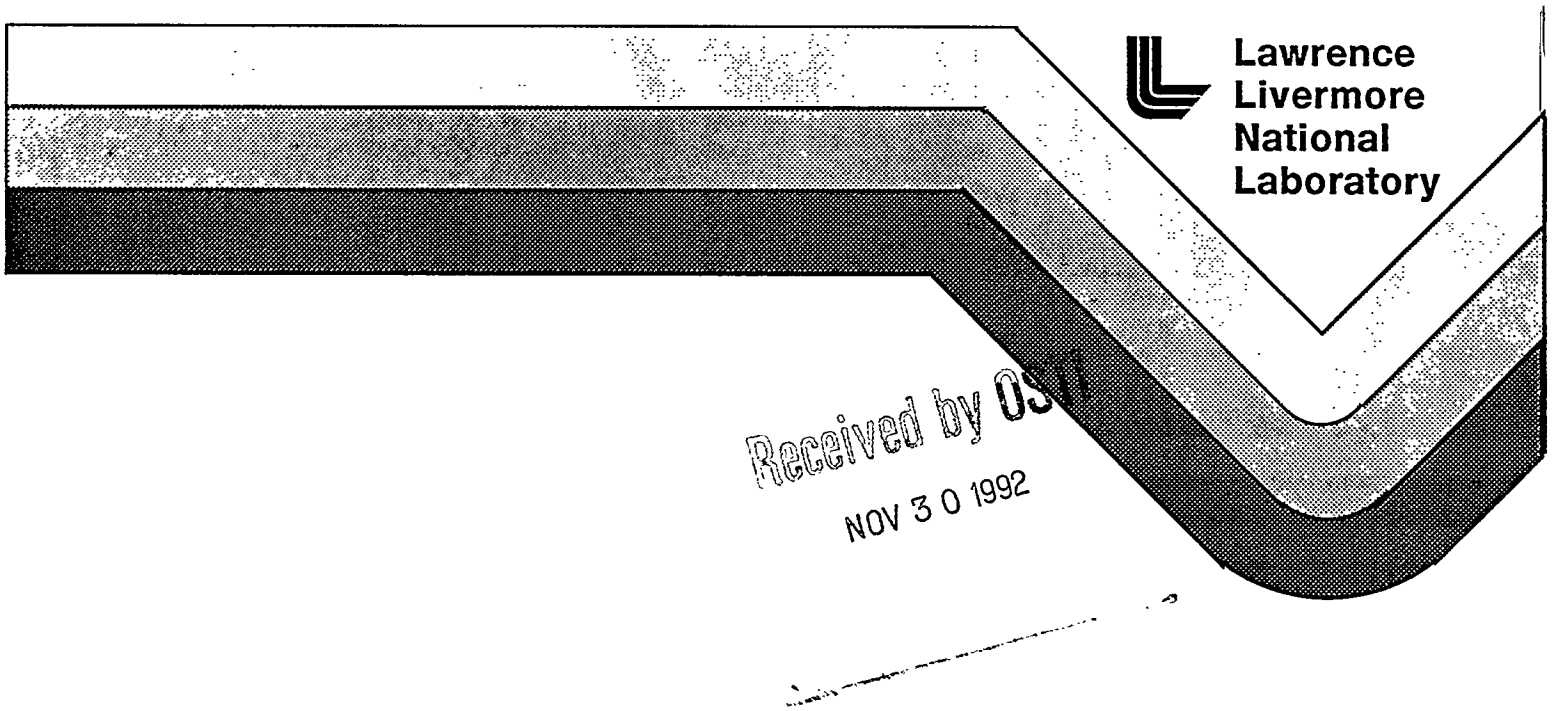
UCRL-MA-110662 PT I

EQ3/6, A Software Package for Geochemical Modeling of Aqueous Systems: Package Overview and Installation Guide (Version 7.0)

Thomas J. Wolery

September 14, 1992



DISCLAIMER

Prepared by Yucca Mountain Site Characterization Project (YMP) participants as part of the Civilian Radioactive Waste Management Program. The Yucca Mountain Site Characterization Project is managed by the Yucca Mountain Site Characterization Project Office of the U.S. Department of Energy, Las Vegas, Nevada.

Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

UCRL-MA--110662-Pt.1

DE93 003359

EQ3/6, A Software Package for Geochemical Modeling of Aqueous Systems: Package Overview and Installation Guide (Version 7.0)

Thomas J. Wolery

Preface

This report is the first in a set of documenting version 7.0 (version 3245.1090 under the old numbering system) of the EQ3/6 software package. This set includes:

I. The EQ3/6 Package Overview and Installation Guide (this report).

II. The EQPT User's Guide (Daveler and Wolery, 1992).

III. The EQ3NR Theoretical Manual and User's Guide (Wolery, 1992).

IV. The EQ6 Theoretical Manual and User's Guide (Wolery and Daveler, 1992).

EQ3NR is the speciation-solubility code in the EQ3/6 package. EQ6 is a reaction path code and hence deals with the evolution of a water/rock system as reaction progress or time advances. EQPT is the EQ3/6 data file preprocessor.

The development of EQ3/6 has been supported by a number of programs concerned with geologic disposal of high level nuclear waste, including the Office of Nuclear Waste Isolation, the Salt Repository Project Office, the Waste Isolation Pilot Plant (through Sandia National Laboratory), the Nevada Nuclear Waste Storage Investigations, and the Yucca Mountain Site Characterization Project. Documentation for the package is aimed at satisfying the requirements of the U.S. Nuclear Regulatory Commission for software used for this purpose (Silling, 1983).

The Lawrence Livermore National Laboratory has not certified that EQ3/6 constitutes approved code for the conduct of quality affecting work for the Yucca Mountain Project.

No source codes or data files are reproduced in this report, nor are any computer media containing such items a part of this report or any of the other reports documenting this version of EQ3/6. The software itself must be obtained as described below.

The examples presented in this series of reports correspond to version 7.0 of the software and the R10 set of supporting thermodynamic data files. As of the date of publication of this report, the most recent version of the software is version 7.1 (containing bug fixes, but no enhancements), and the most recent set of data files is R16.

Agencies of the United States Government and their contractors may obtain copies of the software and its documentation from:

Energy Science and Technology Software Center
P. O. Box 1020
Oak Ridge, TN 37831-1020

Telephone: (615) 576-2606

Requests to obtain the software under a licensing agreement should be addressed to:

Technology Transfer Initiatives Program, L-795
Attn: Diana (Cookie) West
Lawrence Livermore National Laboratory
P.O. Box 808
Livermore, CA 94550

Telephone: (510) 423-7678
Fax: (510) 422-6416
Secretary: (510) 422-6416

Comments and questions concerning EQ3/6 exclusive of the thermodynamic data base should be addressed to the code custodian:

Thomas J. Wolery, L-219
Lawrence Livermore National Laboratory
P.O. Box 808
Livermore, CA 94550

E-mail: wolery1@llnl.gov
Telephone: (510) 422-5789
Fax: (510) 422-0208
Secretary: (510) 423-2970

Comments and questions which concern the EQ3/6 thermodynamic data base should be addressed to the data base custodian:

James W. Johnson, L-219
Lawrence Livermore National Laboratory
P.O. Box 808
Livermore, CA 94550

E-mail: johnson@s05.es.llnl.gov
Telephone: (510) 423-7352
Fax: (510) 422-0208
Secretary: (510) 423-2970

Contents

EQ3/6, A Software Package for Geochemical Modeling of Aqueous Systems: Package Overview and Installation Guide (Version 7.0)

Abstract

EQ3/6 is a software package for geochemical modeling of aqueous systems. This report describes version 7.0. The major components of the package include: EQ3NR, a speciation-solubility code; EQ6, a reaction path code which models water/rock interaction or fluid mixing in either a pure reaction progress mode or a time mode; EQPT, a data file preprocessor; EQLIB, a supporting software library; and five supporting thermodynamic data files. The software deals with the concepts of thermodynamic equilibrium, thermodynamic disequilibrium, and reaction kinetics. The five supporting data files contain both standard state and activity coefficient-related data. Three support the use of the Davies or B-dot equations for the activity coefficients; the other two support the use of Pitzer's equations. The temperature range of the thermodynamic data on the data files varies from 25°C only to 0-300°C. EQPT takes a formatted data file (a **data0** file) and writes an unformatted near-equivalent called a **data1** file, which is actually the form read by EQ3NR and EQ6. EQ3NR is useful for analyzing groundwater chemistry data, calculating solubility limits, and determining whether certain reactions are in states of partial equilibrium or disequilibrium. It is also required to initialize an EQ6 calculation. EQ6 models the consequences of reacting an aqueous solution with a set of reactants which react irreversibly. It can also model fluid mixing and the consequences of changes in temperature. This code operates both in a pure reaction progress frame and in a time frame. In a time frame calculation, the user specifies rate laws for the progress of the irreversible reactions. Otherwise, only relative rates are specified. EQ3NR and EQ6 use a hybrid Newton-Raphson technique to make thermodynamic calculations. This is supported by a set of algorithms which create and optimize starting values. EQ6 uses an ODE integration algorithm to solve rate equations in time mode. The codes in the EQ3/6 package are written in FORTRAN 77 and have been developed to run under the UNIX operating system on computers ranging from workstations to supercomputers. This report describes in general terms how to install the software, with specific instructions on how to install it on systems running UNIX.

1. Package Overview

1.1. Introduction

The EQ3/6 software package originated in the mid-1970s (Wolery, 1978). It was originally developed at Northwestern University to model seawater-basalt interactions in mid-ocean ridge hydrothermal systems. It was brought to Lawrence Livermore Laboratory in 1978 by the original author and described in an early report (Wolery, 1979). Since then, there has been a succession of new versions, each representing a new set of improvements. The present report is one of four documents describing version 7.0 (3245.1090 under the old numbering system; see Appendix E). The other three documents are the EQPT User's Guide (Daveler and Wolery, 1992), the EQ3NR Theoretical Manual and User's Guide (Wolery, 1992), and the EQ6 Theoretical Manual and User's Guide (Wolery and Daveler, 1992).

EQ3NR and the other codes in the EQ3/6 software package are written in FORTRAN 77 and have been developed to run under UNIX operating systems on computers ranging from workstations to supercomputers, including Sun SPARCstations, Alliants (CONCENTRIX operating system), and Crays (UNICOS operating system). They are fairly readily portable to VAX computers running ULTRIX (a UNIX operating system) and VMS (a non-UNIX operating system). They may be portable to 386 and 486 PCs (see Chapter 3). Platforms used at LLNL include Sun SPARCstations and an Alliant FX/80.

The purpose of the present report is to provide an overview of EQ3/6, version 7.0, to describe the process of installing the software, and to document some aspects of the software package that pertain to more than one of the individual codes. The present chapter provides the overview. The EQ3/6 export package is described in Chapter 2, and instructions for installing the software are given in Chapter 3. The installation instructions are given in general terms, illustrated by specific directions for installing the software on computers running the UNIX operating system. The programming philosophy and FORTRAN coding standards employed in writing the software package are presented in Appendix A. The modules in the EQLIB library which supports the EQPT, EQ3NR, and EQ6 codes are listed and described in Appendix B. A list of error messages generated by EQLIB modules, along with related notes, is given in Appendix C. Notes pertaining to known bugs and such in the codes and supporting data files as of the date of this report are given in Appendix D. A list of the various versions of the EQ3/6 software package is given in Appendix E.

The basic purpose of EQ3/6 is to make two kinds of calculations pertaining to aqueous solutions and aqueous systems. The first kind is called a *speciation-solubility* calculation (Jenne, 1981). This function is provided in EQ3/6 by the EQ3NR code (Wolery, 1983, 1992). The purpose of such a calculation is to describe the chemical and thermodynamic state of the solution using as input analytical data and/or theoretical assumptions, such as states of partial equilibrium with specified minerals. Such a calculation utilizes the concepts of thermodynamic equilibrium, activity coefficients, and ion pairing and complexation, along with the corresponding mathematical descriptions and model parameters. It is possible to introduce some elements of thermodynamic disequilibrium in such a calculation, so it is not necessarily a pure equilibrium calculation. The price that must be paid is the requirement of additional analytical data. The output of a speciation-solubility calculation consists of two parts. One of these is the *speciation*: the concentration and thermodynamic activity of each individual chemical species. The other, the *solubility* part, consists of the saturation indices ($SI = log\ Q/K$, where Q is the activity product and K the equilibrium constant) for the various reactions not in a state of partial equilibrium. Most of these reactions pertain to mineral dissolution, and one often speaks of the saturation index of a given mineral. Many aspects of such calculations can be inverted, so that an output and an input to the problem can change roles. For details, see the EQ3NR Theoretical Manual and User's Guide (Wolery, 1992).

The second kind of calculation is called a *reaction path* calculation. This function is provided in EQ3/6 by the EQ6 code (Wolery and Daveler, 1992). A reaction path calculation follows (or predicts) the evolution of a reacting system. In EQ6, this usually refers to a system consisting of an aqueous solution and some minerals with which it is not in partial equilibrium (the "reactants"). As the overall reaction progresses, the composition of the aqueous solution changes (including *pH*, *Eh*, etc.) and the solution may become saturated with new minerals. These are generally al-

lowed to form under the condition of partial equilibrium and are generally referred to as "product" minerals. Other types of reaction progress calculations are possible. For example, the calculation might follow mixing with a second fluid (with no reactant minerals). Here the mixing is treated as an irreversible "reaction." Again, the solution composition changes and product minerals form. Change in temperature, to simulate heating or cooling, can also be included in a reaction path calculation. Conceptually, what a reaction progress calculation really deals with is a set of **irreversible** reactions (associated with the "reactants") and a set of **reversible** reactions (reactions in a state of partial equilibrium, including those describing ion pairing, complexing, and the formation and possible re-dissolution of "product" minerals). In this more general sense, a "reactant" mineral, if growing irreversibly, is really a **product** in the true sense of that word. Similarly, a "product" mineral, when it re-dissolves reversibly, is really a **reactant** in the true sense of that word.

Reaction path calculations are dynamic, in the sense that the user must provide some kind of rate expressions to describe the progress of the irreversible reactions. These may take one of two forms. One is to specify so-called relative rates, which is really just a way of relating the magnitudes of the irreversible rates among one another. Technically, these are written relative to the advancement of an overall reaction progress variable. The other method is to specify actual rates. These are written relative to a time variable. It is possible to mix actual and relative rates in the same calculation, though there are a few dangers (see Appendix D). As long as an actual rate is specified for at least one irreversible reaction, the calculation takes place in a time frame. Otherwise, the calculation is independent of time and represents a kind of titration process. In addition to the inclusion of rate laws, EQ6 calculations also include all of the phenomena addressed by EQ3NR (e.g., speciation, activity coefficients). For a more detailed description of EQ6 and its modeling capabilities, see the EQ6 Theoretical Manual and User's Guide (Wolery and Daveler, 1992).

Reaction path calculations are sometimes called *equilibrium step* calculations. This follows from the historical fact that most such calculations have involved the calculation of the equilibrium states corresponding to a sequence of systems whose gross compositions differ due to the effect of incremental addition of the components of which the reactants are composed. However, this only represents an extreme case. A reaction path calculation may also include elements of simple disequilibrium, for example by specifying that certain minerals are not to be precipitated, regardless of the values of the corresponding saturation indices. Furthermore, a reaction path calculation may also be defined by introducing rate laws to control not only the addition of reactants, but also the formation of product phases (see for example Delany, Puigdomenech, and Wolery, 1986). In principle, a reaction path calculation may be defined entirely in terms of rate laws. However, this represents an extreme case which is not yet generally feasible in the case of aqueous geochemical systems because the understanding of the kinetics pertinent to such systems is presently too limited.

The term *mass transfer calculation* is sometimes used in modeling aqueous geochemical systems. This generally refers to calculation of the equilibrium state of a system under conditions in which components are added to or removed from the aqueous solution. For example, one might ask the question, what would be the composition (*pH*, etc.) of a ground water if it were stripped of its CO_2 content? For another, what would be the composition of surface sea water if calcite were precipitated to eliminate supersaturation? How much calcite would be precipitated? The

same questions may be answered by reaction path calculations. Indeed, reaction path calculations include mass transfer calculations. However, not all mass transfer calculations are reaction path calculations.

The term *reaction path calculation* implies a focus on the path by which a process reaches an end point. Thus, the calculational process must proceed in small steps. The term "mass transfer calculation" often implies a focus just on the end point itself. It also usually implies a small number of heterogeneous reactions, each of which is specified in advance and continues from the beginning to the end point. In a reaction path calculation, the operative set of heterogeneous reactions usually changes along the path. For example, dissolving reactants may become exhausted, new product phases appear from time to time, and some of these may later also become exhausted. In this sense, a reaction path calculation consists of a sequence of mass transfer calculations.

Geochemical modeling calculations are not limited to speciation-solubility calculations and reaction path calculations. Another type of calculation involves the construction of activity-activity diagrams, the most common of which is the *Eh-pH* diagram (see for example Garrels and Christ, 1965; Krauskopf, 1967; Stumm and Morgan, 1981; or Nordstrom and Munoz, 1985). Such diagrams are two-dimensional representations of chemical equilibria (other examples include diagrams for *pe-pH*, $log a_{SiO_{2(aq)}}$-*pH*, and $log\,(a_{Ca^{2+}}/a^2_{H^+})$-$log\,(a_{Na^+}/a_{H^+})$. A common purpose of such a diagram is to graphically illustrate constraints on phase equilibria. For example, it may be possible to show that the solution may be in equilibrium simultaneously with minerals A and B, or B and C, but not A and C. Another purpose, especially of *Eh-pH* diagrams, is to show fields corresponding to dominant solution species. A major limitation on the use of such diagrams is the fact that they are only two-dimensional. This generally means that a fair number of *side conditions* are present, such as assumptions of saturation with one or more minerals. Commonly used side conditions also include specified constant activities or concentrations for species not associated with the diagram coordinates. When concentrations are used in this manner, there must also be some assumption about the activity coefficients, such as a constant value of ionic strength.

None of the codes in the EQ3/6 software package generate activity-activity diagrams. However, the results of EQ3NR and EQ6 calculations may be plotted on such diagrams. This can be very helpful in understanding or depicting a process. However, readers are cautioned when doing this to be sensitive to the issue of side conditions. These conditions may vary for a sequence of related water samples, whose compositions might be used in making speciation-solubility calculations. Plotting the results for these waters on the same activity-activity diagram may therefore involve the assumption that the effects of these variations are negligible, which they may or may not be. The side conditions may also vary along a computed reaction path.

In running EQ3NR and EQ6, users should keep in mind that geochemical modeling of most natural systems requires some element of thermodynamic disequilibrium. Thus, geochemical modeling must be more than mere equilibrium modeling of natural systems. Its goal, insofar as is possible, should be to assess the extent of such disequilibrium, and to predict the approach to overall equilibrium. This is especially true with regard to redox reactions.

The EQPT code (Daveler and Wolery, 1992) does not carry out any modeling calculations. It is a data file preprocessor which performs a number of functions. It checks the composition, charge, and reaction coefficient data in a data file for internal consistency and fits interpolating polynomials to various temperature dependent data which are organized in the data file on temperature grids. Such data include certain activity coefficient parameters, such as Debye-Hückel parameters, and the equilibrium constants for the reactions represented on the data file. In addition, in the case of data files specific to the formalism of Pitzer's (1973, 1975, 1979, 1987) equations, observable interaction coefficients are mapped to a set of conventionally defined primitive interaction coefficients. It then writes an unformatted copy of the data file, making the following substitutions: coefficients for interpolating polynomials for corresponding data originally on a temperature grid, and in the case of files specific to the Pitzer formalism, conventionally defined primitive interaction coefficients for the corresponding observable interaction coefficients. Although there is a user's guide for this code (Daveler and Wolery, 1992), the installation instructions given in Chapter 3 of the present report are sufficient to run this code in the absence of any trouble. The EQPT User's Guide exists primarily to satisfy the NUREG-0856 documentation requirements (Silling, 1983).

The type of modeling performed by EQ3/6 is sometimes referred to as *chemical modeling*. This implies attempting to understand the properties of solutions in terms of the chemical species present (the speciation). In aqueous solutions, these may include the following: simple ions, oxyions, and hydroxyions; molecular neutral species such as $CO_{2(aq)}$ and $CH_{4(aq)}$; strong complexes such as $ZnCl_{3(aq)}$ and $UO_2(CO_3)_2^{2-}$; and less strongly bound ion pairs such as $NaSO_4^-$ and $CaCO_3^o$. Although speciation is often treated according to equilibrium conditions, this chemical approach also permits treatment of disequilibrium as well, though with a concomitant increase in the amount of input data required (e.g., additional chemical analysis, rate laws).

Chemical modeling is not synonymous with *thermodynamic modeling* or *equilibrium modeling* for another very important reason. Thermodynamic modeling is mainly focused on explaining and predicting thermodynamic properties. This does not always emphasize an understanding based on formal recognition of the actual chemical species present. Thus, weak chemical phenomena such as ion pairing may ignored, and the corresponding thermodynamic effects interpreted by means of effects on the activity coefficients.

It is part of the nature of chemical models that speciation, equilibrium constants, and activity coefficients must be determined simultaneously from measurements on various physical properties of solutions. In most cases, however, the experimental data that are available are insufficient to define everything without introducing some assumptions. For example, in some systems, it is possible to make direct measurement of speciation, such as by spectroscopic techniques. In a great many systems, however, such direct evidence is lacking and the available measurements must be interpreted by assuming a given speciation. Furthermore, equilibrium constants and activity coefficients are not parameters which can themselves be directly measured. Instead, they must be inferred from measurements of other properties such as the osmotic coefficient or the electrical conductivity, (see for example Harned and Owen, 1958; Robinson and Stokes, 1965; and Baes and Mesmer, 1976). This interdependence among speciation, equilibrium constants, and activity coefficients often makes it difficult to arrive at a demonstrably unique model, particularly without simultaneously considering a lot of data, especially of different kinds.

