messages are now completely self-contained and include suggested actions where appropriate.

In Version 8, the EQLIB library has been split into three libraries: EQLIB, EQLIBG, and EQLIBU. EQLIBU is now the EQ3/6 utility and math library. It contains a number of utility routines, many of which function as language extensions. These include various routines which were previously duplicated in the XCON3 and XCON6 input file reformatter codes. The linear equation solver subroutine **msolvr.f** and various math routines which support it are also in EQLIBU. The supporting routines which had been modified from a public-domain version of LINPACK have been changed to conform with the standard LINPACK names for REAL*8 arithmetic on 32-bit machines (e.g., **saxpy.f** is now **daxpy.f**) and calling sequences to facilitate their possible replace on specific platform with platform-optimized equivalents. Most of the math routines in EQLIBU (LINPACK equivalent or not) have been written making extensive use of unrolled loops. EQLIBG now contains all of the EQ3/6 subroutines dealing with activity coefficients, except for **ngcadv.f** (updates activity coefficients in Newton-Raphson iteration), which remains in EQLIB. XCON3 and XCON6 must now be loaded with EQLIB and EQLIBU. In Version 7, these codes were self-contained. EQPT must now be loaded with EQLIBU as well as EQLIB. EQ3NR and EQ6 must now be loaded with EQLIBG and EQLIBU, in addition to EQLIB.

In a number of cases, a pair of analogous routines in EQ3NR and EQ6 have been replaced in Version 8 by a single EQLIB routine (e.g., EQ3NR's **indatx.f** and EQ6's **indatz.f** by EQLIB's **indata.f**; these functions read the supporting data file). Other key pairs of EQ3NR/EQ6 modules which have been replaced by a single EQLIB routine are as follows. The pair **betas.f/betaz.f**, which compute Newton-Raphson residual functions, has been replaced by **beta.f**. The pair **matrix.f/matrixz.f**, which compute the Jacobian matrix used in Newton-Raphson iteration, has been replaced by a new **matrix.f**. The pair **ncmpx.f/ncmpz.f**, which expand the system description from values of the primary Newton-Raphson variables, has been replaced by **ncmpex.f**. The actual functions of all of these routines are slightly different for EQ3NR and EQ6. Accordingly, a variable called **q6mode** has been placed in the calling sequence. If **q6mode** is false, the calculations are made for EQ3NR, if true, for EQ6.

EQ/6 Version 8 Letter Report, Sept. 29, 1994

The EQ3NR module **scripx.f** oversees the writing of data on the output file. The EQ6 module **scripz.f** is closely analogous, overseeing the writing on the output file of data describing the system at a given point of reaction progress. These have retained their separate identities, as not all of the output data are common. Coding for the writing of common data tables has been stripped out of these modules and put in new EQLIB modules (whose names all start with “prt” for “print table”).

All dimensioning for EQ3NR and EQ6 is now controlled by Fortran parameters set in PARAMETER statements in the EQLIB INCLUDE file **eq1par.h**. Previously, EQ3NR and EQ6 also had their own analogous INCLUDE files, which provided part of this function. Each dimensioning parameter has a corresponding dimensioning variable (e.g., parameter **nctpar** and variable **nctmax**). Dimensioning variables are used to pass dimensioning data through calling sequences.

Name substitutions in calling sequences are occasionally invoked in subroutine calls. To help avoid losing any necessary substitutions during code development, each substitution is now noted in a comment statement immediately above the call. This is done using the following standard format:

```
c      Calling sequence substitutions:
c          a for x
c          b for y
c
c      call subr(n,a,b)
```

In Version 7, the **iopt**, **iopr**, **iodb**, and **iopeg** option switches were set of scalar variables. For example the **iopt** switches were, **iopt1**, **iopt2**, **iopt3**, etc. In Version 8, they are arrays (e.g., **iopt1** is now **iopt(1)**, **iopt2** is now **iopt(2)**, and so forth). Some of these switches had different functions in EQ6 than in EQ3NR, or the values corresponded to different option selections. All of these switches are now the same for both codes. In some cases, a given switch may only be relevant to one code or the other. The EQ6 setscrew variables **screw1**, **screw2**, etc. have been likewise combined into the **sscrew** array.

In the Version 7 EQ6, the data objects which contained rate law parameters existed in a dual structure, one part applying to reactions going in the forward direction, the other, the backward direction. For example, **nrk(nrc)** was the rate law code for the forward direction, **nrpk(nrc)** that for the backward direction. In the Version 8 code, all such paired data objects have been combined, and the issue of directionality has been addressed by the addition of an extra dimension. For example, **nrk(nrc)** has been replaced by **nrk(1,nrc)** and **nrpk(nrc)** by **nrk(2,nrc)**.

In Version 7, the structures of the major data arrays in the EQ3NR and EQ6 codes were not identical. In EQ3NR, mass balances were (and still are) defined in terms of a set of chemical components known as basis species (e.g., Na^+ , Fe^{2+} , SO_4^{2-}). This consists of two parts, a strict basis set, in which each basis species corresponds one-to-one with a chemical element (e.g., Fe^{2+} to Fe) plus $\text{O}_{2(g)}$, which is used as the redox species, and an auxiliary basis set, which consists of

EQ/6 Version 8 Letter Report, Sept. 29, 1994

similar species representing for the most part corresponding chemical elements in different oxidation states (e.g., Fe^{3+} , HS^-). In EQ6, in contrast, mass balances were defined in terms of chemical elements, and the total basis set was equivalent to just a strict basis set. In EQ3NR, an auxiliary basis species may be treated as either an active basis species (in which case it has a corresponding mass balance) or a dependent (non-basis) species (in which case it has no corresponding mass balance of its own). The user controls which is the case through the **jflag** array: a **jflag** value of 30 marks the corresponding species to be treated as a dependent species. In Version 7, an EQ3NR-generated model of an aqueous solution fed into an EQ6 calculation would first proceed by a forced equilibration step in which any auxiliary basis species not in equilibrium with their strict basis counterparts would be put into such equilibrium and treated as dependent species.

In Version 8, the auxiliary basis concept originally in EQ3NR has been implemented in EQ6. The **jflag** control array has been carried over as part of this and **jflag** data now appear on the EQ6 input file. The redox disequilibrium capability in the new EQ6 is itself discussed further in the following section. However, the move from defining mass balances in terms of chemical elements to ones in terms of basis species has implications apart from the introduction of an auxiliary basis concept, and these are discussed below.

As noted in the EQ3NR User's Guide for Version 7 (Wolery, 1992b), mass balances in terms of basis species may or may not correspond to physical quantities. If not, they correspond to linear transformations of physical quantities. In EQ3/6 up through Version 7, and continuing in Version 8 and subsequent versions, physical meaning of mass balance totals for basis species as defined on the supporting data files is insured by requiring each basis species to consist of no more than one chemical element other than oxygen and hydrogen. The basis species $\text{H}_2\text{O}_{(l)}$, H^+ , and $\text{O}_{2(g)}$ are exceptions in that physically meaning mass balance totals for these species are unrealizable no matter what one does. However, confusion over physical meaning of mass balance totals for these species is unlikely, as such quantities are not used for analytical or descriptive purpose outside the realm of purely calculational operations. For example, one never talks of the total molality of H^+ ; in stead, one talks of the pH. In the Version 7 EQ6, the balance relation corresponding to the redox species was the charge balance equation. In Version 8, this is replaced by a mass balance relation for $\text{O}_{2(g)}$.

