SKB TECHNICAL REPORT 93-21

Development of "CHEMFRONTS", a coupled transport and geochemical program to handle reaction fronts

Catharina Bäverman

Department of Chemical Engineering,

Royal Institute of Technology, Stockholm, Sweden

October 1993

SVENSK KÄRNBRÄNSLEHANTERING AB

SWEDISH NUCLEAR FUEL AND WASTE MANAGEMENT CO

BOX 5864 S-102 40 STOCKHOLM

TEL. 08-665 28 00 TELEX 13108 SKB S
TELEFAX 08-661 57 19

DEVELOPMENT OF "CHEMFRONTS", A COUPLED TRANSPORT AND GEOCHEMICAL PROGRAM TO HANDLE REACTION FRONTS

Catharina Bäverman

Department of Chemical Engineering, Royal Institute of Technology, Stockholm, Sweden

October 1993

ABSTRACT

A computer program to calculate coupled mass transport and fluid rock interactions has been developed. The program, CHEMFRONTS, is based on the quasi-stationary state approximation and uses a kinetic expression for the mineral dissolution and precipitation coupled to a transport model. It is adapted to handle sharp reaction fronts. Such fronts evolve in the ground and typical examples are redox fronts and dissolution and precipitation fronts.

CHEMFRONTS calculates the chemical reactions for one-dimensional advective flow through a porous medium. Reactions between the water and the solid phase such as dissolution and precipitation are included in the model. In the water phase, complexation and redox reactions are also computed.

To verify the program, comparisons have been made with results obtained with other computer programs, CHEQMATE, PHASEQL/FLOW, and DYNAMIX. Natural analogues, such as Poços de Caldas and Cigar Lake, are also studied. The results from the simulations and comparisons are encouraging.

ABSTRACT (Swedish)

Ett dataprogram har utvecklats för beräkning av masstransport kopplad till reaktioner mellan vätska och berg. Programmet, CHEMFRONTS, bygger på den kvasistationära tillståndsapproximationen och beskriver kinetisk mineralupplösning och -utfällning kopplad till en transportmodell. Det är anpassat för att hantera skarpa reaktionsfronter. Sådana bildas i marken och typiska exempel är redoxfronter samt upplösnings- och utfällningsfronter.

CHEMFRONTS beräknar kemiska reaktioner för ett en-dimensionellt advektivt flöde genom ett poröst medium. Modellen omfattar reaktioner mellan fast fas och vatten såsom upplösning och utfällning. Komplexbildning och redoxreaktioner i vätskefas beräknas också.

Jämförelser med andra program, CHEQMATE, PASEQL/FLOW och DYNAMIX, har gjorts för att kontrollera programmet. Studier har gjorts av naturliga analogier som Poços de Caldas och Cigar Lake. Resultaten från simuleringarna och jämförelserna är uppmuntrande.

CONTENTS Page

SAMMANFATTNING

BAKGRUND Vattenflöde genom poröst material intresserar många forskare inom olika områden. Många geokemiska reaktioner orsakas av vattenflöde och infiltrering av reaktiva ämnen som väte och syre. Möjligheten att förutsäga geokemiska reaktioner under en geologisk tidsperiod av tusen och kanske miljoner år är av intresse för till exempel slutförvar av radioaktivt avfall och andra riskavfall, och för att förutsäga vittring av betong.

MÅL Målet med projektet var att utveckla ett datorprogram som kopplar geokemiska reaktioner med transport för att förutsäga geokemiska reaktioner med geologiskt tidsperspektiv. Programmet skulle kunna hantera skarpa reaktionsfronter som till exempel redoxfronter och upplösningsfronter som förekommer i reducerande berg och i betong. Programmet skulle skrivas i FORTRAN77 för att kunna flyttas mellan olika typer av datorer. Programmet skulle också vara enkelt att modifiera.

METOD Dataprogrammer CHEMFRONTS baserar sig på "the quasi-stationary state approximation" utvecklad av Lichtner (1988). Ämnena i vattenfasen antas vara i jämvikt medan den fasta fasens upplösning och utfällning beskrivs som ett kinetiskt förlopp.

RESULTAT CHEMFRONTS kan användas för att beräkna komplicerade problem med flera samtidigt vandrande fronter inklusive redoxfronter. För att verifiera programmet har exempel som finns i litteraturen beräknats och resultaten jämförts. Detta har visat att CHEMFRONTS ger resultat som är väl jämförbara med med program som baserar sig på andra modeller. Beräkningar från problem som tagits från naturliga analoger som Poços de Caldas och Cigar Lake har gett uppmuntrande resultat.

SUMMARY

BACKGROUND Water flow through porous media is of interest to many scientists in various fields. Many geochemical reactions are caused by the water flow and infiltration of reactive species such as hydrogen ions and oxygen. Prediction of geochemical reactions for geological time periods of thousands and maybe millions of years are useful for matters such as the final disposal of nuclear and other hazardous waste, and the degradation of concrete.

OBJECTIVE The objective of the project was to develop a coupled geochemical and transport computer program to predict geochemical reactions over geological time scales. The program should be able to handle sharp reaction fronts such as the redox and dissolution fronts that occur in reducing bedrock and concrete. It should be written in FORTRAN77, be portable, and be easy to modify.

APPROACH The computer program, CHEMFRONTS, is based on the quasi-stationary state approximation developed by Lichtner (1988). The species in the aqueous phase are assumed to be in equilibrium, whereas the solid phase dissolves and precipitates with a kinetic reaction rate.

RESULTS CHEMFRONTS can be used for calculating complicated problems with many simultaneous moving fronts including redox fronts. The program has been verified by applying it to examples found in the literature. The results from CHEMFRONTS compare well with those from programs based on other models. Computations of problems taken from natural analogues as Poços de Caldas and Cigar Lake have shown encouraging results.

1 INTRODUCTION AND BACKGROUND

The transport of water through porous media is relevant to many scientific fields, including environmental protection. Most geological reactions are a result of water flow. Chemical erosion occurs when the more soluble minerals in rocks dissolve in the water flowing through them. The less soluble parts eventually become particles that are small enough to be transported by wind and water streams.

When the ground has been contaminated with hazardous species, it is important to predict the consequences. Once contaminants are released into the subsurface system, they will interact with both groundwater and solids in the ground. During subsurface transport, reactive solutes are subjected to a variety of hydrophysical and chemical processes. If these reactions are anticipated, the correct treatment of the contaminated area can be applied.

The modelling of geochemical systems is useful for solving many environmental problems. With modelling, it is possible to predict the equilibrium between the water phase and the solid phase, the transport of a liquid phase through a solid matrix, the dissolution and precipitation of rocks, and several related matters. Various models are available for predicting geochemical reactions. Some give the equilibrium between the water phase and the solid phase, some in addition calculate the transport of water through a solid matrix with chemical changes.

Final repositories for radioactive waste will probably be in bedrock (Herbert et al., 1987). The most likely way in which radionuclides from buried waste might reach the biosphere is through dissolution and transport by groundwater flow. Mathematical models are used to understand these complicated physical and chemical processes, and to predict the groundwater flow and transport over the very long time-scale involved. These models are often solved numerically.

Natural analogue studies provide a unique opportunity to test and validate numerical models of fluid/rock interaction involving weathering processes. Such models cannot be tested with laboratory or field experiments, because of the generally slow reaction rates and large time scales associated with these processes. The age of a rock formation and its initial composition can be determined by standard geological investigations. This information can be obtained with other data, such as water infiltration. The data can then be processed in a coupled computer program for transport and geochemical phenomena, and the chemical evolution of the rock can be predicted. A comparison of the predicted data with the real rock formation is used to validate the model.

There are many computer programs for calculating geochemical evolution (see section 2.2). Most fail to calculate sharp redox fronts, and some need very long computing times even with the largest computers. One promising approach for sharp reaction fronts is to use the kinetic dissolution of minerals and the quasi-stationary state approximation (section 2.4). This does not require excessive computer resources (Lichtner, 1988).

2 THEORY

Equations describing geochemical processes in natural systems are used to describe mass transport coupled to fluid-rock interaction over time periods of geological interest. The situation is complicated by changes in the mineral reaction zones.

2.1 The system

The situation to be modelled is the following: A "column" filled with solid particles of different minerals, figure 2.1.1, is subjected to water flow. The incoming water has a known composition of dissolved species. As the water passes through the column, some minerals dissolve, some do not react and some precipitate. Entirely new minerals may form in the reactions.

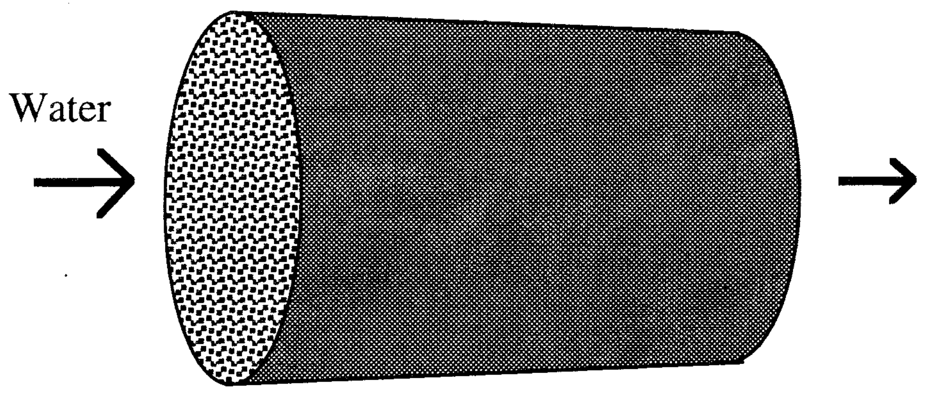


Figure 2.1.1 A column of porous medium with water infiltrating.

Consider the example where the minerals in the column are in reduced form. Oxygen enters the rock with the water and oxidizes the reduced minerals in the column. A sharp redox front forms, with the reduced minerals downstream and the oxidized minerals upstream, figure 2.1.2. Various reactions occur at the front, so the chemistry of the water can be completely different on either side of the front.

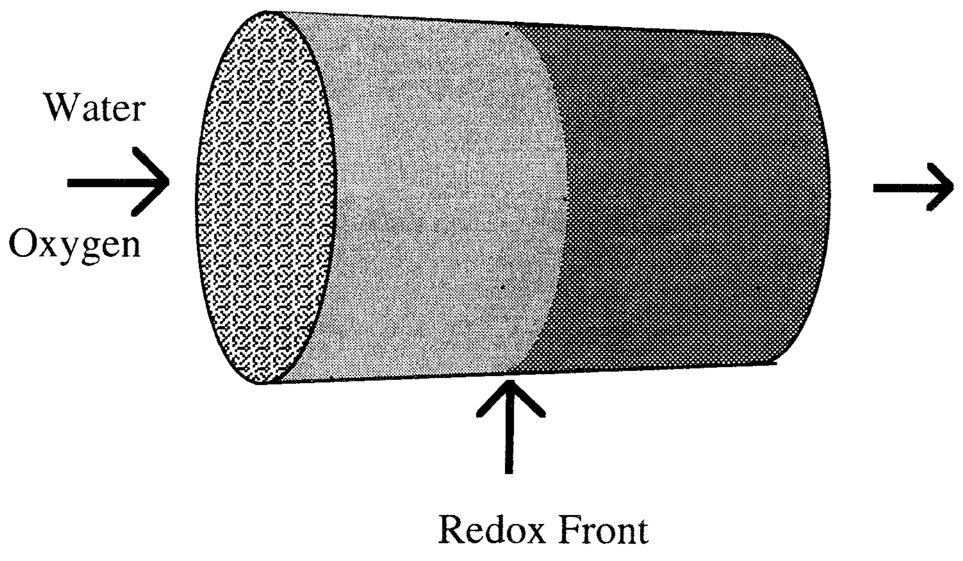


Figure 2.1.1 A redox front, with reduced minerals downstream and oxidized minerals upstream, is formed by reactions of the reduced minerals with the infiltrated oxygen.

An equilibrium program can be used to determine the reactions between solid and aqueous phases, and to calculate the speciation of the aqueous species in equilibrium with the solid phase, section 2.2.1. To predict chemical reactions and water transport through a porous medium, a coupled geochemical and transport program can be used, section 2.2.2. There are coupled programs that calculate both equilibrium and water transport; others use a kinetic reaction rate formulation coupled to transport.

2.2 Approaches to model equilibrium, kinetics and transport

Several computer programs have been developed to predict geochemical evolution. Various approaches have been used, distinguished mainly by the use of batch or flow-through systems. The batch examples are made with equilibrium programs and the flow-through systems use transport models.

2.2.1 Solving equilibrium problems and equilibrium programs

Several computer programs have been developed to predict reactions between an aqueous solution and solid material. In this section, the equilibrium approach and some common computer programs based on it are presented.

2.2.1.1 Solution approaches for general equilibrium problems

The main purpose of an equilibrium program is to estimate the speciation of the aqueous species from the analytical, total concentration of the solution. The hydrogen concentration, however, is often represented by the pH, based on the free concentration of hydrogen ions.

The total amount of material in a system is the sum of the materials in the species in the system. This is expressed by the material balance equation

$$\sum_i \nu_{ij} C_{xi} + C_j - Y_j = \varepsilon_j = 0 \tag{2.2.1.1.1}$$

where ν_{ij} is the stoichiometric coefficient for the component j in the complex i, C_j is the concentration of the free component, Y_j is the total concentration of the component j and ε_j is the error in the material balance. When ε_j is zero the problem is solved. The complex concentration C_{xi}, or rather the complex activity a_{xi}, is expressed in the equation for mass law

$$a_{xi} = K_i \prod_j a_j^{\nu_{ij}} \tag{2.2.1.1.2}$$

where K_i is the equilibrium constant of the complex i and a_j is the activity of the component j. The activity of the complex is defined by

$$a_{xi} = C_{xi}\,\gamma_{xi} \tag{2.2.1.1.3}$$

and γ_{xi} is the activity coefficient of the complex i.

The aim of an equilibrium program is to calculate the free equilibrium concentrations of all aqueous species, C_j and C_{xi}, from the total, analytical concentration of the components, Y_j. From an initial guess of C_j the complex concentrations, C_{xi}, are calculated by equation (2.2.1.1.2). The error in the mass balance, ε_j, is found from equation (2.2.1.1.1). An iterative technique is used to adjust the value of C_j to minimize the error, ε_j. When ε_j is zero, or less than the permitted error, the problem is solved and the equilibrium concentrations for species are found. A more detailed description is given by Westall (1979). This method is used in HALTAFALL (Ingri et al., 1967), MINEQL (Westall et al., 1976) and MICROQL (Westall, 1979).

The saturation indices for the minerals in the example can be calculated from their aqueous concentration. If any of the minerals present are undersaturated, the mineral can dissolve to saturation level. If a mineral is found to be supersaturated, it may precipitate to saturation level. If several minerals are involved in the reaction, there can be more than one supersaturated mineral. A trial and error method can then be used. In the first step, the mineral with the highest saturation index is precipitated to saturation level. Then new saturation indices are calculated, undersaturated minerals are dissolved, and new saturation indices calculated again. If there are still some supersaturated minerals, the next mineral precipitates to saturation level. This procedure is repeated until there are no undersaturated minerals present, and no mineral involved in the example is supersaturated. This method is used in EQ6 (Wolery and Daveler, 1989).

2.2.1.2 EQ3NR

The EQ3NR program (Wolery, 1983) models the thermodynamic state of an aqueous solution by using a modified Newton-Raphson algorithm to calculate the distribution of aqueous species such as simple ions, ion pairs and aqueous complexes. The program evaluates the degree of disequilibrium for various reactions and computes either the saturation index or thermodynamic affinity for minerals in the data base DATA0 (see below). Input to EQ3NR consists primarily of data derived from total analytical concentrations of dissolved components but can also include pH, alkalinity, electrical balance, phase equilibrium (solubility) constraints, and a default value for either Eh, pe[1], or the logarithm of the oxygen fugacity.

EQ3NR can be used alone, and must be used to initialize the reaction-path calculations by EQ6, its companion program (see below). Both EQ3NR and its supporting thermodynamic database have extensive documentation.

2.2.1.3 EQ6

EQ6 (Wolery and Daveler, 1989) is a computer program to calculate reaction paths (chemical evolution) in reacting systems consisting of water and minerals or other solids. Speciation in aqueous solution is an integral part of these calculations. EQ6 computes models of titration processes (including fluid mixing), irreversible reactions in closed systems, irreversible reactions in some simple kinds of open systems, and heating or cooling processes. EQ6 also solves "single-point" thermodynamic equilibrium problems.

[1] $pe = 20.78 + \frac{1}{2} \log (\sqrt{p_{O_2}}\,[H^+])$ (Stumm and Morgan, 1981)

2.2.1.4 PHREEQE

PHREEQE (Parkhurst et al., 1980) can simulate several types of reactions, including the addition of reactants to a solution, the mixing of two waters, and titrating one solution with another. During the reaction simulation, the program calculates the pH, the pe, the total concentration of elements, the amounts of minerals (or other phases) transferred into or out of the aqueous phase, the distribution of aqueous species, and the saturation state of the aqueous phase with respect to specified mineral phases. PHREEQE is used in CHEQMATE (Harworth et al.,1988), DYNAMIX (Liu and Narasimhan, 1989a), and PHREEQM-2D (Willemsen, 1992) described below.

2.2.1.5 SOLMINEQ.88

SOLMINEQ.88 (Perkins et al., 1990) calculates the speciation among the aqueous components and the saturation indices of minerals. It is based on the 1973 version of SOLMNEQ (Kharaka and Barnes, 1973) and various updated versions thereof. The new options in SOLMINEQ.88 enable it to calculate the effects of boiling and mixing solutions. It can also predict the effects of dissolution and precipitation of minerals. SOLMINEQ.88 has a user-friendly input program, SOLINPUT, to update the input files.

SOLMINEQ.88 is particularly useful for modelling interactions in sedimentary basins and in thermally stimulated oil reservoirs where petroleum, organic species, and high temperatures, pressures and salinities prevail.

2.2.1.6 WATEQ

The computer program WATEQ (Truesdell and Jones, 1974) calculates the equilibrium distribution of inorganic aqueous species of major and important minor elements in natural waters by using chemical analysis and *in situ* measurements of temperature, pH and redox potential. From this model, the state of reaction of the water with solid and gaseous phases is calculated. Thermodynamic stabilities of aqueous species, minerals and gases have been selected by means of a careful consideration of all available experimental data.

2.2.2 Solving transport problems and transport programs

There are various approaches for solving the problem of solute transport in water flow through porous media. The basic ideas of the models and some of the programs based on them are described in this section.

2.2.2.1 Solution approaches for transport problems

Transport models are based on the mass balance equation for the aqueous solution

$$\begin{bmatrix}\text{Rate of}\\ \text{mass in}\end{bmatrix} - \begin{bmatrix}\text{Rate of}\\ \text{mass out}\end{bmatrix} + \begin{bmatrix}\text{Rate of production}\\ \text{of mass by}\\ \text{chemical reaction}\end{bmatrix} = 0 \qquad (2.2.2.1.1)$$

The rate of production of mass in equation 2.2.2.1.1 is found from the dissolution of the solid phase (or negative production rate for precipitation)

$$\begin{bmatrix}\text{Rate of production}\\ \text{of mass by}\\ \text{chemical reaction}\end{bmatrix} = \begin{bmatrix}\text{Rate of mass}\\ \text{dissolved}\end{bmatrix} - \begin{bmatrix}\text{Rate of mass}\\ \text{precipitated}\end{bmatrix} \qquad (2.2.2.1.2)$$

Many coupled geochemical and transport programs are a combination of the equilibrium model, described in section 2.2.1.1, and a transport model (Neretnieks, 1992). One version is the box model, figure 2.2.2.1.1, where the system is described by a series of coupled cells containing the solids. The inlet water flows into cell number 1 where it reacts with the solid phase, dissolves minerals and undergoes redox reactions in which complexes are formed and new solid phases may precipitate. When the reactions have taken place in all the cells, and they are in equilibrium, the water is transported by advection from cell number n to cell number n+1. Aqueous species may also move between the cells in both directions by diffusion. The compositions in the various cells are calculated independently of the other cells in every reaction step.

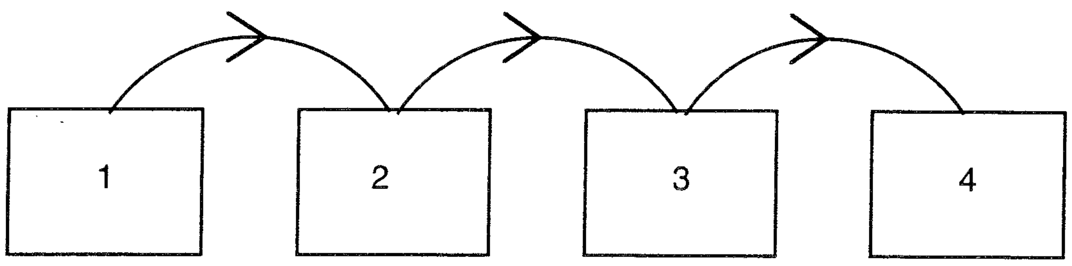


Figure 2.2.2.1.1 The box model.

Many computer programs are based on this coupled equilibrium and transport model, for example CHEQMATE (Harworth et al., 1988), TRANQL (Cederberg et al., 1985), and PHASEQL/FLOW (Walsh et al., 1984), or extended to two or three dimensions, DYNAMIX (Liu and Narasimhan, 1989a) and HYDROGEOCHEM (Yeh and Tripathi, 1991). CHEMTRN (Miller, 1983, and Miller and Benson, 1983), although it uses discretization in cells, solves the transport and equilibrium equations simultaneously.

Rate equations can be used to assess the precipitation and dissolution of solid phases, as an alternative to the coupled mass action/equilibrium model. The driving force in the rate equations is assumed to be proportional to the difference between the ion activity product and the solubility product. The reaction rates also depend on the surface of the solid phase. This approach is used in programs such as CHMTRNS (Noorishad and Carnahan, 1987), PRECIP (Noy, 1990) and MPATH (Lichtner 1990). PRECIP and MPATH are based on the quasi-stationary state approximation (Lichtner, 1988) described in section 2.4.

2.2.2.2 CHEMTRN

CHEMTRN (Miller, 1983, and Miller and Benson, 1983) is a computer program that simulates the transport of chemical species in groundwater systems. Equilibrium is assumed in all chemical reactions, and the thermodynamic activities of all reacting species are related by mass-action expressions. The program includes dispersion/diffusion, advection, sorption of ions and complexes onto the solid matrix, formation of complexes in the aqueous phase, precipitation and dissolution of solids. No database is provided with the program.

The programs CHMTRNS (2.2.2.4) and THCC (2.2.2.10) are extensions of CHEMTRN.

2.2.2.3 CHEQMATE

CHEQMATE (Harworth et al., 1988) models one-dimensional diffusion and electromigration of ionic species with chemical equilibration. The program consists of two parts, chemical-equilibria and ion-migration processes, iteratively coupled, so that local equilibrium is maintained as the transport processes evolve.

The chemical part is based on PHREEQE (Parkhurst et al., 1980). CHEQMATE predicts the evolution of the aqueous chemistry and mineral inventory in time and space. It includes an automatic mineral-accounting procedure, so that solid phases may be added or removed from the system as precipitation or dissolution occurs. Although CHEQMATE is a very versatile program, like most other coupled programs it assumes local chemical equilibrium at all times. This would mean that chemical equilibrium processes occur much more rapidly than ionic transport.

2.2.2.4 CHMTRNS

The computer program CHMTRNS (Noorishad and Carnahan, 1987) is an extension of the chemical transport program CHEMTRN (Miller, 1983). CHMTRNS can simulate the kinetic dissolution or precipitation of solids as well as the irreversible dissolution of glass. Oxidation-reduction reactions are treated by defining a hypothetical electron activity as a basis species subject to transport, as are other aqueous basis species. For multivalent elements, a species in the highest oxidation state is chosen to be the basis species. Reduction to a lower oxidation state is described formally by a half-cell reaction in which the higher valent species "reacts" with a hypothetical electron to form the lower-valent species.

Including a heat transport formulation in CHMTRNS does not alter the mass transport formulation in any way. The only necessary addition is the formulation of a functional relationship between the thermodynamic constants and the temperature.

2.2.2.5 DYNAMIX

DYNAMIX (Liu and Narasimhan, 1989a) is a redox-controlled, multiple-species, multidimensional, chemical transport model. The model includes advection, diffusion-dispersion, transport of oxygen, redox reactions and acid-base reactions, aqueous complexation, precipitation-dissolution, and kinetic mineral dissolution. A correct mineral distribution is automatically located on basis of minimized Gibbs free energy.

The coupled transport and reaction equations are solved by a two-step dynamic mixing algorithm. The transport equation is first solved by the explicit finite-difference method. The chemical equilibrium submodel is then called to calculate the distribution of chemical species under thermodynamic partial-equilibrium conditions.

DYNAMIX couples the chemical speciation program PHREEQE (Parkhurst et al., 1980) with the transport program TRUMP (Edwards, 1972). The model is limited by the Gibbs phase rule to having no more solid phases than components. It can be used for two-dimensional flow fields.

By using the Gibbs free energy to calculate the saturation index (driving force) it is possible to identify quantitatively which mineral solubility product has the highest index and is therefore most likely to appear in the system as it approaches thermodynamic equilibrium. The kinetic dissolution of minerals is based on the results from the equilibrium calculation.

2.2.2.6 HYDROGEOCHEM

HYDROGEOCHEM (Yeh and Tripathi, 1991) is a coupled hydrogeochemical model for simulating the transport of reactive contaminants in groundwater. The model is designed for flow through heterogeneous, anisotropic, saturated-unsaturated media, under transient or steady flow conditions. It simultaneously simulates the chemical processes of dissolution-precipitation, adsorption-desorption, ion exchange, redox, acid-base reaction, and the formation of complexes. The precipitation-dissolution is determined by the equilibrium in the water. The program can be used for two-dimensional calculations.

2.2.2.7 MPATH

The computer program MPATH (Lichtner, 1990) is based on a kinetic description of mineral reaction rates, which in turn is based on the quasi-stationary state approximation (Lichtner, 1988). It solves a time-space representation of mass conservation equations. The equations describe a multi-component geochemical system of minerals reacting in an aqueous solution.

The program uses an extensive database of minerals and aqueous species, equivalent to the EQ3/6 database (Wolery, 1983). The user must select which minerals to include in the calculation for the particular geochemical system considered. An option in the code allows minerals to become super-saturated without reacting until a specified threshold affinity is reached.

MPATH cannot account for diffusion and dispersion, as can the other programs described in this section. On the other hand, it can handle very sharp fronts.

2.2.2.8 PHASEQL/FLOW

PHASEQL/FLOW (Walsh et al., 1984) calculates the aqueous compositions of a flowing solution as a function of time and space in a one-dimensional porous medium. The program calculates the dissolution and precipitation of solids by an equilibrium approach, plus redox reactions and adsorption. The calculations are based on the assumptions that the porous medium is homogeneous with constant porosity, the viscosity and density of the fluid phase is independent of composition, and the fluid and solid phases are in chemical equilibrium.

2.2.2.9 PHREEQM-2D

PHREEQM-2D (Willemsen, 1992) is a coupled geochemical-transport model consisting of the geochemical reaction program PHREEQE (Parkhurst et al., 1980) and the two-dimensional heat and solute groundwater transport program HST2D (Hagoort, 1989).

2.2.2.10 PRECIP

The computer program PRECIP (Noy, 1990) is based on the 'local equilibrium' assumption (Lichtner, 1988). It calculates advective and dispersive flow, precipitation and dissolution of minerals, density changes of the water solution due to hydraulic head and concentration changes, and porosity changes due to compressibility of solid components and fluid.

2.2.2.11 THCC

The program THCC (Carnahan, 1990) couples precipitation and dissolution reactions with diffusive mass transport via porosity changes. The program simulates the transport of reactive chemical species by advection and by hydrodynamic dispersion or mass diffusion in one-dimensional or cylindrically symmetric geometry. Chemical reactions are assumed to be in a state of local equilibrium. The reactions simulated are complexation, oxidation-reduction, ionization of water in the aqueous phase, reversible precipitation of solid phases, and ion exchange. The program can simulate systems with temporally and spatially variable fields of temperature, and the radioactive decay of selected reactants.

2.2.2.12 TRANQL

TRANQL (Cederberg et al., 1985) is a mass transport model for a multicomponent solution system. It can include a wide range of significant chemical equilibrium processes. Significant equilibrium chemical reactions, such as complexation, ion exchange and competitive adsorption, may be included. The technique is to deal with the equilibrium interaction chemistry independently of the mass transport equations. This leads to a set of algebraic equations for the chemistry coupled to a set of differential equations for the mass transport.

MICROQL (Westall, 1979) is used to fully describe all chemical processes in TRANQL, while the general mass transport equations are used to describe advective-dispersive transport of the chemical species.

2.2.3 Databases

To calculate the equilibrium composition of the water phase and saturation index of the minerals or dissolution and precipitation of solid phases, thermodynamic data are needed for the complexes, gases and solids involved in the reactions. Some databases are available for this purpose.

2.2.3.1 EQ3/6 database, DATA0

DATA0 (Wolery 1983) is the database belonging to the EQ3/6 package. It consists of a list of the elements, atomic weights, oxides and gravimetric factors, and a list of all the basic aqueous species, one for each element. Several inorganic aqueous ions and complexes are included. For each one, the chemical elements are specified, as are the stoichiometric constants for the reaction that are needed to make one formula of the complex. The equilibrium constants for the complexes and ions are listed for 0, 25, 60, 100, 150, 200, 250 and 300°C, a pressure of 1.013 bars up to 100°C, and the steam/water equilibrium pressure at higher temperatures. The list of complexes is followed by a list of minerals and other inorganic solids, with the same data as for the complexes, plus the mineral molar volumes. Finally, there is a list of gases followed by a list of solid solutions.

2.2.3.2 HATCHES

The data in HATCHES (Cross and Ewart, 1991) have been obtained from the literature, when available, and validated where possible for conditions of interest by experiment. The database is used in the PHREEQE program (Parkhurst et al., 1980), and in the CHEQMATE program (Harworth et al., 1988). It is stored by using the Ashton-Tate dBase III database management program on an MS-DOS type personal computer, and has been built onto the original USGS database (Parkhurst et al., 1980) supplied with the PHREEQE program.

2.2.3.3 SKB database

In the Swedish program for the disposal of spent nuclear fuel, the recommendations from the Nuclear Energy Agency Thermochemical Data Base Project (NEA-TDB) data selection group (Wanner, 1990) are to be adopted. Until then, there is a need for workable databases. The EQ3/6 database has been updated for uranium and plutonium (Puigdomènech and Bruno, 1991) with thermodynamic data from the literature.

2.2.3.4 NEA-TDB

The Organisation for Economic Co-operation and Development (OECD) Nuclear Energy Agency (NEA) is developing a chemical thermodynamic database for elements of interest in various areas of nuclear technology (Wanner, 1990), especially areas of radioactive waste management research, such as the safety analysis of nuclear waste repositories. Elements considered are uranium, neptunium, plutonium, americium and technetium.

2.3 Reasons for a new model

The accuracy of the coupled equilibrium and transport box model programs depends on the size of the boxes, section 2.2.2. The finer the column divisions are, the longer the computing time needed, and the larger the memory required. Because an explicit time-stepping procedure is very often used, small time steps must be used to obtain sufficient accuracy. This also affects the computing time. The "box" based programs do not seem to handle well the sharp fronts that occur in rocks with reducing minerals like pyrite (FeS_2). The chemistry at the redox fronts is needed to explain why solids that are insoluble both in the oxidized region and in the reduced region accumulate at the redox front and move with it. Cross et al. (1991) discuss the use and limitations of using a box-based model to handle reaction fronts.

The kinetic programs based on the quasi-stationary state approximation, described in section 2.4, have the potential to compute geochemical problems involving sharp redox fronts. The computer program PRECIP (Noy, 1990) was still under development and was not available. MPATH (Lichtner, 1990) was in research mode and not fully available to us.

It was impractical to calculate the evolution of redox fronts and other simultaneously moving fronts with the programs available. The quasi-stationary state approximation looked promising for these calculations, but the programs based on the model were not available either. For these reasons, CHEMFRONTS was developed.

2.4 The quasi-stationary state approximation

The quasi-stationary state approximation (Lichtner, 1988) describes the evolution of geochemical processes, by including advective, diffusive and dispersive mass transport, in a sequence of stationary states. The theory is based on the very slow changes in the physical quantities of the minerals compared to the flow rate, and on the species having a much lower concentration in the water than in the minerals. This leads to nearly stationary states within a single volume of fluid. When the quantity of the components in the solution is very small, compared to that in the mineral phases, the volume of water that must flow through the system to dissolve a substantial amount of a mineral is very large in relation to the volume of the column and its minerals. It is then often possible to neglect the local accumulation of species in the liquid. Because the changes in the host rock take longer time than is required for the fluid composition to establish a stationary state, the formation of a stationary state may be considered to be fast or even instantaneous.

The mineral reactions are described by kinetic rate laws for both precipitation and dissolution. This has several advantages over a local equilibrium formulation. In a local equilibrium description of mineral reactions, it is necessary to use trial and error methods to determine the correct sequence of mineral reaction products. If mineral reactions are described by pseudo-kinetic-rate expressions, when more accurate rate laws are not available, the stringent conditions of local equilibrium may be relaxed. The sequence of mineral reaction products is then determined directly from the transport equations without the need for trial and error.

A single volume of water that flows through the column reacts with the solid phase, dissolves parts of the mineral until the water volume is saturated with the existing minerals, and forms new minerals that precipitate if supersaturation is reached. The reactions of a subsequent volume of water are similar to the previous ones, as the changes in the mineral phase are small. The time step size depends on the changes in reactivity in the solid phase. It is the rate of the changes that determines the life-time of the stationary state. Provided the changes in the solid phase are slow compared to the time needed to establish the stationary state, the quasi-stationary state approximation holds. The time step size is not limited by numerical stability but by the lifetime of the stationary states. Geological-scale periods can thus be simulated for complicated systems.

3 MATHEMATICAL MODEL

3.1 Development of model equations

This model uses matrix notation to represent the chemical reactions of the complexes and minerals. One line in the matrix defines each reaction. The formation of the complexes is expressed by

$$\sum_{j=1}^{N} \nu_{ij} A_j = A_i \qquad (3.1.1)$$

where j is the number of the component, i is the number of the complex, ν_{ij} is the stoichiometric coefficient, A_j is the component and A_i is the complex. For example, in the reaction

$$Ca^{2+} + 2H_2O \rightarrow Ca(OH)^+ + H^+$$

the component Ca^{2+} is $A_{(j=1)}$, H^+ is $A_{(j=2)}$, and the complex $Ca(OH)^+$ is $A_{(i=1)}$. Then ν_{11} is 1 and ν_{12} is -1. Water is not considered to be a component. The surplus of water is large and its concentration does not change much.

A similar expression is used for the minerals

$$\sum_{j=1}^{N} \nu_{mj} A_j = A_m \qquad (3.1.2)$$

where m is the number of the mineral, and ν_{mj} is the stoichiometric coefficient for the mineral m and the component j.

A mass balance for the system gives

$$\frac{\partial}{\partial t}(\phi Y_j) + \nabla \mathbf{W}_j = -\sum_{m=1}^{M} \nu_{mj} \frac{\partial X_m}{\partial t} \qquad (j = 1, \ldots, N) \qquad (3.1.3)$$

where the first term is the amount of component j that has accumulated in the system,

ϕ is the porosity, Y_j is the total aqueous concentration of component j, $\nabla \mathbf{W}_j$, is the transport of the component j by fluid flow and diffusion, M is the number of minerals, X_m is the concentration of mineral m in the solid phase, and t is the time.

The term Y_j refers to the concentration of component j both as a free component and in any complex. It can be found from equation (3.1.6) below. The right-hand side of the equation is the sum of the amounts of component j transferred from the mineral to the aqueous phase by dissolution of minerals (negative for precipitation).

When the amount of compounds in the solution is very small compared to that in the mineral phases, the volume of water that must flow through the system to dissolve a substantial amount of a mineral is very large compared to the volume of the column and its minerals. Then the accumulation of the species in the system can usually be ignored, and $\frac{\partial}{\partial t}(\phi Y_j)$ be taken as zero without any substantial loss in accuracy. Equation (3.1.3) then becomes

$$\nabla \mathbf{W}_j = -\sum_m \nu_{mj} \frac{\partial X_m}{\partial t} \quad (j = 1, \ldots, N) \tag{3.1.4}$$

The volume change of the minerals in the column is

$$\frac{\partial}{\partial t}(\phi_m V_m^{-1}) = \frac{\partial X_m}{\partial t} \quad (m = 1, \ldots, M) \tag{3.1.5}$$

where ϕ_m is the volume fraction of mineral m, and V_m is the molar volume of mineral m.

The total concentration of component j, Y_j, is the sum of all aqueous forms of j

$$Y_j(\mathbf{r},t) = C_j(\mathbf{r},t) + \sum_i \nu_{ij} C_{xi}(\mathbf{r},t) \tag{3.1.6}$$

where C_j is the free concentration of component j, C_{xi} is the free concentration of complex i, and $\mathbf{r}$ is the distance from the column inlet. The free concentration of the complexes is calculated from the assumption that the aqueous species are always in local equilibrium

$$C_{xi}(\mathbf{r},t) = K_{xi}\, \gamma_{xi}^{-1}(\mathbf{r},t) \prod_{j=1}^{N} (\gamma_j(\mathbf{r},t)\, C_j(\mathbf{r},t))^{v_{ij}} \tag{3.1.7}$$

where K_{xi} is the equilibrium constant for complex i, γ_{xi} is the activity coefficient of complex i, and γ_j is the activity coefficient of component j.

The flux of mass in the system, $\mathbf{W}_j$, is related to the sum of the fluxes of the components, $\mathbf{J}_j$, and the complexes, $\mathbf{J}_i$

$$\mathbf{W}_j(\mathbf{r},t) = \mathbf{J}_j(\mathbf{r},t) + \sum_i v_{ij}\, \mathbf{J}_i(\mathbf{r},t) \tag{3.1.8}$$

The flux of any component or complex in the aqueous solution, $\mathbf{J}_l$ (where l represents any i or j), is the sum of the diffusive flux (the first term on the right-hand side of equation (3.1.9)), and the advective flux (the last term in equation (3.1.9))

$$\mathbf{J}_l(\mathbf{r},t) = -\,\phi(\mathbf{r},t)\,(\sum_k D_{lk}(\mathbf{r},t)\, \nabla C_k) + \mathbf{v}(\mathbf{r},t)\, C_l(\mathbf{r},t) \tag{3.1.9}$$

where D_{lk} is the diffusion coefficient for species l in relation to species k, and k represents both the complexes i and the components j. The advective flux is the product of the flux of water, v, and the free concentration of species l.

The total reactive volume fraction occupied by water and minerals, ϕ_R, is the sum of the porosity, ϕ, and the volume fraction of the minerals

$$\phi_R = \phi(\mathbf{r},t) + \sum_{m=1}^{M} \phi_m(\mathbf{r},t) \tag{3.1.10}$$

where

$$0 \leq \phi_m(\mathbf{r},t) \leq \phi_R \tag{3.1.11}$$

The mineral precipitation or dissolution, $\frac{\partial X_m}{\partial t}$, rate is given by

$$\frac{\partial X_m}{\partial t}(\mathbf{r},t) = \zeta_m(\mathbf{r},t)\, I_m(\mathbf{r},t) \tag{3.1.12}$$

where

$$\zeta_m(\mathbf{r},t)=\begin{cases} 1 \ (\phi_m(\mathbf{r},t) \neq 0 \text{ or } I_m(\mathbf{r},t) \ > 0 \\ 0 \ \ (\text{otherwise}) \end{cases} \tag{3.1.13}$$

where ζ_m is a logical factor, which is unity both if the solution is supersaturated and precipitation is possible, and if minerals are present so that dissolution is possible. Otherwise, ζ_m is zero.

The rate of dissolution or precipitation, I_m, is

$$I_m(\mathbf{r},t) = \alpha_m(\mathbf{r},t)k^f_m(Q_m(\mathbf{r},t)-K^{-1}_m) \tag{3.1.14}$$

It is the product of the specific surface of the mineral, α_m, the mineral reaction rate, k^f_m, and the driving force of the system $(Q_m(\mathbf{r},t)-K^{-1}_m)$. The driving force is the difference between the ion activity product of the water solution, Q_m, and the ion activity product at saturation (the inverse of the equilibrium constant of the formation for the mineral m)

$$Q_m(\mathbf{r},t) = \prod_{j=1}^{N} (a_j(\mathbf{r},t))^{v_m} \tag{3.1.15}$$

The activity of the ions is the product of the activity coefficient, γ_j, and the free concentration of component j

$$a_j(\mathbf{r},t) = \gamma_j(\mathbf{r},t)C_j(\mathbf{r},t) \tag{3.1.16}$$

The dissolution and precipitation rate divided by the porosity is equal to the sum of the diffusive (dispersive) flux and the advective flux

$$D_L \frac{d^2Y_j}{dz^2} - \frac{v}{\phi}\frac{dY_j}{dz} = \frac{1}{\phi}\sum_m v_{mj}\frac{\partial X_m}{\partial t}(z) \qquad (3.1.17)$$

When there is a large advective flux, the diffusive flux can be small by comparison. In this model the dispersive flux is assumed to be negligible. Therefore the first term of equation (3.1.17) is approximately zero, which gives

$$v\frac{dY_j}{dz} = -\sum_m v_{mj}\frac{\partial X_m}{\partial t}(z) \qquad (3.1.18)$$

The gradient $\frac{dY_j}{dz}$ can be expressed in terms of C_{xi} and C_j

$$\frac{dY_j}{dz} = \frac{dC_j}{dz} + \sum_i v_{ij}\frac{dC_{xi}}{dz} \qquad (3.1.19)$$

where

$$\frac{dC_{xi}}{dz} = \sum_k \frac{dC_{xi}}{dC_k}\frac{dC_k}{dz} = \sum_k v_{ik}\frac{C_{xi}}{C_k}\frac{dC_k}{dz} =$$

$$= v_{i1}\frac{C_{xi}}{C_1}\frac{dC_1}{dz} + v_{i2}\frac{C_{xi}}{C_2}\frac{dC_2}{dz} \cdots v_{ik}\frac{C_{xi}}{C_k}\frac{dC_k}{dz} \qquad (3.1.20)$$

where k refers to the components. Substituting equation (3.1.20) into (3.1.19) for j=1 gives

$$\frac{dY_1}{dz} = \frac{dC_1}{dz} + v_{11}v_{11}\frac{C_{x1}}{C_1}\frac{dC_1}{dz} + v_{11}v_{12}\frac{C_{x1}}{C_2}\frac{dC_2}{dz} + \cdots$$

$$\cdots + v_{11}v_{1j}\frac{C_{x1}}{C_j}\frac{dC_j}{dz} + v_{21}v_{21}\frac{C_{x2}}{C_1}\frac{dC_1}{dz} + v_{21}v_{22}\frac{C_{x2}}{C_2}\frac{dC_2}{dz} + \cdots$$

$$\cdots + v_{i1}v_{i(j-1)}\frac{C_{xi}}{C_{(j-1)}}\frac{dC_{(j-1)}}{dz} + v_{i1}v_{ij}\frac{C_{xi}C_j}{C_j\,dz} \qquad (3.1.21)$$

This can be expressed as

$$\frac{dY_1}{dz} = \left(1 + v_{11}v_{11}\frac{C_{x1}}{C_1} + v_{21}v_{21}\frac{C_{x2}}{C_1} + \cdots + v_{i1}v_{i1}\frac{C_{xi}}{C_1} \right) \frac{dC_1}{dz} +$$

$$+ \left(v_{12}v_{12}\frac{C_{x1}}{C_2} + v_{22}v_{22}\frac{C_{x2}}{C_2} + \cdots + v_{i2}v_{i2}\frac{C_{xi}}{C_2} \right) \frac{dC_2}{dz} + \cdots$$

$$\cdots + \left(v_{1j}v_{1j}\frac{C_{x1}}{C_j} + v_{2j}v_{2j}\frac{C_{x2}}{C_j} + \cdots + v_{ij}v_{ij}\frac{C_{xi}}{C_j} \right) \frac{dC_j}{dz} \qquad (3.1.22)$$

The system of equations has the form

$$\overline{\frac{dY}{dz}} = \overline{\overline{A}}\,\overline{\frac{dC}{dz}} \qquad (3.1.23)$$

where A is a square matrix whose size is equal to the number of components. When k = j (the diagonal elements), then

$$a_{kj} = 1 + \sum_i v_{ik}v_{ij}\frac{C_{xi}}{C_j} \qquad (3.1.24)$$

when k ≠ j then

$$a_{kj} = \sum_i v_{ik}v_{ij}\frac{C_{xi}}{C_j} \qquad (3.1.25)$$

Substituting (3.1.23) into equation (3.1.18) gives

$$\overline{\overline{A}}\,\overline{\frac{dC}{dz}} = -\frac{1}{v}\sum_m v_{mj}\frac{\partial X_m}{\partial t}(z) = \text{a function of } \overline{C} \qquad (3.1.26)$$

where the second term must be interpreted as a column vector. This is the system of equations to be solved, with the appropriate boundary conditions.

3.2 The sensitivity of the system to the reaction rate constant

In a system of minerals where some dissolve and others precipitate, the rate of change depends on the reaction rates for dissolution and precipitation. Reaction rates are very seldom known. Furthermore, they are fast in relation to slowly evolving geochemical systems where mass transport dominates. As long as the reaction rate is fast, the frontal velocity is independent of the reaction rate of the mineral, as will be demonstrated below. This makes it possible to use generalized rates to determine the velocities of the fronts when the real rates are unknown. From equation (3.1.12) and (3.1.14) the reaction rate for a mineral can be written generally as

$$\frac{dX_1}{dt} = k_1(C_1-K_1) \qquad (3.2.1)$$

where $\frac{dX_1}{dt}$ is the dissolution or precipitation rate of mineral 1, and C_1 is the free concentration of component 1. K_1 is the equilibrium constant written in a different form to the earlier expression, $K_1 = K_m^{-1}$ if K_m is from equation (3.1.14) where K_m is the equilibrium constant for formation of the mineral while K_1 is the equilibrium constant for dissolution of the mineral. It can be assumed that k_1 is constant. Provided a pseudo steady state[2] has been attained, the concentration change with distance (z) is

$$\frac{dC_1}{dz} = -\frac{1}{v}\cdot\frac{dX_1}{dt} \qquad (3.2.2)$$

where v is the flow rate of the water. The front velocity v_f is obtained by integrating the rate of dissolution of mineral over the very long column which gives the total change of mass of mineral 1. This is equal to the change from the original concentration X_0 of the mineral times the rate of the frontal movement v_f

$$v_f = -\frac{1}{X_o}\cdot\int_0^\infty \frac{dX_1}{dt}dz \qquad (3.2.3)$$

Substituting equation (3.2.1) into (3.2.2) gives a new expression for the concentration profile (concentration change with distance)

[2]The pseudo steady state is discussed in 2.4

$$\frac{dC_1}{dz} = -\frac{k_1}{v}(C_1-K_1) \tag{3.2.4}$$

Integration gives

$$\ln(C_1-K_1) = -\frac{k_1}{v}\cdot z + A \tag{3.2.5}$$

or

$$C_1-K_1 = \mathrm{const}\cdot e^{-\frac{k_1}{v}z} \tag{3.2.6}$$

When the concentration of component C_1 is zero (at the inlet, where $z = 0$), the constant becomes $-K_1$. Equation (3.2.6) then becomes

$$C_1 = K_1(1-e^{-\frac{k_1}{v}z}) \tag{3.2.7}$$

Substitution into equation (3.2.1) gives

$$\frac{dX_1}{dt} = -k_1K_1e^{-\frac{k_1}{v}z} \tag{3.2.8}$$

This can be used with equation (3.2.3) to get the velocity of the front

$$v_f = \frac{k_1K_1}{X_o}\cdot\int_0^\infty e^{-\frac{k_1}{v}z}dz = \frac{k_1K_1}{X_o}\left(-\frac{v}{k_1}\right)\left[e^{-\frac{k_1}{v}z}\right]_0^\infty \tag{3.2.10}$$

$$v_f = \frac{K_1v}{X_o} \tag{3.2.11}$$

Equation (3.2.11) shows that the front velocity is independent of the reaction rate, once the "pseudo" steady state is reached and when the reaction $\frac{dX}{dt}$ becomes “zero” within the practical bounds of the column, i.e. for $z < \infty$.

4 THE COMPUTER PROGRAM CHEMFRONTS

CHEMFRONTS is a computer program written in FORTRAN77. It consists of a short main program and many small subroutines. This makes it easy to modify. The main structure is shown in figure 4.1.

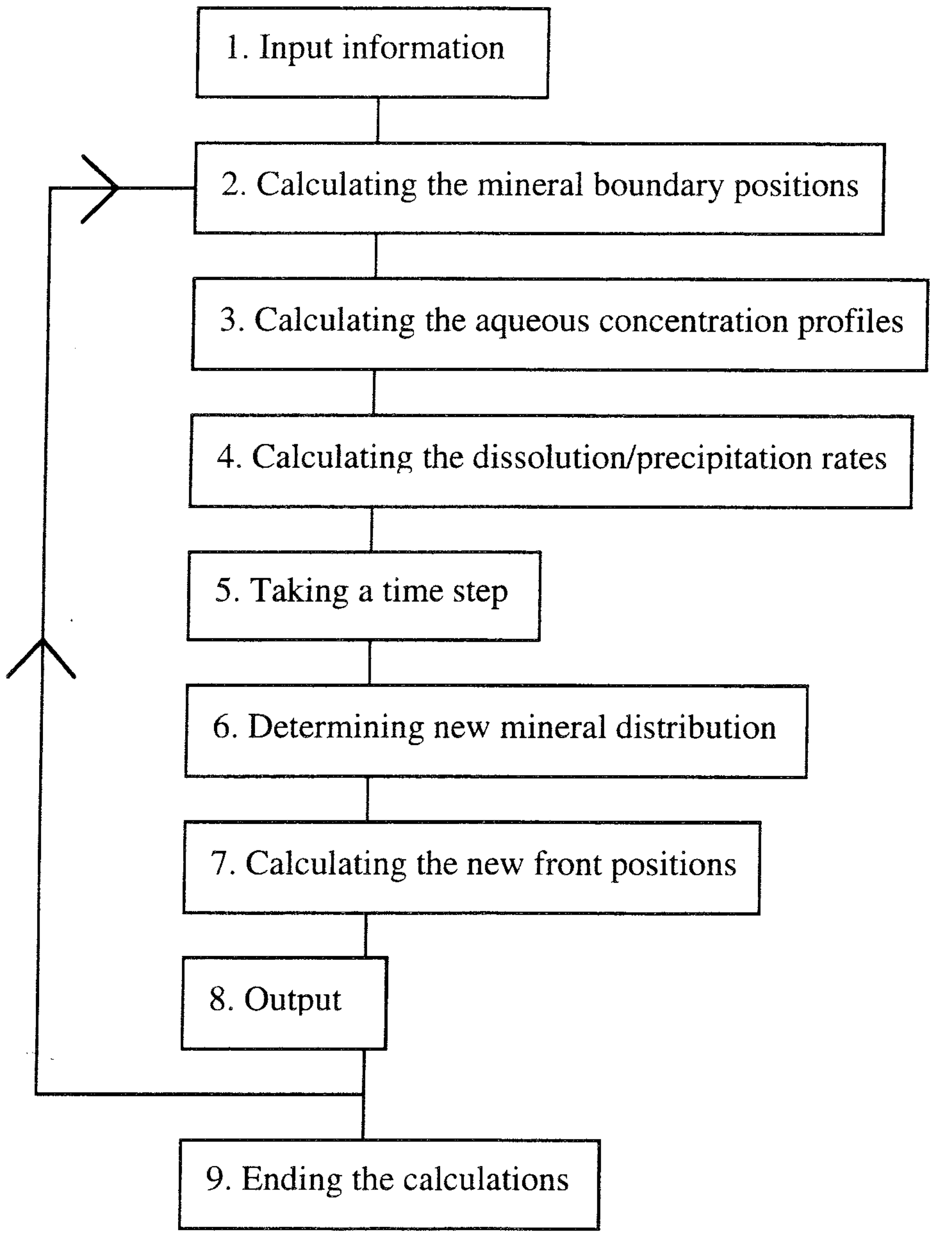

1. Input information
2. Calculating the mineral boundary positions
3. Calculating the aqueous concentration profiles
4. Calculating the dissolution/precipitation rates
5. Taking a time step
6. Determining new mineral distribution
7. Calculating the new front positions
8. Output
9. Ending the calculations

Figure 4.1 The structure of CHEMFRONTS

The calculation procedure is outlined below, with some examples.

4.1 Input information

Initially, the program starts by reading the two input files *exinput* and *ode.dat*. *Exinput* includes data on the chemical problem, input concentration of the mineral, porosity, concentration of the inflowing water, thermodynamic data on the minerals and the complexes, flow rate. *Ode.dat* contains information about the calculation, such as maximum deviation for different calculations, or if it is a restart of the calculation. The input files are extensively described in the user´s guide.

4.2 Calculating the mineral boundary positions

The column consists of several regions with different minerals. Within a region, the mineral concentrations may differ but the minerals are the same. Figure 4.2.1 shows a column with four minerals and two boundaries, in addition to the boundary at the inlet.

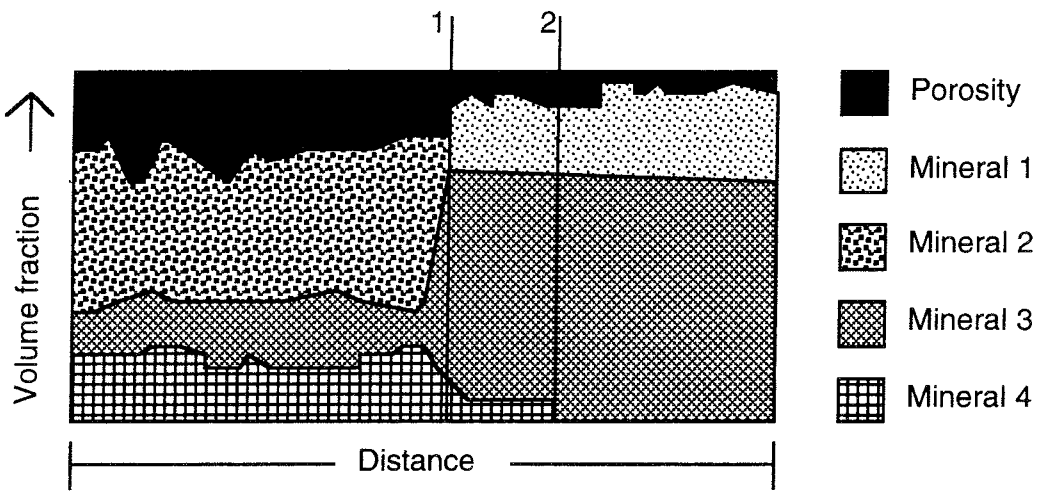


Figure 4.2.1 Column with four minerals and two boundaries.

4.3 Calculating the aqueous concentration profile

When the boundaries have been established, the concentration profile can be calculated by solving the system of differential equations (3.1.26) for the components, for every region having the same minerals. The computer program CHEMFRONTS uses a Gear package solver, the SDRIV2 package (Kahaner et al., 1989). The package is modified for this model. Figure 4.3.1 shows a concentration profile from section 5.4 below. Three boundaries are shown. The first is at the inlet, where some solid phase containing silica is dissolved. The second is 0.15 metres from the inlet, where a potassium-containing mineral is dissolved. The third boundary is 0.9 metres from the inlet, where sulphur is dissolved. At the fronts, many more reactions take place that are not shown in this graph. Changes in the silica and potassium concentration are shown at the second and the third boundary respectively. This indicates two dissolution fronts for the same mineral. In this example they are caused by pH changes. (They could also be caused by two different minerals containing potassium and silica.)

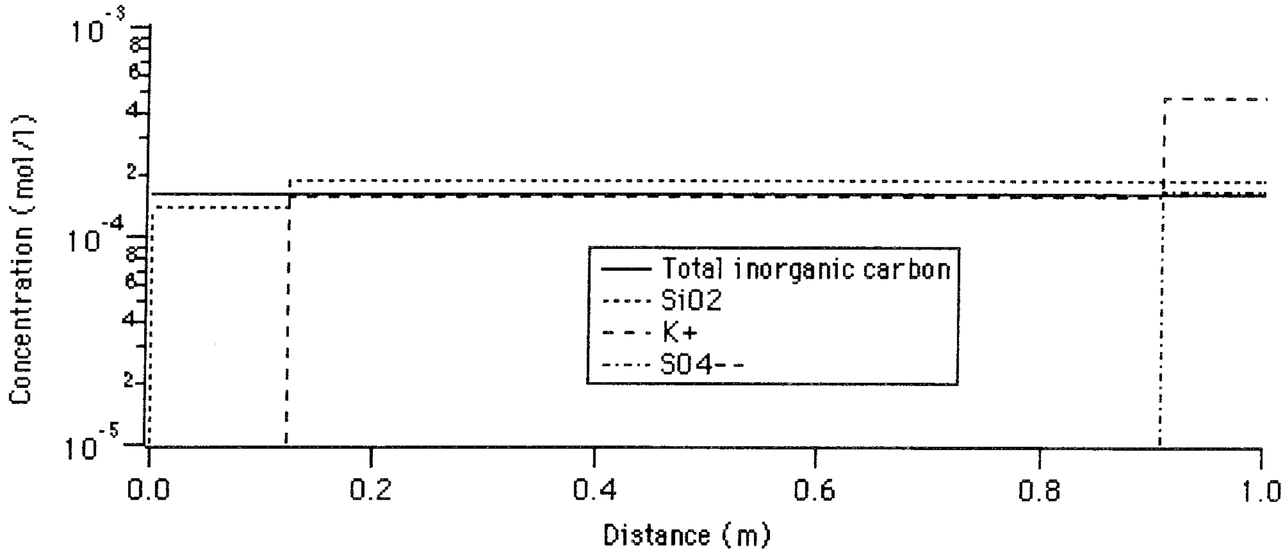


Figure 4.3.1 The concentration profile of the aqueous species.

4.4 Calculating the dissolution/precipitation rates

The mineral dissolution and precipitation profiles are calculated from equations (3.1.12) - (3.1.16). The dissolution profile can look like figure 4.4.1, from the example in section 5.4, where K-feldspar ($KAlSi_3O_8$) dissolves and chalcedony (SiO_2) and kaolinite ($Al_2Si_2O_5(OH)_4$) precipitate according to the following reaction

$$KAlSi_3O_8\ (s) + H^+\ (aq) + 0.5\ H_2O \rightarrow$$
$$K^+\ (aq) + 2\ SiO_2\ (s) + 0.5\ Al_2Si_2O_5(OH)_4\ (s) \qquad (4.4.1)$$

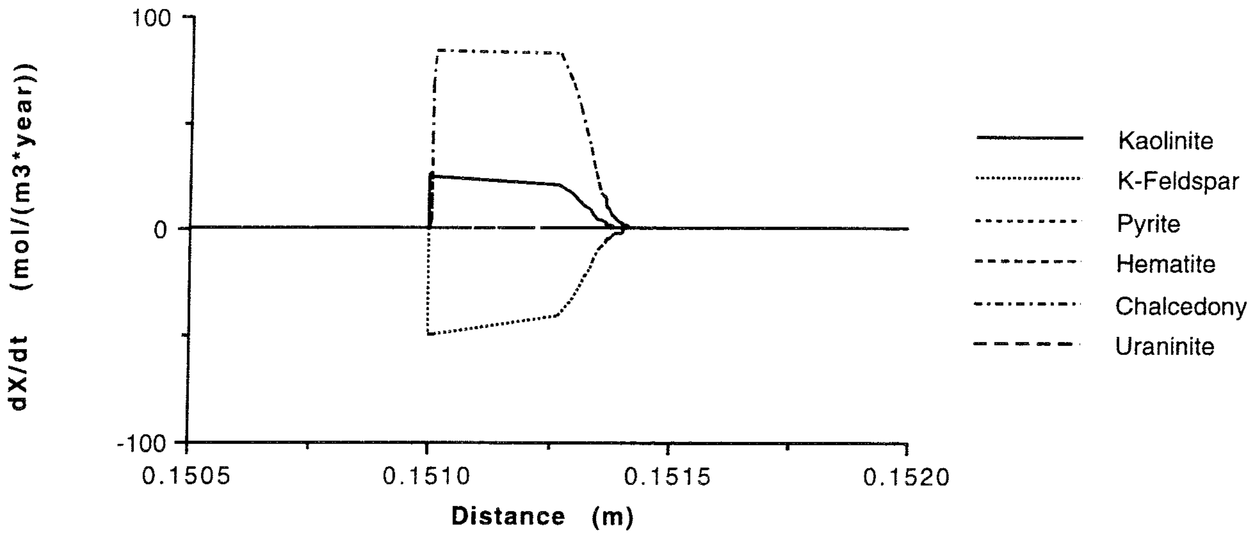


Figure 4.4.1 The dissolution front of K-feldspar.

When water comes into contact with the K-feldspar, the latter starts to dissolve at a maximum rate of 50 moles/(m^3·year) assumed in this example, according to the kinetic dissolution coefficient, k_m^f, in equation (3.1.14). This causes supersaturation of chalcedony and kaolinite, which thus precipitate. According to reaction (4.4.1), every dissolved K-feldspar produces two chalcedony and half a kaolinite, so their precipitation rates should be 100 and 25 moles/(m^3·year) respectively. Because of the high solubility of silica, the precipitation of chalcedony is less than 100 moles/(m^3·year). Some silica is carried away by the water.

4.5 Taking a time step

The column is divided into a large number of "slices". The number and location of the slices may change with every time step. The evolution of the column is calculated stepwise. For every slice in the column, the time needed to dissolve all of a mineral is calculated from the mineral concentration and the mineral dissolution rate

$$\text{Dissolution time} = \frac{\text{Mineral concentration}}{\text{Mineral dissolution rate}} \tag{4.5.1}$$

The minimum dissolution time, greater than zero, gives the size of the time step for this calculation. The overall prediction time is increased by the time step for every calculation.

4.6 Determining the new mineral distribution

The new mineral distribution is calculated from the dissolution and precipitation profiles and the time step. The concentration profiles of the aqueous species and the mineral dissolution and precipitation profiles are assumed to be constant during the time step. The mineral concentration at each slice increases or decreases with the amount dissolved or precipitated during a time step.

Consider a slice 0.15101 metres from the inlet. The mineral precipitation and dissolution rate at this slice can be seen in in figure 4.4.1. The changes in mineral concentrations are shown in table 4.6.1.

Table 4.6.1 The mineral concentrations for a time step.

	K-feldspar	Chalcedony	Kaolinite
Mineral concentration before the time step (mol/m^3)	432	120	35
Mineral reaction mol/(m^3·year)	-50	86	24
Mineral reaction for a time step of 0.5 years	-25	43	12
Mineral concentration after the time step (mol/m^3)	407	163	47

These calculations are performed for every slice and every mineral in the system. This gives the new mineral distribution within the column.

4.7 Calculating the new front position

From the mineral profiles, the positions of the mineral dissolution front are calculated. The position of the fronts can be used to calculate the front rate by plotting the front position versus the time, as in figure 4.7.1, taken from the example in section 5.4.

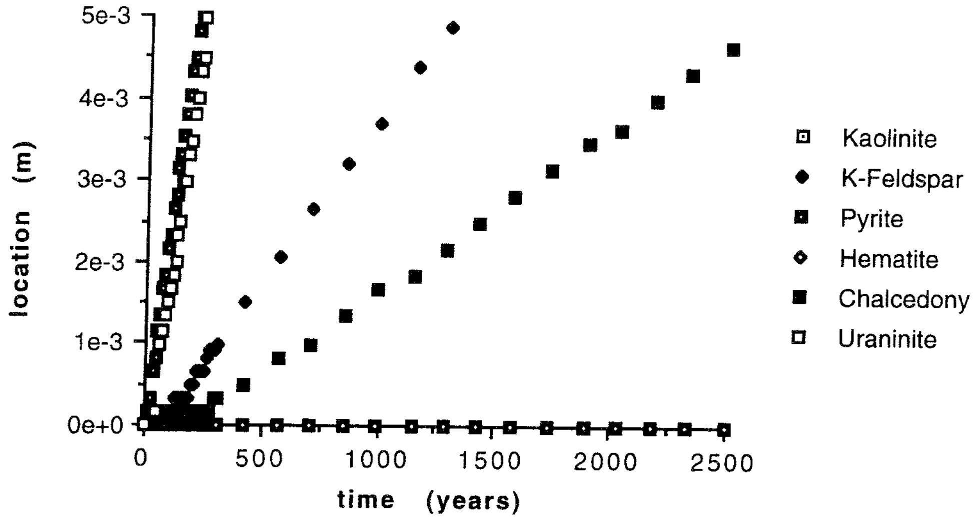


Figure 4.7.1 Front position versus time.