Because of these problems, a major issue concerning speciation, equilibrium constants, and activity coefficients is not whether they are absolutely correct, but whether or not a given set used in chemical modeling (or thermodynamic modeling) is internally consistent. Geochemical modeling has seen the development of a number of computer codes (see Section 1.3) which represent a combination of submodels and supporting data taken from a variety of sources. These codes have sometimes been referred to as "computerized chemical models." This terminology may be somewhat misleading in that the internal consistency of the overall product is usually not well established. Indeed, it is usually possible to do this only for limited systems. The Harvie, Møller, and Weare (1984) model of the sea-salt system at 25°C is one of the best examples of a consistent model. Users of geochemical modeling codes need to be aware of the strengths and limitations of the submodels and data employed in these codes. They should not be used as black boxes.

EQ3/6 has been designed primarily as a calculational tool which allows a number of user-controlled options concerning the treatment of speciation, equilibrium constants, or activity coefficients. It does not represent one specific model. The species employed and the thermodynamic data used in a given run are determined by the contents of the supporting data file chosen for that run by the user. Currently, the user may choose from among three activity coefficient submodels. Choices are restricted only by certain consistency considerations, as noted below.

The relationship of the codes and data files in version 7.0 of EQ3/6 is depicted in Figure 1. This figure also shows the flow of information involving these elements of the software package. These are provided in formatted ASCII and are called **data0** files. At present, there are five such data files, denoted by the suffixes **com**, **sup**, **nea**, **hmw**, and **pit** (e.g., **data0.com**). EQPT processes these one at a time (looking for a file named simply **data0**, though these files are normally stored under names which include the relevant suffixes) and writes a corresponding unformatted data file, which is called **data1**. These are also normally stored under names including the relevant suffixes. To run EQ3NR or EQ6, the user must provide one of these files, which is known to each code simply as **data1**.

The user must select which of the five data files is most appropriate to a given problem. Each data file corresponds to a general formalism for treating the activity coefficients of the aqueous species and contains the relevant activity coefficient data as well as standard state thermodynamic data. The speciation (number and identity of species in the corresponding model) also varies from data file to data file. The activity coefficient formalisms currently built into EQ3/6 are discussed in Chapter 3 of the EQ3NR Theoretical Manual and User's Guide (Wolery, 1992). The **com**, **sup**, and **nea** data files are specific to a general extended Debye-Hückel formalism and can be used by EQ3NR and EQ6 with either the Davies (1962) equation or the B-dot equation (Helgeson, 1969). These equations are only valid in relatively dilute solutions. The **hmw** and **pit** data files are specific to the formalism proposed by Pitzer (1973, 1975, 1979, 1987) and can be used to model solutions extending to high concentrations. However, the scope of chemical components covered is smaller. The temperature limits on the data files also vary, from 25°C only to 0-300°C.

Some important characteristics of the data files in version 7.0 of EQ3/6 are given in Table 1. The statistical data pertain to the "R10" data files, where "R10" is a stage number (a low level configuration control identifier). The **com** (for "composite") data file is the largest of the three data files specific to the extended Debye-Hückel formalism. It is a product of Lawrence Livermore National Laboratory (LLNL) drawing on many data sources, including those on which the other

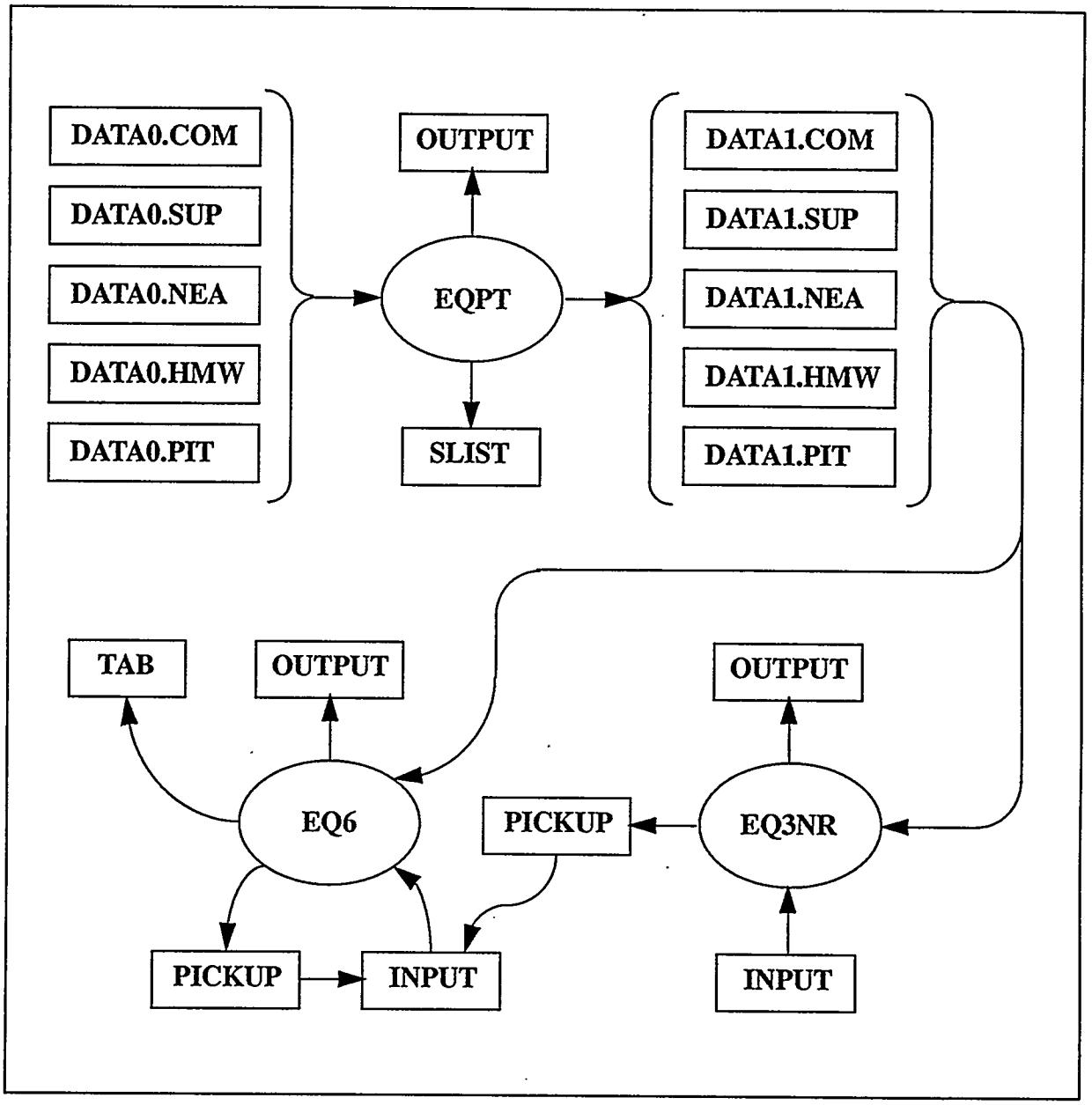


Figure 1. The flow of information among the computer codes EQPT, EQ3NR, and EQ6. Computer codes are represented by ovals, files by rectangles.

four data files are based. The **sup** data file is based entirely on SUPCRT92 (Johnson, Oelkers, and Helgeson, 1992), a software package and data base founded on the work of Helgeson and Kirkham (1974ab, 1976), Helgeson et al. (1978), Tanger and Helgeson (1988), Shock and Helgeson (1988, 1989, 1990), Shock, Helgeson, and Sverjensky (1989), Johnson and Norton (1991), and Shock et al. (1992). The **nea** data file is based entirely on Grenthe et al. (1989, draft report), a product of the Data Bank of the Nuclear Energy Agency of the European Community. This report has recently been published as Grenthe et al. (1992). The **hmw** data file is based on Harvie, Møller, and Weare (1984). The **pit** data file is based mostly on data summarized by Pitzer (1979).

Table 1. Major characteristics of the current five EQ3/6 data files ("R10" versions).

File Name (Suffix)	Source	Activity Coefficient Formalism	Temperature Limits	Number of Chemical Elements	Number of Basis Species	Number of Aqueous Species	Number of Pure Minerals	Number of Solid Solutions	Number of Gas Species
com	GEMBOCHS (LLNL)	Extended Debye-Hückel	0-300°C	78	147	852	886	12	76
sup	SUPCRT92	Extended Debye-Hückel	0-300°C	69	105	315	130	0	16
nea	NEA draft report	Extended Debye-Hückel	0-300°C	32	50	158	188	0	76
hmw	Harvie, Møller, and Weare (1984)	Pitzer's Equations	25°C only	9	13	17	51	0	3
pit	Pitzer (1979)	Pitzer's Equations	0-100°C	52	62	68	381	0	38

All five data files are maintained at LLNL in a relational data base and manipulated by various FORTRAN-like scripts (Delany and Lundeen, 1991). This relational data base is part of the Yucca Mountain Site Characterization Project's Technical Data Base. It is also part of a system which supersedes one based on the MCRT code (Jackson et al., 1988).

The **sup** data file has a high level of internal consistency among the standard state thermodynamic data. In addition, the temperature-pressure dependence of these data are represented by a suite of equations of state for minerals, gases, and aqueous species that are well established in the geochemical literature (see references noted above). This data file covers a wide range of chemical elements and species of interest in the study of rock/water interactions (e.g., components which make up the major rock-forming and ore-forming minerals). It also includes a large number of organic species, mostly of small carbon number (C_2-C_8). The **nea** data file is something of a specialty item. Its strongest point is a thorough representation of the thermodynamics of uranium species.

The **com** (composite) data file encompasses a much broader range of chemical elements and species. It includes the data found on the **sup** and **nea** data files, with preference given to data from the former in cases of overlap. It also includes some data found in the **hmw** data file, as well as other data which do not appear in any of the other data files. Some of these data are estimates based on correlations or extrapolations (as to higher temperature), and are not tied directly to experimental measurements. The **com** data file thus represents a melange of data, which by its nature offers less assurance of internal consistency. However, this offers the only means presently available for modeling aqueous solutions with a high degree of compositional complexity, such as the fluids expected to be found in and about a facility for the geologic disposal of industrial or nuclear waste (e.g., the potential repository for high-level nuclear waste at Yucca Mountain, Nevada).

The **hmw** data file has the highest degree of internal consistency of any of the five data files, including mutual consistency of activity coefficient data and standard state thermodynamic data. It can be applied to dilute waters or concentrated brines. However, it only treats the set of components present in the "sea-salt" system (the major cations and anions present in seawater, including carbonate and bicarbonate). The geochemically important components aluminum and silica are not included. Also, this data file is limited to a temperature of 25°C. The **pit** data file can also be applied to concentrated brines. It covers a larger set of components, but these mostly involve other cations and anions of strong electrolytes. Examples include lithium and bromide. This data file nominally covers the temperature range of 0-100°C. However, it represents a melange of data, not a carefully crafted internally consistent set.

The data file preprocessor EQPT takes as its input a formatted data file known as a **data0** file, producing as its primary output an unformatted equivalent (with the substitutions discussed previously) known as **data1**. A formatted equivalent known **data1f** (not shown in Figure 1) is also produced, but is used only for debugging purposes. EQPT also writes to the screen file and an **output** file, both of which are generally significant only if an error condition is encountered. In addition, it writes an **slist** (species list) file. This is very useful to the user, as it provides a compact list of the species that are represented on the data file.

A speciation-solubility problem to be run with EQ3NR is described on the EQ3NR **input** file. The code then produces an **output** file describing the results of the calculation. While the code is running, it writes to a screen file, primarily to apprise the user of what is happening. It also writes a **pickup** file, which contains a compact description of the aqueous solution. This may be used as the bottom part of an EQ6 **input** file. It has no other real use.

A reaction-path problem to be run with EQ6 is similarly described on the EQ6 **input** file. EQ6 in turn writes its own **output** file, as well as a **tab** file which contains certain data in tabular form suitable for supporting local graphics postprocessing. Like EQPT and EQ3NR, EQ6 also writes to the screen file while it is running. In addition, EQ6 writes its own **pickup** file, which may be used as an **input** file to restart a reaction path calculation where a previous run segment ended. The EQ6 **pickup** file is a complete EQ6 **input** file, whereas the EQ3NR **pickup** file is only the bottom half of an EQ6 **input** file.

In order to use EQ3/6 properly, the user should have a good grasp of aqueous systems geochemistry and major concepts which underlie it. This topic is addressed in a number of books which provide an entry to the subject via formal or informal training. Such books include the works of Garrels and Christ (1965), Krauskopf (1967), Berner (1971), Freeze and Cherry (1979), Snoeyink and Jenkins (1980), Stumm and Morgan (1981), Drever (1982), and Nordstrom and Munoz (1985). Physical chemistry and thermodynamics are both generally very important in geochemistry, and users may wish to consult more advanced texts on these subjects, such as Pitzer and Brewer (1961), Robinson and Stokes (1965), Denbigh (1971), and Baes and Mesmer (1976). The volumes on chemical modeling edited by Jenne (1979) and Melchior and Bassett (1990) are valuable resources at an advanced level. Shorter reviews of the state of the art in chemical modeling are given by Jenne (1981) and Bassett and Melchior (1990). Examples of the usage of EQ3/6 are given in the EQ3NR Theoretical Manual and User's Guide (Wolery, 1992) and the EQ6 Theoretical Manual and User's Guide (Wolery and Daveler, 1992). A short review of EQ3/6, including recent applications, is given by Wolery et al., 1990).

Geochemical modeling codes are not black boxes. Much of the usefulness (or lack of) that comes from their use is determined by the level of knowledge brought to bear by the user. The most important prerequisite of successful geochemical modeling is the ability to pose the problem to be solved in precise terms. Often, the results of a calculation indicate that the details of the problem need to be reformulated or that new data need to be obtained to define the problem. Any results obtained by modeling calculations should be weighed against descriptive knowledge of the system being modeled.

One of the reasons that this is very important is that **no geochemical modeling code now in existence provides for all of the possible physical and chemical phenomena which may potentially be required.** Furthermore, **no data base supporting such a code is truly complete for all of the chemical elements of potential interest.** The user must be able to identify and evaluate the "holes" in the existing modeling technology (codes and data bases) that pertain to a given application. Otherwise, the results may be spectacularly incorrect. Often, what the user must do is evaluate which parts of the application, if any, can be successfully addressed by existing modeling technology, and which parts can not.

There are two reasons why existing modeling technology is not globally adequate for applications in aqueous geochemistry. The first is that the requisite science itself is not adequately developed in a number of areas. This applies to both theory and data. Geochemical modeling is in consequence very much in a period of significant evolution. The other reason is the difficulty of incorporating even all of the existing pertinent theory and data in a single piece of software. Such a software product would be much larger than the existing EQ3/6, which is already a fairly large code.

Although there is currently no globally adequate set of geochemical modeling codes, the existing codes do cover many of the pieces allowed by the existing science. To some extent, therefore, the user may be able "cover the bases" by choosing the right code or combination of codes. Some of the other codes of significance are briefly discussed at the end of this chapter. First, however, we review what is new in EQ3/6, version 7.0.

1.2. What's New in Version 7.0

The purpose of this section are to briefly summarize how version 7.0 differs from versions 6.0 (3245.0288) and 6.1 (3245.0888). In brief, the improvements made to the new version are:

- The set of five new data files described earlier in this chapter. Organizationally, these attempt to provide within any single file a more consistent model encompassing speciation, equilibrium constants, and activity coefficient parameters. The user should choose the data file that is most appropriate to his application. In terms of content, many improvements have been made in terms of additions, substitutions, and deletions. Those data files based on Pitzer's equations now contain values for the A_ϕ Debye-Hückel parameter. They were formerly calculated from the related $A_{\gamma,10}$ parameter, a procedure that in certain circumstances gave rise to small inconsistencies (see Chapter 3 of the EQ3NR Theoretical Manual and User's Guide, Wolery 1992).
- The approximation for the activity of water formerly used in conjunction with the B-dot equation has been replaced by one which uses the B-dot parameter itself. An approximation for the activity of water consistent with the Davies (1962) equation has been added for use in conjunction with the Davies equation itself.
- The approximation for the activity coefficient of $CO_{2(aq)}$ which was formerly used in conjunction with the B-dot equation has been replaced by one which is more stable and has fewer parameters. The old approximation was sufficiently unstable when evaluated even moderately out of range that it occasionally caused runs of both EQ3NR and EQ6 to terminate unsuccessfully.
- Addition of optional menu-style input file formats for EQ3NR and EQ6. The export package includes some utilities to convert files in the older "W" format to the menu-style "D" format. "D" format may be more suitable for occasional users.
- The convergence behavior in dealing with brine solutions has been significantly improved by changes to the hybrid Newton-Raphson algorithm employed in EQ3NR and EQ6. A computational singularity in the method that occurred in EQ6 when a mineral assemblage fixed the activity of water (e.g., gypsum plus anhydrite) was removed. The method still exhibits some convergence trouble in the case of extremely concentrated solutions.

- EQ6 code was modified to include values for activation energies or activation enthalpies as part of the **input** file data for actual kinetic rate laws.
- Addition to the EQ3NR code of a capability to deal with the quantity *pHCl*, which is part of an experimental/theoretical methodology for dealing with *pH* in brines (Knauss, Wolery, and Jackson, 1990). See also Knauss, Wolery, and Jackson (1991) and Mesmer (1991).
- Deletion from the EQ3NR code of the capability for direct input of alkalinity. The reasons for doing this and suggested alternatives are discussed in Chapter 2 of the EQ3NR Theoretical Manual and User's Guide (Wolery, 1992).
- Deletion of experimental and obsolete options.
- More extensive testing and elimination of bugs.

Some of the other features and capabilities discussed in the original EQ3/6 development plan (McKenzie et al., 1986) have not been implemented in version 7.0. These include:

- Rock-centered flow-through system model to complement the existing fluid-centered flow-through system model (EQ6). These are both pseudo-1-dimensional transport plus chemistry models.
- Coupled 1-dimensional transport model (EQ6).
- Explicit gas phase model (EQ6). Fugacities can currently be fixed. Rigorously, addition of an explicit gas phase requires variable pressure (specified or computed), hence the pressure corrections noted below.
- Surface chemistry models (sorption).
- More advanced solid solution models, primarily of clays and zeolites.
- More advanced models of mineral dissolution and precipitation kinetics.
- New models of activity coefficients in aqueous solutions.
- Pressure corrections (pressure is currently parameterized as a function of temperature in the following manner: 1.013 bar up to 100°C and the steam-liquid water curve from 100°C to 300°C).

The first three items (rock-centered flow-through system, coupled 1-D transport, explicit gas phase) are things which only involve code development to incorporate existing science. The last five (surface chemistry, advanced solid solution models, more advanced dissolution-precipitation kinetics models, new activity coefficient models, and pressure corrections) are all significantly science-limited; that is, the state of the underlying science must be further developed to allow much in the range of broad application. Two of these areas (surface chemistry and mineral dissolution and precipitation kinetics) are currently among the most active and productive areas of research in aqueous geochemistry. In fact, the trend in the latter is now strongly aimed at developing kinetics models which are directly linked to surface chemistry (see for example Blum and Lasaga, 1991, and references therein).

A few of these items have been the subject of experimental prototypes at LLNL. Richard Knapp has developed a version of EQ6 for coupled 1-D transport (not part of work for the Yucca Mountain Project). Brian Viani has developed another for the study of cation exchange and advanced solid solution models. These experimental versions are not available for general distribution outside of LLNL.

The items mentioned above can be supplemented by the following, which have also at one time or another been considered for inclusion in EQ3/6:

- Aqueous redox disequilibrium (EQ6).
- Aqueous redox kinetics (EQ6).
- Isotopic fractionation (EQ6).

This list and the larger one taken from McKenzie et al. (1986) give a pretty good picture of what is presently not in EQ3/6 that one might conceivably need for certain applications.

1.3. Other Codes of Interest

The purpose of this section is to note some other codes which are used in the modeling of aqueous geochemical systems. It is not possible to present here a complete listing of all such codes, nor to provide a critical review of those which are listed. No slight is intended by the omission here of any code. For a more comprehensive listing of codes used in geochemical modeling of aqueous systems, the reader is referred to Nordstrom et al., (1979), Nordstrom and Ball (1984), and Bassett and Melchior (1990).

The codes noted here more or less have the following in common: fairly widespread usage, developmental aspects or capabilities of significant interest, and some kind of documentation (preferably including a user's guide) in the open literature. Most of these come in families, reflecting continued development (not always by the same people or groups). Some have been compared with EQ3/6 in code-to-code verification exercises, which makes them of special note.

Some of these codes have "mass transfer" capabilities, but not full reaction path capability. It is possible to use some of these codes to calculate reaction paths. In general, this requires running such codes in segments. This puts more of a burden on the user. Apart from that, there are two important issues to consider in choosing to use such a code in this manner. The first is that it must be possible to stop a segment, or at least obtain a description of the system, whenever a "significant" event occurs, such as when a reactant becomes exhausted, a new product mineral forms, or an existing product mineral exhausts. Put another way, the code must be able to find the point of reaction progress corresponding to any such event. If it doesn't do this, the user may have to locate such an event by iteratively running the code. The second issue to consider is whether or not the code handles the mass transfer calculations in a rigorous or only approximate manner. Some codes do not carry a full suite of balance equations. This can also be a problem in a "reaction path" code. For example, the earliest version of the PATHI (Helgeson, 1968) code did not include a balance equation to calculate changes in the amount of solvent water; this was later corrected (Helgeson et al., 1970).

In most of the codes considered below, the speciation model is determined by the contents of a supporting data file (this is the case for EQ3/6), or in some cases, the input file. It is then a fairly straightforward process to add new species or delete old ones. In some codes, however, notably in the WATEQ and SOLMNEQ families (see below), the speciation model is directly written into the source code. This "hard-coding" of the speciation model generally puts modification beyond the capability of the typical user.