Mass balances in terms of basis species are defined using stoichiometric factors defined from chemical reactions. In the Version 7 EQ6, special reactants could be defined by the user by specifying the corresponding elemental composition. In the Version 8 code, a formal reaction is also required in order to adjust the system mass balance totals to reflect an increment of reaction involving such a reactant. The XCON6 input file reformatter composes such a reaction if necessary when converting to Version level 8. The Version 8 EQ6 also does this itself, if necessary. However, in either case, the composed reaction is written in terms of strict basis species, which may not always be the reaction desired. Users may have to compose the desired reaction themselves.

Problems may occur in the definition of mass balances if one rewrites the reactions. This could happen, in a basis switching operation in which for example, HgCl_3^- is switched into the basis set

EQ/6 Version 8 Letter Report, Sept. 29, 1994

for Hg^{2+} (one might do this to improve convergence of associated calculations). Because the Cl^- in the HgCl_3^- is part of that basis species, it does not count in the mass balance for Cl^- (which is itself another basis species). Even more peculiar results may follow (see Wolery, 1992b, p. 72-74). To preserve maximum physicality, it is necessary to define mass balances in terms of the data file basis set. This was accomplished in the Version 7 EQ3NR by a complicated manner of calculating the stoichiometric factors in a way that effectively undid any switch.

In Version 8, this problem has been addressed by implementing a dual-basis approach in which all basis-dependent quantities are available for both the "data file" basis set (the "d" set) and a regular set derived from it which may be affected by basis switching. The "d" set is not an image of the actual basis-dependent quantities as read from the data file. It is effected by the paring of unused basis species, by the elimination from the active basis set of auxiliary basis species for which the *jflag* elements are set to 30, and by so-called "special" basis switching, which is peculiar to Version 8 and are described below. The objective of having the regular set is to optimize the numerical aspects of calculation through "ordinary" basis switching. The objective of having the "d" set is to provide a constant reference for input and output so that the user does not become confused in setting up code input or (more importantly) analyzing code output. The "d" set is used exclusively as the i/o reference. Thus, an output value for the total concentration of a dissolved species always has the meaning that matches the quantity one would get from a chemical analysis. Also, quantities that are output for reactions (e.g., saturation indices) are always referenced to the reactions in the "d" set.

In Version 7, EQ3NR and EQ6 would first read a problem from the input file, then read the data file, extracting only the data needed to solve the problem at hand. They would then solve that problem and look for another problem on the input file, re-read the data file, and continue, until no new problems were found. In Version 8, each code now reads the data file only once. A complete image of the data file is maintained in a so-called "a" set of data. When a problem is read from the input file, the "d" set is constructed from this "a" set by a new EQLIB module called **cmpdat.f**.

In Version 7, the basis set was defined as the first group of aqueous species, and basis switching changed the indices of the affected species in the aqueous species list. Furthermore, only aqueous species could be switched into the basis set, because the array structure employed was such that different arrays were used for corresponding quantities, depending on the species type (aqueous, pure mineral, gas, etc.). For example, the names of the aqueous species were kept in the **uspec** array, those of the pure minerals in the **umin** array, and those of the gas species in the **ugas** array, and so forth. This treatment followed for all related quantities, from molecular weights to reaction coefficients and equilibrium constants.

In Version 8, there is one list of species. Continuous ranges in this list contain the species belonging to various phases. For example, the **uspec** array now contains the names of all species, regardless of type. A pointer array (**ncmpr**) gives the indices of the first and last species belonging to a given phase. The basis sets are defined by pointer arrays. The **nbaspd** array contains the indices of all species in the "d" basis set, the **nbaspr** array those of all species in the regular set. Counterparts of such arrays exist in the "a" set (e.g., **uspeca**, **ncmpr**, **nbaspa**), from which these arrays are constructed.

EQ/6 Version 8 Letter Report, Sept. 29, 1994

In Version 8, there is an explicit phase list, represented by the **uphase** array, which contains the names of all phases. This is arranged so that all phases of a given type (e.g., the aqueous solution, the pure minerals, the solid solutions) appear in contiguous blocks. The beginnings and endings of some such blocks are marked by certain variables (e.g., **imrn1** and **imrn2** for the pure minerals, **ixrn1** and **ixrn2** for the solid solutions). An explicit phase list was lacking in Version 7.

One consequence of explicit species and phase lists is that components of solid solutions are now treated in a different manner. In Version 7, the species albite as a component of a solid solution called plagioclase was represented in a more limited manner in which the thermodynamic properties of this species were obtained at all times from those of the pure mineral albite. The two species, for example, shared the same equilibrium constant. In Version 8, a species such as albite in plagioclase is created by cloning the corresponding pure mineral (this is done by a new EQLIB module called **clones.f**). The equilibrium constants upon cloning are identical in value, but represented by separate data objects and hence separably manipulable, for example by an **nxmod** alter option. A pointer array identifying the pure minerals serving as end-member components of solid solutions (the **nend** array) is no longer maintained.

The fixed fugacity option in EQ6 is implemented by the creation of fictive pure minerals (see Wolery and Daveler, 1992). In Version 7, these minerals were created as new pure minerals by cloning the relevant gas species. In Version 8, these fictive pure minerals occupy their own range in the list of phases. The variables **ifrn1** and **ifrn2** mark the corresponding range.

A more compact and efficient array structure for storing elemental composition and reaction coefficient data is now employed in Version 8. This avoids storing a lot of zeroes and depends heavily on the use of pointer arrays. The details are not relevant to understanding the functionality of the software, and for a detailed description, the reader is referred to Appendix A. This Appendix also describes in detail other data objects, including some which have been described above in more general terms.

The methodology of event-searching in the Version 8 EQ6 (in module **path.f** and various immediately supporting modules) has been significantly simplified from that employed in the Version 7 code. For example, one kind of event searched for is the point of reaction progress at which the aqueous solution becomes saturated with a mineral which is initially not present in the equilibrium system. In the Version 7 code, a Taylor's series representation of the primary calculational variables was expanded in a complex series of calculations in order to estimate the affinity or saturation index as a function of reaction progress. This search process has been simplified in Version 8 by creating a Taylor's series representation of the affinity functions themselves. This requires the use of additional memory, which is not a significant drawback. Besides reducing the amount of calculation required, the new method considerably simplifies both the mechanics of the calculations and also the data flow among the **path.f** module and the relevant supporting modules (e.g., **fpbnpp.f**, **search.f**, and **sfncaf.f**).

In EQ6, equilibrium calculations are carried out by module **eqcalc.f**. The reaction path calculation is directed by **path.f**. In the Version 7 code, the main program, **eq6.f**, called **eqcalc.f** directly to solve for the state of the system at the initial point of reaction progress; the main program then

EQ/6 Version 8 Letter Report, Sept. 29, 1994

called **path.f**, which in turn called **eqcalc.f** to do this at subsequent points. In Version 8, **eq6.f** simply calls **path.f** to deal with the calculation at all points, including the initial point.

3.0 Redox Disequilibrium in Reaction-Path Calculations (EQ6)

As discussed in the previous section, the Version 7 EQ3NR code permitted redox disequilibrium in static speciation-solubility models, but the corresponding EQ6 code did not permit such disequilibrium in reaction-path models. This was because mass balances in EQ3NR were defined in terms of basis species, and the auxiliary basis concept was implemented, allowing mass balances to be defined for basis species representing chemical elements in different oxidation states. In EQ6, the mass balances were defined in terms of chemical elements, and the basis set was limited to a strict basis set consisting of one species per chemical element, plus a redox species ($\text{O}_{2(g)}$). Thus, while EQ3NR could treat separate mass balances for Fe^{2+} and Fe^{3+} , EQ6 could only treat one for Fe in which both Fe^{2+} and Fe^{3+} were combined. The distribution of such species in the latter mass balance (and any other such mass balances) was then tied to the value of a single master redox parameter.