Figure 4.7.1 shows the front position versus time for six minerals. These with high front velocities separate from the others early in the calculations, and the slower fronts separate later. In figure 4.7.1 the fastest fronts are the pyrite and uraninite fronts. They are coupled and will not separate, as explained in section 5.3. The K-feldspar and chalcedony fronts are well separated from the others, but kaolinite and hematite have not separated at all. When the fronts have separated from each other, they move with a constant velocity. When the reaction rates and reactive surfaces do not change with mineral concentration, the straight lines can be extrapolated indefinitely without additional calculations.

4.8 Output

The output is written in various files for different data. The User's Guide gives a more extensive description of the output files. The output is not written into the files for every time step, but the intervals are chosen as an input parameter. The reason for this option is that the output files are lengthy and will fill even a large disk if there are many time steps. It is possible, however, to choose the output for every time step.

4.9 Ending calculations

The calculation of the problem is finished when the column is out of mineral or when the requested number of time steps has been accomplished. If some of the fronts have not had time to stabilize, the calculation can be continued. In figure 4.7.1, the K-feldspar, pyrite, chalcedony and uraninite fronts have separated but the kaolinite and hematite fronts have not. The computations can be continued without the separated fronts by making the column shorter, so that the separated fronts have already moved out of the column. The calculations are much faster when the concentration profiles are only calculated for the first fronts. The time steps will also increase, as the fast reactions often determine the time step size, see section 4.5.

4.10 Some properties of the program

The time stepping used in the program sometimes causes abruptness in the mineral profiles. This does not affect the front velocity or other results.

One example of this is the dissolution of pyrite (FeS_2). Initially there is a homogeneous column with 25 moles of pyrite. When the pyrite dissolves some ferric-oxy-hydroxide ($Fe(OH)_3$), FOH for short, will precipitate as follows

$$FeS_2\ (s) + 3.75\ O_2\ (aq) + 3.5\ H_2O\ (l) \rightarrow 4\ H^+\ (aq) + 2\ SO_4^{2-}\ (aq) + Fe(OH)_3\ (s) \qquad (4.10.1)$$

The mineral dissolution and precipitation profiles are shown in figure 4.10.1.

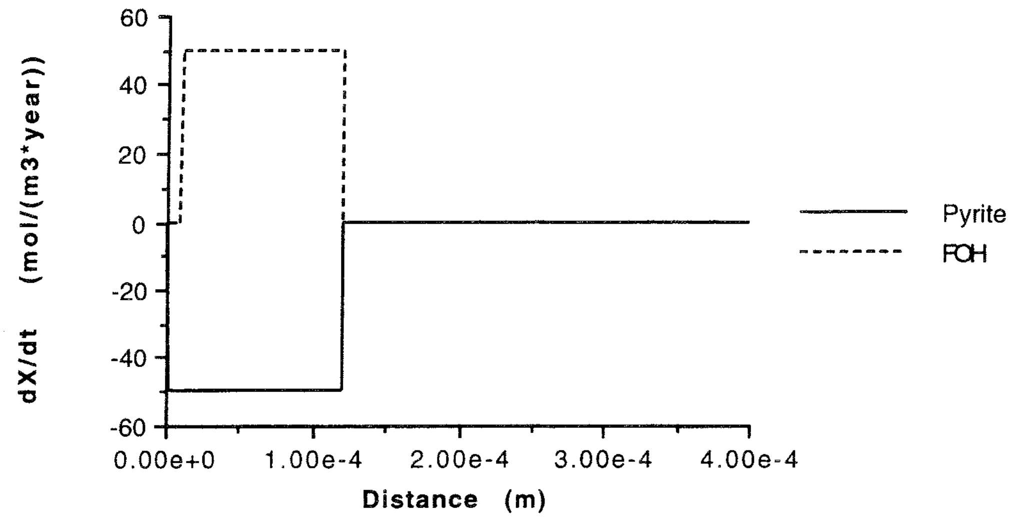


Figure 4.10.1 The dissolution and precipitation rate profiles for pyrite and FOH.

Figure 4.10.1 shows that the dissolution of pyrite starts when the water containing oxygen comes in contact with the pyrite. On the other hand, the FOH precipitation starts when the water solution is supersaturated with FOH. This delay in the precipitation causes the irregularities in the mineral profiles. The mineral profiles after a time step of 0.5 years are shown in figure 4.10.2.

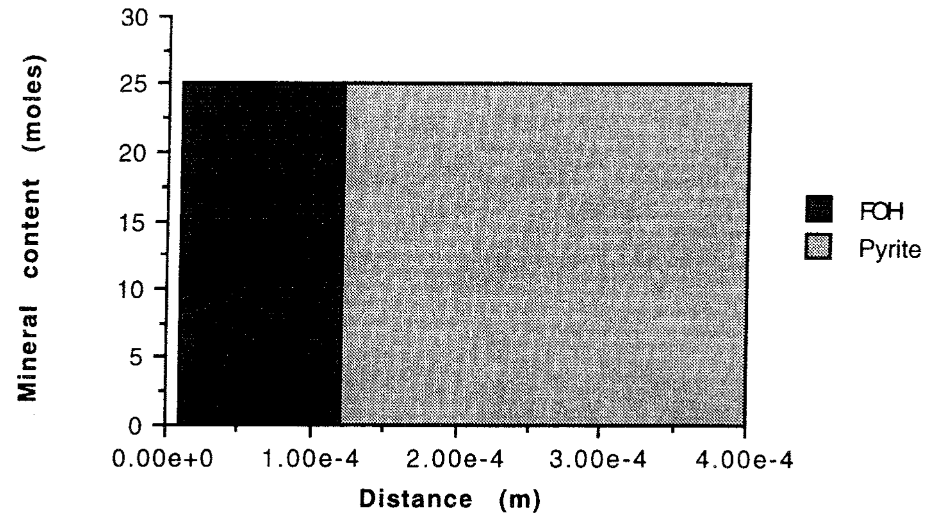


Figure 4.10.2 The mineral concentration profiles after 0.5 years.

The mineral dissolution and precipitation profiles during the second time step are shown in figure 4.10.3.

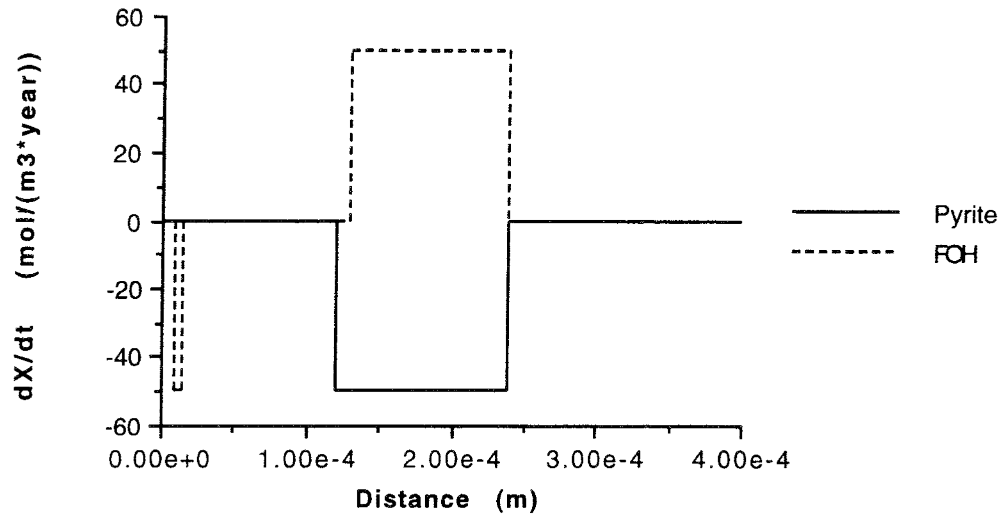


Figure 4.10.3 The dissolution and precipitation rate profiles for pyrite and FOH for the second time step.

During this time step the FOH begins to dissolve at the beginning of the column. The mineral profiles after the second time step are shown in figure 10.4.4.

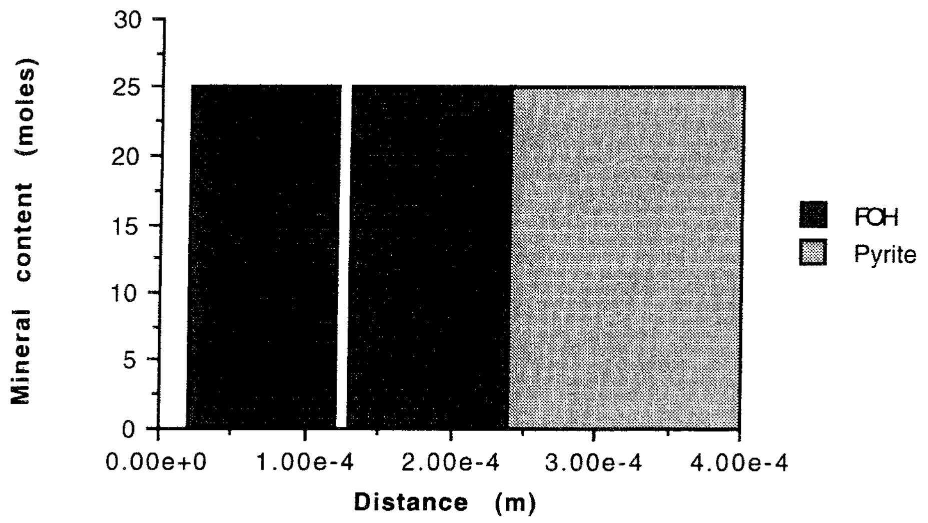


Figure 4.10.4 The mineral concentration profiles after two time steps.

The mineral concentration profiles after four time steps are shown in figure 4.10.5.

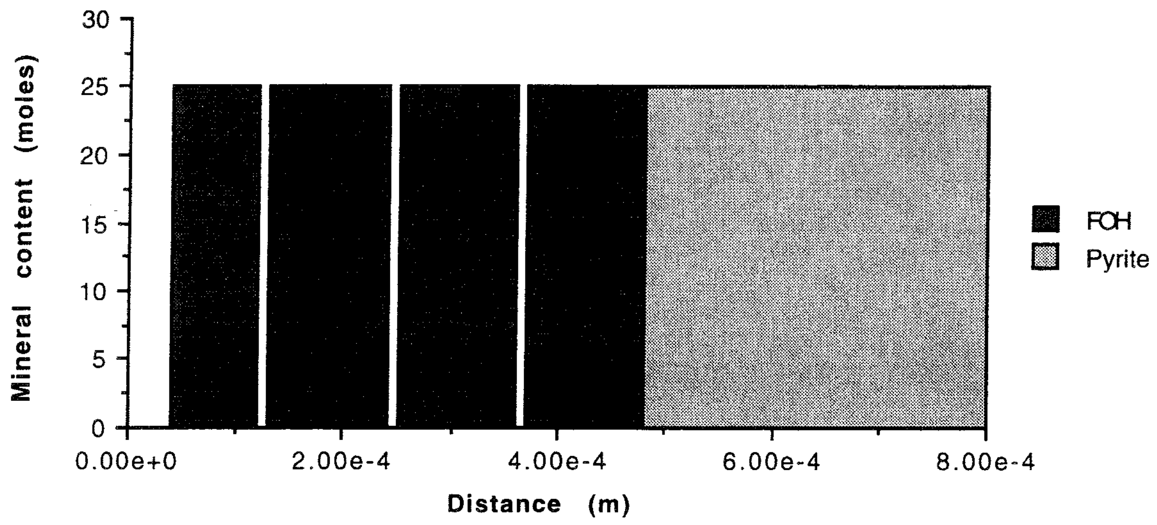


Figure 4.10.5 Mineral concentration profiles after four time steps.

When the time step size depends on more than one mineral, the profiles are more complicated. The irregularities are caused by the time stepping procedure. If the time step size approaches zero the irregularities will disappear, but the mathematical problem will be much more complicated and more difficult to solve. The computation time will probably increase by several orders of magnitude. The irregularities do not affect the results for the rate of the front movement or the liquid concentration profiles.

Another way to approach the problem is to average the mineral composition in various areas. The column was divided into several cells, and the average mineral concentration in each cell was calculated. This frequently gave satisfactory results, but not with a mixture of kaolinite, gibbsite and quartz. If there is initially quartz and kaolinite in a cell and the quartz starts to dissolve, the kaolinite will dissolve as well and cause the gibbsite to precipitate, according to reaction (4.10.2) below. One kaolinite contains the same components as one quartz plus one gibbsite, and gibbsite is much less soluble than kaolinite. The result will be gibbsite, kaolinite, and quartz within that cell. When the mineral concentration in the cell is averaged this will give a homogeneous mixture of kaolinite, gibbsite and quartz. The latter two do not coexist at equilibrium. During the next time step there will be a stationary state where gibbsite and quartz produce kaolinite.

$$\text{Kaolinite (s)} \rightarrow 2\ \text{Gibbsite (s)} + 2\ \text{Quartz (aq)} \qquad (4.10.2)$$

Various ways have been tried to average the mineral concentrations and avoid the irregular mineral profiles, but it is a risk to smooth out the profiles too much and lose accuracy. The irregularities of the mineral profiles do not affect the results of the calculations, only the detailed shapes of the profiles.

The example in section 5.4 describes how a small amount of uranium moves with the redox front, because the solubility of uranium is high in the oxidized region and low in the reduced region. The reactions taking place are

$$U(OH)_4\ (s) + 0.5\ O_2\ (aq) + H_2O \rightarrow U(VI)\ (aq) + 6\ OH^-\ (aq) \qquad (4.10.3)$$

$$U(VI)\ (aq) + 2\ \text{Fe(II)-mineral}\ (s) + 4\ OH^-\ (aq) \rightarrow U(OH)_4\ (s) + 2\ Fe(III)\ (aq) \qquad (4.10.4)$$

When the water containing oxygen reaches the uraninite ($U(OH)_4$), the uranium(IV) is oxidized to uranium(VI) by reaction (4.10.3). The soluble uranium(VI) is then transported by the flowing water to the iron(II)-mineral where the uranium(VI) is reduced to uranium(IV) and precipitated as uraninite by reaction (4.10.4). The precipitation rate is greater than the dissolution rate and therefore the uranium precipitates within a small region. This causes a mineral profile like that in figure 4.10.6.

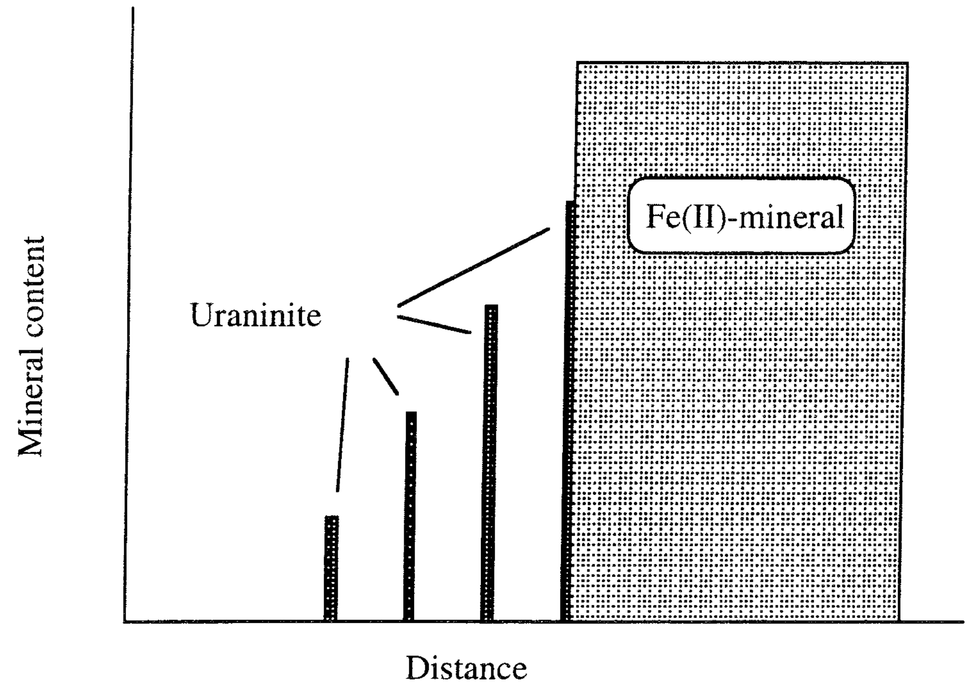


Figure 4.10.6 The mineral distribution at the redox front.

When the water flows through the first slice containing uranium, some of the uranium dissolves. Because the dissolution rate is slow compared to the time taken to flow through the thin slice with uranium, the water is not saturated after the first slice. When the water comes into contact with the Fe(II)-mineral, uranium will have been dissolved from several areas.

CHEMFRONTS usually does not calculate the concentration profile for every time step, see section 4.3. When the minerals are the same for a front, the concentration profiles from the previous calculations are used. When the dissolution fronts are separated, see section 4.7, the old concentration profiles can be used, but starting at another point.

In this uranium dissolution problem the mineral profiles change continuously, because the dissolution fronts for the Fe(II)-mineral and uraninite are coupled. This makes it difficult to use the calculations from the previous time step, so the concentration profiles have to be calculated for each time step. The computing time then becomes very long.

4.11 Limitations of the current program

4.11.1 Porosity calculations

In the current version of the program CHEMFRONTS, there are no porosity calculations. The porosity may even be negative. In the example described in section 5.5, the concentration of the incoming water was taken from a field experiment. The water turned out to be supersaturated for some of the minerals involved. This results in an accumulation of these minerals at the inlet, and thus negative porosity.

4.11.2 Reaction rate

The reaction rate constant for a mineral depends on the mineral surface area. The active surface area of a pure mineral can be measured experimentally. A mixture of minerals behaves differently to a pure mineral. If a mineral dissolves and another mineral is formed, the new mineral may precipitate on the old mineral and thereby reduce the surface area of the old mineral.

Such behavior of the minerals cannot be predicted at present, and so some approximation is necessary. Lichtner (1988) assumes the minerals are spheres and that the active surface area decreases with decreasing mineral concentration. Another assumption is that the porosity is due to thin channels. This approximation gives an increasing specific surface area with decreasing mineral concentration.

As long as the mineral reaches equilibrium within a short distance compared to the column length, the results are independent of the reaction rate constant, see section 3.2. The reaction rate constants in CHEMFRONTS are thus taken to be independent of the mineral concentrations, as long as the mineral is present. The advantage of this is that the results can be extrapolated, see section 4.7. This is not possible with the sphere or channel approximations.

4.11.3 Activity coefficients

To calculate the driving forces, the activities of the various species are used. In CHEMFRONTS, all activity coefficients are approximated to unity. In diluted solutions, the activity coefficients are close to unity. In CHEMFRONTS, there is a subroutine named *ACTCOEFF* that returns the value of the activity coefficient to unity. This makes it easy to insert a routine to calculate the actual activity coefficients.

4.11.4 Local equilibrium in the water phase

In the program CHEMFRONTS, the aqueous species are assumed to be in equilibrium. As the complexation reactions in solution are fast compared to the reactions between solid and liquid phases, this approximation is reasonable. However, there may be slow reactions in the water phase that this program cannot account for.

4.11.5 Diffusion and dispersion

CHEMFRONTS calculates for advective flow without dispersion. No diffusion is incorporated in the program. This limits the usefulness of the program to advection-controlled problems.

5 COMPARISON OF RESULTS FROM OTHER PROGRAMS

5.1 A four-component dissolution problem

Liu and Narasimhan (1989b) compared results from their program DYNAMIX (Liu and Narasimhan, 1989a) with the results from the program PHASEQL/FLOW (Walsh et al., 1984). The test case is a one-dimensional column with one solid phase, AB, which dissolves into A and B with a solubility product of 1. The incoming water contains three components, A, C and D. The concentrations in the incoming water are 0.5 mol/l of A, 2 mol/l of C and 2 mol/l of D. No complexes are formed by the components. The Darcy velocity, or flux, is 1 $m^3/(m^2 \cdot year)$, and the column length is 1 m. In DYNAMIX and in PHASEQL/FLOW, the water within the column is initially pure water in equilibrium with the solid phases. The water concentration then becomes 1 mol/l of both A and B, with no C or D. In the CHEMFRONTS calculations, the water initially in the column is assumed to be the incoming water, but in equilibrium with the solid phases in the column. The water concentration is then 2 mol/l of both C and D, 1.281 mol/l of A and 0.781 mol/l of B.

Liu and Narasimhan (1989b) looked at three regions in their comparison:

Region 1	where mineral is dissolved
Region 2	where the incoming water is in equilibrium with the solid phase
Region 3	where the initial water is present but the inflowing water has not arrived.

As CHEMFRONTS does not have region 3, only regions 1 and 2 are compared. The concentration profiles for the components from the calculations with CHEMFRONTS are shown in figure 5.1.1. The mineral profiles are shown in figure 5.1.2. The curves are not sharp because the maximum dissolution rate has been chosen in this example for the solid phases in CHEMFRONTS. As components C and D are not involved in the mineral dissolution and no complexes are included in this example, the concentrations of C and D do not change at all. The problem is thus a two-component problem in which diffusion and dispersion are not considered.

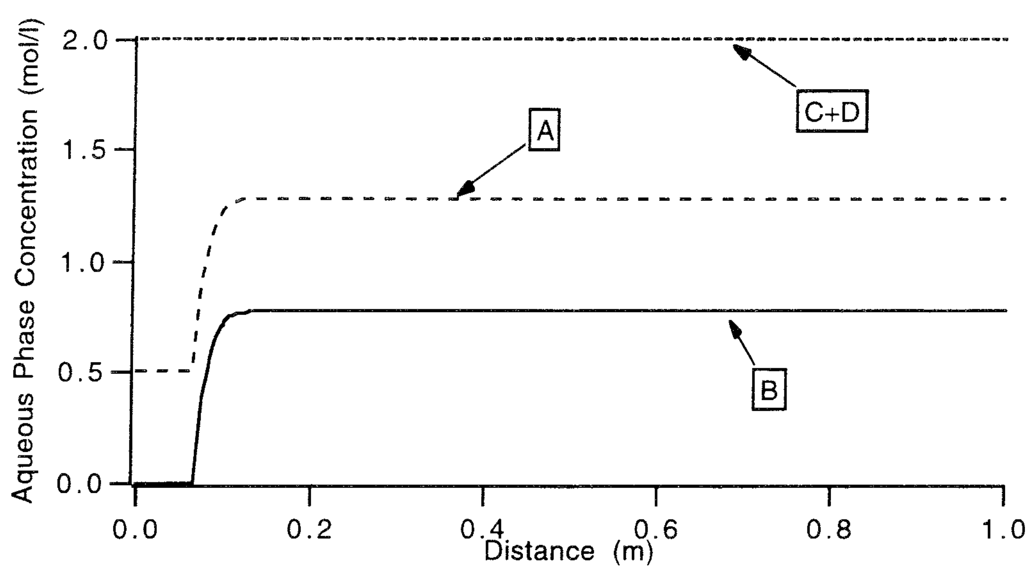


Figure 5.1.1 The concentration profiles for the components when half a pore volume of water has flowed through the column.

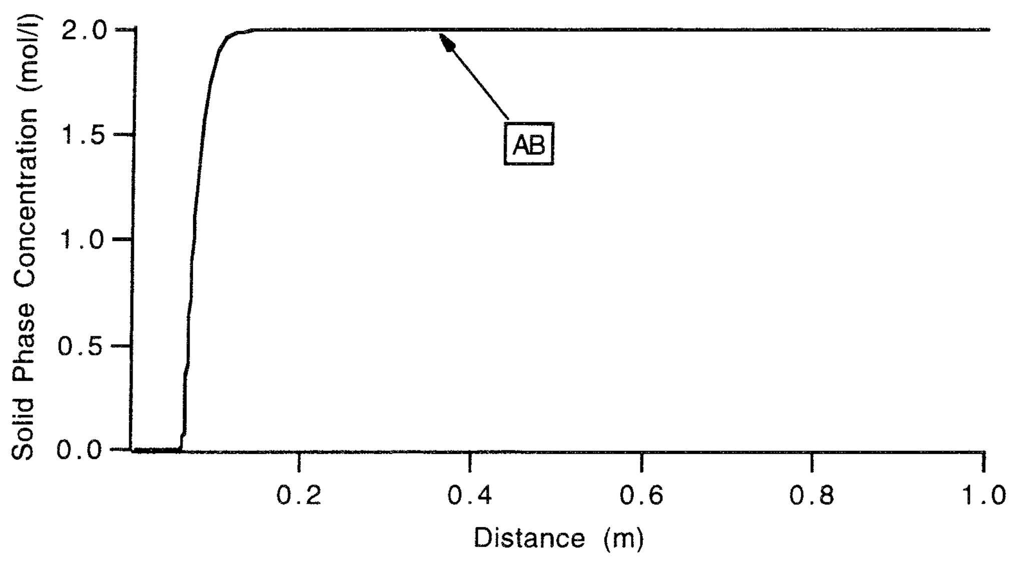


Figure 5.1.2 The mineral profiles when half a pore volume of water has flowed through the column.

The results from the three programs are shown in table 5.1.1. CHEMFRONTS gives the same results as PHASEQL/FLOW (Walsh et al. 1984) and approximately the same results as DYNAMIX (Liu and Narasimhan, 1989a). This indicates that the computation procedure is correct, at least for simple problems. As this is a very simple example, it is not surprising that the agreement is good.

Table 5.1.1. Four-component dissolution problem: comparison of CHEMFRONTS, DYNAMIX (Liu and Narasimhan, 1989a), and PHASEQL/FLOW (Walsh et al. 1984).

Region 1

Concentration	CHEMFRONTS	DYNAMIX	PHASEQL/FLOW
C_A	0.5	0.5	0.5
C_B	0.0	0.0	0.0
C_C	2.0	2.0	2.0
C_D	2.0	2.0	2.0
C_{AB}	0.0	0.0	0.0

Region 2

Concentration	CHEMFRONTS	DYNAMIX	PHASEQL/FLOW
C_A	1.281	1.283	1.281
C_B	0.781	0.780	0.781
C_C	2.0	2.0	2.0
C_D	2.0	2.0	2.0
C_{AB}	2.0	2.0	2.0

5.2 Oxidation of pyrite in the presence of K-feldspar

This example is based on an previously studied example of the evolution of a redox front in a uranium mine in Poços de Caldas (Cross et al., 1991). For a more extensive description, see section 5.4. In this simplified version, the problem has been used to develop the computer program CHEMFRONTS.

Initially, there is a homogeneous column with a porosity of 15% and a mineral content of 501 moles of pyrite and 7697 moles of K-feldspar per cubic metre of the column. The inlet water contains 8 mg of O_2 per litre and has a pH of 5.1. In addition, it has trace amounts of all other components used in this calculation. Their concentrations are so small that they do not noticeably influence the results. The reactions are the dissolution of pyrite and K-feldspar. The dissolved species form various complexes, and some new minerals precipitate. Here, kaolinite, quartz and ferric-oxy-hydroxide (FOH for short) are allowed to precipitate.

The chemical reactions can be summarized as follows: The pyrite reacts with the oxygen in the incoming water

$$FeS_2\ (s) + 3.75\ O_2\ (aq) + 0.5\ H_2O \rightarrow$$

$$H^+\ (aq) + Fe(III)\ (aq) + 2\ SO_4^{2-}\ (aq) \qquad (5.2.1)$$

The ferric species formed then precipitate as FOH, in this example $Fe(OH)_3$

$$Fe(III)\ (aq) + 3\ H_2O \rightarrow Fe(OH)_3\ (s) + 3\ H^+\ (aq) \qquad (5.2.2)$$

One oxidized pyrite molecule will produce a total of four protons. The K-feldspar ($KAlSi_3O_8$) will then react with the protons to form kaolinite ($Al_2Si_2O_5(OH)_4$) and quartz (SiO_2)

$$KAlSi_3O_8\ (s) + H^+\ (aq) + 0.5\ H_2O \rightarrow$$

$$K^+\ (aq) + 2\ Si(IV)\ (aq) + 0.5\ Al_2Si_2O_5(OH)_4\ (s) \qquad (5.2.3)$$

$$Si(IV)\ (aq) \rightarrow SiO_2\ (s) \qquad (5.2.4)$$

Overall, one dissolved pyrite precipitates one FOH, dissolves four feldspar and produces two kaolinite and four quartz. Because of the solubility of the minerals, some components stay in the water phase and flow out of the system.

The mineral profiles after 5404 years are shown in figure 5.2.1a-e. The black parts of the profiles are the irregularities in the mineral profiles caused by the time stepping procedure described in section 4.10. The first (fastest) front is at 0.071 metres. Downstream from the front is the initial rock with pyrite and K-feldspar. Upstream from the first front, the pyrite has reacted completely. A part of the K-feldspar has reacted with the protons produced by the dissolution of pyrite. Kaolinite, FOH and quartz have formed by equations (5.2.1)-(5.2.4). The second front, where the quartz dissolves, is $8.9 \cdot 10^{-3}$ metres from the inlet. The third front, where the K-feldspar dissolves, is $3.3 \cdot 10^{-3}$ metres from the inlet. Most of the aluminium from the feldspar is precipitated as kaolinite, equation (5.2.3). The fourth front shows where the FOH is completely dissolved, and at the fifth front the kaolinite dissolves as the last mineral.

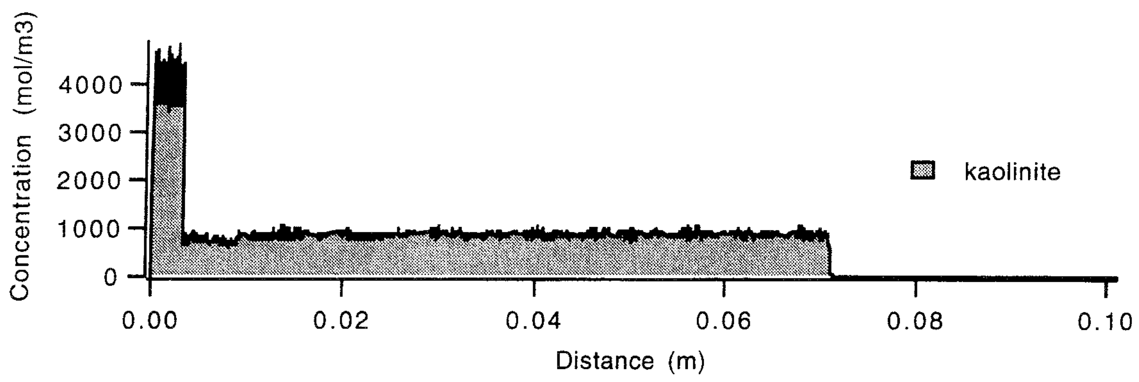


Figure 5.2.1a The kaolinite content in the column.

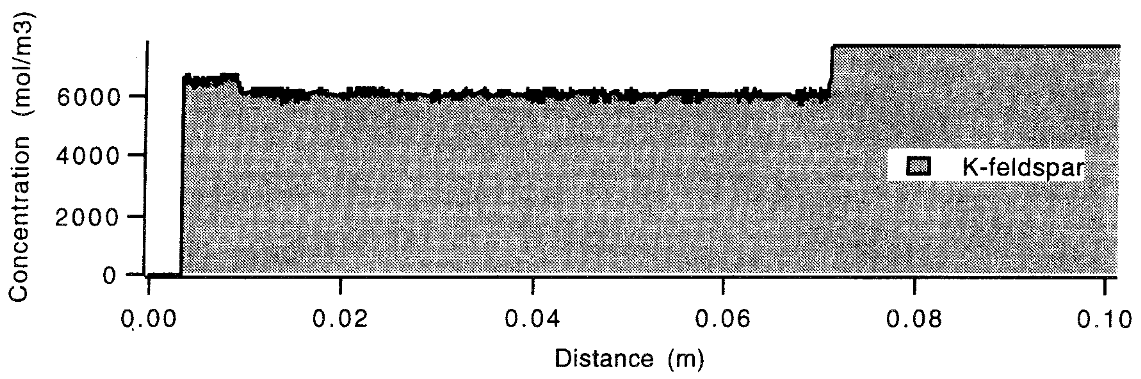


Figure 5.2.1b The K-feldspar content in the column.

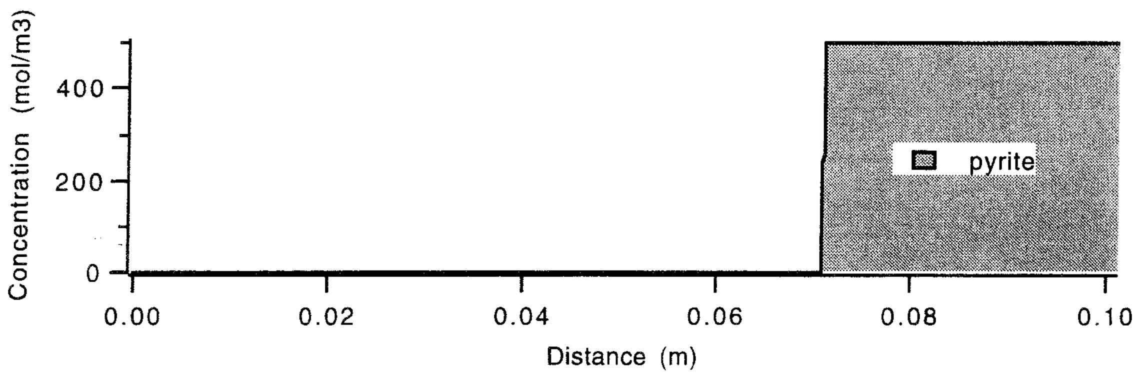


Figure 5.2.1c The pyrite content in the column.

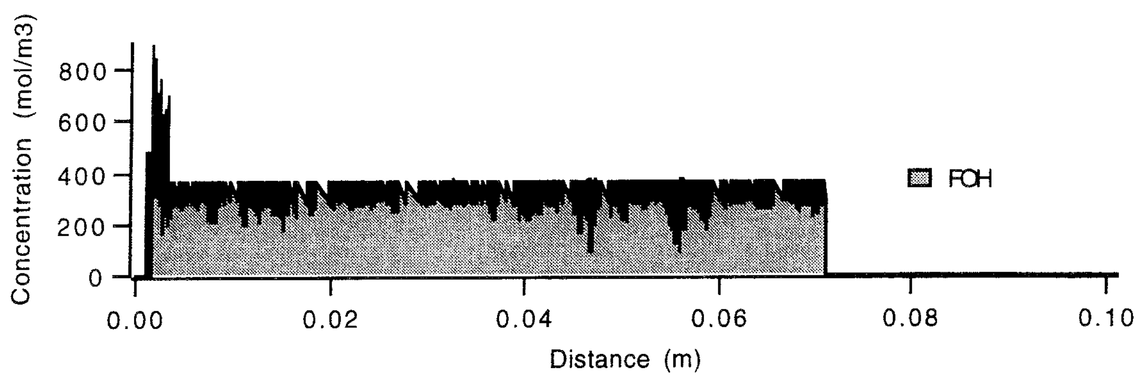


Figure 5.2.1d The FOH content in the column.

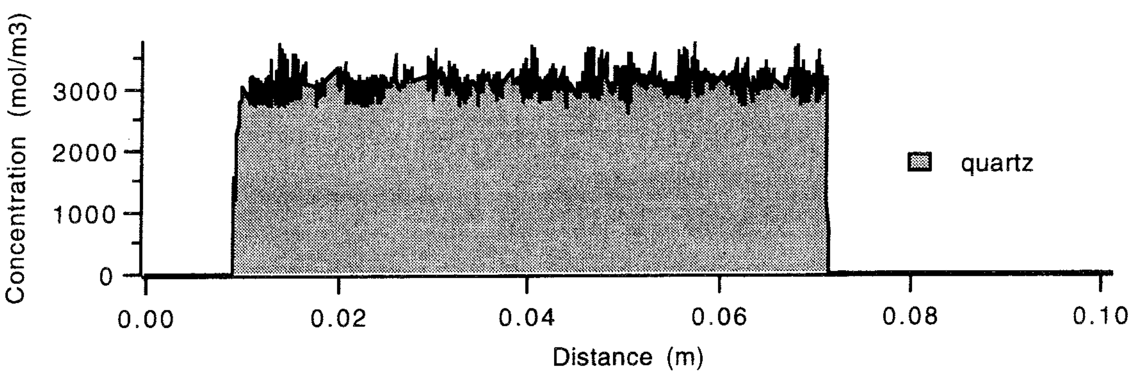


Figure 5.2.1e The quartz content in the column.

In figures 5.2.1a-e, the fronts are located at 0.071, 0.0089, 0.0033, 0.00096 and 0.000083 metres after 5404 years of water infiltrating.

The water concentration profiles for the free concentrations, i.e. no complexes are included, are shown in figure 5.2.2a-c. The incoming water is undersaturated in relation to all the minerals. As it passes the various mineral zones, it equilibrates with the minerals. The pyrite reacts with the oxygen, which changes the oxygen concentration from $2.5 \cdot 10^{-4}$ to $4.6 \cdot 10^{-68}$ mol/l within a very short distance. This forms a sharp redox front. Downstream from the redox front, at 0.071 metres, the environment is reduced. Upstream it is oxidized. The concentration changes in the water are related to the various mineral fronts.

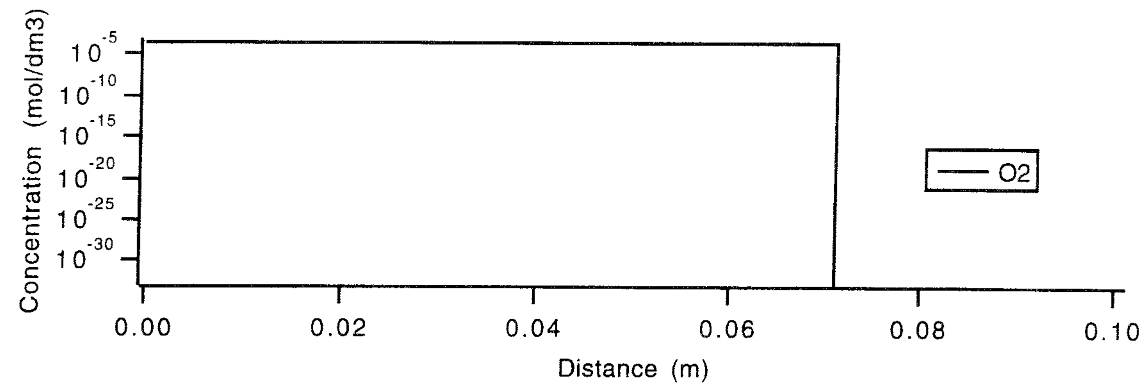


Figure 5.2.2a The oxygen concentration in the column.

Figure 5.2.2b shows the hydrogen ion concentration. At the redox front, the pH decreases to about 4 when the pyrite is dissolved, and increases to about 8 after the K-feldspar dissolution has consumed the protons.

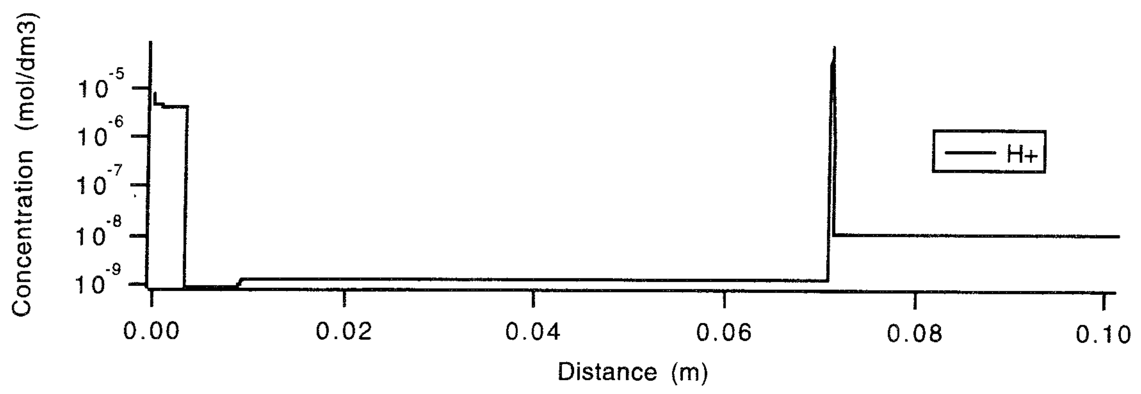


Figure 5.2.2b The hydrogen concentration in the column.

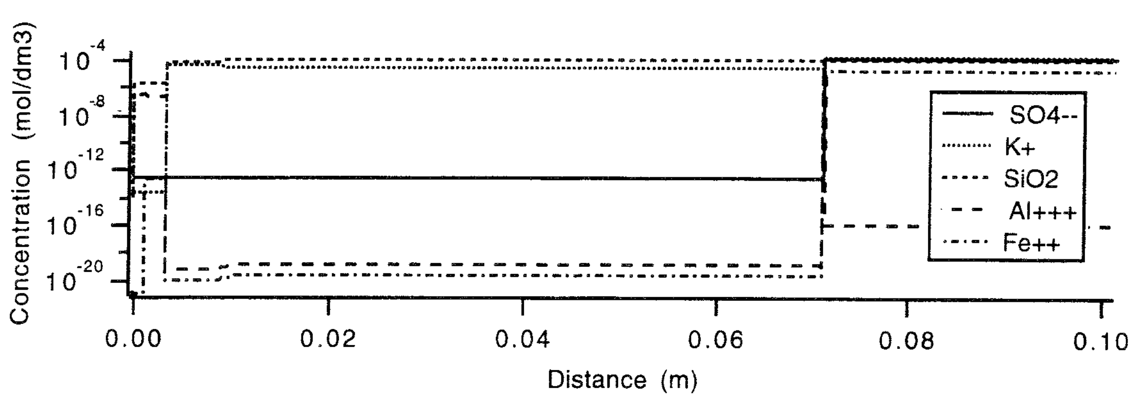


Figure 5.2.2c The concentration of silica, iron, aluminium, potassium and sulphate in the column.

In figures 5.2.2a-c, the free concentrations of O_2, H^+, SiO_2, Al^{3+}, Fe^{2+}, K^+ and SO_4^{2-} in the column are shown after 5404 years of water infiltrating. The redox front is at 0.071 metres. The concentration changes are located at the fronts.

From the concentration profiles, the program computes the dissolution and precipitation rates for the various minerals, dx/dt, see figures 5.2.3a-c. The term dx/dt is negative for dissolution and positive for precipitation. There is a maximum dissolution rate of 50 mol/(m3·year) (assumed value) in this example. The precipitation rate is unlimited.

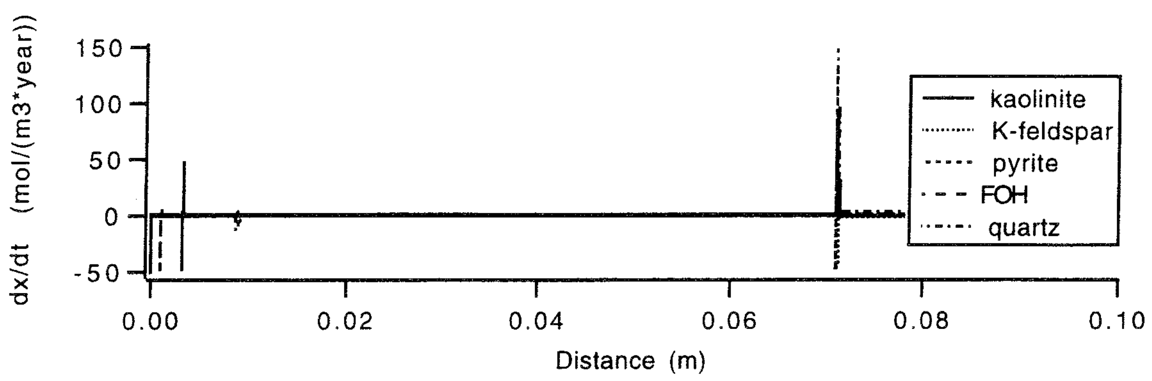


Figure 5.2.3a The dissolution/precipitation rate over all the fronts.

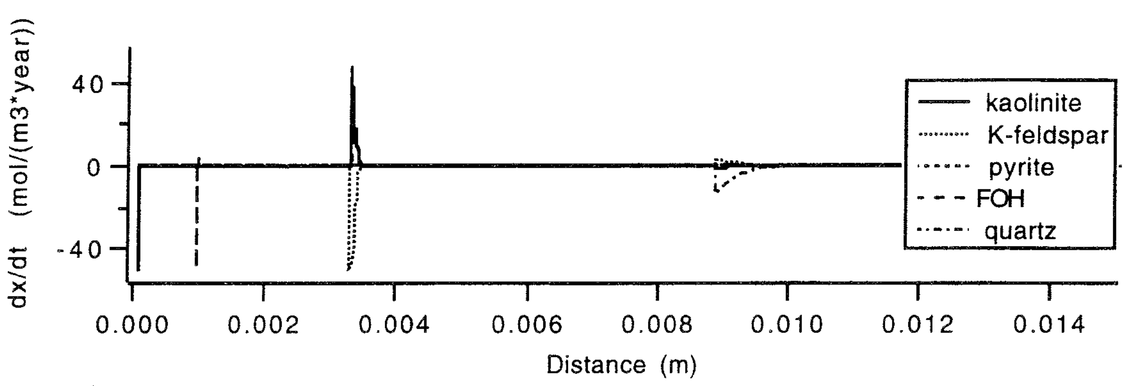


Figure 5.2.3b The dissolution/precipitation rate over the last four fronts.

Figure 5.2.3c shows that the pyrite dissolution involves equation (5.2.1)-(5.2.4). When the pyrite dissolves, FOH is precipitated. These reactions produce four protons for every pyrite. These protons are consumed by the K-feldspar dissolution. As the dissolution rate is limited to 50 mol/(m^3·year), the K-feldspar dissolution front has to be four times wider than the pyrite dissolution front.

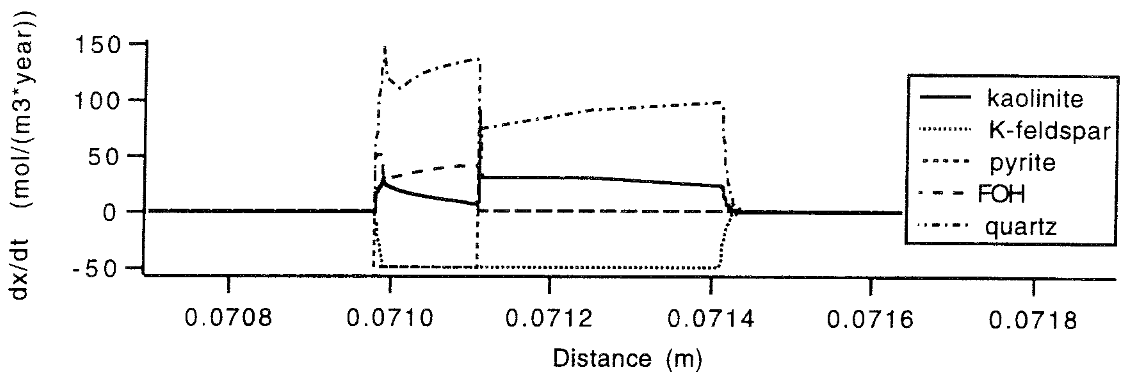


Figure 5.2.3c The dissolution/precipitation rate over the redox fronts.

Figures 5.2.3a-c shows the dissolution/precipitation rates for the minerals after 5404 years of water infiltrating. The irregularities in the profiles are caused by the complexity of the reactions. The dissolution and precipitation profiles are used to calculate the change in mineral concentration over a period of time. The period chosen depends on the situation, see section 4.5, but is about 0.5 years as an average in this example. The evolution of the fronts is shown in figure 5.2.4.

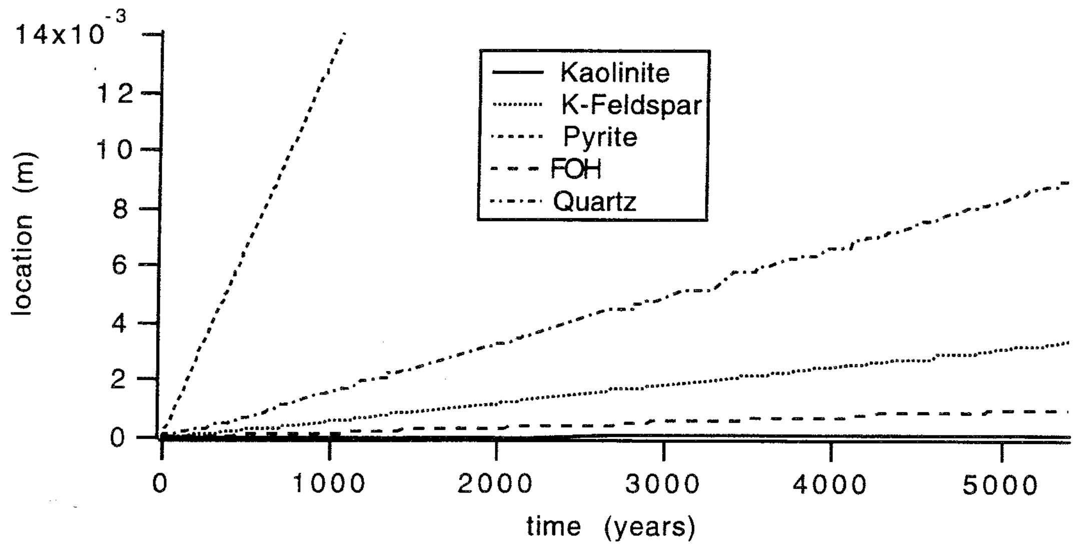


Figure 5.2.4 The evolution of the various fronts over time.

When the fronts are separated from each other they move with a constant velocity. It is thus possible to extrapolate to any time and obtain the front position. The chemical reactions take place at the fronts and, as no dispersion is included in the model, the reactions are independent of the distance between the fronts. It is therefore possible to extrapolate all data when the fronts are separated. After 1 million years, the mineral content would be that shown in figure 5.2.5.

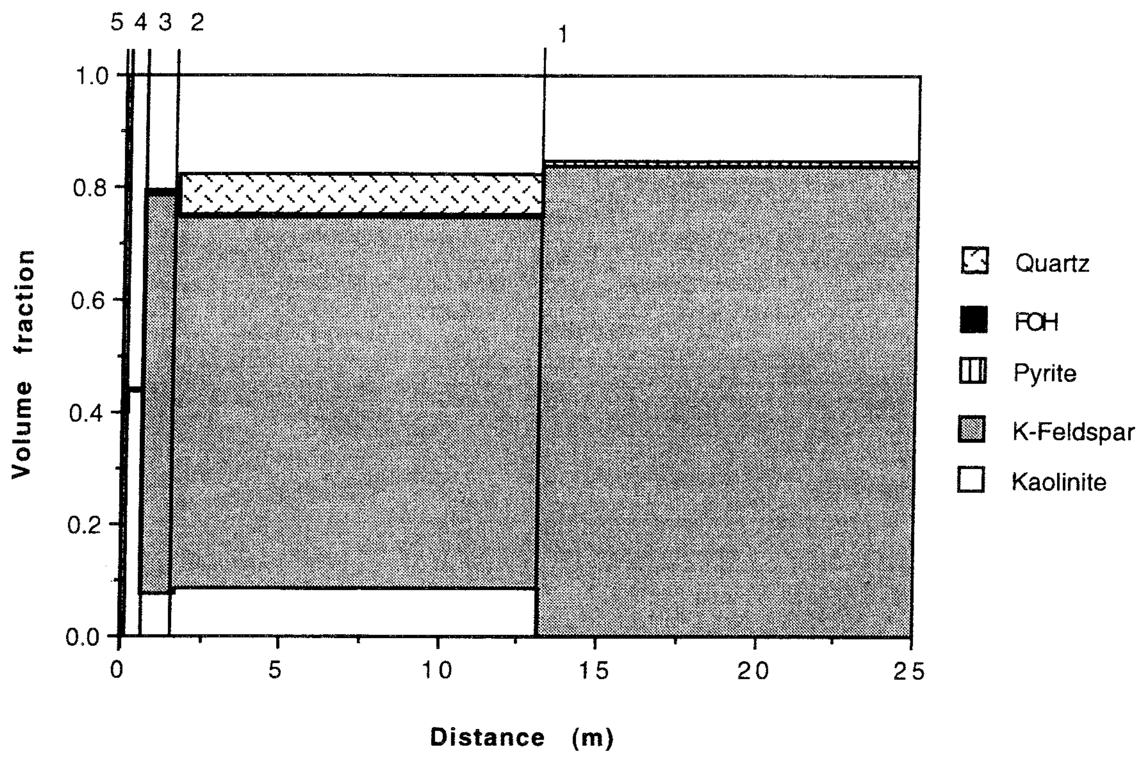


Figure 5.2.5 The mineral content in the column after 1 million years. The fronts are located at (1) 13.2 m, (2) 1.6 m, (3) 0.61 m, (4) 0.18 m, and (5) 0.015 m.

5.3 Oxidation of pyrite with gibbsite precipitation

This example is almost the same as the previous one, but gibbsite ($Al(OH)_3$) is also allowed to precipitate. Gibbsite is much less soluble than kaolinite, so the kaolinite dissolves

$$\text{Kaolinite (s)} \rightarrow 2 \text{ Gibbsite (s)} + 2 \text{ Quartz (aq)} \qquad (5.3.1)$$

The rate of dissolution depends on the solubility of quartz. The mineral profiles after 4771 years of infiltrating water are shown in figures 5.3.1a-f. A comparison between these figures and figures 5.2.1a-e shows that the presence of gibbsite influences both the kaolinite and the quartz content of the column. Pyrite, K-feldspar and kaolinite are not affected.

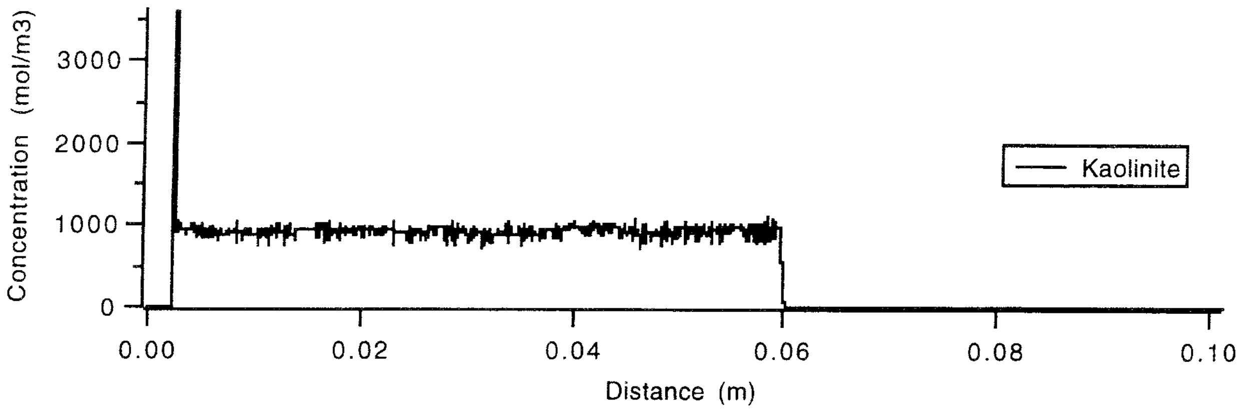


Figure 5.3.1a The kaolinite content in the column after 4771 years of water infiltrating.

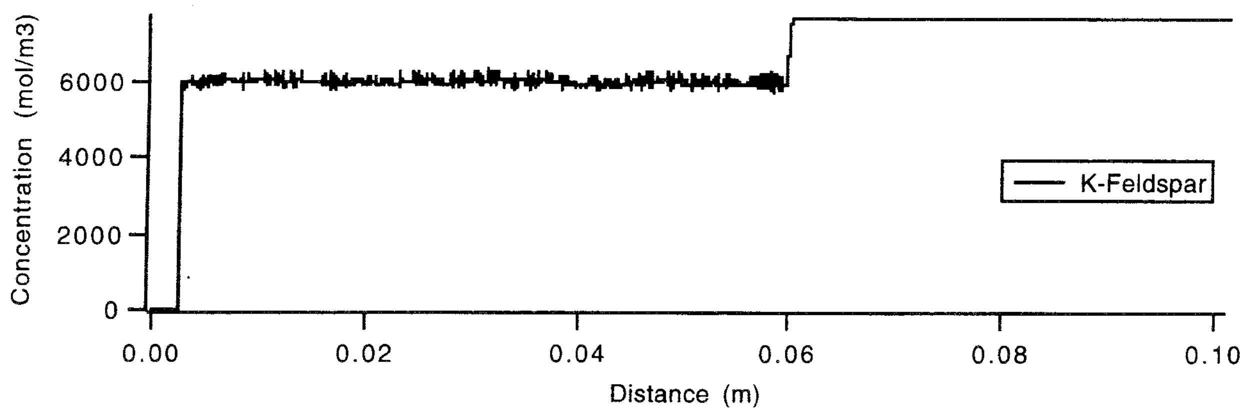


Figure 5.3.1b The K-feldspar content in the column after 4771 years of water infiltrating.

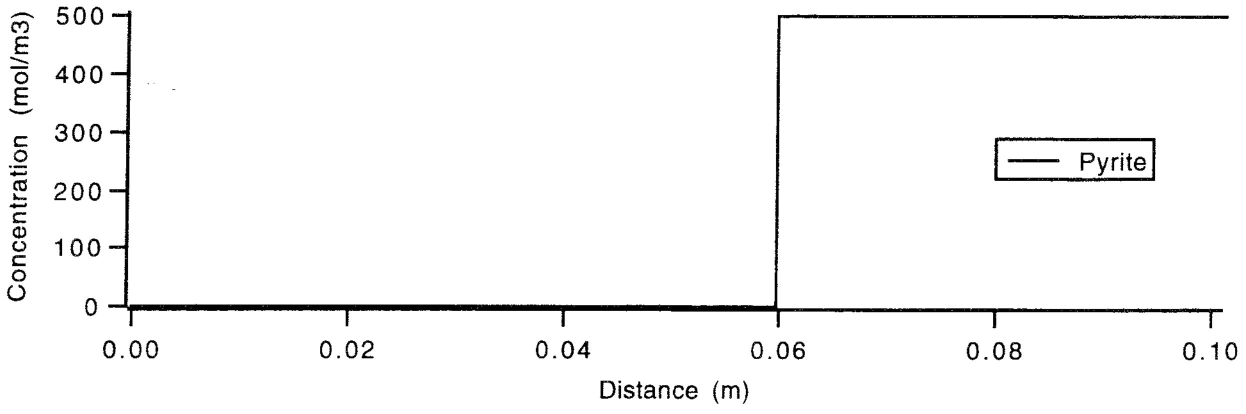


Figure 5.3.1c The pyrite content in the column after 4771 years of water infiltrating.

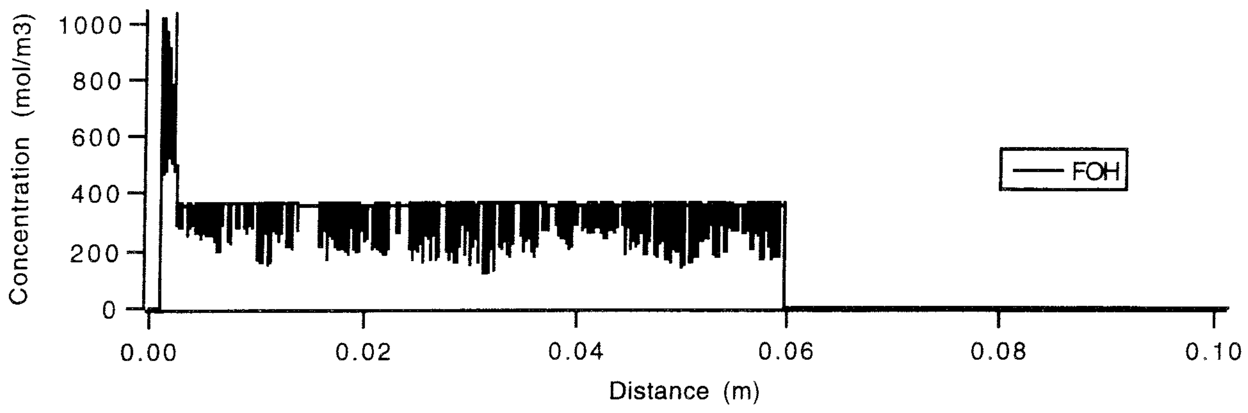


Figure 5.3.1d The FOH content in the column after 4771 years of water infiltrating.

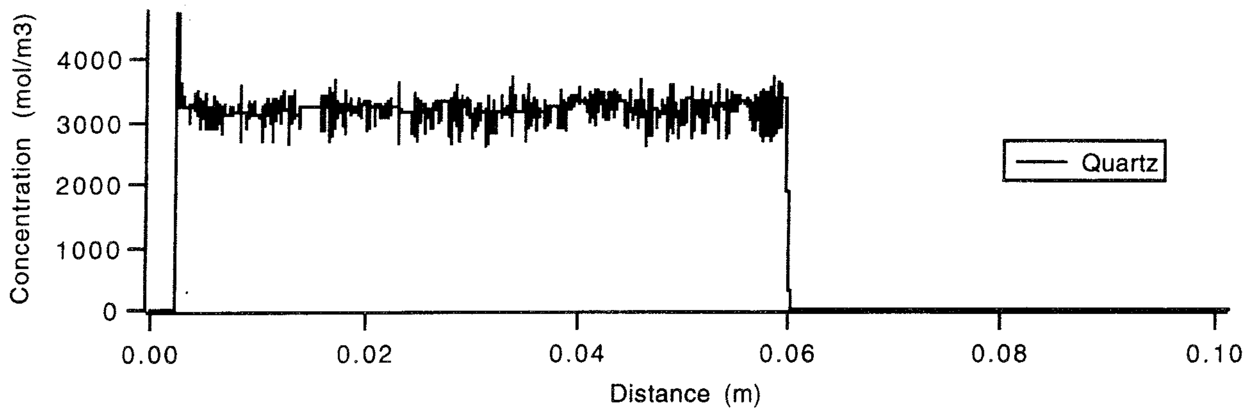


Figure 5.3.1e The quartz content in the column after 4771 years of water infiltrating.

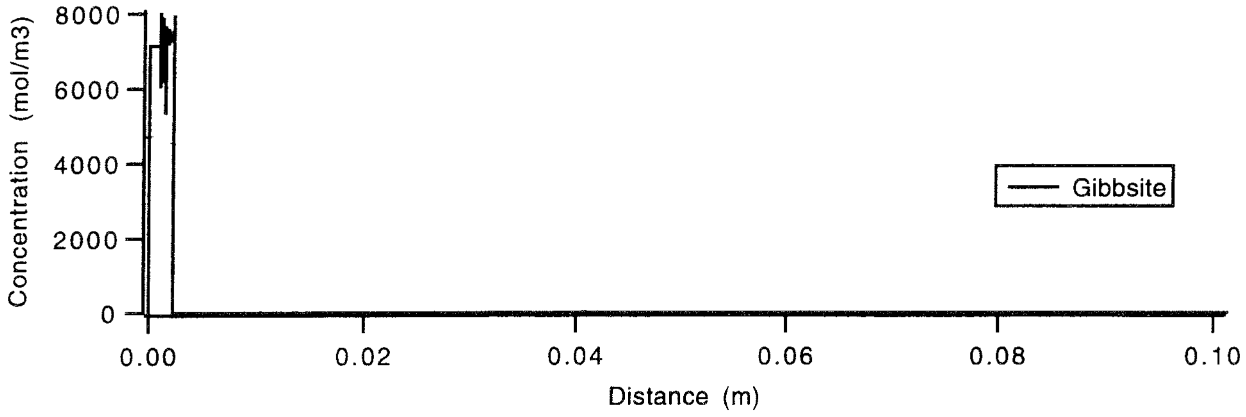


Figure 5.3.1f The gibbsite content in the column after 4771 years of water infiltrating.

In figures 5.3.1a-f, the fronts are located at 0.060, 0.0089, 0.0024, 0.0023, 0.0023, 0.0011 metres and 0.3 micrometres after 4771 years of water infiltrating.

The water concentration profiles for the free concentrations, i.e. with no complexes included, are shown in figures 5.3.2a-c.

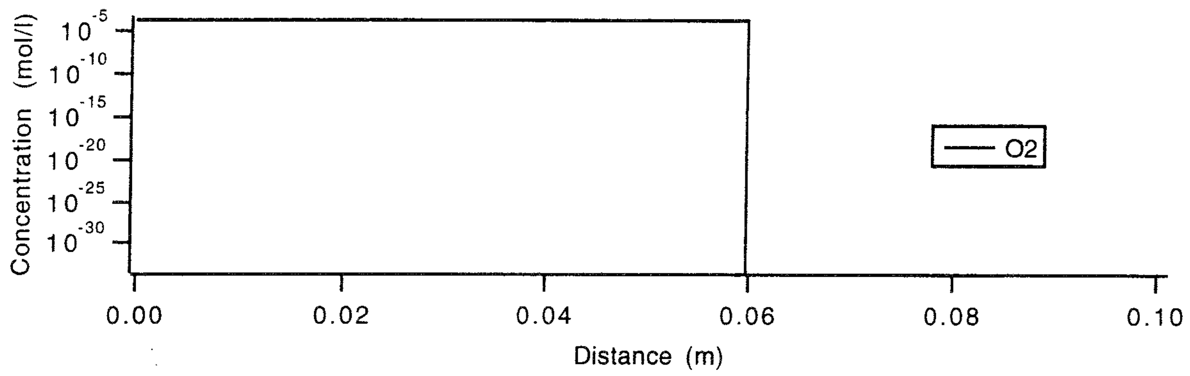


Figure 5.3.2a The oxygen concentration in the column.