The WATEQ (pronounced "WATEK") family comprises one important family of codes used in geochemical modeling of aqueous systems. Its members include:

- WATEQ (Truesdell and Jones, 1974)
- WATEQF (Plummer, Jones, and Truesdell, 1976)
- WATEQ2 (Ball, Jenne, and Nordstrom, 1979; Ball, Nordstrom and Jenne, 1980)
- WATEQ3 (Ball, Jenne, and Cantrell, 1981)
- WATEQ4F (Ball, Nordstrom, and Zachmann, 1987; Ball and Nordstrom, 1991)

All the codes in this series are speciation-solubility codes. They are products of the United States Geological Survey. The original WATEQ was written in PL/1. A FORTRAN version, WATEQF, appeared shortly thereafter. The most recent version, WATEQ4F, is not only written in FORTRAN, but also adapted to run on IBM PCs and Apple Macintoshes. The main thrust in the development in the WATEQ family has been the development of the thermodynamic data base, which supports calculations over the range 0-100°C. In addition to the above references, we also note that the WATEQ literature includes Nordstrom et al. (1984), a data base update not specific to a single WATEQ code in the series, and the more recent data compilation of Nordstrom et al. (1990). All known versions of WATEQ use simple extended Debye-Hückel equations for the activity coefficients of the aqueous species, hence are restricted in application to the modeling of dilute waters. The speciation model in all of the above versions of WATEQ is written directly into the source code.

Another important family consists of the following two codes:

- PHREEQE (Parkhurst, Plummer, and Thorstenson, 1980)
- PHRQPITZ (Plummer et al., 1988)

PHREEQE (pronounced the same as "freak") is a code which performs both speciation-solubility calculations and reaction path calculations. PHRQPITZ ("freak-pitz") is essentially a version of PHREEQE which employs Pitzer's equations for the activity coefficients of the aqueous species. These are also products of the United States Geological Survey. They are written in FORTRAN and run on PCs. PHREEQE has no data base of its own; the original version used the WATEQ2 data base. It incorporates a simple ion exchange model option. The PHRQPITZ data base is an extension of the work of Harvie, Møller, and Weare (1984). The reaction path capabilities of PHREEQE and PHRQPITZ require the user to interact with the code when decisions must be made, such as whether to precipitate a mineral with which the solution has just reached saturation. In contrast, in EQ3/6, such decisions are preprogrammed on the EQ6 **input** file. PHREEQE has been quite widely distributed, and a number of locally modified versions exist.

The reaction path capabilities of PHREEQE and PHRQPITZ are not fully automated. These codes pause at the end of a run segment (or potential end of such a segment) and query the user for further directions. For example, if the solution is initially undersaturated with respect to calcite, the code will pause if the corresponding saturation index rises and reaches a value of zero. The code will then query the user whether or not to begin precipitating this mineral so as to maintain a state of partial equilibrium.

A significant caution concerning these codes is that they do not include a balance condition for tracking changes in the amount of solvent water (Parkhurst, Plummer, and Thorstenson, 1980, p. 47-48; Plummer et al., 1988, p. 14). The amount of solvent is considered to be fixed at a mass of 1 kg. This is likely to be a significant problem in modeling reaction paths involving the dissolution or precipitation of relatively large amounts of hydrous minerals, such as gypsum or epsomite.

PHREEQE and PHRQPITZ have been favorite choices in code comparison exercises involving EQ3/6. A set of code-to-code verification tests comparing PHREEQE and EQ3/6 were conducted by INTERA (1983). A similar exercise was conducted more recently by Puigdomenech and Emrén (1990), who also included in the study a third code. Also, problems from the PHREEQE and PHRQPITZ manuals have been used at LLNL to compare these codes with EQ3/6 (see the EQ3NR Theoretical Manual and User's Guide, Wolery, 1992, and the EQ6 Theoretical Manual and User's Guide, Wolery and Daveler, 1992).

The REDEQL family includes the following members:

- REDEQL (Morel and Morgan, 1972)
- REDEQL2 (McDuff and Morel, 1973)
- GEOCHEM (Mattigod and Sposito, 1979)

These codes are all written in FORTRAN. GEOCHEM is noteworthy for a large data base covering many organic species. In general, the codes in this series appear to have only been developed for application to dilute solutions at 25°C. These codes include ion exchange and adsorption models and have a mass transfer capability that is thought to be similar to that in MINEQL (see below). Apart from GEOCHEM, the codes in this family do not seem to be used much these days.

The MINEQL family includes the following codes:

- MINEQL (Westall, Zachary, and Morel, 1976)
- MINTEQ (Felmy, Girvin, and Jenne, 1984)
- MINTEQA2 (Allison, Brown, and Novo-Gradac, 1991)
- HYDRAQL (Pepelis, Hayes, and Leckie, 1988)

MINEQL is a conceptual descendent of REDEQL. It was intended as a smaller, leaner code. It has been very widely distributed, and appears to have been modified locally by many of the recipients. The original version did not include surface chemistry models, but these have been added in various derivative codes, including MINTEQ, MINTEQA2, and HYRAQL. MINEQL has

a mass transfer capability that has two shortcomings. Like PATHI and PHREEQE/PHRQPITZ, it lacks a balance equation to compute changes in the amount of solvent water. However, it also lacks the ability to compute changes in both *pH* and *Eh*, owing to another missing balance equation. The mass transfer capability in REDEQL is thought to be similar. The original MINEQL used the Davies (1962) equation for the activity coefficients, hence was restricted to the treatment of dilute solutions. Also, the original code appears to have been restricted by means of the original data base to calculations pertaining to 25°C.

MINTEQ was conceptually the product of combining MINEQL with the WATEQ3 data base. The problem in MINEQL of computing changes in *pH* and *Eh* caused by mass transfer was fixed. Several new models for dealing with surface chemistry were also added. MINTEQA2 is a relatively recent derivative of MINTEQ developed by the Environmental Protection Agency.

HYDRAQL is a derivative of the original MINEQL which has been extensively adapted for adsorption modeling. It runs on IBM PCs and Apple Macintoshes.

Another family of interest is the SOLMNEQ family, which includes:

- SOLMNEQ (Kharaka and Barnes, 1973)
- SOLMINEQ.88 (Kharaka et al., 1988; Perkins et al., 1990)

The SOLMNEQ (pronounced "solm-nek") was a speciation-solubility code written in PL/1 and used in modeling geothermal fluids. Its data base extended to 350°C. Like WATEQ and PHREEQE, SOLMNEQ was a product of the United States Geological Survey. Like WATEQ, the speciation model was written directly into the source code.

SOLMINEQ.88 (pronounced "sol-mi-nek eighty-eight") is a major revision of the original SOLMNEQ, produced by the United States Geological Survey and the Alberta Research Council. It is written in FORTRAN, and has been adapted to run on IBM PCs. SOLMINEQ.88 has a number of capabilities that exceed those of the original SOLMNEQ. Among these are pressure corrections, which extend to 1 kilobar. There is also an option to use Pitzer's equations. Beyond that, however, SOLMINEQ.88 is not just a speciation-solubility code. It incorporates various kinds of mass transfer options (boiling, fluid mixing, gas partitioning, mineral dissolution and precipitation) and ion exchange and ion adsorption models. The code appears to be capable of at least some reaction path capability, in the sense of stringing together successive run segments. SOLMINEQ.88 does not appear to carry a balance equation to compute changes in the amount of solvent water. Also, the speciation model is directly written into the source code.

SOLVEQ/CHILLER (Reed, 1977, 1982; Reed and Spycher, 1989; Spycher and Reed, 1989ab) is a software package that also has a strong orientation toward modeling the geochemistry of hydrothermal systems. SOLVEQ is a speciation-solubility code; CHILLER is a reaction-path code (an earlier code in the package called MINSOLV is now obsolete). These codes are written in FORTRAN and run on IBM PCs and Apple Macintoshes. The temperature range of these codes extends to 350°C. The reaction path capabilities of CHILLER deal with boiling, evaporation, fluid-mixing, and mineral-gas-water interactions. It has some fairly advanced capabilities for dealing with boiling and condensing processes, and in the process for dealing with a gas phase. For example, boiling may be modeled in five ways: with an enthalpy balance constraint, following a

set of *P-T* points input by the user, following a standard *P-T* curve (similar to that used in the EQ3/6 data base), as an isothermal process, or as an isobaric process. It appears, however, that the pressure dependence of some thermodynamic data is ignored. CHILLER utilizes a full suite of balance equations for the chemical components.

"The Geochemist's Workbench" is a software package written by Craig Bethke and others of the Hydrogeology Program of the University of Illinois. It is an unpublished work distributed via trade secrecy license. It consists of a set of codes for manipulating chemical reactions (Rxn), plotting various types of stability diagrams (Act2, Tact), speciation and reaction path calculations (React), and plotting reaction path calculations (Gtplot). React, which is most directly comparable to EQ3/6, is a successor to an earlier code called Gt. It includes an option for using Pitzer's equations. It also features options for a fluid-centered flow-through system and a solid-centered flow-through system ("flush" model). The code will integrate simple rate laws for mineral dissolution and precipitation. It can also trace the equilibrium fractionation of stable isotopes during the course of a reaction path. The React code may be used in some code-to-code verification of EQ3/6.

2. The EQ3/6 Export Package

The purpose of this chapter is to describe the EQ3/6 export package. The following chapter describes how to install the software. The exact contents of the export package may vary from time to time, reflecting various updates. This description here is in general terms and exemplified by the contents of the version 7.0x export package as of June 29, 1992. The "**x**" appended to the version number as given above indicates modifications to the original version 7.0 export package of November, 1990. There are three primary differences in these packages: (1) substitution of the R10 data files for the R7 data files (R10 and R7 are stage numbers, which are described below); (2) addition of some new files: **DBNEWS1.txt, FIXES2.txt, FIXES3.txt, runeq36, runeqpt, rwpar.3i, h2so4p1N.3i**, and **rwtitr.6i**; and (3) source code fixes which are recorded in the **FIXES1.txt, FIXES2.txt**, etc. files described below.

Note: As of August, 1992, version 7.0x was superseded by version 7.1. It includes a set of **FIXES?.txt files** (the "**?**" is a UNIX metacharacter matching any character) extending to and including **FIXES5.txt**, and also includes the set of R16 data files in place of the R10 set. It includes 95 files and has a total size of 7.5 megabytes.

Some items in the software package, notably source codes and data files, bear stage numbers. These numbers are preceded by an "**R**" and are generally followed by a period. For example, in the case of the EQ6 source code file **eq6R119x.fsc**, the **R119x** is the stage number. Stage numbers are used to provide a fine degree of configuration control during the code development process. They can also be used to distinguish any software package items which have been updated after initial release of a new software package version. Otherwise, stage numbers are not significant to code users. An "**x**" appended to the stage number as in the above example indicates the existence of fixes. In version 7.1,the stage numbers were incremented and no "**x**" is appended.

The current list of the files in the package is given in alphabetic order in Table 2. This corresponds to the order in which the files are transmitted electronically or written onto magnetic media. Some systems can not deal with file names having more than one extension, a root name of more than 8 characters, or an extension of more than 3 characters. Therefore, an alternate set of names meeting these requirements is listed in Table 3. File name extensions beyond the first are deleted, stage numbers are deleted (e.g., **con3ifR06x.fsc** becomes **con3if.fsc**), and the remaining root names are shortened if necessary to a maximum of eight characters. Data file names are changed in the following manner to preserve stage number identity (e.g., **data0.com.R10** becomes **da0com.R10**).

Note that there are 93 files. The total size of the export package is about 7.3 megabytes. This can be compressed to about 1.9 megabytes by the UNIX utility **compress**. Compression is associated with distribution only on floppy disks. The package includes descriptive text files, FORTRAN source code files, supporting data files, a fairly large selection of sample input files, and a few sample output files.

The files whose root names are capitalized and which end in "**.txt**" are text files which describe various features of the package. The file **ALIST.txt** contains a list of the files in the package. In general, this corresponds to contents of Table 2, but may be more current. The file **ALISTA.txt** similarly contains a list of the corresponding short names. In addition to being more current than Tables 2 and 3, these files may be necessary to aid in installing the software package (see the

Table 2. List of files in the EQ3/6 export package (7.0x). Files are listed alphabetically. These are the regular file names. Table 3 contains a list of the corresponding short forms which must be utilized on some computer systems.

ALIST.txt
ALISTA.txt
BUGS.txt
CONTENTS.txt
DBNEWS1.txt
DOCS.txt
FIXES1.txt
FIXES2.txt
FIXES3.txt
FORMATS.txt
INSTALL.txt
README.txt
WHATSNEW.txt
boratebufs.3i
calhalite.6i
calhalite.6o
calnacl.3i
canshbrine.3i
caso4nacl.3i
co3aqui.3i
con3ifR06x.fsc
con3nfR03x.fsc
con6ifR06x.fsc
con6nfR03x.fsc
crisqtz.6i
cristobalite.3i
data0.com.R10
data0.hmw.R10
data0.nea.R10
data0.pit.R10
data0.sup.R10
deadseabr.3i
dedolo.6i
dedolo.6o
dedolo.6t
eq3nrR124x.fsc
eq6R119x.fsc
eqlibR153x.fsc
eqptR80x.fsc
evswgyphal.6i
fo2mineq.3i
fwbrmix.6i
fwbrmix.6o
gypnacl.6i
gypsumB.6i
h2so4p1N.3i
heatqf.6i
heatqf.6o
heatqf.6t
heatsw.6i
j13wwA.3i
j13wwsf.3i
j13wwsf.6i
j13wwtuff.6i
methane.6i
methane.6o
methane.6t
microhcl.6i
microhcl.6iD
microhcl.6o
microhcl.6t
oxcalhem.3i
pcalciteA.6i
ph4hcl.3i
pquartz.6i
pquartzA.6i
quenchfl.3i
runeq36
runeqpt
rwpar.3i
rwtitr.6i
rwtst.3i
rwtst.3o
sio2.3i
slist.com.R10
slist.hmw.R10
slist.nea.R10
slist.pit.R10
slist.sup.R10
swbasw.3i
swcaarag.3i
swco2.3i
swmaj.3i
swmajd.3i
swmajp.3i
swmajp.3o
swphcl.3i
swphcl.3o
swphclA.3iD
swphclA.3o
swrdxcp.3i
swtst.3i
swtst.3o

Table 3. List of files in the EQ3/6 export package (7.0x). Files are listed alphabetically. These are the short forms which must be utilized on some computer systems.The corresponding regular forms are given in Table 2.

ALIST.txt
ALISTA.txt
BUGS.txt
CONTENTS.txt
DBNEWS1.txt
DOCS.txt
FIXES1.txt
FIXES2.txt
FIXES3.txt
FORMATS.txt
INSTALL.txt
README.txt
WHATSNEW.txt
boratebu.3i
calhalit.6i
calhalit.6o
calnacl.3i
canshbri.3i
caso4nac.3i
co3aqui.3i
con3if.fsc
con3nf.fsc
con6if.fsc
con6nf.fsc
crisqtz.6i
cristoba.3i
da0com.R10
da0hmw.R10
da0nea.R10
da0pit.R10
da0sup.R10
deadseab.3i
dedolo.6i
dedolo.6o
dedolo.6t
eq3nr.fsc
eq6.fsc
eqlib.fsc
eqpt.fsc
evswgyph.6i
fo2mineq.3i
fwbrmix.6i
fwbrmix.6o
gypnacl.6i
gypsumB.6i
h2so4p1N.3i
heatqf.6i
heatqf.6o
heatqf.6t
heatsw.6i
j13wwA.3i
j13wwsf.3i
j13wwsf.6i
j13wwtuf.6i
methane.6i
methane.6o
methane.6t
microhcl.6i
microhcl.6iD
microhcl.6o
microhcl.6t
oxcalhem.3i
pcalcitA.6i
ph4hcl.3i
pquartz.6i
pquartzA.6i
quenchfl.3i
runeq36
runeqpt
rwpar.3i
rwtitr.6i
rwtst.3i
rwtst.3o
sio2.3i
slist.com
slist.hmw
slist.nea
slist.pit
slist.sup
swbasw.3i
swcaarag.3i
swco2.3i
swmaj.3i
swmajd.3i
swmajp.3i
swmajp.3o
swphcl.3i
swphcl.3o
swphclA.3iD
swphclA.3o
swrdxcp.3i
swtst.3i
swtst.3o

following chapter). A list of the package contents is also given in **CONTENTS.txt**. This file also gives data concerning the sizes of the files in terms of number of lines, number of words, and number of characters. These data were obtained using the UNIX utility **wc**. A list of export medium formats is given in **FORMATS.txt**. This information may be helpful in reading the software package from such media, which include magnetic tape reels and tape cartridges. The file **INSTALL.txt** contains installation guidelines and instructions, and is essentially a slightly more primitive version of the material in the following chapter of the present report. The file **DOCS.txt** contains a list of the relevant documents concerning the software package (such as the present report). A summary of changes between version 7.0 and version 6.1 (version 3245.0888) is given in **WHATSNEW.txt**. Any known but as yet uncorrected bugs are described in the file called **BUGS.txt**.Any post-release fixes to the present version may be found in files called **FIXES1.txt, FIXES2.txt**, etc. Any post-release changes to the data base files may be found in the files called **DBNEWS1.txt, DBNEWS2.txt**, etc.

The file **runeq36** is a UNIX shell script for running EQ3NR and EQ6. The file **runeqpt** is a shell script for running EQPT. These scripts are specific to the UNIX C-shell.

The "**.fsc**" files are FORTRAN source files. These are provided for EQLIB, EQPT, EQ3NR, and EQ6, which are major elements of the EQ3/6 package, and for four **input** file conversion utilities called CON3NF, CON6NF, CON3IF, and CON6IF. EQLIB, EQPT, EQ3NR, and EQ6 are maintained at LLNL in a different format in which the source code for each consists of paired "**.inc**" and "**.src**" files. The "**.inc**" files are concatenated "**.h**" or INCLUDE files. The "**.src**" files are like the "**.fsc**" files, but contain references to the "**.h**" files instead of their contents, which are mostly common blocks. The "**.fsc**" format is used in the regular export package to facilitate portability (see discussion in the following chapter).

The function and roles of EQLIB, EQPT, EQ3NR, and EQ6 have been discussed previously in this report. CON3NF is a utility program which converts EQ3NR **input** files in "W" format to "D" format. CON6NF does the same thing for EQ6 **input** files. The export package also includes CON3IF and CON6NF, which are respectively similar to CON3NF and CON6NF, but which convert input files in the "W" format corresponding to versions 6.0 (3245.0288) and 6.1 (3245.0888) to "D" format for version 7.0 (3245.1090).

The **data0** files are supporting thermodynamic data files. The different forms of these (i.e., **com** versus **hmw**) has been discussed in Chapter 1. The **out** and **slist** files are **output** and "species list" files, respectively, obtained by running the EQPT data file preprocessor code on the **data0** files to make the **data1** files which are read by EQ3NR and EQ6. The data files noted here comprise the **R10** set.

The "**.3i**" and "**.3iD**" files in the export package are EQ3NR input files in "W" and "D" format, respectively. Most of the sample input files in the export package are in "W" format. They can be converted to "D" format by running the utility code CON3NF. The "**.3o**" files are EQ3NR output files. The "**.6i**" and "**.6iD**" files are EQ6 input files in "W" and "D" format, respectively. Most of these are in "W" format. They can be converted to "D" format by running the utility code CON6NF.The "**.6o**" and "**.6t**" files are EQ6 **output** and **tab** files, respectively.

All EQ3NR and EQ6 **input** files begin with descriptive titles. These titles are also written on the **output** files, and in the case of EQ6, the **tab** file. Users should refer to these titles for an understanding of the nature of the test cases provided as part of the export package.

3. Installation of EQ3/6

The contents of the export package were listed and described in the previous chapter. The purpose of the present chapter is to explain the installation process. Installation is the responsibility of the recipient. Support for installation is not provided as part of the transfer of the software to the recipient. It may be possible to obtain such support separately if the recipient is willing and able to pay for it.

EQ3/6 is written in FORTRAN 77 and has been developed and run at Lawrence Livermore National Laboratory on an Alliant FX/80 and on Sun SPARCstations. These are 32-bit machines which employ versions of the UNIX operating system (CONCENTRIX in the case of the Alliant). Portability to other UNIX machines, including 64-bit Crays with the UNICOS operating system, should be straightforward. Portability to VAX machines running either the ULTRIX operating system (a UNIX operating system) or VMS (a non-UNIX operating system) requires a few changes but is not very difficult. Portability to machines with odd word lengths (other than 16, 32, or 64 bits), or machines which lack compilers fully consistent with the FORTRAN 77 standard, may be quite difficult.

One of the earlier forms of version 7.0x of EQ3/6 was ported to a 486DX PC (J. Pearson, 1991, written communication), using the Leahy compiler. Various bugs were found in this port and fixed in version 7.0x in January, 1992. LLNL plans to adopt the 486 as a supported platform for EQ3/6, but has not actually done so as of the date of this report. The EQ3NR and EQ6 executables, at their normal dimensioning, would require about 1.2-1.5 megabytes of memory, in excess of the DOS limit of 640 kilobytes. Reducing the dimensioning by lowering the maximum number of aqueous species and minerals treated by the software to make either code fit under the DOS memory limit is feasible for many geochemical purposes. This would certainly be true, for example, if one only wanted to run the codes with the **hmw** data file. Reportedly, the Leahy compiler provides an easy means to allow a program to exceed the DOS memory limit, though LLNL has not yet investigated this. A PC with a 286, 386, or 486 chip (with math co-processor added or built-in) should provide sufficient speed to run EQPT and EQ3NR. The 286 chip is probably not fast enough to yield acceptable run times for EQ6.

Portability to Apple Macintosh computers may also be possible, but again has not been explored at LLNL. Unlike DOS, the operating system appears to offer no problem with regard to a low limit on accessible memory. Again, however, only the higher end models (such as the Quadra) are probably fast enough to use as platforms for EQ3/6.