In Version 7, EQ3NR could be used to create a model of an aqueous solution, which could then be fed into EQ6 by incorporating the EQ3NR PICKUP file into an EQ6 input file. As noted in the previous section, EQ6 would then “force equilibrate” any redox reactions in a state of disequilibrium. This process actually began in EQ3NR. When it would write the PICKUP file, the mass balance totals written would be those for the chemical elements, not those for the set of active basis species actually used in the speciation-solubility calculation performed by this code.

In Version 8, EQ6 contains the same basic data structure for dealing with basis species and reactions as EQ3NR. EQ3NR now writes on its PICKUP file the mass balance totals for the set of active basis species used in the speciation-solubility calculation. A **jflag** value is now written for each such species on this file. A value of 0 instructs EQ6 to preserve a mass balance. A value of 30 (possible only for an auxiliary basis species) instructs EQ6 to combine this mass balance into others and remove the corresponding basis species from the active set (make it a dependent species). EQ3NR always writes a **jflag** value of 0 for a strict basis species and one of 30 for an auxiliary basis species. This makes Version 8 behave, by default, like Version 7. However, the user can change a **jflag** value of 30 on the PICKUP file (or in the EQ6 input file in which this PICKUP file has been included) to 0, thus instructing EQ6 to preserve the corresponding mass balance as a distinct relation and the corresponding basis species as a member of the active basis set.

What happens then in a given run is under the control of the user. Two cases are possible for any auxiliary basis species for which a mass balance is preserved. One is to simply leave the redox disequilibrium in a of frozen state. This allows a treatment of cases in which the relevant oxidation-reduction reaction kinetics are so slow as to be imperceptible. The strong metastability in certain settings of such species as $\text{N}_{2(aq)}$, NO_3^- , SO_4^{2-} , and dissolved organics can be treated in this manner. Alternatively, a kinetic rate law can be specified for the conversion or formation of the auxiliary basis species via a reaction linking it to other basis species. In this case, the auxiliary basis species is the “reactant” is specified on the EQ6 input file, and a corresponding **jcode** value

EQ/6 Version 8 Letter Report, Sept. 29, 1994

of 5 identifies the “reactant type” (here, a redox reaction). The input is otherwise analogous to that for a pure mineral reactant.

In modeling redox disequilibrium in EQ6, some difficulty may result from that fact that there is an asymmetry in the treatment of basis species in the strict set as opposed to the auxiliary set. This is normally only a problem when a chemical element is represented by more than two basis species. If for example one wishes to freeze the mass balance for SO_4^{2-} and allow other forms of sulfur to equilibrate with other members of a redox equilibrium “pool,” then SO_4^{2-} must be an auxiliary basis species. On most supporting data files, it is a strict basis species. This can be dealt with by specifying a “special” basis switch on the EQ6 input file, in which this species is switched with an auxiliary basis species such as HS^- . A special basis switch reverses the roles of a strict basis species and an auxiliary basis species. Once done, the code treats the affected species as though the data file had been written with the roles switched. That is, the new status of the species is reflected in the “data file” or “d” set of basis-dependent data arrays.

The described here is really a bit more general. Fundamentally, it addresses disequilibrium between an auxiliary basis species and a corresponding strict basis species. The reaction relating the two need not actually involve oxidation-reduction.

4.0 Thermodynamic Pressure Corrections

In version 7 of EQ3/6, calculations may be made over the temperature range 0-300C with the pressure constrained to a value of 1.013 bar (1 atm) up to 100C, and to the liquid/vapor equilibrium pressure for a pure water system up to 300C (where the pressure is about 85.8 bars). This is referred to as the original standard pressure curve (really a temperature-pressure curve). If it was necessary to perform calculations, say, at 50C and 500 bars, one had to either ignore the pressure effect on the supporting thermodynamic data, using values corresponding to a pressure on the original standard pressure curve, or create a special data with a different pressure curve (such as a constant pressure of 500 bars at all temperatures).

The effect of pressure on thermodynamic quantities such as equilibrium constants ($\log K$; the **xlksd** array for the “d” set of arrays, the **xlks** array for the regular set; **xlke** for the special “Eh” reaction) has long taken a back seat to the effect of temperature in the arena of geochemical modeling. Nevertheless, some studies (Kharaka et al., 1988; Aggarwal et al., 1992) have shown that pressure corrections are far from insignificant. Hydrostatic pressure increases by about 100 bars per 1000m. Thus, the pressure on deep groundwaters could be several hundred bars. However, pressure effects can be strong even for relatively low pressures if a gas is involved in a reaction.

The pressure dependence of an equilibrium constant is given by:

$$\frac{\partial}{\partial P} \log K = \frac{-\Delta V_r^0}{2.303RT} \quad (1)$$

Thus, a first-order correction to the equilibrium constant requires knowledge of the standard partial molar volume of reaction (ΔV_r^0 ; here P is the pressure, R the gas constant, and T the

EQ/6 Version 8 Letter Report, Sept. 29, 1994

absolute temperature). The standard partial molar volume of reaction, like the equilibrium constant, is a function of temperature and pressure and can be represented on a data file in like manner. In the case of EQ3/6, the values would then appear on the same T-P grid, and hence provide a means of making a first-order correction to pressures off the standard pressure curve. A higher-order pressure correction can be obtained by including similar data grids for the pressure derivatives of the standard partial molar volume of reaction (e.g., for $\partial \Delta V_r^o / \partial P$ and $\partial^2 \Delta V_r^o / \partial P^2$). A general form for the correction from standard curve pressure P_g to pressure P is:

$$\log K(P) = \log K(P_g) + \frac{-1}{2.303RT} \left(\Delta V_r^o (P - P_g) + \sum_{n=1}^i \left(\frac{\partial^n \Delta V_r^o}{\partial P^n} \right) \frac{(P - P_g)^{n+1}}{(n+1)!} \right) \quad (2)$$

This is the integrated form of a truncated Taylor's series. Here i is the maximum order for pressure derivatives of the standard partial molar volume of reaction.

In the software, standard partial molar volumes of reaction (standard partial molar volumes in the case of strict basis species) are kept in the **xvfsd** ("d" set of arrays) and **xvfs** arrays (regular set). The corresponding pressure derivatives are kept in the **dvfsd** and **dvfs** arrays. In the case of the special "Eh" reaction, the **xvfe** variable contains the standard partial molar volume of reaction, and the **dvfe** array contains the pressure derivatives.

This type of representation of thermodynamic pressure corrections has been incorporated into Version 8 of EQ3/6. A special data file containing the requisite additional data is required. At present, this is only available on the SHV data file, which is derived from SUPCRT92 (Johnson et al., 1992). To support future calculations of volume-based functions (such as the aqueous solution density or the volume occupied by a reacting system), corresponding standard partial molar volume data are included for strict basis species, which have no reaction and hence for which no standard partial molar volume of reaction is meaningful. These two types of functions, along with related data described below, are referred to collectively as "volume functions."

The other kind of thermodynamic data which require pressure corrections are various parameters used in calculating activity coefficients. Among these are the Debye-Hückel parameters A_γ (**adh**) and B_γ (**bdh**), and the B-dot parameter (**bdot**). Technically, the same applies to the Debye-Hückel parameter A_ϕ (**aphi**). However, this closely related to A_γ , and the pressure correction for A_γ can be used to correct A_ϕ . At the present time, pressure corrections have not been implemented for the Pitzer interaction parameters. At the present time, sufficient data to make the effort worthwhile appear to be lacking. Therefore, pressure corrections should presently be made only using the B-dot equation or the Davies equation as the aqueous solution activity coefficient model.