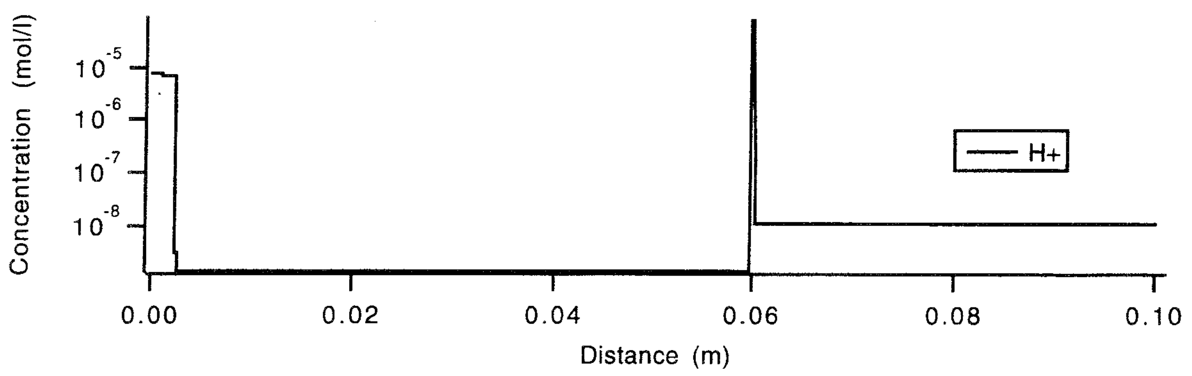


Figure 5.3.2b The hydrogen concentration in the column.

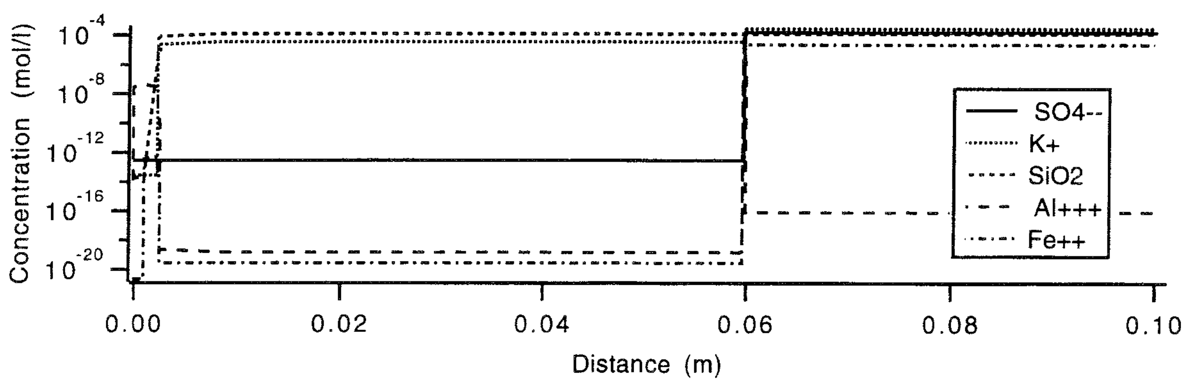


Figure 5.3.2c The concentration of silica, iron, aluminium, potassium and sulphate in the column.

Figures 5.3.2a-c show the free concentration of O_2, H^+, SiO_2, Al^{3+}, Fe^{2+}, K^+ and SO_4^{2-} in the column after 4771 years of water infiltrating. The redox front is at 0.060 metres.

The dissolution and precipitation rates, dx/dt, for the various minerals are shown in figures 5.3.3a-c. A comparison of the profiles for pyrite oxidation with K-feldspar and with precipitation of gibbsite shows that gibbsite does not noticeably affect the redox front. The dissolution fronts of quartz, K-feldspar and kaolinite are coupled and leave FOH and gibbsite behind.

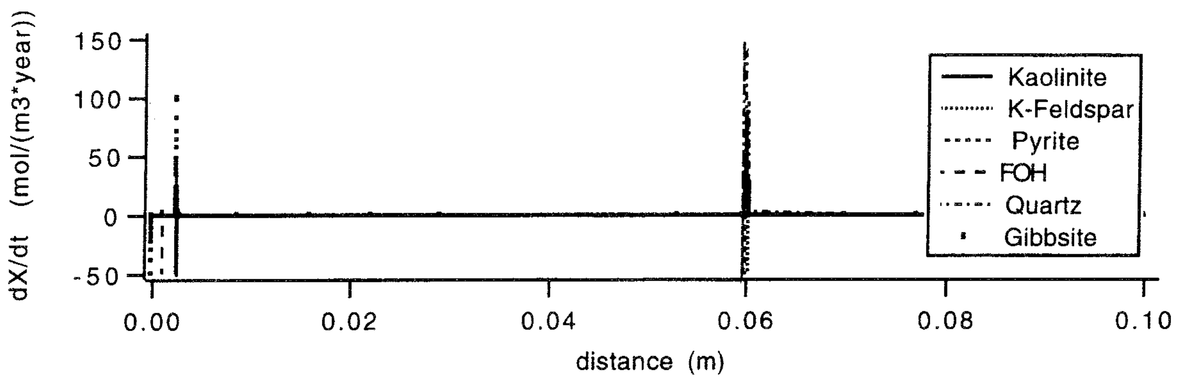


Figure 5.3.3a The dissolution/precipitation rate over all the fronts.

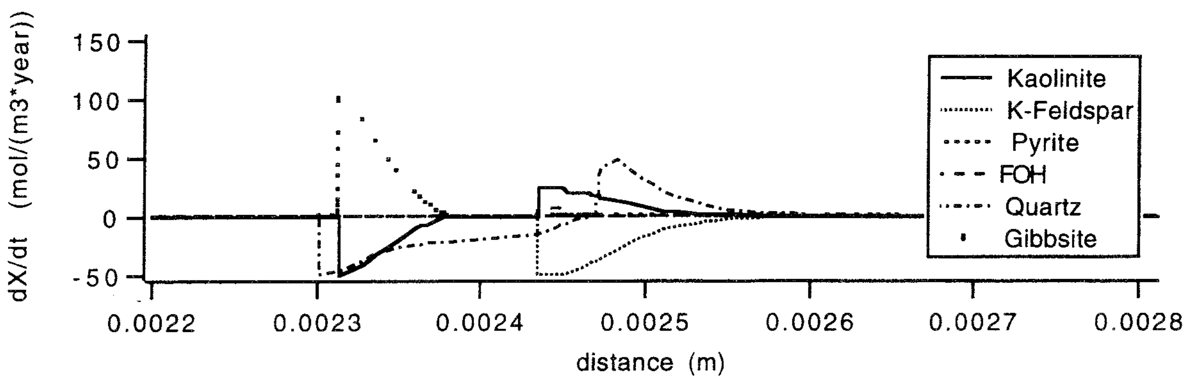


Figure 5.3.3b The dissolution/precipitation rate over fronts 2-4 where K-feldspar, kaolinite and quartz dissolve.

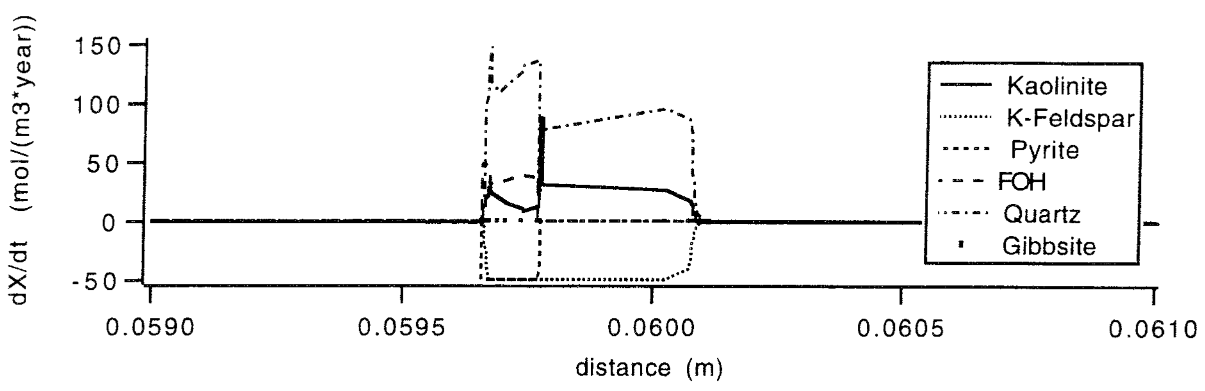


Figure 5.3.3c The dissolution/precipitation rate over the redox fronts.

Figures 5.3.3a-c shows the dissolution/precipitation rates for the minerals after 4771 years of infiltrating water. Figure 5.3.3a shows the overall dissolution/precipitation fronts. The redox front is clearly separated from the others. The figure also shows that the front width is small compared to the distance between the fronts. Figure 5.3.3b shows an enlargement of the kaolinite, K-feldspar and quartz fronts. They are not separated from each other, because the fronts are coupled and will thus not separate. Figure 5.3.3c shows an enlargement of the redox front. At the redox front, pyrite is dissolved by the incoming oxygen. Several other chemical reactions take place at the redox front owing to the pyrite dissolution, see equation (5.2.1)-(5.2.4).

The evolution of the fronts is shown in figure 5.3.4. The coupling between the fronts for K-feldspar, quartz and kaolinite dissolution is clearer in this figure than in figure 5.3.3b.

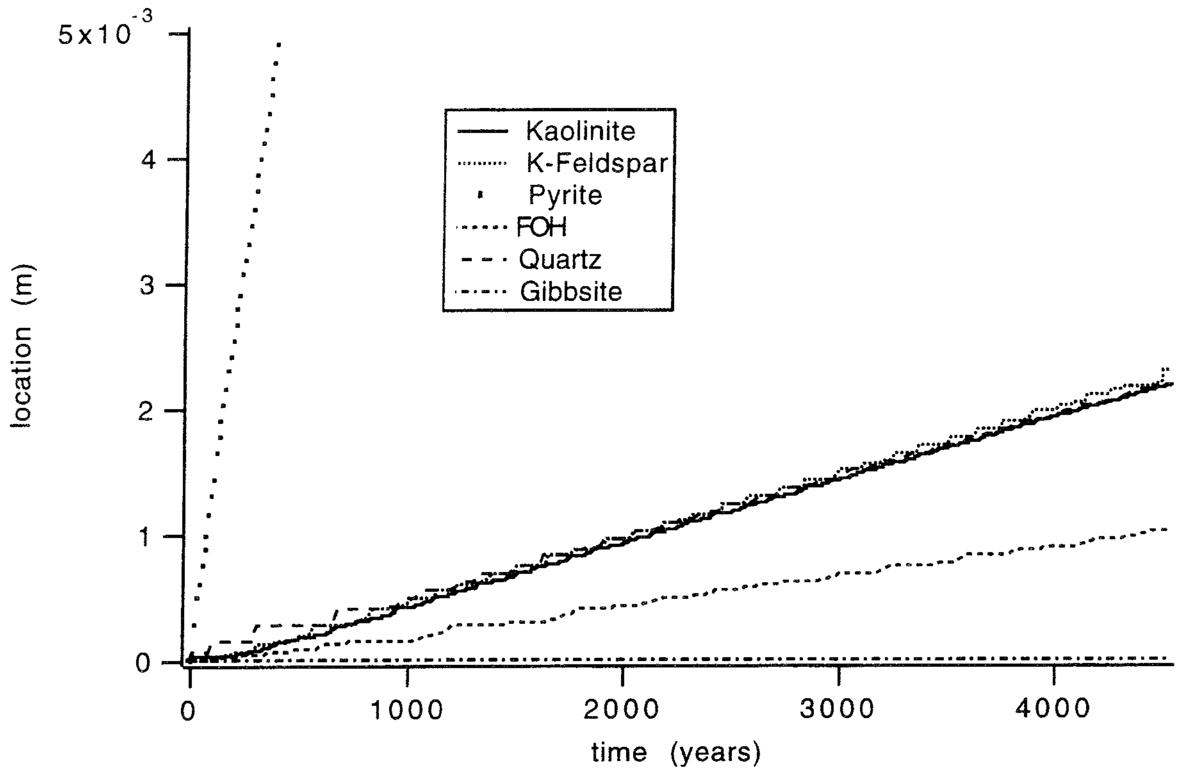


Figure 5.3.4 The evolution of the various fronts.

The location of the fronts after 1 million years can be extrapolated from the curves in figure 5.3.4. The mineral contents and the front positions are shown in figure 5.3.5. The difference between this example and the one in which gibbsite was held in suspension lies in the concentration of quartz and kaolinite resulting from equation (5.3.1).

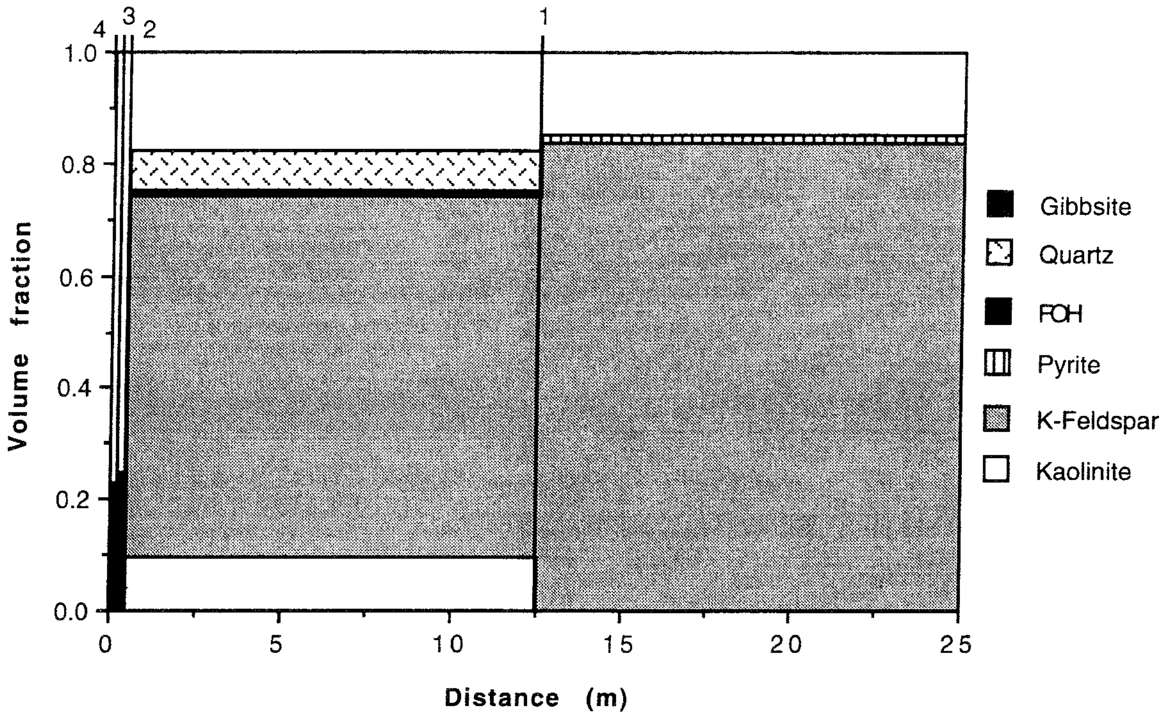


Figure 5.3.5 The mineral content in the column after 1 million years. The fronts are located at (1) 12.5 m, (2) 0.48 m, (3) 0.22 m and (4) 0.0 m.

5.4 Studies of the redox front in a uranium mine at Poços de Caldas

The Poços de Caldas project (Cross et al., 1991) was an international study of analogue processes. This study concerns redox fronts and uranium movement at the Osamu Utsumi mine. The open pit is 100 metres deep at its maximum, and is about 1 km long and 0.5 kilometres wide. The deeper portion of the rock is strongly reducing, whereas the upper part has become oxidizing owing to infiltration of rainwater. There is a sharp redox front separating the two regions.

The upper part of the rock (0-40 metres) is heavily weathered. The oxidized rock at depths of 40 to 194 metres is separated from the deeper-lying reduced rock (below 194 metres) by the redox front. The mineral compositions of the three zones are shown in figure 5.4.1.

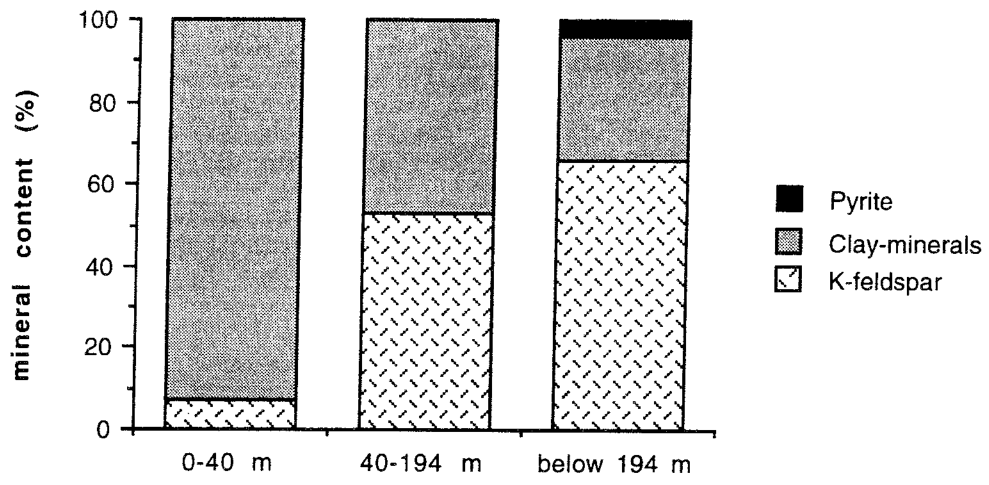


Figure 5.4.1 The mineral composition of the different layers at the open pit uranium mine in Poços de Caldas, Brasil.

Uraninite nodules 0.5-1 centimetres in diameter are found in many places just below the redox front in the reduced rock. Erosion of the rock in this region over the last 90 million years has been estimated from observations of the mineralogy to be between 3 and 9 kilometres. These figures are very approximate, but indicate that erosion must play a major role in the evolution of the site.

Cross et al. (1991) have modelled the movement of the redox front in the mine with the computer program CHEQMATE (Harworth et al., 1988) described in section 2.2.2.3 of this report. Similar input data have been used to validate CHEMFRONTS. As the input data are given in different units for the two programs, the input data are not exactly the same. The input data used by Cross et al. are not fully reported. This makes it impossible to compute exactly the same example as they used. The database used by Cross et al., HATCHES (Cross and Ewart, 1991), were not available to us. For the calculations with CHEMFRONTS the SKB database (Puigdomènech and Brunó, 1991) was used.

In the example used in both calculations, the flow is assumed to have a constant velocity along the flow path. The oxygen concentration in the water is assumed to be in equilibrium with air, and the total carbonate content is about one order of magnitude higher than if it were in equilibrium with air. Such increased levels are assumed to result from the degradation of organic material in the soil covering the rock. The infiltration rate of the water is taken to be 0.1 $m^3/(m^2 \cdot year)$, or about 5% of the rainfall in the area.

The input data for the two calculations are shown in table 5.4.1. To be able to compare the results from CHEMFRONTS with those from CHEQMATE, all data are extrapolated to 38000 years of infiltration of rainwater, as in the report from Harworth et al (1988).

Table 5.4.1. The input data for the two calculations compared in this study.

	Cross et al.		This study	
Program	CHEQMATE		CHEMFRONTS	
Database	HATCHES (Cross and Ewart, 1991)		SKB database (Puigdomènech and Bruno, 1991)	
Mineral content:				
Pyrite	2.3	mol/l pore water	345	mol/m^3*
K-Feldspar	35.2	mol/l pore water	5280	mol/m^3*
Kaolinite	7.4	mol/l pore water	2682	mol/m^3*
Uraninite	1.8·10-3	mol/l pore water	0.2701	mol/m^3*
Porosity	15	%	15	%
Water properties:				
Water flux	0.1	m^3/m^2·year	0.1	m^3/m^2·year
pH	5.1		5.1	
Total carbonate	0.16	mmol/l	0.16	mmol/l
Dissolved oxygen	0.31	mmol/l	0.31	mmol/l

* The input data for both programs are essentially the same, although expressed in different form. See discussion below.

Figures 5.4.2a-f shows the mineral concentration according to the calculations made with CHEMFRONTS. The kaolinite concentration in figure 5.4.2a changes in two steps. Some kaolinite is produced when K-feldspar dissolves at the redox front, because of protons produced by the pyrite dissolution. More kaolinite precipitates at the K-feldspar dissolution front at 0.15 metres.

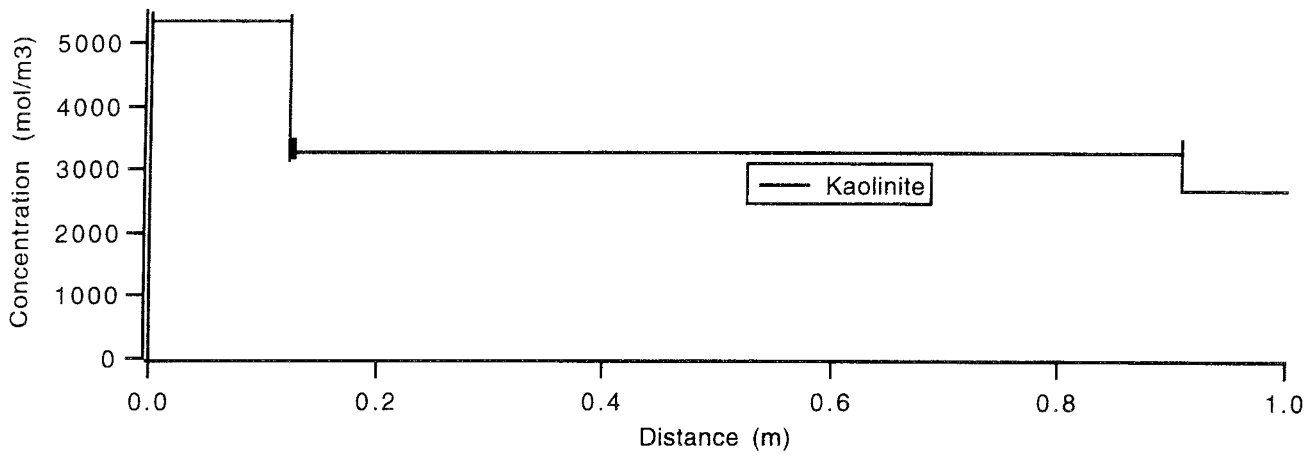


Figure 5.4.2a The kaolinite content in the column after 38000 years of infiltrating rainwater.

The K-feldspar concentration in figure 5.4.2b changes in two steps. At the redox front, pyrite is dissolved and protons are released. These are consumed by the K-feldspar dissolution. K-feldspar also dissolves at 0.15 metres because of protons in the incoming water.

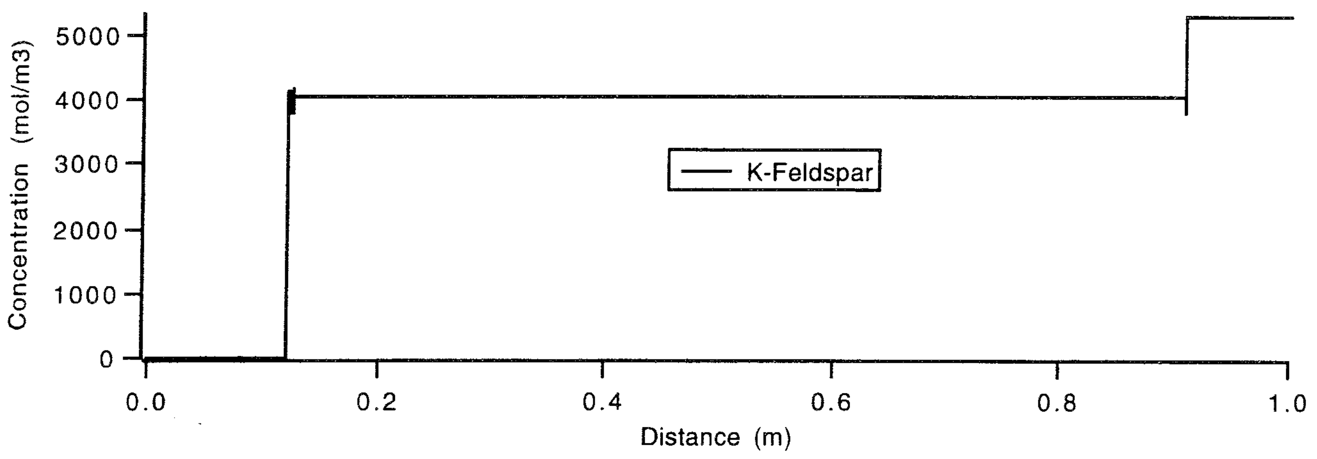


Figure 5.4.2b The K-feldspar content in the column after 38000 years of infiltrating rainwater.

Figure 5.4.2c shows that pyrite only exists downstream from the redox front. The dissolved iron is precipitated mainly as hematite, figure 5.4.2d.

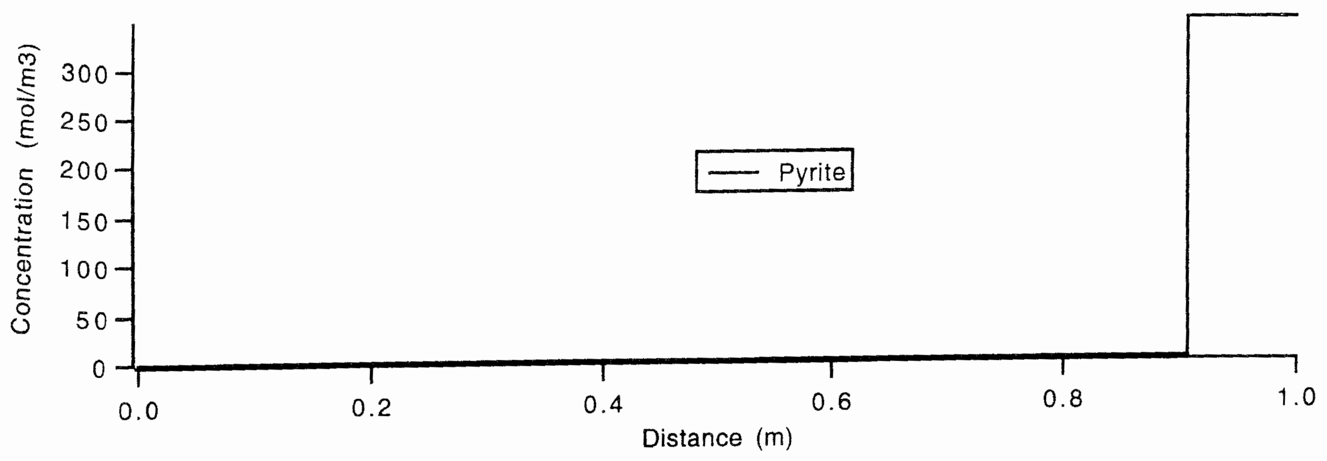


Figure 5.4.2c The pyrite content in the column after 38000 years of infiltrating rainwater.

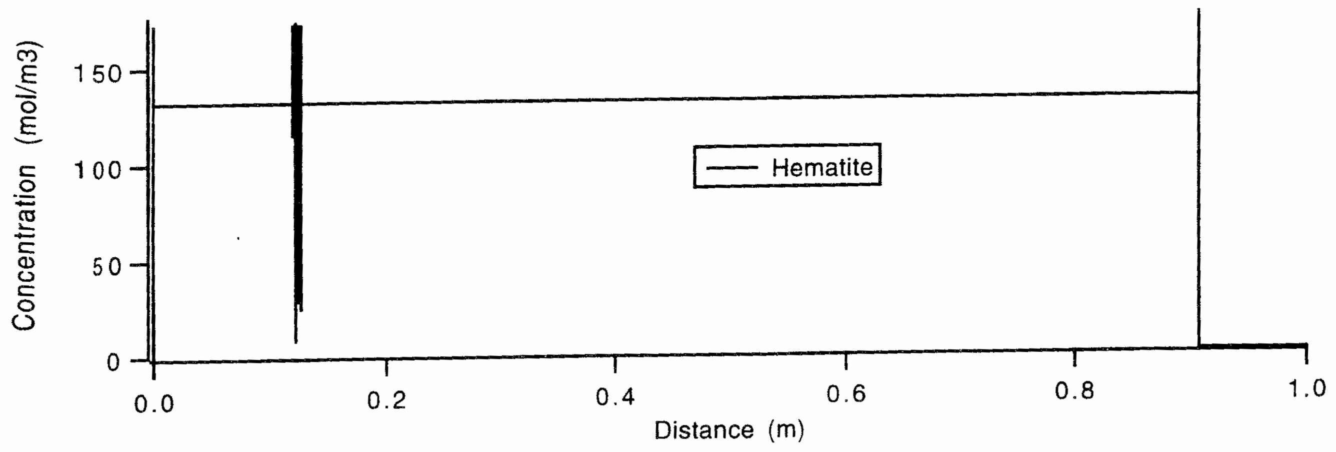


Figure 5.4.2d The hematite content in the column after 38000 years of infiltrating rainwater.

The chalcedony concentration, figure 5.4.2e, changes in two steps, like the K-feldspar and the kaolinite. When feldspar dissolves, chalcedony is precipitated. Every K-feldspar produces two chalcedony.

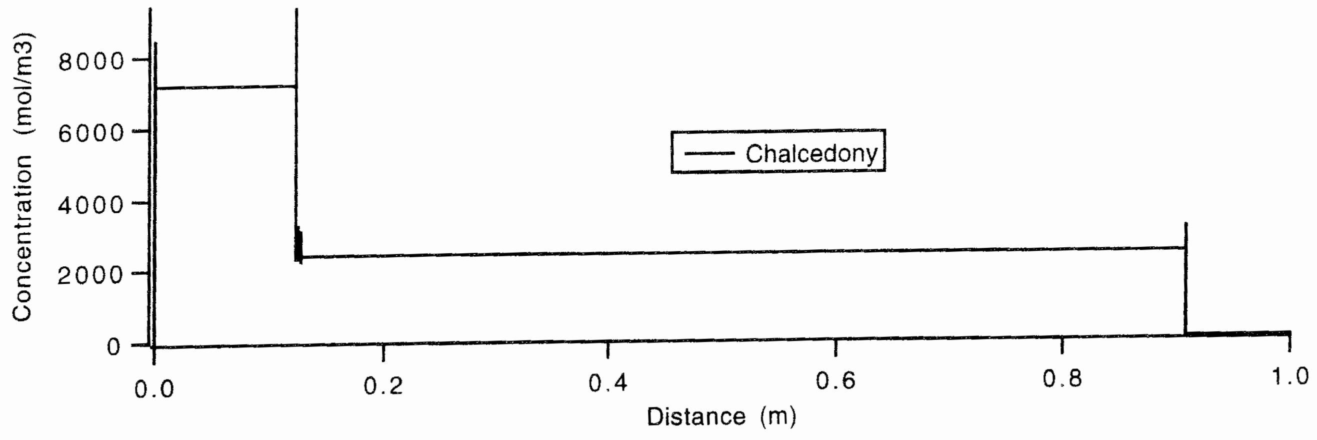


Figure 5.4.2e The chalcedony content in the column after 38000 years of rainwater infiltrating.

The uraninite concentration is shown in figure 5.4.2f. Most of the uraninite is located at the redox front; there is little uraninite downstream from the redox front. The uranium is oxidized by the oxygen in the water solution and dissolved. The ferrous iron in pyrite then reduces the uranium and it precipitates as uraninite again.

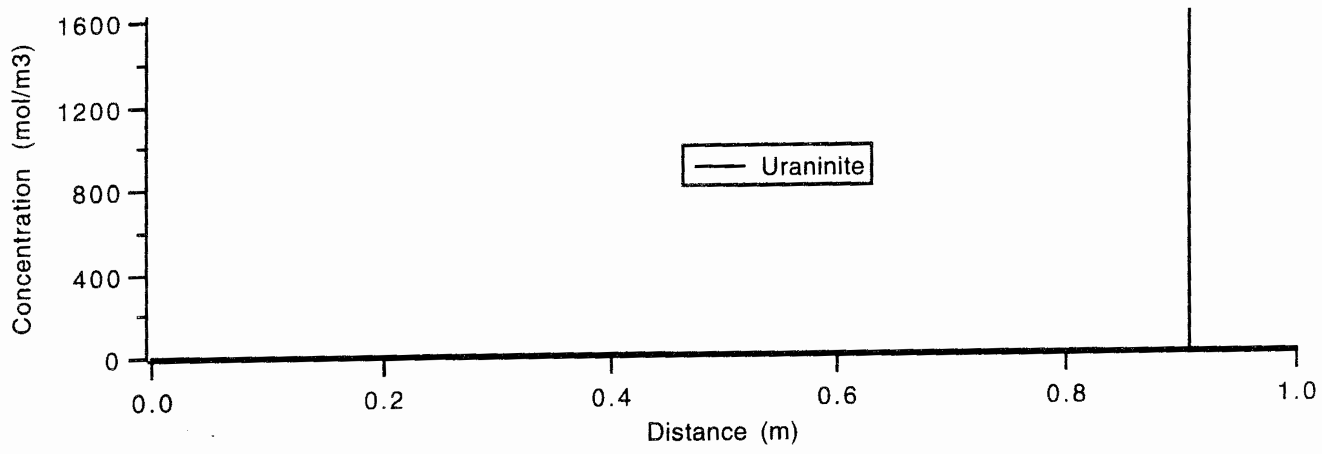


Figure 5.4.2f The uraninite content in the column after 38000 years of rainwater infiltrating.

Figure 5.4.3 shows the free oxygen and hydrogen concentrations in the column after 38000 years of rainwater infiltrating. When the water first comes into contact with K-feldspar, the pH increases to 7.6 owing to the reaction

$$KAlSi_3O_8\ (s) + H^+\ (aq) + 0.5\ H_2O \rightarrow$$

$$K^+\ (aq) + 2\ SiO_2\ (aq) + 0.5\ Al_2Si_2O_5(OH)_4\ (s) \qquad (5.4.1)$$

The pyrite dissolves as follows

$$FeS_2\ (s) + 3.75\ O_2\ (aq) + 3.5\ H_2O \rightarrow$$

$$4\ H^+\ (aq) + 2\ SO_4^{2-}\ (aq) + Fe(OH)_3\ (s) \qquad (5.4.2)$$

This releases more protons, so the pH decreases to 4 at the redox front. As K-feldspar is soluble in acidic water, more feldspar dissolves and the pH increases again to 7.2. Cross et al. (1991) report a pH of 8.6 downstream from the redox front.

The dissolution of pyrite in reaction (5.4.2) causes a sharp drop in the oxygen concentration, so there will be a front with a reducing environment on one side and oxidizing on the other. The pe of the water is -2.41 in the reducing region and 12.97 in the oxidizing region. Cross et al. (1991) report -4.68 for the reducing region and 12.1 for the oxidizing region. The redox front is situated at 0.91 m in the calculation made by CHEMFRONTS. In the calculations made by Cross et al. (1991) the redox front is at 0.75 m after 38000 years. The difference can be caused by differences in the formulation of the input data, and by two databases being used.

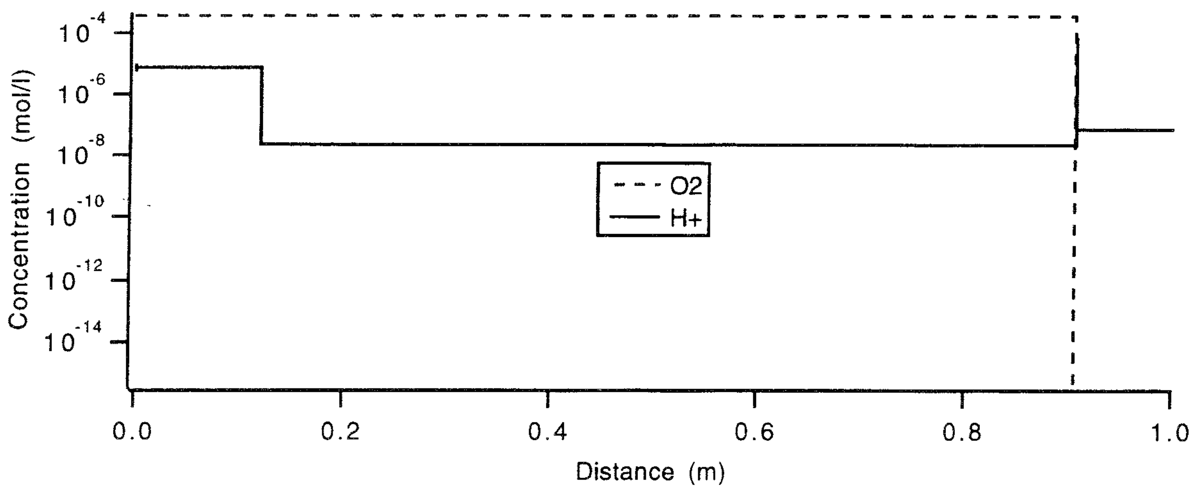


Figure 5.4.3 The oxygen concentration in the column after 38000 years of rainwater infiltrating.

Figures 5.4.4a and b show how the concentration of the aqueous species changes at the various fronts. The concentration of inorganic carbon is constant throughout the column. The silica concentration increases at the inlet where chalcedony dissolves, and at the K-feldspar dissolution front where the pH changes. The potassium concentration increases when K-feldspar dissolves. The sulphur concentration increases at the redox front where pyrite dissolves.

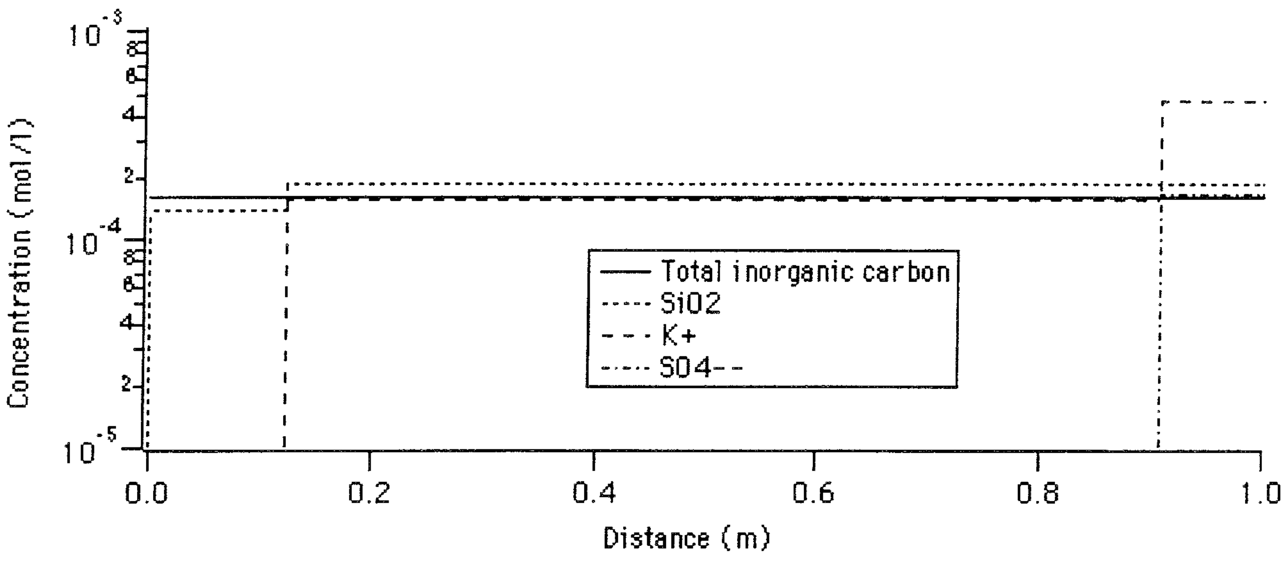


Figure 5.4.4a Total concentration of aqueous species after 38000 years of infiltration of rainwater.

The uranium concentration is high at the redox front and decreases in the reduced region, because the reduced form of uranium, U(IV), is less soluble than the oxidized form, U(VI), in this case. The iron concentration decreases at the K-feldspar dissolution front where the pH changes, and increases at the redox front. The reduced form of iron, Fe(II), is more soluble in this case than the oxidized form, Fe(III). The kaolinite dissolution at the inlet gives a fairly high aluminium concentration, as the pH is rather low. The pH increases at the K-feldspar dissolution front, so the aluminium concentration should decrease, but since more aluminium is produced when the K-feldspar dissolves, the concentration is almost unchanged. The aluminium concentration decreases in the reduced region.

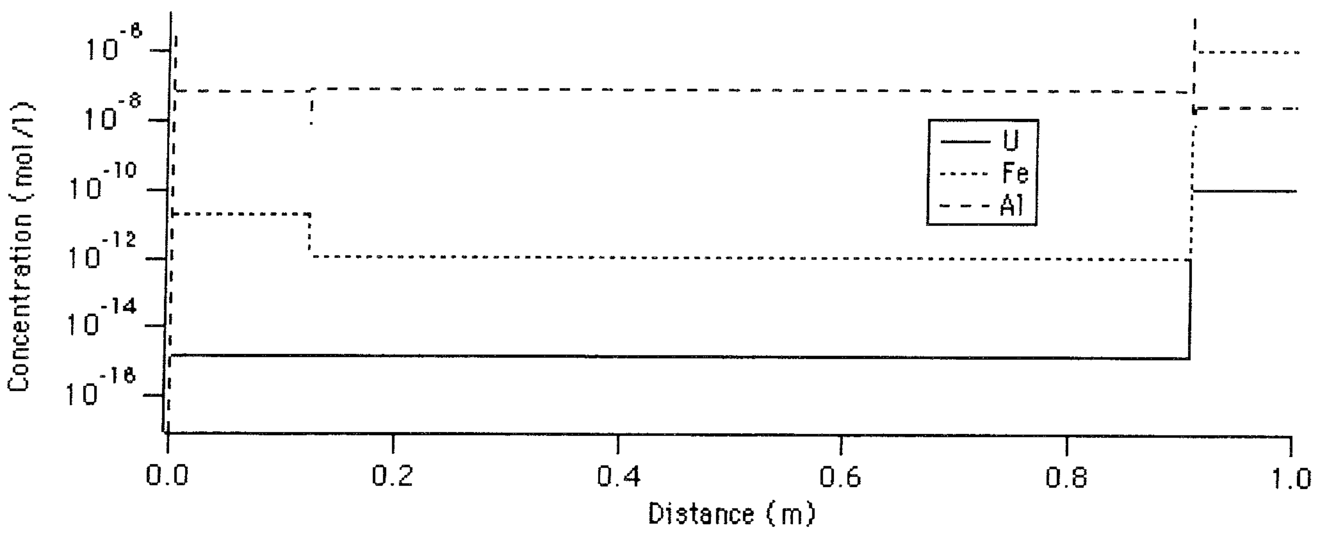


Figure 5.4.4b Total concentration of aqueous species after 38000 years of infiltration of rainwater.

Table 5.4.2 shows a comparison of the total concentrations of aqueous species downstream from the redox front. The agreement is acceptable, considering that different input data were used.

Table 5.4.2. A comparison of the total concentration of the aqueous species downstream from the redox front.

Species	Cross et al. mol/l	This study mol/l
K	$6 \cdot 10^{-4}$	$4.9 \cdot 10^{-4}$
SiO_2	$3 \cdot 10^{-4}$	$1.9 \cdot 10^{-4}$
Total carbon	$2 \cdot 10^{-4}$	$1.6 \cdot 10^{-4}$
SO_4^{2-}	$2 \cdot 10^{-4}$	$1.7 \cdot 10^{-4}$
Al	10^{-8}	$3.0 \cdot 10^{-8}$
Fe	no value published	$1.2 \cdot 10^{-6}$
U	10^{-10}	$1.0 \cdot 10^{-10}$

Figure 5.4.5 shows the uranium distribution through the column and figure 5.4.6 shows the uranium distribution at the redox front. The latter are from the calculations made by Cross et al. (1991). Since CHEQMATE includes diffusion and CHEMFRONTS does not, the best place for comparison is on either side of the redox front. The agreement is good, the differences being due to the use of different

databases.

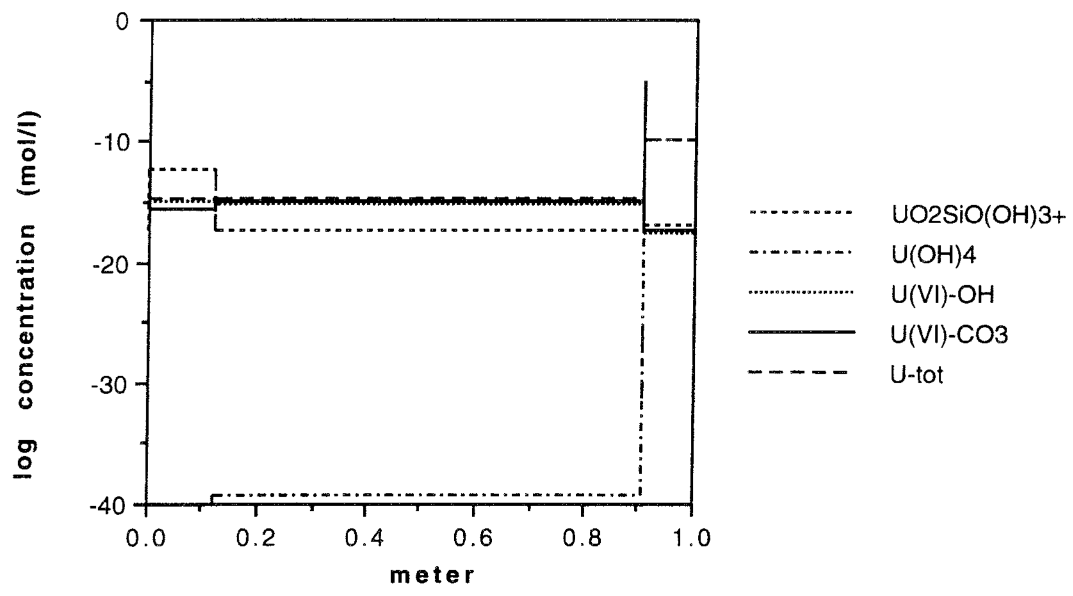


Figure 5.4.5 The uranium distribution after 38000 years of infiltrating rainwater.

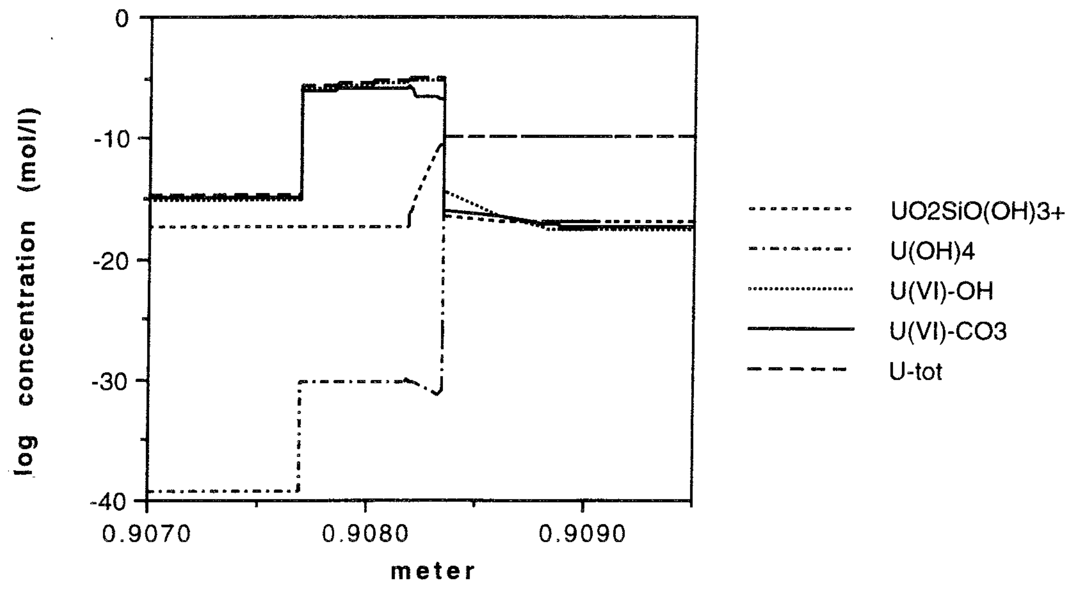


Figure 5.4.6 Detail of the uranium distribution at the redox front.

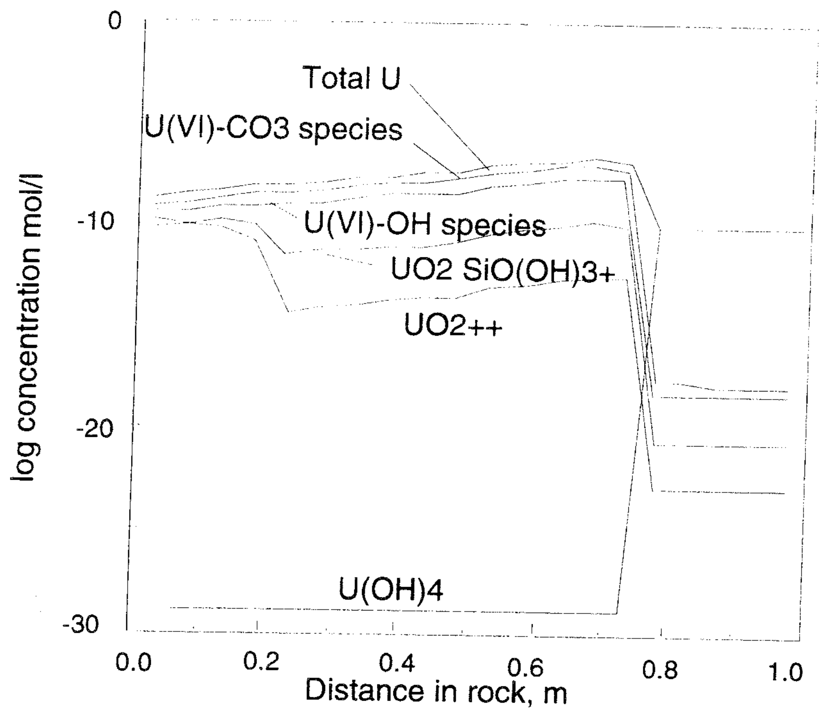


Figure 5.4.7 The uranium distribution in the calculations made by Cross et al. (1991).

Figure 5.4.8 shows the overall mineral dissolution and precipitation profiles. Three reaction zones can be seen. The reactions take place within small regions and the fronts are well separated.

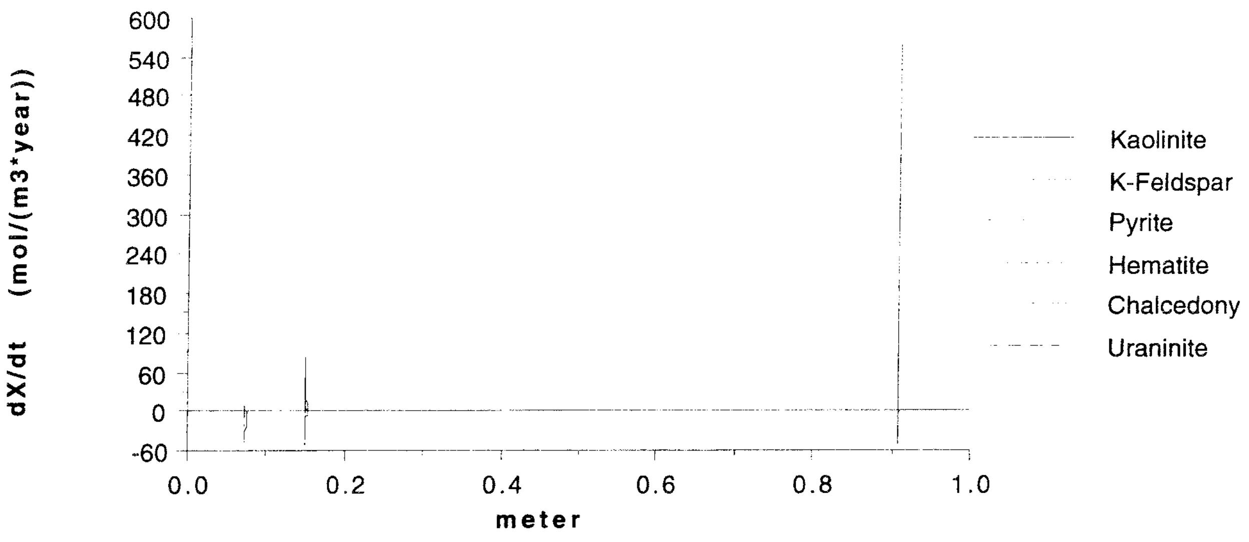


Figure 5.4.8 The precipitation and dissolution rates after 38000 years of rainwater infiltrating.

Details of the redox front are shown in figure 5.4.9. First are the peaks for uraninite dissolution caused by the time stepping routine, see section 4.5. and 4.10. The uraninite is precipitated in thin regions. Although the dissolution rate is slower than the precipitation rate, the water solution is not saturated by uranium in the first region. The dissolution of uranium continues in the following regions with uranium. When the oxygen-rich water comes in contact with the pyrite, the oxygen reacts with the pyrite. The oxygen concentration then decreases, see figure 5.4.3. When the water becomes reducing, the uranium is reduced and precipitated within a short distance.

The protons produced by the pyrite dissolution dissolve K-feldspar. The process continues until the protons are consumed. Meanwhile, chalcedony and kaolinite precipitate.

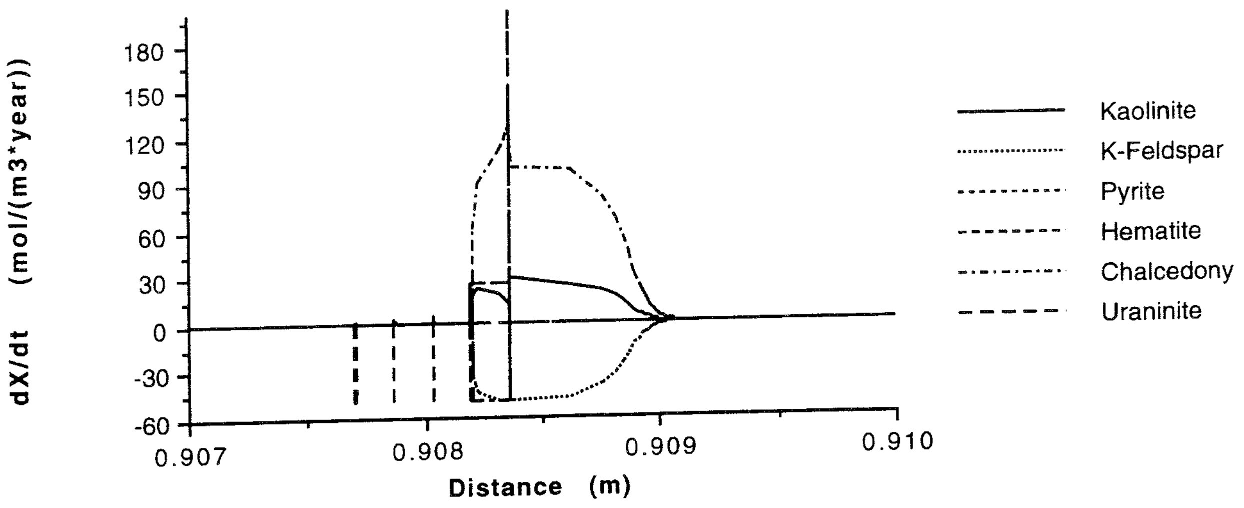


Figure 5.4.9 Details of the dissolution and precipitation at the redox front.

Figure 5.4.10 shows a detail of the K-feldspar dissolution front. The reactions are the same as the K-feldspar dissolution at the redox front, but are caused by protons from the inlet water.

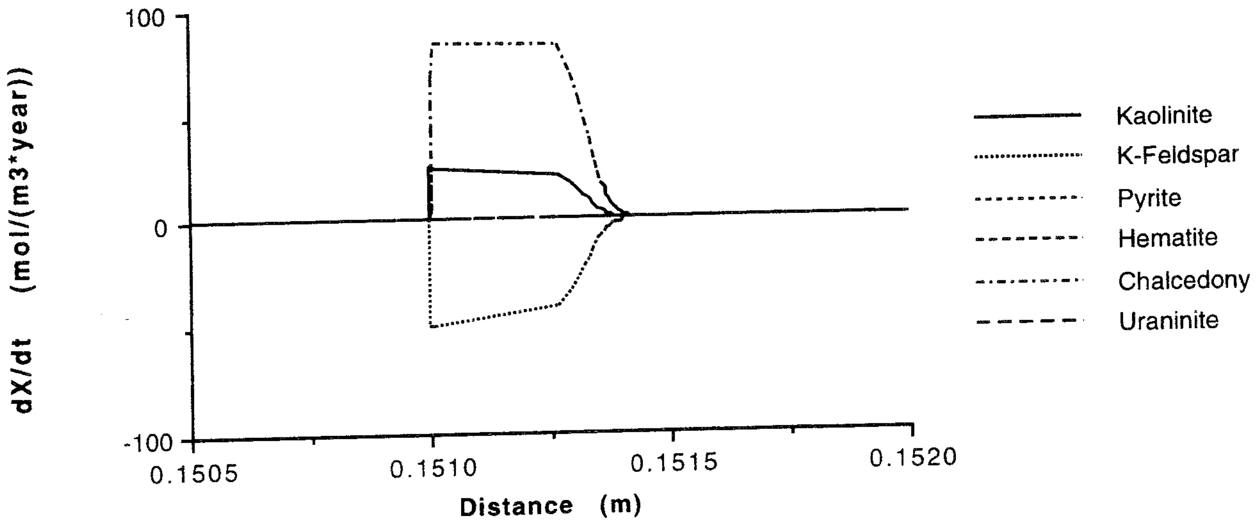


Figure 5.4.10 Details of the dissolution and precipitation at the K-feldspar dissolution front.

Figure 5.4.11 a-d shows the movement of the various dissolution fronts. When the fronts are separated from each other, they move with a constant velocity. The fast fronts separate early in the calculations, but the slow fronts need longer to separate from each other. Figure 5.4.11a shows the movement of the pyrite and uraninite fronts. These fronts are coupled and so will not separate, but move together. The uraninite peak will have a constant width.

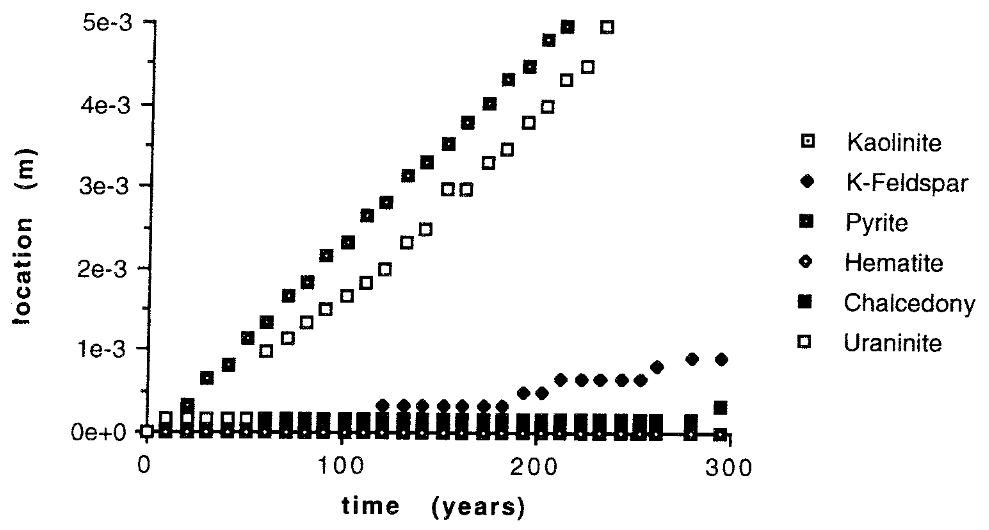


Figure 5.4.11a The movement of the dissolution fronts.

Figure 5.4.11b shows the same front positions but with another scale. The K-feldspar and chalcedony fronts are well separated, but the kaolinite and hematite fronts have not separated at all.

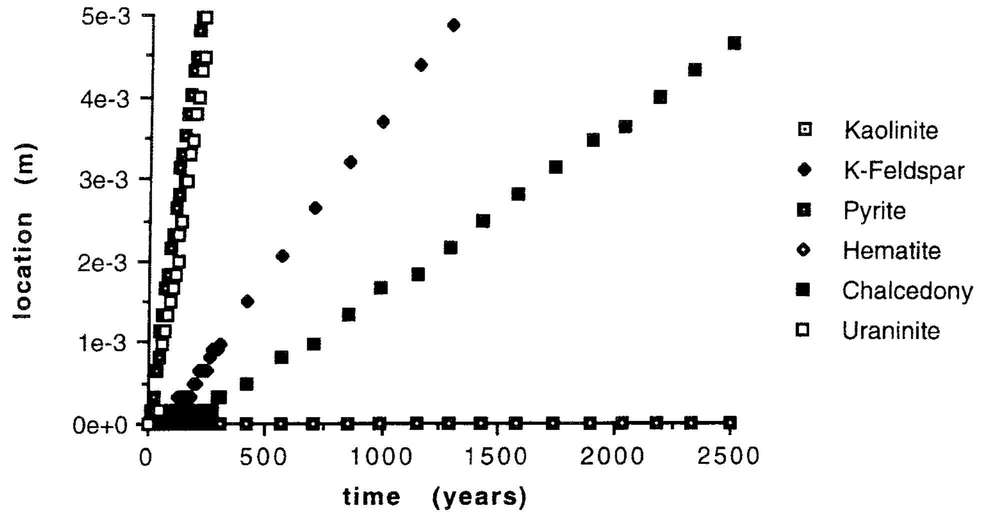


Figure 5.4.11b The movement of the dissolution fronts.

Figure 5.4.11.c shows the front movement of kaolinite. The roughness is caused by the time stepping procedure, see section 4.5.

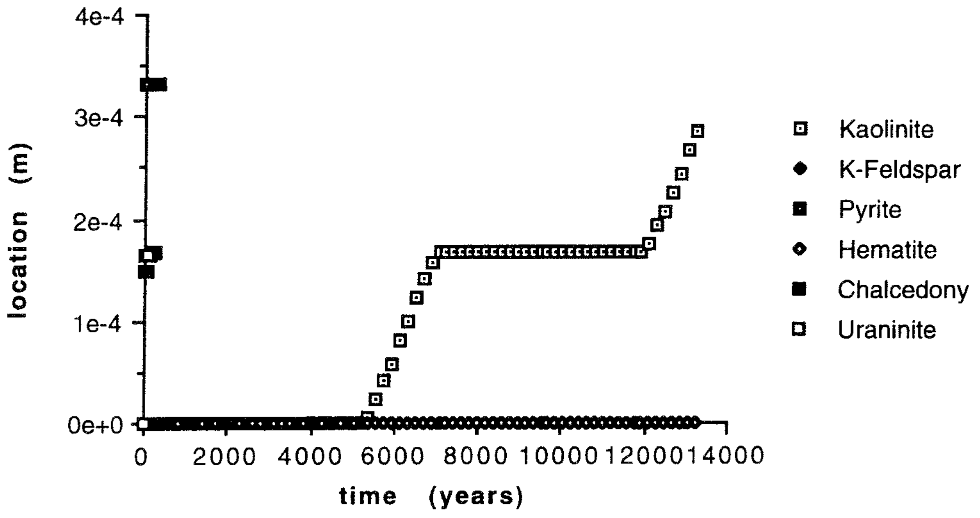


Figure 5.4.11c The movement of the dissolution fronts.

The front movement of the last front, the hematite front, is shown in figure 5.4.11d. Although the front velocity is very low, it is constant and extrapolation is possible.

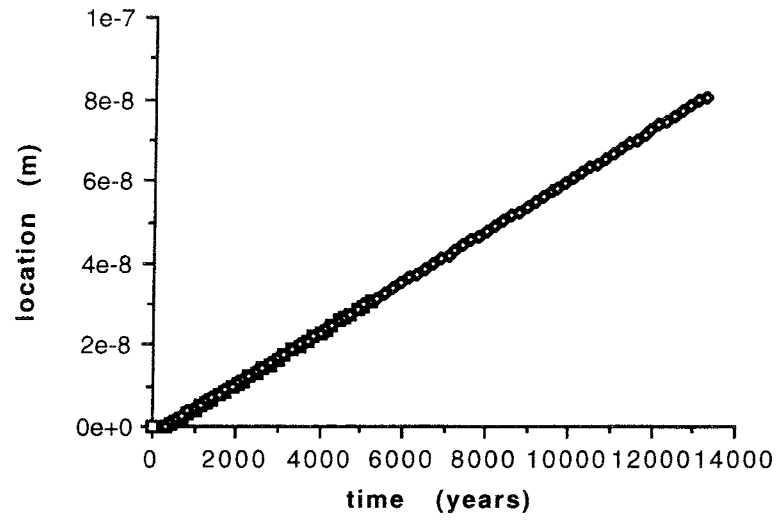


Figure 5.4.11d The movement of the hematite dissolution front.

Figures 5.4.12 and 5.4.13 show the volume fraction of the minerals after 38000 years of rainwater infiltrating. At the inlet, the rock consists of kaolinite and hematite with very low solubilities. These are the minerals that are eroded. Further downstream is chalcedony left by the K-feldspar dissolution. The redox front is at 0.9 metres. At the redox front, the uranium concentration is high. This is found in the uranium mine in Poços de Caldas.