The installation of EQ3/6 will be discussed here in general terms, with specific details provided only for the case of installation on systems running various forms of the UNIX operating system. The recipient is presumed to be familiar with the procedures for compiling and running software on the system on which this software is to be installed. It is strongly recommended that one read this chapter in its entirety before starting the installation process.

It is recommended that you place the files in a subdirectory called **eq3_6** or something similar (e.g. **eq3_6v7.0x** or **eq3_6v7.1**, as appropriate). If you have files from previous versions of EQ3/6 on your system, you should off-load them or put them in a separate directory so as to avoid possible confusion.

If you have received the package electronically over INTERNET, the files in the package may have been placed directly in **~/eq3_6**, where the ~ is shorthand for your home directory. Alternatively, they may have been placed in some other directory of your choice, possibly on a scratch disk. Check the list of transmitted files against the list given in the **CONTENTS.txt** file (or the **ALIST.txt** or **ALISTA.txt** file). Copy the received package onto a tape cartridge, floppy disks, or a magnetic tape so that you have backup. Then move the files into the directory in which you actually wish to keep them, if they were received in a different directory.

If you have received instead floppy disks, a 1/4-inch tape cartridge, or a magnetic tape reel, create such a directory. From your home directory, enter:

mkdir eq3_6

Then move into it by entering:

cd eq3_6

If you have floppy disks, a data cartridge, or a magnetic tape that is written in **tar** format (a UNIX format), read its contents into the **eq3_6** directory by entering:

tar xvpf /dev/*XXXX*

where *XXXX* is the device name of the floppy drive, cartridge drive, or tape reel in which you have placed the medium on which you received the software. If you do not know the device name, obtain it from your system administrator. Check the medium label or the **FORMATS.txt** file for information on how the medium was written. If you have difficulty reading the file, get help from your system administrator. Check the list of extracted files against the list given in the **CONTENTS.txt** file (or the **ALIST.txt** file).

If you received the software on floppy disks, the files are compressed and you must uncompress them. To do this, enter:

uncompress *.Z

All of the originally compressed files end in "**.Z**". This suffix is removed by uncompressing the files (the compressed copies are destroyed by **uncompress**).

If you do not have a UNIX system, you probably can't read any medium created with **tar**. The following discussion pertains to a nine-track ASCII tape. At the present time, this is the only non-**tar** medium used to distributed EQ3/6. Read the ASCII tape using a local tape reading utility. Check the tape label or the **FORMATS.txt** file for information on how the tape was written. If you don't know what your local tape reading utility is, or how to use it, get help from your system administrator. You should read the files into a directory exclusively reserved for the new version of EQ3/6. Unfortunately, an ASCII tape does not preserve the names of the files written on it. The output of the reading process will be a sequence of files named something like **file001**, **file002**, **file003**, etc. You should write a script to assign the names given in the **ALIST.txt** file, assigning the first name in this file to **file001**, the second to **file002**, etc. If your system does not accept the file names listed on **ALIST.txt**, use the names listed on the **ALISTA.txt** file. The names on this

file are restricted to a root name of a maximum of 8 characters and a single extension of a maximum of 3 characters. This will probably be necessary on most VAXes. If you do not know how to write such a script, or otherwise run into difficulty, get help from your system administrator.

FORTRAN source codes are distributed primarily as "**.fsc**" files. These are concatenated main programs and subroutines into which the contents of any INCLUDE files (used mostly to hold common blocks) have been "pre-stuffed." It is recommended that such files be split into "**.f**" or "**.for**" files, each of which contains exactly one module (main program or subroutine). Splitting and compiling should be done in a subdirectory unique to each "**.fsc**" file.

On a UNIX system, create a subdirectory for EQLIB (for example) by entering:

```
mkdir eqlib
```

Move into it by entering:

```
cd eqlib
```

You may be able to split the file by entering:

```
fsplit ../eqlib*.fsc
```

Here **fsplit** is a utility that is available on many but not all UNIX systems. The "**..**" is UNIX shorthand for the directory which is one level up from the current directory, which is where the "**.fsc**" file for EQLIB is presumed to be located. If **fsplit** does not seem to be available on your system, see if there is a **splitf**. If not, or if you do not have a UNIX operating system, ask your system administrator if a splitting utility is available. Make parallel subdirectories for the other principal source code files (**eqpt** for EQPT, **eq3nr** for EQ3NR, **eq6** for EQ6).

Splitting the source files makes some aspects of installation and maintenance more convenient. For example, if you need to edit or recompile a few specific routines, you can work on them directly, rather than having to edit one very large file and having to recompile all the routines in it. Splitting is not absolutely necessary, however. It is possible to directly compile the "**.fsc**" files, though you may first have to change the "**.fsc**" to "**.f**" or "**.for**" to satisfy the compiler.

In rare circumstances, the FORTRAN source codes in the EQ3/6 package may be distributed instead as paired "**.src**" and "**.inc**" files. Source codes are maintained at LLNL in this fashion. The "**.inc**" files are concatenated "**.h**" or INCLUDE files. The "**.src**" files are like the "**.fsc**" files, but contain references to the "**.h**" files instead of their contents, which are mostly common blocks. INCLUDE files can be dealt with directly by the compiler on machines running just about any form of the UNIX operating system. They can also be handled in this fashion by the compilers on VAX machines running either the ULTRIX or VMX operating systems; however, it is necessary to proceed in a slightly different manner as described below.

On a UNIX system, the "**.src**" and "**.inc**" files for the EQLIB library should be handled by creating the **eqlib** subdirectory as above, and creating in it two subdirectories called **src** and **inc**. The "**.inc**" file must be split so that the "**.h**" files appear in the **inc** directory. There is no standard UNIX utility for performing this function, however. At LLNL, this is done by a local utility called

getinc. The "**.src**" file is split so that the "**.f**" files appear in the **src** directory, usually by using **fsplit.** The compiler gets the contents of referenced "**.h**" files when it needs them.

On a VAX (running either ULTRIX or VMS), INCLUDE files must appear in the same directory as the rest of the source code. Here the "**.src**" and "**.inc**" files for EQLIB should be split so that both the "**.h**" files and the "**.f**" files appear in the **eqlib** directory. The "**.f**" files must be modified so that references in the **include** statements to "**../inc/***XX*.**h**" are replaced by *"XX*.**h**," where *XX* denotes the name of any INCLUDE file. The "**.src**" files for EQPT, EQ3NR, and EQ6 also contain references to EQLIB include files. Place copies of the EQLIB "**.h**" files in the **eqpt, eq3nr,** and **eq6** directories and change references in the "**.f**" files to "**../../eqlib/src/***XX*.**h**" to *"XX*.**h**", where *XX* is an INCLUDE file name.

Compile the EQLIB library first, as it is required for the loading/ linking of EQPT, EQ3NR, and EQ6. It is recommended that object files first be created and then assembled in a library file. On a UNIX system, in the **eqlib** (or **eqlib/src**) directory, the following command sequence may be employed:

```
f77 -c *.f
ar rc eqlib.a *.o
ranlib eqlib.a
```

Here **f77** is the FORTRAN compiler, **ar** is the library archiving utility, and **ranlib** is a utility which modifies the library file (here **eqlib.a**) by adding a table of contents. The "**-c**" in the compiler call instructs the compiler to simply make a "**.o**" object file for each "**.f**" source code module. Without this instruction, the compiler would attempt to load/link the object files. The "**rc**" in the call to **ar** causes the library archiving utility to create the library file (the "**c**" part) and place the listed object files into it (the "**r**" part). Once the library has been made, the "**.o**" files can be deleted. You may then compile and make a library ("**.a**") file for each code in the package (EQPT, EQ3NR, EQ6) in like manner.

The FORTRAN 77 compiler may be known by names other than **f77** (**fortran** on Alliant machines, **cf77** on Crays). On some systems, the **ranlib** utility is built into **ar.** If you have a UNIX operating system and don't seem to have **ranlib** (you can find out by entering **which ranlib**), this is probably the case.

The following sequence is similar but produces a library file that is suitable for building an executable that can be used in conjunction with a symbolic debugger (e.g., **dbx**):

```
f77 -g -c *.f
ar rc eqlib.dbg.a *.o
ranlib eqlib.dbg.a
```

Here the "**-g**" in the compiler call instructs the compiler to include symbol tables in the "**.o**" files. The library has been named **eqlib.dbg.a** to indicate that it contains code that has been set up to support debugging.

The following sequence produces a library file for producing optimized (faster running) code:

```
f77 -O -c *.f
ar rc eqlib.opt.a *.o
ranlib eqlib.opt.a
```

Here the "**-O**" in the compiler call instructs the compiler to produce optimized code. Actually, the form "**-O**" is merely typical of how optimization is invoked. Some compilers operate differently, and the local system documentation should be consulted for details. Here the library has been named **eqlib.opt.a** to indicate that it contains optimized code.

Compiling with optimization flags may be dangerous, and the responsibility of doing so rests entirely with the user. It is recommended that one first install EQ3/6 without optimization, run all the test inputs provided with the package and also any inputs representing problems of particular local interest, save the outputs, and then compare outputs for the same problems run with optimized code. Optimized code sometimes produces small, usually negligible differences in the results. This is usually due to differences in the order of floating point operations in conjunction with rounding or truncation. In comparing, the user should only be concerned if significant differences are apparent in those results corresponding to physical parameters. Relatively large differences may be apparent in non-physical parameters, such as the calculated charge imbalance.

On the Alliant FX/80, the "**-O**" optimization flag leads to trouble with the EQLIB modules **newton.f** and **qsort.f**, and the EQ6 routine **path.f**. The compiler fails on the latter routine, apparently due to some internal table overflow. The EQLIB routine **qsort.f** performs a sort; optimization appears to defeat the sorting algorithm. Optimization of **newton.f** and **qsort.f** leads to differences in the output of the EQ3NR and EQ6 codes. Output does not appear to be incorrect, but some outputs are missing (appearing as blanks and zeros) or are out of the expected order. These problems may occur on other systems (but they do not appear on Sun SPARCstations).

On Cray computers with the **cft77** compiler (you may get this if you call **cf77**), it may be necessary to turn off all optimization to obtain correct results. See your system administrator for assistance. One known problem occurs when loops are optimized. The compiler looks to treat certain variables as constants within a loop, storing the entering values in cache memory and using these cache copies in assignment statements. The problem is that in some cases these variables are changed within the loop in subroutine calls. The main memory value is correct, but the cache copy is not.

Two routines specific to UNIX systems are referenced in EQ3/6 source code files. One is **etime**, which obtains the cpu and user times so that these can be written on code output. It is referenced once in each of the main programs for EQPT, EQ3NR, and EQ6 (the files **eqpt.f**, **eq3nr.f**, and **eq6.f**, if the "**.fsc**" or "**.src**" files have been split). These calls can be commented out, in which case zero values are likely to be printed. The other routine specific to UNIX is **fdate**. It is referenced once in each of the EQLIB modules **timdat.f** and **gtime.f**. It obtains the clock time and the date (which are then stamped on code output). This routine is called differently on some UNIX systems; see comments in the calling routines if you have a UNIX system but seem to have problems with this. The calls to this routine could be commented out, in which case blanks or zeros would likely be written on code output in the corresponding places.

Places in the source codes with UNIX-dependent code can be located by searching for the string "UNIX_DEPENDENT_CODE". The string "VAX_DEPENDENT_CODE" appears in places, too, with hints on appropriate modifications for VAX machines. We are not able to provide the exact equivalent coding for such machines. The changes required for VAXes are the same whether the operating system is ULTRIX or VMS, because ULTRIX does not provide for the usual UNIX calls for obtaining run statistics and the time and date.

Once past the compilation and library making stages, executable files for the codes in the EQ3/6 package may be created. In the case of EQ6, "**.fsc**" files, and a UNIX operating system, this may be done in the **eq6** subdirectory by entering:

```
f77 -o eq6R119 eq6.a ../eqlib.a
```

where **eq6R119** will be the name of the executable code. In this example, the stage number has been put into the name of the executable file. However, it could be given other names, such as **eq6.exe** or even **eq6** (be careful, because this may also be used as a directory name). On some systems, it may be necessary to modify the command by inserting the name of the object module of the main program before the names of the library files. The above example is then changed to:

```
f77 -o eq6R119 eq6.o eq6.a ../eqlib.a
```

If the "**.o**" file for the main program is needed, it can be extracted if necessary from the "**.a**" file containing it by using **ar** as in the following example:

```
ar x eq6.a eq6.o
```

An executable suitable for use with a symbolic debugger may be created by entering for example:

```
f77 -g -o eq6R119.dbg eq6.o eq6.dbg.a ../eqlib.dbg.a
```

An optimized executable may be created by entering for example:

```
f77 -o eq6R119.opt eq6.o eq6.opt.a ../eqlib.opt.a
```

If one is installing EQ3/6 on a VAX, it is critical that one compile all of the codes in the package with the G_FLOATING option turned on. It does not matter if the operating system is VMS or ULTRIX. This is necessary to obtain the floating point representation required for proper operation of the package elements. If one fails to do this, the codes are likely to have all sorts of problems.

Before running EQ3NR or EQ6, one must make a set of **data1** files to match the **data0** data files included in the export package. To do this, one must run the EQPT code on each such file. The simplest way to do this on a UNIX system is to run the shell script **runeqpt** that is provided in the export package. The directories in which the EQPT executable file and the **data0** files are located must be correctly specified in this script. See the internal documentation in the script. It may be necessary to modify the script to make it work properly on your system. To process all five data files, enter:

```
runeqpt R10 all
```

(this assumes that **runeqpt** is in the current directory or is in a directory in your search path). To process just the **com** file, for example, enter:

```
runeqpt R10 com
```

The script renames all of the files produced, incorporating the data file key and stage number of each **data0** file processed. For example, the **data1** file for **data0.com.R10** will be named **data1.com.R10**, and the **slist** file will be named **slist.com.R10.** Naming the **data1** files in this manner facilitates running EQ3NR and EQ6 under the shell script **runeq36,** which will be described later in this chapter.

EQPT can also be run without using the **runeqpt** script. A key point is that EQPT expects the **data0** file to be named "**data0**", and it produces a file called **data1.** The following command sequence (which is illustrated for the case of the **com** file) works on many UNIX operating systems and illustrates in detail the major action of the **runeqpt** script:

```
ln -s data0.com.R10 data0
eqptR80
mv data1 data1.com.R10
rm data0
mv slist slist.com.R10
```

(this assumes that all of the referenced files have been copied into the current directory). Here **eqptR80** is the name of the executable file for EQPT. If the "**ln -s**" command does not work, delete the "**-s**" part. The **slist** (species list) file is saved by renaming it to incorporate the specific identifiers of the **data0** file. This file is a useful reference when preparing **input** files for EQ3NR and EQ6. The process should then be repeated with the other **data0** files, replacing **com.R10** by the corresponding strings in their full names which provide specific identification.

EQ3/6 data files from versions predating version 7.0 are incompatible with the codes in the present package owing to changes in the data file structure and formats. If a user has extensively modified such files and desires to continue using them with the codes in the present version, it may be possible to do so by modifying them to match the new structure and formats, using the data files included in the present version as guides.

The simplest way to EQ3NR or EQ6 on a UNIX system is to run the shell script **runeq36** that is provided in the export package. The directories in which the EQ3NR and EQ6 executable files and the **data1** files are located must be correctly specified in this script. See the internal documentation in the script. It may be necessary to modify the script to make it work properly on your system. You must call **runeq36** as **runeq3** to run EQ3NR; you must call it as **runeq6** to run EQ6. You must therefore create **runeq3** and **runeq6** as symbolic links to or copies of **runeq36.** The symbolic links may be created by entering:

```
ln -s runeq36 runeq3

ln -s runeq36 runeq6
```

Be sure to put the directory containing **runeq3** and **runeq6** in your search path. You can then run EQ3NR, for example using the **input** file **rwtst.3i** with the **com** data file, by entering:

```
runeq3 com rwtst.3i
```

EQ6 can be run in similar manner. For example, to run EQ6 on the **input** file **microhcl.6i** with the **com** input file, enter:

```
runeq6 com microhcl.6i
```

In the case of either code, multiple **input** files may be specified in one command. However, only one data file may be specified.

The **runeq36** shell script requires that all EQ3NR **input** files end in "**.3i**" and that all EQ6 **input** files end in "**.6i**". The script renames all files produced in processing each **input** file, adding an appropriate suffix to the root name of the **input** file. For example, EQ3NR **output** files are assigned a "**.3o**" suffix, EQ6 **output** files a "**.6o**" suffix. Thus the **output** file corresponding to the input file **rwtst.3i** is named **rwtst.3o**. The pickup files are similarly assigned suffixes of "**.3p**" or "**.6p**". The EQ6 **tab** file is assigned a suffix of "**.6t**". The EQ6 **tabx** file is assigned a suffix of "**.6tx**".

To run EQ3NR or EQ6 without the **runeq36** shell script, you must provide an **input** file (known to the code as **input**) and a **data1** file (known to the code as "**data1**"). Each code produces an **output** file and a **pickup** file. EQ6 also produces **tab** and **tabx** files. The **tab** file is a set of output tables, suitable for supporting postprocessor graphics via local utilities; **tabx** is a "scrambled" "**tab**" file which may be discarded or saved for input to a pickup run.

In general, EQ3NR **input** files are identified by a "**.3i**" suffix, EQ3NR output files by "**.3o**" and EQ3NR **pickup** files by "**.3p**". EQ6 **input** files are identified by a "**.6i**" suffix, EQ6 **output** files by "**.6o**", EQ6 **pickup** files by "**.6p**", EQ6 tab files by "**.6t**", and EQ6 **tabx** files by "**.6tx**".

The preferred method of allowing the codes to know these by the required names is via symbolic links defined before code execution; for example, one could run EQ3NR using the following sequence (assuming all of the referenced files have been copied into the current directory):

```
ln -s swtst.3i input
ln -s output swtst.3o
ln -s pickup swtst.3p
eq3nrR124
```

Alternatively, one could use the following sequence to obtain the same results:

```
ln -s swtst.3i input
eq3nrR124
mv output swtst.3o
mv pickup swtst.3p
```

If symbolic links are not available, hard links or copies are also possible. The above example in this case then becomes:

```
cp swtst.3i input
eq3nrR124
rm input
mv output swtst.3o
mv pickup swtst.3p
```

Similar command sequences can be used to run EQ6. Users who are running EQ3NR and EQ6 on UNIX-based systems may find it convenient to incorporate such command sequences in shell scripts.

Four utility codes which perform input file conversion are also provided in the export package. CON3NF converts an EQ3NR input file in "W" format to "D" format. CON6NF converts and EQ6 input file in "W" format to "D" format. CON3IF and CON6IF respectively convert old EQ3NR and EQ6 **input** files corresponding to versions 6.0 (3245.0288) and 6.1 (3245.0888) which are all in "W" format to "D" format files compatible with version 7.0. The "**.fsc**" files for these utilities are fully self-contained. These may be compiled and linked/loaded in one step, illustrated by:

```
ln -s con3nfR03.fsc con3nf.f
f77 -o con3nf con3nf.f
rm con3nf.f
```

Note that the compiler will not accept a source code file that ends in "**.fsc**". This is circumvented here by making a symbolic link, which is deleted after compilation and loading/linking. Note that the source file is not split; the "**.f**" file contains several modules. These instructions produce an executable file called **con3nf.** It can be executed as follows. Suppose one wishes to convert the input file **swmaj.3i** from "W" to "D" format. One may then enter (assuming all of the referenced files have been copied into the current directory):

```
ln -s swmaj.3i input
con3nf
rm input
mv newin swmaj.3iN
```

Note that CON3NF expects the input file to be converted to be known as **input.** The converted input file created by CON3NF is initially known as **newin.** The "conversion" does not destroy the original input file. To distinguish the new one, it has been renamed **swmaj.3iN** in the above command sequence. The three related utilities operate analogously and may be installed and run in analogous manner.

A good number of examples of EQ3NR and EQ6 **input** files in "W" format are included in the export package. These may be converted to "D" format using CON3NF and CON6NF. A few examples of **input** files in "D" format are also included in the export package. In a few cases, the corresponding **output** files (ending in "**.3o**" and "**.6o**") are also provided. It is recommended that installation be completed by running the test cases corresponding to these **output** files and com-

paring the results. In general, exact matches are to be expected only if the local computer and its operating system match that used to create the examples transmitted as part of the export package, and then only if the compiler options are also identical. Results corresponding to physically meaningful parameters should be identical to within about 1 part in 10^6 for EQ3NR and EQ6, matching the default convergence tolerances employed in these codes. For the other codes, such results should agree to within 1 part in 10^{12} (about 10^3 times the **real*8** floating point epsilon).

It is recommended that EQLIB, EQPT, EQ3NR, and EQ6 be maintained after installation in the same directory format, keeping the split "**.f**" files and library "**.a**" files in order to facilitate future maintenance. Such maintenance might be required during the installation process itself, if problems are found in running the test cases.Also, recommended changes may be distributed by LLNL in the form of updated or additional files named **FIXES1.txt**, **FIXES2.txt**, etc.

If it is necessary or desirable to change the dimensioning of EQPT, EQ3NR, or EQ6, this may be accomplished by changing the relevant dimensioning parameters. These are defined in PARAMETER statements, which may be found by pattern searching. For source codes which are maintained in the "**.inc**" and "**.src**" format, the majority of dimensioning parameters are collected in a single INCLUDE file for each code, known as **eqtpar.h** for EQPT, **eq3par.h** for EQ3NR, and **eq6par.h** for EQ6. Some dimensioning parameters are defined in the INCLUDE file **eqlpar.h**, which belongs to the EQLIB library. Some parameters may also be set in individual source code modules, especially the main programs.

In the case of files in "**.fsc**" format, the contents of INCLUDE files have been pre-stuffed into the source code modules. Hence, the PARAMETER statements will be largely repeated among the various modules. In this case, one is largely reduced to pattern matching. On a UNIX system, the **grep** utility is useful in listing occurrences, and the **sed** utility is probably the most convenient tool for making the actual changes.