Volume analogs exist for all of the above parameters. However, these are normally defined for convenience in calculating volume-based properties, not pressure corrections. The pressure dependence of A_γ is given by:

$$\frac{\partial A_\gamma}{\partial P} = \frac{-A_V}{2(2.303)RT} \quad (3)$$

EQ/6 Version 8 Letter Report, Sept. 29, 1994

Here A_V (**adhv**) is the analog of the standard partial molar volume of reaction. Higher order correction is handled using its own set of pressure derivatives (the array **dadhv**) in like manner. The situation is simpler for the other activity coefficient parameters:

$$\frac{\partial B_\gamma}{\partial P} = B_V \quad (4)$$

$$\frac{\partial}{\partial P}(B\text{-dot}) = (B\text{-dot})_V \quad (5)$$

These are otherwise treated analogously. B_V is represented by the **bdhv** variable; the corresponding pressure derivatives are in the **dbdhv** array. $(B\text{-dot})_V$ is represented by the **bdotv** variable; the corresponding pressure derivatives are in the **dbdotv** array.

The presence of volume functions on an EQ3/6 data file is indicated by special flags embedded in the data file title. This allows upward compatibility of older EQ3/6 data files which lack the new, additional data. This flagging is illustrated by the following title taken from the X03 (experimental) SHV data file:

```
data0.shv
EQ3/6-V8-DATA0-SHV-X03
THERMODYNAMIC DATABASE
SUPCRT data, including enthalpy and volume functions
Prototype originally generated by gembochs/INGRES 6-oct-92
```

For a strict basis species:

1. There is no reaction and no log K grid
2. xhfs = the partial molar enthalpy
3. xvfs = the partial molar volume

For any other kind of species:

1. There is a reaction and a log K grid
2. xhfs = the partial molar enthalpy of reaction
3. xvfs = the partial molar volume of reaction

Standard data grid parameters:

```
NO. OF TEMPERATURE RANGES= 2
NO. OF POINTS IN RANGE 1= 4
NO. OF POINTS IN RANGE 2= 5
```

Data grid flags:

```
ENTHALPY = ON
dH/dP = OFF
d2H/dP2 = OFF
VOLUME = ON
dV/dP = OFF
d2V/dP2 = OFF
```

End of title

EQ/6 Version 8 Letter Report, Sept. 29, 1994

In this case, the string "VOLUME = ON" indicates the presence of volume functions. The strings "dV/dP = OFF" and "d2V/dP2 = OFF" indicate that first and second pressure derivatives of volume functions, respectively, are not present. Thus, only a first-order correction to the equilibrium constants and related data is possible with this data file. Version 8 is dimensioned for only first and second order pressure derivatives. This is set with the dimensioning parameter **ipcp**. When EQPT reads a data file containing volume functions, it sets the available order of the pressure derivatives in the variable **ipcv**. If volume functions are not present on the data file, this variable is set to -1. This variable is written on the DATA1 file generated by EQPT. EQ3NR and EQ6 then read this variable and use it to make the appropriate order of pressure correction (or no correction). The actual pressure corrections are made by two EQLIB modules: **pcorr.f** corrects the equilibrium constants, except that for the special "Eh" reaction; **pcorrm.f** corrects all other thermodynamic parameters requiring such correction.

Enthalpy functions (standard partial molar enthalpies of reaction, standard partial molar enthalpies for basis species, the Debye-Hückel parameters A_H and B_H , and $(B\text{-dot})_H$, and the pressure derivatives of these) have also been incorporated into Version 8 in a manner analogous to that for volume functions (e.g., **xhfsd** is analogous to **xvfsd**, **xhfs** to **xvfs**, **adhh** to **adhv**, etc.). In theory, these functions could be used to treat the temperature dependence of equilibrium constants, Debye-Hückel parameters, and the like. Values for parameters to be so corrected would then only be required at a single temperature. There are no plans to use enthalpy functions in this manner, however. The plan is to use them in future development to track heat balances. In the present SHV data file, there are no data for the pressure derivatives in the set of enthalpy functions.

In Version 8, the standard pressure grid defaults to the original EQ3/6 grid (two temperature ranges, four points in the first range, five in the second (one being common with the first range)). It is now possible to specify alternate grid parameters in the title, as indicated in the above example. If no such parameters are specified, the original standard grid parameters are used.

In EQ3NR, the user specifies the desired pressure (**press**) in bars on the input file, just after the specification of the desired temperature (C). In EQ6, the pressure is treated according to the setting of the **jpress** flag: 0 = standard curve pressure, 1 = pressure tracking. If the pressure tracking option is selected, the pressure is calculated as a linear function of reaction progress:

$$\text{press} = \text{pressb} + \text{pkt}(1) * \text{xi1} \quad (6)$$

or, in time mode (**iopt**(2) = 1):

$$\text{press} = \text{pressb} + \text{pkt}(1) * \text{time1} \quad (7)$$

Here **pressb** is the base pressure (pressure at $\xi = 0$ or $t = 0$), **xi1** is the current value of reaction progress (ξ), **time1** is the current value of time (sec), and **pkt**(1) is a pressure tracking coefficient. The **jpress** flag, the base pressure, and three pressure tracking coefficients (only the first of which

EQ/6 Version 8 Letter Report, Sept. 29, 1994

is currently used) are read from the input file. The current pressure is calculated by the EQ6 module **gpress.f**.

To provide a measure of protection in making pressure corrections, a pressure band half-width (**prehw**) is included in a DATA0 file supporting pressure corrections. This is also represented on the standard T-P grid. If the magnitude of the difference between the current pressure and the grid pressure exceeds the value of this function at the current temperature, a warning mechanism is activated, and a series of warning messages are printed.

Kinetic data (e.g., rate constants) also depend on pressure. At the present time, however, the understanding of kinetics applicable to geochemical processes is not sufficient to warrant the formal application of pressure corrections to kinetic data. There exists an associated theory for the form of such corrections for some of the rate law forms presently coded in EQ6 (e.g., transition-state theory); this is lacking for others.

5.0 Generic Ion-Exchange Model

Viani and Bruton (1992ab) developed a generic ion-exchange model in the context of a prototype branch version of EQ3/6 in the Version 7 line. The exchange substrate can be interpreted as a two-dimensional mineral surface, a three-dimensional solid solution, or a colloid. The model requires the user to define an exchange substrate, a set of exchange reactions, a choice of exchange reaction formalism (Vanselow or Gapon), and a set of exchange reactions and corresponding thermodynamic data. In the prototype, multiple exchangers could be defined. Also, an exchanger with multiple exchange sites could be simulated by defining multiple exchangers with the same name. For a discussion of the model equations and parameters, and results obtained in actual modeling, see Viani and Bruton (1992ab).

The prototype implementation of the ion exchange model worked by treating the exchange species as aqueous species. Template exchange species and reactions were required to be defined on the supporting thermodynamic data file. Each exchange species except a bare substrate species was required to be electrically neutral. The thermodynamic stability of the bare substrate species was set arbitrarily low (large log K for taking up an exchangeable ion), so that the concentration or mass of these species was negligible. The theory was that the only exchange species sensibly present would be electrically neutral, therefore not contribute to the ionic strength, and thereby not disturb the calculation of aqueous species activity coefficients. The exchange species (excepting the bare substrate species) themselves were expected to have unit activity coefficients because of their zero charge. Unfortunately, only an activity coefficient model which depends on the solution composition only through the ionic strength (which was the case with an old version of the B-dot model) can be used with such an implementation (of exchange species as aqueous species). Occasional numerical problems developed with the prototype implementation, owing the use of the bare substrate species as a basis species and it being required to have an essentially negligible concentration. Regardless, the prototype served well to develop the conceptual framework and to demonstrate the superiority of the Vanselow formalism over the Gapon formalism (Viani and Bruton, 1992ab). It also was used to show how the concentration dependence of empirical distribution coefficients could be explained.