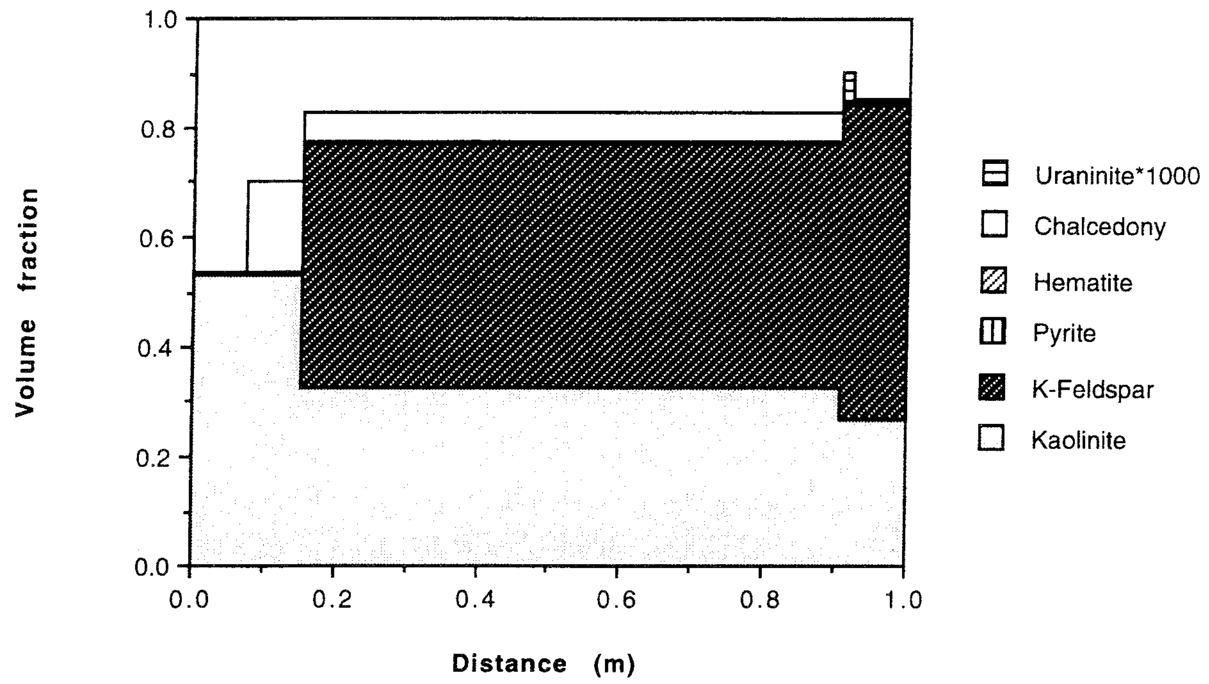


Figure 5.4.12 The mineral distribution in the column after 38000 years of infiltrating rainwater.

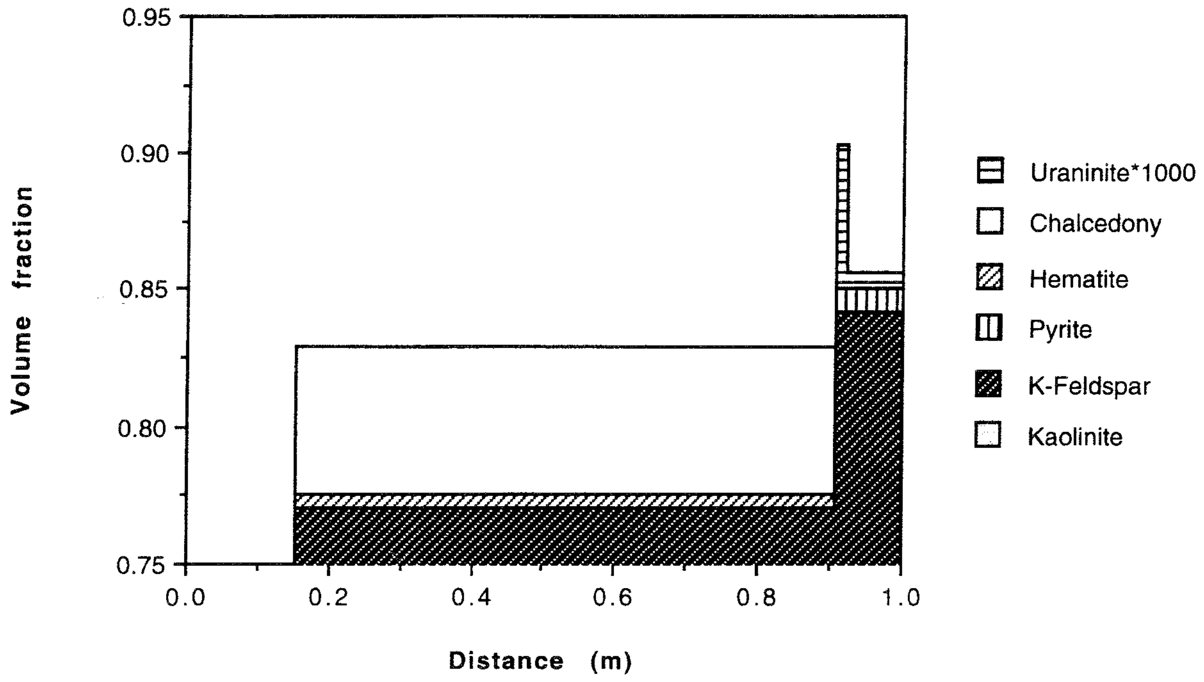


Figure 5.4.13 The upper part of figure 5.4.12.

The comparison between the results from CHEQMATE and CHEMFRONTS is satisfactory, even though the structure of the input data and the databases are different for the two programs. Cross et al. (1991) scaled the mineral content in the calculations. The results are presented in the scaled form. These differences make it difficult to compare results such as the mineral composition of the column. The differences in the position of the redox front and the potassium and sulphate concentration indicate that Cross et al. have used 20% less pyrite or 20% more oxygen in their calculations than we have.

5.5 Effects of radioactive decay in the Cigar Lake ore, Canada

The Cigar Lake uranium deposit (Cramer, 1986) is located in northern Saskatchewan at the southwestern tip of Waterbury Lake. The host rock to the ore is sandstone. The Cigar Lake deposit was discovered in 1981 and has since been extensively drilled to demonstrate the presence of sufficient ore to build a mine.

The ore body lies at a depth of 430 m in the form of an irregularly shaped lens (2000 m long by 25-100 m wide by 1-20 m high) inside a 5- to 30-m-thick clay-rich halo in the sandstone. The primary uranium minerals are uranium oxides (uraninite and pitchblende) and uranium silicate (coffinite). The ore has been dated at 1.3 billion years. The inner part of the clay-rich halo is oxidized and the outer part is reduced.

It has been suggested that the decay of uranium will cause radiolysis of water, which will produce hydrogen, oxygen or other oxidizing species and other radiation products. The hydrogen easily moves away, but some of the oxygen reacts with the uranium

$$H_2O + 0.5\ O_2\ (aq) + U(IV)\ (s) \rightarrow 2\ OH^-\ (aq) + U(VI)\ (aq) \qquad (5.5.1)$$

The hexavalent uranium is much more soluble in water than the tetravalent form.

In this example, radioactive decay is assumed to produce 10 ppm of oxygen in the water, similar to if the water has been in contact with air. The oxygenated water is then equilibrated with uraninite ($U(OH)_4$). This gives a total uranium concentration of 0.636 mmol/l. The pH of the water is 7.08. The water also contains 12.3 mmol/l of carbonate, 0.325 mmol/l of silica, 33.6 μmol/l of iron and 5.57 μmol/l of aluminium. Except for the uranium, the water concentrations are taken from field measurements (Cramer and Smellie, 1991).

This water is submitted to a column of clay composed of 6882 moles of illite

($K_{0.6}Mg_{0.25}Al_{2.3}Si_{3.5}O_{10}(OH)_2$), 308.3 moles of kaolinite ($Al_2Si_2O_5(OH)_4$), 49.06 moles of quartz (SiO_2), 147.6 moles of hematite (Fe_2O_3), and 254.4 moles of siderite ($FeCO_3$) per cubic metre of the column. There is no uraninite initially, but uraninite is allowed to precipitate. The flux of the water is taken to be 10^{-4} $m^3/(m^2 \cdot year)$.

The uranium-rich water reacts with the siderite in the column

$$U(VI)\,(aq) + 2\,FeCO_3\,(s) + 7\,H_2O \rightarrow$$

$$U(OH)_4\,(s) + Fe_2O_3\,(s) + 2\,CO_3^{2-}\,(aq) + 10\,H^+\,(aq) \qquad (5.5.2)$$

The protons produced dissolve the illite

$$K_{0.6}Mg_{0.25}Al_{2.3}Si_{3.5}O_{10}(OH)_2\,(s) + 0.75\,H_2O + 1.1\,H^+ \rightarrow$$

$$1.15\,Al_2Si_2O_5(OH)_4\,(s) + 1.2\,SiO_2\,(s) + 0.6\,K^+\,(aq) + 0.25\,Mg(II)\,(aq) \qquad (5.5.3)$$

The mineral profiles after 0.42 million years of infiltrating water are shown in figures 5.5.1a-f. Figure 5.5.1a shows that illite is dissolved in two regions: at the redox front by the protons produced when siderite dissolves, and by the protons flowing into the system with the incoming water.

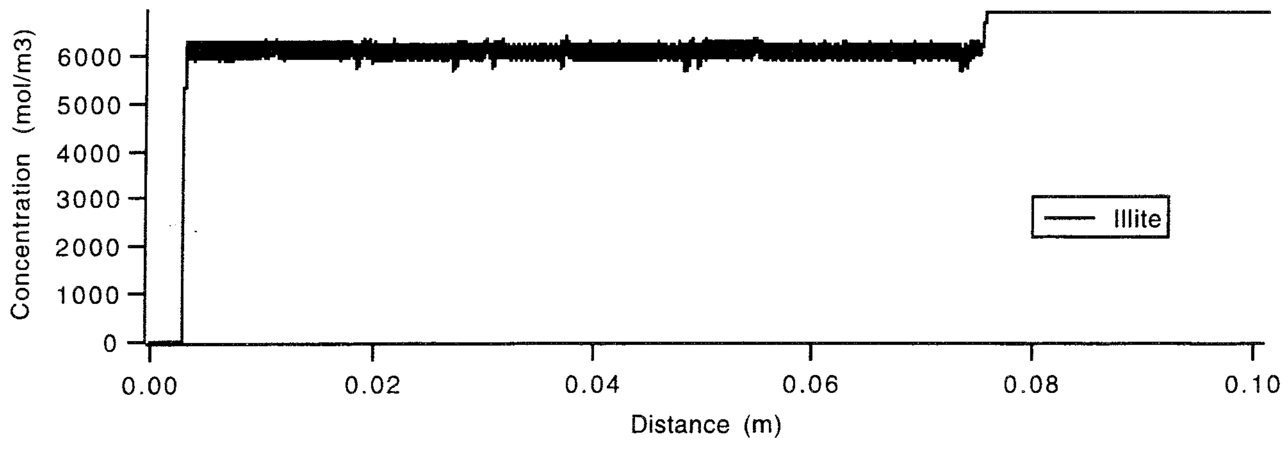


Figure 5.5.1a The illite content in the column after 0.42 million years of infiltrating water.

Figure 5.5.1b shows that kaolinite is produced when illite dissolves according to reaction (5.5.3). Quartz reacts like kaolinite according to the same reaction, (5.5.3), shown in figure 5.5.1c.

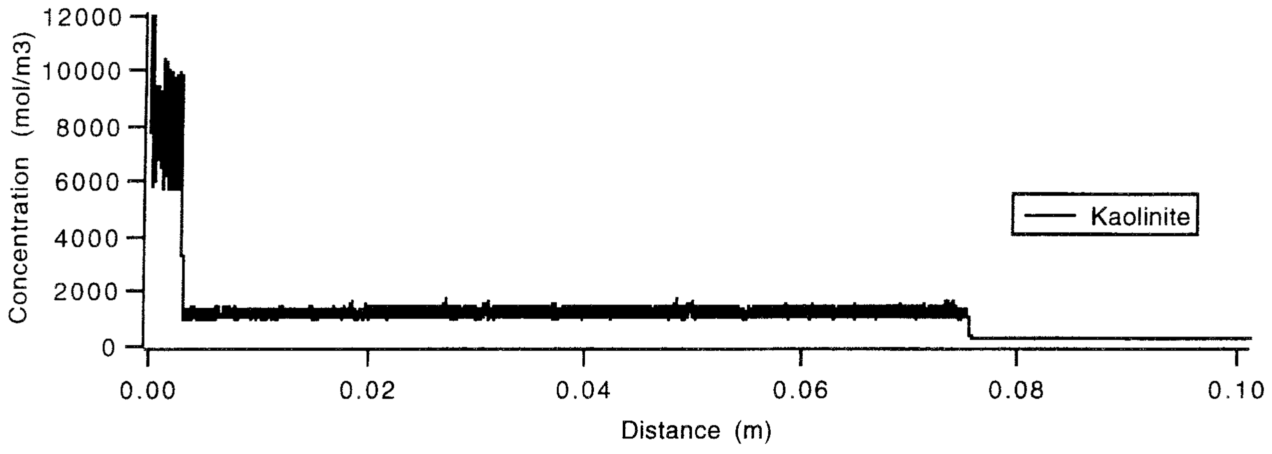


Figure 5.5.1b The kaolinite content in the column after 0.42 million years of water infiltrating.

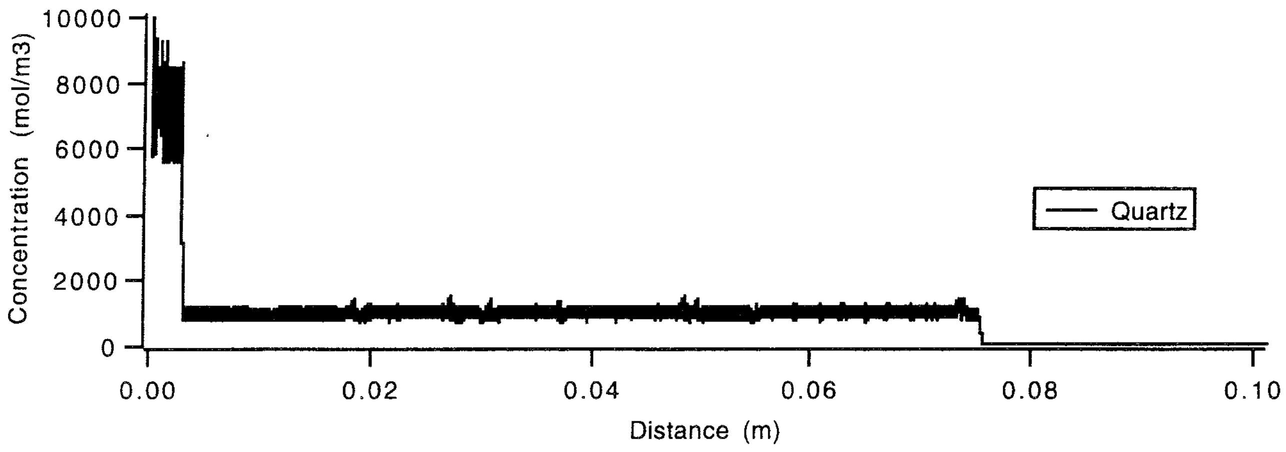


Figure 5.5.1c The quartz content in the column after 0.42 million years of water infiltrating.

Hematite is produced by the siderite dissolution according to reaction (5.5.2). The irregularities of the profile, figure 5.5.1d, are caused by the time stepping procedure, see section 4.5.

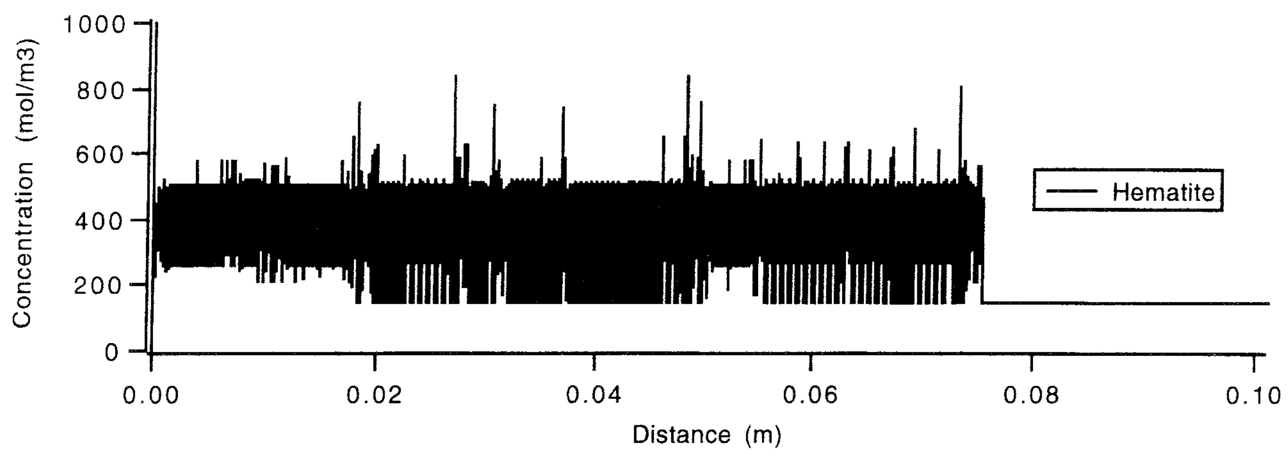


Figure 5.5.1d The hematite content in the column after 0.42 million years of water infiltrating.

Siderite, figure 5.5.1e, is dissolved by the incoming oxygen according to reaction (5.5.2). The siderite dissolution front is called the redox front.

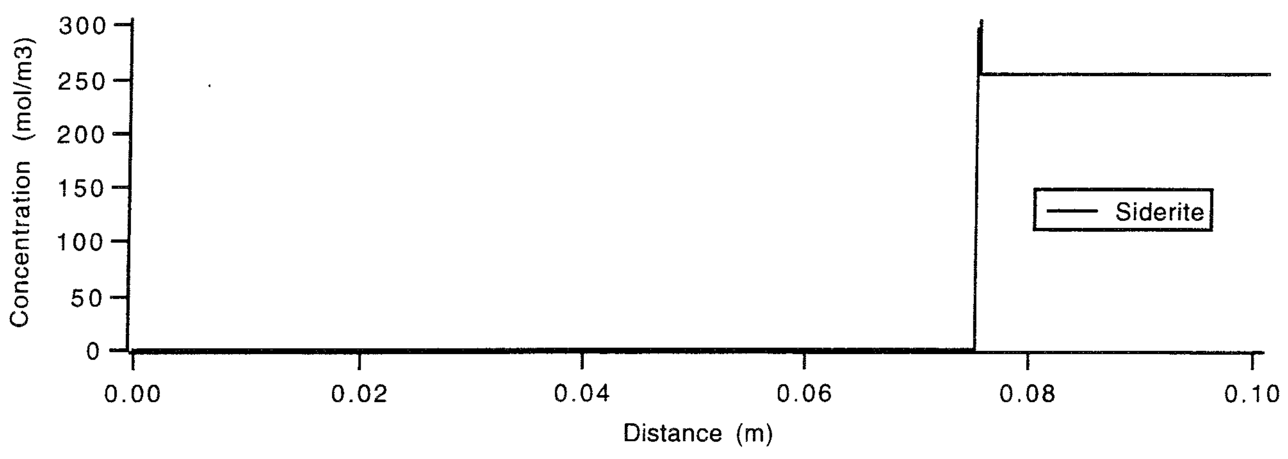


Figure 5.5.1e The siderite content in the column after 0.42 million years of water infiltrating.

Uraninite, shown in figure 5.5.1f, is precipitated when the hexavalent uranium is reduced to the tetravalent form. The irregularities in the profile are caused by the time stepping procedure used in CHEMFRONTS, see section 5.4.

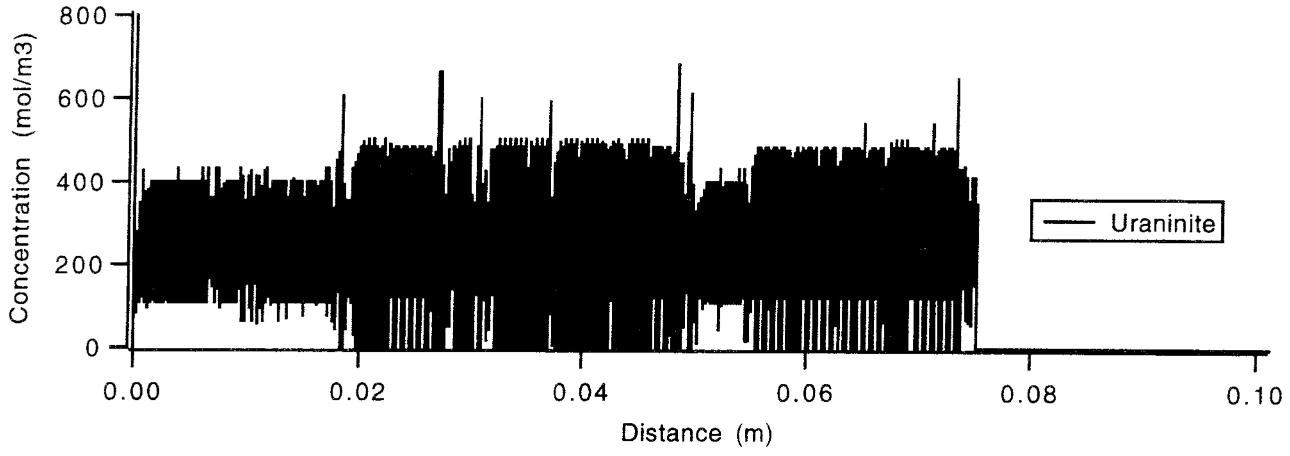


Figure 5.5.1f The uraninite content in the column after 0.42 million years of water infiltrating.

Figure 5.5.2 shows the dissolution and precipitation rates for the various minerals. The dissolution rate have negative values.

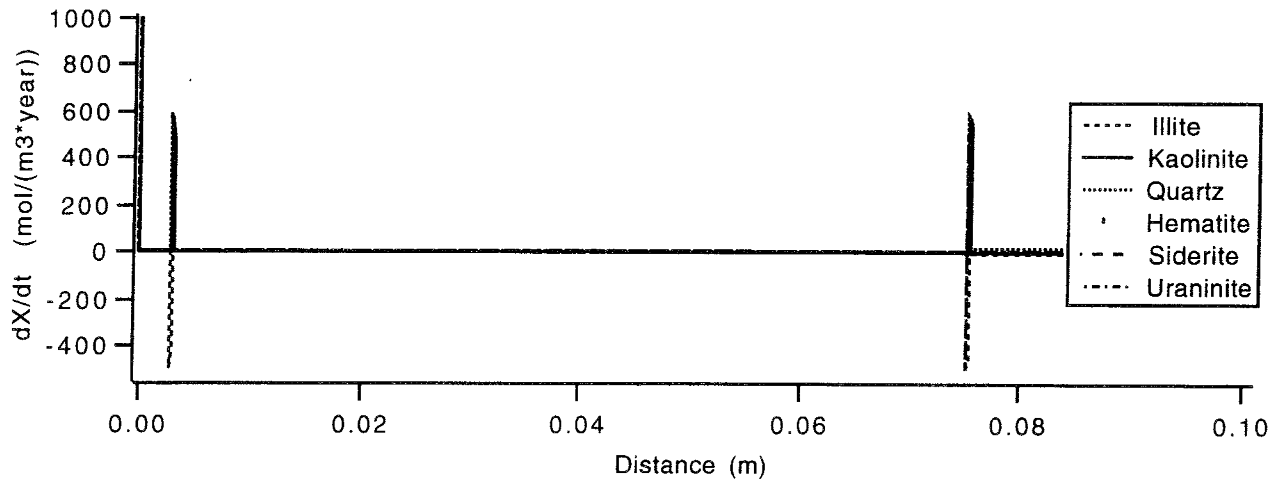


Figure 5.5.2 The precipitation and dissolution rates after 0.42 million years of water infiltrating.

Figure 5.5.3 shows the redox front in detail. At the redox front, the siderite dissolves and the ferric ions are oxidized and precipitated as hematite. The hematite precipitation curve in figure 5.5.3 follows the uraninite curve, as the same amounts are precipitated according to reaction (5.5.2). The release of protons in reaction (5.5.2) causes the dissolution of illite and the precipitation of quartz and kaolinite. As the pH increases, the free concentration of CO_3^{2-} increases due to the reaction

$$HCO_3^- \leftrightarrow CO_3^{2-} + H^+ \qquad (5.5.4)$$

This causes the precipitation of siderite seen in figure 5.5.3.

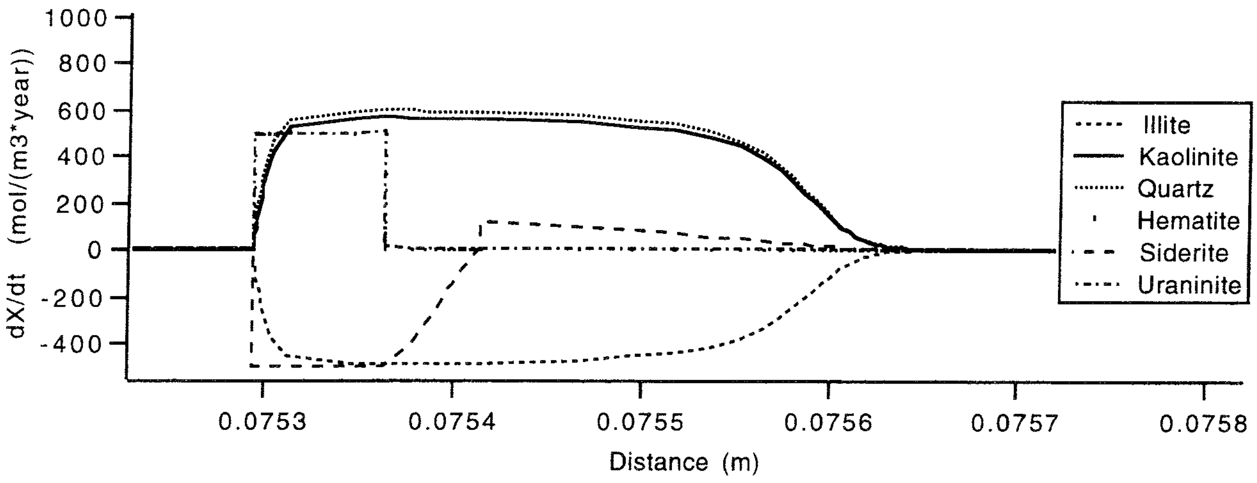


Figure 5.5.3 The precipitation and dissolution rates for the minerals at the redox front. Hematite follows uraninite.

Figure 5.5.4 shows the dissolution of illite and precipitation of kaolinite and quartz, as in reaction (5.5.3), when the water first comes into contact with illite.

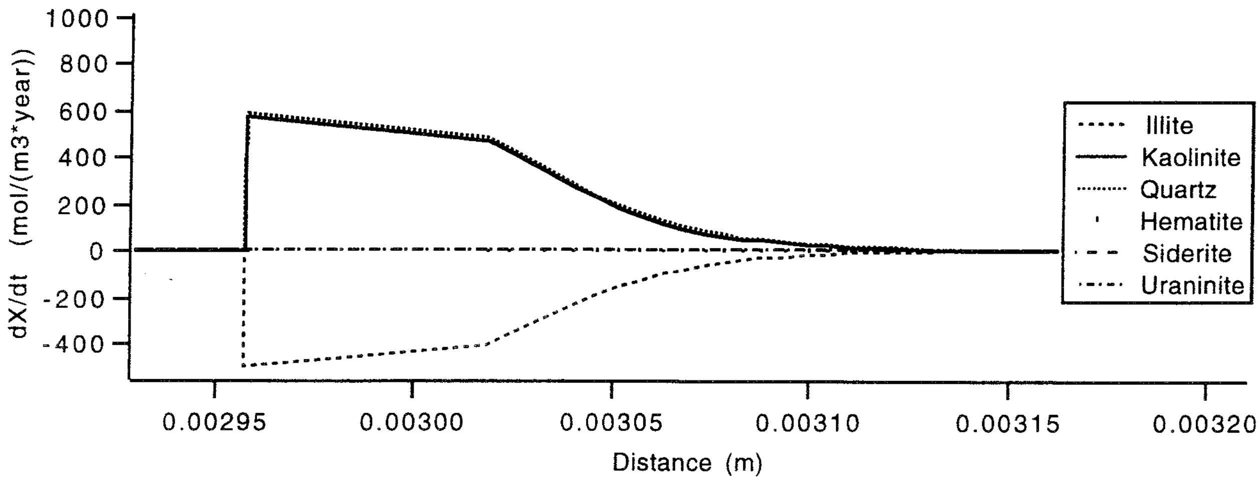


Figure 5.5.4 The dissolution and precipitation rates at the illite front after 0.42 million years of water infiltrating.

Figure 5.5.5 shows how the hydrogen ion concentration changes in the column. When the water comes into contact with the illite, the pH increases from 6.9 to 8.6.

When the water reaches the siderite, the pH first decreases to about 6 before it stabilizes at 7.8. This is caused by the dissolution of siderite, which releases a lot of protons that are consumed when the illite is dissolved, equations (5.5.2) and (5.5.3). Because of the finite dissolution rate, there are two steps in the pH profile.

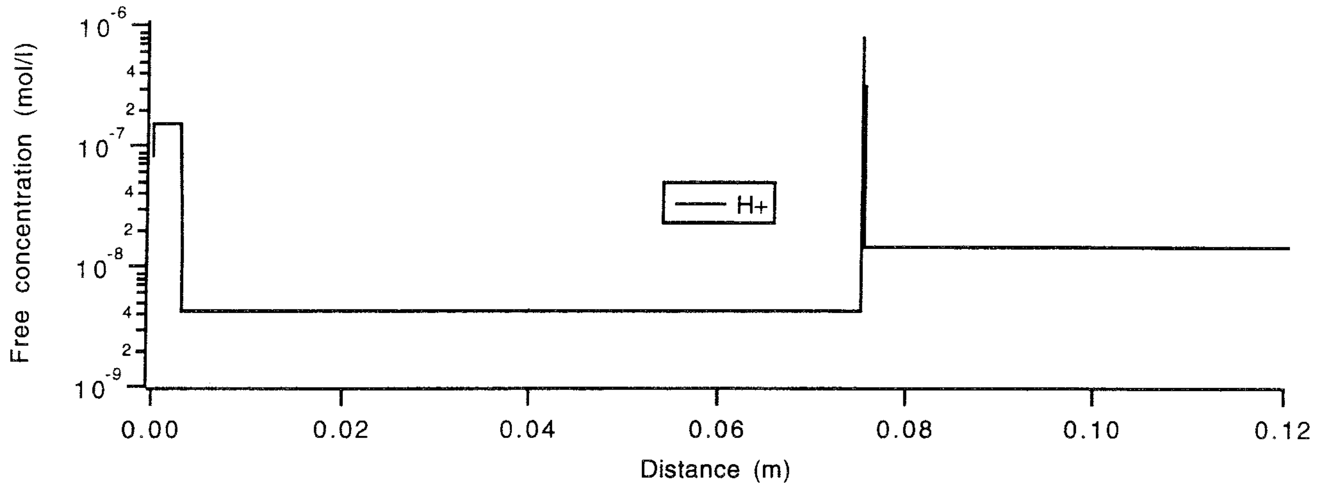


Figure 5.5.5 The free hydrogen ion concentration after 0.42 million years of water infiltrating.

Figure 5.5.6 shows how the total concentration of iron and uranium changes in the water phase when the water passes through the column. At the inlet, the concentration of iron decreases rapidly because the inlet water is supersaturated with hematite. When the water comes into contact with the illite the iron concentration decreases even more, because the pH increases and hematite is less soluble at higher pH. At the redox front, the uranium concentration decreases as hexavalent uranium is reduced to tetravalent and precipitated as uraninite. In addition, the ferric ions are oxidized and the ferrous ions form hematite when the siderite dissolves. Downstream from the redox front, the water is reducing. As iron is much more soluble in a reducing environment, the concentration is higher downstream than upstream from the redox front.

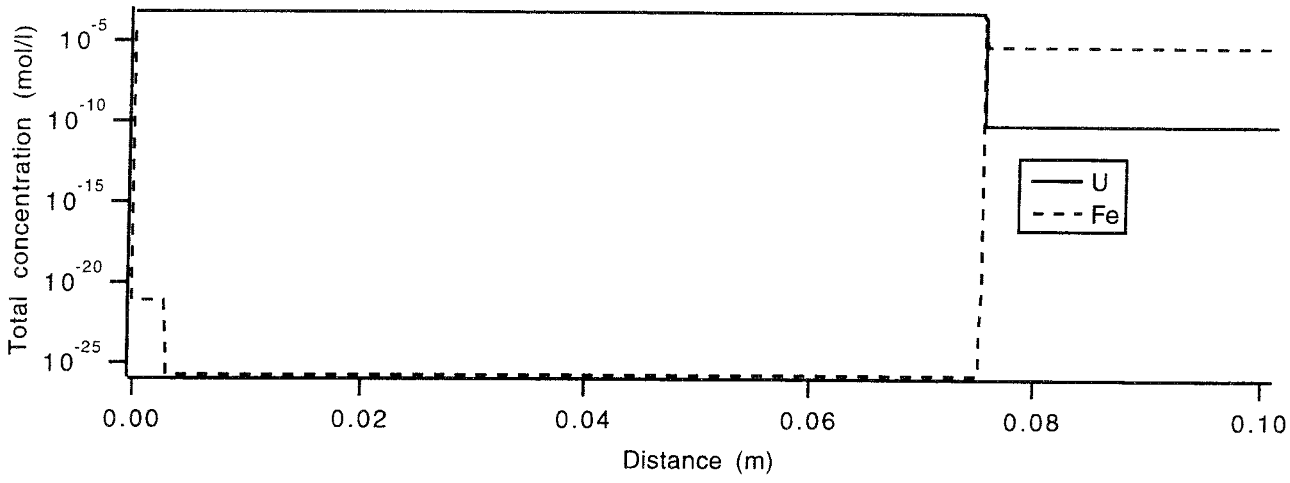


Figure 5.5.6 The total concentration of uranium and iron after 0.42 million years of water infiltrating.

Figure 5.5.7 shows the changes in total concentration of the other components. The aluminium, potassium and magnesium concentrations change when illite dissolves. The carbonate concentration changes at the redox front when the siderite dissolves. The concentration of silica only changes at the inlet, where it is supersaturated. In the rest of the column, silica is saturated.

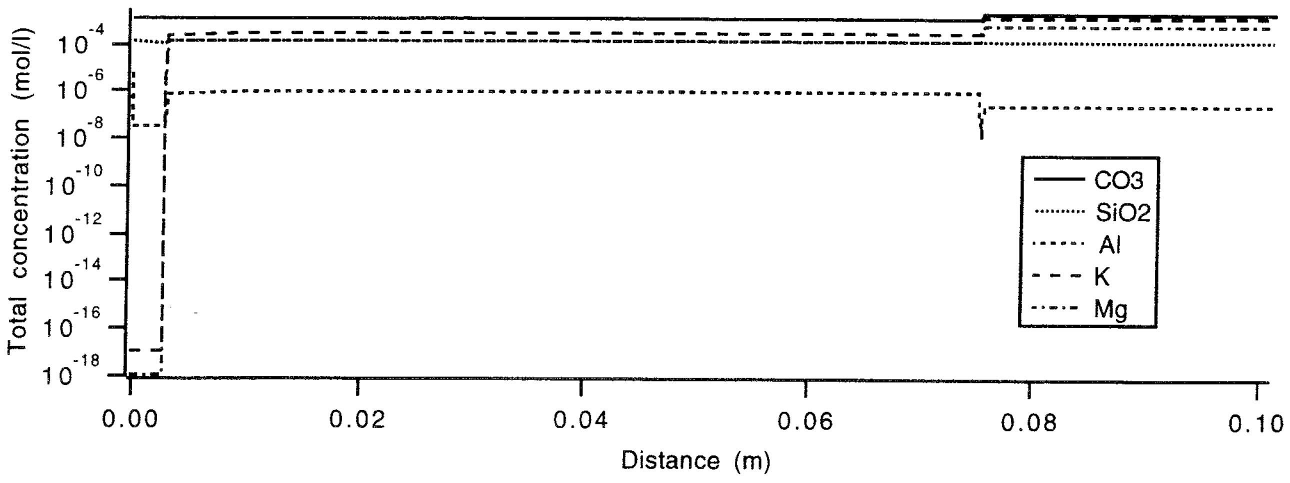


Figure 5.5.7 The total concentration of carbonate, silica, aluminium, potassium and magnesium after 0.42 million years of water infiltrating.

The positions of the various dissolution fronts are shown in figures 5.5.8 and 5.5.9. When the fronts are separated, they move with a constant velocity. The straight lines can therefore be extrapolated to any given time, so the location of the fronts can be determined without any further calculations. For example, after one billion years the siderite front would be at 178 m and the illite front would be at 7.2 metres. As the inlet water turned out to be saturated or super-saturated for the other minerals in the calculations, they will not dissolve.

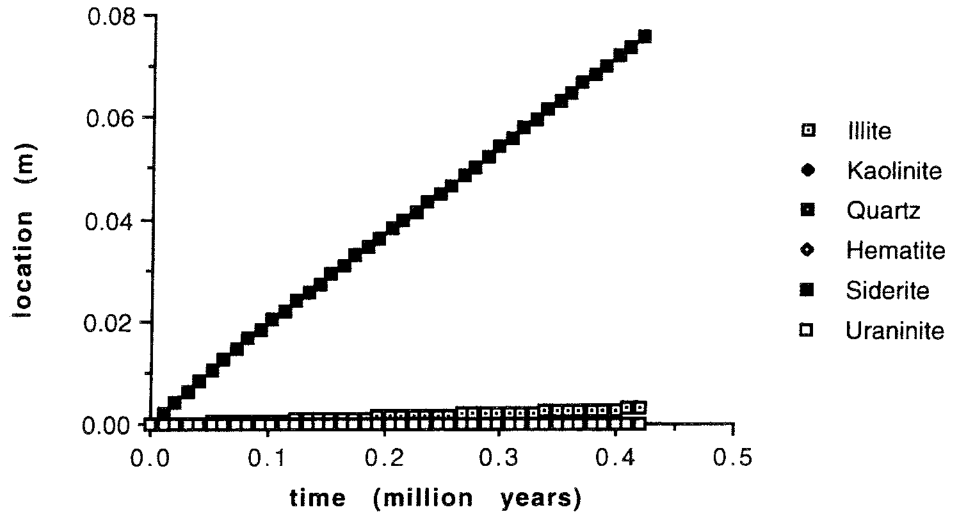


Figure 5.5.8 The position of the dissolution front for the various minerals.

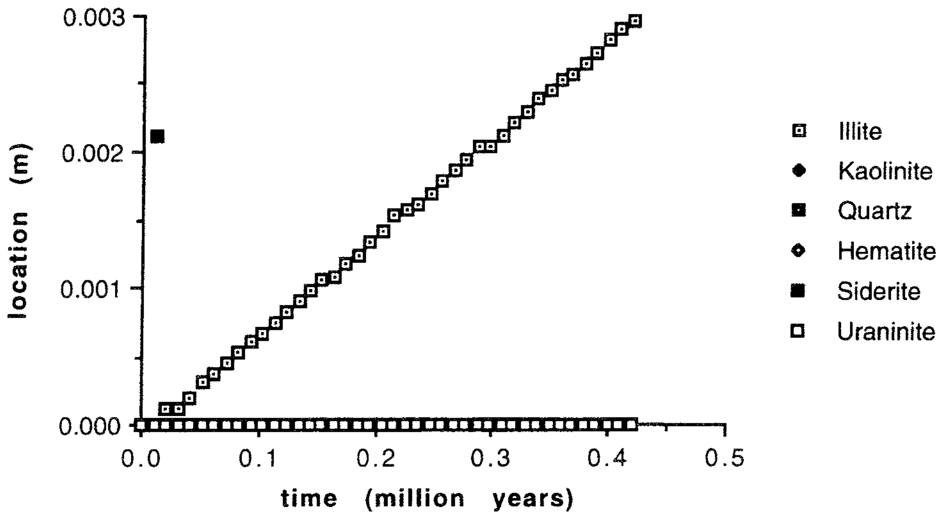


Figure 5.5.9 The position of the dissolution front for the various minerals.

The mineral distribution after one billion years of infiltration of this water is shown in figures 5.5.10 and 5.5.11. Because the incoming water is supersaturated with some of the minerals and because there is no limit on the program, the total volume at the inlet is more than 1. This, however, does not affect the rate of movement of the fronts or the chemical composition of the water.

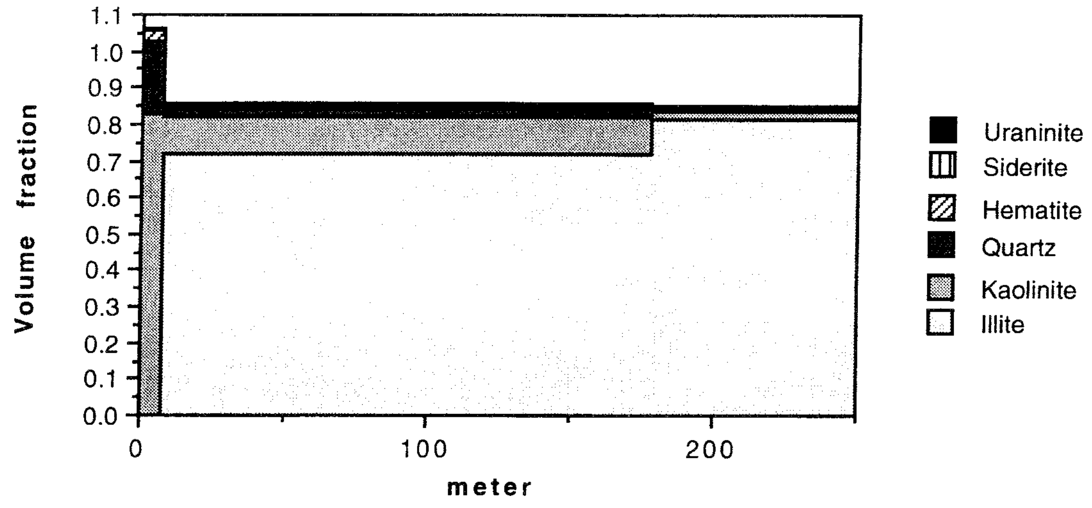


Figure 5.5.10 The mineral distribution in the column after one billion years of water infiltrating.

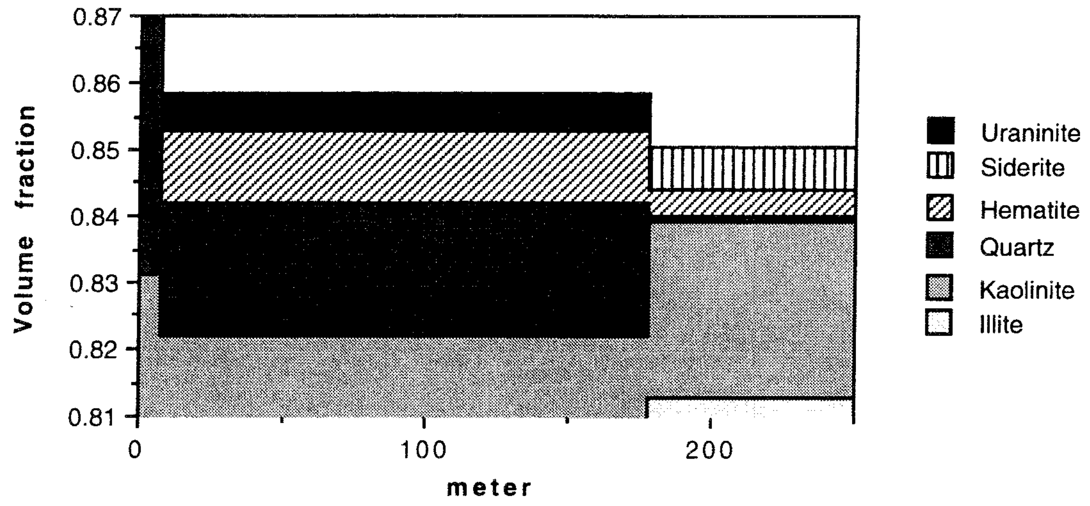


Figure 5.5.11 The upper part of figure 5.5.10.

According to this calculation, the redox fronts should be 230 metres from the ore after 1.3 billion years. At the Cigar Lake ore, only the innermost metre of the clay-rich halo is oxidized. This indicates that the radiolysis of water caused by the decay of uranium only is the source of about 0.1 ppm of oxygen, instead of 10 ppm as assumed in this example.

In this example, advection was assumed to be the only active process. Diffusion was not included, because of the limitations of the program. Transport by diffusion would be at least as important as by advection over distances up to several metres, as can be seen from the following simplified comparison.

The advective flux, N_A, of a dissolved component is u_0c, where u_0 is the flux and c is the concentration. The diffusive flux, N_D, over a distance Δz with a concentration difference Δc of c, is $D_pc/\Delta z$. For a pore diffusivity D_p of $3 \cdot 10^{-12}$ m^2/s, the two fluxes, N_A and N_D, are equal when Δz is 1 metre. This means that for shorter distances diffusion dominates the transport, whereas for longer distances advection dominates.

The mineral composition predicted by the calculations only partly agrees with those observed at Cigar Lake. As the concentrations of the minerals in the various zones are given in decreasing orders of abundance, it is difficult to compare the results. However, oxidized iron, seen as a red clay, and an increased concentration of uranium near the ore have been found. Field observations have shown that there is no quartz produced by the illite dissolution according to reaction (5.5.3). They also show that the water is supersaturated in relation to quartz (Cramer and Nesbitt, 1992).

These calculations are first attempts at modelling the conditions at Cigar Lake and may be refined later when the full data from the Cigar Lake project are available.

6 DISCUSSION AND CONCLUSIONS

The results from the calculations with CHEMFRONTS indicate that the program is a powerful tool for predicting geochemical reactions. The program has so far been able to calculate all problems tested. There is no numerical instability, but the computing time turned out to be quite long for some problems. The results from the calculations are satisfactory compared to results obtained with the programs DYNAMIX (Liu and Narasimhan, 1989a) and PHASEQL/FLOW (Walsh et al., 1984), and the comparision with results from CHEQMATE (Cross et al., 1991) is acceptable.

Diffusion is ignored in CHEMFRONTS. Flow processes are often controlled by either advective or diffusive flow. When advection is the main transport mechanism, the effect of ignoring diffusion is small. But CHEMFRONTS cannot be used for diffusion-controlled problems. The main reason for ignoring diffusion is that the numerical problem becomes much more complex when diffusion is included, if the pseudo-steady state approximation is used.

In the local equilibrium approach for solving coupled geochemical and transport models, the flow path is divided into cells. The solid and the liquid phase are assumed to be in equilibrium within the cells. If the solution is supersaturated with more than one mineral, trial and error methods are needed to determine the mineral reaction products. With the expressions for kinetic dissolution and precipitation rate, the mineral reaction products are determined directly from the transport equations.

Computer programs based on the pseudo-kinetic rate expression can describe the reactions taking place at the various reaction fronts, and can also predict the front positions. Programs based on the local equilibrium expression cannot give accurate information about the conditions of the fronts. They can predict in which cell the fronts are, but not exactly where. Special problems arise when several fronts are present in one cell. On the other hand, diffusion can be included in the kinetic rate expression, which is very complicated in the pseudo-kinetic rate expression.

As CHEMFRONTS does not include dispersion, it is possible to scale the rate of front propagation over time. When the various reaction fronts are separated, the fronts move with a constant velocity. All the reactions take place at the fronts and are thus independent of the distance between the fronts. When this constant front movement has been established, no further calculation is needed. The results can be extrapolated to any given time.

CHEMFRONTS is a stable program for calculating advective flow through porous media. It can compute sharp reaction fronts, such as redox fronts, and coupled reaction fronts. The program is not suitable for calculations of diffusion problems, as it does not account for diffusion.

7 NOTATION

A_i	the complex
A_j	the component
C_j	free concentration of component j
C_{xi}	free concentration of complex i
D_{lk}	diffusion coefficient for species l in relation to species k
i	index of the complex
I_m	rate of dissolution or precipitation
j	index of the component
$\mathbf{J}_i$	fluxes of the complexes
$\mathbf{J}_j$	fluxes of the components
k	represents both the complexes, i, and the components, j
k_m^f	the mineral reaction rate
K_{xi}	equilibrium constant for complex i
m	index of the mineral
Q_m	ion activity product of the aqueous solution
$\mathbf{r}$	distance from inlet of the column
t	time
v	flux of water
v_f	front velocity
V_m	molar volume of mineral m
$\mathbf{W}_j$	flux of mass in the system
X_m	concentration of mineral m in the solid phase
Y_j	total aqueous concentration of component j
z	distance from inlet in the z direction
α_m	specific surface of the mineral
γ_j	activity coefficient of component j
γ_{xi}	activity coefficient of complex i
ζ_m	logical factor
ν_{ij}	stoichiometric coefficients for the complex
ν_{mj}	stoichiometric coefficient for the mineral
ϕ	porosity
ϕ_m	volume fraction of mineral m
ϕ_R	total reactive volume fraction occupied by water and minerals

8 REFERENCES

Bird, R. B., W. E. Stewart, and E. N. Lightfoot, Transport phenomena, *John Wiley & Sons*, 1960.

Carnahan, C. L., Coupling of precipitation-dissolution reactions to mass diffusion via porosity changes, *ACS Symposium series No. 416, American Chemical Society, Washington*, 234-242, 1990.

Cederberg, G. A., R. L. Street, and J. O. Leckie, A groundwater mass transport and equilibrium chemistry model for multicomponent systems, *Water Resour. Res. 21*(8), 1095-1104, 1985.

Cramer, J. J., A natural analog for a fuel waste disposal vault, *Proceedings of 2nd Int. Conf. Radioactive Waste Management,* Can. Nuclear Soc. Winnipeg, Canada, Sept. 7-11, 697-702, 1986.

Cramer, J., and J. Smellie, *Second annual report of the AECL/SKB Cigar Lake Project, Year 2: 1990-1991*, CLR-91-3, 1991.

Cramer, J. J., and H. W. Nesbitt, Hydrologic, isotopic & major element constraints on genesis and evolution of water from the Cigar Lake area, *Geochemistry research branch, AECL research, Whiteshell Laboratories, Pinawa, Manitoba, Canada R0E 1L0,* 1992.

Cross, J. E., and F. T. Ewart, HATCHES - A thermodynamic database and management system, *Radiochim. Acta 52/53*, 421-422, 1991.

Cross, J. E., A. Harworth, I. Neretnieks, S. M. Sharland, and C. J. Tweed, Modeling of redox front and uranium movement in a uranium mine at Poços de Caldas, *Radiocim. Acta, 52/53*, 445-451, 1991.

Edwards, A. L., TRUMP: A computer program for transient and steady state temperature distribution in multi-dimensional system, *Rep. 14754, Rev. 3*, Lawrence Livermore Natl. Lab., Livermore, Calif., 1972.

Hagoort, HST2D-PC, Heat and solute transport simulator for personal computers, User's Manual, *Hagoort & Associates BV, Delft, The Netherlands,* 1989.

Harworth, A., S. M. Sharland, P. W. Tasker, and C. J. Tweed, A guide to the coupled chemical equilibria and migration code CHEQMATE, *Harwell Laboratory Report, NSS R113*, 1988.

Herbert, A. W., C. P. Jackson, and D. A. Lever, Coupled groundwater flow and solute transport with fluid density strongly dependent upon concentration, *DE 88752764/XAD; AERE-TP-1207*, 1987.

Ingri, N., W. Kakolowicz, L. G. Sillen, and B. Warnquist, High speed computers as a supplement of graphical methods - V. HALTAFALL: A general program for calculating the composition of equilibrium mixtures, *Talanta, 14, p 1261,* 1967.

Kahaner, D., C. Moler, and S. Nash, Numerical methods and software, *Prentice Hall series in computational mathematics, ISBN 0-13-627258-4*, 1989.

Kharaka, Y. K., and I. Barnes, *U.S. Geol. Surv. Computer Contributions, Nat'l. Technical Information Service # PB-215 899*, 81 pp., 1973.

Lichtner, P., The quasi-stationary state approximation to coupled mass transport and fluid-rock interactions in a porous medium, *Geochim. Cosmochim. Acta 52*, 143-165, 1988.

Lichtner, P., Redox front geochemistry and weathering: theory with application to the Osamu Utsumi uranium mine, Poços de Caldas, Brazil, *Submitted to Chemical Geology, Special issue on the Poços de Caldas, December 3*, 1990.

Liu, C. W., and T. N. Narasimhan, Redox-controlled multiple reactive chemical transport, 1. Model development, *Water Resour. Res.*, *25*(5), 869-882, 1989a.

Liu, C. W., and T. N. Narasimhan, Redox-controlled multiple reactive chemical transport, 2. Verification and application, *Water Resour. Res.*, *25*(5), 883-910, 1989b.

Miller, C. W., CHEMTRN user's manual, *DE-AC03-76SF00098; LBL-16152*, 1983.

Miller, C. W., and L. V. Benson, Simulation of solute transport in a chemically reactive heterogeneous system: Model development an application, *Water Resour. Res. 19*(2), 381-391, 1983.

Neretnieks, I., Diffusion, flow, and fast reactions in porous media, Migration and fate of pollutants in soils and subsoils, *NATO advanced study institute May 24 - June 5*, 1992.

Noorishad, J., and C. L. Carnahan, Development of the non-equilibrium reactive chemical transport code CHMTRNS, *DE-AC03-76SF00098; LBL-22361*, 1987.

Noy, D. J., PRECIP: A program for coupled groundwater flow and precipitation/dissolution reactions, *National Environment Research Council British Geological Survey, Technical Report WE/90/38C*, August 1990.

Parkhurst, D. L., D. C. Thorstenson, and L. N. Plummer, PHREEQE - A computer program for geochemical calculations, *Report USGS/WRI 80-96, NTIS Tech. Rep. PB81-167801*, 1980.

Perkins, E. H., Y. K. Kharaka, W. D. Gunter, and J. D. DeBraal, Geochemical modeling of water-rock interactions using SOLMINEQ.88, *ACS Symposium series No. 416, American Chemical Society, Washington, 117-127*, 1990.

Puigdomènech, I., and J. Bruno, Plutonium solubilities, *Swedish Nucl. Fuel Waste Manag. Co., Box 5864, S-102 48 Stockholm, SKB Techn. Rep. 91-04, 78 pp*, 1991.

Stumm, W., and J. J. Morgan, Aquatic chemistry, an introduction emphasizing chemical equilibria in natural waters, 2nd ed., *John Wiley & Sons, ISBN 0-471-04831-3*, 1981.

Truesdell, A. H., and B. F. Jones, WATEQ, a computer program for calculating chemical equilibria of natural waters, *J. Res. U.S. Geol. Surv.* *2*(2), 233-248, 1974.

Walsh M. P., S. L. Bryant, R. S. Schechter, and L. W. Lake, Precipitation and dissolution of solids attending flow through porous media, *AIChE J.*, *30*(2), 317-328, 1984.

Wanner, H., TDB-0.1, The NEA thermochemical data base project, NEA-TDB, OECD Nuclear Energy Agency, Data Bank, F-91191 Gif-sur-Yvette, France, January, 1990.

Westall, J. C., J. L. Zachary, and F. M. M. Morel, MINEQL: A computer program for the calculation of chemical equilibrium composition of aqueous system, *Tech. Note 18, 91 pp., Dep. of Civ. Eng., MIT, Cambridge, Mass.*, 1976.

Westall, J., MICROQL: 1 A chemical equilibrium program in BASIC, *EAWAG, Swiss Fed. Inst. of Technol., Duebendorf, Switzerland*, 1979.

Willemsen, A., PHREEQM-2D, PHREEQE in a multicomponent mass transport model, coupled to HST2D, Manual and user guide version 0.1, *IF Technology bv, Frombergsrtaat 1, 6814 EA Arnhem, Nederland*, 1992.

Wolery, T., EQ3NR a computer program for geochemical aqueous speciation-solubility calculations: user's guide and documentation, *Lawrence Livermore National Laboratory, April 18, UCRL-53414*, 1983.

Wolery, T. J., and S. A. Daveler, EQ6 a computer program for reaction path modeling of aqueous geochemical systems: user's guide and documentation, *Lawrence Livermore National Laboratory, March 30*, 1989.

Yeh, G. T., and V. S. Tripathi, A model for simulating transport of reactive multispecies components: model development and demonstration, *Water Resour. Res., 27*(12), 3075-3094, 1991.

APPENDIX A

User´s guide to CHEMFRONTS

CHEMFRONTS consists of many different subroutines and functions, for the flow sheets se appendix B, and appendix C for the source code. These are divided into six different files, *timstep.f*, *main.f*, *util.f*, *solver.f*, *input.f*, and *rec.f*. CHEMFRONTS uses two input files, *exinput* and *ode.dat.*

The input file , *exinput*, defines the chemical problem. It contains the number of components, minerals, and complexes, and also the names of the components and the complexes. The components are chosen so that for every complex there is only one possible combination of the components used to form the complex. On the other hand there are several possible ways of choosing the components.

For example, formation of the complex $CaHCO_3^+$. If the components are Ca^{2+}, H^+, and CO_3^{2-} the chemical reaction would be:

$$Ca^{2+} + H^+ + CO_3^{2-} \rightarrow CaHCO_3^+ \qquad (1)$$

If the components were Ca^{2+} and HCO_3^- the reaction would be:

$$Ca^{2+} + HCO_3^- \rightarrow CaHCO_3^+ \qquad (2)$$

If the components were $CaOH^+$, CO_3^{2-}, and OH^- the reaction would be:

$$CaOH^+ + CO_3^{2-} + H_2O \rightarrow CaHCO_3^+ + 2\ OH^- \qquad (3)$$

Water are not counted as a component in our examples, because the reactions are taken place in water solution and the concentration of water do not change noticeably. The water concentration is also included in the equilibrium constant for the complexes in the database (Wolery, 1983) used for the examples in this report. The components chosen to define the chemical problem do not influence the calculations, but it is convenient to use the same components as are used in the database.

The input file contains the stoichiometric coefficients for the complexes and the minerals. In formula (1) above the stoichiometric coefficients would be 1, 1, 1, and in formula (3) they would be 1, 1, -2.

The concentrations of the components can be given both as free and as total concentrations in the input file. The analytical concentration of elements are often given in total concentration, but for hydrogen the concentration often are represented by the pH, and the oxygen concentration by the eh of the water solution.

Table 1 shows an example of the input file *exinput* used in the calculations described in chapter 5.4 in the final report. The first data are the number of components, minerals and complexes. The text written with upper case letters are only information to help the user. It is red by the program but not used. Then comes a list of the components followed by a list of the complexes. This information is used in the output files to make them easier to follow.

Next comes a table with the stoichiometric coefficients for the complexes followed by the minerals. The first number is the number of the complex or the mineral. This is not used by the program. The second number is the stoichiometric coefficient for the first component etc. For example for complex number 1, 1.5 of component number 1, -11 of component number 2 and 3 of component number 4. This gives the reaction

$$1.5\ O_2 - 11\ H^+ + 3\ U^{4+} \rightarrow (UO_2)_3(OH)_5^+ \qquad (4)$$

As water is not included in the calculation the eight water that should be on the left side of the equation 4 is omitted. The stoichiometric coefficients for the minerals are used in the same way.

Then comes a list of the concentration of the components followed by a letter. For component 1 the concentration is $0.3100\cdot10^{-3}$ mol/l. The letter "f" stands for free concentration. This means that there is $0.3100\cdot10^{-3}$ mol/l of O_2(aq) in the solution. For component number 3 the concentration is $0.1600\cdot10^{-3}$ mol/l. The letter "t" stands for total concentration. This means that there is $0.1600\cdot10^{-3}$ mol/l of all forms of carbonate in the solution together, HCO_3^-, CO_3^{2-}, CO_2(aq), $(UO_2)_3(CO_3)(OH)_3^+$ etc.

The number of cells is an output parameter. The column is divided into a number of cells with equal size. The average mineral concentration within each cell is reported in a table. The porosity of the column is in this example 0.15 m^3/m^3. The mineral concentrations are given in the unit mol/m^3.

The specific surfaces for the minerals are in this program assumed to be equal for all the minerals. This affects the mineral dissolution rate. In this example it is taken to be 50 m^2/m^3.

EQMODE = 1 means that the reaction rate constant is computed by the program due to equation 5.

Reaction rate constant = CONST / Equilibrium constant (5)

If EQMODE = 0 the reaction rates for the minerals (mol/(m^3·year)) are given instead of CONST. The area of the column is given in square meters, the flux in m^3/(m^2·year), and the length of the column in meters. The number of points is the initial value. This changes when the program needs more points for the calculations.

The mineral molar volume is used for volume fraction calculations and for porosity calculations. The unit is cm^3/mol. The mineral and complex equilibrium constants are taken for the reaction of formation of the mineral or the complex, as for example in equation 4.

Table 1. An example of the input file *exinput.*

```
FILE EXINPUT
COMPONENTS
9
MINERALS
6
COMPLEXES
48
NAME OF THE COMPONENTS
o2(aq)
h+
hco3-
u++++
sio2(aq)
fe++
al+++
k+
so4--
NAME OF THE COMPLEXES
(uo2)3(oh)5+
(uo2)4(oh)7+
(uo2)2(oh)2++
uo2oh+
(uo2)3(oh)4++
uo2(oh)2
(uo2)2oh+++
oh-
uo2(oh)3-
co2(aq)
(uo2)3(oh)7-
uo2co3
(uo2)3(co3)(oh)3+
uo2(oh)4--
```

```
co3--
uo2+
fe(oh)2+
aloh++
al(oh)2+
al(oh)3
fe(oh)3
feoh++
uo2sio(oh)3+
al(oh)4 -
uo2(co3)2--
fe+++
h3sio4-
hs-
uo2(co3)3----
al2(oh)2++++
u(oh)3+
fe(oh)2
feh3sio4++
feco3
u(oh)2++
fe2(oh)2++++
uoh+++
fe(oh)3-
al3(oh)4+++++
u+++
h6(h2sio4)4--
h4(h2sio4)4----
feco3+
fe(oh)4-
koh
feoh+
u(oh)4
h2s
COMPLEXES\COMPONENTS
         1        2         3        4        5        6        7        8        9

1        1.5    -11.        0.       3.       0.       0.       0.       0.       0.
2        2.     -15.        0.       4.       0.       0.       0.       0.       0.
3        1.      -6.        0.       2.       0.       0.       0.       0.       0.
4        0.5     -3.        0.       1.       0.       0.       0.       0.       0.
5        1.5    -10.        0.       3.       0.       0.       0.       0.       0.
6        0.5     -4.        0.       1.       0.       0.       0.       0.       0.
7        1.      -5.        0.       2.       0.       0.       0.       0.       0.
8        0.      -1.        0.       0.       0.       0.       0.       0.       0.
9        0.5     -5.        0.       1.       0.       0.       0.       0.       0.
10       0.       1.        1.       0.       0.       0.       0.       0.       0.
11       1.5    -13.        0.       3.       0.       0.       0.       0.       0.
12       0.5     -3.        1.       1.       0.       0.       0.       0.       0.
13       1.5    -10.        1.       3.       0.       0.       0.       0.       0.
14       0.5     -6.        0.       1.       0.       0.       0.       0.       0.
15       0.      -1.        1.       0.       0.       0.       0.       0.       0.
16       0.25 -3.0.         1.       0.       0.       0.       0.       0.
```

```
 17    0.25 -1.0.    0.      0.      1.      0.      0.      0.
 18    0.     -1.    0.      0.      0.      0.      1.      0.      0.
 19    0.     -2.    0.      0.      0.      0.      1.      0.      0.
 20    0.     -3.    0.      0.      0.      0.      1.      0.      0.
 21    0.25 -2.0.    0.      0.      1.      0.      0.      0.
 23    0.25    0.    0.      0.      0.      1.      0.      0.      0.
 24    0.5    -3.    0.      1.      1.      0.      0.      0.      0.
 25    0.     -4.    0.      0.      0.      0.      1.      0.      0.
 26    0.5    -4.    2.      1.      0.      0.      0.      0.      0.
 27    0.25    1.    0.      0.      0.      1.      0.      0.      0.
 29    0.     -1.    0.      0.      1.      0.      0.      0.      0.
 30   -2.      1.    0.      0.      0.      0.      0.      0.      1.
 31    0.5    -5.    3.      1.      0.      0.      0.      0.      0.
 34    0.     -2.    0.      0.      0.      0.      2.      0.      0.
 35    0.     -3.    0.      1.      0.      0.      0.      0.      0.
 36    0.     -2.    0.      0.      0.      1.      0.      0.      0.
 37    0.25    0.    0.      0.      1.      1.      0.      0.      0.
 38    0.     -1.    1.      0.      0.      1.      0.      0.      0.
 39    0.     -2.    0.      1.      0.      0.      0.      0.      0.
 40    0.5     0.    0.      0.      0.      2.      0.      0.      0.
 42    0.     -1.    0.      1.      0.      0.      0.      0.      0.
 44    0.     -3.    0.      0.      0.      1.      0.      0.      0.
 46    0.     -4.    0.      0.      0.      0.      3.      0.      0.
 49   -0.25 -1.0.    1.      0.      0.      0.      0.      0.
 50    0.     -2.    0.      0.      4.      0.      0.      0.      0.
 51    0.     -4.    0.      0.      4.      0.      0.      0.      0.
 53    0.25    0.    1.      0.      0.      1.      0.      0.      0.
 54    0.25 -3.0.    0.      0.      1.      0.      0.      0.
 56    0.     -1.    0.      0.      0.      0.      0.      1.      0.
 57    0.     -1.    0.      0.      0.      1.      0.      0.      0.
 58    0.     -4.    0.      1.      0.      0.      0.      0.      0.
 59   -2.      2.    0.      0.      0.      0.      0.      0.      1.

MINERALS\COMPONENTS
        1       2      3       4       5       6       7       8       9

1      0.     -6.    0.      0.      2.      0.      2.      0.      0.
2      0.     -4.    0.      0.      3.      0.      1.      1.      0.
3     -3.5     2.    0.      0.      0.      1.      0.      0.      2.
4      0.5    -4.    0.      0.      0.      2.      0.      0.      0.
5      0.      0.    0.      0.      1.      0.      0.      0.      0.
6      0.     -4.    0.      1.      0.      0.      0.      0.      0.

CONCENTRATION, COMPONENTS
1  0.3100d-3     f
2  0.7940d-5     f
3  0.1600d-3     t
4  0.1808d-17    t
5  0.1000d-16    t
6  0.7581d-17    t
7  0.2622d-17    t
8  0.1000d-16    t
```

```
 9  0.9985d-17    t
 NUMBER OF CELLS
 50
 POROSITY
 0.15
 MINERAL CONCENTRATION
 1    2682
 2    5282
 3     345
 4       0.
 5       0.
 6       0.2701
 SPECIFIC SURFACE
 50
 EQMODE
 1
 CONST
 1
 AREA
 1.
 FLUX
 0.1
 LENGTH OF COLUMN
 1.D0
 NUMBER OF POINTS
 1000
 MINERAL MOLAR VOLUME
 1       99.520
 2      108.870
 3       23.940
 4       30.274
 5       22.688
 6       24.618
 MINERAL EQUILIBRIUM CONSTANT
 1        7.4292
 2        0.0832
 3      207.2535
 4      -15.4827
 5       -3.7281
 6       -4.8388
 COMPLEXES EQUILIBRIUM CONSTANT
 1       81.8275
 2      108.0554
 3       59.3300
 4       27.2461
 5       85.4845
 6       20.4373
 7       62.1807
 8      -13.9893
 9       12.5290
 10       6.3660
 11      65.0580
```

12	31.6494
13	89.8878
14	0.7358
15	-10.3438
16	13.1250
17	2.0941
18	-4.9345
19	-10.1035
20	-16.1667
21	-4.2550
23	5.5736
24	30.0954
25	-22.1567
26	28.4376
27	7.7630
29	-9.8088
30	-132.5463
31	22.8069
34	-7.6793
35	-3.9344
36	-21.4222
37	7.1766
38	-6.6636
39	-2.3664
40	12.5765
42	-0.3952
44	-34.2245
46	-13.8714
49	-29.3961
50	-13.4341
51	-35.7315
53	7.1378
54	-13.8828
56	-14.4877
57	-10.1790
58	-5.1359
59	-125.5963

Ode.dat contains information about the calculations. Table 2 shows an example of *ode.dat.* XSTEP is the maximum distance between the saved points.