If it is desired to down-size EQ3NR and EQ6, the most obvious dimensioning parameters to reduce are the following (the values in EQ3NR and EQ6 are identical in version 7.0x and are given in parentheses; these values are unchanged in version 7.1):

nstpar: the maximum number of aqueous species (800)

nmtpar: the maximum number of pure minerals (850)

nctpar: the maximum number of chemical elements (100)

nsqpar: the maximum number of basis species (200)

If this should not be sufficient, a second group is composed of the following:

ngtpar: the maximum number of gas species (80)

nxtpar: the maximum number of solid solutions (50)

iktpar: the maximum number of end member components per solid solution (20).

If you are running the EQ3/6 codes on a SUN SPARCstation, you may see the following at the end of a run:

> **normal exit Warning: the following IEEE floating-point arithmetic exceptions occurred in this program and were never cleared: Inexact; Underflow;**

This occurs because the codes do not implement the system call **f77_floatingpoint.** As long as you see "normal exit," don't worry about it. If it really bothers you, implement **f77_floatingpoint** yourself.

Acknowledgments

Thanks are due to many individuals for their contributions in one form or another to the development of EQ3/6. Among these are Roger Aines, Carol Bruton, Bill Bourcier, Manny Clinnick, Paul Cloke, Stephanie Daveler, Joan Delany, Don Emerson, Mandy Goldner, Bob Herrick, Dana Isherwood, Ken Jackson, Jim Johnson, Suzanne Lundeen, Bill McKenzie, Judith Moody, Miki Moore, Ignasi Puigdomenech, Larry Ramspott, Terry Steinborn, and Brian Viani. The author wishes to also acknowledge stimulating conversations with Craig Bethke, Harold Helgeson, Jerry Kerrisk, Bill Murphy, Neil Plummer, Mark Reed, and many others too numerous to list here. I thank Brian Viani for his technical review of this report, which led to a number of improvements. I also thank Jim Blink for his comments on the manuscript. In addition, I thank the users of EQ3/6 for years of encouragement, many suggestions, and calling various bugs to my attention.

References

Allison, J. D., Brown, D. S., and Novo-Gradac, K. J., 1991, MINTEQA2/PRODEFA2, A Geochemical Assessment Model for Environmental Systems: Version 3.0 User's Manual: EPA/600/3-91/021, Environmental Protection Agency, Athens, Georgia.

ANSI, 1978,Progamming Language FORTRAN: ANSI X3.9-1978, ISO 1539-1980 (E): American National Standards Institute, New York.

Baes, C. F., Jr., and Mesmer, R. E., 1976, The Hydrolysis of Cations: Wiley Interscience, New York.

Ball, J. W., Jenne, E. A., and Cantrell, M. W., 1981, WATEQ3- A Geochemical Model with Uranium Added: United States Geological Survey Open-File Report 81-1183, United States Geological Survey, Menlo Park, California.

Ball, J. W., Jenne, E. A., and Nordstrom, D. K., 1979, WATEQ2: A computerized chemical model for trace and major element speciation and mineral equilibria for natural water, *in* Jenne, E. A., editor, Chemical Modeling in Aqueous Systems, American Chemical Society Symposium Series, v. 93, American Chemical Society, Washington, D. C., p. 815-835.

Ball, J. W., Nordstrom, D. K., and Jenne, E. A., 1980, Additional and Revised Thermochemical Data and Computer Code for WATEQ2--A Computerized Chemical Model for Trace and Major Element Speciation and Mineral Equilibria of Natural Waters: United States Geological Survey Water Resources Investigations 78-116, United States Geological Survey, Menlo Park, California.

Ball, J. W., and Nordstrom, D. K., 1991, User's Manual for WATEQ4F, with Revised Thermodynamic Data Base and Test Cases for Calculating Speciation of Major, Trace, and Redox Elements in Natural Waters: United States Geological Survey Open-File Report 91-183, United States Geological Survey, Menlo Park, California.

Ball, J. W., Nordstrom, D. K., and Zachmann, D. W., 1987, WATEQ4F- A Personal Computer FORTRAN Translation Of The Geochemical Model WATEQ2 With Revised Data Base: United States Geological Survey Water Resources Investigations 87-50, United States Geological Survey, Menlo Park, California.

Bassett, R. L., 1977, The Geochemistry of Boron in Thermal Waters: Ph.D. thesis, Stanford University, Stanford, California.

Bassett, R. L., and Melchior, D. C., 1990, Chemical modeling of aqueous systems: An overview, *in* Melchior, D. C., and Bassett, R. L., editors, Chemical Modeling of Aqueous Systems II, American Chemical Society Symposium Series, v. 416, American Chemical Society, Washington, D. C., p. 1-14.

Berner, R. A., 1971, Principles of Chemical Sedimentology: McGraw-Hill, New York.

Blum, A. E., and Lasaga, A. C., 1991, The role of surface speciation in the dissolution of albite: Geochimica et Cosmochimica Acta, v. 55, p. 2193-2201.

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.

Davies, C. W., 1962, Ion Association: Butterworths, London.

Delany, J. M., and Lundeen, S. R., 1991, The LLNL Thermochemical Data Base- Revised Data and File Format for the EQ3/6 Package: UCID-21658, Lawrence Livermore National Laboratory, Livermore, California.

Denbigh, K., 1971, The Principles of Chemical Equilibrium: The University Press, Cambridge, United Kingdom, 494 p.

Drever, J. I., 1982, The Geochemistry of Natural Waters: Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Felmy, A. R., Girvin, D. C., and Jenne, E. A., 1984, MINTEQ- A Computer Program for Calculating Aqueous Geochemical Equilibria: EPA-600/3-84-032, United States Environmental Protection Agency, Athens, Georgia.

Freeze, R. A., and Cherry, J. A., 1979, Groundwater: Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Garrels, R. M., and Christ, C. L., 1965, Solutions, Minerals and Equilibria: Freeman, Cooper and Company, San Francisco.

Grenthe, I., Lemire, R. J., Muller, A. B., Nguyen-Trung, C., and Wanner, H., 1989, NEA-TDB Chemical Thermodynamics of Uranium, OECD-Nuclear Energy Agency, Saclay, France (draft manuscript; NNA.900815.0013).

Grenthe, I., Fuger, J., Lemire, R. J., Muller, A. B., Nguyen-Trung, C., and Wanner, H., editors, 1992, Chemical Thermodynamics of Uranium: Elsevier Science Publishing Company, New York.

Harned, H. S., and Owen, B. B., 1958, The Physical Chemistry of Electrolyte Solutions, 3rd Edition: ACS Monograph 137, Reinhold, New York.

Harvie, C. E., Møller, N., and Weare, J. H., 1984, The prediction of mineral solubilities in natural waters: the Na-K-Mg-Ca-H-Cl-SO_4-OH-HCO_3-CO_3-CO_2-H_2O system to high ionic strengths at 25°C: Geochimica et Cosmochimica Acta, v. 48, p. 723-751.

Helgeson, H. C., 1969, Thermodynamics of hydrothermal systems at elevated temperatures and pressures: American Journal of Science, v. 267, p. 729-804.

Helgeson, H. C., Delaney, J. M., Nesbitt, H. W., and Bird, D. K., 1978, Summary and critique of the thermodynamic properties of rock-forming minerals: American Journal of Science, v. 278-A, p. 1-229.

Helgeson, H. C., and Kirkham, D. H., 1974a, Theoretical prediction of the thermodynamic behavior of aqueous electrolytes at high pressures and temperatures: I. Summary of the thermodynamic/electrostatic properties of the solvent: American Journal of Science, v. 274, p. 1089-1198.

Helgeson, H. C., and Kirkham, D. H., 1974b, Theoretical prediction of the thermodynamic behavior of aqueous electrolytes at high pressures and temperatures: II. Debye-Hückel parameters for activity coefficients and relative partial-molal properties: American Journal of Science, v. 274, p. 1199-1261.

Helgeson, H. C., and Kirkham, D. H., 1976, Theoretical prediction of the thermodynamic behavior of aqueous electrolytes at high pressures and temperatures: IV. Calculation of activity coefficients, osmotic coefficients, and apparent molal and standard and relative partial molal properties to 600°C and 5 kb: American Journal of Science, v. 281, p. 1249-1516.

Helgeson, H. C., Kirkham, D. H., and Flowers, G. C., 1981, Theoretical prediction of the thermodynamic behavior of aqueous electrolytes at high pressures and temperatures: III. Equation of state for aqueous species at infinite dilution: American Journal of Science, v. 276, p. 97-240.

INTERA Environmental Consultants, Inc., 1983, Geochemical Models Suitable for Performance Assessment of Nuclear Waste Storage: Comparison of PHREEQE and EQ3/EQ6: ONWI-473, Office of Nuclear Waste Isolation, Battelle Project Management Division, Columbus, Ohio.

Jackson, K. J., Wolery, T. J., Bourcier, W. L., Delany, J. M., Moore, R. M., Clinnick, M. L., and Lundeen, S. R., 1988, MCRT User's Guide and Documentation: UCID-21406, Lawrence Livermore National Laboratory, Livermore, California.

Jenne, E. A., editor, 1979, Chemical Modeling in Aqueous Systems, American Chemical Society Symposium Series, v. 93, American Chemical Society, Washington, D. C. p. 857-892.

Jenne, E. A., 1981, Geochemical Modeling: A Review: PNL-3574, Battelle Pacific Northwest Laboratory, Richland, Washington.

Johnson, J. W., and Norton, D., 1991, Critical phenomena in hydrothermal systems: State, thermodynamic, electrostatic, and transport properties of H_2O in the critical region: American Journal of Science, v. 291, p. 541-648.

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., and Barnes, I., 1973, SOLMNEQ: Solution-Mineral Equilibrium Computations: United States Geological Survey Computer Contributions Publication 215-899.

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.

Knauss, K. G., Wolery, T. J., and Jackson, K. J., 1990, A new approach to measuring pH in brines and other concentrated electrolytes: Geochimica et Cosmochimica Acta, v. 54, p. 1519-1523.

Knauss, K. G., Wolery, T. J., and Jackson, K. J., 1991, Reply to comment by R. E. Mesmer on "A new approach to measuring pH in brines and other concentrated electrolytes": Geochimica et Cosmochimica Acta, v. 55, p. 1177-1179.

Krauskopf, K., 1967, Introduction to Geochemistry: McGraw-Hill, New York.

Mattigod, S. V., and Sposito, G., 1979, Chemical modeling of trace metal equilibria in contaminated soil solutions using the computer program GEOCHEM, *in* Jenne, E. A., editor, Chemical Modeling in Aqueous Systems, American Chemical Society Symposium Series, v. 93, American Chemical Society, Washington, D. C., p. 837-856.

McDuff, R. E., and Morel, F. M. M., 1973, Description and Use of the Chemical Equilibrium Program REDEQL2: Technical Report EQ-73-02, Keck Laboratory, California Institute of Technology, Pasadena, California.

McKenzie, W. F., Wolery, T. J., Delany, J. M., Silva, R. J., Jackson, K. J., Bourcier, W. L., and Emerson, D. O., 1986, Geochemical Modeling (EQ3/6) Plan, Office of Civilian Radioactive Waste Management Program: UCID-20864, Lawrence Livermore National Laboratory, Livermore, California.

Melchior, D. C., and Bassett, R. L., editors, 1990, Chemical Modeling of Aqueous Systems II, American Chemical Society Symposium Series, v. 416, American Chemical Society, Washington, D. C., p. 1-14.

Mesmer, R. E., 1991, Comments on "A new approach to measuring pH in brines and other concentrated electrolytes" by K. G. Knauss, T. J. Wolery, and K. J. Jackson: Geochimica et Cosmochimica Acta, v. 55, p. 1175-1176.

Metcalf, M., 1985, Effective FORTRAN 77: Clarendon Press, Oxford.

Morel, F., and Morgan, J. J., 1972, A numerical method for computing equilibria in aqueous chemical systems. Environmental Science and Technology, v. 6, p. 58-67.

Nordstrom, D. K., and Ball, J. W., 1984, Chemical models, computer programs, and metal complexation in natural waters, *in* Kramer, C. J. M., and Duinker, J. C., editors, International Symposium on Trace Metal Complexation in natural waters, Martinus Nijhoff/Dr. J. W. Junk Publishing Company, Amsterdam, p. 149-169.

Nordstrom, D. K. and Munoz, J. L., 1985, Geochemical Thermodynamics: Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, 477 p.

Nordstrom, D. K., Plummer, L. N., Langmuir, D., Busenberg, E., May, H. M., Jones, B. F., and Parkhurst, D. L., 1990, Revised chemical equilibrium data for major water-mineral reactions and their limitations, *in* Melchior, D. C., and Bassett, R. L., editors, Chemical Modeling of Aqueous Systems II, American Chemical Society Symposium Series, v. 416, American Chemical Society, Washington, D. C., p. 398-413.

Nordstrom, D. K., Plummer, L. N., Wigley, T. M. L., Wolery, T. J., Ball, J. W., Jenne, E. A., Bassett, R. L., Crerar, D. A., Florence, T. M., Fritz, B., Hoffman, M., Holdren, G. R., Jr., Lafon, G. M., Mattigod, S. V., McDuff, R. E., Morel, F., Reddy, M. M., Sposito, G., and Thrailkill, J., 1979, A comparison of computerized chemical models for equilibrium calculations in aqueous systems, *in* Jenne, E. A., editor, Chemical Modeling in Aqueous Systems, American Chemical Society Symposium Series, v. 93, American Chemical Society, Washington, D. C. p. 857-892.

Nordstrom, D. K., Valentine, S. D., Ball, J. W., Plummer, L. N., and Jones, B. F., 1984, Partial Compilation And Revision Of Basic Data In The WATEQ Programs: United States Geological Survey Water-Resources Investigations Report 84-4186, United States Geological Survey, Menlo Park, California.

Papelis, C., Hayes, K. F., and Leckie, J. O., 1988, HYDRAQL: A Program for the Computation of Chemical Equilibrium Composition of Aqueous Batch Systems Including Surface-Complexation Modeling of Ion Adsorption at the Oxide/Solution Interface: Technical Report Number 306, Department of Civil Engineering, Stanford University, Stanford, California.

Parkhurst, D. L., Thorstenson, D. C., and Plummer, L. N., 1980, PHREEQE - A Computer Program for Geochemical Calculations: United States Geological Survey Water Resources Investigations 80-96.

Perkins, E. H., Kharaka, Y. K., Gunter, W. D., and DeBraal, J. D., 1990, Geochemical modeling of water-rock interactions using SOLMINEQ.88, *in* Melchior, D. C., and Bassett, R. L., editors, Chemical Modeling of Aqueous Systems II, American Chemical Society Symposium Series, v. 416, American Chemical Society, Washington, D. C., p. 117-127.

Pitzer, K. S., 1973, Thermodynamics of electrolytes - I. Theoretical basis and general equations: Journal of Physical Chemistry, v. 77, p. 268-277.

Pitzer, K. S., 1975, Thermodynamics of electrolytes. V. Effects of higher-order electrostatic terms: Journal of Solution Chemistry, v. 4, p. 249-265.

Pitzer, K. S., 1979, Theory: ion interaction approach, *in* Pytkowicz, R. M., editor, Activity Coefficients in Electrolyte Solutions, CRC Press, Boca Raton, Florida, p. 157-208.

Pitzer, K. S., 1987, Thermodynamic model for aqueous solutions of liquid-like density, *in* Carmichael, I. S. E., and Eugster, H. P., editors, Thermodynamic Modeling of Geological Materials: Minerals, Fluids and Melts, Reviews in Mineralogy, v. 17, Mineralogical Society of America, Washington, D. C., p. 97-142

Pitzer, K. S., and Brewer, L., 1961, Thermodynamics, second edition, revision of Lewis, G. N., and Randall, M.: McGraw-Hill Book Company, New York, 723 p.

Plummer, L. N., Jones, B. F., and Truesdell, A. H., 1976, WATEQF - A FORTRAN IV Version of WATEQ, A Computer Program for Calculating Chemical Equilibrium of Natural Waters: United States Geological Survey Water Resources Investigations 76-13, United States Geological Survey, Reston, Virginia.

Plummer, L. N., and Parkhurst, D. L., 1990, Application of the Pitzer equations to the PHREEQE geochemical model, *in* Melchior, D. C., and Bassett, R. L., editors, Chemical Modeling of Aqueous Systems II, American Chemical Society Symposium Series, v. 416, American Chemical Society, Washington, D. C., p. 128-137.

Plummer, L. N., Parkhurst, D. L., Fleming, G. W., and Dunkle, S. A., 1988, A Computer Program Incorporating Pitzer's Equations for Calculation of Geochemical Reactions in Brines: United States Geological Survey Water-Resources Investigations Report 88-4153, 310 p.

Puigdomenech, I., and Emrén, A., 1990, Performance of EQ3/6, Phreeqe, and Solgaswater on Geochemical Path Calculations Involving Redox and Solid Phase Changes: SKI Technical Report 90:16, Swedish Nuclear Power Inspectorate.

Reed, M. H. 1977. Calculations of Hydrothermal Metasomatism and Ore Deposition in Submarine Volcanic Rocks with Special Reference to the West Shasta District, California: Ph.D. dissertation, University of California, Berkeley.

Reed, M. H., 1982, Calculation of Multicomponent Chemical Equilibria and Reaction Processes in Systems involving Minerals, Gases, and an Aqueous Phase: Geochimica et Cosmochimica Acta, v. 46, p. 513-528.

Reed, M. H., and Spycher, N. F., 1989, SOLTHERM: Data Base of EQuilibrium Constants for Aqueous-Mineral-Gas Equilibria: Department of Geological Sciences, University of Oregon, Eugene, Oregon.

Robinson, R. A., and Stokes, R. H., 1965, Electrolyte Solutions, second edition, revised: Butterworths, London.

Shock, E. L., and Helgeson, H. C., 1988, Calculation of the thermodynamic and transport properties of aqueous species at high pressures and temperatures: Correlation algorithms for ionic species and equation of state predictions to 5 kb and 1000°C: Geochimica et Cosmochimica Acta, v. 52, p. 2009-2036.

Shock, E. L., and Helgeson, H. C., 1989, Corrections to Shock and Helgeson (1988) *Geochimica et Cosmochimica Acta* **52**, 2009-2036: Geochimica et Cosmochimica Acta, v. 53, p. 215.

Shock, E. L., and Helgeson, H. C., 1990, Calculation of the thermodynamic and transport properties of aqueous species at high pressures and temperatures: Standard partial molal properties of organic species: Geochimica et Cosmochimica Acta, v. 54, p. 915-945.

Shock, E. L., Helgeson, H. C., and Sverjensky, D. A., 1989, Calculation of the thermodynamic and transport properties of aqueous species at high pressures and temperatures: Standard partial molal properties of inorganic species: Geochimica et Cosmochimica Acta, v. 53, p. 2157-2183.

Shock, E. L., Oelkers, E. H., Johnson, J. W., Sverjensky, D. A., and Helgeson, H. C., 1992, Calculation of the thermodynamic properties of aqueous species at high pressures and temperatures: Effective electrostatic radii, dissociation constants and standard partial molal properties to 1000°C and 5 kbar: Journal of the Chemical Society Faraday Transactions, v. 88, p. 803-826.

Silling, S. A., 1983, Final Technical Position on Documentation of Computer Codes for High-Level Waste Management: NUREG-0856, United States Nuclear Regulatory Commission, Washington, D. C.

Snoeyink, V. L., and Jenkins, D., 1980, Water Chemistry: John Wiley and Sons, New York.

Spycher, N. F., and Reed, M. H., 1989a, SOLVEQ: A Computer Program for Computing Aqueous-Mineral-Gas Equilibria (Revised Preliminary Edition): Department of Geological Sciences, University of Oregon, Eugene, Oregon.

Spycher, N. F., and Reed, M. H., 1989b, CHILLER: A Program for Computing Water-Rock Reactions, Boiling, Mixing and Other Reaction Processes in Aqueous-Mineral-Gas Systems (Revised Preliminary Edition): Department of Geological Sciences, University of Oregon, Eugene, Oregon.

Stumm, W., and Morgan, J. J., 1981, Aquatic Chemistry, second edition: John Wiley and Sons, New York.

Tanger, J. C., and Helgeson, H. C., 1988, Calculation of the thermodynamic and transport properties of aqueous species at high pressures and temperatures: Revised equations of state for the standard partial molal properties of ions and electrolytes: American Journal of Science, v. 288, p. 19-98.

Truesdell, A. H., and Jones, B. F., 1974, WATEQ, a computer program for calculating chemical equilibria of natural water: Journal of Research of the United States Geological Survey, v. 2, p. 233-248.

Westall, J. C., Zachary, J. L., and Morel, F. M. M., 1976, MINEQL, A Computer Program for the Calculation of Chemical Equilibrium Composition of Aqueous Systems: Technical Note 18, Ralph M. Parsons Laboratory for Water Resources and Environmental Engineering, Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Wolery, T. J., 1978, Some Chemical Aspects of Hydrothermal Processes at Mid-Ocean Ridges-A Theoretical Study: Ph.D. thesis, Northwestern University, Evanston, Illinois.

Wolery, T. J., 1979, Calculation of Chemical Equilibrium between Aqueous Solutions and Minerals: The EQ3/6 Software Package: UCRL-52658, Lawrence Livermore National Laboratory, Livermore, California.

Wolery, T. J., 1980, Chemical Modeling of Geologic Disposal of Nuclear Waste: Progress Report and a Perspective: UCRL-52748, Lawrence Livermore National Laboratory, Livermore, California.

Wolery, T. J., 1983, EQ3NR, A Computer Program for Geochemical Aqueous Speciation-Solubility Calculations: User's Guide and Documentation: UCRL-53414, Lawrence Livermore National Laboratory, Livermore, California.