EQ/6 Version 8 Letter Report, Sept. 29, 1994

In Version 8, the ion-exchange model has been implemented in an improved manner. It avoids both conflict with aqueous species activity coefficient models by defining a unique range in the list of phases for the exchanger phases (marked by the range variables **iern1** and **iern2**). Each exchange phase is conceptually divided into one or more sub-phases, each corresponding to an exchange site. In the list of species, species belonging to an exchanger phase appear in a contiguous range, and within that, in contiguous ranges corresponding to the exchanger site sub-phases. For technical reasons, a substrate site is also defined. Thus, in Version 8 an exchanger species is no longer treated as an aqueous species. Also, an exchange phase with multiple sites is not treated as a set of distinct exchange phases, each with one exchange site.

In Version 7, each exchange phase (each exchange site) must be electrically neutral (or very nearly so. This aspect has been retained in Version 8. However, it would be possible to relax this so that non-negligible charges could accrue to individual sites or even a whole exchange phase.

In Version 8, no exchange templates are required to be present on the supporting thermodynamic data file. All phases, species, and reactions are created by the software as needed. However, the data required of the user is slightly different. In addition to the exchanger phase name, the user must provide the number of exchange sites, names for the exchange sites, the electrical charge of each site when bare, and the number of moles of each site per mole of substrate. One must also provide the amount of substrate present in the modeled system (e.g., the molality in EQ3NR, the number of moles in EQ6). In contrast, in Version 7, the user provided instead the amount of each “exchanger” (exchange phase or exchange site of an exchange phase) and also gave this data in a different set of units (equivalents/per kg of solvent water in EQ3NR, equivalents in EQ6). The interpretation of the ion-exchange model input and the requisite set-up work is mostly handled by a new EQLIB module, **intexi.f**. Output tables are handled by a new EQLIB module called **prtge.f**.

6.0 Other New Things

In Version 8, alkalinity has returned to EQ3/6 in a somewhat new form. A previous treatment was removed in Version 7 development because of frequent problems in using it as a constraint on the basis species HCO_3^- . See Wolery (1992b, p. 21-27) for a discussion of these problems. In the new form, a total alkalinity in the odd units of mg/L equivalent CaCO_3 can be input to EQ3NR as a constraint on HCO_3^- (the units are odd, but this is the standard in groundwater chemistry reporting). To avoid convergence problems, this is required to be interpreted as a pure $\text{HCO}_3\text{-CO}_3\text{-OH}$ alkalinity; that is, no attempt is made to correct for “interferences” from species such as acetate, borate, phosphate, or metal hydroxy-complexes. The “complete” alkalinity is then calculated and compared with the input value, and the user is warned of interference effects. If interferences are large, the total alkalinity can not be used as a constraint on HCO_3^- . EQ6 now computes both pure $\text{HCO}_3\text{-CO}_3\text{-OH}$ and “complete” alkalinities as part of its output.

Alkalinity is computed by a new EQLIB module called **calk.f**. The coefficients for computing the pure $\text{HCO}_3\text{-CO}_3\text{-OH}$ alkalinity are contained in DATA statements in the EQLIB INCLUDE file **eqltf1.h**. Those for the complete alkalinity are contained in such statements in the analogous INCLUDE file **eqltf2.h**.

EQ/6 Version 8 Letter Report, Sept. 29, 1994

As was noted in the section on thermodynamic pressure corrections, the old standard temperature-pressure grid parameters may be replaced in Version 8 data files according to data embedded in the data file title. In particular, the number of temperature ranges can now be greater or lesser than the original standard of 2, and the number of points in any range can now be greater or lesser than in the original standard of 4 in the first temperature range and 5 in the second (the first point in any temperature range other than the first always overlaps with the last point in the preceding temperature range, so there were 8 points total in the original standard grid).

References

- Aggarwal, P. K., Gunter, W. D., and Kharaka, Y. K., 1990, Effect of pressure on aqueous equilibria: In Melchior, D.C., and Bassett, R.L., eds., *Chemical Modeling in Aqueous Systems II*, American Chemical Society Symposium Series, 416, American Chemical Society, Washington, DC, p. 87-101.
- ANSI, 1978, Programming Language FORTRAN: ANSI X3.9-1978, ISO 1539-1980 (E): American National Standards Institute, New York.
- ANSI, 1991, Fortran 90: ANSI X3.198-1991, ISO/IEC 1539-1991 (E): American National Standards Institute, New York.
- Daveler, S. A., and Wolery, T. J., 1992, EQPT, A Data File Preprocessor for the EQ3/6 Software Package: User's Guide and Related Documentation (Version 7.0): UCRL-MA-110662-PT-II, Lawrence Livermore National Laboratory, Livermore, California.
- Delany, J. M., Puigdomenech, I. P., and Wolery, T. J., 1986, Precipitation Kinetics Option for the EQ6 Geochemical Reaction Path Code: UCRL-53642, Lawrence Livermore National Laboratory, Livermore, California.
- Johnson, J. W., Oelkers, E. H., and Helgeson, H. C., 1992, SUPCRT92: A software package for calculating the standard molal thermodynamic properties of minerals, gases, aqueous species, and reactions from 1 to 5000 bars and 0° to 1000°C: Computers and Geosciences, v. 18, p. 899-947.
- Kharaka, Y. K., Gunter, W. D., Aggarwal, P. K., Perkins, E. H., and DeBraal, J. D., 1988, SOLMINEQ.88: A Computer Program for Geochemical Modeling of Water-Rock Interactions: United States Geological Survey Water Resources Investigation Report 88-4227.
- Viani, B. E., and Bruton, C. J., 1992a, Modeling ion exchange in clinoptilolite using the EQ3/6 geochemical modeling code: In Kharaka, Y. K., and Maest, A. S., eds., *Water-Rock Interaction, Volume 1, Low Temperature Environments*, A. A. Balkema, Rotterdam, p. 73-77.

EQ/6 Version 8 Letter Report, Sept. 29, 1994

Viani, B. E., and Bruton, C. J., 1992b, Modeling Fluid-Rock Interactions at Yucca Mountain, Nevada: A Progress Report: UCRL-ID-109921, Lawrence Livermore National Laboratory, Livermore, CA.

Wolery, T. J., 1992a, EQ3/6, A Software Package for Geochemical Modeling of Aqueous Systems: Package Overview and Installation Guide (Version 7.0): UCRL-MA-110662-PT-I, Lawrence Livermore National Laboratory, Livermore, California.

Wolery, T. J., 1992b, EQ3NR, A Computer Program for Geochemical Aqueous Speciation-Solubility Calculations: Theoretical Manual, User's Guide, and Related Documentation (Version 7.0): UCRL-MA-110662-PT-III, Lawrence Livermore National Laboratory, Livermore, California.

Wolery, T. J., and Daveler, S. A., 1992, EQ6, A Computer Program for Reaction-Path Modeling of Aqueous Geochemical Systems: Theoretical Manual, User's Guide, and Related Documentation (Version 7.0): UCRL-MA-110662-PT-IV, Lawrence Livermore National Laboratory, Livermore, California.

Appendix A. More on New Data Objects

A data object as the term is used here is a variable or array, or a group of related variables and arrays. It could refer to a formally defined group of variables or arrays, such as might be found together in a COMMON block or a NAMELIST statement. Constructing data objects involves mapping theoretical or real-world quantities into corresponding variables and arrays. There is no unique way to do this, and usually there are trade-offs to consider (e.g., involving clarity, functional considerations, efficiency of storage, effect on execution speed). We note in this section the data objects that have been replaced and what they have been replaced with. Unless otherwise noted, these changes pertain to both EQ3NR and EQ6.