CDIFF is the maximum relative difference for the concentrations of the components before a front to be taken as equal. If the aqueous concentrations are equal to the concentrations in the previous timestep and the mineral distributions are equal, the saved profiles can be used for this timestep too. This procedure saves a lot of computation time without changing the computation results.

EPS is requested relative error for the solver. EWT is the minimum concentration taken into account, smaller value are set to zero. In this example zero is chosen as the minimum value. This makes the smallest value of the computer to the smallest value of the calculation. HMAX is the maximum step size for the solver.

STSAVE indicates if information about the calculation will be saved so the calculation can be restarted. STREST indicates if the calculation is a restart or not. In this example information shall be saved and it is not a restart.

DOFF is the minimum dissolution or precipitation rate to be taken into account. The dissolution/precipitation profiles are often oscillating a little due to numerical accuracy even if the concentration profiles are stable. This phenomena is caused by the numerical accuracy in the calculations. TIMA is the maximum timestep. IPR indicates how often output will be written. As the tables can be very long it is advisable not to save information for every timestep if the calculation is long. If IPR is 1 the information is saved for every timestep. If IPR is 100 only information for timestep 1, 101, 201...etc. are saved.

Table 2. An example of *ode.dat.*

0.6D-1	0.1D-1		! XSTEP, CDIFF
0.1D-4	0.0D0	0.1D0	! EPS,EWT,HMAX
1	0		! STSAVE, STREST
3.D0	10	1	! DOFF, TIMA, IPR

CHEMFRONTS produces twelve output files. Three of them are *mydata1*, *mydata2*, and *mydata3*. *Mydata1* contains information about the input files, exinput and ode.dat. The listing if the input information is followed by lists of the vector XPOINT and the matrix POINTS. XPOINT contains the coordinates for the values in POINTS. POINTS contains information about the mineral concentration at every coordinate in XPOINT. The concentration is presented in moles per cubic meter, and is assumed to be constant from one coordinate to the next. These tables can be quite lengthy if the mineral profile changes a lot. The information for the first ten minerals are written in mydata1, for mineral 11 to 20 in mydata2, and the rest in mydata3.

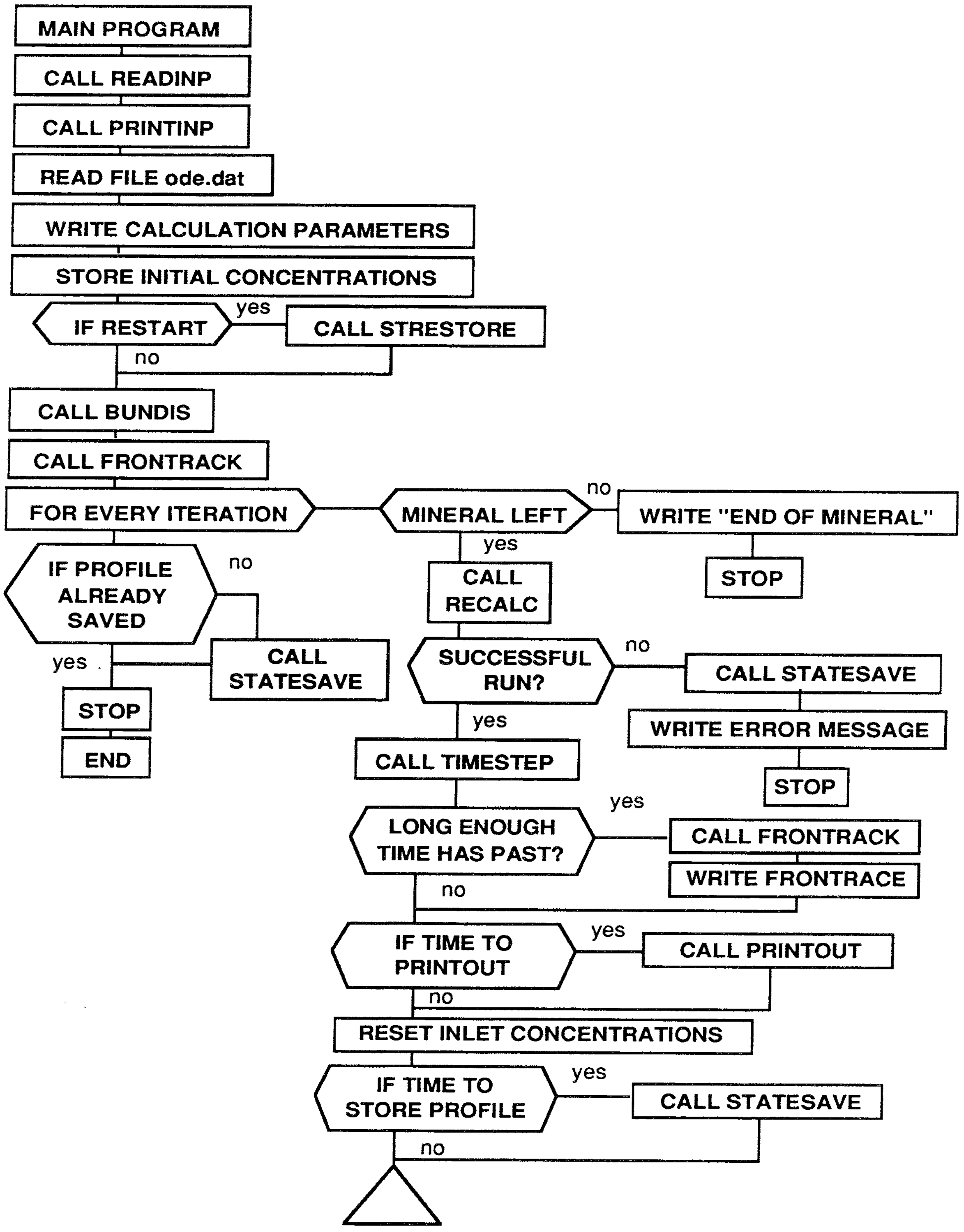
Out1, *out2*, and *out3* contains the vector XP and the matrix DXDT. XP contains the coordinates chosen by the solver in meter. DXDT contains the mineral precipitation and dissolution rates in moles per year. If the value is negative it is the dissolution rate, and positive for the precipitation rate. *Out1* contains information about the first ten minerals, *out2* for mineral 11 to 20, and *out3* for mineral 21 to 30.

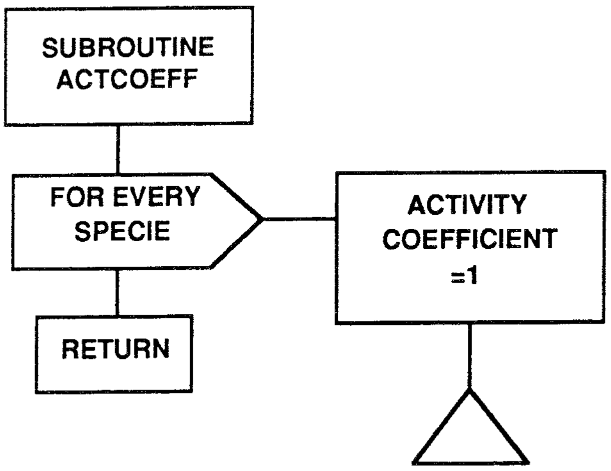
The file *conc.dat* contains the vector XP and the matrix YP. These are the results from the solver when the equation 3.1.26 is solved. XP contains the coordinates in meter and YP the free component concentration in moles per liter. *Compl.dat* contains the concentration profiles of the complexes, and *totconc.dat* the profiles for the total concentration of the components.

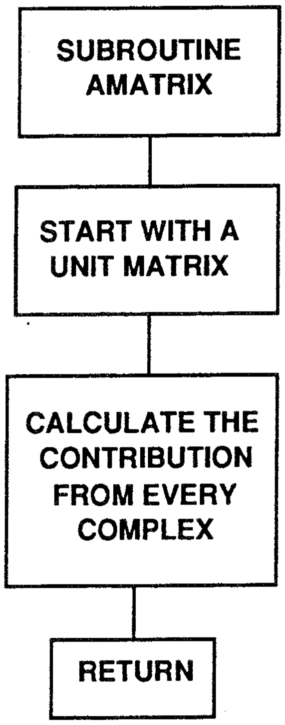
Testdata contains information about the cache system. The concentration profiles for every boundary are stored. If the boundary conditions are the same and the mineral composition also is the same the profiles are taken from the stored value instead of calculating the same profile again. Information about if the "old" profiles have been used or if new profiles had to be calculated is found in the file testdata.

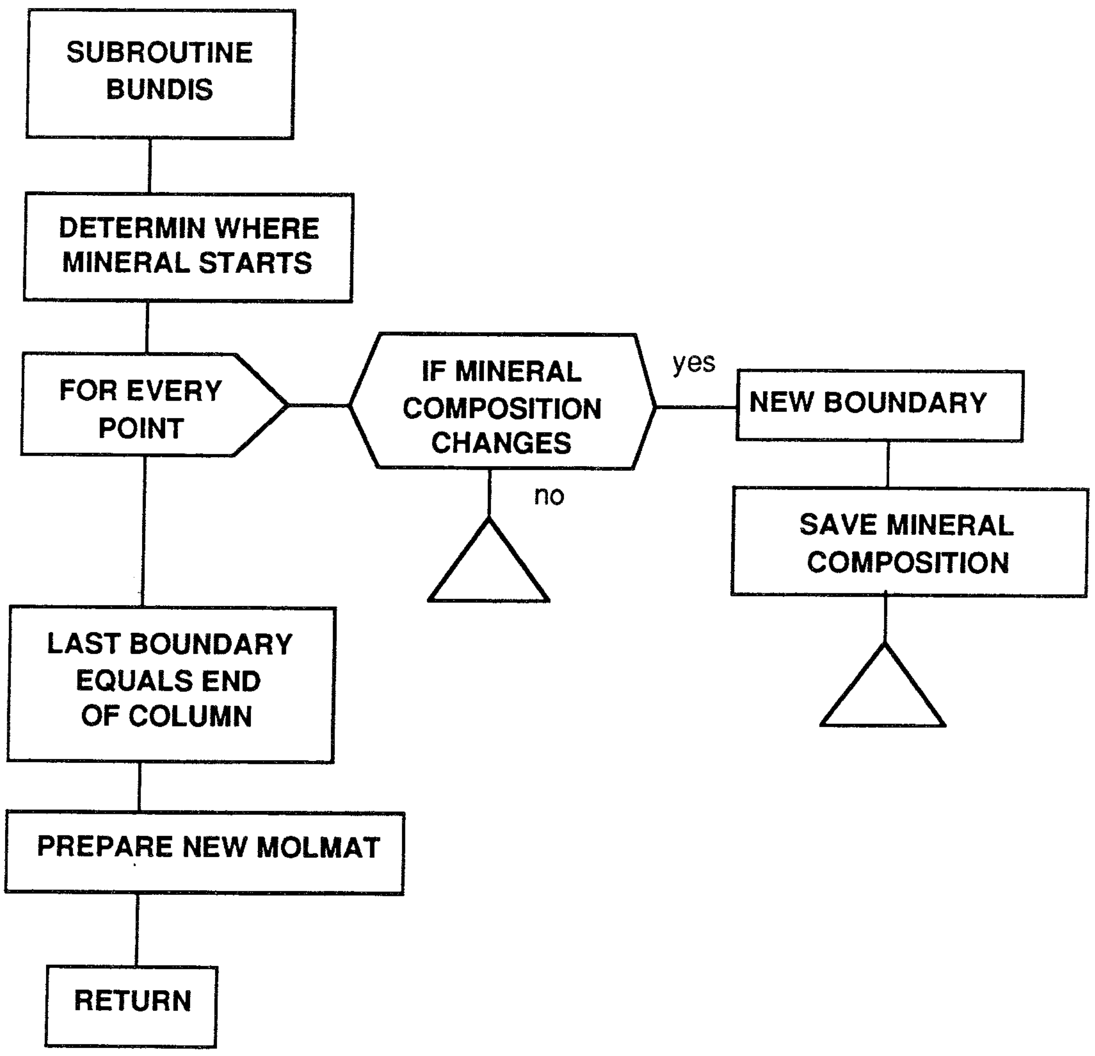
The file *frontrace* contains the dissolution front position, in meters, versus time, in years.

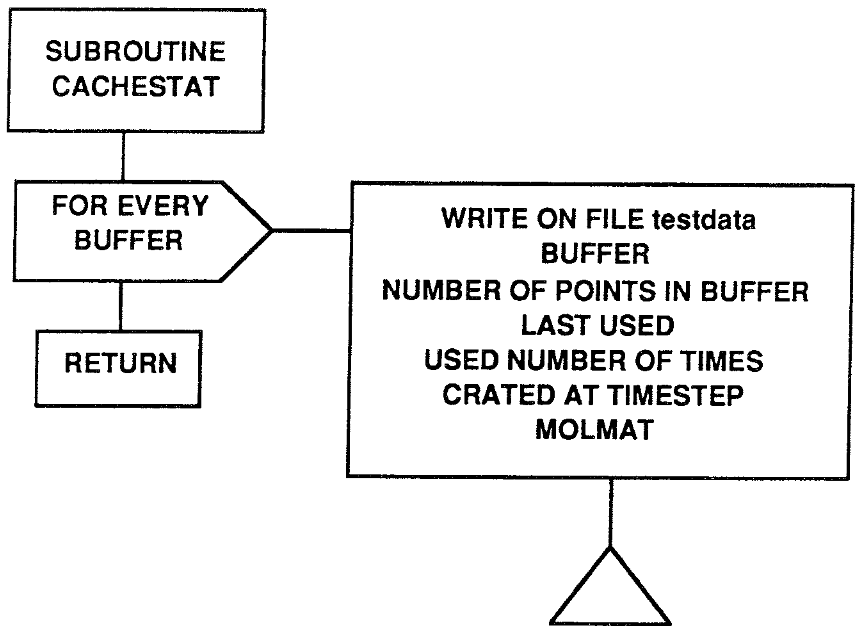
APPENDIX B
THE FLOW SHEETS

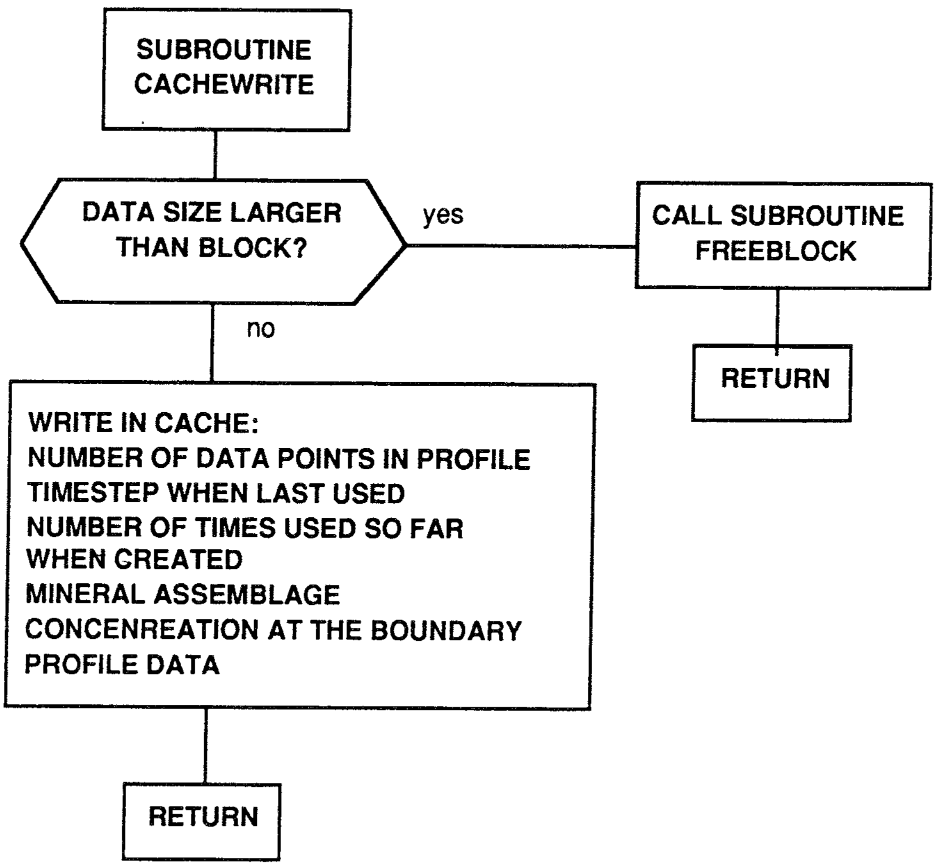

MAIN PROGRAM
CALL READINP
CALL PRINTINP
READ FILE ode.dat
WRITE CALCULATION PARAMETERS
STORE INITIAL CONCENTRATIONS
IF RESTART
yes
CALL STRESTORE
no
CALL BUNDIS
CALL FRONTRACK
FOR EVERY ITERATION
MINERAL LEFT
no
WRITE "END OF MINERAL"
STOP
yes
CALL RECALC
IF PROFILE ALREADY SAVED
no
CALL STATESAVE
yes
STOP
END
SUCCESSFUL RUN?
no
CALL STATESAVE
WRITE ERROR MESSAGE
STOP
yes
CALL TIMESTEP
LONG ENOUGH TIME HAS PAST?
yes
CALL FRONTRACK
WRITE FRONTRACE
no
IF TIME TO PRINTOUT
yes
CALL PRINTOUT
no
RESET INLET CONCENTRATIONS
IF TIME TO STORE PROFILE
yes
CALL STATESAVE
no

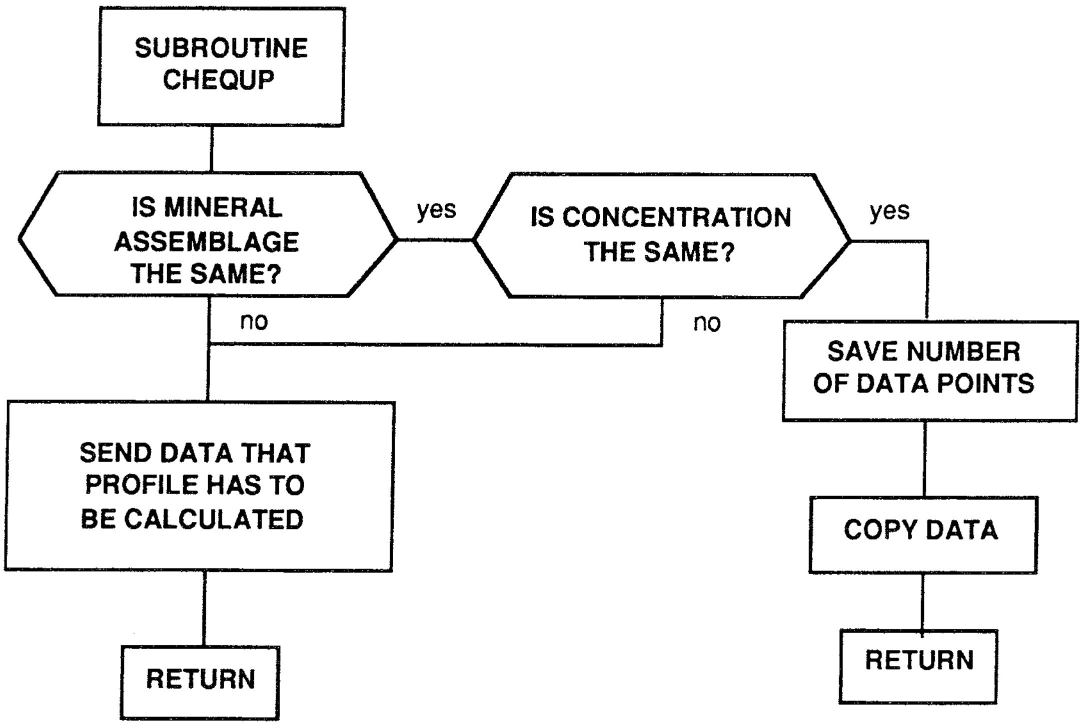

SUBROUTINE ACTCOEFF
FOR EVERY SPECIE
ACTIVITY COEFFICIENT =1
RETURN

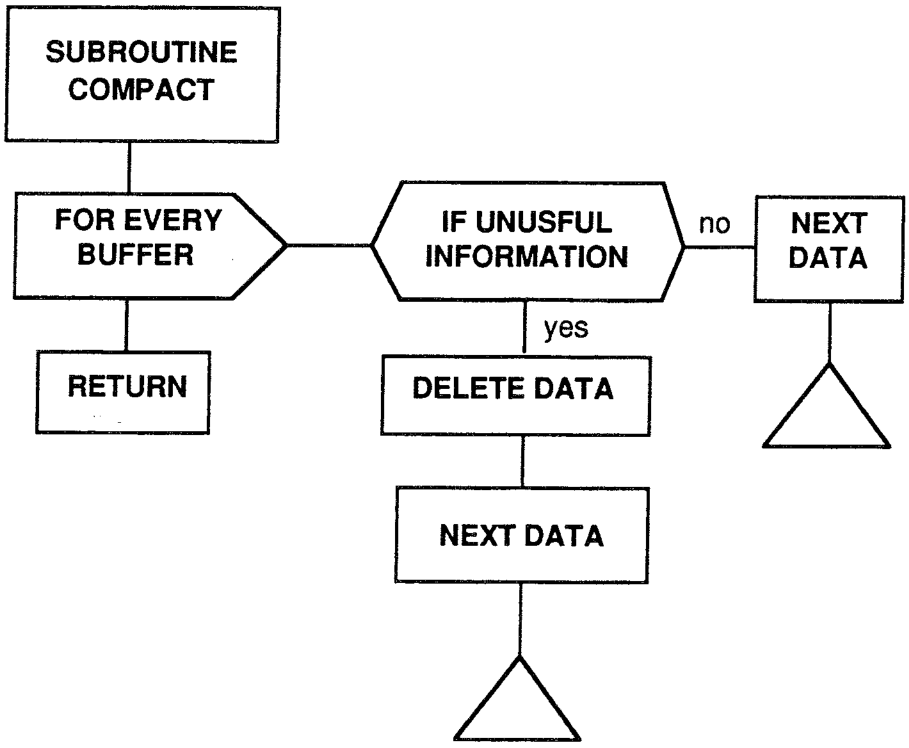

SUBROUTINE AMATRIX
START WITH A UNIT MATRIX
CALCULATE THE CONTRIBUTION FROM EVERY COMPLEX
RETURN

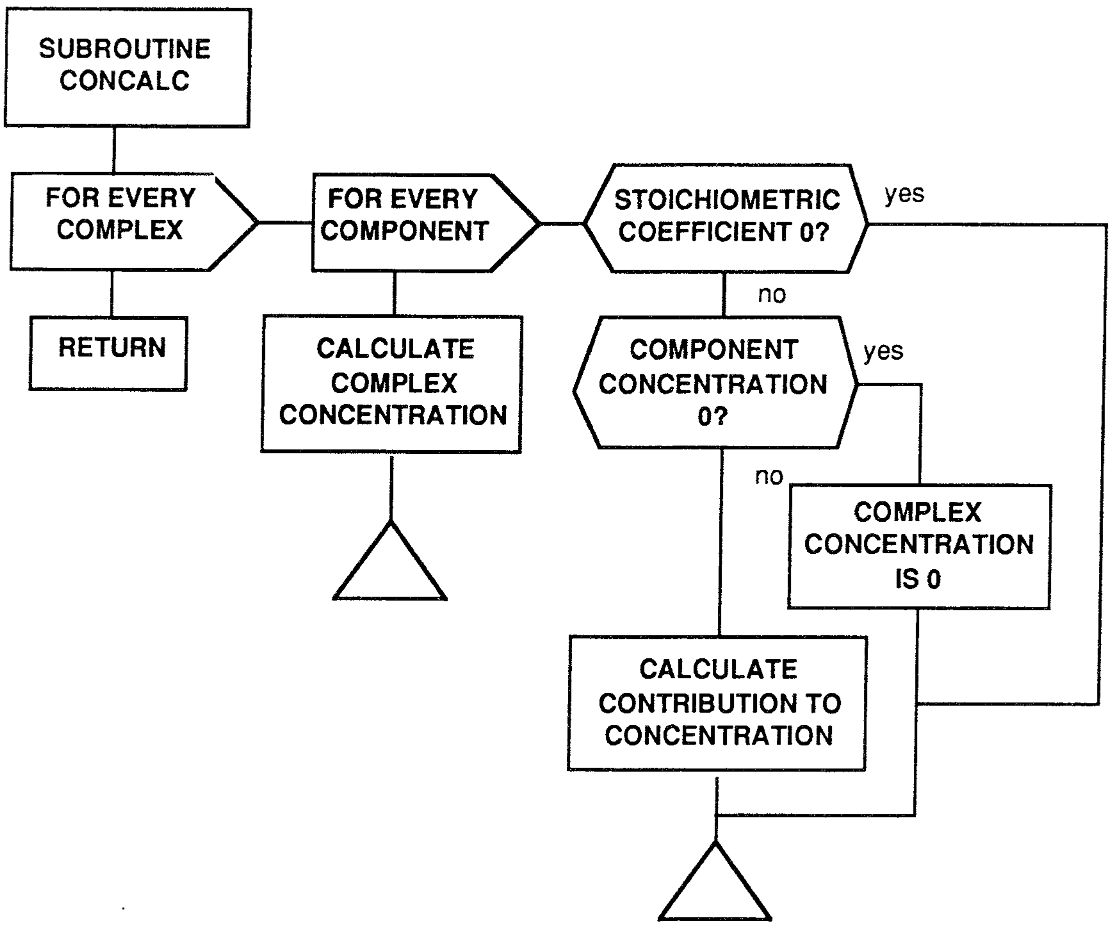

SUBROUTINE BUNDIS
DETERMIN WHERE MINERAL STARTS
FOR EVERY POINT
IF MINERAL COMPOSITION CHANGES
yes
NEW BOUNDARY
SAVE MINERAL COMPOSITION
no
LAST BOUNDARY EQUALS END OF COLUMN
PREPARE NEW MOLMAT
RETURN

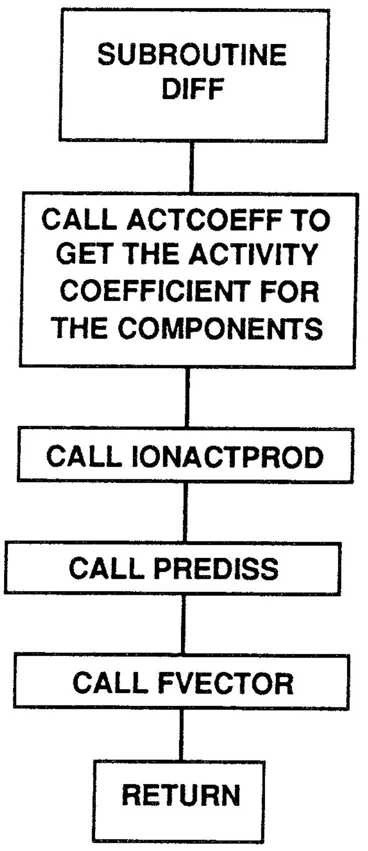

SUBROUTINE CACHESTAT
FOR EVERY BUFFER
RETURN
WRITE ON FILE testdata
BUFFER
NUMBER OF POINTS IN BUFFER
LAST USED
USED NUMBER OF TIMES
CRATED AT TIMESTEP
MOLMAT

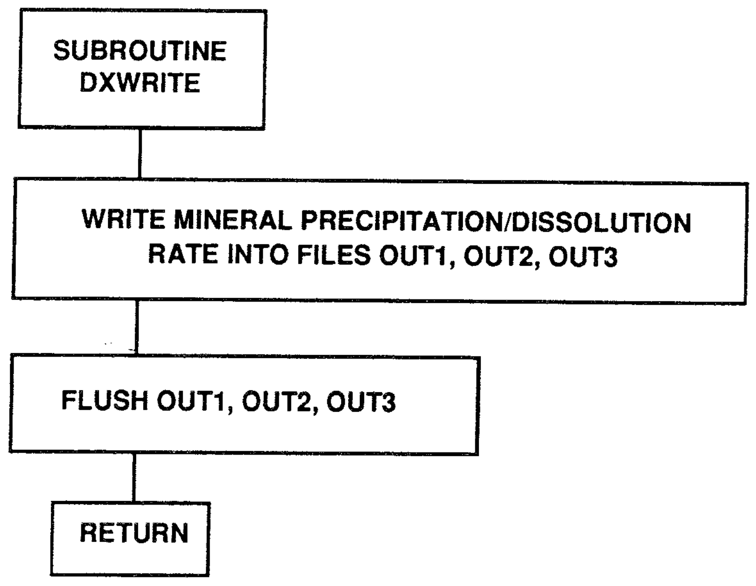

SUBROUTINE CACHEWRITE
DATA SIZE LARGER THAN BLOCK?
yes
CALL SUBROUTINE FREEBLOCK
RETURN
no
WRITE IN CACHE:
NUMBER OF DATA POINTS IN PROFILE
TIMESTEP WHEN LAST USED
NUMBER OF TIMES USED SO FAR
WHEN CREATED
MINERAL ASSEMBLAGE
CONCENREATION AT THE BOUNDARY
PROFILE DATA
RETURN

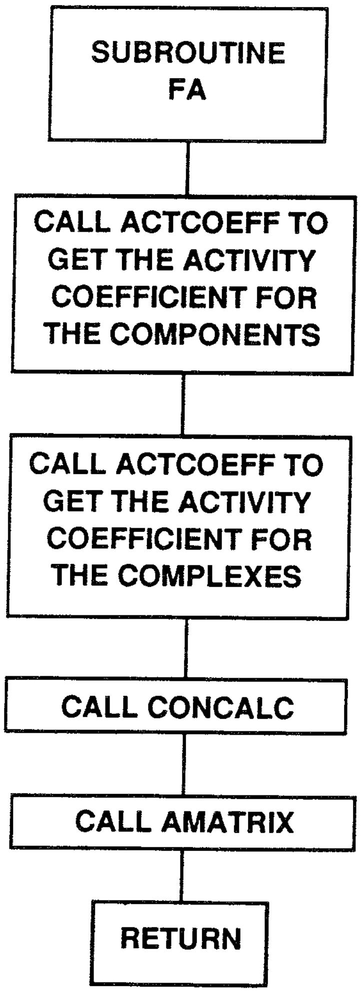

SUBROUTINE CHEQUP
IS MINERAL ASSEMBLAGE THE SAME?
yes
IS CONCENTRATION THE SAME?
yes
no
no
SAVE NUMBER OF DATA POINTS
SEND DATA THAT PROFILE HAS TO BE CALCULATED
COPY DATA
RETURN
RETURN

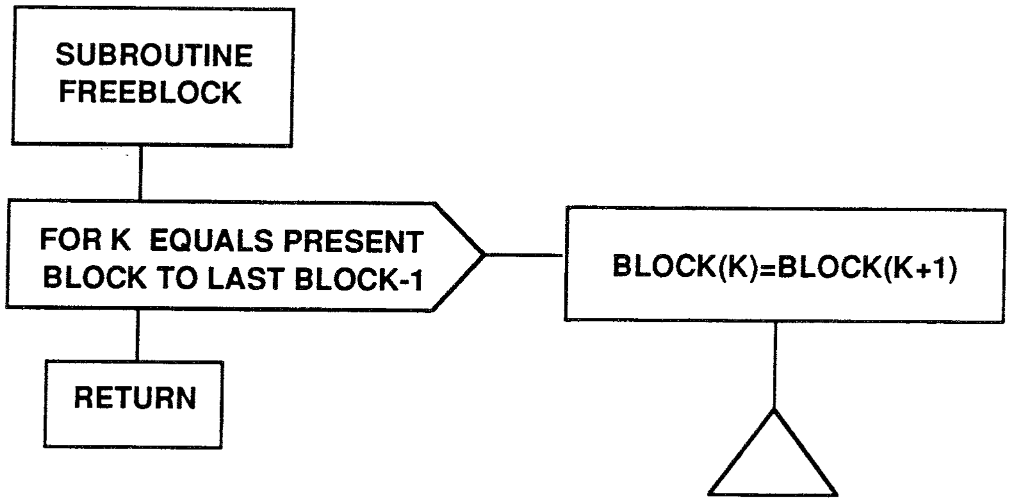

SUBROUTINE COMPACT
FOR EVERY BUFFER
IF UNUSFUL INFORMATION
no
NEXT DATA
yes
RETURN
DELETE DATA
NEXT DATA

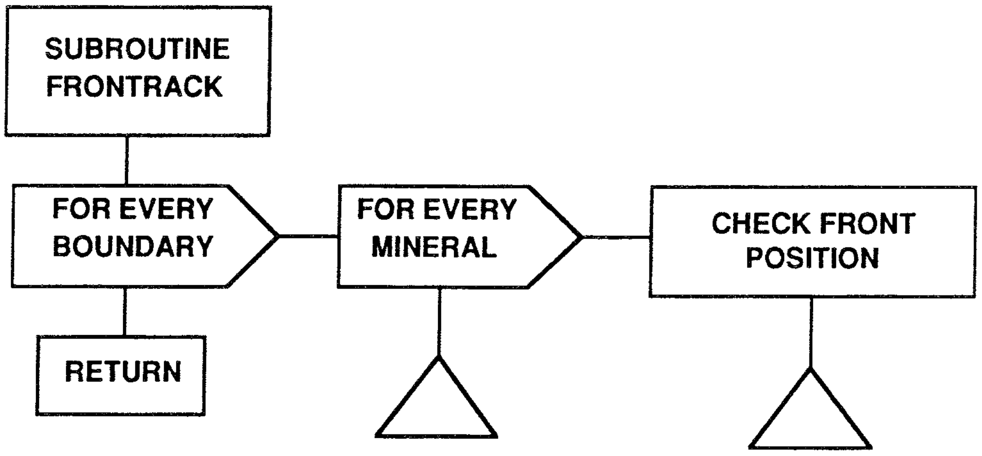

SUBROUTINE CONCALC
FOR EVERY COMPLEX
RETURN
FOR EVERY COMPONENT
CALCULATE COMPLEX CONCENTRATION
STOICHIOMETRIC COEFFICIENT 0?
yes
no
COMPONENT CONCENTRATION 0?
yes
no
COMPLEX CONCENTRATION IS 0
CALCULATE CONTRIBUTION TO CONCENTRATION

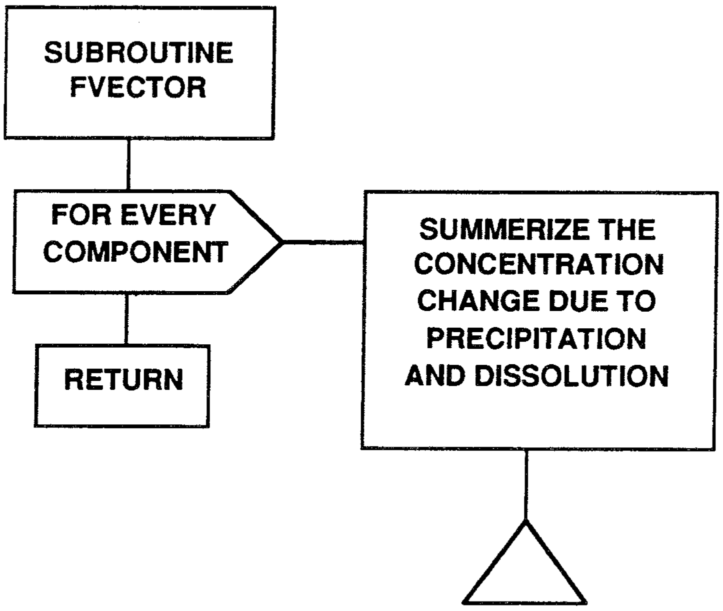

SUBROUTINE DIFF
CALL ACTCOEFF TO GET THE ACTIVITY COEFFICIENT FOR THE COMPONENTS
CALL IONACTPROD
CALL PREDISS
CALL FVECTOR
RETURN

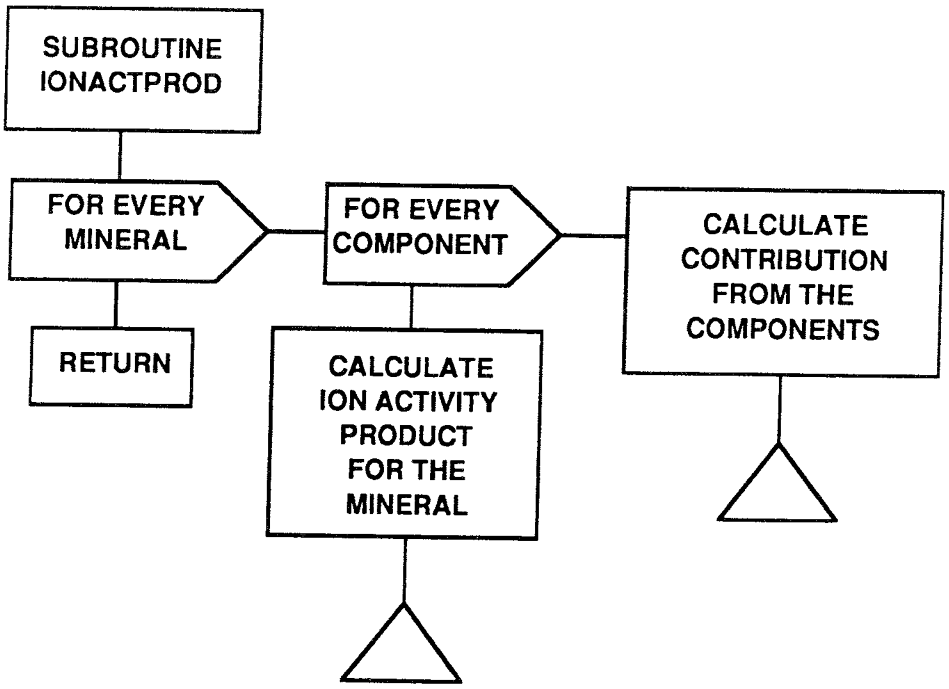

SUBROUTINE DXWRITE
WRITE MINERAL PRECIPITATION/DISSOLUTION RATE INTO FILES OUT1, OUT2, OUT3
FLUSH OUT1, OUT2, OUT3
RETURN

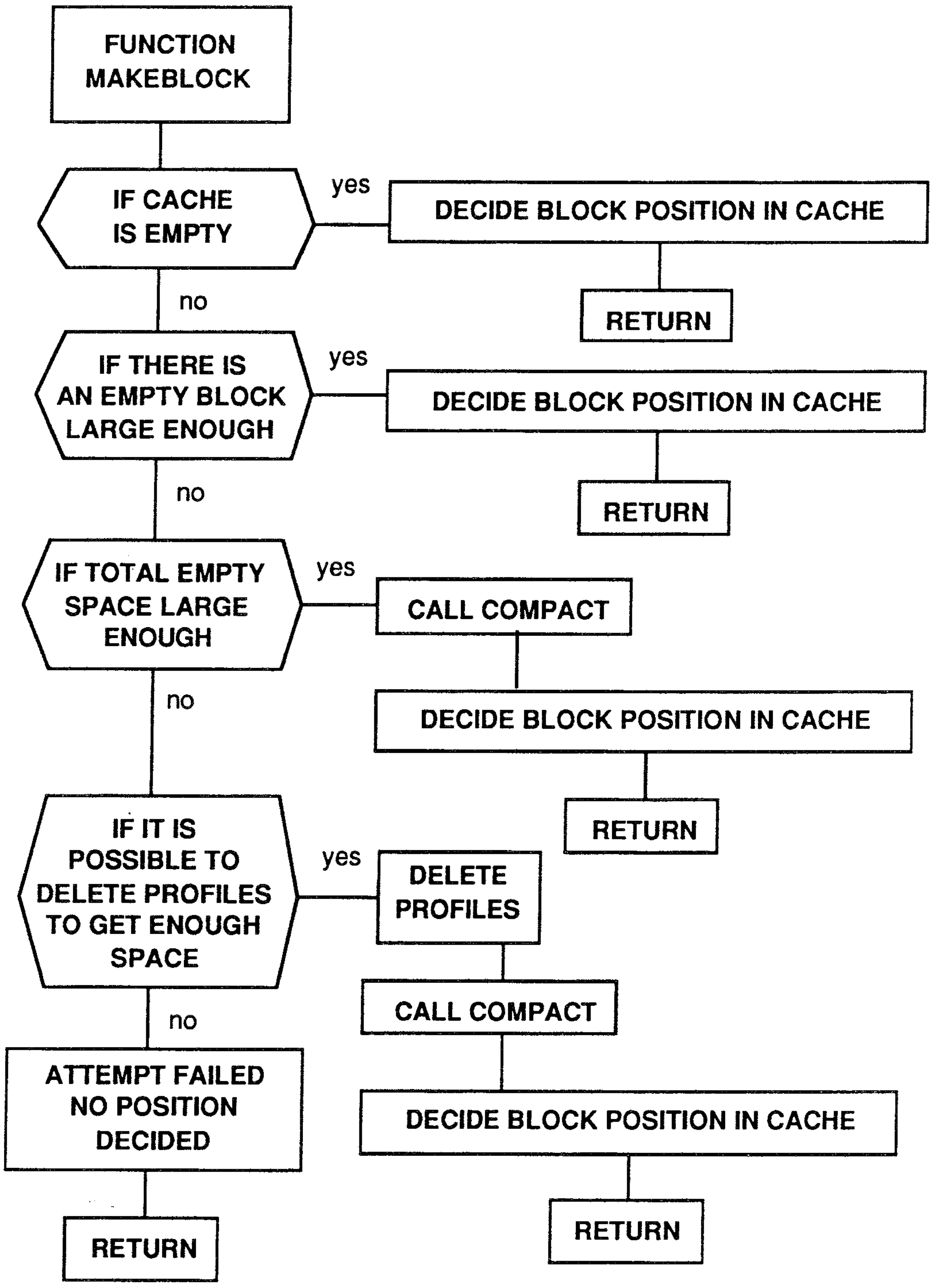

SUBROUTINE FA
CALL ACTCOEFF TO GET THE ACTIVITY COEFFICIENT FOR THE COMPONENTS
CALL ACTCOEFF TO GET THE ACTIVITY COEFFICIENT FOR THE COMPLEXES
CALL CONCALC
CALL AMATRIX
RETURN

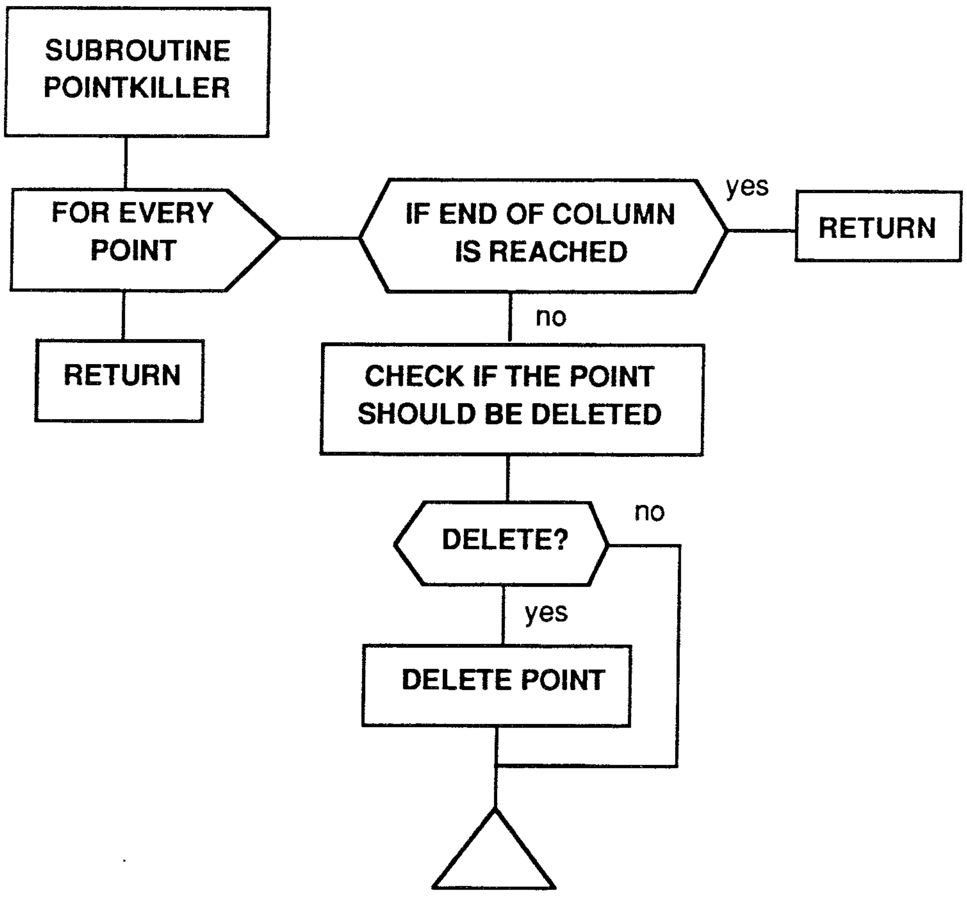

SUBROUTINE FREEBLOCK
FOR K EQUALS PRESENT BLOCK TO LAST BLOCK-1
BLOCK(K)=BLOCK(K+1)
RETURN

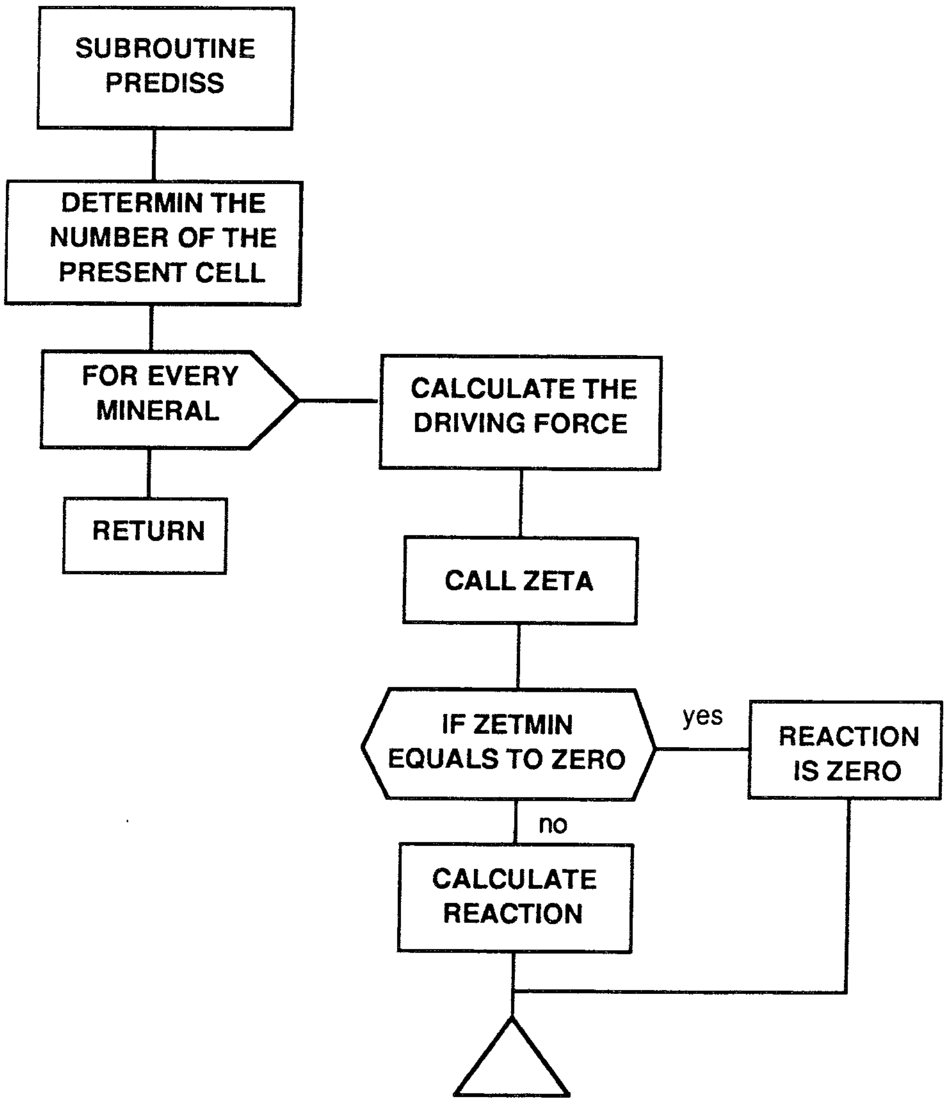

SUBROUTINE FRONTRACK
FOR EVERY BOUNDARY
FOR EVERY MINERAL
CHECK FRONT POSITION
RETURN

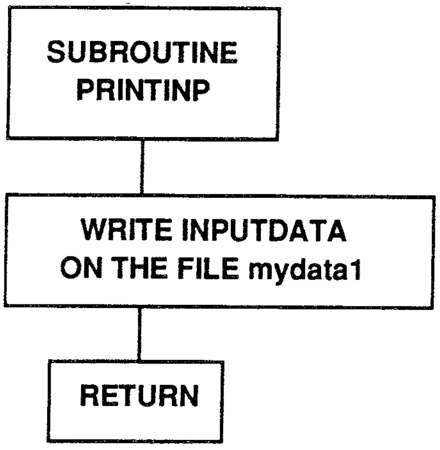

SUBROUTINE FVECTOR
FOR EVERY COMPONENT
SUMMERIZE THE CONCENTRATION CHANGE DUE TO PRECIPITATION AND DISSOLUTION
RETURN

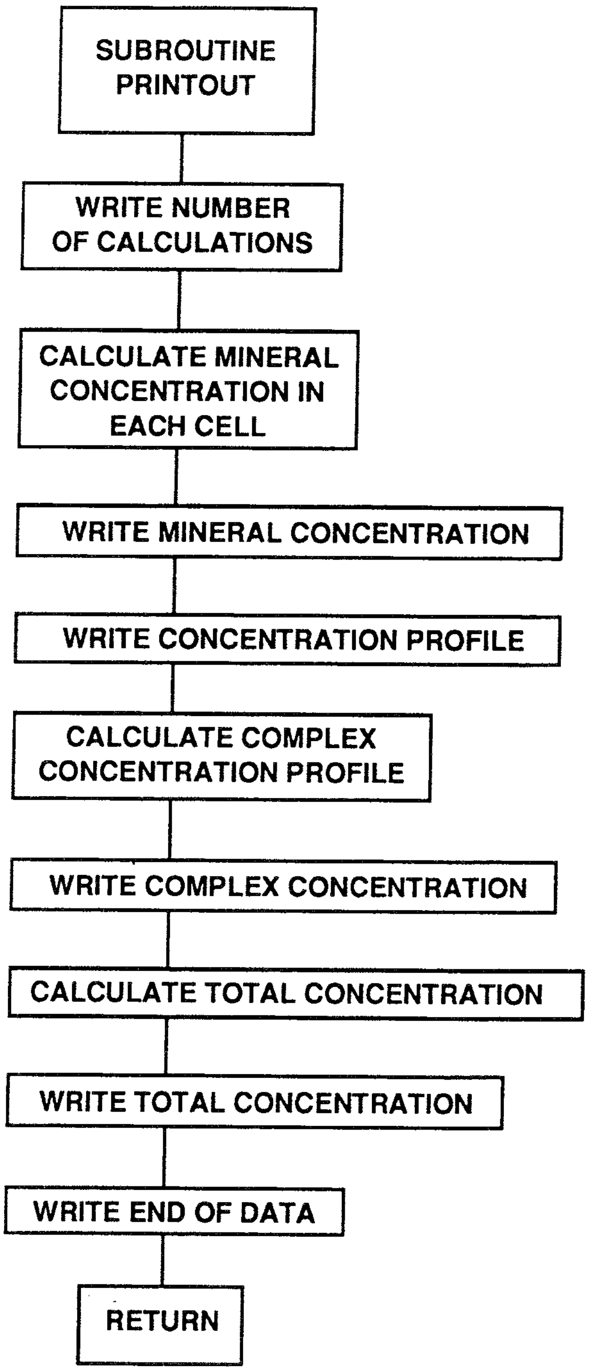

SUBROUTINE IONACTPROD
FOR EVERY MINERAL
RETURN
FOR EVERY COMPONENT
CALCULATE ION ACTIVITY PRODUCT FOR THE MINERAL
CALCULATE CONTRIBUTION FROM THE COMPONENTS

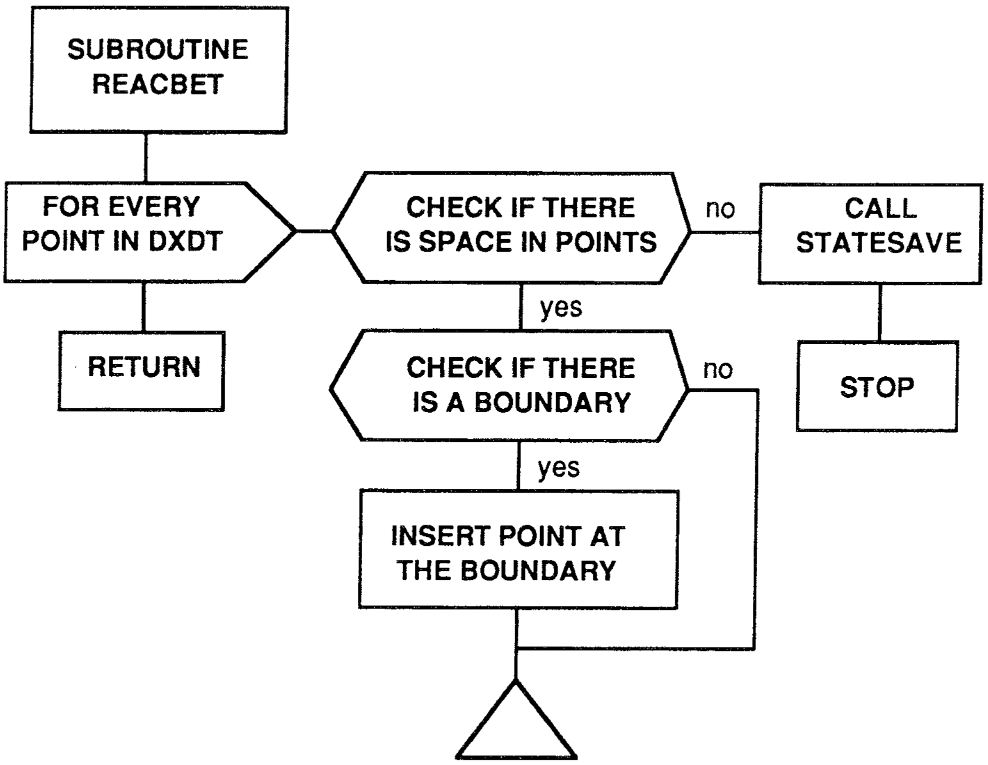

FUNCTION MAKEBLOCK
IF CACHE IS EMPTY
yes
DECIDE BLOCK POSITION IN CACHE
RETURN
no
IF THERE IS AN EMPTY BLOCK LARGE ENOUGH
yes
DECIDE BLOCK POSITION IN CACHE
RETURN
no
IF TOTAL EMPTY SPACE LARGE ENOUGH
yes
CALL COMPACT
DECIDE BLOCK POSITION IN CACHE
RETURN
no
IF IT IS POSSIBLE TO DELETE PROFILES TO GET ENOUGH SPACE
yes
DELETE PROFILES
CALL COMPACT
DECIDE BLOCK POSITION IN CACHE
RETURN
no
ATTEMPT FAILED NO POSITION DECIDED
RETURN

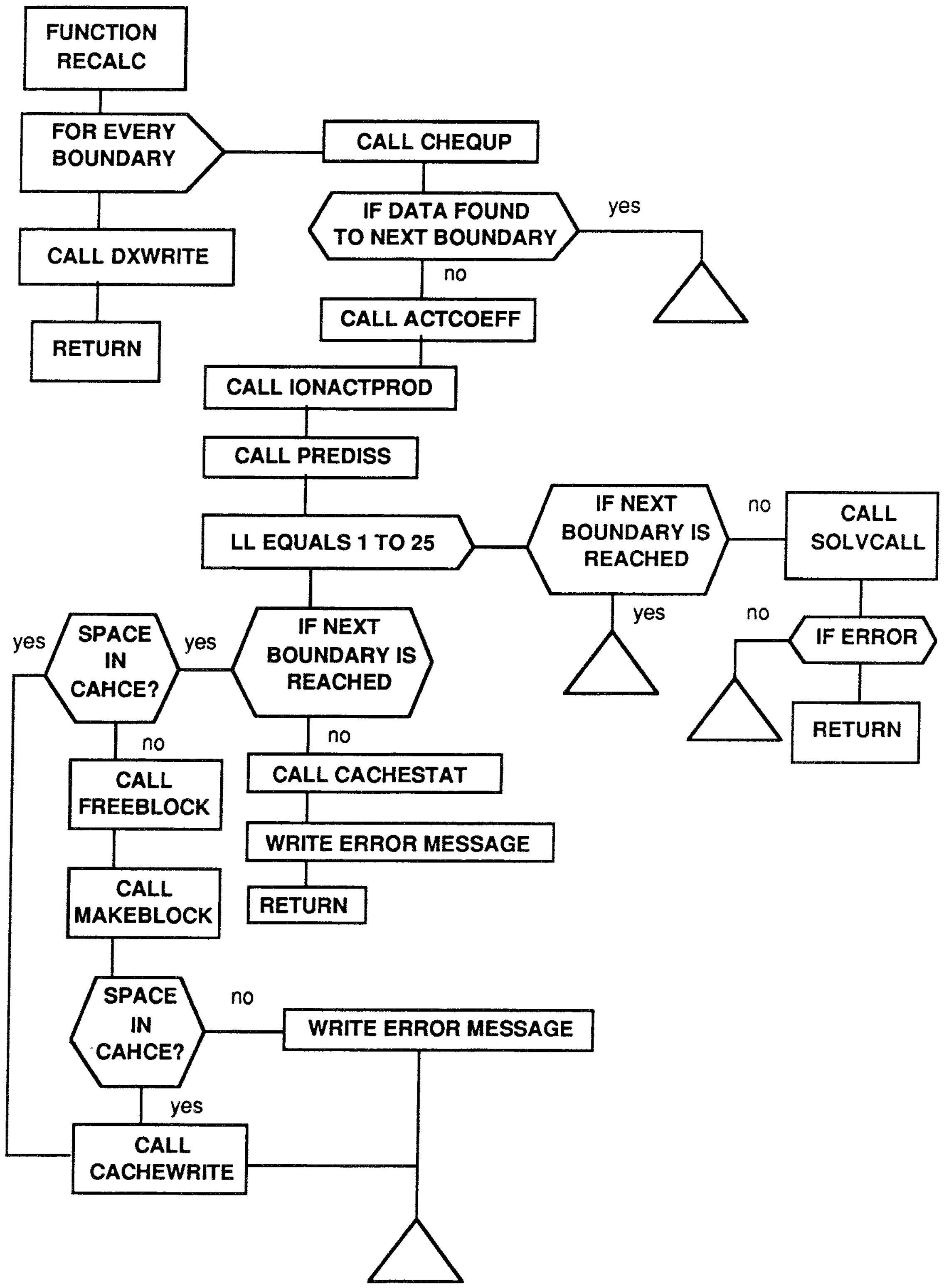

SUBROUTINE POINTKILLER
FOR EVERY POINT
RETURN
IF END OF COLUMN IS REACHED
yes
RETURN
no
CHECK IF THE POINT SHOULD BE DELETED
DELETE?
no
yes
DELETE POINT

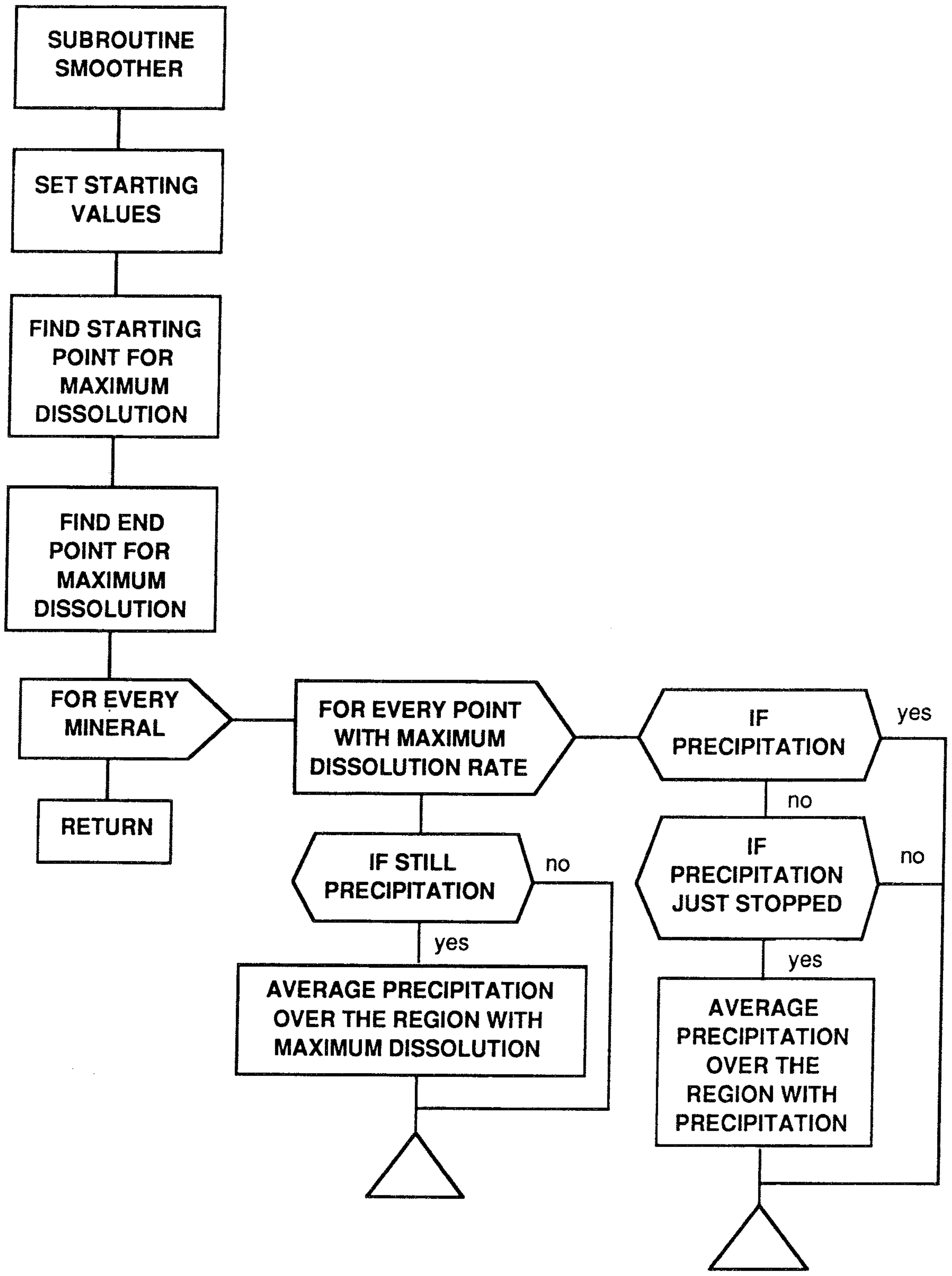

SUBROUTINE PREDISS
DETERMIN THE NUMBER OF THE PRESENT CELL
FOR EVERY MINERAL
RETURN
CALCULATE THE DRIVING FORCE
CALL ZETA
IF ZETMIN EQUALS TO ZERO
yes
REACTION IS ZERO
no
CALCULATE REACTION