Wolery, T. J., 1992, 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 Code 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.

Wolery, T. J., Jackson, K. J., Bourcier, W. L., Bruton, C. J., Viani, B. E., Knauss, K. G., and Delany, J. M., 1992, Current status of the EQ3/6 software package for geochemical modeling, *in* Melchior, D. C., and Bassett, R. L., editors, Chemical Modeling of Aqueous Systems II, American Chemical Society Symposium Series, v. 416, American Chemical Society, Washington, D. C., p. 104-116.

Appendix A. Source Code Conventions

The ease of portability, development, and maintenance of a FORTRAN code can be greatly improved by certain restrictions and conventions. This section describes those that have been adopted in developing the codes in the EQ3/6 software package. These conventions were adopted in the light of experience in both code development and transferring EQ3/6 to a variety of different machines. Most of the conventions that are described here could be usefully applied to other scientific codes.

This appendix is primarily intended to serve as reference material for users who, for whatever reason, desire to examine or make changes in the source code. The conventions here are a summary of the guidelines and requirements from document 033-NWMP-R 19.2 (EQ3/6), "Coding Standards for FORTRAN Computer Codes." This document was superseded by the issuance on October 18, 1990 of TIP-YM-10, "Documentation and Coding Standards for FORTRAN Programs." This issuance preceded the release of version 7.0 of EQ3/6 by about one month. The major difference between the two sets of conventions is that the older set was specific to EQ3/6 and thus contained a somewhat larger set of conventions. Nearly all of the source code conventions in the newer set were in the older set, in the same or very similar form.

The conventions described here are supplementary to the ANSI FORTRAN 77 standard (ANSI X3.9-1978; ANSI, 1978). Readers desiring a more concise and readable treatment of the subject are referred to Metcalf (1985). Readers are cautioned that the ANSI FORTRAN 77 standard is not a single standard, but consists of both a "full language" and a "subset language." Furthermore, the wording of specific items in the standard has not always been interpreted in the same way by all compiler developers. Beyond that, the ANSI FORTRAN 77 contains a number of deficiencies, some of which will be pointed out below.

The earliest versions of EQ3/6 were written in FORTRAN 66 (ANSI X3.9-1966). Some parts of version 7.0 have not been fully updated to conform with the best programming practices under FORTRAN 77. However, they are still consistent with the FORTRAN 77 standard, which was designed to be upwardly compatible with FORTRAN 66.

The supplementary source code conventions described here have also evolved. The reasons for adopting these conventions vary, but generally deal with minimizing real or potential problems regarding portability, development, and maintenance of the software. In many cases, the rationale behind a convention is obvious and one may wonder why the convention (or some reasonable alternative) was not incorporated into the ANSI standard. The answer is generally that the desire for upward compatibility has tended to be overwhelming. In other cases, the rationale behind a convention may not be obvious. In such cases, the basis of the convention is usually to avoid pitfalls associated with bugs or other deficiencies commonly found in compilers and symbolic debuggers.

One of the major problems in programming is how to handle data structures. More properly, this problem is one of data flow, or how to make the data structures available to the various modules (main program and subroutines) which must use various elements of them. In FORTRAN, the traditional answer to this problem has been the COMMON block. EQ3/6 still has many COMMON blocks; however, there has been a movement in more recent development toward the utilization of calling sequences as a means of handling the data flow problem.

When a COMMON block is used as the means of data flow, a copy is inserted into each module requiring relevant data. It is critical that each copy be identical. Otherwise, data may be inadvertently overwritten or put in the wrong place. This problem can be addressed by having one copy of the COMMON block (or a related set of COMMON blocks) in a special file and putting a reference to that file in each of the source code modules requiring that COMMON block. Such files are usually called INCLUDE files.

The usage of INCLUDE files is not addressed by the ANSI FORTRAN 77 standard. In the UNIX operating system, such a file is designated by the suffix, "**.h.**" A statement of the form:

include '../inc/eqlpar.h'

is placed in the source code. In this example, the INCLUDE file is **eqlpar.h** and "**../inc**" is the path name designating a directory named **inc** that is parallel to the one containing the source code module. A similar system is available under the VMS and ULTRIX operating systems on VAX machines. However, path names are not permitted; thus, the INCLUDE file must be in the same directory as the source code which references it. Some systems do not provide for the use of INCLUDE files. To avoid portability problems, EQ3/6 is normally distributed with the contents of the INCLUDE files "pre-stuffed" into the source code.

Ideally, all the dimensions of an array passed in calling sequences should also be passed for use in dimensioning this array in the called routine, although FORTRAN 77 allows the lowest dimension to be represented in that routine by the indeterminate "*". Array dimensions in EQ3/6 are generally defined by "dimensioning parameters," which are set in PARAMETER statements. Some systems allow formal parameters to be passed in calling sequences; others do not. To accommodate the latter, "dimensioning variables" can be created to pass in calling sequences. Dimensioning variables are much used in EQ3/6. However, there is still much usage of the "*" methodology. This usage causes some difficulties in the use of symbolic debuggers, and also precludes array bound checks in executable code compiled to make such checks.

The source code conventions used in developing version 7 of EQ3/6 are summarized as follows:

1. Module names, INCLUDE file names, and COMMON block names should be unique within a software package.

2. Only the generic forms of the FORTRAN 77 intrinsic functions should be used. For example, use **int** instead of **ifix** to convert a floating point variable to an integer.

3. Module names should not match those of any of the FORTRAN 77 intrinsic functions.

4. Variable and subroutine names are restricted to six characters. This is all the ANSI FORTRAN standard requires and all one gets on many systems. Usage of longer names is more convenient in many ways, but dangerous as far as portability is concerned. If one uses longer names, the compiler is only guaranteed to recognize the first six characters. Thus, two variables **column1** and **column2** would each appear to the compiler as the same variable, **column.**

5. The following type conventions generally apply to the first letter of a variable name:

(a) **i, j,** and **k: integer**

(b) **l** and **m: real*8**; Note that **real*8** is not officially recognized by the ANSI FORTRAN 77 standard. It corresponds to **real l** on 64-bit machines and **double precision** on 32 bit-machines.

(c) **q: logical**

(d) **u: character** (variable character length).

This convention is highly specific to EQ3/6. It is motivated by the desire to instantly know the type of a variable from its name. This convention is strongly associated with the "standard" EQ3/6 IMPLICIT statement:

```
      implicit logical(q),integer(i,j,k,n),real*8(a-h,l,m,o,p,r-t,v-z),
     $ character*8(u)
```

There are exceptions to this within some modules which deal with no floating point numbers. In these, the letters that normally imply type **real*8** instead imply type **integer.** Note that no **real*4** floating point variables are used in EQ3/6. Note that character lengths, if other than 8, must be set elsewhere in CHARACTER statements.

6. Character lengths of character variables should be multiples of 8, unless the character length is less than 8. This is especially important in the case of character variable arrays. This restriction addresses a common defect in FORTRAN compilers of misaddressing an element of such an array if the element does not start at a word boundary. With this convention, every element will start at a word boundary on a 32-bit or 64-bit machine.

7. Names of all types in EQ3/6 are restricted to lower case. Capitalization appears in the source code only in comments and in strings that are output by the software (e.g., table headers, error messages). Mostly this is an attempt to keep the software "single case." It can be converted to upper case if need be by various system utilities.

8. Arrays are dimensioned using dimensioning parameters which are defined in PARAMETER statements. This assists rapid redimensioning. Corresponding "dimensioning variables" are defined in assignment statements and should be passed in calling sequences whenever the corresponding arrays are passed. Dimensioning parameters appearing in COMMON blocks kept in INCLUDE files should also be defined in an INCLUDE file. For example, in the case of EQ3NR, this file is **eq3par.h**; in the case of EQ6, it is **eq6par.h.** Dimensioning parameters that do not appear in COMMON blocks should be defined in main programs, not subroutines. This requires that arrays dimensioned according to such parameters should also be defined in the main programs.

9. INCLUDE files should contain only COMMON blocks, PARAMETER declarations, and IMPLICIT statements.

10. Any software should contain its own version identification and write this on any output that it produces. Any supporting data file should also contain such identification, and any code using it should read this and also write it on the output produced.

11. Any code should write a time and date stamp on any output produced.

12. Comments should be used frequently in the source code. In particular, comments describing the function of a module and the principal data elements employed or affected should appear at the top of each source code module. Comments should also be used to note anything which is unusual or might appear to be unusual when examined at a later date. Comments should not be used to belabor the obvious, as in the following example:

```
c
c       Return to calling program.
c
        return
        end
```

13. All COMMON blocks pertaining to a code should appear in the main program. This facilitates symbolic debugging, because one can then always get the value of a COMMON block variable by accessing the main program's symbol table. For the same reason, any variable of a global nature not in a COMMON block should be defined in the main program.

14. Character variables should not be mixed with other kinds of variables in the same COMMON block. Some systems allow this; others do not.

15. COMMON blocks should be "aligned" to avoid problems on some systems. The character variables in a given common block should appear in order of decreasing character length. A COMMON block containing other kinds of variables should be arranged so that the variables appear in decreasing order of word length. This requires that **real*8** variables appear before **real*4, integer**, and **logical** variables.

16. Only one statement appears per line of source code. This makes the source code easier to read and reduces the chance of an error in development or maintenance if a programmer fails to note an extra statement. It may also help to avoid problems when running the code under a symbolic debugger.

17. Multiple assignment statements should not be used.

18. The usage of EQUIVALENCE statements is prohibited. Any other double usage of a variable should be avoided.

19. Character variables should not be specified in the archaic Hollerith notation.

20. All statement labels are numeric; none are alphabetic or alphanumeric. This is done to maintain consistency with the FORTRAN 77 standard. In order to make the source

code easy to follow, it is usually desirable to minimize the use of such labels. Some features of FORTRAN 77, such as DO loops, require the use of such labels. Unnecessary statement labels are commonly associated with GO TO statements.

21. The usage of GO TO statements should be minimized in order to make the source code easy to follow. In most instances, these can be avoided by the use of block IF (IF-THEN-ELSE) structures or by pulling blocks of code out into separate modules. There are occasions, however, when a well-placed GO TO statement makes the source code easier to follow.

22. Arithmetic IF statements should not be used. These are archaic.

23. No non-integer subscripts should be used.

24. Loops should be entered only at their beginning.

25. RETURN statements should not be used.

26. The ALTERNATE ENTRY feature should not be used.

27. The coding should not presume any specific initialization of variables by a loader/-linker.

28. Constants, particularly floating point constants, should not be specified in subroutine calling sequences. This prevents certain kinds of conflicts in converting to double precision (**real*8**); i.e.,use:

```
fvar=1.e-6
call subr(fvar)
```

not

```
call subr(1.e-6)
```

In the first example, if **fvar** is **real*8**, the assigned value if **real*4** will be converted to **real*8**, and the subroutine call will be correct. In the second case, an execution error will result because of a **real*4-real*8** mismatch.

29. To avoid programming errors, the character length should always be specified when comparing two character strings. If this is not done, the two strings are taken to be unequal if of unequal length, even if the contents are otherwise the same and the extra portion of the longer string consists of blanks. For example, it is desirable to use the construction:

```
if (uvar1(1:8) .eq. uvar2(1:8)) then
```

instead of:

```
if (uvar1 .eq. uvar2) then
```

30. To avoid possible portability problems, a character string which is to be compared with other character strings should be set by means of a DATA statement or by reading it from a file, rather than by specifying the string directly as a constant (some systems do not allow this). For example, it is desirable to use the construction:

```
data uendit/'endit.'
     ...
if (uvar(1:6) .eq. uendit(1:6)) then
```

instead of the more direct

```
if (uvar(1:6) .eq. 'endit') then
```

31. The unit numbers of the relevant files are always specified in reading and writing data. These unit numbers are specified in the source code by integer variables, not constants. This convention is followed primarily to minimize portability problems. Adapting the software to run on certain operating systems may require the user to change some of the unit numbers. This may be necessary if certain values are reserved for special files, such as the screen file. Using a variable to represent the unit number means that only the value of the variable need be changed, not every read or write instruction dealing with the corresponding file. Also, the variable name is usually adapted from the file name (e.g., **noutpt** for the **output** file), which makes the source code easier to understand.

Appendix B. Glossary of EQLIB Modules

EQLIB is a library of 138 modules which support EQPT, EQ3NR, and EQ6. The source code consists of the main program and a number of subroutines. Similar lists of modules are given in Appendix B in each of the User's Manuals for EQPT (Daveler and Wolery, 1992), EQ3NR (Wolery, 1992), and EQ6 (Wolery and Daveler, 1992). The modules are described as "**.f**" files, as this is how they are normally worked with under a UNIX operating system.

alters.f This module is called by the EQ3NR module **indatx.f** and the EQ6 module **indat1.f**. It executes the alter part of the **nxmod** alter/suppress options. It alters the *log K* values of reactions after they are read from the **data1** file. This is done before the reactions are rewritten by any basis switching.

autosw.f This module is called by the EQ3NR module **arrset.f** and the EQ6 module **pabssw.f**. It executes automatic basis switching.

bdmlx.f This module is called by the EQ3NR module **eq3nr.f**, the EQ6 module **indat1.f**, and fellow EQLIB module **swchlm.f**. It builds the **nmxi** and **nmxx** pointer arrays used to handle the μ_{ijk} parameters in Pitzer's equations.

bdslx.f This module is called by the EQ3NR module **eq3nr.f**, the EQ6 module **indat1.f**, and fellow EQLIB module **swchlm.f**. It builds the **nsxi** and **nsxx** pointer arrays used to handle the ${}^{S}\lambda_{ij}$ parameters in Pitzer's equations.

betae.f This is not a module, but a dummy external which refers to either the EQ3NR module **betas.f** or the EQ6 module **betaz.f**. This dummy external is called by the EQLIB routines **newton.f** and **nrstep.f**. The actual modules compute the Newton-Raphson residual functions.

betgam.f This module is called by fellow EQLIB module **ngcadv.f**. It finds the activity coefficient residual for an aqueous species with the largest magnitude.

chreal.f This module is called by fellow EQLIB module **rdtyp7.f**, by EQ3NR modules **getspc.f**, **getrdx.f**, **getss.f**, **rd3tds.f**, **rdtyp1.f**, and **rdtyp9.f**, and by EQ6 modules **rd6ff.f**, **rd6log.f**, **rd6new.f**, **rd6prs.f**, **rd6rea.f**, **rd6tol.f**, **rdmole.f**, and **rdrate.f**. It converts a character string to a **real*8** floating point number.

chrint.f This module is called by fellow EQLIB module **rdtyp8.f**, by the EQ3NR module **rdtyp9.f**, and by EQ6 modules **rd6new.f**, **rd6rea.f**, **rd6tol.f**, and **rdrate.f**. It converts a character string to an integer.

chump.f This module is called by fellow EQLIB module **eqlib.f**. It tests the machine epsilon and the floating point exponent range of the host computer (see **flpars.f**) to see if they are adequate for running EQ3/6. If they are not, an error message is written and the code terminates.

ckbndi.f This module is called by fellow EQLIB module **rdtyp8.f**, by the EQ3NR module **rdtyp9.f**, and by EQ6 modules **rd6new.f** and **rd6rea.f**. It checks an integer variable read from the **input** file in "D" format to see if it falls into a specified acceptable range.

ckbndr.f This module is called by fellow EQLIB module **rdtyp6.f**, by the EQ3NR modules **getrdx.f**, **getspc.f**, **rd3tds.f**, **rdtyp1.f**, and **rdtyp9.f**, and by EQ6 modules **rd6new.f** and **rd6tol.f**. It checks a **real*8** variable read from the **input** file in "D" format to see if it falls into a specified acceptable range.

ckpars.f This module is called by fellow EQLIB modules **getdes.f**, **rdastr.f**, **rdtyp6.f**, **rdtyp7.f**, and **rdtyp8.f**, by the EQ3NR modules **getrdx.f**, **getss.f**, **rd3tds.f**, **rdtyp1.f**, **rdtyp2.f**, **rdtyp4.f**, **rdtyp5.f**, and

rdtyp9.f, and by EQ6 modules **rd6ff.f, rd6log.f, rd6new.f, rd6prs.f, rd6rea.f, rd6sup.f, rd6tol.f, rdmole.f**, and **rdrate.f**. It checks a block of data read from the **input** file in "D" format to see if the number of variables found matches that which is expected.

dscram.f This module is called by the EQ6 module **eq6.f**. It orders the lines in the **tabx** file (the scrambled tab file), which was created during the run by the EQ6 module **wrtabx.f**. The lines belong to different tables, and are interleaved or "scrambled" on **tabx**. The table to which each line belongs is identified by the character in column 1. This module writes the tables sequentially to the **tab** file, removing the table identifier.

dsiplx.f This module is called by fellow EQLIB module **hpsat.f**. It uses a simplex algorithm to find the composition of a solid solution that maximizes its affinity to precipitate, given the composition of the aqueous phase.

echolk.f This module is called by the EQ3NR module **eq3nr.f** and the EQ6 module **echoz.f**. It writes species and reactions that are active in the current problem, along with the corresponding *log K* values, on the **output** file or, in some cases when called by **eq3nr.f**, the **rlist** file. Optionally, coefficients of the corresponding interpolating polynomials are also written.

elim.f This module is called by the EQ3NR modules **eq3nr.f** and **scripx.f** and the EQ6 module **indatz.f**.It rewrites reaction equations so that a specified auxiliary basis species is eliminated from all reactions except the one linking it with its corresponding strict basis variable. The polynomial coefficients for computing the equilibrium coefficients are recomputed accordingly.

elmdd.f This module is called by fellow EQLIB module **gelam.f**. It calculates the electrostatic second-order virial coefficient ($^{E}\lambda_{ij}$)and its derivative with respect to ionic strength for two ions of specified charge numbers. These quantities are needed to compute the "higher order electrostatic" terms in Pitzer's equations.

eqlib.f This module is called by the main program (**eqpt.f, eq3nr.f**, and **eq6.f**) of each code in the package. It provides EQLIB version identification to the calling code. It also gets the floating point parameters of the host computer, has them tested for adequacy, and sets a UNIX system flag if necessary to have the system not halt execution in case of floating point underflow.

evdata.f This module is called by the EQ3NR module **eq3nr.f** and the EQ6 modules **indatz.f** and **redatz.f**. It computes standard state thermodynamic properties as a function of temperature and pressure.

evdatc.f This module is called by fellow EQLIB modules **alters.f** and **evdata.f**. It computes a standard state thermodynamic property as a function of temperature from an interpolating polynomial whose coefficients are stored in a two-dimensional array. The second dimension of the array corresponds to a temperature range.

evdatk.f This module is presently not used. It is very similar to **evdatc.f**. The difference is that it evaluates an interpolating polynomial whose coefficients are stored in a three-dimensional array.

evdatr.f This module is called by fellow EQLIB module **evdata.f** and by the EQ3NR modules **arrset.f, eq3nr.f**, and **scripx.f**. It computes standard state thermodynamic data (logarithms of equilibrium constants) as functions of temperature from interpolating polynomials.

fbassw.f This module is called by the EQ3NR module **betas.f** and the EQ6 module **pabssw.f**. It finds candidates for basis switching in order to obtain the optimal basis set. This mechanism tends to associate with a given mass balance the aqueous species which makes the largest contribution to it.

fcopya.f This module is called by the EQ6 module **eq6.f**. It appends the contents of one file to another file. In EQ6 this routine is used to append the information from the **tabs** file to the **tab** file.

fdd.f This module is called by fellow EQLIB module **gcoeff.f**. It computes the Debye-Huckel functions f and f'.

fdomsp.f This module is called by the EQ3NR module **betas.f** and the EQ6 module **betaz.f**. It finds the species which makes the largest contribution to a specified mass balance.

flen.f This module is presently not used. It finds the length of a character string.

flpars.f This module is called by fellow EQLIB module **eqlib.f**. It calculates the **real*8** floating point parameters of the host computer, including the smallest positive number (**smpos**), the exponent range (**irang**), and the machine epsilon (**eps**).

gabar.f This module is presently not used. It computes a compositionally averaged ion size parameter.

gabswx.f This module is called by the EQ3NR module **arrset.f** and the EQ6 module **pabssw.f**. It supports automatic basis switching as a means of optimization prior to hybrid Newton-Raphson iteration. It resolves conflicts in the **ibswx** array, which contains data describing the candidate switches. This array could otherwise call for a given non-basis species to be switched with more than one species in the existing basis set.

gafscl.f This module is called by the EQ6 modules **indatz.f** and **pabssw.f**. It computes the array **cscale**, which contains affinity scaling factors. Affinity scaling is used when selecting a phase to precipitate from a set of supersaturated phases.

gbdot.f This module is called by fellow EQLIB module **gcoeff.f**. It calculates the activity coefficients of aqueous solute species using the B-dot equation and related approximations.

gbfac.f This module is called by the EQ3NR module **arrset.f** and the EQ6 module **ophelp.f**. It calculates the array **bfac**, which is used to make pre-Newton-Raphson optimization corrections. In the absence of a conflict, **bfac(kcol) = (beta(kcol) + 1.) ** efac(kcol)**. A conflict occurs when the same basis species dominates more than one mass balance. The optimization algorithm cannot be applied under these conditions, else oscillatory behavior occurs. If a conflict is found, **bfac(kcol)** is calculated as above for the mass balance with the greatest residual **beta(kcol)**; **bfac(kcol)** is set to unity for other mass balances involved in the conflict.

gcdrst.f This module is called by the EQ6 modules **indatz.f** and **pabssw.f**. It calculates the arrays **cdrst**, **cdrmt**, and **cdrgt**.

gcoeff.f This module is called by fellow EQLIB module **ngcadv.f**, the EQ3NR module **arrset.f**, and the EQ6 module **eq6.f**. It computes the activity coefficients of aqueous species. These vary according to the model specified by the user with the switch **iopg1**. After the activity coefficients are calculated, they may be normalized to make them consistent with a given *pH* convention (specified by **iopg2**).