A.1. Review of Old Data Objects

Chemical elements are currently represented by names (standard chemical symbols) in the **uelem** array. Each standard symbol only requires 1 or 2 characters. However, the character length of each name is set to 8 in the software (the minimum for a character variable array under the applicable coding standards). Each element is referenced in the software by an index (usually referenced by the variable **nc**) which corresponds to its position in this array. This representation continues in Version 8. However, significant changes have been made in the related arrays dealing with chemical species, phases, chemical compositions, and chemical reactions.

In Version 7, chemical species are represented by names of up to 24 characters. The names of aqueous species are in the **uspec** array. The index of each aqueous species (usually referenced by the variable **ns**) corresponds to its position in this array. Similarly, the names of pure minerals are in the **umin** array (typical index variable **nm**) and the names of gas species are in the **ugas** array

EQ/6 Version 8 Letter Report, Sept. 29, 1994

(typical index variable **ng**). The names of solid solution minerals are in the **usolx** array (typical index variable **nx**). However, solid solutions are phases, not species. The components of solid solutions are species. Currently (and in the foreseeable future), these are end-members corresponding to pure minerals. The variable **ik** is usually used to reference a solid solution component. The **nend** array currently acts as a pointer to the corresponding pure mineral index (**nm = nend(ik,nx)**).

In Version 7, phases are not explicitly represented. Each solid solution and each pure mineral is a phase, as is the aqueous solution implied by the aqueous species and the gas phase implied by the gas species. There is no explicit gas phase in either Version 7 or Version 8, though such a phase is planned for Version 9.

The chemical composition of a chemical species is fixed. That of a phase is variable, and it is generally described by the concentrations (molalities or mole fractions of the contained species. In Version 7, the composition of aqueous species is stored in the **cess** array. Here **cess(nc,ns)** is the compositional coefficient of the **nc**-th element for the **ns**-th aqueous species. The first dimension of this array spans the total number of elements, and this must be chosen to accommodate all the elements on any data file employed (currently closely to 50). On the other hand, the average aqueous species is composed of no more than 3 elements. Thus, this array is rather sparse. The chemical composition of the pure minerals is stored in an analogous **cemn** array. Most minerals consist of fewer than 5 elements, so this array is sparse as well. The composition of gas species is in the analogous **cegs** array, which is also pretty sparse. The composition of solid solution components is represented by the compositions of the corresponding pure minerals. The sparsity of the **cess**, **cemn**, and **cegs** arrays was not a major factor when the software was originally designed, as the first data file represented only about 16 chemical elements. Subsequent development of the data files has aggravated the problem.

Chemical reactions are represented by an analogous set of arrays. Those for aqueous species are stored in the **cdrs** array. Here **cdrs(nse,nrs)** is the reaction coefficient of the **nse**-th aqueous species (a basis species) and **nrs** is the reaction index corresponding to the **ns**-th aqueous species (**nrs = ns - nsb**, where **nsb** is the number of strict basis species). The first **nsq** aqueous species are defined as the set of basis species (**nsq** may be greater than or equal to **nsb**), which are treated as a set of generalized "building blocks" for writing chemical reactions. Each reaction is written so that the associated species (here the **ns**-th aqueous species) is destroyed by the reaction. This associated species is the only non-basis species which may appear in the reaction. Its coefficient is stored as **cdrs(nsq1,nrs)**, where **nsq1 = nsq + 1**. Chemical reactions of pure minerals are represented by a nearly analogous **cdrm** array. Here **cdrm(nse,nm)** is the reaction coefficient for the **nse**-th basis species in the reaction for the **nm**-th mineral. The reaction coefficients for gas species are stored in a **cdrg** array analogous to **cdrm**. Because the first dimension of the **cdrs**, **cdrm**, and **cdrg** arrays must span the number of basis species plus one, these arrays have also become rather sparse. The number of basis species on some data files is approaching 200, while fewer than 7 or 8 appear in a typical reaction. This extent of sparseness is again a result of data base development.

EQ/6 Version 8 Letter Report, Sept. 29, 1994

The storage of equilibrium constants is similarly split into different arrays for different species types. Those for aqueous species are in the **xlks** array (**xlks(nrs)** is the equilibrium constant for the **nrs**-th aqueous reaction). Those for pure minerals are in the **xlkm** array, those for gas species in the **xlkg** array. The equilibrium constants themselves are computed as functions of temperature from interpolating polynomial coefficients. The coefficients for computing the **xlks** array are stored in the **ars** array, those for **xlkm** in the **amn** array, and those for **xlkg** in the **ags** array.

The set of basis species itself is fixed at the first **nsq** aqueous species. In order to make a basis switch (exchange the roles of a basis and non-basis species), it is necessary to exchange their positions among the aqueous species. This can make things complicated. This method of defining the basis set also has the disadvantage that other kinds of species can not be switched into the basis set (numerical reasons make this possibility desirable). The basis set consists of a minimal part called the strict basis set, which is equal in number to the number of chemical elements (**nct**) plus 1 (**nsb** = **nct** + 1). On the data file, each of the first **nct** basis species represents a chemical element (e.g., Fe^{2+} for Fe). The **nsb**-th species is a fictive O_2 species associated with the redox variable (the oxygen fugacity). Each strict basis species, being a minimal level "building block," has no associated chemical reaction. This accounts for the treatment of the second dimension of the **cdrs** array.

The set of basis species may be extended to the case of **nsq** > **nsb** by including an auxiliary basis set. Auxiliary basis species generally correspond to chemical elements in oxidation states other than those represented by the species in the strict basis set. On the data file, each auxiliary basis set has a reaction written in terms of itself as the associated species and in terms of strict basis species and possibly other auxiliary basis species of lower index number. An auxiliary basis species may be treated as either a basis species, in which case the associated reaction is not required to satisfy equilibrium, or as a non-basis species, in which case it must. The choice is made by the code user. Setting the corresponding element of the **jflag** array to a value less than 30 causes an auxiliary basis species to be treated as a basis species. However, in the present version of EQ3/6, this can be done only with the EQ3NR code. The current version of EQ6 forces the treatment of auxiliary basis species as non-basis species. Here **jflag(ns)** set in the code to 30 for all such species. The coding is not present to deal with the situation otherwise.

In EQ3NR, mass balance relations are computed using stoichiometric coefficients computed from the reaction coefficients in the **cdrs** array (because EQ3NR only considers mass balances in the aqueous phase, coefficients based on the **cdrm** and **cdrg** arrays are not required). These stoichiometric coefficients are themselves stored in the **csts** array. The mass balance relations themselves are in terms of total molalities of basis species. In EQ6, mass balance relations are computed using the **cess** and **cemn** arrays (the **cegs** array is not required as the code does not consider the presence of an explicit gas phase). These mass balances are in terms of the total number of moles of the chemical elements.

The problems of the data structure in Version 7 may be summarized as follows. The existing storage of chemical composition coefficients and reaction coefficients is now quite inefficient. Also, the set of basis species is restricted to only aqueous species. There is a complicating offset in the second dimension of the **cdrs** array. There is no compelling reason to split the storage of

EQ/6 Version 8 Letter Report, Sept. 29, 1994

basic data of the same type into different arrays on the basis of species types (e.g., **cdrs/cdrm/cdrg/xlks/xlkm/xlkg**). There is a major difference between the data structure in EQ3NR and that in EQ6. That in the latter does not permit consideration of redox disequilibrium.

A.2. New Data Objects

The new data objects described here are used in Version 8, in both the EQ3NR and EQ6 codes. The new structure contains an explicit list of phases, represented by the 24-character names in the new **uphase** array. This subsumes the old **usolx** array, which applied only to solid solutions. The index of a phase (generally referenced by the variable **np**) is its position in this array. All species, of whatever type, are grouped in a common list, represented by 48 character names in the **uspec** array (thus combining the old **uspec**, **umin**, and **ugas** arrays). The first 24 characters consist of the species name proper, the second 24 characters the name of the corresponding phase. This last part is designed as a check, and to insure uniqueness of the complete species name.