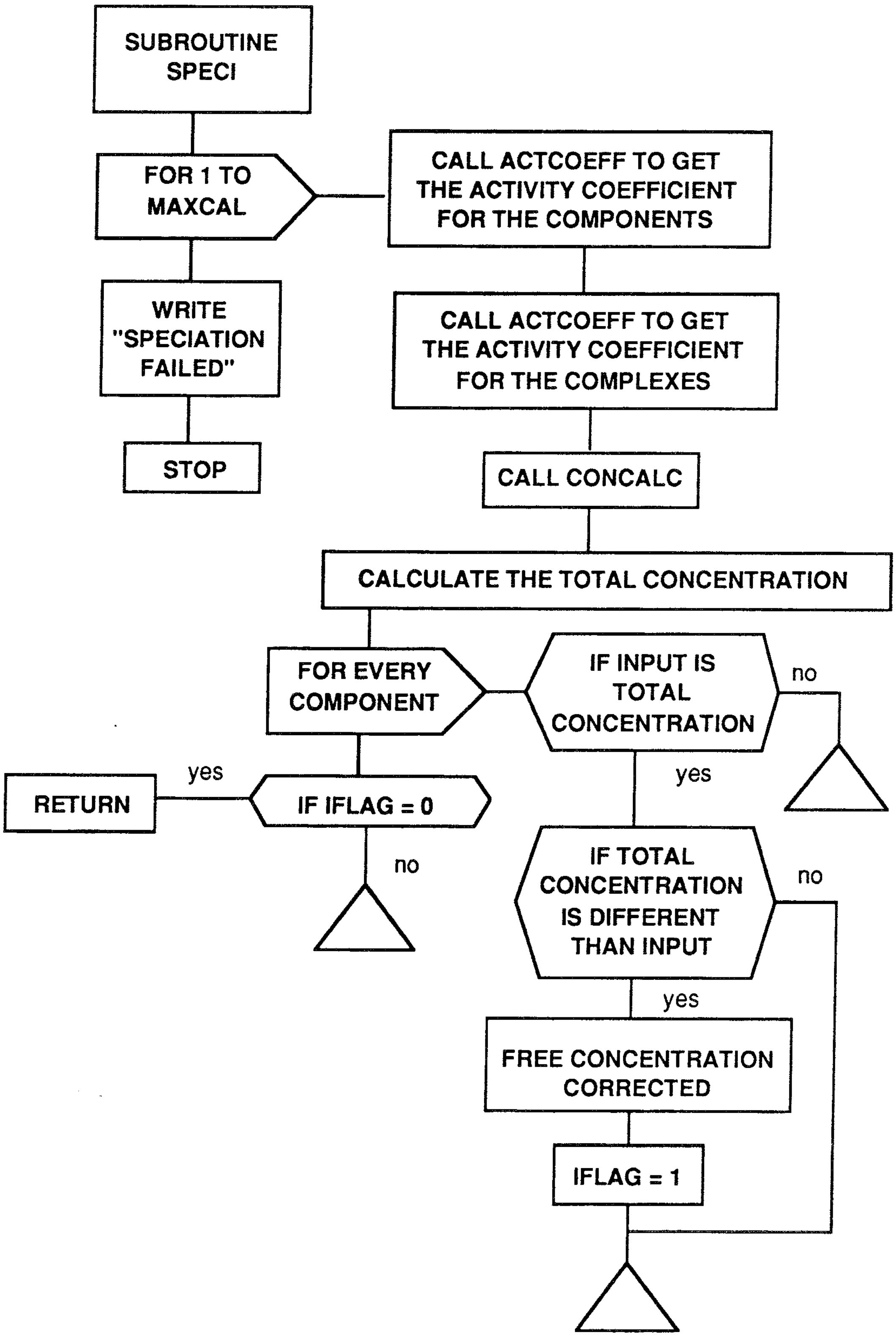

SUBROUTINE PRINTINP
WRITE INPUTDATA ON THE FILE mydata1
RETURN

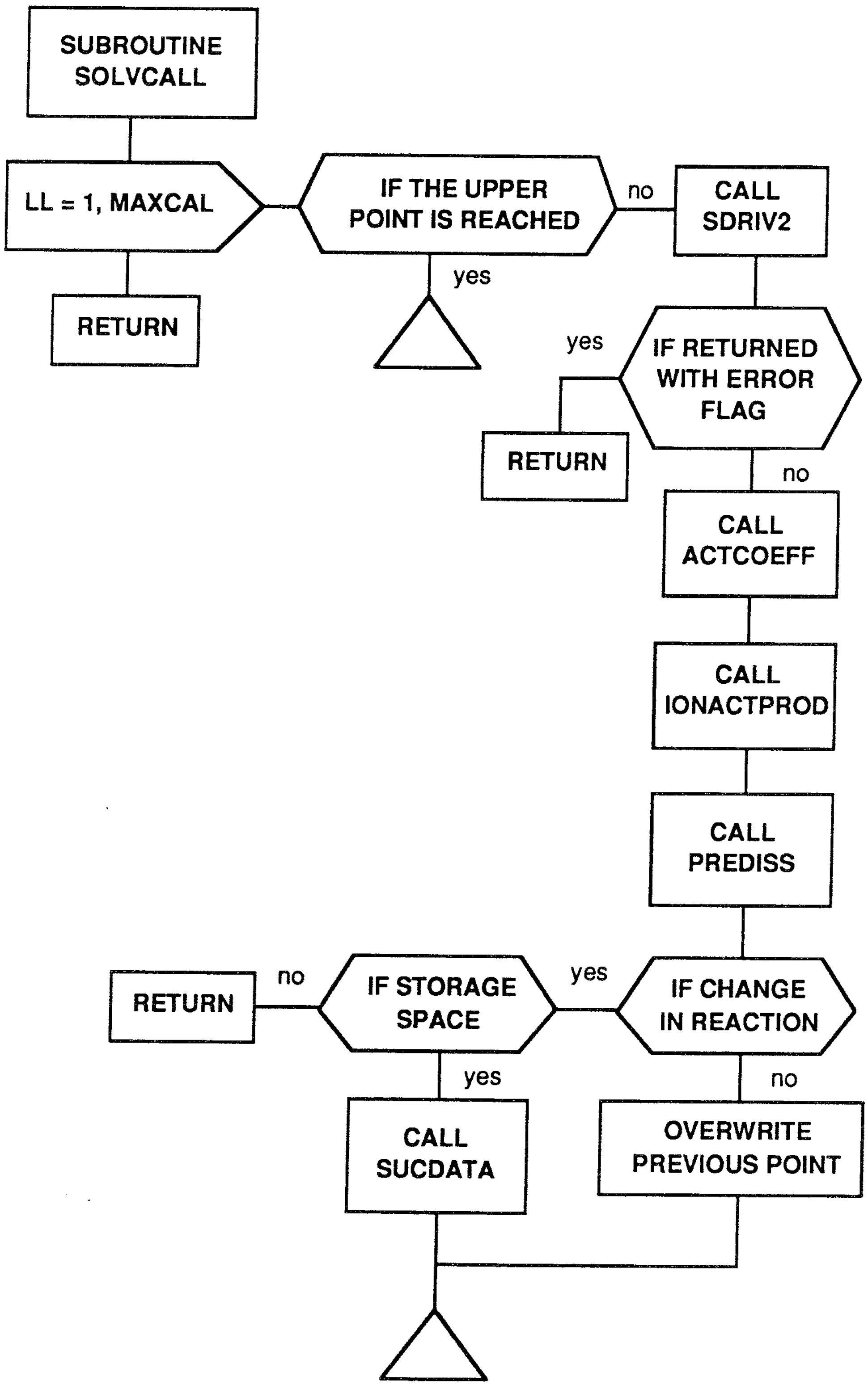

SUBROUTINE PRINTOUT
WRITE NUMBER OF CALCULATIONS
CALCULATE MINERAL CONCENTRATION IN EACH CELL
WRITE MINERAL CONCENTRATION
WRITE CONCENTRATION PROFILE
CALCULATE COMPLEX CONCENTRATION PROFILE
WRITE COMPLEX CONCENTRATION
CALCULATE TOTAL CONCENTRATION
WRITE TOTAL CONCENTRATION
WRITE END OF DATA
RETURN

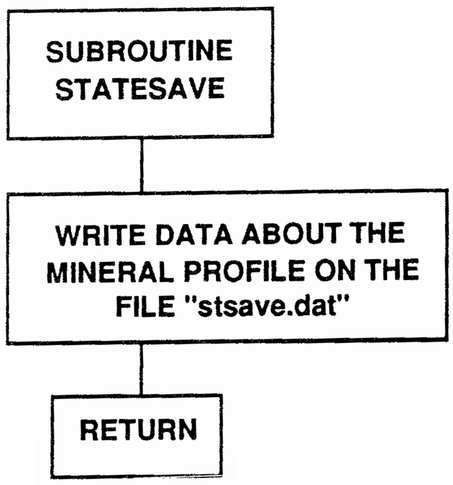

SUBROUTINE REACBET
FOR EVERY POINT IN DXDT
RETURN
CHECK IF THERE IS SPACE IN POINTS
no
CALL STATESAVE
STOP
yes
CHECK IF THERE IS A BOUNDARY
no
yes
INSERT POINT AT THE BOUNDARY

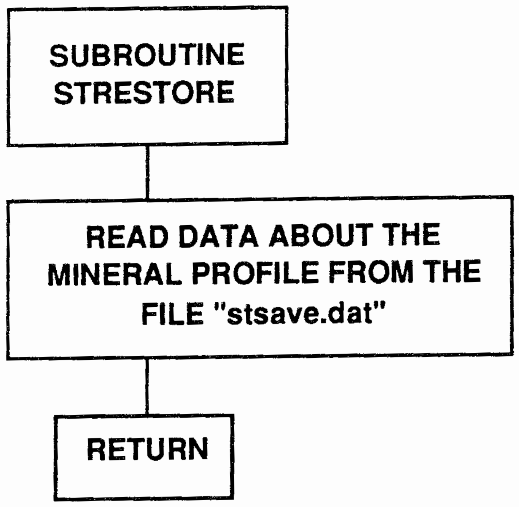

FUNCTION RECALC
FOR EVERY BOUNDARY
CALL CHEQUP
IF DATA FOUND TO NEXT BOUNDARY
yes
no
CALL DXWRITE
RETURN
CALL ACTCOEFF
CALL IONACTPROD
CALL PREDISS
LL EQUALS 1 TO 25
IF NEXT BOUNDARY IS REACHED
no
CALL SOLVCALL
yes
no
IF ERROR
RETURN
yes
SPACE IN CAHCE?
yes
IF NEXT BOUNDARY IS REACHED
no
no
CALL FREEBLOCK
CALL CACHESTAT
WRITE ERROR MESSAGE
RETURN
CALL MAKEBLOCK
SPACE IN CAHCE?
no
WRITE ERROR MESSAGE
yes
CALL CACHEWRITE

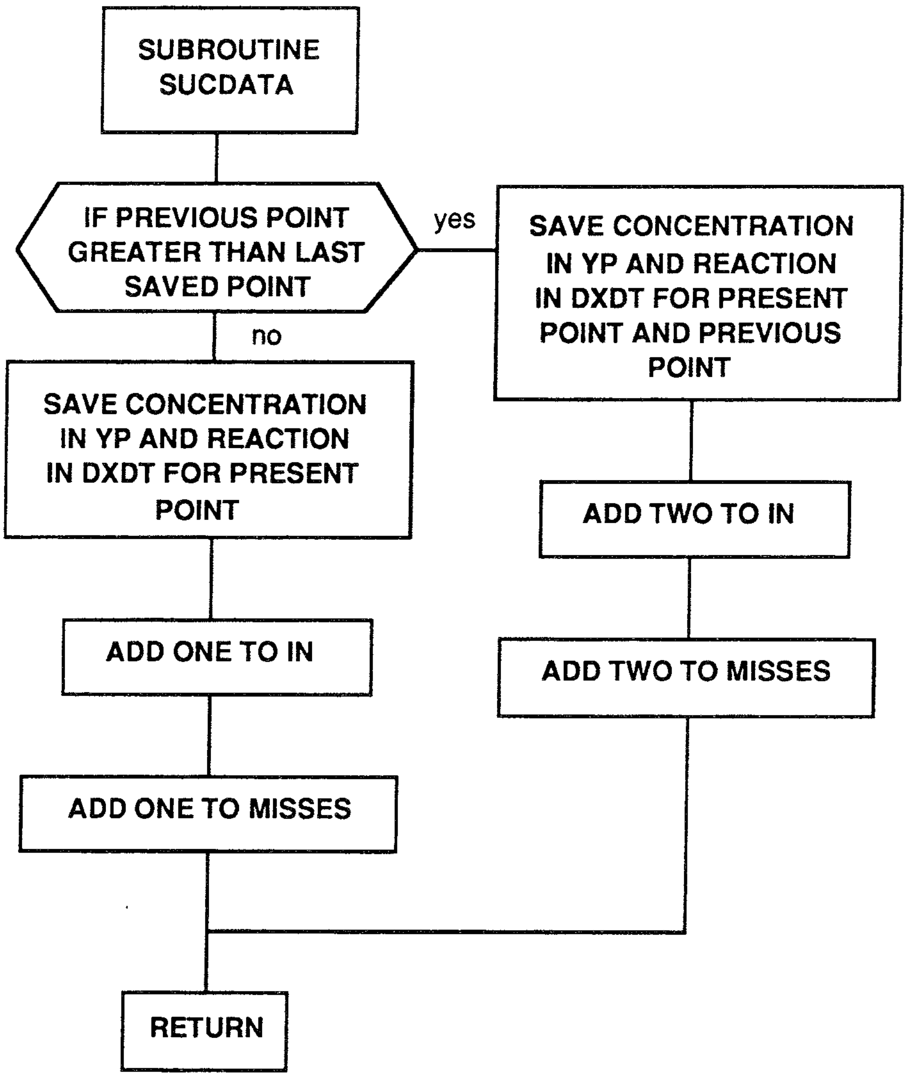

SUBROUTINE SMOOTHER
SET STARTING VALUES
FIND STARTING POINT FOR MAXIMUM DISSOLUTION
FIND END POINT FOR MAXIMUM DISSOLUTION
FOR EVERY MINERAL
RETURN
FOR EVERY POINT WITH MAXIMUM DISSOLUTION RATE
IF STILL PRECIPITATION
no
yes
AVERAGE PRECIPITATION OVER THE REGION WITH MAXIMUM DISSOLUTION
IF PRECIPITATION
yes
no
IF PRECIPITATION JUST STOPPED
no
yes
AVERAGE PRECIPITATION OVER THE REGION WITH PRECIPITATION

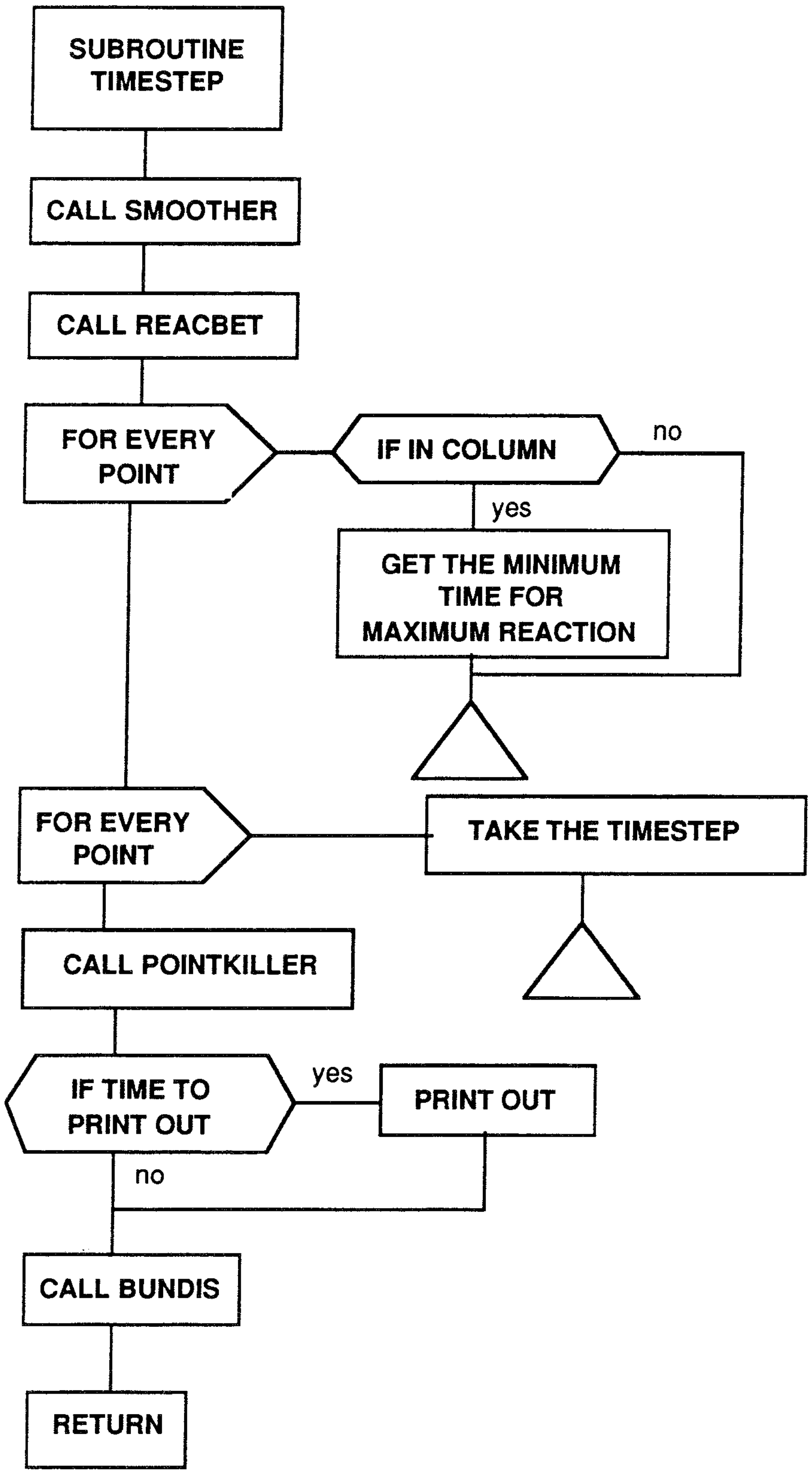

SUBROUTINE SPECI
FOR 1 TO MAXCAL
CALL ACTCOEFF TO GET THE ACTIVITY COEFFICIENT FOR THE COMPONENTS
CALL ACTCOEFF TO GET THE ACTIVITY COEFFICIENT FOR THE COMPLEXES
CALL CONCALC
CALCULATE THE TOTAL CONCENTRATION
WRITE "SPECIATION FAILED"
STOP
FOR EVERY COMPONENT
IF INPUT IS TOTAL CONCENTRATION
no
yes
RETURN
yes
IF IFLAG = 0
no
IF TOTAL CONCENTRATION IS DIFFERENT THAN INPUT
no
yes
FREE CONCENTRATION CORRECTED
IFLAG = 1

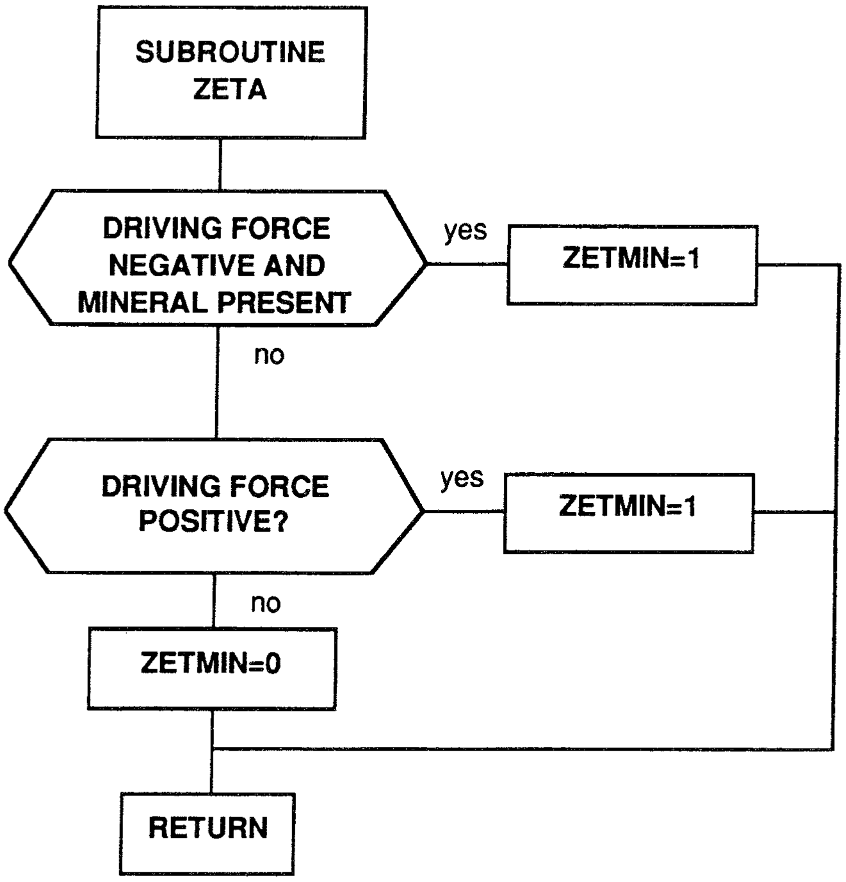

SUBROUTINE SOLVCALL
LL = 1, MAXCAL
RETURN
IF THE UPPER POINT IS REACHED
yes
no
CALL SDRIV2
IF RETURNED WITH ERROR FLAG
yes
RETURN
no
CALL ACTCOEFF
CALL IONACTPROD
CALL PREDISS
IF CHANGE IN REACTION
yes
no
OVERWRITE PREVIOUS POINT
IF STORAGE SPACE
no
RETURN
yes
CALL SUCDATA


SUBROUTINE STATESAVE
WRITE DATA ABOUT THE MINERAL PROFILE ON THE FILE "stsave.dat"
RETURN


SUBROUTINE STRESTORE
READ DATA ABOUT THE MINERAL PROFILE FROM THE FILE "stsave.dat"
RETURN


SUBROUTINE SUCDATA
IF PREVIOUS POINT GREATER THAN LAST SAVED POINT
yes
SAVE CONCENTRATION IN YP AND REACTION IN DXDT FOR PRESENT POINT AND PREVIOUS POINT
no
SAVE CONCENTRATION IN YP AND REACTION IN DXDT FOR PRESENT POINT
ADD TWO TO IN
ADD ONE TO IN
ADD TWO TO MISSES
ADD ONE TO MISSES
RETURN


SUBROUTINE TIMESTEP
CALL SMOOTHER
CALL REACBET
FOR EVERY POINT
IF IN COLUMN
no
yes
GET THE MINIMUM TIME FOR MAXIMUM REACTION
FOR EVERY POINT
TAKE THE TIMESTEP
CALL POINTKILLER
IF TIME TO PRINT OUT
yes
PRINT OUT
no
CALL BUNDIS
RETURN


SUBROUTINE ZETA
DRIVING FORCE NEGATIVE AND MINERAL PRESENT
yes
ZETMIN=1
no
DRIVING FORCE POSITIVE?
yes
ZETMIN=1
no
ZETMIN=0
RETURN