gcscal.f This module is called by fellow EQLIB module **gcoeff.f**. It normalizes the activity coefficients of aqueous species to make them consistent with a chosen *pH* scale.

gcsts.f This module is called by the EQ3NR modules **arrset.f**, **eq3nr.f**, and **scripx.f**. It computes the stoichiometric factors which relate each non-basis species to the basis species. These factors appear in mass balance relations.

gdavie.f This module is called by fellow EQLIB module **gcoeff.f.** It calculates the activity coefficients of aqueous solute species using the Davies (1962) equation.

gdd.f This module is called by fellow EQLIB module **gslam.f.** It computes the $g(x)$ and $g'(x)$ functions used to compute the ${}^{S}\lambda_{ij}$ interaction parameters which appear in Pitzer's equations.

gdlgxw.f This module is called by the EQ3NR modules **arrsim.f** and **matrix.f** and the EQ6 module **matrxz.f.** It computes the **dlogxw** array, which contains partial derivatives of $log\ x_w$ with respect to $log\ m_{s'}$, where s' denotes an aqueous basis species.

gelam.f This module is called by fellow EQLIB module **gcoeff.f.** It computes all of the ${}^{E}\lambda_{ij}$ and ${}^{E}\lambda'_{ij}$ coefficients which appear in Pitzer's equations.

gesum.f This module is called by fellow EQLIB module **gcoeff.f.** It computes the following summations of "higher order electrostatic" second order virial coefficients used in Pitzer's equations: $\sum_i \sum_j ({}^{E}\lambda_{ij} + I\,{}^{E}\lambda'_{ij})\, m_i m_j$ and $\sum_i \sum_j {}^{E}\lambda'_{ij} m_i m_j$. The first is used in computing the activity of water, the second in computing solute activity coefficients.

getdes.f This module is called by fellow EQLIB module **rdtyp7.f.** It finds an option description specified by an asterisk on a "D" format **input** file.

getflg.f This module is called by the EQ3NR module **getspc.f.** It finds the **jflag** value corresponding to a character string specified on a "D" format **input** file.

getlin.f This module is called by fellow EQLIB modules **rdtyp0.f, rdtyp6.f, rdtyp7.f,** and **rdtyp8.f,** by EQ3NR modules **getrdx.f, getss.f, rdninp.f, rdtyp4.f, rdtyp5.f,** and **rdtyp9.f,** and by EQ6 modules **rd6ff.f, rd6log.f, rd6new.f, rd6prs.f, rd6rea.f, rd6sup.f, rd6tol.f, rdmole.f,** and **rdrate.f.** It gets a line of input from a "D" format **input** file.

getlu.f This module is called by fellow EQLIB modules **openin.f** and **openou.f.** It gets the next available logical unit number.

gkey.f This module is called by the EQ3NR module **indatx.f** and the EQ6 module **indatz.f.** It checks a flag read from the **data1** file and compares it with the **iopg1** option switch read from the **input** file. This is done to insure that the data file employed is consistent with the activity coefficient option selected by the user. If the wrong type of data file was provided, an error message is displayed and the program stops.

gmdsm.f This module is called by the EQ3NR module **gcoeff.f.** It computes the following summation of third order virial coefficients: $\sum_j \sum_k \mu_{ijk} m_j m_k$. This is used in Pitzer's equations to compute the molal activity coefficient of the i-th aqueous solute species.

gmsum.f This module is called by the EQ3NR module **gcoeff.f.** It computes the following summation of third order virial coefficients: $\sum_i \sum_j \sum_k \mu_{ijk} m_i m_j m_k$. This is used in Pitzer's equations to compute the activity of water.

gntpr.f This module is called by the EQ3NR module **eq3nr.f** and the EQ6 module **eq6.f.** It sets a temperature range flag corresponding to the desired temperature. For temperatures less than or equal to 100°C, the value is 1. For higher temperatures, the value is 2.

gpheh.f This module is called by the EQ3NR module **scripx.f** and the EQ6 module **scripz.f**. It calculates the *pH*, redox potential, and *pe* using the operational *pH* scale. The operational *pH* scale is defined by the **iopg2** option switch.

gsdsm.f This module is called by fellow EQLIB module **gcoeff.f**. It computes the following summation of "short range" second order virial coefficients used in Pitzer's equations: $\sum_j \sum_k {}^{S}\lambda'_{jk} m_j m_k$. This is used to compute the molal activity coefficient of the *i*-th aqueous solute species.

gselm.f This module is called by fellow EQLIB module **gcoeff.f**. It computes the following summation of "higher order electrostatic" second order virial coefficients used in Pitzer's equations: $\sum_j {}^{E}\lambda_{ij} m_j$. This is used to compute the molal activity coefficient of the *i*-th aqueous solute species.

gsgsm.f This module is called by fellow EQLIB module **gcoeff.f**. It computes the following summation of "short range" second order virial coefficients used in Pitzer's equations: $\sum_j {}^{S}\lambda_{ij} m_j$. This is used to compute the molal activity coefficient of the *i*-th aqueous solute species.

gshm.f This module is presently not used. It calculates the variables **shmc** and **dshmc** that are used in hydration theory models of the activity coefficients of aqueous species.

gsigm.f This module is called by fellow EQLIB module **ngcadv.f**, by EQ3NR modules **arrset.f** and **scripx.f**, and by EQ6 modules **eq6.f** and **scripz.f**. It calculates Σm, the sum of aqueous solute species molalities.

gslam.f This module is called by fellow EQLIB module **gcoeff.f**. It computes all of the ${}^{S}\lambda_{ij}$ and ${}^{S}\lambda'_{ij}$ coefficients which appear in Pitzer's equations.

gspion.f This module is called by the EQ3NR module **eq3nr.f** and the EQ6 module **indatz.f**. It finds the indices of the aqueous hydrogen and chloride ions.

gssum.f This module is called by fellow EQLIB module **gcoeff.f**. It computes the following summations of "short range" second order virial coefficients used in Pitzer's equations: $\sum_i \sum_j ({}^{S}\lambda_{ij} + I\,{}^{S}\lambda'_{ij}) m_i m_j$. This is used in computing the activity of water.

gszm.f This module is called by the EQ3NR modules **arrset.f**, **betas.f**, and **scripx.f** and the EQ6 module **betaz.f**. It calculates the sum of equivalent concentrations of aqueous cations, the sum of equivalent concentrations of aqueous anions, the sum of equivalent concentrations of all aqueous ions, and the charge imbalance. An equivalent concentration is the product of the absolute value of the electrical charge number and molality.

gtime.f This module is presently not used. It gets the system clock time. See also **timdat.f**.

gxi.f This module is called by fellow EQLIB module **ngcadv.f**, by EQ3NR modules **arrset.f** and **scripx.f**, and by EQ6 module **eq6.f**. It computes the ionic strength.

gxisto.f This module is called by the EQ3NR module **scripx.f**. It computes the "stoichiometric" ionic strength.

hjdd.f This module is called by fellow EQLIB module **elmdd.f.** It computes the $J(x)$ and $J'(x)$ functions required to calculate the ${}^{E}\lambda_{ij}$ and ${}^{E}\lambda'_{ij}$ coefficients which appear in Pitzer's equations. It uses the method of Harvie (1981).

hpsat.f This module is called by the EQ3NR module **scripx.f** and by the EQ6 modules **satchk.f** and **sfncaf.f** It calculates the hypothetical affinity of a solid solution. This is defined as the maximum of the affinity to precipitate. This calculation also gives the corresponding (most stable) composition of that solid solution. This computation is based on the composition of the aqueous phase. This routine first solves for the case of ideal mixing. If a non-ideal mixing law is specified, iterative correction is made.

inbdot.f This module is called by the EQ3NR module **eq3nr.f** and by the EQ6 module **indat1.f.** It reads from the **data1** file the hard core diameters (**azero** array) and neutral solute species flags (**insgfl** array) needed as part of the B-dot model for the activity coefficients of aqueous species.

indapo.f This module is called by the EQ3NR module **indatx.f** and by the EQ6 module **indatz.f.** It reads from the **data1** file a one-dimensional array of polynomial coefficients. It is presently used to read the coefficients of the Drummond (1981) polynomial used to compute the activity coefficient of aqueous CO_2. This polynomial is used as part of the B-dot model for the activity coefficients of aqueous species.

indatc.f This module is called by the EQ3NR module **indatx.f** and by the EQ6 module **indatz.f.** It reads from the **data1** file a two-dimensional array of polynomial coefficients, the second dimension corresponding to a given temperature range. It is presently used to read interpolating polynomial coefficients for the pressure (P), the $A_{\gamma,10}$, B_{γ}, and A_{ϕ},Debye-Hückel parameters, the $\dot{B}$ parameter, and the equilibrium constant for the Eh reaction ($log\ K_{Eh}$).

indatk.f This module is presently not used. It is very similar to **indatc.f.** The difference is that it reads a set of interpolating polynomial coefficients which are stored in a three-dimensional array.

intchr.f This module is called by fellow EQLIB modules **ckpars.f** and **rdtyp8.f.** It puts an integer into a character string.

inupt.f This module is called by the EQ3NR module **eq3nr.f.** and by EQ6 module **indat1.f.** It reads parameters for Pitzer's equations from the **data1** file. A header line is read before the parameter data. If an invalid header is found, an error message is written and the code stops.

isamax.f This module is called by fellow EQLIB module **sgefa.f.** It finds the element of a floating point **(real*8)** vector with the largest magnitude. This module is adapted from a module of the same name from a non-proprietary version of LINPACK.

iswch.f This module is called by fellow EQLIB module **swchlm.f.** It does conditional switching of species indices in the **nslmx** and **nmux** arrays to reflect the species index interchanges that accompany basis switching.

itrefn.f This module is called by fellow EQLIB module **msolvr.f.** It iteratively improves a solution to a matrix equation.

kilsum.f This module is presently not used. It computes a part of the HKF equations for the activity coefficients of aqueous species. These equations were deleted from version 7.0 of EQ3/6.

lambda.f This module is called by fellow EQLIB modules **hpsat.f** and **ssfunc.f,** by the EQ3NR module **eq3nr.f,** and by the EQ6 modules **ncmpz2.f** and **raff.f.** It computes the activity coefficients of solid solution end-member components.

lindep.f This module is called by the EQ3NR module **dawfix.f** and by the EQ6 modules **eqcalc.f** and **jgibbs.f**. It tests the first **jdim** elements in the first **idim** rows of the matrix **aa** for linear dependence. If such is found, this module returns **qfcp2** = .true.

lnblnk.f This module is presently not used. It finds the last non-blank character in a character string.

locase.f This module is presently not used. It converts any upper case characters in a character string to lower case.

lsqp.f This module is called by the EQPT module **intrp.f**. It makes an exact or least-squares fit of an interpolating polynomial to data in *x*, *y* format. Whether the fit is exact or not depends on the requested order and the number of points.

matrxe.f This is not a module, but a dummy external which refers to either the EQ3NR module **matrix.f** or the EQ6 module **matrxz.f**. This dummy external is called by the EQLIB routine **nrstep.f**. The actual modules compute the Jacobian matrix for hybrid Newton-Raphson iteration.

molint.f This module is called by the EQ3NR module **scripx.f**. It computes the total molalities of the basis species.

msolvr.f This module is called by fellow EQLIB modules **lsqp.f nrstep.f**, and **polx.f**, by the EQ3NR module **arrsim.f**, and by the EQ6 module **jgibbs.f**. It oversees the solving of a matrix equations. The method used is L-U decomposition. If the reciprocal condition number is less than 100 times the machine epsilon, the solution is iteratively improved.

nactop.f This module is called by the EQ3NR module **eq3nr.f** and by the EQ6 module **eq6.f**. It uses the option flag **iopg1** to obtain the name of the model to be used for calculating the activity coefficients of aqueous species. It also sets associated logical flags defining the generic type of activity coefficient model.

nbsgam.f This module is called by fellow EQLIB modules **gcoeff.f** and **gpheh.f**. It computes the molal activity coefficient of the chloride ion, as defined by the NBS *pH* convention.

ncmpe.f This is not a module, but a dummy external which refers to either the EQ3NR module **ncmpx.f** or the EQ6 module **ncmpe.f**. This dummy external is called by the EQLIB routines **ngcadv.f** and **nrstep.f**. The actual modules compute all parameters that derive from the primary iteration variables and are necessary to write the Jacobian matrix (e.g., all aqueous species concentrations and activities).

nequal.f This module is presently not used. It searches a character string for the first occurrence of a character that does not match a specified character.

newton.f This module is called by the EQ3NR module **eq3nr.f** and by the EQ6 module **eqcalc.f**. It performs hybrid Newton-Raphson iteration to solve for the equilibrium state of a system for which temperature, pressure, and overall system composition are constrained.

ngcadv.f This module is called by fellow EQLIB module **newton.f**, by the EQ3NR module **arrset.f**, and by the EQ6 modules **eq6.f**, **optmzr.f**, **path.f**, and **sfncaf.f**. It recalculates activity coefficients and then recalculates the concentrations and numbers of moles of dependent species.

nrstep.f This module is called by fellow EQLIB module **newton.f**. It executes a single step of hybrid Newton-Raphson correction. The activity coefficients are held constant during this step.

openin.f This module is called by the EQPT module **ofiles.f**, by the EQ3NR modules **eq3nr.f** and **rdinp.f**, and by the EQ6 modules **eq6.f** and **rd6inp.f**. It opens an existing file, using an available device number.

Both formatted and unformatted files can be opened by this routine. It is used to open files such as **input** and **data1**.

openou.f This module is called by the EQPT modules **eqpt.f** and **ofiles.f**, by the EQ3NR modules **eq3nr.f** and **rdinp.f**, and by the EQ6 modules **eq6.f** and **rd6inp.f**. It opens a new or existing file, using an available device number. If the file already exists, it is first destroyed. Both formatted and unformatted files can be created using this routine. It is used to open files such as **output**.

parsln.f This module is called by fellow EQLIB module **getlin.f**. It parses an input character string based on the delimiter "|". This is done as part of the process of interpreting the **input** file in "D" format.

prcndi.f This module is called by fellow EQLIB module **eqlib.f**. It writes copyright language and legal disclaimers to the screen and **output** files.

polx.f This module is called by the EQPT module **lsqp.f**. It fits an interpolating polynomial to a set of input data.

prpheh.f This module is called by the EQ3NR module **scripx.f** and the EQ6 module **scripz.f**. It writes the *pH*, *Eh*, and *pe* values each corresponding to the operational *pH* scale corresponding to the choice of the option flag **iopg2**, the modified NBS *pH* scale and the scale on which the *pH* is numerically equal to $-log m_{H^+}$.

prreac.f This module is called by fellow EQLIB modules **alters.f**, **echolk.f**, **switch.f**, and **swtchk.f**, and by the EQ3NR modules **arrsim.f**, **dawfix.f**, **echox.f**, and **eq3nr.f**. It writes a specified reaction to the screen or **output** file.

ptztab.f This module is called by fellow EQLIB module **inupt.f**. It tabulates the species pairs and triplets corresponding to the coefficients appearing in Pitzer's equations.

qsort.f This module is called by the EQ3NR module **ncmpx.f** and by the EQ6 module **ncmpz.f**. It sorts the elements of a floating point array, using the quicksort algorithm. It is used by the two calling modules to sort the aqueous species in order of increasing concentration or number of moles, respectively.

rdastr.f This module is called by the EQ3NR module **rd3tds.f** and by the EQ6 module **rd6new.f**. It finds the integer option switch value corresponding to an option selected on a "D" format **input** file by a preceding asterisk.

rdtyp0.f This module is called by the EQ3NR module **rdninp.f** and by the EQ6 module **rd6new.f**. It reads a title from a "D" format **input** file.

rdtyp6.f This module is called by the EQ3NR module **rdninp.f** and by the EQ6 module **rd6new.f**. It reads the **nxmod** alter/suppress options from a "D" format **input** file.

rdtyp7.f This module is called by the EQ3NR module **rdninp.f** and by the EQ6 module **rd6new.f**. It reads the **iopt1**, **iopt2**, etc., **iopg1**, **iopg2**, etc., and **iopr1**, **iopr2**, etc., options from a "D" format **input** file.

rdtyp8.f This module is called by the EQ3NR module **rdninp.f** and by the EQ6 module **rd6new.f**. It reads the **iodb1**, **iodb2**, etc., options from a "D" format **input** file

readr.f This module is presently not used. It reads a specified number of lines (records) from a direct access file.

realch.f This module is presently not used. It puts a **real*8** floating point number into a character string.

rindex.f This module is called by fellow EQLIB module **parsln.f.** It searches for a specified substring within an input character string, returning the position of the first character in the substring if the substring is present.

rscal.f This module is called by the EQPT module **intrp.f.** It rescales the interpolating polynomial coefficients computed by the fellow EQLIB module **lsqp.f,** which are normalized. The rescaling removes the normalization.

sasum.f This module is called by fellow EQLIB module **sgeco.f.** It calculates the sum of the magnitudes of the elements of a floating point **(real*8)** vector. This module is adapted from a module of the same name from a non-proprietary version of LINPACK.

saxpy.f This module is called by fellow EQLIB modules **sgeco.f, sgefa.f,** and **sgesl.f.** It multiplies a floating point **(real*8)** vector by a floating point **(real*8)** scalar and adds the result to a second such vector. This module is adapted from a module of the same name from a non-proprietary version of LINPACK.

scal.f This module is called by the EQPT module **wrpar.f.** It scales a set of floating point **(real*8)** values to the interval (-1,1).

scribn.f This module is called by **scribe.f.** It writes the top half of the **pickup** file in "D" format.

sdot.f This module is called by fellow EQLIB module **sgeco.f.** It is a function subroutine which calculates the dot product of two floating point **(real*8)** vectors. This module is adapted from a module of the same name from a non-proprietary version of LINPACK.

sgeco.f This module is called by fellow EQLIB modules **msolvr.f** and **sgesl.f.** It factors a floating point **(real*8)** matrix into L-U form and estimates the reciprocal condition number. This module is adapted from a module of the same name from a non-proprietary version of LINPACK.

sgefa.f This module is called by fellow EQLIB module **sgeco.f.** It factors a floating point **(real*8)** matrix into L-U form. This module is adapted from a module of the same name from a non-proprietary version of LINPACK.

sgesl.f This module is called by fellow EQLIB modules **itrefn.f** and **msolvr.f.** It solves a matrix equation, given an L-U decomposition of the matrix and a right hand side vector. This module is adapted from a module of the same name from a non-proprietary version of LINPACK.

shelv.f This module is presently not used. It does a shell sort of the elements of a character variable array.

sortr8.f This module is called by fellow EQLIB module **srtsum.f.** It makes a shell sort of the absolute values of the elements of a floating point **(real*8)** vector.

srchn.f This module is called by fellow EQLIB modules **inbdot.f** and **inupt.f.** It searches a character variable array to find a match to a specified character string.

srtsum.f This module is called by fellow EQLIB module **itrefn.f.** It computes the sum of the elements of a floating point **(real*8)** vector. The summation is done in order of increasing magnitude of the array elements in order to preserve as much accuracy as possible.

sscal.f This module is called by fellow EQLIB modules **sgeco.f** and **sgefa.f.** It multiplies a floating point **(real*8)** vector by a floating point **(real*8)** scalar. This module is adapted from a module of the same name from a non-proprietary version of LINPACK.

ssfunc.f This module is called by fellow EQLIB module **dsiplx.f**. It is a function subroutine which gives the saturation index of a solid solution as a function of its composition.

stpp.f This module is called by fellow EQLIB module **itrefn.f**. It stuffs a floating point (**real*8**) variable into an array and checks to see if the magnitude of the element is the largest or smallest loaded thus far.

stripl.f This module is called by the EQPT module **ofiles.f**, by the EQ3NR module **rdinp.f**, and by the EQ6 module **rd6inp.f**. It copies the **input** file to the **inputs** file, stripping all comment lines (which have an asterisk in column one).

supprs.f This module is called by the EQ3NR module **flgstx.f** and by the EQ6 module **flgstz.f**. It executes the suppression function of the **nxmod** alter/suppress options.

swchlm.f This module is called by fellow EQLIB module **switch.f**. It swaps Pitzer coefficients when a basis switch is made.

switch.f This module is called by fellow EQLIB module **autosw.f**, by the EQ3NR module **eq3nr.f**, and by the EQ6 module **ibswch.f**. It executes a basis switch.

swtchk.f This module is called by fellow EQLIB module **switch.f**. It checks a proposed basis switch to see if any rules are violated.

texp.f This module is called by fellow EQLIB module **hpsat.f**, by the EQ3NR modules **arrset.f**, **arrsim.f**, **gases.f**, **getrdx.f**, **init3v.f**, **ncmpx.f**, **readx.f**, and **scripx.f**, and by the EQ6 modules **derspc.f**, **eq6.f**, **eqcalc.f**, **evratc.f**, **indat1.f**, **inndx.f**, **modexz.f**, **mshift.f**, **ncmpz.f**, **ncmpz2.f**, **optmzr.f**, **path.f**, **rsatch.f**, **scripz.f**, **sshift.f**, and **taylor.f**. It is a function subroutine which returns 10^x, where x is input to the routine. An argument test is made to avoid overflow.

timdat.f This module is called by the EPT module **eqpt.f**, by the EQ3NR module **eq3nr.f**, and by the EQ6 module **eq6.f**. It obtains the current date and time in ASCII format.

tlg.f This module is called by fellow EQLIB modules **gcoeff.f**, **gpheh.f**, **hpsat.f**, **ngcadv.f**, and **ssfunc.f**, by the EQ3NR modules **arrset.f**, **betas.f**, **eq3nr.f**, **gases.f**, **indatx.f**, **ncmpx.f**,and **scripx.f**, and by the EQ6 modules**affunc.f**, **comp1.f**, **derspc.f**, **eq6.f**, **eqcalc.f**, **indat1.f**, **ncmpz.f**, **ncmpz2.f**, **optmzr.f**, **path.f**, **phsdrp.f**, **raff.f**, **rsatch.f**, **scripz.f**, **taylor.f**, **timer.f**, and **wrtabx.f**. It is a function subroutine which returns the base ten logarithm of an argument. If the argument is zero, the function is returned with a value of -999.

tmpcor.f This module is called by fellow EQLIB module **evdata.f**. It makes temperature corrections to the interaction coefficients in Pitzer's equations.

undflw.f This module is called by fellow EQLIB module **eqlib.f**. It sets a UNIX system flag so that the system will not halt execution when a floating point underflow occurs.

upcase.f This module is called by fellow EQLIB modules **rdtyp6.f**, **rdtyp7.f**, and **rdtyp8.f**, by the EQ3NR modules **getrdx.f**, **rdninp.f**, **rdtyp4.f**, **rdtyp5.f**, and **rdtyp9.f**, and by the EQ6 modules **rd6ff.f**, **rd6log.f**, **rd6new.f**, **rd6prs.f**, **rd6rea.f**, **rd6sup.f**, **rd6tol.f**, **rdmole.f**, and **rdrate.f**. It converts any lower case characters in a character string to upper case.

writr.f This module is presently not used. It writes a specified number of lines (records) to a specified direct access file.

wterm.f This module is called by fellow EQLIB module **evdata.f**. It computes interaction parameters for solid solution mixing laws. More specifically it computes the w array from the apx array.

zsrt.f This module is called by the EQ3NR module **eq3nr.f** and the EQ6 module **indat1.f.** It calculates the array **zsq2** of one-half the charge number squared for aqueous species.