The **uphase** array is organized so that phases of a given type (e.g., pure minerals, solid solutions) are contiguous. Range variables have been created to mark the start and end of the corresponding ranges. The pure minerals range from **imrn1** to **imrn2**, the pure liquids from **ilrn1** to **ilrn2**, and the solid solutions from **ixrn1** to **ixrn2**. The pure liquids represent a new category. The index of the aqueous solution is denoted by **iaqsln**, that of the gas phase by **igas** (but even Version 8 does not actually treat a gas phase).

The **uspec** array is similarly organized, but at two levels. First, all the species belonging to a given phase are contiguous. Following the organization of the **uphase** array, all such groups of species belonging to a given phase type are also contiguous. A pointer array called **ncmpr** has been created to mark the first and last species belonging to each phase. Thus, for the **np**-th phase, the index of the first constituent species is given by **ncmpr(1,np)**, that of the last by **ncmpr(2,np)**. The first and last species are the same for a one-species phase (a pure phase), which is the case for the pure minerals and pure liquids. A set of special range pointers is employed with the **uspec** array. The first and last species in the aqueous solution phase are represented by **narn1** and **narn2**, respectively. Solvent water is the first species in this range. Thus, its index is **narn1**. The first and last species in the gas phase are marked by **ngrn1** and **ngrn2**. The first and last species in the range of pure minerals are marked by **nmrn1** and **nmrn2**, those of the pure liquids by **nlrn1** and **nlrn2**, and those of the solid solutions by **nxrn1** and **nxrn2**. Note that the **nmrn1/nmrn2**, **nlrn1/nlrn2**, and **nxrn1/nxrn2** pairs cover phase type groups, not just a single phase. Also, these are simple variables, not arrays. Thus, to access the range of species for a solid solution phase, it is necessary to utilize the **ncmpr** array.

In the old data structure, **nst** was the number of aqueous species. In the new data structure, **nst** is the number of all species, regardless of type, and **nat** is the number of aqueous species. Also, **npt** is the number of phases.

The **ncmpr** array subsumes the function of the old **nend** array, which pertained only to solid solutions. However, there is a key difference. In the old data structure, **nend** gave the index of the pure mineral corresponding to the solid solution end-member. The pure mineral and that end-

EQ/6 Version 8 Letter Report, Sept. 29, 1994

member were not treated as fully distinct species in the sense that some properties belonging to the end-member were simply obtained by getting those of the pure mineral whenever those properties were needed. In the new data structure, a solid solution end-member is created as a completely distinct entity by cloning the pure mineral. This allows the possibility of modifying some of the properties of the end-member, such as its thermodynamic stability, while leaving that of the pure mineral untouched.

In EQ6, the fixed fugacity option requires the creation by the code of one fictive pure mineral per gas whose fugacity is to be fixed. In the Version 7 EQ6, this was done by adding to the pure mineral list, as represented by the **umin** array. If this were implemented analogously in version 8, the range of phases between **imrn1** and **imrn2** would be extended (the value of **imrn2** would be increased; so of course would **nmrn2**). However, for convenience, and because these phases are special, they are instead placed in their own ranges, marked by **ifrn1** and **ifrn2** in the **uphase** array, and **nfrn1** and **nfrn2** in the **uspec** array.

Chemical compositions are now stored in a set of **cess/ness/nessr** arrays. The composition coefficients themselves are stored in **cess**, which subsumes the **cess**, **cemn**, and **cegs** arrays of the old data structure. Only non-zero values are stored. Thus, the indices of the corresponding chemical elements do not match their positions in this array. Instead, these indices are stored in the **ness** array. The **cess** and **ness** arrays are one-dimensional. Thus **cess(n)** is the coefficient for the element whose index is **ness(n)**. However, the corresponding species must be found by using the **nessr** pointer array. The composition coefficients for the **ns**-th species begin at **n = nessr(1,ns)** and terminate at **n = nessr(2,ns)**. To find the coefficient for the **nc**-th element, it is necessary to search this range to find **n** such that **ness(n) = nc**. The coefficient is then **cess(n)**. If there is no match, the coefficient is zero.

The above structure is highly efficient in terms of storage. However, if one wanted to compute the total number of moles for the **nc**-th element, the structure would appear to be computationally less efficient than other possibilities in that given element **nc** and species **ns**, one must do some searching to find the corresponding coefficient. One might write code like the following:

```
mtot = 0.
do 100 ns = 1,nst
  do n = nessr(1,ns),nessr(2,ns)
    if (ness(n).eq. nc) then
      mtot = mtot + cess(n)*moss(ns)
    go to 100
  endif
enddo
100 end
mte(nc) = mtot
```

However, mass balances in terms of chemical elements are of limited interest in Version 8, as most of the calculations in EQ3NR and EQ6 involving mass balances are in terms of basis species. Mass balances in terms of chemical elements are used only in calculations for informational output. The following coding is more efficient, because the mass balance for each element is computed simultaneously:

EQ/6 Version 8 Letter Report, Sept. 29, 1994

```

do nc = 1,nct
  mte(nc) = 0.
enddo

c
do ns = 1,nst
  do n = nssr(1,ns),nssr(2,ns)
    nc = nss(n)
    mte(nc) = mte(nc) + css(n)*moss(ns)
  enddo
enddo

```

Reaction coefficients are stored in a set of analogous **cdrs**/**ndrs**/**ndrsr** arrays. The reaction coefficients themselves are stored in **cdrs**, which subsumes the **cdrs**, **cdrm**, and **cdrg** arrays of the old data structure. Again, only non-zero values are stored. Here, **cdrs(n)** is the coefficient for the species (either the associated species or a basis species) whose index is **ndrs(n)**. However, the associated species must be found by using the **ndrsr** pointer array. The coefficients for the reaction associated with the **ns**-th species begin at **n = ndrsr(1,ns)** and terminate at **n = ndrsr(2,ns)**. The coefficients are arranged so that the first coefficient for a species is that of the associated species (e.g., **ndrs(ndrsr(1,ns)) = ns**).

To simplify matters, every species now has an associated reaction. Thus, **ns** can be treated as a reaction index as well as a species index. In the case of a strict basis species, this reaction is formally treated as a null reaction or identity reaction exemplified for the case of the sodium ion by $\text{Na}^+ = \text{Na}^+$. Technically, such a reaction has two coefficients (-1 for the sodium ion on the left hand side, +1 for that on the right). However, to simply matters and save storage, a null reaction is marked by the condition that **ndrsr(1,ns) = ndrsr(2,ns)**. This is adequate because no real reaction has only one coefficient. The contents of **cdrs(ndrsr(1,ns))** and **ndrs(ndrsr(1,ns))** are immaterial and are set to zero.

In the old data structure, the set of basis species was defined as a subset of the aqueous species (the first **nsq** such species). In the new data structure, the set of basis species is defined by a list represented by 48-character names in the **ubasp** array. The basis index of a basis species, generally represented by the variable **nb**, is now determined by the position of its name in this array. The species indices of the basis species are stored in a pointer array called **nbasp**. Thus, the species index is obtained by a statement of the form **ns = nbasp(nb)**. It follows that **ubasp(nb) = uspec(nbasp(nb))**. The total number of basis species are stored in the variable **nbt**, which replaces **nsq**.

In the old data structure, the **jflag** array pertained only to basis species. In the new data structure, it pertains to all species. The condition **jflag(ns) = 30** marks the condition that the reaction for the **ns**-th species is required to satisfy thermodynamic equilibrium.