Appendix C
Source Code

```
C*************************************************************
C                    P R O G R A M   C H E M F R O N T S
C*************************************************************
C
C      CDIFF  = MAXIMUM RELATIVE DIFFERENCE FOR THE CONCENTRATIONS
C      COLUMN = LENGTH OF THE COLUMN    m
C      COMCON = CONCENTRATION OF COMPLEXES   mol/dm3
C      COMP   = NAME OF COMPONENTS   -
C      COMPON = NUMBER OF COMPONENTS   -
C      CONC   = FREE CONCENTRATION OF COMPONENTS   mol/dm3
C      DOFF   = MINIMUM DISSOLUTION AND PRECIPITATION RATE
C      DXCELL = LENGTH OF A CELL
C      DXDT   = MINERAL CHANGE AT THE CELL BOUNDARIES   mol/year
C      EPS    = MAXIMUM RELATIVE ERROR   -
C      ETIME  = LAST TIME FRONT POSITION WAS SAVED
C      EWT    = SMALLEST NUMBER IN THE CALCULATIONS   -
C      F      = LOOPING VARIABLE FOR THE TIMESTEPS   -
C      FTRACK = VECTOR WITH FRONT POSITION IN TIME
C      HMAX   = MAXIMUM STEPSIZE
C      I      = LOOPING VARIABLE FOR THE COMPLEXES   -
C      IN     = NUMBER OF VALUES STORED IN XP AND YP   -
C      IPR    = HOW OFTEN RESULTS SHALL BE PRINTED
C      ISTATE = IF THE CALCULATION WAS SUCCESSFUL
C      IWHERE =
C      J      = LOOPING VARIABLE FOR THE COMPONENTS   -
C      K      = LOOPING VARIABLE
C      L      = LOOPING VARIABLE FOR THE CELLS   -
C      M      = LOOPING VARIABLE FOR THE MINERALS   -
C      MAXCEL = MAXIMUM NUMBER OF CELLS   -
C      MAXCOM = MAXIMUM NUMBER OF COMPONENTS   -
C      MAXMIN = MAXIMUM NUMBER OF MINERALS   -
C      MAXP   = MAXIMUM NUMBER OF POINTS   -
C      MAXPLX = MAXIMUM NUMBER OF COMPLEXES   -
C      MILLE  = MAXIMUM NUMBER OF POINTS IN POINTS AND XPOINT
C      MINE   = NUMBER OF MINERALS   -
C      MOLMAT = MOLAR AMOUNT OF MINERAL IN EACH CELL   mol
C      MXITER = MAXIMUM NUMBER OF ITERATION   -
C      NAME   = NAME OF COMPLEXES   -
C      NBOUND = NUMBER OF BOUNDARIES   -
C      NCALL  = NUMBER OF CALLS TO RECALC
C      NFRONT = NUMBER OF FRONTS   -
C      NO     = NUMBER OF COMPLEXES   -
C      RECALC = FUNCTION TO CALCULATE THE CONCENTRATION PROFILE
C      SCOMCO = START CONCENTRATION FOR COMPLEXES   mol/dm3
C      SCONC  = START CONCENTRATION FOR COMPONENTS   mol/dm3
C      STOCOM = STOICHIOMETRIC MATRIX FOR COMPLEXES   -
C      STREST = IF RECALCULATION
C      STSAVE = IF PROFILE SHALL BE SAVED OR NOT
C      TIMA   = MAXIMUM TIME FOR A TIME STEP
C      TIME   = TIME DURING SIMULATION   years
C      TIMEMI = MINIMUM TIME FOR STEP SIZE   years
C      XP     = X COORDINATE CHOSEN BY THE PROGRAM   m
C      XSTEP  = MAXIMUM DISTANCE BETWEEN SAVED POINTS
C      YP     = CONCENTRATION OF COMPONENTS IN XP   mol/dm3
C
```

```
      IMPLICIT           NONE
      INTEGER            COMPON, F, I, IN, IPR, ISTATE, IWHERE, J, K, L, M
      INTEGER            MAXCEL, MAXCOM, MAXMIN, MAXP, MAXPLX, MILLE, MINE
      INTEGER            MXITER, NBOUND, NCALL, NFRONT, NO, RECALC, STSAVE
      INTEGER            STREST
      PARAMETER          (MAXCOM=25, MAXMIN=25, MAXP=10000, MAXPLX=100,
     +                   MAXCEL=MAXMIN*500, MILLE=20000, MXITER=5000)
      CHARACTER          COMP(MAXCOM)*10, NAME(MAXPLX)*10
      REAL               STOCOM(MAXPLX,MAXCOM)
      DOUBLE PRECISION CDIFF, COLUMN, COMCON(MAXPLX), CONC(MAXCOM), DOFF
      DOUBLE PRECISION DXCELL, DXDT(MAXP,MAXMIN), EPS, ETIME, EWT
      DOUBLE PRECISION FTRACK(1024), HMAX, MOLMAT(MAXCEL,MAXMIN)
      DOUBLE PRECISION SCOMCO(MAXPLX), SCONC(MAXCOM), TIMA, TIME, TIMEMI
      DOUBLE PRECISION XP(MAXP), XSTEP, YP(MAXP,MAXCOM)

      COMMON/ONE/COMPON
      COMMON/TWO/DXCELL,MINE
C     COMMON/THREE/CELLS,AREA
C     COMMON/FOUR/FLUX,STOMIN
      COMMON/FIVE/NO,STOCOM
C     COMMON/SIX/POROS,WV,MINVOL
      COMMON/SEVEN/COMCON
C     COMMON/EIGHT/EQCOMP
C     COMMON/NINE/EQMIN,SURFSP,VREACT
      COMMON/TEN/MOLMAT
      COMMON/ELEVEN/COMP,NAME,COLUMN
      COMMON/TWELVE/CONC
C     COMMON/THIRTEEN/XBOUND
      COMMON/FOURTEEN/NFRONT,TIME
      COMMON/FIFTEEN/IWHERE
C     COMMON/SIXTEEN/POINTS,XPOINT
      COMMON/SEVENTEEN/NBOUND
      COMMON/EIGHTEEN/DOFF
      COMMON/NINETEEN/TIMA
      COMMON/TWENTY/EPS,EWT
      COMMON/TWENTYONE/XP,YP,IN
C     COMMON/TWENTYTWO/NBUFF,CPTR,CACHE
      COMMON/TWENTYTHREE/NCALL
C     COMMON/TWENTYFOUR/MXRATE
      COMMON/TWENTYFIVE/IPR

      OPEN(21,FILE='mydata1',STATUS='UNKNOWN', RECL=132)
      OPEN(31,FILE='mydata2',STATUS='UNKNOWN', RECL=132)
      OPEN(41,FILE='mydata3',STATUS='UNKNOWN', RECL=132)

      OPEN(23,FILE='out1',STATUS='UNKNOWN', RECL=132)
      OPEN(33,FILE='out2',STATUS='UNKNOWN', RECL=132)
      OPEN(43,FILE='out3',STATUS='UNKNOWN', RECL=132)

      OPEN(24,FILE='conc.dat',STATUS='UNKNOWN', RECL=132)
      OPEN(25,FILE='compl.dat',STATUS='UNKNOWN', RECL=132)
      OPEN(27,FILE='totconc.dat',STATUS='UNKNOWN', RECL=132)
      OPEN(28,FILE='testdata',STATUS='UNKNOWN', RECL=132)
      OPEN(29,FILE='ode.dat',STATUS='OLD')
      OPEN(30,FILE='frontrace',STATUS='UNKNOWN', RECL=512)
      CALL READINP
      CALL PRINTINP

      TIME=0.
      READ(29,*) XSTEP, CDIFF
      READ(29,*) EPS,EWT,HMAX
      READ(29,*) STSAVE,STREST
      READ(29,*) DOFF,TIMA,IPR
      WRITE(21,*) 'EPS=',EPS,'PROBLEM ZERO=',EWT,'MAXIMUM STEPSIZE='
     +,HMAX
      IWHERE=-1
      NFRONT=-1

C*****************************************************************
C                         Prepare data
C*****************************************************************

C     ***** Store the initial concentrations *****

      DO 10 J=1,COMPON
         SCONC(J)=CONC(J)
 10   CONTINUE
      DO 20 I=1,NO
         SCOMCO(I)=COMCON(I)
 20   CONTINUE


      IF (STREST.NE.0) THEN
C     ***** RESTART A CALCULATION *****
         WRITE(*,*) 'Restoring previous calculation ... '
         CALL STRESTORE(I,M,TIME,NFRONT)
         IF (I.NE.MILLE.OR.M.NE.MINE) THEN
            WRITE(*,*) 'Wrong data in stsave.dat or exinput...'
            WRITE(*,*) 'Previous state was not restored'
          STOP
         ENDIF
      ENDIF
C     ***** GET THE BOUNDARIES *****
      CALL BUNDIS(COLUMN)

      CALL FRONTRACK(MINE, FTRACK, COLUMN)
      WRITE(30,2000) TIME, (FTRACK(M),M=1,MINE)
      ETIME=TIME

C*****************************************************************
C             Enter the loop for reaction front calculations.
C*****************************************************************

      DO 30 F=1,MXITER

C     ***** Check if there are any minerals left in the column. *****
         J=0
         DO 40 L=1,NBOUND
            DO 50 M=1,MINE
               IF (MOLMAT(L,M).GT.1.D-20) J=1
 50         CONTINUE
 40      CONTINUE
```

```
C     *****  Exit procedure if no minerals left. *****
         IF (J.EQ.0) THEN
            WRITE(*,*) 'END OF MINERAL'
         STOP
         ENDIF

C     ***** CALCULATE THE PROFILE *****

         ISTATE = RECALC(MINE,HMAX,XSTEP,CDIFF,DXDT)
         IF (ISTATE.NE.2) THEN
            WRITE(*,*) 'Function RECALC returned error flag ',ISTATE
            CALL STATESAVE(MILLE,MINE,TIME,NFRONT)
         WRITE(23,*)
            WRITE(23,*) 'Problems encountered calculating this profile'
         WRITE(24,*)
            WRITE(24,*) 'Problems encountered calculating this profile'
            WRITE(24,200) 'X', (COMP(J),J=1,COMPON)
         DO 60 L=1,IN
               WRITE(23,210) XP(L), (DXDT(L,J),J=1,MINE)
               WRITE(24,210) XP(L), (YP(L,J),J=1,COMPON)
60          CONTINUE
200         FORMAT(T7,A,TR9,12(A,TR2))
210         FORMAT(E16.8,10(1X,E11.4))
            STOP
        ENDIF

C     ***** TAKE A STEP *****
        CALL TIMESTEP(IN,DXDT,XP,COLUMN,TIMEMI)

        NFRONT=NFRONT+1
        TIME=TIME+TIMEMI
        WRITE(*,*) NFRONT,'TIME',TIME

      IF (TIME-ETIME.GE.10.0D0) THEN
        CALL FRONTRACK(MINE, FTRACK,COLUMN)
        WRITE(30,2000) TIME, (FTRACK(M),M=1,MINE)
        ETIME = TIME
        CALL FLUSH(30)
       ENDIF

       IF (MOD(NCALL,IPR).EQ.1) THEN
C     ***** TIME TO PRINT OUT RESULTS *****
          CALL PRINTOUT
       ENDIF


C**** Set the concentrations to inlet concentrations ****
       DO 70 J=1, COMPON
          CONC(J)=SCONC(J)
70     CONTINUE
       DO 80 I=1,NO
          COMCON(I)=SCOMCO(I)
80     CONTINUE
      IF(STSAVE.NE.0.AND.MOD(NCALL,IPR).EQ.0) THEN
          CALL STATESAVE(MILLE,MINE,TIME,NFRONT)
       ENDIF

C***** END OF THE MAIN LOOP *****
 30    CONTINUE

      CLOSE(21)
      CLOSE(22)
      CLOSE(23)
      CLOSE(24)
      CLOSE(25)
      CLOSE(26)
      CLOSE(29)
      IF (STSAVE.NE.0) THEN
           CALL STATESAVE(MILLE,MINE,TIME,NFRONT)
      ENDIF

      STOP
 2000 FORMAT(25(2X,G16.8))
      END

C***********************************************************************

      SUBROUTINE STATESAVE(IT,MINE,TIME,NSTEPS)

C     THIS SUBROUTINE WRITES THE FILE stsave.dat
C
C     I      = LOOPING VARIABLE FOR THE POINTS
C     IT     = NUMBER OF POINTS
C     M      = LOOPING VARIABLE FOR THE MINERALS
C     MAXMIN = MAXIMUM NUMBER OF MINERALS
C     MILLE  = MAXIMUM NUMBER OF POINTS
C     MINE   = NUMBER OF MINERALS
C     NSTEPS = NUMBER OF STEPS
C     POINTS = MINERAL COMPOSITION
C     TIME   = TIME CALCULATED
C     XPOINT = X COORDINATE FOR POINTS

      INTEGER I, IT, M, MAXMIN, MILLE, MINE, NSTEPS
      PARAMETER (MAXMIN=25, MILLE=20000)
      DOUBLE PRECISION POINTS(MILLE,MAXMIN), TIME, XPOINT(MILLE)

      COMMON/SIXTEEN/POINTS,XPOINT
      SAVE /SIXTEEN/

      OPEN(67,FILE='stsave.dat',RECL=512,STATUS='UNKNOWN')
      WRITE(67,*) TIME
      WRITE(67,*) NSTEPS
      WRITE(67,*) IT
      WRITE(67,*) MINE
      DO 10 I=1,IT
         WRITE(67,1000) XPOINT(I),(POINTS(I,M),M=1,MINE)
10    CONTINUE
      CLOSE(67)
1000  FORMAT(25(1X,G22.16))
      RETURN
      END
```

```
C***********************************************************************

      SUBROUTINE STRESTORE(IT,MINE,TIME,NSTEPS)

C     THIS SUBROUTINE READS THE FILE stsave.dat
C
C     I      = LOOPING VARIABLE FOR THE POINTS
C     IT     = NUMBER OF POINTS
C     M      = LOOPING VARIABLE FOR THE MINERALS
C     MAXMIN = MAXIMUM NUMBER OF MINERALS
C     MILLE  = MAXIMUM NUMBER OF POINTS
C     MINE   = NUMBER OF MINERALS
C     NSTEPS = NUMBER OF STEPS
C     POINTS = MINERAL COMPOSITION
C     TIME   = TIME CALCULATED
C     XPOINT = X COORDINATE FOR POINTS

      INTEGER I, IT, M, MAXMIN, MILLE, MINE, NSTEPS
      PARAMETER (MAXMIN=25, MILLE=20000)
      DOUBLE PRECISION POINTS(MILLE,MAXMIN), TIME, XPOINT(MILLE)

      COMMON/SIXTEEN/POINTS,XPOINT
      SAVE /SIXTEEN/

      OPEN(67,FILE='stsave.dat',RECL=512,STATUS='UNKNOWN')
      READ(67,*) TIME
      READ(67,*) NSTEPS
      READ(67,*) IT
      READ(67,*) MINE
      DO 10 I=1,IT
         READ(67,1000) XPOINT(I),(POINTS(I,M),M=1,MINE)
10    CONTINUE
1000  FORMAT(25(1X,G22.16))
      CLOSE(67)
      RETURN
      END


C***********************************************************************

      SUBROUTINE PRINTOUT

C     THIS SUBROUTINE PRINTS OUT DATA IN OUTPUT FILES

C     AREA   = AREA OF THE COLUMN   m2
C     CELLS  = NUMBER OF CELLS   -
C     COLUMN = LENGTH OF THE COLUMN   m
C     COMCON = CONCENTRATION OF COMPLEXES   mol/dm3
C     COMP   = NAME OF COMPONENTS   -
C     COMPON = NUMBER OF COMPONENTS   -
C     CONC   = FREE CONCENTRATION OF COMPONENTS   mol/dm3
C     DXCELL = LENGTH OF A CELL
C     GAMBAS = ACTIVITY COEFFICIENT OF COMPONENTS   -
C     GAMCOM = ACTIVITY COEFFICIENT FOR COMPLEXES   -
C     IN     = NUMBER OF VALUES STORED IN XP AND YP   -
C     J      = LOOPING VARIABLE FOR THE COMPONENTS   -
C     K      = LOOPING VARIABLE
```

```
C     L      = LOOPING VARIABLE
C     M      = LOOPING VARIABLE FOR THE MINERALS   -
C     MAXCEL = MAXIMUM NUMBER OF CELLS   -
C     MAXCOM = MAXIMUM NUMBER OF COMPONENTS   -
C     MAXMIN = MAXIMUM NUMBER OF MINERALS   -
C     MAXP   = MAXIMUM NUMBER OF POINTS   -
C     MAXPLX = MAXIMUM NUMBER OF COMPLEXES   -
C     MICELL = CELL CONCENTRATION OF MINERAL
C     MILLE  = MAXIMUM NUMBER OF POINTS IN XPOINT AND POINTS
C     MINE   = NUMBER OF MINERALS    -
C     MINVOL = MOLAR VOLUME OF MINERALS   cm3/mol
C     N      = LOOPING VARIABLE
C     NAME   = NAME OF COMPLEXES   -
C     NFRONT = NUMBER OF CALCULATIONS
C     NO     = NUMBER OF COMPLEXES   -
C     POINTS = MINERAL CONCENTRATION PROFILE
C     POROS  = POROSITY   m3/m3 bulk
C     STOCOM = STOICHIOMETRIC MATRIX FOR COMPLEXES   -
C     SUM    = HELP VARIABLE FOR SUMMATION
C     TIME   = TIME DURING SIMULATION   years
C     TOTCON = TOTAL CONCENTRATION OF THE COMPONENTS   mol/dm3
C     WV     = WATER VOLUME   m2
C     XP     = X COORDINATE FOR YP
C     XPOINT = X COORDINATE FOR POINTS
C     YP     = CONCENTRATION OF COMPONENTS IN XP   mol/dm3
C

      IMPLICIT          NONE
      INTEGER           CELLS, COMPON, I, IN, J, K, L, M, MAXCEL, MAXCOM
      INTEGER           MAXMIN, MAXP, MAXPLX, MILLE, MINE, N, NFRONT, NO
      PARAMETER         (MAXCOM=25, MAXPLX=100, MAXMIN=25, MAXP=10000,
     +                  MAXCEL=MAXMIN*500, MILLE=20000)
      CHARACTER         COMP(MAXCOM)*10, NAME(MAXPLX)*10
      REAL              STOCOM(MAXPLX,MAXCOM)
      DOUBLE PRECISION AREA, COLUMN, COMCON(MAXPLX), CONC(MAXCOM)
      DOUBLE PRECISION DXCELL, GAMBAS(MAXCOM), GAMCOM(MAXPLX)
      DOUBLE PRECISION MICELL(MAXCEL,MAXMIN), MINVOL(MAXMIN)
      DOUBLE PRECISION POINTS(MILLE,MAXMIN),POROS(MILLE), SUM, TIME
      DOUBLE PRECISION TOTCON(MAXP,MAXCOM), WV(MAXCEL), XP(MAXP)
      DOUBLE PRECISION XPOINT(MILLE), YP(MAXP,MAXCOM)

      COMMON/ONE/COMPON
      COMMON/TWO/DXCELL,MINE
      COMMON/THREE/CELLS,AREA
      COMMON/FIVE/NO,STOCOM
      COMMON/SIX/POROS,WV,MINVOL
      COMMON/SEVEN/COMCON
      COMMON/ELEVEN/COMP,NAME,COLUMN
      COMMON/FOURTEEN/NFRONT,TIME
      COMMON/SIXTEEN/POINTS,XPOINT
      COMMON/TWENTYONE/XP,YP,IN

      SAVE /THREE/, /SIXTEEN/

      WRITE(21,*) NFRONT,' TIME',TIME
      WRITE(23,*) NFRONT,' TIME',TIME
      WRITE(24,*) NFRONT,' TIME',TIME
      WRITE(25,*) NFRONT,' TIME',TIME
      WRITE(27,*) NFRONT,' TIME',TIME
      WRITE(28,*) NFRONT,' TIME',TIME

C     ***** CALCULATE MICELL *****
      DO 10 M=1,MINE
         SUM=0
         K=1
         DO 20 L=1,MILLE-1
            IF (XPOINT(L).GE.DXCELL*K) THEN
               MICELL(K,M)=SUM/DXCELL*MINVOL(M)*1.D-6
               SUM=POINTS(L-1,M)*(XPOINT(L)-DXCELL*K)
               K=K+1
            ENDIF
            IF (K.LE.CELLS) THEN
               IF (XPOINT(L+1).GT.K*DXCELL) THEN
                  SUM=SUM+POINTS(L,M)*(K*DXCELL-XPOINT(L))
               ELSE
                  SUM=SUM+POINTS(L,M)*(XPOINT(L+1)-XPOINT(L))
               ENDIF
            ENDIF
 20      CONTINUE
 10   CONTINUE
      WRITE(21,*) 'MICELL AFTER TIMESTEP',NFRONT
      DO 30 K=1,CELLS
         WRITE(21,100) K,(MICELL(K,M),M=1,MINE)
 30   CONTINUE
 100  FORMAT(I3,10(TR2,G9.4))

C     ***** Output of concentrations into file *conc.dat*.*****
      WRITE(24,*)
      WRITE(24,110) 'X',(COMP(J),J=1,COMPON)
      DO 40 L=1,IN
         WRITE(24,120) XP(L),(YP(L,J),J=1,COMPON)
 40   CONTINUE
      CALL FLUSH(24)
 110  FORMAT(T7,A,TR9,12(A,TR2))
 120  FORMAT(E16.8,10(1X,E11.4))

C     ***** Calculate the complex concentrations *****
      WRITE(25,130) 'X',(NAME(I),I=1,NO)
 130  FORMAT(20(TR2,A8))
      DO 50 N=1,IN
         DO 60 J=1,COMPON
            CONC(J)=YP(N,J)
 60      CONTINUE
         CALL ACTCOEFF(CONC,COMPON,GAMBAS,MAXCOM)
         CALL ACTCOEFF(COMCON,NO,GAMCOM,MAXPLX)
         CALL CONCALC(GAMCOM,GAMBAS)
         WRITE(25,140) XP(N),(COMCON(I),I=1,NO)
 140     FORMAT(E16.8,18(1X,E9.3))
C     ***** Calculate the total concentration of the components *****
         DO 70 J=1,COMPON
            TOTCON(N,J)=0.D0
            DO 80 I=1, NO
               TOTCON(N,J)=TOTCON(N,J)+STOCOM(I,J)*COMCON(I)
 80            CONTINUE
               TOTCON(N,J)=TOTCON(N,J)+YP(N,J)
 70         CONTINUE
 50      CONTINUE
         WRITE(27,*) 'TOTAL CONCENTRATION IN SOLUTION'
         WRITE(27,110) 'X',(COMP(J),J=1,COMPON)
         DO 90 N=1,IN
            WRITE(27,150) XP(N),(TOTCON(N,J),J=1,COMPON)
 90      CONTINUE
         CALL FLUSH(27)
 150     FORMAT(E16.8,10(1X,E12.4))

         WRITE(21,*) '******  END OF DATA FOR THE STEP ',NFRONT,' *******'
         WRITE(21,*)
         WRITE(23,*) '******  END OF DATA FOR THE STEP ',NFRONT,' *******'
         WRITE(23,*)


         RETURN
         END

C**********************************************************************

         SUBROUTINE IONACTPROD(GAMBAS,CONC,QAQPRD)
C
C        THIS SUBROUTINE CALCULATES THE ION ACTIVITY PRODUCT, QAQPRD.
C
C        COMPON = NUMBER OF COMPONENTS   -
C        CONC   = CONCENTRATION OF COMPONENTS   mol/dm3
C        DXCELL = CELL LENGTH
C        FLUX   = WATER FLUX
C        GAMBAS = ION ACTIVITY COEFFICIENT   -
C        J      = LOOPING VARIABLE FOR THE COMPONENTS
C        M      = LOOPING VARIABLE FOR THE MINERALS
C        MAXCOM = MAXIMUM NUMBER OF COMPONENTS
C        MAXMIN = MAXIMUM NUMBER OF MINERALS
C        MINE   = NUMBER OF MINERALS   -
C        QAQPRD = ION ACTIVITY PRODUCT   dependent of reaction
C        STOMIN = STOICHIOMETRIC CONSTANTS FOR MINERALS   -
C
         IMPLICIT         NONE
         INTEGER          COMPON, J, M, MAXCOM, MAXMIN, MINE
         PARAMETER        (MAXCOM=25, MAXMIN=25)
         DOUBLE PRECISION CONC(MAXCOM), DXCELL, FLUX, GAMBAS(MAXCOM)
         DOUBLE PRECISION QAQPRD(MAXMIN)
         REAL             STOMIN(MAXMIN,MAXCOM)

         COMMON/ONE/COMPON
         COMMON/TWO/DXCELL,MINE
         COMMON/FOUR/FLUX,STOMIN

         SAVE /FOUR/

         DO 10 M=1,MINE
            QAQPRD(M)=0.0D0
         DO 20 J=1,COMPON
               IF (STOMIN(M,J).NE.0.) THEN
```

```
                IF (CONC(J).LT.0) THEN
                QAQPRD(M)=0.D0
                ELSE
              QAQPRD(M)=QAQPRD(M)+DLOG10(GAMBAS(J)*CONC(J))*STOMIN(M,J)
                ENDIF
            ENDIF
 20       CONTINUE
          IF (QAQPRD(M).GT.307.) THEN
             QAQPRD(M)=1.D307
          ELSEIF (QAQPRD(M).LT.-307.) THEN
             QAQPRD(M)=1.D-307
          ELSE
             QAQPRD(M)=10.**QAQPRD(M)
          ENDIF
 10   CONTINUE
      RETURN
      END

C***********************************************************************

      SUBROUTINE PREDISS(QAQPRD,X,DMXDT)
C
C     THIS SUBROUTINE CALCULATES IF THERE IS ANY POSSIBLE
C     PRECIPITATION OR DISSOLUTION
C
C     CELL   = PRESENT CELLNUMBER   -
C     DMXDT  = VALUE OF POSSIBLE PRECIPITATION/DISSOLUTION
mol/dm3*year
C     DRFORC = DRIVING FORCE FOR PRECIPITATION/DISSOLUTION   dep. of
react.
C     DXCELL = CELL LENGTH
C     EQMIN  = MINERAL EQILIBRIUM CONSTANT   dependent of reaction
C     FM     = CONSTANT USED IN PETER LICHTNER'S VERSION
C     L      = LOOPING VARIABLE FOR THE BOUNDARIES
C     M      = LOOPING VARIABLE FOR THE MINERALS
C     MAXCEL = MAXIMUM NUMBER OF CELLS
C     MAXMIN = MAXIMUM NUMBER OF MINERALS
C     MINE   = NUMBER OF MINERALS   -
C     MOLMAT = MOLAR CONCENTRATION OF MINERAL IN CELLS   mol
C     MSURF  = SURFACE USED IN CALCULATION
C     NBOUND = NUMBER OF BOUNDARIES
C     QAQPRD = ION ACTIVITY PRODUCT   dependent of reaction
C     SURFSP = SPECIFIC SURFACE   m2/m3 bulk
C     VREACT = REACTION RATE FOR MINERALS   dependent of reaction
C     X      = X COORDINATE FOR THE POINT
C     XBOUND = X COORDINATE FOR THE BOUNDARIES
C     ZETMIN = LOGICAL FACTOR, 1. IF POSSIBLE, 0. OTHERWISE   -
C
      IMPLICIT         NONE
      INTEGER          CELL, L, M, MAXCEL, MAXMIN, MINE,NBOUND
      PARAMETER        (MAXMIN=25,MAXCEL=MAXMIN*500)
      DOUBLE PRECISION DMXDT(MAXMIN), DRFORC, DXCELL, EQMIN(MAXMIN), FM
      DOUBLE PRECISION MOLMAT(MAXCEL,MAXMIN), MSURF, QAQPRD(MAXMIN)
      DOUBLE PRECISION SURFSP, VREACT(MAXMIN), X, XBOUND(MAXCEL+2)
      DOUBLE PRECISION ZETMIN
      COMMON/TWO/DXCELL,MINE
      COMMON/NINE/EQMIN,SURFSP,VREACT
      COMMON/TEN/MOLMAT
      COMMON/THIRTEEN/XBOUND
      COMMON/SEVENTEEN/NBOUND

      SAVE /NINE/, /THIRTEEN/


      CELL=0
      DO 10 L=1,NBOUND
         IF (X.GE.XBOUND(L).AND.X.LT.XBOUND(L+1)) CELL=L
 10   CONTINUE

C       FM=4
      DO 20 M=1,MINE
         DRFORC = QAQPRD(M)-EQMIN(M)
C     ***** PETER LICHTNER'S VERSION *****
C          DRFORC = -(1-EQMIN(M)*QAQPRD(M))/(1+EQMIN(M)*QAQPRD(M)/FM)
         CALL ZETA(MOLMAT(CELL,M),DRFORC,EQMIN(M),ZETMIN)
         IF (ZETMIN.EQ.0.D0) THEN
            DMXDT(M)=0.D0
         ELSE
            IF (DRFORC.GT.0.D0) THEN
               MSURF=100.D0*SURFSP
            ELSE
               MSURF=SURFSP
            ENDIF
            DMXDT(M)=MSURF*VREACT(M)*DRFORC
         ENDIF
 20   CONTINUE
      RETURN
      END

C*********************************************************************

            SUBROUTINE ZETA(XMIN,DRFORC,EQ,ZETMIN)
C
C     THIS SUBROUTINE DETERMINES IF THERE IS PRECIPITATION/DISSOLUTION
C     OR NOT AND SETS ZETMIN TO ZERO OR ONE ACCORDINGLY
C
C     DRFORC = DRIVING FORCE OF PRECIPITATION/DISSOLUTION   dep.of
react.
C     XMIN   = MOLMAT(M,CELL), MASSFRACTION OF MINERAL M   kg/kg
C     ZETMIN = LOGICAL FACTOR, 1. IF POSSIBLE, 0. OTHERWISE   -
C     EQ     = EQILIBRIUM CONSTANT
C
      IMPLICIT          NONE
      DOUBLE PRECISION DRFORC, EQ, XMIN, ZETMIN

      IF (DRFORC.LT.0.D0.AND.XMIN.GT.0.D0) THEN
         ZETMIN=1.D0
      ELSEIF (DRFORC.GT.0.D0) THEN
         ZETMIN=1.D0
      ELSE
         ZETMIN=0.D0
      ENDIF
      RETURN
      END

C***********************************************************************


      SUBROUTINE CONCALC(GAMCOM,GAMBAS,CONC)

C     THIS SUBROUTINE CALCULATES THE FREE CONCENTRATION OF THE COMPLEXES
C
C     COMCON = FREE CONCENTRATION OF THE COMPLEXES   mol/dm3
C     COMPON = NUMBER OF COMPONENTS   -
C     CONC   = CONCENTRATION OF THE COMPONENTS
C     DF     = CONTRIBUTION FRON THE COMPONENTS
C     EQCOMP = EQUILIBRIUM CONSTANT FOR COMPLEXES   dep. of react.
C     GAMBAS = ACTIVITY COEFFICIENT FOR COMPONENTS   -
C     GAMCOM = ACTIVITY COEFFICIENT FOR COMPLEXES   -
C     I      = LOOPING VARIABLE FOR THE COMPLEXES
C     J      = LOOPING VARIABLE FOR THE COMPONENTS
C     MAXCOM = MAXIMUM NUMBER OF COMPONENTS
C     MAXPLX = MAXIMUM NUMBER OF COMPLEXES
C     NO     = NUMBER OF COMPLEXES   -
C     STOCOM = STOICHIOMETRIC CONSTANTS FOR COMPLEXES   -
C     X      = SUM OF CONTRIBUTIONS

      IMPLICIT         NONE
      INTEGER          COMPON, I, J, MAXCOM, MAXPLX, NO
      PARAMETER        (MAXCOM=25, MAXPLX=100)
      DOUBLE PRECISION COMCON(MAXPLX), CONC(MAXCOM), DF, EQCOMP(MAXPLX)
      DOUBLE PRECISION GAMBAS(MAXCOM), GAMCOM(MAXPLX), X
      REAL             STOCOM(MAXPLX,MAXCOM)

      COMMON/ONE/COMPON
      COMMON/FIVE/NO,STOCOM
      COMMON/SEVEN/COMCON
      COMMON/EIGHT/EQCOMP

      SAVE /EIGHT/

      DO 10 I=1,NO
         X=1.D0
         DO 20 J=1,COMPON
            IF (STOCOM(I,J).NE.0.) THEN
               IF (CONC(J).LE.0.0D0) X=0.0D0
               IF (X.GT.0.0D0) THEN
                 DF=(GAMBAS(J)*CONC(J))**STOCOM(I,J)
                 X=X*DF
               ENDIF
            ENDIF
 20      CONTINUE
         COMCON(I)=X*(EQCOMP(I)/GAMCOM(I))
 10   CONTINUE
      RETURN
      END

C***********************************************************************
```

```
      SUBROUTINE ACTCOEFF(C,N,GAMMA,M)
C
C     THIS SUBROUTINE CALCULATES THE ACTIVITY COEFFICIENT FOR THE
C     COMPONENTS OR THE COMPLEXES
C
C     C      = CONCENTRATION IN CALCULATION   mol/dm3
C     GAMMA  = ACTIVITY COEFFICIENT FOR THE SPECIE
C     K      = LOOPING VARIABLE
C     M      = MAXIMUM NUMBER OF SPECIES
C     N      = NUMBER OF SPECIES IN CALCULATION   -

      IMPLICIT NONE
      INTEGER           K, M, N
      DOUBLE PRECISION C(M), GAMMA(M)

      DO 10 K=1,N
         GAMMA(K)=1.0D0
 10   CONTINUE

C     THIS VERSION OF ACTCOEFF IS TO BE CHANGED LATER ON.

      RETURN
      END

C***********************************************************************

      SUBROUTINE TIMESTEP(IN,DXDT,XP,COLUMN,TIMEMI)

C     THIS SUBROUTINE CALCULATES THE TIME STEP SIZE.
C
C     COLUMN = LENGTH OF THE COLUMN
C     DXCELL = CELL LENGTH   m
C     DXDT   = MINERAL DISSOLUTION AND PRECIPITATION RATE
C     IN     = NUMBER OF POINTS IN XP AND DXDT
C     K      = HELP FOR PRINTOUT
C     L      = LOOPING VARIABLE FOR THE CELLS
C     LL     =COUNTING VARIABLE
C     M      = LOOPING VARIABLE FOR THE MINERALS
C     MAXCEL = MAXIMUM NUMBER OF CELLS
C     MAXMIN = MAXIMUM NUMBER OF MINERALS
C     MAXP   = MAXIMUM NUMBER OF POINTS IN XP AND DXDT
C     MILLE  = MAXIMUM NUMBER OF POINTS IN POINTS AND XPOINT
C     MINE   = NUMBER OF MINERALS   -
C     MINMIN = MINERAL WITH SHORTEST TIME FOR MAXIMUM REACTION
C     MOLMAT = MINERAL MOLAR AMOUNT IN CELLS   mol
C     NBOUND = NUMBER OF BOUNDARIES
C     NCALL  = NUMBER OF CALLS TO RECALC
C     POINTS = MINERAL CONCENTRATION
C     SUM    = HELP VARIABLE FOR INTEGRATION OF DXDT
C     TIMA   = MAXIMUM TIME STEP SIZE
C     TIMEMI = MINIMUM TIME FOR STEP SIZE   year
C     TIMMIN = THE MINIMUM TIME FOR MAXIMUM REACTION
C     TMINST = MINIMUM VALUE FOR A TIMESTEP
C     XBOUND = X COORDINATE FOR THE BOUNDARIES
C     XLO    = LOWEST X COORDINATE IN COMPARISION
C     XP     = X COORDINATE FOR DXDT
```

```
C       XPOINT = X COORDINATE FOR POINTS

        IMPLICIT         NONE
        INTEGER          IN, IPR, K, L, LL, M, MAXCEL, MAXMIN, MAXP, MILLE
        INTEGER          MINE, MINMIN, NBOUND, NCALL
        PARAMETER        (MAXMIN=25, MAXCEL=MAXMIN*500,MAXP=10000)
        PARAMETER        (MILLE=20000)
        DOUBLE PRECISION COLUMN, DXCELL, DXDT(MAXP,MAXMIN)
        DOUBLE PRECISION MOLMAT(MAXCEL,MAXMIN), POINTS(MILLE,MAXMIN)
        DOUBLE PRECISION SUM(MAXMIN), TIMA, TIMEMI, TIMMIN, TMINST
        DOUBLE PRECISION XBOUND(MAXCEL+2), XLO, XP(MAXP), XPOINT(MILLE)

        COMMON/TWO/DXCELL,MINE
        COMMON/TEN/MOLMAT
        COMMON/THIRTEEN/XBOUND
        COMMON/SIXTEEN/POINTS,XPOINT
        COMMON/SEVENTEEN/NBOUND
        COMMON/NINETEEN/TIMA
        COMMON/TWENTYTHREE/NCALL
        COMMON/TWENTYFIVE/IPR

        SAVE /THIRTEEN/, /SIXTEEN/

        TIMEMI=TIMA
        TMINST = 0.5D-2
        CALL SMOOTHER( MINE, IN, XP, DXDT )
        CALL REACBET( MINE, XP, DXDT, IN)

        MINMIN=0
        LL=1
        XP(IN+1)=XP(IN)+COLUMN
        DO 10 M=1,MINE
           DXDT(IN+1,M)=DXDT(IN,M)
 10     CONTINUE

        DO 20 L=1,MILLE-1
           IF (XPOINT(L).LT.XBOUND(NBOUND+1).AND.XPOINT(L).GE.XP(1)) THEN

C       ***** GET THE MINIMUM TIME FOR MAXIMUM REACTION *****

           DO 30 M=1,MINE
                SUM(M)=0.D0
 30           CONTINUE

           IF (XPOINT(L).GE.XP(LL)) LL=LL+1
           IF (LL.LE.IN) THEN
                IF (XP(LL).GT.XPOINT(L)) THEN
                   XLO=MIN(XP(LL),XPOINT(L+1))
              DO 40 M=1,MINE
                      SUM(M)=SUM(M)+DXDT(LL-1,M)*(XLO-XPOINT(L))
 40                CONTINUE
                ENDIF

 50             IF (XP(LL).LT.XPOINT(L+1)) THEN
                   XLO=MIN(XP(LL+1),XPOINT(L+1))
              DO 60 M=1,MINE
                 SUM(M)=SUM(M)+DXDT(LL,M)*(XLO-XP(LL))
 60                 CONTINUE
                  ENDIF
                IF  (XLO.LT.XPOINT(L+1).AND.LL.LT.IN)  THEN
                    LL=LL+1
                GOTO 50
                  ENDIF

                  DO 70 M=1,MINE
                IF  (SUM(M).LT.0.D0)  THEN
                    TIMMIN=-POINTS(L,M)*(XPOINT(L+1)-XPOINT(L))/SUM(M)
                  IF  (TIMMIN.GT.TMINST)  THEN
                                                      IF(TIMEMI.GT.TIMMI
N)  THEN
                                                        TIMEMI=TIMMIN
                                                        MINMIN=M
                                                      ENDIF
                  ELSE
                                                      POINTS(L,M)=0.0D0
                    ENDIF
                  ENDIF
 70               CONTINUE
              ENDIF
           ENDIF
 20    CONTINUE

       WRITE(28,*)  'TIMMIN=',TIMEMI,' FOR MINERAL #',MINMIN

C      ***** TAKE THE TIMESTEP
       LL=1
       DO 80 L=1,MILLE-1
          IF  (XPOINT(L).LT.XBOUND(NBOUND+1).AND.XPOINT(L).GE.XP(1))  THEN
          DO 90 M=1,MINE
              SUM(M)=0.D0
 90         CONTINUE

          IF(XPOINT(L).GE.XP(LL))  LL=LL+1
          IF(LL.LE.IN)  THEN

              IF  (XP(LL).GT.XPOINT(L))  THEN
                 XLO=MIN(XP(LL),XPOINT(L+1))
                 DO 100 M=1,MINE
                    SUM(M)=SUM(M)+DXDT(LL-1,M)*(XLO-XPOINT(L))
 100             CONTINUE
              ENDIF

 346        IF  (XP(LL).LT.XPOINT(L+1))  THEN
                XLO=MIN(XP(LL+1),XPOINT(L+1))
            DO 110 M=1,MINE
                SUM(M)=SUM(M)+DXDT(LL,M)*(XLO-XP(LL))
 110             CONTINUE
            ENDIF
            IF  (XLO.LT.XPOINT(L+1).AND.LL.LT.IN)  THEN
                LL=LL+1
            GOTO 346
            ENDIF

            IF  (XPOINT(L+1)-XPOINT(L).GT.0.0D0)  THEN
```

```
                DO 120 M=1,MINE
                 POINTS(L,M)=POINTS(L,M)+SUM(M)*TIMEMI/(XPOINT(L+1)-
     +                 XPOINT(L))
                       IF (POINTS(L,M).LT.0.1D-10) THEN
                    POINTS(L,M)=0.0D0
                 ENDIF
 120               CONTINUE
                ELSE
                WRITE(*,*) 'ZERO AT L=', L,' x=', XPOINT(L)
                ENDIF
             ENDIF
          ENDIF
 80    CONTINUE

       CALL POINTKILLER(MINE)

C      ***** PRINT THE DATA IF ASKED FORE *****

       IF (MOD(NCALL,IPR).EQ.0) THEN
          WRITE(21,*) 'NEW POINTS'
          WRITE(31,*) 'NEW POINTS'
          WRITE(41,*) 'NEW POINTS'
          DO 130 L=1,MILLE
             IF (XPOINT(L).LE.XBOUND(NBOUND+1)) THEN
                DO 140 K=1,MINE/10+1
                   IF (K.EQ.1) THEN
                      WRITE(21,800) XPOINT(L),(POINTS(L,M),M=1,MIN(MINE,
     +                10))
                   ELSEIF(K.EQ.2) THEN
                      WRITE(31,800) XPOINT(L),(POINTS(L,M),M=11,MIN(MINE,
     +                20))
                   ELSEIF(K.EQ.3) THEN
                      WRITE(41,800) XPOINT(L),(POINTS(L,M),M=21,MIN(MINE,
     +                30))
 800                  FORMAT(G16.8,10G12.4)
                   ENDIF
 140            CONTINUE
             ENDIF
 130      CONTINUE
          CALL FLUSH(21)
          CALL FLUSH(31)
          CALL FLUSH(41)
       ENDIF

C      ***** GET THE NEW BOUNDARIES *****

       CALL BUNDIS(COLUMN)
       RETURN

       END

C***********************************************************************

       SUBROUTINE REACBET(MINE,XP,DXDT,IN)

C      THIS SUBROUTINE INSERTS A POINT WHERE REACTION STARTS OR STOPS
```

```
C
C     DXDT   = DISSOLUTION OR PRECIPITATION RATE
C     IN     = NUMBER OF POINTS IN XP AND DXDT
C     K      = LOOPING VARIABLE
C     L      = LOOPING VARIABLE
C     LL     = LOOPING VARIABLE
C     M      = LOOPING VARIABLE
C     MAXMIN = MAXIMUM NUMBER OF MINERALS
C     MAXP   = MAXIMUM NUMBER OF POINTS IN XP AND DXDT
C     MILLE  = MAXIMUM NUMBER OF POINTS IN POINTS
C     MINE   = NUMBER OF MINERALS
C     N      = LOOPING VARIABLE
C     NCALL  = NUMBER OF CALLS
C     XP     = X COORDINATE FOR DXDT
C     XPOINT = X COORDINATE FOR POINTS


      IMPLICIT         NONE
      INTEGER                                                   IN, K, L,
LL, M, MAXMIN, MAXP, MILLE, MINE, N
      INTEGER          NCALL
      PARAMETER        (MAXMIN=25, MAXP=10000, MILLE=20000)
      DOUBLE PRECISION DXDT(MAXP,MAXMIN), POINTS(MILLE,MAXMIN)
      DOUBLE PRECISION XPOINT(MILLE), XP(MAXP)

      COMMON/SIXTEEN/POINTS,XPOINT
      COMMON/TWENTYTHREE/NCALL

      SAVE /SIXTEEN/

      L=0
      DO 20 LL=1,IN

C     ***** SEE IF THERE IS MORE SPACE *****

 544    IF (XPOINT(L+1).LE.XP(LL))  THEN
          L=L+1
          IF (L.GE.MILLE) THEN
          WRITE(*,*) 'Out of MILLE POINTS. Terminating...'
            CALL STATESAVE(XPOINT,POINTS,MILLE,MINE,0.0,NCALL)
          STOP
        ENDIF
          GOTO 544
        ENDIF

C     ***** SEE IF THERE IS A BOUNDARY *****

        DO 10 M=1,MINE
          IF (XPOINT(L).LT.XP(LL)) THEN
            IF ((DXDT(LL,M).LT.0.D0.AND.DXDT(LL-1,M).GE.0.D0).OR.
     +          (DXDT(LL,M).EQ.0.D0.AND.DXDT(LL-1,M).NE.0.D0).OR.
     +          (DXDT(LL,M).GT.0.D0.AND.DXDT(LL-1,M).LE.0.D0)
     +         ) THEN

C     *****  INSERT A POINT AT THE BOUNDARY *****

              DO 15 K=MILLE,L+1,-1
```

```
                  DO 25 N=1,MINE
                     POINTS(K,N)=POINTS(K-1,N)
 25               CONTINUE
                  XPOINT(K)=XPOINT(K-1)
 15             CONTINUE
           L=L+1
                XPOINT(L)=XP(LL)
              ENDIF
            ENDIF
 10       CONTINUE
 20     CONTINUE

        RETURN
        END

C*********************************************************************

        SUBROUTINE BUNDIS(COLUMN)

C       THIS SUBROUTINE CALCULATES THE BOUNDARIES WHERE THE MINERALS
C       APPEAR OR DISSAPPEAR
C
C       AREA   = COLUMN AREA   m2
C       CELLS  = NUMBER OF CELLS    -
C       DXCELL = CELL LENGTH    m
C       IC     = FIRST CELL WITH MINERAL
C       IFLAG  = FLAG, 1 IF THERE ARE ANY CHANGES
C       L      = LOOPING VARIABLE FOR CELLS
C       M .    = LOOPING VARIABLE FOR MINERALS
C       MAXCEL = MAXIMUM NUMBER OF CELLS
C       MAXMIN = MAXIMUM NUMBER OF MINERALS
C       MILLE  = MAXIMUM NUMBER OF POINTS
C       MINE   = NUMBER OF MINERALS    -
C       MINNE  = VECTOR WITH THE MINERAL COMPOSITION
C       MOLMAT = MOLAR CONCENTRATION OF MINERAL IN CELLS    mol
C       NBOUND = NUMBER OF BOUNDARIES    -
C       POINTS = MATRIX WITH THE MINERAL COMPOSITION
C       XBOUND = THE BOUNDARIES    m
C       XM1    = MINERAL CONTENT IN UPPER CELL
C       XM2    = MINERAL CONTENT IN LOWER CELL
C       XPOINT = X COORDINATE IN POINTS
C
        IMPLICIT          NONE
        INTEGER*8                                         MINNE(500)
        INTEGER           CELLS, IC, IFLAG, L, M, MAXCEL, MAXMIN, MINE
        INTEGER           MILLE, NBOUND
        PARAMETER         (MAXMIN=25, MAXCEL=MAXMIN*500,MILLE=20000)
        DOUBLE PRECISION AREA, COLUMN, DXCELL, MOLMAT(MAXCEL,MAXMIN)
        DOUBLE PRECISION POINTS(MILLE,MAXMIN), XBOUND(MAXCEL+2), XM1, XM2
        DOUBLE PRECISION XPOINT(MILLE)

        COMMON/TWO/DXCELL,MINE
        COMMON/THREE/CELLS,AREA
        COMMON/TEN/MOLMAT
        COMMON/THIRTEEN/XBOUND
        COMMON/SIXTEEN/POINTS,XPOINT
        COMMON/SEVENTEEN/NBOUND
```

```
      SAVE /THREE/, /THIRTEEN/, /SIXTEEN/

C     FIRST DETERMINE WHERE THE MINERALS START

      IFLAG=0
      IC=1
      MINNE(1)=0
      DO 64 L=1,MILLE
         IF (IFLAG.EQ.0) THEN
            DO 63 M=1,MINE
               MINNE(1)=MINNE(1)*2
               IF (POINTS(L,M).GT.0.0D0) THEN
                  IFLAG=1
                  MINNE(1)=MINNE(1)+1
               ENDIF
 63         CONTINUE
            XBOUND(1)=XPOINT(L)
            IC=L+1
        ENDIF
 64   CONTINUE
      IF (XBOUND(1).GE.COLUMN) THEN
      NBOUND=0
      RETURN
      ENDIF
C
C     DETERMINE WHERE THE BOUNDARIES BETWEEN CELLS WITH DIFFERENT
C     MINERALS ARE
C
      NBOUND=1
      DO 66 L=IC,MILLE
       IF(XPOINT(L).GE.COLUMN) GOTO 150
         IFLAG=0
         DO 65 M=1,MINE
            XM1 = POINTS(L,M)
            XM2 = POINTS(L-1,M)
            IF (((XM1.LE.0.0D0).AND.(XM2.GT.0.0D0)).OR.((XM1.GT.0.0D0)
     +         .AND.(XM2.LE.0.0D0))) IFLAG=1
 65      CONTINUE
         IF (IFLAG.EQ.1)THEN
            NBOUND=NBOUND+1
            XBOUND(NBOUND)=XPOINT(L)
            MINNE(NBOUND)=0
            DO 11 M=1,MINE
               MINNE(NBOUND)=MINNE(NBOUND)*2
               IF (POINTS(L,M).GT.0.D0) MINNE(NBOUND)=MINNE(NBOUND)+1
 11         CONTINUE

         ENDIF
 66   CONTINUE

 150  XBOUND(NBOUND+1)=COLUMN

C     ***** CREATE A NEW MOLMAT *****

      DO 81 L=1,NBOUND
```

```
          DO 82 M=MINE,1,-1
             MOLMAT(L,M)=DBLE(MOD(MINNE(L),2))
             MINNE(L)=INT(MINNE(L)/2)
 82       CONTINUE
 81    CONTINUE

       RETURN
       END

C*************************************************************************

       SUBROUTINE DIFF(NEQ,X,CONC,YDOT)
C
C      THIS SUBROUTINE CALCULATES THE RIGHT HAND SIDE OF THE EQUATION
C      THE FOR SDRIV2-PACAGE
C
C      CELL   = PRESENT CELL   -
C      COMPON = NUMBER OF COMPONENTS   -
C      CONC   = CONCENTRATION OF COMPONENTS   mol/dm3
C      DMXDT  = VALUE OF POSSIBLE PRECIPITATION/DISSOLUTION   mol/year
C      DXCELL = CELL LENGTH
C      GAMBAS = ACTIVITY COEFFICIENT FOR COMPONENTS   -
C      MAXCEL = MAXIMUM NUMBER OF CELLS
C      MAXCOM = MAXIMUM NUMBER OF COMPONENTS
C      MAXMIN = MAXIMUM NUMBER OF MINERALS
C      MINE   = NUMBER OF MINERALS   -
C      MOLMAT = MOLAR CONCENTRATION OF MINERALS IN CELLS   mol
C      NEQ    = NUMBER OF EQUATIONS
C      QAQPRD = ION ACTIVITY PRODUCT   dependent of reaction
C      X      = DISTANCE FROM START OF COLUMN   m
C      YDOT   = THE RIGHT HAND SIDE OF THE EQUATION
C
       IMPLICIT          NONE
       INTEGER           COMPON, MINE
       INTEGER           MAXCEL, MAXCOM, MAXMIN, NEQ
       INTEGER           GETCELL
       PARAMETER         (MAXCOM=25, MAXMIN=25,MAXCEL=MAXMIN*500)
       DOUBLE PRECISION CONC(MAXCOM), DMXDT(MAXMIN), DXCELL
       DOUBLE PRECISION GAMBAS(MAXCOM), QAQPRD(MAXMIN), X, YDOT(*)

       COMMON/ONE/COMPON
       COMMON/TWO/DXCELL,MINE


       CALL ACTCOEFF(CONC,COMPON,GAMBAS,MAXCOM)
       CALL IONACTPROD(GAMBAS,CONC,QAQPRD)
       CALL PREDISS(QAQPRD,X,DMXDT)
       CALL FVECTOR(MINE,DMXDT,YDOT)

       RETURN
       END

C*************************************************************************

       SUBROUTINE FA(NEQ,X,CONC,A,MATDIM,ML,MU,NDE)
C
```

```
C     THIS SUBROUTINE IS CALLED BY SDRIV3 TO GIVE THE MATRIX A IN THE
C     LEFT HAND SIDE OF THE EQUATION
C
C     A      = THE MATRIX A
C     COMCON = CONCENTRATION OF COMPLEXES   mol/dm3
C     COMPON = NUMBER OF COMPONENTS   -
C     CONC   = CONCENTRATION OF COMPONENTS   mol/dm3
C     GAMBAS = ACTIVITY COEFFICIENT FOR COMPONENTS   -
C     GAMCOM = ACTIVITY COEFFICIENTS FOR COMPLEXES   -
C     MATDIM = MATRIX DIMENTION OF A, FROM SDRIV3
C     MAXCOM = MAXIMUM NUMBER OF COMPONENTS
C     MAXPLX = MAXIMUM NUBBER OF COMPLEXES
C     ML     = NUMBER OF LOWER DIAGONALS
C     MU     = NUMBER OF UPPER DIAGONALS
C     NDE    = NUMBER OF DIFFERENTIAL EQUATIONS
C     NEQ    = NUMBER OF DIFFERENTIAL EQUATIONS
C     NO     = NUMBER OF COMPLEXES   -
C     STOCOM = STOICHIOMETRIC CONSTANT FOR COMPLEXES   -
C
      IMPLICIT          NONE
      INTEGER           COMPON, MATDIM, MAXCOM, MAXPLX
      INTEGER           ML, MU, NDE, NEQ, NO
      PARAMETER         (MAXCOM=25, MAXPLX=100)
      DOUBLE PRECISION A(MATDIM,*), COMCON(MAXPLX),  CONC(1)
      DOUBLE PRECISION GAMBAS(MAXCOM), GAMCOM(MAXPLX)
      REAL              STOCOM(MAXPLX,MAXCOM)

      COMMON/ONE/COMPON
      COMMON/FIVE/NO,STOCOM
      COMMON/SEVEN/COMCON

      CALL ACTCOEFF(CONC,COMPON,GAMBAS,MAXCOM)
      CALL ACTCOEFF(COMCON,NO,GAMCOM,MAXPLX)
      CALL CONCALC(GAMCOM,GAMBAS,CONC)
      CALL AMATRIX(COMPON,NO,STOCOM,COMCON,CONC,A,MATDIM)

      RETURN
      END

C*********************************************************************
      SUBROUTINE AMATRIX(COMPON,NO,STOCOM,COMCON,CONC,A,MATDIM)
C
C     THIS SUBROUTINE MAKES THE A MATRIX IN THE LEFT HAND SIDE OF THE
SYSTEM
C
C     A      = THE MATRIX A IN THE LEFT HAND SIDE OF THE SYSTEM
C     COMCON = CONCENTRATION OF THE COMPLEXES
C     COMPON = NUMBER OF COMPONENTS
C     CONC   = CONCENTRATION OF THE COMPONENTS
C     I      = LOOPING VARIABLE FOR THE COMPLEXES
C     J      = LOOPING VARIABLE FOR THE COMPONENTS
C     K      = LOOPING VARIABLE FOR THE COMPONENTS
C     MATDIM = MATRIX DIMENTION, COMMING FROM SDRIV3
C     MAXCOM = MAXIMUM NUMBER OF COMPONENTS
C     MAXPLX = MAXIMUM NUMBER OF COMPLEXES
C     NO     = NUMBER OF COMPLEXES
C     STOCOM = STOICHIOMETRIC COEFFICIENT FOR THE COMPLEXES
```

```
C
      IMPLICIT NONE
      INTEGER          COMPON, I, J, K, MATDIM, MAXCOM, MAXPLX, NO
      PARAMETER         (MAXCOM=25, MAXPLX=100)
      REAL             STOCOM(MAXPLX,MAXCOM)
      DOUBLE PRECISION A(MATDIM,*), COMCON(MAXPLX), CONC(MAXCOM)

C     START WITH A AS A UNIT MATRIX

      DO 10 K=1, COMPON
         DO 20 J=1, COMPON
            A(J,K)=0.D0
 20      CONTINUE
 10   CONTINUE
      DO 30 J=1, COMPON
         A(J,J)=1.D0
 30   CONTINUE

      DO 40 J=1, COMPON
         DO 50 I=1, NO
            DO 60 K=1, COMPON
               A(J,K)=A(J,K)+STOCOM(I,J)*STOCOM(I,K)*COMCON(I)/CONC(K)
 60         CONTINUE
 50      CONTINUE
 40   CONTINUE

      RETURN
      END

C***********************************************************************

      SUBROUTINE FVECTOR(MINE,DMXDT,F)
C
C     THIS SUBROUTINE MAKES THE F VECTOR AS THE RIGHT HAND SIDE OF THE
SYSTEM
C
C     COMPON = NUMBER OF COMPONENTS
C     DMXDT  = MINERAL DISSOLUTION AND PRECIPITATION RATE
C     F      = DISSOLUTION/PRECIPITATION RATE
C     FLUX   = THE WATER FLUX
C     J      = LOOPING VARIABLE FOR THE COMPONENTS
C     M      = LOOPING VARIABLE FOR THE MINERALS
C     MAXCOM = MAXIMUM NUMBER OF COMPONENTS
C     MAXMIN = MAXIMUM NUMBER OF MINERALS
C     MINE   = NUMBER OF MINERALS
C     STOMIN = STOICHIOMETRIC COEFFICIENT FOR THE MINERAL
C     SUM    = SUMATION VARIABLE
C
      IMPLICIT NONE
      INTEGER          COMPON, J, M, MAXCOM, MAXMIN, MINE
      PARAMETER         (MAXCOM=25, MAXMIN=25)
      REAL             STOMIN(MAXMIN,MAXCOM)
      DOUBLE PRECISION DMXDT(MAXMIN), F(COMPON), FLUX, SUM

      COMMON/ONE/COMPON
      COMMON/FOUR/FLUX,STOMIN
      SAVE /FOUR/

      DO 10 J=1, COMPON
         F(J)=-1.0D-3/FLUX
         SUM=0.D0
         DO 20 M=1,MINE
            SUM=SUM+STOMIN(M,J)*DMXDT(M)
 20      CONTINUE
         F(J)=F(J)*SUM
 10   CONTINUE

      RETURN
      END

C**********************************************************************
      SUBROUTINE READINP

C     THIS SUBROUTINE READS THE INPUT FILE EXINPUT

C     A      = STRING FOR READING TEXT NOT INTERESTING   -
C     AREA   = AREA OF THE COLUMN   m2
C     CELLS  = NUMBER OF CELLS   -
C     COLUMN = LENGTH OF THE COLUMN   m
C     COMCON = CONCENTRATION OF COMPLEXES   mol/dm3
C     COMP   = NAME OF COMPONENTS   -
C     COMPON = NUMBER OF COMPONENTS   -
C     CONC   = FREE CONCENTRATION OF COMPONENTS   mol/dm3
C     CONST  = CONSTANT TO CALCULATE VREACT AT EQMODE 1
C     DXCELL = CELL LENGTH   m
C     EQCOM  = LOGARITHMIC EQUILIBRIUM CONSTANT FOR THE COMPLEXES   -
C     EQCOMP = EQUILIBRIUM CONSTANS FOR COMPLEXES   dependent of
reaction
C     EQMI   = LOGARITHMIC EQUILIBRIUM CONSTANTS FOR MINERALS   -
C     EQMIN  = EQUILIBRIUM CONSTANT FOR MINERALS   dependent of reaction
C     EQMODE = 1 IF AUTOMATIC EQUILIBRIUM MODE IS CHOOSEN
C     FLUX   = FLUX OF THE WATER   m3(water)/(m2(column)*year)
C     I      = LOOPING VARIABLE FOR THE COMPLEXES   -
C     J      = LOOPING VARIABLE FOR THE COMPONENTS   -
C     L      = VARIABLE USED FOR READING NOT INTERESTING VALUES   -
C     M      = LOOPING VARIABLE FOR THE MINERALS   -
C     MAXCOM = MAXIMUM NUMBER OF COMPONENTS   -
C     MAXMIN = MAXIMUM NUMBER OF MINERALS   -
C     MAXPLX = MAXIMUM NUMBER OF COMPLEXES   -
C     MINE   = NUMBER OF MINERALS   -
C     MINCON = MINERAL CONCENTRATION   m3/m3
C     MINVOL = MOLAR VOLUME OF MINERALS   cm3/mol
C     MOLMAT = MOLAR AMOUNT OF MINERAL IN EACH CELL   mol
C     MXRATE = MAXIMUM DISSOLUTION RATE
C     NAME   = NAME OF COMPLEXES   -
C     NO     = NUMBER OF COMPLEXES   -
C     POR    = POROSITY
C     POROS  = POROSITY   m3/m3 bulk
C     STATUS = UNIT OF THE CONCENTRATION IN INPUT
C     STOCOM = STOICHIOMETRIC MATRIX FOR COMPLEXES   -
C     STOMIN = STOICHIOMETRIC MATRIX FOR MINERALS   -
C     SURFSP = SPECIFIC SURFACE   m2/m3 bulk
```

```
C     VREACT = REACTION RATE OF MINERALS   dependent of reaction
C     WV     = WATER VOLUME IN EACH CELL   m3


      IMPLICIT         NONE
      INTEGER          CELLS, COMPON, EQMODE, I, J, L,MILLE,PO
      INTEGER          M, MAXCEL, MAXCOM, MAXMIN, MAXPLX, MINE, NO
      PARAMETER        (MAXCOM=25, MAXPLX=100, MAXMIN=25)
      PARAMETER        (MAXCEL=MAXMIN*500, MILLE=20000)
      CHARACTER        A*80, COMP(MAXCOM)*10, NAME(MAXPLX)*10
      CHARACTER        STATUS(MAXCOM)*1
      REAL             STOCOM(MAXPLX,MAXCOM), STOMIN(MAXMIN,MAXCOM)
      DOUBLE PRECISION AREA, COMCON(MAXPLX), CONC(MAXCOM),X1,X2
      DOUBLE PRECISION COLUMN, CONST, DXCELL, POINTS(MILLE,MAXMIN)
      DOUBLE PRECISION EQCOM, EQCOMP(MAXPLX), EQMI, EQMIN(MAXMIN)
      DOUBLE PRECISION FLUX, MINCON(MAXMIN), MINVOL(MAXMIN)
      DOUBLE PRECISION MOLMAT(MAXCEL,MAXMIN), POR, POROS(MILLE)
      DOUBLE PRECISION SURFSP, VREACT(MAXMIN), WV(MAXCEL)
      DOUBLE PRECISION XPOINT(MILLE), MXRATE

      COMMON/ONE/COMPON
      COMMON/TWO/DXCELL,MINE
      COMMON/THREE/CELLS,AREA
      COMMON/FOUR/FLUX,STOMIN
      COMMON/FIVE/NO,STOCOM
      COMMON/SIX/POROS,WV,MINVOL
      COMMON/SEVEN/COMCON
      COMMON/EIGHT/EQCOMP
      COMMON/NINE/EQMIN,SURFSP,VREACT
      COMMON/TEN/MOLMAT
      COMMON/ELEVEN/COMP,NAME,COLUMN
      COMMON/TWELVE/CONC
      COMMON/SIXTEEN/POINTS,XPOINT
      COMMON/TWENTYFOUR/MXRATE

      SAVE /THREE/, /FOUR/, /SIX/, /EIGHT/, /NINE/, /SIXTEEN/

      OPEN(22,FILE='exinput',STATUS='OLD')
C
C
C     ***** READ INPUT DATA FROM THE FILE EXINPUT *****

      READ(22,*) A
      READ(22,*) A
      READ(22,*) COMPON
      READ(22,*) A
      READ(22,*) MINE
      READ(22,*) A
      READ(22,*) NO
      READ(22,*) A
      DO 10 J=1,COMPON
         READ(22,*) COMP(J)
 10   CONTINUE
      READ(22,*) A
      DO 20 I=1,NO
         READ(22,*) NAME(I)
 20   CONTINUE
```

```
      READ(22,*) A
      READ(22,*) L
      DO 30 I=1,NO
         READ(22,*) L,(STOCOM(I,J),J=1,COMPON)
 30   CONTINUE
      READ(22,*) A
      READ(22,*) L
      DO 40 M=1,MINE
         READ(22,*)  L,(STOMIN(M,J),J=1,COMPON)
 40   CONTINUE
      READ(22,*) A
      DO 50 J=1,COMPON
         READ(22,*) L, CONC(J), STATUS(J)
 50   CONTINUE
      DO 60 I=1,NO
         COMCON(I)=1.0D-55
 60   CONTINUE
      READ(22,*) A
      READ(22,*) CELLS
      READ(22,*) A
      READ(22,*) POR
      READ(22,*) A
      DO 70 M=1,MINE
         READ(22,*) L,MINCON(M)
 70   CONTINUE
      READ(22,*) A
      READ(22,*) SURFSP
      READ(22,*) A
      READ (22,*) EQMODE
      IF (EQMODE.EQ.1) THEN
         READ(22,*) A
         READ(22,*) CONST
      ELSE
         READ(22,*) A
         DO 80 M=1,MINE
            READ(22,*) L,VREACT(M)
 80      CONTINUE
      ENDIF
      READ(22,*) A
      READ(22,*) AREA
      READ(22,*) A
      READ(22,*) FLUX
      READ(22,*) A
      READ(22,*) COLUMN
      READ(22,*) A
      READ(22,*) PO
      READ(22,*) A
      DO 90 M=1,MINE
         READ(22,*) L, MINVOL(M)
 90   CONTINUE
      READ(22,*) A
      DO 100 M=1,MINE
         READ(22,*) L, EQMI
         EQMIN(M)=10.D0**EQMI
 100  CONTINUE
C     IF EQMODE IS ONE CALCULATE VREACT
```

```
      IF (EQMODE.EQ.1) THEN
         DO 110 M=1,MINE
            VREACT(M)=CONST/EQMIN(M)
 110     CONTINUE
      ENDIF
      READ(22,*) A
      DO 120 I=1, NO
         READ(22,*) L,EQCOM
         EQCOMP(I)=10.0D0**EQCOM
 120  CONTINUE

C     ***** END OF READING INPUTDATA *****

C     ***** CALCULATE SOME DATA *****
C
C     ***** CALCULATE CELL LENGTH, DXCELL *****
C
      DXCELL=(COLUMN/DBLE(CELLS))
C
C     ***** CALCULATE MAXIMUM DISSOLUTION RATE *****
C
      MXRATE=CONST*SURFSP*0.9998
C
C     ***** PREPATE MOLMAT *****

      DO 130 L=1,CELLS
         DO 140 M=1,MINE
            IF (MINCON(M).GT.0.D0) THEN
               MOLMAT(L,M)=1
            ELSEIF (MINCON(M).EQ.0.D0) THEN
               MOLMAT(L,M)=0
            ELSE
               WRITE(*,*) 'WRONG INPUT DATA, MINERAL CONCENTRATION =',
     +                    MINCON(M)
               STOP
            ENDIF
 140     CONTINUE
 130  CONTINUE

C     ***** PREPARE XPOINT *****

      DO 150 L=1,MILLE
         XPOINT(L)=DBLE(L-1)*COLUMN/DBLE(PO)
 150  CONTINUE

C     ***** PREPARE POINTS *****
      DO 160 L=1,MILLE
         DO 170 M=1,MINE
           POINTS(L,M)=MINCON(M)
 170     CONTINUE
 160  CONTINUE
C     ***** PREPARE POROS *****

      DO 180 L=1, CELLS
         POROS(L)=POR
 180  CONTINUE
```

```
C     ***** CALCULATE WATER VOLUME IN EACH CELL. *****

      DO 190 L=1,CELLS
         WV(L)=AREA*DXCELL*POROS(L)
 190  CONTINUE

C     ***** MAKE THE SPECIATION OF THE AQUEOUS SPECIES *****

      CALL SPECI(STATUS)

      RETURN
      END

C************************************************************************

      SUBROUTINE PRINTINP

C     THIS SUBROUTINE PRINTS THE INPUT DATA ON THE FILE MYDATA
C
C     AREA   = AREA OF THE COLUMN   m2
C     CELLS  = NUMBER OF CELLS   -
C     COLUMN = LENGTH OF THE COLUMN   m
C     COMCON = CONCENTRATION OF COMPLEXES   mol/dm3
C     COMP   = NAME OF COMPONENTS   -
C     COMPON = NUMBER OF COMPONENTS   -
C     CONC   = FREE CONCENTRATION OF COMPONENTS   mol/dm3
C     DXCELL = CELL LENGTH   m
C     EQCOMP = EQUILIBRIUM CONSTANS FOR COMPLEXES   dependent of
reaction
C     EQMIN  = EQUILIBRIUM CONSTANT FOR MINERALS   dependent of reaction
C     FLUX   = FLUX OF THE WATER   m3(water)/(m2(column)*year)
C     I      = LOOPING VARIABLE FOR THE COMPLEXES   -
C     J      = LOOPING VARIABLE FOR THE COMPONENTS   -
C     L      = LOOPING VARIABLE FOR THE CELLS   -
C     M      = LOOPING VARIABLE FOR THE MINERALS   -
C     MAXCEL = MAXIMUM NUMBER OF CELLS   -
C     MAXCOM = MAXIMUM NUMBER OF COMPONENTS   -
C     MAXPLX = MAXIMUM NUMBER OF COMPLEXES   -
C     MINE   = NUMBER OF MINERALS   -
C     MINVOL = MOLAR VOLUME OF MINERALS   cm3/mol
C     MOLMAT = MOLAR AMOUNT OF MINERAL IN EACH CELL   mol
C     NAME   = NAME OF COMPLEXES   -
C     NO     = NUMBER OF COMPLEXES   -
C     POROS  = POROSITY   m3/m3 bulk
C     STOCOM = STOICHIOMETRIC MATRIX FOR COMPLEXES   -
C     STOMIN = STOICHIOMETRIC MATRIX FOR MINERALS   -
C     SURFSP = SPECIFIC SURFACE   m2/m3 bulk
C     VREACT = REACTION RATE OF MINERALS   dependent of reaction
C     WV     = WATER VOLUME IN EACH CELL   m3
C

      IMPLICIT         NONE
      INTEGER          CELLS, COMPON, I, J, L, M, MAXCEL, MAXCOM, MAXMIN
      INTEGER          MAXPLX, MILLE, MINE, NO
      PARAMETER        (MAXCOM=25, MAXPLX=100, MAXMIN=25, MILLE=20000)
      PARAMETER        (MAXCEL=MAXMIN*500)
      CHARACTER        COMP(MAXCOM)*10, NAME(MAXPLX)*10
      REAL             STOCOM(MAXPLX,MAXCOM), STOMIN(MAXMIN,MAXCOM)
      DOUBLE PRECISION AREA, COMCON(MAXPLX), CONC(MAXCOM), COLUMN
      DOUBLE PRECISION DXCELL, EQCOMP(MAXPLX), EQMIN(MAXMIN), FLUX
      DOUBLE PRECISION MINVOL(MAXMIN), MOLMAT(MAXCEL,MAXMIN)
      DOUBLE PRECISION POROS(MILLE), POINTS(MILLE,MAXMIN), SURFSP
      DOUBLE PRECISION VREACT(MAXMIN), WV(MAXCEL),XPOINT(MILLE)

      COMMON/ONE/COMPON
      COMMON/TWO/DXCELL,MINE
      COMMON/THREE/CELLS,AREA
      COMMON/FOUR/FLUX,STOMIN
      COMMON/FIVE/NO,STOCOM
      COMMON/SIX/POROS,WV,MINVOL
      COMMON/SEVEN/COMCON
      COMMON/EIGHT/EQCOMP
      COMMON/NINE/EQMIN,SURFSP,VREACT
      COMMON/TEN/MOLMAT
      COMMON/ELEVEN/COMP,NAME,COLUMN
      COMMON/TWELVE/CONC
      COMMON/SIXTEEN/POINTS,XPOINT

      SAVE /THREE/, /FOUR/, /SIX/, /EIGHT/, /NINE/, /SIXTEEN/

C     ***** WRITE THE INPUTDATA ON THE FILE MYDATA1 *****

      WRITE(21,*) 'FILE MYDATA'
      WRITE(21,*)
      WRITE(21,200) 'NUMBER OF COMPONENTS', COMPON
 200  FORMAT(A,T25,I2)
      WRITE(21,200) 'NUMBER OF MINERALS', MINE
      WRITE(21,200) 'NUMBER OF COMPLEXES', NO
      DO 10 J=1,COMPON
         WRITE(21,210) 'NAME OF COMPONENT',J,COMP(J)
 210     FORMAT(A,T18,I2,T26,A)
 10   CONTINUE
      DO 20 I=1,NO
         WRITE(21,210) 'NAME OF COMPLEX', I, NAME(I)
 20   CONTINUE
      WRITE(21,*) 'COMPLEXES COMPONENTS'
      WRITE(21,220)  (J,J=1,COMPON)
 220  FORMAT(T2,10(TR8,I2))
      DO 30 I=1,NO
         WRITE(21,230) I, (STOCOM(I,J),J=1,COMPON)
 230     FORMAT(I2,TR6,10(F6.2,TR4))
 30   CONTINUE
      WRITE(21,*) ' MINERALS COMPONENTS'
      WRITE(21,220)  (J,J=1,COMPON)
      DO 40 M=1,MINE
         WRITE(21,230) M, (STOMIN(M,J),J=1,COMPON)
 40   CONTINUE
      DO 50 J=1,COMPON
        WRITE(21,240) 'FREE CONCENTRATION OF COMPONENT',J,'IS',CONC(J)
 50   CONTINUE
 240     FORMAT(A,T33,I2,TR2,A,TR2,E9.3)
      WRITE(21,250) 'NUMBER OF CELLS ',CELLS
```

```
  250  FORMAT(A,T18,I3)
       WRITE(21,260) 'POROSITY',POROS(1)
  260  FORMAT(A,T18,F6.4)
       DO 60 M=1,MINE
          WRITE(21,270) 'CONCENTRATION OF MINERAL',M,POINTS(1,M),
     +       '(mol/m3)'
  270     FORMAT(A,T26,I2,F14.4,TR5,A)
  60   CONTINUE
       WRITE(21,280) 'SPECIFIC SURFACE IS',SURFSP,'(m2/m3 bulk)'
  280  FORMAT(A,TR2,F10.2,TR5,A)
       DO 70 M=1,MINE
          WRITE(21,290) 'REACTION RATE FOR MINERAL',M,'IS'
     +          ,VREACT(M),'(MOL/YEARS)'
  70   CONTINUE
  290  FORMAT(A,I2,TR1,A,E15.2,TR5,A)
       WRITE(21,300) 'AREA',AREA, '(m2)'
  300  FORMAT(A,T22,F6.2,TR5,A)
       WRITE(21,300) 'FLUX',FLUX,'(m3(water)/m2(column)*year)'
       WRITE(21,300) 'LENGTH OF COLUMN', COLUMN,'(m)'
       DO 80 M=1,MINE
          WRITE(21,310) 'MOLAR VOLUME OF MINERAL',M,MINVOL(M),
     +                  '(cm3/mol)'
  80   CONTINUE
  310  FORMAT(A,I6,F12.3,TR5,A)
       DO 90 M=1,MINE
          WRITE(21,320) 'LOG EQUILIBRIUM CONSTANT FOR MINERAL',M,
     +                  LOG10(EQMIN(M))
  90   CONTINUE
  320  FORMAT(A,I5,F14.4)
       DO 100 I=1,NO
          WRITE(21,320) 'LOG EQUILIBRIUM CONSTANT FOR COMPLEXES',
     +       I,LOG10(EQCOMP(I))
  100  CONTINUE

       WRITE(21,*) '***** HERE IS THE END OF INPUTDATA *****'


       RETURN
       END

C**********************************************************************


       SUBROUTINE SPECI(STATUS)
C
C      THIS SUBROUTINE MAKES THE SPECIATION OF THE AQUEOUS SPECIES
C
C      COMCON = COMPLEX CONCENTRATION
C      COMPON = NUMBER OF COMPONENTS
C      CONC   = COMPONENT CONCENTRATION
C      CONC1  = COMPONENT CONCENTRATION IN CALCULATION
C      ERRLIM = MAXIMUM RELATIVE ERROR IN TOTAL CONCENTRATION
C      GAMBAS = ION ACTIVITY COEFFICIENT FOR THE COMPONENTS
C      GAMCOM = ION ACTIVITY COEFFICIENT FOR THE COMPLEXES
C      I      = LOOPING VARIABLE FOR THE COMPLEXES
C      IFLAG  = FLAG THO INDICATE WHEN THE SPECIATION IS READY
C      J      = LOOPING VARIABLE FOR THE COMPONENTS
```

```
C     K      = LOOPING VARIABLE FOR THE COMPONENTS
C     MAXCOM = MAXIMUM NUMBER OF COMPONENTS
C     MAXPLX = MAXIMUM NUMBER OF COMPLEXES
C     MXCALC = MAXIMUM NUMBER OF CALCULATIONS
C     NO     = NUMBER OF COMPLEXES
C     TOTCON = TOTAL CONCENTRATION OF THE COMPONENT
C     STATUS = UNIT OF INCOMMING CONCENTRATION
C     STOCOM = STOICHIOMETRIC COEFFICIENT FOR THE COMPLEXES
C


      IMPLICIT         NONE
      INTEGER          COMPON, I, IFLAG, J, K, MAXCOM, MAXPLX, MXCALC
      INTEGER          NO
      PARAMETER        (MAXCOM=25, MAXPLX=100, MXCALC=5000)
      DOUBLE PRECISION COMCON(MAXPLX), CONC(MAXCOM), CONC1(MAXCOM)
      DOUBLE PRECISION ERRLIM, GAMBAS(MAXCOM), GAMCOM(MAXPLX)
      DOUBLE PRECISION TOTCON(MAXCOM)
      CHARACTER        STATUS(MAXCOM)*1
      REAL             STOCOM(MAXPLX,MAXCOM)

      COMMON/ONE/COMPON
      COMMON/FIVE/NO,STOCOM
      COMMON/SEVEN/COMCON
      COMMON/TWELVE/CONC

      DATA ERRLIM/1.D-6/

      DO 10 J=1, COMPON
         CONC1(J)=CONC(J)
 10   CONTINUE
      DO 20 K=1, MXCALC
         IFLAG=0.
C     ***** CALCULATE THE COMPLEX CONCENTRATION *****
         CALL ACTCOEFF(CONC,COMPON,GAMBAS,MAXCOM)
         CALL ACTCOEFF(COMCON,NO,GAMCOM,MAXPLX)
         CALL CONCALC(GAMCOM,GAMBAS,CONC)
C     ***** CALCULATE THE TOTAL CONCENTRATION *****
         DO 30 J=1, COMPON
            TOTCON(J)=0.D0
            DO 40 I=1,NO
               TOTCON(J)=TOTCON(J)+STOCOM(I,J)*COMCON(I)
 40         CONTINUE
            TOTCON(J)=TOTCON(J)+CONC(J)
 30      CONTINUE
         DO 50 J=1, COMPON
            IF (STATUS(J).EQ.'T'.OR.STATUS(J).EQ.'t') THEN
                  IF (ABS(TOTCON(J)/CONC1(J)-1).GT.ERRLIM) THEN
                     CONC(J)=CONC(J)*CONC1(J)/TOTCON(J)
                     IF (CONC(J).LT.1.D-280) CONC(J)=1.D-280
                     IFLAG=1
                  ENDIF
            ENDIF
 50      CONTINUE
         IF (IFLAG.EQ.0) THEN
            RETURN
         ENDIF
 20   CONTINUE

      WRITE(*,*) 'SPECIATION FAILED'

      STOP
      END

C*******************************************************************

      SUBROUTINE SMOOTHER( MINE,IN,XP,DXDT )

C     THIS SUBROUTINE AVERAGE THE PRECIPITATION RATE DURING MAXIMUM
DISSOLUTION

C     AREA   = THE AREA OF THE DXDT
C     DXDT   = PRECIPITATION AND DISSOLUTION RATE
C     I      = LOOPING VARIABLE
C     IB     = STORES THE NUMBER OF BOUNDARIES
C     IBX    = WHERE PRECIPITATION STARTS
C     II     = LOOPING VARIABLE
C     IN     = NUMBER OF POINTS IN XP AND DXDT
C     ISMAX  = STORES HOW MANY MINERALS THAT HAVE MAXIMUM DISSOLUTION
RATE
C     K      = LOOPING VARIABLE
C     M      = LOOPING VARIABLE FOR THE MINERALS
C     MAXB   = MAXIMUM NUMBER OF BOUNDARIES
C     MAXMIN = MAXIMUM NUMBER OF MINERALS
C     MAXP   = MAXIMUM NUMBER OF POINTS
C     MINE   = NUMBER OF MINERALS
C     MXRATE = MAXIMUM DISSOLUTION RATE
C     NXA    = STORES THE BOUNDARY POINTS FOR MAXIMUM DISSOLUTION RATE
C     XP     = X COORDINATE FOR DXDT

      IMPLICIT         NONE

      INTEGER          I, IBX, II, IN, ISMAX, K, M, MAXB, MAXMIN, MAXP
      INTEGER          MINE
      PARAMETER       (MAXMIN=25, MAXB=10*MAXMIN, MAXP=10000)
      INTEGER          IB(0:MAXMIN), NXA(2*MAXB,0:MAXMIN)
      DOUBLE PRECISION AREA, DXDT(MAXP,MAXMIN), MXRATE, XP(MAXP)

      COMMON/TWENTYFOUR/MXRATE
      SAVE /TWENTYFOUR/

C     ***** SET THE STARTING VALUES *****

      DO 10 M=0,MINE
         IB(M)=0
 10   CONTINUE
      ISMAX=0

C     ***** Find the regions with the maximum dissolution rate *****
      DO 20 I=1,IN
         DO 30 M=1,MINE
            IF (MOD(IB(M),2).EQ.0) THEN
             IF (-DXDT(I,M).GE.MXRATE) THEN
```

```
                IB(M)=IB(M)+1
                   NXA(IB(M),M)=I
                   ISMAX=ISMAX+1
               ENDIF
           ELSE
            IF (-DXDT(I,M).LT.MXRATE) THEN
                IB(M)=IB(M)+1
            NXA(IB(M),M)=I-1
                ISMAX=ISMAX-1
           ENDIF
           ENDIF
 30      CONTINUE

C     ***** In NXA(*,0) and IB(0) we store the union of all regions with
C     maximum dissolution rate. *****
         IF (MOD(IB(0),2).EQ.0) THEN
          IF (ISMAX.GT.0) THEN
               IB(0)=IB(0)+1
               NXA(IB(0),0)=I
          ENDIF
      ELSE
         IF (ISMAX.EQ.0) THEN
               IB(0)=IB(0)+1
               NXA(IB(0),0)=I-1
         ENDIF
      ENDIF
20    CONTINUE

C     ***** Average the precipitation rates. *****
      DO 40 M=1,MINE
      IBX=0
      DO 50 K=1,IB(0),2
         DO 60 I = NXA(K,0), NXA(K+1,0)
               IF (DXDT(I,M).GT.0.0D0.AND.IBX.EQ.0) IBX=I
               IF (DXDT(I,M).LE.0.0D0.AND.IBX.GT.0) THEN
           AREA=0.0D0
           DO 70 II=IBX,I-1
               AREA=AREA+DXDT(II,M)*(XP(II+1)-XP(II))
 70                CONTINUE
                   IF (XP(I)-XP(IBX).NE.0.D0) THEN
                      AREA=AREA/( XP(I)-XP(IBX) )
                   ELSE
                      AREA=0.D0
                      WRITE(*,*) 'XP(I)-XP(IBX)=0'
                   ENDIF
                   DO 80 II=IBX,I-1
               DXDT(II,M)=AREA
 80                CONTINUE
                   IBX=0
              ENDIF
 60        CONTINUE
           IF (IBX.GT.0) THEN
              AREA=0.0D0
              DO 90 II=IBX,NXA(K+1,0)
           AREA=AREA+DXDT(II,M)*(XP(II+1)-XP(II))
 90           CONTINUE
              IF (XP(NXA(K+1,0)+1)-XP(IBX).NE.0.D0) THEN
```

```
                  AREA=AREA/( XP(NXA(K+1,0)+1)-XP(IBX) )
               ELSE
                  AREA=0.D0
                  WRITE(*,*) 'XP(NXA(K+1,0)+1)-XP(IBX)=0'
               ENDIF
               DO 100 II=IBX,NXA(K+1,0)
             DXDT(II,M)=AREA
 100           CONTINUE
               IBX=0
          ENDIF
 50       CONTINUE
 40   CONTINUE

      RETURN
      END

C***********************************************************************

      SUBROUTINE POINTKILLER(MINE)

C     THIS SUBROUTINE TAKES AWAY UNNECESSARY INFORMATION IN POINTS AND
XPOINT

C     COLUMN = LENGTH OF THE COLUMN
C     COMP   = NAME OF THE COMPONENTS
C     I      = COUNTING VARIABLE
C     II     = LOOPING VARIABLE
C     K      = LOOPING VARIABLE
C     KILL   = IF THE POINT WILL BE DELETED OR NOT
C     M      = LOOPING VARIABLE FOR THE MINERALS
C     MAXCOM = MAXIMUM NUMBER OF COMPONENTS
C     MAXMIN = MAXIMUM NUMBER OF MINERALS
C     MAXPLX = MAXIMUM NUMBER OF COMPLEXES
C     MILLE  = MAXIMUM NUMBER OF POINTS IN POINTS AND XPOINT
C     MINE   = NUMBER OF MINERALS
C     NAME   = NAME OF THE COMPONENTS
C     POINTS = MINERAL COMPOSITION WITHIN THE COLUMN
C     XDIFF  = DISTANCE BETWEEN POINTS
C     XPOINT = X COORDINATE FOR THE VALUES IN POINTS

      IMPLICIT         NONE
      LOGICAL          KILL
      INTEGER          I, II, K, M
      INTEGER          MAXCOM, MAXMIN, MAXPLX, MILLE, MINE
      PARAMETER        (MAXCOM=25, MAXMIN=25, MAXPLX=100, MILLE=20000)
      DOUBLE PRECISION COLUMN, POINTS(MILLE,MAXMIN), XDIFF
      DOUBLE PRECISION XPOINT(MILLE)
      CHARACTER        COMP(MAXCOM)*10, NAME(MAXPLX)*10

      COMMON/ELEVEN/COMP,NAME,COLUMN
      COMMON/SIXTEEN/POINTS,XPOINT

      SAVE /SIXTEEN/

      I=1
      DO 10 K=1,MILLE
         IF (I+1.GT.MILLE.OR.XPOINT(I).GE.COLUMN) RETURN
```

```
            KILL=.TRUE.
            DO 20 M=1,MINE
             IF (XPOINT(I)-XPOINT(I-1).GE.1.0D-1*COLUMN.OR.
     +            POINTS(I-1,M).NE.0.0D0.AND.POINTS(I,M).EQ.0.0D0.OR.
     +            POINTS(I-1,M).EQ.0.0D0.AND.POINTS(I,M).NE.0.0D0.OR.
     +            DABS(POINTS(I-1,M)-POINTS(I,M)).GT.
     +            DABS(POINTS(I-1,M)+POINTS(I,M))*0.1)
     +            KILL=.FALSE.
 20         CONTINUE

            IF (KILL) THEN

C     ***** Recalculate mineral concentration to hold mass conservation
*****

              XDIFF=XPOINT(I+1)-XPOINT(I-1)
              IF(XDIFF.GT.0.0D0) THEN
                DO 30 M=1,MINE
                  POINTS(I-1,M)=(POINTS(I-1,M)*(XPOINT(I)-XPOINT(I-1))
     +                +POINTS(I,M)*(XPOINT(I+1)-XPOINT(I)))/XDIFF
 30             CONTINUE
              ENDIF

              DO 40 II=I,MILLE-1
                XPOINT(II)=XPOINT(II+1)
                DO 50 M=1,MINE
                  POINTS(II,M)=POINTS(II+1,M)
 50             CONTINUE
 40           CONTINUE
              XPOINT(MILLE)=XPOINT(MILLE-1)+0.1D-3
            ELSE
              I=I+1
            ENDIF
 10       CONTINUE

          END

C*********************************************************************

          SUBROUTINE FRONTRACK(MINE, XPOS, COLUMN)

C     THIS SUBROUTINE CHECKS THE POSITION OF THE MINERAL DISSOLUTION
FRONTS ARE

C     COLUMN = LENGTH OF THE COLUMN
C     I      = LOOPING VARIABLE FOR THE BOUNDARIES
C     M      = LOOPING VARIABLE FOR THE MINERALS
C     MAXCEL = MAXIMUM NUMBER OF CELLS
C     MAXMIN = MAXIMUM NUMBER OF MINERALS
C     MINE   = NUMBER OF MINERALS
C     MOLMAT = MINERAL DISTRIBUTION
C     NBOUND = NUMBER OF BOUNDARIES
C     XBOUND = THE BOUNDARIES
C     XPOS   = POSITION OF THE DISSOLUTION FRONTS
```

```
      IMPLICIT         NONE
      INTEGER          I, K, M, MAXCEL, MAXMIN, MINE, NBOUND
      PARAMETER        (MAXMIN=25, MAXCEL=500*MAXMIN)
      DOUBLE PRECISION COLUMN, MOLMAT(MAXCEL,MAXMIN), XBOUND(MAXCEL*2)
      DOUBLE PRECISION XPOS(*)

      COMMON/TEN/MOLMAT
      COMMON/THIRTEEN/XBOUND
      COMMON/SEVENTEEN/NBOUND

      SAVE /THIRTEEN/

      DO 10 M=1,MINE
         XPOS(M)=COLUMN
 10   CONTINUE

      DO 20 I=1,NBOUND
         DO 30 M=1,MINE
            IF (MOLMAT(I,M).GT.0.0D0) THEN
            XPOS(M)=MIN(XPOS(M),XBOUND(I))
         ENDIF
 30      CONTINUE
 20   CONTINUE

      DO 40 M=1,MINE
       IF(XPOS(M).EQ.COLUMN) XPOS(M)=0.0D0
 40   CONTINUE

      RETURN
      END

C***********************************************************************

      INTEGER FUNCTION RECALC(MINE,HMAX,XSTEP,CDIFF,DXDT)

C     THIS FUNCTION CALCULATES THE PROFILES

C     BLOCKS = DIMENSION OF CPTR
C     CACHE  = VECTOR WITH INFORMATION ABOUT THE PROFILES
C     CACHED = LOGICAL TO TELL IF THE FRONT IS SAVED OR NOT
C     CDIFF  = MAXIMUM RELATIVE DIFFERENCE FOR THE CONCENTRATIONS
C     COMPON = NUMBER OF COMPONENTS
C     CONC   = FREE CONCENTRATION OF THE COMPONENTS
C     CPTR   = POSITION WHERE BUFFER STARTS
C     CSIZE  = DIMENSION OF CACHE
C     DDIV   = MINIMUM RELATIVE DIFFERENCE IN REACTION
C     DMXDT  = MINERAL DISSOLUTION/PRECIPITATION RATE
C     DOFF   = MINIMUM DISSOLUTION OR PRECIPITATION
C     DXDT   = MINERAL DISSOLUTION/PRECIPITATION RATE
C     EPSYL  = RELATIVE DISTANCE BETWEEN XHI AND NEXT XBOUND
C     FIRST  = FIRST POINT IN A BLOCK
C     GAMBAS = ACTIVITY COEFFICIENT FOR THE COMPONENTS
C     HITS   = NUMBER OF TIMES ANY PROFILE IN CACHE ARE USED
C     HMAX   = MAXIMUM STEPSIZE
C     IFLAG  = FLAG TO INDICATE THE SUCCESS OF THE CALCULATION
C     IN     = NUMBER OF POINTS IN XP AND YP
C     ITOWR  = NUMBER OF THE BUFFER TO DELETE
```

```
C      J       = LOOPING VARIABLE FOR THE COMPONENTS
C      KK      = LOOPING VARIABLE
C      L       = LOOPING VARIABLE
C      LAST    = LAST POINT IN A BLOCK
C      LL      = LOOPING VARIABLE
C      M       = LOOPING VARIABLE
C      MAKEBLOCK = FUNCTION TO CREATES A NEW BUFFER IN CACHE
C      MAXCAL = MAXIMUM NUMBER OF CALLS