Appendix C. EQLIB Error Messages

All EQ3/6 error messages fit into one of three categories: *error*, *warning*, and *note*. An *error* implies a fatal error. Execution of the current input problem will cease without completion, immediately in some cases, later in others. Which is the case depends on whether it makes more sense to stop immediately or to continue checking for other errors before ceasing execution. A *warning* indicates a condition which may or may not represent a real error. A *note* indicates a condition which could possibly indicate an error, but normally does not. All three types of messages are written to both the screen file and the **output** file. If an *error* message is issued, analysis of the problem may be facilitated by checking any preceding *error*, *warning*, or *note* messages.

Each EQ3/6 error message has the following format:

* *msgtype* - (*source/module*) *Message.*

where *msgtype* = *error*, *warning*, or *note*, *source* is the root name of the source file (e.g., *eqlib*, *eqpt*, *eq3nr*, or *eq6*) containing the *module*, *module* is the name of the module (main program or subroutine) which writes the message, and *Message* is the message itself. The messages are designed to be as self-explanatory as possible. The messages are reproduced here using *AAAA* to stand for a character variable, *IIII* for an integer, and *RRRR* for a floating point number.

Most of the error messages that users are likely to encounter when running EQ3NR or EQ6 deal with problems regarding the **input** file, the supporting data file, or both of these. EQPT has no input file; most of the error messages that users are likely to encounter pertain to problems regarding the **data0** data file that this code is processing. In most instances, the meaning of these messages should be immediately clear to the user. In other instances, it may be necessary to search out other information. In such cases, there are three principal actions that users should take. The first is to check the **output** file for additional diagnostic messages (*warnings* and *notes*) which may bear on the matter. If running EQ3NR or EQ6 and this does not suffice to identify corrective action, compare the echo of the **input** file on the **output** file with the original **input** file. You may find that certain data were not entered in the correct fields, that certain inputs fail to correspond with the necessary lines to follow, or that a line is missing or you have an extra line. In addition, it may help to re-run the problem with the debugging option switch **iodb1** set to 1 or 2. This will trigger the printing of additional information which should help to identify the problem. A small number of messages deal with installation errors. These should also be quite clear.

Some messages deal with programming errors. The user should see these rarely if ever. These are likely to appear somewhat more cryptic to users. Problems of this type must be dealt with by diagnosing the problem (probably with the help of a symbolic debugger) and modifying the code. Most users should probably not attempt corrective action of this sort. The code custodian should be notified of suspected programming errors and may be able to provide fixes.

Some of the messages displayed in this appendix are followed by *Comments* that may help to explain them. The list of messages given here include only those generated by EQLIB modules. Users of EQPT, EQ3NR, and EQ6 may also encounter error messages from modules that are part of the corresponding codes. These messages are listed in similar format in Appendix C of each of the User's Guides for the respective codes (EQPT: Daveler and Wolery, 1992, EQ3NR: Wol-

ery, 1992; EQ6: Wolery and Daveler, 1992). The *errors* are listed first, then the *warnings* and finally the *notes*.

Message: * error - (eqlib/bdmlx) Have overflowed the mu parameter index array (nmxx).

Comment: Increase the dimensioned limit.

Message: * error - (eqlib/bdslx) Have overflowed the s-lambda parameter index array (nsxx).

Comment: Increase the dimensioned limit.

Message: * error - (eqlib/chreal) Could not convert the input string to a real number.

Message: * error - (eqlib/chrint) Could not convert the input string to an integer.

Message: * error - (eqlib/chump) Have insufficient floating point epsilon = RRRR. This must be no greater than RRRR. You need a more powerful computer.

Message: * error - (eqlib/chump) Have insufficient floating point exponent range = +/- IIII. This must be at least +/- IIII to run this code. You may need a more powerful computer or you may need to recompile with a special option.

Comment: This may be followed by the following note:

* note - (eqlib/chump) This computer appears to be a VAX. Try recompiling with the G_FLOATING option. This should increase the exponent range to +/- 308. Some older VAXES do not have the hardware to implement this option. If you have one of these, consider getting a new computer.

VAX machines have a default floating point representation which corresponds to an exponent range of +/- 38. Some older models lack the G_FLOATING option and should not be used to run EQ3/6. There is apparently a Fujitsu supercomputer with an exponent range of +/- 76. We do not recommend running EQ3/6 on machines with exponent ranges less than +/- 100.

Message: * error - (eqlib/ckbndi) Programmer error: Bad variable specified.

Message: * error - (eqlib/ckbndr) Programmer error: Unrecognized value of the range check type specifier variable (itype).

Message: * error - (eqlib/ckpars) Wrong number of delimiters on INPUT file line: IIII Expected IIII delimiters "|"

Message: * error - (eqlib/dsiplx) The argument n = IIII is invalid.

Comment: This is a programming error.

Message: * error - (eqlib/dsiplx) The argument nm = IIII is invalid.

Comment: This is a programming error.

Message: * error - (eqlib/dsiplx) The argument array alpha is invalid- alpha(1) = RRRR alpha(2) = RRRR alpha(3) = RRRR

Comment: This is a programming error.

Message: * error - (eqlib/dsiplx) The argument array deps is invalid- deps(1) = RRRR deps(2) = RRRR

Comment: This is a programming error.

Message: * error - (eqlib/dsiplx) The argument itr = IIII is invalid.

Comment: This is a programming error.

Message: * error - (eqlib/dsiplx) The following element of argument array axx is invalid- axx(IIII,IIII) = RRRR

Comment: This is a programming error.

Message: * error -(eqlib/flpars) The machine epsilon is less than or equal to the smallest positive number-, eps = RRRR smpos = RRRR The compiler is too clever.

Comment: Recompile this module with a lower level of optimization.

Message: * error - (eqlib/gcoeff) Have encountered an illegal value of the activity coefficient option flag- iopg1 = IIII

Message: * error - (eqlib/getdes) Could not find a selected switch for option- "AAAA". Check below INPUT file line: IIII

Message: * error - (eqlib/getdes) Multiple switches were selected for option- "AAAA". Check below INPUT file line: IIII

Message: * error - (eqlib/getdes) Invalid switch selected for option- "AAAA". Check INPUT file line: IIII

Message: * error - (eqlib/getflg) Could not match "AAAA" with a known jflag value. Check INPUT file line: IIII

Message: * error - (eqlib/getlin) Could not write to the OUTPUT file.

Comment: This is either a programming error or a system error.

Message: * error - (eqlib/getlin) Error reading the INPUT file on line IIII.

Message: * error - (eqlib/gkey) The specified option for activity coefficients of aqueous species is not compatible with the supporting data file (AAAA)- iopg1= IIII data file key= AAAA correct key= AAAA Provide the correct set of data files and try again. Do not change the keys on the data files.

Message: * error - (eqlib/inbdot) There are more entries for B-dot model species parameters than there are aqueous species, which number IIII.

Message: * error - (eqlib/intchr) String too small for integer.

Comment: This is a probably a programming error.

Message: * error - (eqlib/intchr) Error writing number.

Comment: This is a system error.

Message: * error - (eqlib/inupt) Bad E-lambda flag on the data file = "AAAA". Allowed values are "on" and "off".

Message: * error - (eqlib/inupt) Have overflowed the lambda index array (nslmx).

Comment: Increase the dimensioned limit.

Message: * error - (eqlib/inupt) Have overflowed the palpha array.

Comment: Increase the dimensioned limit.

Message: * error - (eqlib/inupt) Have overflowed the mu index array (nmux).

Comment: Increase the dimensioned limit.

Message: * error - (eqlib/lambda) The solid solution activity coefficient flag (jsol) has an invalid value of IIII.

Comment: The data file contains a **jsol** *value that is unknown to the code.*

Message: * error - (eqlib/nactop) Have iopg1= IIII. This does not correspond to a valid option for treating the activity coefficients of aqueous species.

Message: * error - (eqlib/nbsgam) Can not find a species index for the chloride ion. Therefore, it is not possible to utilize the extended NBS pH scale in the case of the present model. If the chloride ion is not present in the system being modeled, try adding a trace of it. If the chloride ion is not on the supporting data file, the extended NBS pH scale can not be utilized in any case.

Message: * error - (eqlib/openin) The file AAAA does not exist.

Message: * error - (eqlib/openin) The file AAAA should be formatted but appears to be unformatted.

Message: * error - (eqlib/openin) No logical unit is available to assign to file AAAA.

Message: * error - (eqlib/openin) Can not open file AAAA. Check file existence and permission status.

Message: * error - (eqlib/openou) No logical unit is available to open AAAA copy of file AAAA.

Message: * error - (eqlib/openou) Can not open AAAA copy of file AAAA. Check file existence and permission status.

Message: * error - (eqlib/openou) Can not delete old copy of file AAAA. Check file permission status.

Message: * error - (eqlib/rdastr) Exactly one option must be selected. Place an asterisk before the selected option string. Check INPUT file line IIII.

Message: * error - (eqlib/rdtyp0) The maximum number of entries allowed is IIII. This was exceeded on INPUT file line IIII.

Message: * error - (eqlib/rdtyp6) Valid suppress options are "suppress", "replace", "augmentk", or "augmentg". Value read was "AAAA". Check INPUT file line: IIII

Message: * error - (eqlib/rdtyp6) The maximum allowed number of entries is IIII. Have exceeded this value on INPUT file line IIII.

Message: * error - (eqlib/rdtyp6) Valid species types are "aqueous", "gas", "mineral", or "solid solution".Value read was: "AAAA". Check INPUT file line: IIII.

Message: * error - (eqlib/rdtyp6) Species name can not be blank. Check INPUT file line: IIII.

Message: * error - (eqlib/rdtyp7) Invalid option specified on INPUT file line IIII.

Message: * error - (eqlib/rdtyp7) Programmer error: do not recognize an option flag type named AAAA. Was expecting "iopt", "iopr", "iopg", or "iodb".

Message: * error - (eqlib/rdtyp8) Programmer error: bad variable specified.

Message: * error - (eqlib/rdtyp8) Invalid option specified on INPUT file line IIII.

Message: * error - (eqlib/readr) Could not read record no. IIII of the direct access file whose logical unit number is IIII (iostat=IIII).

Message: * error - (eqlib/realch) The output string is too small to represent a real number.

Comment: This is a probably a programming error.

Message: * error - (eqlib/realch) Can not convert real number to string.

Comment: This is a system error.

Message: * error - (eqlib/stpp) Have overflowed the dotp array.

Comment: Increase the dimensioned limit.

Message: * error - (eqlib/swtchk) The basis species AAAA can not be switched with itself.

Message: * error - (eqlib/swtchk) The basis species AAAA can not be switched with AAAA.

Message: * error - (eqlib/swtchk) Can not do a basis switch of AAAA with AAAA when it is also to be switched with AAAA.

Message: * error - (eqlib/swtchk) The species AAAA can not be involved in a basis switch.

Message: * error - (eqlib/tlg) Can not compute the logarithm of RRRR.

Message: * error - (eqlib/writr) Can not write block IIII on the direct access file whose logical unit number is IIII.

Message: * error - (eqlib/wterm) The activity coefficient option flag (jsol) has invalid value of IIII for solid solution AAAA.

Comment: The data file contains a jsol *value which is unknown to the code.*

Message: * warning - (eqlib/evdata) The temperature of RRRR Celsius is greater than the data file maximum value of RRRR Celsius.

Message: * warning - (eqlib/evdata) The temperature of RRRR Celsius is less than the data file minimum value of RRRR Celsius.

Message: * warning - (eqlib/inbdot) Have duplicate entry on the data file for the B-dot parameters for "AAAA". The first entry will be used.

Message: * warning - (eqlib/ptztab) The species AAAA is in the present model but not represented in any Pitzer coefficients.

Message: * note - (eqlib/alters) Alter species AAAA was not among the loaded AAAA.

Message: * note - (eqlib/alters) The species AAAA is in the strict basis so it has no alter function.

Message: * note - (eqlib/alters) The species AAAA is a fictive fugacity fixing mineral so it has no alter function.

Message: * note - (eqlib/alters) The entity AAAA is a solid solution so it has no alter function.

Message: * note - (eqlib/chump) This computer appears to be a VAX. Try recompiling with the G_FLOATING option. This should increase the exponent range to +/- 308. Some older VAXES do not have the hardware to implement this option. If you have one of these, consider getting a new computer.

Message: * note - (eqlib/inbdot) The following aqueous species have been assigned a default hard core diameter of RRRR Angstroms-

Message: * note - (eqlib/msolvr) The call to eqlib/sgeco for LU decomposition of a matrix failed.

Message: * note - (eqlib/msolvr) The call to eqlib/itrefn to iteratively improve the solution to a matrix equation failed.

Message: * note - (eqlib/ngcadv) Hydration theory has blown up- The calculated value of sigma (hm) is RRRR, which exceeds the physical limit of RRRR. No physical solution exists.

Message: * note - (eqlib/supprs) Suppress species AAAA was not among the loaded AAAA.

Message: * note - (eqlib/supprs) The species AAAA is a fictive fugacity fixing mineral, so it can not be suppressed.

Message: * note - (eqlib/texp) Have exponential truncation- texp (RRRR) has been set to RRRR.

Appendix D. Known Bugs and Such

This appendix presents notes on known bugs and other known unusual phenomena. The discussion here covers the **R10** set of data files, the EQLIB library, and the EQPT, EQ3NR, and EQ6 codes.

The following notes pertain to the supporting data files in the **R10** set: These notes do not reflect errors in the **R7** set which have been corrected in the **R10** set. They also do not reflect other changes between the data files in the two sets.

1. Thermodynamic data for aqueous polyborate species (Bassett, 1977) in **data0.com.R10** appear to understate the stability of the species $B_2O(OH)_5^-$, $B_3O_3(OH)_4^-$, and $B_4O_5(OH)_4^{2-}$. This is shown in the results of the *pH* 8 borate buffer solution problem in the EQ3NR test case input file **boratebufs.3i**, in which the calculated *pH* is 8.15. It should be closer to 8.0, as it was when older EQ3/6 data files were used to evaluate this test case. This problem is of significance for calculations involving borate buffers at *pH* values less than 9.0 and scenarios involving the dissolution of borosilicate glass.

2. Thermodynamic data for the two magnesium hydroxysulfate hydrate phases $Mg_{1.25}SO_4(OH)_{0.5}{:}0.5H_2O$ and $Mg_{1.5}SO_4(OH)$ in **data0.com.R10** are apparently inconsistent with high temperature data for ions and ion-pairs taken from SUPCRT92. These data were reported by Janecky (1982), who derived them assuming high temperature data for the ions taken from an older version of SUPCRT. These phases are now less stable than they should be. They no longer appear when the EQ6 test case input file **heatsw.6i** is run. This test case simulates the heating of seawater to 350°C. These phases are not known to be significant outside the scenario of heating pure seawater. The data for these minerals need to be adjusted for consistency with SUPCRT92.

The following notes pertain to the EQLIB library:

3. The hybrid Newton-Raphson method (used in support of both EQ3NR and EQ6) has trouble dealing with extremely concentrated solutions. Roughly speaking, these are solutions whose ionic strengths are greater than about 12 molal. However, there is no simple way to accurately categorize the method's performance envelope.

The following notes pertain to the EQPT code:

4. EQPT does not currently provide for treating all types of observable interaction coefficients belonging to Pitzer's equations involving interactions with electrically neutral species. See the EQPT User's Guide (Daveler and Wolery, 1992) for more information. See Chapter 3 of the EQ3NR Theoretical Manual and User's Guide (Wolery, 1992) for a discussion of the various types of observable interaction coefficients belonging to Pitzer's equations.

The following notes pertain to the EQ3NR code:

5. The code tends to converge slowly or not at all in dealing with extremely concentrated electrolyte solutions (see Note 3 above).

The following notes pertain to the EQ6 code:

6. The differential equation integration algorithm in EQ6 will not handle "stiff systems" of ordinary differential equations (ODEs). A system of ODEs *per se* is not inherently stiff or non-stiff; rather, stiffness is a condition that appears under certain circumstances. This is a problem only in kinetic mode; the actual (as opposed to relative) rate laws comprise the set of ODEs. Stiffness can occur only if there are two or more such rate laws. The condition of stiffness arises when the magnitude of one of these rates is much greater than that of the other or others, and it begins to rapidly change. Typically, this happens when the reaction in question begins to closely approach a state of partial equilibrium. When stiffness is encountered, the step size is decreased and soon becomes stuck at the minimum value. The code now stops if the step size is stuck at this value, instead of running on. The only solution to this is to add a stiff-system integration algorithm to the code. There are currently no active plans to do this.

7. Another problem in EQ6 exhibits much the same symptoms. It also requires more than one reactant, and may occur in kinetic or reaction progress mode. If one chooses to set **nrk** = 1 (specified relative rate) to control the dissolution of a reactant and **nrpk** = 0 (follow partial equilibrium) to control its precipitation, the **nrpk** = 0 option can not be followed if one has also suppressed the formation of the reactant in question as a product mineral. The **nrpk** = 0 option requires the formation of such product mineral in order to hold the reactant affinity at zero. Without this, the dissolution rate goes to zero, but continued advancement in overall reaction progress may cause undersaturation, returning the relative rate of dissolution to the specified value. After another small advancement in reaction progress, the dissolution rate again goes to zero, and the pattern repeats itself. The step size is decreased and usually soon becomes stuck at the minimum value. The code then notes this and stops. Mathematically, the problem has become ill-posed. The solution is to not suppress a phase if **nrpk** is set to 0. When this problem has been seen to manifest itself, the phase has been suppressed unintentionally by a subset-selection (**nxopt**) option and not specified as an exception to that option.

8. EQ6 periodically exhibits trouble when it encounters mineral assemblages which fix the activity of water (e.g., gypsum plus anhydrite, or epsomite plus hexahydrite). The step size may drop to and become stuck at the minimum value, leading to early termination of the run. The cause of this is presently unknown. It is probably either an error in the coding for evaluating the Jacobian matrix or a problem in the way activity coefficients are re-evaluated in the hybrid Newton-Raphson algorithm (suggested by Note 3 above).

9. The code tends to converge slowly or not at all in dealing with extremely concentrated electrolyte solutions (see Note 3 above).

10. The EQ6 module **jgibbs.f** tests the mineral assemblage in the equilibrium system for violations of the mineralogical phase rule. The module as presently written only tests pure minerals. It does not look at any solid solutions. If a solid solution must be deleted to resolve a violation of the mineralogical phase rule, the code must rely on other phase deletion algorithms to provide the proper course of action.

11. When EQ6 operates without a redox variable, the log oxygen fugacity variable is assigned a value of -999. The saturation indices and affinities for certain solid phases such as *al*, *graphite*, *k*, and *si* are calculated and reported using this value. These phases are irrelevant to the computed model and the reported *SI* and affinity data for them should be ignored. These phases are technically suppressed when there is no redox variable. Thus, they can not be precipitated.

12. The maximum time parameter **timemx** which appears in the EQ6 **input** file may be exceeded when the code is running in time mode. The code does not find the value of reaction progress corresponding to the desired maximum time. Instead, it simply stops the reaction path calculation when this is exceeded. Note also that the code does not provide for the use of print intervals defined in terms of time.

Appendix E. Previous Versions of EQ3/6

With this release of EQ3/6, the version numbering system is changed to that now commonly used for most software. The present report deals with version 7.0 of EQ3/6. The previous versions released from Lawrence Livermore National Laboratory are listed below. The new numbering system has been retrofit to the previous releases.

New System	Old System	Date	Comment
1.0	2020	Feb. 1979	First release from LLNL
2.0	2055	Nov. 1979	Second release from LLNL
3.0	3015	Dec. 1980	Third release from LLNL
3.1	3015B	April 1981	Version 3105 with corrections
4.0	3175	Dec. 1981	Fourth release from LLNL
4.1	3175B	Sept. 1982	Version 3175 with corrections
5.0	3230	May 1983	Fifth release from LLNL
5.1	3230B	March 1984	Version 3230 with corrections
6.0	3245.0288	Feb. 1988	Sixth release from LLNL
6.1	3245.0888	Aug. 1988	Version 3245.0288 with corrections
7.0	3245.1090	Oct. 1990	New version (a super-extension of the 3245 series)
7.1		Aug. 1992	Version 7.0 with corrections

Work on version 8.0 (version 3270) was halted in early 1989. Development is expected to resume after publication of the present series of reports documenting version 7.0.