The equilibrium constants corresponding to the reactions are stored in the **xlks** array, which subsumes the **xlks**, **xlkm**, and **xlkg** arrays of the old data structure. The polynomial coefficients for computing equilibrium constants as a function of temperature are stored in the **axlks** array, which subsumes the old **ars**, **amn**, and **ags** arrays.

EQ/6 Version 8 Letter Report, Sept. 29, 1994

Mass balance coefficients are stored in a set of **csts/nsts/nstsr** arrays analogous to the **cess/ness/nessr** arrays. These are calculated from the reaction coefficients stored in the **cdrs/ndrs/ndrsr** arrays. The mass balance coefficients themselves are stored in **csts**, which subsumes the **csts** array of the old data structure in the EQ3NR code. Again, only non-zero values are stored. Here, **csts(n)** is the mass balance coefficient for the basis species whose index is **nsts(n)**. However, the species whose contribution to the mass balance of this basis species is determined by this coefficient must be found by using the **nsts** pointer array. The coefficients for the **ns**-th species begin at **n = nsts(1,ns)** and terminate at **n = nsts(2,ns)**. Thus, to calculate the total number of moles of the basis species, one utilizes coding such as:

```
      do nb = 1,nbt
        mtb(nb) = 0.
      enddo
c
      do ns = 1,nst
        do n = nstsr(1,ns),nstsr(2,ns)
          nb = nsts(n)
          mtb(nb) = mtb(nb) + csts(n)*moss(ns)
        enddo
      enddo
```

In the Version 7 EQ3NR or EQ6, the code first reads a problem from the input file. It then reads the supporting data file (**data1**). In doing so, it does not load all of the data. Rather, it excludes certain data that are not necessary to solve the current problem. For example, the data for the aluminum-bearing minerals are not loaded if there is no aluminum in the current problem. The code then works on this problem. When it finishes, it loops back, reads another problem from the input file, and quits if none is found. If another problem is present, it re-reads the supporting data file, works on the new problem, and so forth.

In the new version, each code first reads the data file, which it does only once. The data are stored in what is called the “a” set of data arrays. This letter serves as a suffix to identify versions of the previously described variables and arrays which hold an exact and complete image of the data file contents. For example, the array of phase names read from the data file is **uphasa** (note the “e” was dropped to keep the name length at 6 characters), the array of species names, **uspeca**, and the array of basis species names, **ubaspa**. The set of variables and arrays for which there are corresponding entities in the “a” set is large and generally corresponds to all of those previously discussed in this section. However, equilibrium constants are not read from the data file, so there is no **xlksa** array corresponding to the **xlks** array. The corresponding polynomial coefficients are read from the data file, so there is an **axlksa** array corresponding to the **axlks** array. The **csts/nsts/nstsr** arrays are computed from reaction coefficients, hence have no corresponding arrays in the “a” set. The **jflag** array is not read from the data file, hence there is also no array corresponding to it in that set.

In the Version 8 EQ3NR or EQ6, a problem is read from the input file. The code then creates a “d” set of data by compressing the “a” set to exclude chemical elements, phases, species, and associated data which are not pertinent to the current problem. The “d” set is intended to be a working set of data, which represents the reactions and associated properties as they are written

EQ/6 Version 8 Letter Report, Sept. 29, 1994

on the data file. In particular, the “d” set basis species are preserved in the **ubaspd** array and the corresponding species indices in the **nbaspd** array. The reactions and associated thermodynamic data are preserved in the **cdrsd/ndrsd/ndrsdr/axlksd** arrays. Equilibrium constants corresponding to the “d” set are calculated and stored in the **xlksd** array. All other pertinent arrays and variables from the “a” set map directly during compression to their corresponding “regular” arrays and have no corresponding special “d” set entities. For example, the **uphasa** array is compressed to produce the **uphase** array.

The “d” set of data (**ubaspd/nbaspd/cdrsd/ndrsd/ndrsdr/axlksd/xlksd**) is not necessarily an exact copy of what was on the data file, apart from compression. First, it may be affected by special basis switching. This involves switching of species within the basis set. For example, an auxiliary basis species may be switched with a strict basis species, or one auxiliary basis species with another. Such switches are read from the input file for the current problem. Second, it may be affected by the elimination of auxiliary basis species from the active basis set, depending on **jflag** settings also read from the input file. Third, the associated thermodynamic data may be altered by **nxmod** options also read from the input file (See Chapter 6 of the EQ3NR User’s Guide, Wolery, 1992b). Lastly, the code’s functionality has been extended a bit by allowing any non-basis species on the data file (“a” set) to be made an auxiliary basis species (in the “d” set) during a code run. This is done if there is an entry for such a species in the basis species part of the input file. Such a basis species has no dependent species.

After the final “d” set modifications, the **ubaspd/nbaspd/cdrsd/ndrsd/ndrsdr/axlksd** data are copied to the corresponding “normal” arrays (**ubasp/nbasp/cdrs/ndrs/ndrsr/axlks**) and the “normal” set of equilibrium constants in the **xlks** array are calculated from the coefficients in the **axlks** array.

Version 8 uses a dual-basis concept in which the “d” set of basis species, reactions, and reaction data are retained throughout the calculations while also employing the regular set. Initially, the two sets are identical. However, the “regular” set may be changed in the subsequent course of calculations by ordinary basis switching. This may be caused by directives read from the input file for the current problem, or by the code itself as part of procedures to optimize the numerics of the code calculations.

The purpose of utilizing this dual basis is as follows. First, unrestricted basis switching has the potential of producing corresponding mass balance total quantities (molalities or numbers of moles) with no physical significance. This has the potential to create problems with user misunderstandings of the nature of related code inputs and outputs (see Chapter 5 of the EQ3NR User’s Guide, Wolery, 1992b). Therefore, it is highly desirable to keep all such quantities defined in terms of those species which were basis species on the data file. There is no reason not to handle all internal mass balance relations in the same way. This is done in Version 8. If this were not done, an extra set of arrays corresponding to the **csts/nsts/nstsr** arrays would have to be utilized (one fixed and matching the **cdrsd/ndrsd/ndrsdr** arrays, the other matching the **cdrs/ndrs/ndrsr** arrays and subject to change by ordinary basis switching). Thus, the **csts/nsts/nstsr** arrays are calculated from the reactions as represented in the **cdrsd/ndrsd/ndrsdr** arrays, and remain invariant through the calculations for the current problem.

EQ/6 Version 8 Letter Report, Sept. 29, 1994

Handling mass balances in the above manner only requires the **ubaspd/nbaspd/csts/nsts/nstsr** arrays. The **cdrsd/ndrsd/ndrsrd** arrays are no longer needed for this purpose once the **csts/nsts/nstsr** arrays have been computed. This need could be met by just using the “regular” **cdrs/ndrs/ndrsr** arrays in their place, as long as the **csts/nsts/nstsr** arrays are computed prior to any ordinary basis switching. However, the **cdrsd/ndrsd/ndrsrd** arrays in conjunction with the **xlksd** and **axlksd** arrays continue to be useful throughout the calculations pertinent to a given problem by providing a standard and constant basis for the computed reaction properties that EQ3NR and EQ6 produce as part of the code output. Reaction properties depend on how reactions are written, and ordinary basis switching involves rewriting reactions. In writing code output, it is generally quite inconvenient to write out the reactions corresponding to reaction properties. The reaction properties are ordinarily labelled only by the name of the species associated with the reaction. Thus, the separate existence of the **cdrsd/ndrsd/ndrsrd/xlksd/axlksd** arrays is justified.

Technical Information Department • Lawrence Livermore National Laboratory
University of California • Livermore, California 94551