C      MAXCEL = MAXIMUM NUMBER OF CELLS
C      MAXCOM = MAXIMUM NUMBER OF COMPONENTS
C      MAXMIN = MAXIMUM NUMBER OF MINERALS
C      MAXP   = MAXIMUM NUMBER OF POINTS
C      MAXPLX = MAXIMUM NUMBER OF COMPLEXES
C      MINE   = NUMBER OF MINERALS
C      MISSES = NUMBER OF TIME TO WRITE NEW PROFILE IN CACHE
C      MOLMAT = MINERAL DISTRIBUTION
C      NBOUND = NUMBER OF BOUNDARIES
C      NBUFF  = NUMBER OF BUFFERS
C      NCALL  = NUMBER OF CALLS
C      NPTR   = POSITION IN CACHE
C      NRETRY = NUMBER OF TIMES SOLVER IS CALLED
C      OLDCON = CONCENTRATION IN THE PREVIOUS POINT
C      OLDRE  = MINERAL REACTION IN THE PREVIUOS STEP
C      OVERWR = LOGICAL TO OVERWRITE PROFILE IN CACHE
C      QAQPRD = ION ACTIVITY PRODUCT
C      SIZE   = SIZE OF BLOCK
C      XBOUND = THE BOUNDARIES
C      XHI    = UPPER BOUNDARY
C      XLOW   = PRESENT POINT
C      XOLD   = PREVIOUS POINT
C      XP     = X COORDINATE CHOSEN BY THE PROGRAM
C      XSTEP  = MAXIMUM DISTANCE BETWEEN SAVED POINTS
C      YP     = FREE CONCENTRATION OF COMPONENTS IN XP

      LOGICAL          CACHED, OVERWR
      INTEGER          BLOCKS, COMPON, CSIZE, FIRST, HITS,IFLAG
      INTEGER          IN, ITOWR, J, KK, L, LAST, LL, M, MAKEBLOCK
      INTEGER          MAXCAL, MAXCEL, MAXCOM, MAXMIN, MAXP, MAXPLX
      INTEGER          MINE, MISSES, NBOUND, NBUFF, NCALL, NPTR
      INTEGER          NRETRY, SIZE
      PARAMETER       (BLOCKS=125, CSIZE=250000, MAXCAL=5000, MAXCOM=25,
     +                 MAXMIN=25, MAXCEL=MAXMIN*500, MAXPLX=100,
     +                 MAXP=10000)
      INTEGER          CPTR(2*BLOCKS)
      DOUBLE PRECISION CACHE(CSIZE), CDIFF, CONC(MAXCOM), DDIV
      DOUBLE PRECISION DMXDT(MAXMIN), DOFF, DXDT(MAXP,MAXMIN), EPSYL
      DOUBLE PRECISION GAMBAS(MAXCOM), HMAX, MOLMAT(MAXCEL,MAXMIN)
      DOUBLE PRECISION OLDCON(MAXCOM), OLDRE(MAXMIN), QAQPRD(MAXMIN)
      DOUBLE PRECISION XBOUND(MAXCEL+2), XHI, XLOW, XOLD, XP(MAXP)
      DOUBLE PRECISION XSTEP, YP(MAXP,MAXCOM)

      SAVE             HITS,MISSES
      DATA             NCALL/0/,NBUFF/0/,HITS/0/,MISSES/0/

      COMMON/ONE/COMPON
      COMMON/TEN/MOLMAT
      COMMON/TWELVE/CONC
      COMMON/THIRTEEN/XBOUND
      COMMON/SEVENTEEN/NBOUND
      COMMON/EIGHTEEN/DOFF
      COMMON/TWENTYONE/XP,YP,IN
      COMMON/TWENTYTWO/NBUFF,CPTR,CACHE
      COMMON/TWENTYTHREE/NCALL

      SAVE /THIRTEEN/, /TWENTYTWO/

      NCALL=NCALL+1
      EPSYL=1.0D-6
      DDIV=0.1D0

C     ***** BLOCK BEGIN: LOOP FOR REACTION FRONTS *****


      IN=0
      DO 10 L=1, NBOUND

         XLOW=XBOUND(L)
         XHI=XBOUND(L+1)-EPSYL*(XBOUND(L+1)-XBOUND(L))
         WRITE(*,*) XLOW,XHI
         FIRST=IN+1
      NRETRY=0
         CACHED=.FALSE.
         OVERWR=.FALSE.

         CALL CHEQUP(MINE,CACHED,OVERWR,NPTR,L,
     +           CDIFF,HITS,XLOW,DXDT,XHI,ITOWR,IFLAG)

C     ***** IF NEXT BOUNDARY IS REACHED *****
         IF (IFLAG.EQ.1) GOTO 10

         XOLD=XLOW
         IF (.NOT.CACHED) THEN
            IN=IN+1
            XP(IN)=XLOW
            DO 20 J=1,COMPON
               YP(IN,J)=CONC(J)
               OLDCON(J)=CONC(J)
20          CONTINUE
            CALL ACTCOEFF(CONC, COMPON, GAMBAS, MAXCOM)
            CALL IONACTPROD(GAMBAS, CONC, QAQPRD)
            CALL PREDISS (QAQPRD, XLOW, DMXDT)
            DO 30 M=1, MINE
               IF (DABS(DMXDT(M)).LT.DOFF) THEN
                  DMXDT(M)=0.0D0
               ENDIF
30          CONTINUE
            DO 40 M=1, MINE
               DXDT(IN,M)=DMXDT(M)
               OLDRE(M)=DMXDT(M)
40          CONTINUE
         ELSE
            DO 50 J=1,COMPON
               CONC(J)=YP(IN,J)
```

```
              OLDCON(J)=CONC(J)
 50         CONTINUE
            DO 60 M=1, MINE
              DMXDT(M)=DXDT(IN,M)
              OLDRE(M)=DMXDT(M)
 60         CONTINUE
          ENDIF

C     ***** BLOCK BEGIN: LOOP FOR SOLVER CALLS *****
C     ***** Check if the next reaction front at XHI has been reached.
*****
          DO 70 LL=1,25
            IF (XLOW.LT.XHI) THEN
              NRETRY=NRETRY+1

              CALL SOLVCALL(XLOW,XHI,HMAX,RECALC,MINE,OLDRE,
    +                DXDT,DDIV,XSTEP,FIRST,MISSES,OLDCON,XOLD,IFLAG)

              IF (IFLAG.EQ.1) RETURN

          ENDIF
 70       CONTINUE
          IF (XLOW.LT.XHI) THEN
            WRITE(*,1000) NRETRY*MAXCAL, XHI
            WRITE(*,*) 'Stopped at X=', XLOW
            CALL CACHESTAT(MINE)
            WRITE(28,*)
            WRITE(28,*) ' MINERAL COMPOSITION'
            WRITE(28,*)
            DO 80 KK=1,NBOUND
              WRITE(28,1100) XBOUND(KK), (MOLMAT(KK,M),M=1,MINE)
 80         CONTINUE
            WRITE(28,*)
            WRITE(28,*) 'X         dX/dt'
            DO 90 KK=1, IN
              WRITE(28,1100) XP(KK), (DXDT(KK,M),M=1,MINE)
 90         CONTINUE
            RECALC=-11
            RETURN
          ENDIF
          LAST=IN
          SIZE=(LAST-FIRST+1)*(COMPON+MINE+1)+COMPON+MINE+4

          IF (OVERWR) THEN
            IF (CPTR(2*ITOWR)-CPTR(2*ITOWR-1)+1.GE.SIZE) THEN
              CALL CACHEWRITE(ITOWR,FIRST,LAST,MINE,XP,YP,DXDT,L,MOLMAT)
              GOTO 10
            ELSE
              CALL FREEBLOCK(ITOWR,CPTR,NBUFF)
            ENDIF
          ENDIF

          ITOWR = MAKEBLOCK (SIZE)
          IF (ITOWR.GT.0) THEN
            CALL CACHEWRITE(ITOWR,FIRST,LAST,MINE,XP,YP,DXDT,L,MOLMAT)
          ELSE
            WRITE(*,*) 'Cannot make block in cache for front ', L
          ENDIF

C         ***** END OF LOOP FOR REACTION FRONTS *****

   10     CONTINUE
          write(28,*) 'Cache hits =',HITS,',misses =',MISSES

          CALL DXWRITE (IN,MINE,XP,DXDT)

 1000 FORMAT(I5,' steps taken and requested output point',E14.7,
     +       ' not  reached.')
 1100 FORMAT(1X,E16.8,10(1X,E10.4))
      XP(IN+1)=0.
C     Value 2 returned indicates success.
      RECALC=2
      RETURN
      END

C***********************************************************************

      SUBROUTINE CHEQUP(MINE,CACHED,OVERWR,NPTR,L,
     +           CDIFF,HITS,XLOW,DXDT,XHI,ITOWR,IFLAG)

C     THIS SUBROUTINE LOOKS IF PROFILE IS STORED IN CACHE

C     CACHE  = VECTOR WITH INFORMATION ABOUT THE PROFILES
C     CACHED = IF PROFILE IS SAVED OR NOT
C     CDIFF  = MAXIMUM RELATIVE DIFFERENCE FOR THE CONCENTRATIONS
C     COMPON = NUMBER OF COMPONENTS
C     CONC   = FREE CONCENTRATION OF THE•COMPONENTS
C     CPTR   = POSITION WHERE BUFFER STARTS
C     DXDT   = MINERAL PRECIPITATION/DISSOLUTION RATE
C     HITS   = NUMBER OF TIME THE PROFILE IN CACHE HAS BEEN USED
C     I      = LOOPING VARIABLE
C     IFLAG  = INDICATES IF THE UPPER BOUNDARY IS REACED
C     II     = LOOPING VARIABLE
C     IN     = NUMBER OF POINTS IN XP AND YP
C     ITOWR  = NUMBER OF THE BUFFER TO DELETE
C     J      = LOOPING VARIABLE FOR THE COMPONENTS
C     L      = PRESENT BOUNDARY
C     M      = LOOPING VARIABLE FOR MINERALS
C     MINE   = NUMBER OF MINERALS
C     MOLMAT = MINERAL DISTRIBUTION
C     NBUFF  = NUMBER OF BUFFERS IN CACHE
C     NCALL  = NUMBER OF TIMES RECALC IS CALLED
C     NPTR   = POSITION IN CACHE
C     NSAVED = NUMBER OF POINTS USED IN CACHE
C     OVERWR = IF PROFILE SHALL BE OVERWRITTEN OR NOT
C     XBOUND = THE BOUNDARY POSITION
C     XHI    = THE UPPER BOUNDARY
C     XLOW   = THE PRESENT POINT
C     XP     = X COORDINATE CHOSEN BY THE PROGRAM
C     YP     = FREE CONCENTRATION OF THE COMPONENTS IN XP

      IMPLICIT         NONE
      LOGICAL          CACHED, OVERWR
      INTEGER          BLOCKS,COMPON,CSIZE,HITS, I, IFLAG, II, IN, ITOWR
      INTEGER          J, L, M, MAXCEL, MAXCOM, MAXMIN, MAXP, MINE
      INTEGER          NBUFF, NCALL, NPTR, NSAVED
      PARAMETER       (BLOCKS=250000, CSIZE=125, MAXMIN=25,
     +                 MAXCEL=MAXMIN*500, MAXCOM=25, MAXP=10000)
      DOUBLE PRECISION CACHE(CSIZE), CDIFF, CONC(MAXCOM), CPTR(2*BLOCKS)
      DOUBLE PRECISION DXDT(MAXP,MAXMIN), MOLMAT(MAXCEL,MAXMIN)
      DOUBLE PRECISION XBOUND(MAXCEL*2), XHI, XLOW, XP(MAXP)
      DOUBLE PRECISION YP(MAXP,MAXCOM)

      COMMON/ONE/COMPON
      COMMON/TEN/MOLMAT
      COMMON/TWELVE/CONC
      COMMON/THIRTEEN/XBOUND
      COMMON/TWENTYONE/XP,YP,IN
      COMMON/TWENTYTWO/NBUFF,CPTR,CACHE
      COMMON/TWENTYTHREE/NCALL

      SAVE /THIRTEEN/, /TWENTYTWO/

      IFLAG=0
C     ***** Perform lookup in the cache buffer. *****
      DO 10 I=1,NBUFF
         CACHED=.TRUE.
         OVERWR=.FALSE.
         NPTR=CPTR(2*I-1)+4

C     ***** First compare mineral assemblages *****
         DO 20 M=1,MINE
            IF ((MOLMAT(L,M).NE.0.0D0.AND.CACHE(NPTR).EQ.0.0D0).OR.
     +          (MOLMAT(L,M).EQ.0.0D0.AND.CACHE(NPTR).NE.0.0D0)
     +          ) CACHED=.FALSE.
            NPTR=NPTR+1
 20      CONTINUE

C     ***** Now compare concentrations at the boundary *****
         IF (CACHED) THEN
            DO 30 J=1,COMPON
               IF (DABS(CONC(J)-CACHE(NPTR)).GT.
     +             CDIFF*(CONC(J)+CACHE(NPTR)) ) THEN
C     ***** Buffer with the same mineral assemblage was found, but
C     inlet concentrations were different. Remember this buffer
C     to overwrite the data and break from the cycle. *****
                  CACHED=.FALSE.
               ENDIF
               NPTR=NPTR+1
 30         CONTINUE
         ENDIF

         IF  (CACHED) THEN
c  number of data points saved
            NPTR=CPTR(2*I-1)
            NSAVED=INT(CACHE(NPTR))
            NPTR=NPTR+1
c  Increase the usage counter and set the last usage of cache block.
            CACHE(NPTR)=DBLE(NCALL)
            NPTR=NPTR+1
```

```
              CACHE(NPTR)=CACHE(NPTR)+1.d0
              NPTR=NPTR+1
c  Skip "when created", min.assemblage and concentration fields.
              NPTR=NPTR+1+MINE+COMPON

C     ***** Everything OK, so copy the data up to the last point in
cache. *****
              DO 40 II=1,NSAVED
                 HITS=HITS+1
                 XLOW=XBOUND(L)+CACHE(NPTR)
                 NPTR=NPTR+1
                 IN=IN+1
                 XP(IN)=XLOW
                 DO 50 J=1,COMPON
                    YP(IN,J)=CACHE(NPTR)
                    NPTR=NPTR+1
 50              CONTINUE
                 DO 60 M=1,MINE
                    DXDT(IN,M)=CACHE(NPTR)
                    NPTR=NPTR+1
 60              CONTINUE
                 IF (XLOW.GE.XHI) THEN
                    XP(IN)=XHI
                    DO 70 J=1,COMPON
                       CONC(J)=YP(IN,J)
 70                 CONTINUE
C     ***** If the next boundary reached, goto the next front. *****
                    IFLAG=1
                    RETURN
                 ENDIF
 40           CONTINUE
C     ***** We get here only if end not reached yet.
C     Drop thru to the solver to complete. *****
              ITOWR=I
              OVERWR=.TRUE.
              RETURN
           ENDIF
 10     CONTINUE

        RETURN
        END

C*************************************************************************

      SUBROUTINE SOLVCALL(XLOW,XHI,HMAX,RECALC,MINE,OLDRE,
     +              DXDT,DDIV,XSTEP,FIRST,MISSES,OLDCON,XOLD,IFLAG)

C      THIS SUBROUTINE CALLS THE SOLVER

C      COMPON = NUMBER OF COMPONENTS
C      CONC   = FREE CONCENTRATION OF THE COMPONENTS
C      DDIV   = MINIMUM RELATIVE DIFFERENCE IN REACTION
C      DIFF   = EXTERNAL SUBROUTINE SENT TO THE SOLVER
C      DMXDT  = MINERAL PRECIPITATION/DISSOLUTION RATE
C      DOFF   = MINIMUM REACTION
C      DXDT   = MINERAL PRECIPITATION/DISSOLUTION RATE
```

```
C       EPS    = REQUESTED RELATIVE ACCURACY FOR THE SOLVER
C       EWT    = MINIMUM CONCENTRATION BEFORE IT IS ZERO
C       FIRST  = FIRST POINT WITHIN A BOUNDARY
C       GAMBAS = ACTIVITY PRODUCT
C       HMAX   = MAXIMUM STEP SIZE
C       IFLAG  = FLAG TO INDICATE RETURN FROM RECALC
C       IN     = NUMBER OF POINTS IN XP AND YP
C       ISFLAG = INDICATION OF CHANGE IN PROFILE
C       IWORK  = SPACE FOR SOLVER TO WORK IN
C       J      = LOOPING VARIABLE FOR THE COMPONENTS
C       LENIW  = LENGTH OF IWORK
C       LENW   = LENGTH OF WORK
C       LL     = LOOPING VARIABLE
C       M      = LOOPING VARIABLE FOR THE MINERALS
C       MAXCAL = MAXIMUM NUMBER OF CALLS
C       MAXCEL = MAXIMUM NUMBER OF CELLS
C       MAXCOM = MAXIMUM NUMBER OF COMPONENTS
C       MAXMIN = MAXIMUM NUMBER OF MINERALS
C       MAXP   = MAXIMUM NUMBER OF POINTS
C       MAXWOR = DIMENSION OF WORK
C       MINE   = NUMBER OF MINERALS
C       MINT   = SOLVING ALGORITHM CHOSED FOR SOLVING THE EQUATIONS
C       MISSES = NUMBER OF TIMES TO WRITE BLOCKS IN CACHE
C       MOLMAT = MINERAL DISTRIBUTION
C       MSTATE = INDICATOR IF THE CALL TO THE SOLVER WAS SUCCESSFUL
C       NROOT  = NUMBER OF EQUATIONS WHOSE ROOT ARE DESIRED
C       OLDCON = CONCENTRATION IN PREVIOUS POINT
C       OLDRE  = REACTION IN PREVIOUS POINT
C       QAQPRD = ION ACTIVITY PRODUCT
C       RECALC = FUNCTION CALLING THIS ROUTINE
C       WORK   = WORKSPACE FOR THE SOLVER
C       XHI    = UPPER BOUNDARY
C       XLOW   = THE COORDINATE FOR THE PRESENT POINT
C       XOLD   = X COORDINATE FOR PREVIOUS POINT
C       XP     = X COORDINATE CHOSEN BY THE SOLVER
C       XSTEP  = MAXIMUM DISTANCE BETWEEN SAVED POINTS
C       YP     = FREE CONCENTRATION OF THE COMPONENTS IN XP

        IMPLICIT          NONE
        INTEGER           COMPON, FIRST, IFLAG, IN, ISFLAG, IWORK(3000), J
        INTEGER           LENIW
        INTEGER           LENW, LL, M, MAXCAL, MAXCEL, MAXCOM, MAXMIN, MAXP
        INTEGER           MAXWOR, MINE, MINT, MISSES, MSTATE, NROOT, RECALC
        PARAMETER        (MAXCAL=5000, MAXCOM=25, MAXMIN=25,
     +                    MAXCEL=MAXMIN*500, MAXP=10000, MAXWOR=10000)
        DOUBLE PRECISION CONC(MAXCOM), DDIV, DIFF, DMXDT(MAXMIN), DOFF
        DOUBLE PRECISION DXDT(MAXP,MAXMIN), EPS, EWT, GAMBAS(MAXCOM), HMAX
        DOUBLE PRECISION MOLMAT(MAXCEL,MAXMIN), OLDCON(MAXCOM)
        DOUBLE PRECISION OLDRE(MAXMIN), QAQPRD(MAXMIN), WORK(MAXWOR), XHI
        DOUBLE PRECISION XLOW, XOLD, XP(MAXP), XSTEP, YP(MAXP,MAXCOM)

        COMMON/ONE/COMPON
        COMMON/TEN/MOLMAT
        COMMON/TWELVE/CONC
        COMMON/EIGHTEEN/DOFF
        COMMON/TWENTY/EPS,EWT
```

```
      COMMON/TWENTYONE/XP,YP,IN

      EXTERNAL DIFF

C     ***** DATA FOR THE SOLVER *****

      MSTATE=-1
      IFLAG=0
      NROOT=0
      MINT=2
      LENW=(2*COMPON+10)*COMPON+2*NROOT+250
      LENIW=COMPON+50

      DO 10 LL=1, MAXCAL
         IF (XLOW.LT.XHI) THEN

C     ***** CALL THE SOLVER *****

            CALL SDRIV2(COMPON,XLOW,CONC,DIFF,XHI,MSTATE,NROOT,EPS,EWT,
     +           MINT,WORK,LENW,IWORK,LENIW,DIFF,HMAX)

            IF (IABS(MSTATE).NE.2) THEN
               WRITE(*,*) 'RETURNED FROM SDRIV2 WITH ERROR FLAG', MSTATE
               RECALC=MSTATE
               IFLAG=1
               RETURN
            ENDIF

C     ***** Calculate the precipitation/dissolution rates along the
column
C     to see where significant reactions take place. *****

            CALL ACTCOEFF(CONC, COMPON, GAMBAS, MAXCOM)
            CALL IONACTPROD(GAMBAS, CONC, QAQPRD)
            CALL PREDISS (QAQPRD, XLOW, DMXDT)

            DO 20 M=1, MINE
               IF (DABS(DMXDT(M)).LT.DOFF) THEN
                  DMXDT(M)=0.0D0
               ENDIF
 20         CONTINUE
C     ***** Look if smth. serious happens. *****
            ISFLAG=0
            DO 30 M=1, MINE
               IF ((DABS(OLDRE(M)).GT.DOFF.AND.DABS(DMXDT(M)).LE.DOFF)
     +            .OR.(DABS(OLDRE(M)).LE.DOFF.AND.DABS(DMXDT(M)).GT.DOFF
     +            ).OR.(OLDRE(M).GT.DOFF.AND.DMXDT(M).LT.-DOFF).OR.
     +            (OLDRE(M).LT.-DOFF.AND.DMXDT(M).GT.DOFF)) THEN

C     ***** IF THE REACTION IS STARTING, STOPPING OR CHANGING SIGN *****
                  ISFLAG=3
                  GOTO 71
               ELSEIF ((DABS(DMXDT(M)-DXDT(IN,M)).GT.DABS(DMXDT(M)+
     +
                  DXDT(IN,M))*DDIV).AND.(DABS(DMXDT(M)).GT.DOFF).AND
     +               .(DABS(DXDT(IN,M)).GT.DOFF)) THEN
```

```
                ISFLAG=2
             ENDIF
 30        CONTINUE

           IF (ISFLAG.EQ.0) THEN
              IF (XLOW-XP(IN).GE.XSTEP) ISFLAG=1
           ENDIF

        IF( ISFLAG.EQ.0.AND.XLOW.EQ.XHI ) THEN
C     ***** Save the last point only if it adds some valuable
information.
C     If it does not, overwrite the last saved. *****
          IF (IN.EQ.FIRST ) THEN
           ISFLAG=1
          ELSE
             ISFLAG=-1
             DO 40 M=1,MINE
           IF ((DMXDT(M).NE.DXDT(IN,M)).OR.(DXDT(IN-1,M).NE.
     +                                                  DXDT(IN,M)))
ISFLAG=1
 40          CONTINUE
          ENDIF
       ENDIF

       IF (ISFLAG.EQ.-1) THEN

C     ***** Replacing the recently saved point
C     /Unnecessary if profile caching is not used./
C     No need to save dxdt, as they are equal anyway. *****

          XP(IN)=XLOW
             DO 50 J=1,COMPON
                YP(IN,J)=CONC(J)
 50          CONTINUE
          ENDIF

          IF (ISFLAG.EQ.1.OR.ISFLAG.EQ.2) THEN
C     ***** Normal save of profile points. *****
             MISSES=MISSES+1
             IN=IN+1
             IF (IN.GT.MAXP) THEN
                WRITE(*,*)
     +  "No storage space for calculated concentration profiles"
                RECALC=-1
                IFLAG=1
                RETURN
             ENDIF
             XP(IN)=XLOW
             DO 60 J=1,COMPON
                YP(IN,J)=CONC(J)
60           CONTINUE
             DO 70 M=1, MINE
                DXDT(IN,M)=DMXDT(M)
70           CONTINUE
          ENDIF

71        IF (ISFLAG.GT.2) THEN
```

```
C     ***** Notice that usually TWO points will be saved, if not
already. *****
                IF (IN+2.GT.MAXP) THEN
                   WRITE(*,*)
     +  "No storage space for calculated concentration profiles"
                   RECALC=-1
                   IFLAG=1
                   RETURN
                ENDIF

                CALL SUCDATA(XOLD,XLOW,OLDCON,MINE,DXDT,
     +               OLDRE,DMXDT,MISSES)

             ENDIF

C     ***** Store the old results so that it is possible to restore them
later. *****
             DO 80 J=1,COMPON
                OLDCON(J)=CONC(J)
 80          CONTINUE
             DO 90 M=1,MINE
                OLDRE(M)=DMXDT(M)
 90          CONTINUE
             XOLD=XLOW
          ENDIF
 10    CONTINUE

       RETURN
       END

C*********************************************************************

       SUBROUTINE SUCDATA(XOLD,XLOW,OLDCON,MINE,DXDT,OLDRE,
     +                    DMXDT,MISSES)

C      THIS SUBROUTINE TAKES CARE OF THE DATA AT A SUCCESSFUL RUN.

C      COMPON = NUMBER OF COMPONENTS
C      CONC   = FREE CONCENTRATION OF THE COMPONENTS
C      DMXDT  = MINERAL PRECIPITATION/DISSOLUTION RATE
C      DXDT   = MINERAL PRECIPITATION/DISSOLUTION RATE
C      IN     = NUMBER OF POINTS IN XP AND YP
C      J      = LOOPING VARIABLE FOR THE COMPONENTS
C      M      = LOOPING VARIABLE FOR THE MINERALS
C      MINE   = NUMBER OF MINERALS
C      MISSES = NUMBER OF TIMES NEW BLOCKS ARE WRITTEN IN CACHE
C      OLDCON = CONCENTRATION IN THE PREVIOUS POINT
C      OLDRE  = MINERAL PRECIPITATION/DISSOLUTION RATE IN PREVIOUS POINT
C      XLOW   = X VALUE FOR PRESENT POINT
C      XOLD   = X VALUE FOR PREVIOUS POINT
C      XP     = VECTOR WITH THE X COORDINATES
C      YP     = VECTOR WITH THE FREE CONCENTRATIONS OF THE COMPONENTS

       IMPLICIT         NONE
       INTEGER          COMPON, IN,J,M,MAXCOM, MAXMIN, MAXP, MINE, MISSES
       PARAMETER       (MAXCOM=25,MAXMIN=25,MAXP=10000)
       DOUBLE PRECISION CONC(MAXCOM), DMXDT(MAXMIN), DXDT(MAXP,MAXMIN)
```

```
      DOUBLE PRECISION OLDCON(MAXCOM), OLDRE(MAXMIN), XLOW, XOLD
      DOUBLE PRECISION XP(MAXP), YP(MAXP,MAXCOM)

      COMMON/ONE/COMPON
      COMMON/TWELVE/CONC
      COMMON/TWENTYONE/XP,YP,IN

      IF (XOLD.GT.XP(IN)) THEN
         XP(IN+1)=XOLD
         XP(IN+2)=XLOW
         DO 10 J=1,COMPON
            YP(IN+1,J)=OLDCON(J)
            YP(IN+2,J)=CONC(J)
 10      CONTINUE
         DO 20 M=1, MINE
            DXDT(IN+1,M)=OLDRE(M)
            DXDT(IN+2,M)=DMXDT(M)
 20      CONTINUE
         IN=IN+2
         MISSES=MISSES+2

      ELSE
C     ***** The previous was already saved. *****
         XP(IN+1)=XLOW
         DO 30 J=1,COMPON
            YP(IN+1,J)=CONC(J)
 30      CONTINUE
         DO 40 M=1, MINE
            DXDT(IN+1,M)=DMXDT(M)
 40      CONTINUE
         IN=IN+1
         MISSES=MISSES+1
      ENDIF

      RETURN
      END

C*********************************************************************

      SUBROUTINE DXWRITE (IN,MINE,XP,DXDT)

C     ***** THIS SUBROUTINE WRITES DXDT INTO THE FILES *****

C     DXDT   = MINERAL DISSOLUTION/PRECIPITATION RATE
C     IN     = NUMBER OF POINTS ON THE DXDT CURVE
C     K      = LOOPING VARIABLE
C     L      = LOOPING VARIABLE
C     M      = LOOPING VARIABLE FOR THE MINERALS
C     MAXMIN = MAXIMUM NUMBER OF MINERALS
C     MAXP   = MAXIMUM NUMBER OF POINTS
C     MINE   = NUMBER OF MINERALS
C     NCALL  = NUMBER OF CALLS
C     XP     = X COORDINATE CHOSEN BY THE PROGRAM

      IMPLICIT          NONE
      INTEGER           IN, L, K, M, MAXMIN, MAXP, MINE, NCALL
      PARAMETER        (MAXMIN=25,MAXP=10000)
      DOUBLE PRECISION DXDT(MAXP,MAXMIN), XP(MAXP)

      COMMON/TWENTYTHREE/NCALL

      IF(MOD(NCALL,500).EQ.1) THEN
         WRITE(23,*)
         WRITE(33,*)
         WRITE(43,*)
         WRITE(23,*) 'X        dX/dt'
         WRITE(33,*) 'X        dX/dt'
         WRITE(43,*) 'X        dX/dt'
         DO 10 L=1, IN
          DO 20 K=1,MINE/10+1
               IF(K.EQ.1) THEN
                  WRITE(23,1100) XP(L),(DXDT(L,M),M=1,MIN(MINE,10))
               ELSEIF(K.EQ.2) THEN
                  WRITE(33,1100) XP(L),(DXDT(L,M),M=11,MIN(MINE,20))
            ELSEIF(K.EQ.3) THEN
                  WRITE(43,1100) XP(L),(DXDT(L,M),M=21,MIN(MINE,30))
            ENDIF
 20         CONTINUE
 10      CONTINUE
         CALL FLUSH(23)
         CALL FLUSH(33)
         CALL FLUSH(43)
      ENDIF

 1100 FORMAT(1X,E16.8,10(1X,E10.4))
      XP(IN+1)=0.

      RETURN
      END
C*************************************************************************

      INTEGER FUNCTION MAKEBLOCK(SIZE)

C     THIS FUNCTION CREATES A NEW BUFFER IN CACHE

C     BLOCKS = DIMENSION OF CPTR
C     CACHE  = VECTOR WITH INFORMATION ABOUT THE PROFILES
C     CPTR   = POSITION WHERE THE BUFFER STARTS
C     CSIZE  = DIMENSION OF CACHE
C     I      = LOOPING VARIABLE
C     INX    = SIZE OF EMPTY BLOCK IN CACHE
C     K      = COUNTING VARIABLE
C     NBUFF  = NUMBER OF BUFFERS
C     NCALL  = NUMBER OF CALLS
C     NFREE  = NUMBER OF EMPTY POSISIONS IN CACHE
C     SIZE   = SIZE OF THE BUFFER TO CREATE
C     START  = FIRST EMPTY POSITION IN BLOCK
C     TOFREE = NUMBER OF BUFFER TO DELETE

      IMPLICIT          NONE
      INTEGER           BLOCKS,CSIZE
      PARAMETER        (BLOCKS=125,CSIZE=250000)
      INTEGER           CPTR(2*BLOCKS), I, INX, K, NBUFF, NCALL, NFREE
      INTEGER           SIZE, START, TOFREE(BLOCKS)
      DOUBLE PRECISION CACHE(CSIZE)

      COMMON/TWENTYTWO/NBUFF,CPTR,CACHE
      COMMON/TWENTYTHREE/NCALL

      SAVE /TWENTYTWO/

      IF (SIZE.GT.CSIZE) THEN
         MAKEBLOCK=0
         RETURN
      ENDIF

C     ***** No problems if cache is empty *****
      IF(NBUFF.EQ.0) THEN
         NBUFF=1
         CPTR(1)=1
         CPTR(2)=SIZE
         MAKEBLOCK=1
         RETURN
      ENDIF

C     ***** Try to find some free chunk of appropriate size, and count
the
C     available free space in cache. *****
      NFREE=0
      DO 10 I=1,NBUFF+1
         IF (I.EQ.1) THEN
            INX=CPTR(1)-1
            START=1
         ELSEIF (I.EQ.NBUFF+1) THEN
            INX=CSIZE-CPTR(2*NBUFF)
            START=CPTR(2*NBUFF)+1
         ELSE
            INX=CPTR(2*I-1)-CPTR(2*(I-1))-1
            START=CPTR(2*(I-1))+1
         ENDIF

         IF (INX.GE.SIZE.AND.NBUFF.LT.BLOCKS) THEN
C     ***** We were lucky to find a chunk of requested size, just
prepare it. *****
            DO 20 K=NBUFF,I,-1
               CPTR(2*K+1)=CPTR(2*K-1)
               CPTR(2*K+2)=CPTR(2*K)
 20         CONTINUE
            CPTR(2*I-1)=START
            CPTR(2*I)=START+SIZE-1
            NBUFF=NBUFF+1
            MAKEBLOCK=I
            RETURN
         ENDIF
         NFREE=NFREE+INX
 10   CONTINUE

C     ***** Can compactification alone help? *****
```

```
      IF (NBUFF.LT.BLOCKS.AND.NFREE.GE.SIZE) THEN
         CALL COMPACT(CACHE, CPTR, NBUFF)
         NBUFF=NBUFF+1
         CPTR(2*NBUFF-1)=CPTR(2*(NBUFF-1))+1
         CPTR(2*NBUFF)=CPTR(2*NBUFF-1)+SIZE-1
         MAKEBLOCK=NBUFF
         RETURN
      ENDIF

C     ***** Now we must delete some profile. *****
      K=0
      DO 30 I=1,NBUFF
         INX=CPTR(2*I)-CPTR(2*I-1)+1
         IF ( ((NFREE.LT.SIZE).OR.(BLOCKS.EQ.NBUFF.AND.K.EQ.0)).AND.
     +        (CACHE(CPTR(2*I-1)+1).LT.NCALL-1) ) THEN
            NFREE=NFREE+INX
            K=K+1
            TOFREE(K)=I
         ENDIF
 30   CONTINUE

      IF (NFREE.GE.SIZE) THEN
         DO 40 I=1,K
            CALL FREEBLOCK(TOFREE(I),CPTR,NBUFF)
 40      CONTINUE
         CALL COMPACT(CACHE, CPTR, NBUFF)
         NBUFF=NBUFF+1
         CPTR(2*NBUFF-1)=CPTR(2*(NBUFF-1))+1
         CPTR(2*NBUFF)=CPTR(2*NBUFF-1)+SIZE-1
         MAKEBLOCK=NBUFF
         RETURN
      ENDIF

C     ***** Profile cannot be saved, return zero as indication of
failure *****
      MAKEBLOCK=0
      RETURN
      END

C*********************************************************************
****
      SUBROUTINE FREEBLOCK(I,CPTR,NBUFF)

C     THIS SUBROUTINE REMOVES AN UNUSFUL BLOCK

C     BLOCKS = DIMENSION OF CPTR
C     CPTR   = POSITION WHERE BUFFER STARTS
C     CSIZE  = DIMENSION OF CACHE
C     I      = NUMBER OF THE BUFFER TO DELETE
C     K      = LOOPING VARIABLE
C     NBUFF  = NUMBER OF BUFFERS

      IMPLICIT    NONE
      INTEGER     BLOCKS, CSIZE
      PARAMETER ( BLOCKS = 125, CSIZE = 250000 )
      INTEGER     CPTR(2*BLOCKS), K, I, NBUFF
      IF (I.LE.0.OR.I.GT.NBUFF) RETURN

      DO 10 K=I,NBUFF-1
         CPTR(2*K-1)=CPTR(2*(K+1)-1)
         CPTR(2*K)=CPTR(2*(K+1))
 10   CONTINUE
      NBUFF=NBUFF-1

      RETURN
      END

C***********************************************************************

         SUBROUTINE COMPACT(CACHE, CPTR, NBUFF)

C     THIS SUBROUTINE COMPACTS CACHE

C     BLOCKS = DIMENSION OF CPTR
C     CACHE  = VECTOR WITH INFORMATION ABOUT THE PROFILES
C     CPTR   = POSITION WHERE BUFFER STARTS
C     CSIZE  = DIMENSION OF CACHE
C     I      = LOOPING VARIABLE
C     IT     = HELP FOR COUNTING
C     ITS    = HELP FOR COUNTING
C     K      = LOOPING VARIABLE
C     NBUFF  = NUMBER OF BUFFERS IN CACHE

      IMPLICIT           NONE
      INTEGER            BLOCKS, CSIZE
      PARAMETER        ( BLOCKS = 125, CSIZE = 250000 )
      DOUBLE PRECISION CACHE(CSIZE)
      INTEGER            CPTR(2*BLOCKS), I, IT, ITS, K, NBUFF

      IT=1
      DO 10 K=1,NBUFF
         IF (IT.NE.CPTR(2*K-1)) THEN
            ITS=IT
            DO 20 I=CPTR(2*K-1),CPTR(2*K)
               CACHE(IT)=CACHE(I)
               IT=IT+1
 20         CONTINUE
            CPTR(2*K-1)=ITS
            CPTR(2*K)=IT-1
         ELSE
            IT=CPTR(2*K)+1
         ENDIF
 10   CONTINUE
      RETURN
      END

C***********************************************************************

      SUBROUTINE CACHEWRITE(ITOWR,START,END,MINE,XP,YP,DXDT,CELL,MOLMAT)

C     THIS SUBROUTIONE WRITES THE INFORMATION ABOUT THE PROFILES IN
C     THE VECTOR CACHE
```

```
C      BLOCKS = DIMENSION OF CPTR
C      CELL   = PRESENT CELL
C      CACHE  = VECTOR WITH INFORMATION ABOUT THE PROFILES
C      COMPON = NUMBER OF COMPONENTS
C      CPTR   = POSITION WHERE BUFFER STARTS
C      CSIZE  = DIMENSION OF CACHE
C      DXDT   = MINERAL PRECIPITATION AND DISSOLUTION RATE
C      END    = LAST POINT OF THE BLOCK
C      I      = LOOPING VARIABLE
C      ITOWR  = NUMBER OF THE BUFFER TO DELETE
C      J      = LOOPING VARIABLE
C      M      = LOOPING VARIABLE
C      MAXCEL = MAXIMUM NUMBER OF CELLS
C      MAXCOM = MAXIMIM NUMBER OF COMPONENTS
C      MAXMIN = MAXIMUM NUMBER OF MINERALS
C      MAXP   = MAXIMUM NUMBER OF POINTS
C      MINE   = NUMBER OF MINERALS
C      MOLMAT = MINERAL COMPOSITION
C      NBUFF  = NUMBER OF BUFFERS IN CACHE
C      NCALL  = NUMBER OF CALLS
C      NPTR   = POSITION IN CACHE
C      START  = FIRST POINT OF THE BLOCK IN XP AND YP
C      XP     = X COORDINATE CHOOSEN BY THE PROGRAM
C      X0     = FIRST X-VALUE
C      YP     = CONCENTRATION OF COMPONENTS AT THE POINTS XP

       IMPLICIT         NONE
       INTEGER          BLOCKS, CSIZE
       INTEGER          MAXCEL,MAXCOM,MAXMIN,MAXP
       PARAMETER       (BLOCKS = 125, CSIZE = 250000 )
       PARAMETER       (MAXCOM=25, MAXMIN=25, MAXP=10000,
     +                  MAXCEL=MAXMIN*500)
       INTEGER          CELL, COMPON, CPTR(2*BLOCKS), END, I, ITOWR, J, M
       INTEGER          MINE, NBUFF, NCALL, NPTR, START
       DOUBLE PRECISION CACHE(CSIZE), DXDT(MAXP,MAXMIN)
       DOUBLE PRECISION MOLMAT(MAXCEL,MAXMIN), XP(MAXP), X0
       DOUBLE PRECISION YP(MAXP,MAXCOM)

       COMMON/ONE/COMPON
       COMMON/TWENTYTWO/NBUFF,CPTR,CACHE
       COMMON/TWENTYTHREE/NCALL

       SAVE /TWENTYTWO/

       IF  (CPTR(2*ITOWR)-CPTR(2*ITOWR-1)+1.LT.
     +    (END-START+1)*(MINE+COMPON+1)+MINE+COMPON+4) THEN
          WRITE(*,*) 'Cache error - data size larger than block.'
          CALL FREEBLOCK (ITOWR,CPTR,NBUFF)
          RETURN
       ENDIF

       NPTR=CPTR(2*ITOWR-1)

C      ***** Number of data points in profile *****
       CACHE(NPTR)=DBLE(END-START+1)
       NPTR=NPTR+1
```

```
C     ***** Timestep when last used *****
      CACHE(NPTR)=DBLE(NCALL)
      NPTR=NPTR+1

C     ***** Number of times used so far *****
      CACHE(NPTR)=1
      NPTR=NPTR+1

C     ***** When created *****
      CACHE(NPTR)=DBLE(NCALL)
      NPTR=NPTR+1

C     ***** Mineral assemblage *****
      DO 10 M=1,MINE
         CACHE(NPTR)=MOLMAT(CELL,M)
         NPTR=NPTR+1
 10   CONTINUE
C     ***** Concentrations at the boundary *****
      DO 20 J=1,COMPON
         CACHE(NPTR)=YP(START,J)
         NPTR=NPTR+1
 20   CONTINUE

C     ***** Now comes the actual profile data *****
      X0=XP(START)
      DO 30 I=START,END
         CACHE(NPTR)=XP(I)-X0
         NPTR=NPTR+1
         DO 40 J=1,COMPON
            CACHE(NPTR)=YP(I,J)
            NPTR=NPTR+1
 40      CONTINUE
         DO 50 M=1,MINE
            CACHE(NPTR)=DXDT(I,M)
            NPTR=NPTR+1
 50      CONTINUE
 30   CONTINUE

      RETURN
      END

C**********************************************************************

      SUBROUTINE CACHESTAT(MINE)

C     THIS SUBROUTINE WRITES STORED INFORMATION ABOUT THE PROFILES
C     IN THE FILE TESTDATA

C     BLOCKS = DIMENSION OF CPTR
C     CACHE  = VECTOR WITH INFORMATION ABOUT THE PROFILES
C     COMPON = NUMBER OF COMPONENTS
C     CPTR   = POSITION WHERE THE BUFFER STARTS
C     CSIZE  = DIMENSION OF CACHE
C     K      = LOOPING VARIABLE
C     L      = LOOPING VARIABLE
```

```
C      M      = LOOPING VARIABLE FOR MINERALS
C      MINE   = NUMBER OF MINERALS
C      NBUFF  = NUMBER OF BUFFERS IN CACHE
C      NPTR   = POSITION IN CACHE
C      NUMOFP = NUMBER OF POINTS IN THE BUFFER

       IMPLICIT          NONE
       INTEGER           BLOCKS, CSIZE
       PARAMETER        (BLOCKS = 125, CSIZE = 250000 )
       INTEGER           COMPON, CPTR(2*BLOCKS), K, L, M, MINE, NBUFF
       INTEGER           NPTR, NUMOFP
       DOUBLE PRECISION CACHE(CSIZE)

       COMMON/ONE/COMPON
       COMMON/TWENTYTWO/NBUFF,CPTR,CACHE

       SAVE /TWENTYTWO/

       WRITE(28,*)
       WRITE(28,*) 'NUMBER OF BLOCKS IN CACHE ', NBUFF
       WRITE(28,*)

       DO 10 L=1,NBUFF
          WRITE (28,*) 'BUFFER ',L
          NPTR=CPTR(2*L-1)
       NUMOFP=INT(CACHE(NPTR))
          WRITE (28,*) 'Number of points in buffer ',NUMOFP
          NPTR=NPTR+1
          WRITE(28,*) 'Last used ',INT(CACHE(NPTR))
          NPTR=NPTR+1
          WRITE(28,*) 'Used ',INT(CACHE(NPTR)),' times'
          NPTR=NPTR+1
          WRITE(28,*) 'Created at timestep ',INT(CACHE(NPTR))
          NPTR=NPTR+1
          DO 20 M=1,MINE
             WRITE(28,*) 'Molmat (',M,') = ',CACHE(NPTR)
             NPTR=NPTR+1
 20       CONTINUE
          DO 30 K=1,NUMOFP
          NPTR=CPTR(2*L-1)+4+MINE+COMPON+(K-1)*(MINE+COMPON+1)
             WRITE(28,1000) CACHE(NPTR),(CACHE(NPTR+COMPON+M),M=1,MINE)
 30      CONTINUE
 10   CONTINUE

      RETURN
 1000 FORMAT(1X,E16.8,8(2X,E12.4))
      END

C*********************************************************************
***
      SUBROUTINE SDRIV2 (N,T,Y,F,TOUT,MSTATE,NROOT,EPS,EWT,MINT,WORK,
     8   LENW,IWORK,LENIW,G,HMAX)
C***BEGIN PROLOGUE  SDRIV2
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  871105   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***KEYWORDS  ODE,STIFF,ORDINARY DIFFERENTIAL EQUATIONS,
C              INITIAL VALUE PROBLEMS,GEAR'S METHOD,
C              DOUBLE PRECISION
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***PURPOSE  The function of SDRIV2 is to solve N ordinary differential
C            equations of the form dY(I)/dT = F(Y(I),T), given the
C            initial conditions Y(I) = YI.  The program has options to
C            allow the solution of both stiff and non-stiff differential
C            equations.  SDRIV2 uses double precision arithmetic.
C***DESCRIPTION
C    From the book "Numerical Methods and Software"
C       by  D. Kahaner, C. Moler, S. Nash
C           Prentice Hall 1988
C
C  I.  ABSTRACT  ..............................................
C
C    The function of SDRIV2 is to solve N ordinary differential
C    equations of the form dY(I)/dT = F(Y(I),T), given the initial
C    conditions Y(I) = YI.  The program has options to allow the
C    solution of both stiff and non-stiff differential equations.
C    SDRIV2 is to be called once for each output point of T.
C
C  II.  PARAMETERS  ...........................................
C
C    The user should use parameter names in the call sequence of SDRIV2
C    for those quantities whose value may be altered by SDRIV2.  The
C    parameters in the call sequence are:
C
C    N      = (Input) The number of differential equations.
C
C    T      = The independent variable.  On input for the first call, T
C             is the initial point.  On output, T is the point at which
C             the solution is given.
C
C    Y      = The vector of dependent variables.  Y is used as input on
C             the first call, to set the initial values.  On output, Y
C             is the computed solution vector.  This array Y is passed
C             in the call sequence of the user-provided routines F and
C             G.  Thus parameters required by F and G can be stored in
C             this array in components N+1 and above.  (Note: Changes
C             by the user to the first N components of this array will
C             take effect only after a restart, i.e., after setting
C             MSTATE to +1(-1).)
C
C    F      = A subroutine supplied by the user.  The name must be
C             declared EXTERNAL in the user's calling program.  This
C             subroutine is of the form:
C                   SUBROUTINE F (N, T, Y, YDOT)
C                   REAL*8 Y(*), YDOT(*)
C                     .
C                     .
C                   YDOT(1) = ...
C                     .
C                     .
C                   YDOT(N) = ...
C                   END (Sample)
C             This computes YDOT = F(Y,T), the right hand side of the
```

```
C              differential equations.  Here Y is a vector of length at
C              least N.  The actual length of Y is determined by the
C              user's declaration in the program which calls SDRIV2.
C              Thus the dimensioning of Y in F, while required by FORTRAN
C              convention, does not actually allocate any storage.  When
C              this subroutine is called, the first N components of Y are
C              intermediate approximations to the solution components.
C              The user should not alter these values.  Here YDOT is a
C              vector of length N.  The user should only compute YDOT(I)
C              for I from 1 to N.  Normally a return from F passes
C              control back to  SDRIV2.  However, if the user would like
C              to abort the calculation, i.e., return control to the
C              program which calls SDRIV2, he should set N to zero.
C              SDRIV2 will signal this by returning a value of MSTATE
C              equal to +6(-6).  Altering the value of N in F has no
C              effect on the value of N in the call sequence of SDRIV2.
C
C    TOUT   = (Input) The point at which the solution is desired.
C
C    MSTATE = An integer describing the status of integration.  The user
C             must initialize MSTATE to +1 or -1.  If MSTATE is
C             positive, the routine will integrate past TOUT and
C             interpolate the solution.  This is the most efficient
C             mode.  If MSTATE is negative, the routine will adjust its
C             internal step to reach TOUT exactly (useful if a
C             singularity exists beyond TOUT.)  The meaning of the
C             magnitude of MSTATE:
C               1  (Input) Means the first call to the routine.  This
C                  value must be set by the user.  On all subsequent
C                  calls the value of MSTATE should be tested by the
C                  user.  Unless SDRIV2 is to be reinitialized, only the
C                  sign of MSTATE may be changed by the user.  (As a
C                  convenience to the user who may wish to put out the
C                  initial conditions, SDRIV2 can be called with
C                  MSTATE=+1(-1), and TOUT=T.  In this case the program
C                  will return with MSTATE unchanged, i.e.,
C                  MSTATE=+1(-1).)
C               2  (Output) Means a successful integration.  If a normal
C                  continuation is desired (i.e., a further integration
C                  in the same direction), simply advance TOUT and call
C                  again.  All other parameters are automatically set.
C               3  (Output)(Unsuccessful) Means the integrator has taken
C                  1000 steps without reaching TOUT.  The user can
C                  continue the integration by simply calling SDRIV2
C                  again.  Other than an error in problem setup, the
C                  most likely cause for this condition is trying to
C                  integrate a stiff set of equations with the non-stiff
C                  integrator option. (See description of MINT below.)
C               4  (Output)(Unsuccessful) Means too much accuracy has
C                  been requested.  EPS has been increased to a value
C                  the program estimates is appropriate.  The user can
C                  continue the integration by simply calling SDRIV2
C                  again.
C               5  (Output) A root was found at a point less than TOUT.
C                  The user can continue the integration toward TOUT by
C                  simply calling SDRIV2 again.
```

```
C                   6  (Output)(Unsuccessful) N has been set to zero in
C                      SUBROUTINE F.
C                   7  (Output)(Unsuccessful) N has been set to zero in
C                      FUNCTION G.  See description of G below.
C
C    NROOT  = (Input) The number of equations whose roots are desired.
C             If NROOT is zero, the root search is not active.  This
C             option is useful for obtaining output at points which are
C             not known in advance, but depend upon the solution, e.g.,
C             when some solution component takes on a specified value.
C             The root search is carried out using the user-written
C             function G (see description of G below.)  SDRIV2 attempts
C             to find the value of T at which one of the equations
C             changes sign.  SDRIV2 can find at most one root per
C             equation per internal integration step, and will then
C             return the solution either at TOUT or at a root, whichever
C             occurs first in the direction of integration.  The index
C             of the equation whose root is being reported is stored in
C             the sixth element of IWORK.
C             NOTE: NROOT is never altered by this program.
C
C    EPS    = On input, the requested relative accuracy in all solution
C             components.  EPS = 0 is allowed.  On output, the adjusted
C             relative accuracy if the input value was too small.  The
C             value of EPS should be set as large as is reasonable,
C             because the amount of work done by SDRIV2 increases as
C             EPS decreases.
C
C    EWT    = (Input) Problem zero, i.e., the smallest physically
C             meaningful value for the solution.  This is used inter-
C             nally to compute an array YWT(I) = DMAX1(DABS(Y(I)), EWT).
C             One step error estimates divided by YWT(I) are kept less
C             than EPS.  Setting EWT to zero provides pure relative
C             error control.  However, setting EWT smaller than
C             necessary can adversely affect the running time.
C
C    MINT   = (Input) The integration method flag.
C               MINT = 1  Means the Adams methods, and is used for
C                         non-stiff problems.
C               MINT = 2  Means the stiff methods of Gear (i.e., the
C                         backward differentiation formulas), and is
C                         used for stiff problems.
C               MINT = 3  Means the program dynamically selects the
C                         Adams methods when the problem is non-stiff
C                         and the Gear methods when the problem is
C                         stiff.
C             MINT may not be changed without restarting, i.e., setting
C             the magnitude of MSTATE to 1.
C
C    WORK
C    LENW   = (Input)
C             WORK is an array of LENW real words used
C             internally for temporary storage.  The user must allocate
C             space for this array in the calling program by a statement
C             such as
C                       REAL*8 WORK(...)
C             The length of WORK should be at least
```

```
 C              16*N + 2*NROOT + 204          if MINT is 1, or
 C              N*N + 10*N + 2*NROOT + 204    if MINT is 2, or
 C              N*N + 17*N + 2*NROOT + 204    if MINT is 3,
 C            and LENW should be set to the value used.  The contents of
 C            WORK should not be disturbed between calls to SDRIV2.
 C
 C    IWORK
 C    LENIW  = (Input)
 C            IWORK is an integer array of length LENIW used internally
 C            for temporary storage.  The user must allocate space for
 C            this array in the calling program by a statement such as
 C                      INTEGER IWORK(...)
 C            The length of IWORK should be at least
 C              21       if MINT is 1, or
 C              N+21     if MINT is 2 or 3,
 C            and LENIW should be set to the value used.  The contents
 C            of IWORK should not be disturbed between calls to SDRIV2.
 C
 C    G      = A real FORTRAN function supplied by the user
 C            if NROOT is not 0.  In this case, the name must be
 C            declared EXTERNAL in the user's calling program.  G is
 C            repeatedly called with different values of IROOT to
 C            obtain the value of each of the NROOT equations for which
 C            a root is desired.  G is of the form:
 C                  REAL*8 FUNCTION G (N, T, Y, IROOT)
 C                  REAL*8 Y(*)
 C                  GO TO (10, ...), IROOT
 C             10   G = ...
 C                    .
 C                    .
 C                  END (Sample)
 C            Here, Y is a vector of length at least N, whose first N
 C            components are the solution components at the point T.
 C            The user should not alter these values.  The actual length
 C            of Y is determined by the user's declaration in the
 C            program which calls SDRIV2.  Thus the dimensioning of Y in
 C            G, while required by FORTRAN convention, does not actually
 C            allocate any storage.  Normally a return from G passes
 C            control back to  SDRIV2.  However, if the user would like
 C            to abort the calculation, i.e., return control to the
 C            program which calls SDRIV2, he should set N to zero.
 C            SDRIV2 will signal this by returning a value of MSTATE
 C            equal to +7(-7).  In this case, the index of the equation
 C            being evaluated is stored in the sixth element of IWORK.
 C            Altering the value of N in G has no effect on the value of
 C            N in the call sequence of SDRIV2.
 C
 C    HMAX    /Vidvuds/The maximum magnitude of the step size that will
be
C             used for the problem. This is useful for ensuring that
C             important details are not missed. If this is not the case,
C             a large value, such as the interval length, may be used.
C
C***LONG DESCRIPTION
C
C  III.  OTHER COMMUNICATION TO THE USER  .............................
C
```

```
C     A. The solver communicates to the user through the parameters
C        above.  In addition it writes diagnostic messages through the
C        standard error handling program XERROR.  That program will
C        terminate the user's run if it detects a probable problem setup
C        error, e.g., insufficient storage allocated by the user for the
C        WORK array.  Messages are written on the standard error message
C        file.  At installations which have this error handling package
C        the user should determine the standard error handling file from
C        the local documentation.  Otherwise the short but serviceable
C        routine, XERROR, available with this package, can be used.  That
C        program writes on logical unit 6 to transmit messages.  A
C        complete description of XERROR is given in the Sandia
C        Laboratories report SAND78-1189 by R. E. Jones.
C
C     B. The first three elements of WORK and the first five elements of
C        IWORK will contain the following statistical data:
C          AVGH     The average step size used.
C          HUSED    The step size last used (successfully).
C          AVGORD   The average order used.
C          IMXERR   The index of the element of the solution vector that
C                   contributed most to the last error test.
C          NQUSED   The order last used (successfully).
C          NSTEP    The number of steps taken since last initialization.
C          NFE      The number of evaluations of the right hand side.
C          NJE      The number of evaluations of the Jacobian matrix.
C
C  IV.  REMARKS  ..................................................
C
C     A. On any return from SDRIV2 all information necessary to continue
C        the calculation is contained in the call sequence parameters,
C        including the work arrays.  Thus it is possible to suspend one
C        problem, integrate another, and then return to the first.
C
C     B. If this package is to be used in an overlay situation, the user
C        must declare in the primary overlay the variables in the call
C        sequence to SDRIV2.
C
C     C. When the routine G is not required, difficulties associated with
C        an unsatisfied external can be avoided by using the name of the
C        routine which calculates the right hand side of the differential
C        equations in place of G in the call sequence of SDRIV2.
C
C  V.  USAGE  .....................................................
C
C               PROGRAM SAMPLE
C               EXTERNAL F
C               PARAMETER(MINT = 1, NROOT = 0, N = ...,
C              8          LENW = 16*N + 2*NROOT + 204, LENIW = 21)
C                                           N is the number of equations
C               REAL*8 EPS, EWT, T, TOUT, WORK(LENW), Y(N)
C               INTEGER IWORK(LENIW)
C               OPEN(FILE='TAPE6', UNIT=6, STATUS='NEW')
C               T = 0.                           Initial point
C               DO 10 I = 1,N
C          10     Y(I) = ...                     Set initial conditions
C               TOUT = T
```

```
C                 EWT = ...
C                 MSTATE = 1
C                 EPS = ...
C           20    CALL SDRIV2 (N, T, Y, F, TOUT, MSTATE, NROOT, EPS, EWT,
C               8             MINT, WORK, LENW, IWORK, LENIW, F)
C                                            Last argument is not the same
C                                            as F if rootfinding is used.
C                 IF (MSTATE .GT. 2) STOP
C                 WRITE(6, 100) TOUT, (Y(I), I=1,N)
C                 TOUT = TOUT + 1.
C                 IF (TOUT .LE. 10.) GO TO 20
C          100    FORMAT(...)
C                 END (Sample)
C
C***REFERENCES  GEAR, C. W., "NUMERICAL INITIAL VALUE PROBLEMS IN
C                 ORDINARY DIFFERENTIAL EQUATIONS", PRENTICE-HALL, 1971.
C***ROUTINES CALLED  SDRIV3,XERROR
C***END PROLOGUE  SDRIV2
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      EXTERNAL F, G, FA
      DIMENSION EWTCOM(1), WORK(*), Y(*)
      INTEGER IWORK(*)
      CHARACTER MSG*81
      PARAMETER(IMPL = 1, MXSTEP = 1000)
C***FIRST EXECUTABLE STATEMENT  SDRIV2
      IF (MINT .LT. 1 .OR. MINT .GT. 3) THEN
        WRITE(MSG, '(''SDRIV21FE Illegal input.  Improper value for '',
     8  ''the integration method flag,'', I8)') MINT
        CALL XERROR(MSG(1:81), 81, 21, 2)
        RETURN
      END IF
      IF (MSTATE .GE. 0) THEN
        NSTATE = MSTATE
        NTASK = 1
      ELSE
        NSTATE = - MSTATE
        NTASK = 2
      END IF
      EWTCOM(1) = EWT
      IF (EWT .NE. 0.E0) THEN
        IERROR = 3
      ELSE
        IERROR = 2
      END IF
      IF (MINT .EQ. 1) THEN
        MITER = 0
        MXORD = 12
      ELSE IF (MINT .EQ. 2) THEN
        MITER = 2
        MXORD = 5
      ELSE IF (MINT .EQ. 3) THEN
        MITER = 2
        MXORD = 12
      END IF
      CALL SDRIV3 (N, T, Y, F, NSTATE, TOUT, NTASK, NROOT, EPS, EWTCOM,
     8             IERROR, MINT, MITER, IMPL, ML, MU, MXORD, HMAX, WORK,
     8             LENW, IWORK, LENIW, F, FA, NDE, MXSTEP, G, F)
```

```
      IF (MSTATE .GE. 0) THEN
        MSTATE = NSTATE
      ELSE
        MSTATE = - NSTATE
      END IF
      END
      SUBROUTINE SDCOR (DFDY,EL,FA,H,IMPL,IPVT,MATDIM,MITER,ML,MU,N,
     8   NDE,NQ,T,USERS,Y,YH,YWT,EVALFA,SAVE1,SAVE2,A,D,JSTATE)
C***BEGIN PROLOGUE  SDCOR
C***REFER TO  SDRIV3
C  Subroutine SDCOR is called to compute corrections to the Y array.
C  In the case of functional iteration, update Y directly from the
C  result of the last call to F.
C  In the case of the chord method, compute the corrector error and
C  solve the linear system with that as right hand side and DFDY as
C  coefficient matrix, using the LU decomposition if MITER is 1, 2, 4,
C  or 5.
C***ROUTINES CALLED  SGESL,SGBSL,SNRM2
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870401   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDCOR
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(MATDIM,*), DFDY(MATDIM,*), EL(13,12),
     +          SAVE1(*), SAVE2(*), Y(*), YH(N,*), YWT(*)
      INTEGER IPVT(*)
      LOGICAL EVALFA
C***FIRST EXECUTABLE STATEMENT  SDCOR
      IF (MITER .EQ. 0) THEN
        DO 100 I = 1,N
 100      SAVE1(I) = (H*SAVE2(I) - YH(I,2) - SAVE1(I))/YWT(I)
        D = SNRM2(N, SAVE1, 1)/DSQRT(DBLE(N))
        DO 105 I = 1,N
 105      SAVE1(I) = H*SAVE2(I) - YH(I,2)
      ELSE IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
        IF (IMPL .EQ. 0) THEN
          DO 130 I = 1,N
 130        SAVE2(I) = H*SAVE2(I) - YH(I,2) - SAVE1(I)
        ELSE IF (IMPL .EQ. 1) THEN
          IF (EVALFA) THEN
            CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
            IF (N .EQ. 0) THEN
              JSTATE = 9
              RETURN
            END IF
          ELSE
            EVALFA = .TRUE.
          END IF
          DO 150 I = 1,N
 150        SAVE2(I) = H*SAVE2(I)
          DO 160 J = 1,N
            DO 160 I = 1,N
 160          SAVE2(I) = SAVE2(I) - A(I,J)*(YH(J,2) + SAVE1(J))
        ELSE IF (IMPL .EQ. 2) THEN
          IF (EVALFA) THEN
```

```
              CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
              IF (N .EQ. 0) THEN
                JSTATE = 9
                RETURN
              END IF
            ELSE
              EVALFA = .TRUE.
            END IF
            DO 180 I = 1,N
 180          SAVE2(I) = H*SAVE2(I) - A(I,1)*(YH(I,2) + SAVE1(I))
          END IF
          CALL SGESL (DFDY, MATDIM, N, IPVT, SAVE2, 0)
          DO 200 I = 1,N
            SAVE1(I) = SAVE1(I) + SAVE2(I)
 200        SAVE2(I) = SAVE2(I)/YWT(I)
          D = SNRM2(N, SAVE2, 1)/DSQRT(DBLE(N))
        ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
          IF (IMPL .EQ. 0) THEN
            DO 230 I = 1,N
 230          SAVE2(I) = H*SAVE2(I) - YH(I,2) - SAVE1(I)
          ELSE IF (IMPL .EQ. 1) THEN
            IF (EVALFA) THEN
              CALL FA (N, T, Y, A(ML+1,1), MATDIM, ML, MU, NDE)
              IF (N .EQ. 0) THEN
                JSTATE = 9
                RETURN
              END IF
            ELSE
              EVALFA = .TRUE.
            END IF
            DO 250 I = 1,N
 250          SAVE2(I) = H*SAVE2(I)
            MW = ML + 1 + MU
            DO 260 J = 1,N
              I1 = MAX0(ML+1, MW+1-J)
              I2 = MIN0(MW+N-J, MW+ML)
              DO 260 I = I1,I2
                I3 = I + J - MW
 260            SAVE2(I3) = SAVE2(I3) - A(I,J)*(YH(J,2) + SAVE1(J))
          ELSE IF (IMPL .EQ. 2) THEN
            IF (EVALFA) THEN
              CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
              IF (N .EQ. 0) THEN
                JSTATE = 9
                RETURN
              END IF
            ELSE
              EVALFA = .TRUE.
            END IF
            DO 280 I = 1,N
 280          SAVE2(I) = H*SAVE2(I) - A(I,1)*(YH(I,2) + SAVE1(I))
          END IF
          CALL SGBSL (DFDY, MATDIM, N, ML, MU, IPVT, SAVE2, 0)
          DO 300 I = 1,N
            SAVE1(I) = SAVE1(I) + SAVE2(I)
 300        SAVE2(I) = SAVE2(I)/YWT(I)
```

```
          D = SNRM2(N, SAVE2, 1)/DSQRT(DBLE(N))
        ELSE IF (MITER .EQ. 3) THEN
          IFLAG = 2
          CALL USERS (Y, YH(1,2), YWT, SAVE1, SAVE2, T, H, EL(1,NQ), IMPL,
     8                N, NDE, IFLAG)
          IF (N .EQ. 0) THEN
            JSTATE = 10
            RETURN
          END IF
          DO 320 I = 1,N
            SAVE1(I) = SAVE1(I) + SAVE2(I)
 320        SAVE2(I) = SAVE2(I)/YWT(I)
          D = SNRM2(N, SAVE2, 1)/DSQRT(DBLE(N))
        END IF
        END
        SUBROUTINE SDCST (MAXORD,MINT,ISWFLG,EL,TQ)
C***BEGIN PROLOGUE  SDCST
C***REFER TO  SDRIV3
C  SDCST is called by SDNTL and sets coefficients used by the core
C  integrator SDSTP.  The array EL determines the basic method.
C  The array TQ is involved in adjusting the step size in relation
C  to truncation error.  EL and TQ depend upon MINT, and are calculated
C  for orders 1 to MAXORD(.LE. 12).  For each order NQ, the coefficients
C  EL are calculated from the generating polynomial:
C    L(T) = EL(1,NQ) + EL(2,NQ)*T + ... + EL(NQ+1,NQ)*T**NQ.
C  For the implicit Adams methods, L(T) is given by
C    dL/dT = (1+T)*(2+T)* ... *(NQ-1+T)/K,   L(-1) = 0,
C    where      K = factorial(NQ-1).
C  For the Gear methods,
C    L(T) = (1+T)*(2+T)* ... *(NQ+T)/K,
C    where      K = factorial(NQ)*(1 + 1/2 + ... + 1/NQ).
C  For each order NQ, there are three components of TQ.
C***ROUTINES CALLED  (NONE)
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870216   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDCST
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION EL(13,12), FACTRL(12), GAMMA(14), TQ(3,12)
C***FIRST EXECUTABLE STATEMENT  SDCST
      FACTRL(1) = 1.E0
      DO 10 I = 2,MAXORD
 10     FACTRL(I) = DBLE(I)*FACTRL(I-1)
C                                                COMPUTE ADAMS COEFFICIENTS
      IF (MINT .EQ. 1) THEN
        GAMMA(1) = 1.E0
        DO 40 I = 1,MAXORD+1
          SUM = 0.E0
          DO 30 J = 1,I
 30         SUM = SUM - GAMMA(J)/DBLE(I-J+2)
 40       GAMMA(I+1) = SUM
        EL(1,1) = 1.E0
        EL(2,1) = 1.E0
        EL(2,2) = 1.E0
        EL(3,2) = 1.E0
        DO 60 J = 3,MAXORD
          EL(2,J) = FACTRL(J-1)
          DO 50 I = 3,J
 50         EL(I,J) = DBLE(J-1)*EL(I,J-1) + EL(I-1,J-1)
 60       EL(J+1,J) = 1.E0
        DO 80 J = 2,MAXORD
          EL(1,J) = EL(1,J-1) + GAMMA(J)
          EL(2,J) = 1.E0
          DO 80 I = 3,J+1
 80         EL(I,J) = EL(I,J)/(DBLE(I-1)*FACTRL(J-1))
        DO 100 J = 1,MAXORD
          TQ(1,J) = -1.E0/(FACTRL(J)*GAMMA(J))
          TQ(2,J) = -1.E0/GAMMA(J+1)
 100      TQ(3,J) = -1.E0/GAMMA(J+2)
C                                              COMPUTE GEAR COEFFICIENTS
      ELSE IF (MINT .EQ. 2) THEN
        EL(1,1) = 1.E0
        EL(2,1) = 1.E0
        DO 130 J = 2,MAXORD
          EL(1,J) = FACTRL(J)
          DO 120 I = 2,J
 120        EL(I,J) = DBLE(J)*EL(I,J-1) + EL(I-1,J-1)
 130      EL(J+1,J) = 1.E0
        SUM = 1.E0
        DO 150 J = 2,MAXORD
          SUM = SUM + 1.E0/DBLE(J)
          DO 150 I = 1,J+1
 150        EL(I,J) = EL(I,J)/(FACTRL(J)*SUM)
        DO 170 J = 1,MAXORD
          IF (J .GT. 1) TQ(1,J) = 1.E0/FACTRL(J-1)
          TQ(2,J) = DBLE(J+1)/EL(1,J)
 170      TQ(3,J) = DBLE(J+2)/EL(1,J)
      END IF
C                             Compute constants used in the stiffness test.
C                             These are the ratio of TQ(2,NQ) for the Gear
C                             methods to those for the Adams methods.
      IF (ISWFLG .EQ. 3) THEN
        MXRD = MIN0(MAXORD, 5)
        IF (MINT .EQ. 2) THEN
          GAMMA(1) = 1.E0
          DO 190 I = 1,MXRD
            SUM = 0.E0
            DO 180 J = 1,I
 180          SUM = SUM - GAMMA(J)/DBLE(I-J+2)
 190        GAMMA(I+1) = SUM
        END IF
        SUM = 1.E0
        DO 200 I = 2,MXRD
          SUM = SUM + 1.E0/DBLE(I)
 200      EL(1+I,1) = -DBLE(I+1)*SUM*GAMMA(I+1)
      END IF
      END
      SUBROUTINE SDNTL (EPS,F,FA,HMAX,HOLD,IMPL,JTASK,MATDIM,MAXORD,
     8   MINT,MITER,ML,MU,N,NDE,SAVE1,T,UROUND,USERS,Y,YWT,H,MNTOLD,
     8   MTROLD,NFE,RC,YH,A,CONVRG,EL,FAC,IER,IPVT,NQ,NWAIT,RH,RMAX,
     8   SAVE2,TQ,TREND,ISWFLG,JSTATE)
C***BEGIN PROLOGUE  SDNTL
```

```
C***REFER TO  SDRIV3
C  Subroutine SDNTL is called to set parameters on the first call
C  to SDSTP, on an internal restart, or when the user has altered
C  MINT, MITER, and/or H.
C  On the first call, the order is set to 1 and the initial derivatives
C  are calculated.  RMAX is the maximum ratio by which H can be
C  increased in one step.  It is initially RMINIT to compensate
C  for the small initial H, but then is normally equal to RMNORM.
C  If a failure occurs (in corrector convergence or error test), RMAX
C  is set at RMFAIL for the next increase.
C  If the caller has changed MINT, or if JTASK = 0, SDCST is called
C  to set the coefficients of the method.  If the caller has changed H,
C  YH must be rescaled.  If H or MINT has been changed, NWAIT is
C  reset to NQ + 2 to prevent further increases in H for that many
C  steps.  Also, RC is reset.  RC is the ratio of new to old values of
C  the coefficient L(0)*H.  If the caller has changed MITER, RC is
C  set to 0 to force the partials to be updated, if partials are used.
C***ROUTINES CALLED  SDCST,SDSCL,SGEFA,SGESL,SGBFA,SGBSL,SNRM2
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870810   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDNTL
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(MATDIM,*), EL(13,12), FAC(*), SAVE1(*), SAVE2(*),
     8     TQ(3,12), Y(*), YH(N,*), YWT(*)
      INTEGER IPVT(*)
      LOGICAL CONVRG, IER
      PARAMETER(RMINIT = 10000.E0)
C***FIRST EXECUTABLE STATEMENT  SDNTL
      IER = .FALSE.
      IF (JTASK .GE. 0) THEN
        IF (JTASK .EQ. 0) THEN
          CALL SDCST (MAXORD, MINT, ISWFLG,  EL, TQ)
          RMAX = RMINIT
        END IF
        RC = 0.E0
        CONVRG = .FALSE.
        TREND = 1.E0
        NQ = 1
        NWAIT = 3
        CALL F (N, T, Y, SAVE2)
        IF (N .EQ. 0) THEN
          JSTATE = 6
          RETURN
        END IF
        NFE = NFE + 1
        IF (IMPL .NE. 0) THEN
          IF (MITER .EQ. 3) THEN
            IFLAG = 0
            CALL USERS (Y, YH, YWT, SAVE1, SAVE2, T, H, EL, IMPL, N,
     8                  NDE, IFLAG)
            IF (N .EQ. 0) THEN
              JSTATE = 10
              RETURN
          END IF
        ELSE IF (IMPL .EQ. 1) THEN
          IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
            CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
            IF (N .EQ. 0) THEN
              JSTATE = 9
              RETURN
            END IF
            CALL SGEFA (A, MATDIM, N, IPVT, INFO)
            IF (INFO .NE. 0) THEN
              IER = .TRUE.
              RETURN
            END IF
            CALL SGESL (A, MATDIM, N, IPVT, SAVE2, 0)
          ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
            CALL FA (N, T, Y, A(ML+1,1), MATDIM, ML, MU, NDE)
            IF (N .EQ. 0) THEN
              JSTATE = 9
              RETURN
            END IF
            CALL SGBFA (A, MATDIM, N, ML, MU, IPVT, INFO)
            IF (INFO .NE. 0) THEN
              IER = .TRUE.
              RETURN
            END IF
            CALL SGBSL (A, MATDIM, N, ML, MU, IPVT, SAVE2, 0)
          END IF
        ELSE IF (IMPL .EQ. 2) THEN
          CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
          IF (N .EQ. 0) THEN
            JSTATE = 9
            RETURN
          END IF
          DO 150 I = 1,NDE
            IF (A(I,1) .EQ. 0.E0) THEN
              IER = .TRUE.
              RETURN
            ELSE
              SAVE2(I) = SAVE2(I)/A(I,1)
            END IF
 150        CONTINUE
          DO 155 I = NDE+1,N
 155        A(I,1) = 0.E0
        END IF
      END IF
      DO 170 I = 1,NDE
 170    SAVE1(I) = SAVE2(I)/YWT(I)
      SUM = SNRM2(NDE, SAVE1, 1)
      SUM0 = 1.E0/DMAX1(1.D0, DABS(T))
      SMAX = DMAX1(SUM0, SUM)
      SMIN = DMIN1(SUM0, SUM)
      SUM = SMAX*DSQRT(1.E0 + (SMIN/SMAX)**2)/DSQRT(DBLE(NDE))
      H = DSIGN(DMIN1(2.E0*EPS/SUM, DABS(H)), H)
      DO 180 I = 1,N
 180    YH(I,2) = H*SAVE2(I)
      IF (MITER .EQ. 2 .OR. MITER .EQ. 5 .OR. ISWFLG .EQ. 3) THEN
        DO 20 I = 1,N
```

```
   20         FAC(I) = DSQRT(UROUND)
          END IF
        ELSE
          IF (MITER .NE. MTROLD) THEN
            MTROLD = MITER
            RC = 0.E0
            CONVRG = .FALSE.
          END IF
          IF (MINT .NE. MNTOLD) THEN
            MNTOLD = MINT
            OLDL0 = EL(1,NQ)
            CALL SDCST (MAXORD, MINT, ISWFLG,  EL, TQ)
            RC = RC*EL(1,NQ)/OLDL0
            NWAIT = NQ + 2
          END IF
          IF (H .NE. HOLD) THEN
            NWAIT = NQ + 2
            RH = H/HOLD
            CALL SDSCL (HMAX, N, NQ, RMAX,  HOLD, RC, RH, YH)
          END IF
        END IF
        END
        SUBROUTINE SDNTP (H,K,N,NQ,T,TOUT,YH,Y)
C***BEGIN PROLOGUE  SDNTP
C***REFER TO  SDRIV3
C   Subroutine SDNTP interpolates the K-th derivative of Y at TOUT,
C   using the data in the YH array.  If K has a value greater than NQ,
C   the NQ-th derivative is calculated.
C***ROUTINES CALLED  (NONE)
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870216   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDNTP
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION Y(*), YH(N,*)
C***FIRST EXECUTABLE STATEMENT  SDNTP
      IF (K .EQ. 0) THEN
        DO 10 I = 1,N
 10       Y(I) = YH(I,NQ+1)
        R = ((TOUT - T)/H)
        DO 20 JJ = 1,NQ
          J = NQ + 1 - JJ
          DO 20 I = 1,N
 20         Y(I) = YH(I,J) + R*Y(I)
      ELSE
        KUSED = MIN0(K, NQ)
        FACTOR = 1.E0
        DO 40 KK = 1,KUSED
 40       FACTOR = FACTOR*DBLE(NQ+1-KK)
        DO 50 I = 1,N
 50       Y(I) = FACTOR*YH(I,NQ+1)
        DO 80 JJ = KUSED+1,NQ
          J = K + 1 + NQ - JJ
          FACTOR = 1.E0
          DO 60 KK = 1,KUSED
 60          FACTOR = FACTOR*DBLE(J-KK)
           DO 70 I = 1,N
 70          Y(I) = FACTOR*YH(I,J) + R*Y(I)
 80        CONTINUE
         DO 100 I = 1,N
 100       Y(I) = Y(I)*H**(-KUSED)
       END IF
       END
       SUBROUTINE SDPSC (KSGN,N,NQ,YH)
C***BEGIN PROLOGUE  SDPSC
C***REFER TO  SDRIV3
C     This subroutine computes the predicted YH values by effectively
C     multiplying the YH array by the Pascal triangle matrix when KSGN
C     is +1, and performs the inverse function when KSGN is -1.
C***ROUTINES CALLED  (NONE)
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  841119   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDPSC
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION YH(N,*)
C***FIRST EXECUTABLE STATEMENT  SDPSC
      IF (KSGN .GT. 0) THEN
        DO 10 J1 = 1,NQ
          DO 10 J2 = J1,NQ
            J = NQ - J2 + J1
            DO 10 I = 1,N
 10           YH(I,J) = YH(I,J) + YH(I,J+1)
      ELSE
        DO 30 J1 = 1,NQ
          DO 30 J2 = J1,NQ
            J = NQ - J2 + J1
            DO 30 I = 1,N
 30           YH(I,J) = YH(I,J) - YH(I,J+1)
      END IF
      END
      SUBROUTINE SDPST (EL,F,FA,H,IMPL,JACOBN,MATDIM,MITER,ML,MU,N,NDE,
     8   NQ,SAVE2,T,USERS,Y,YH,YWT,UROUND,NFE,NJE,A,DFDY,FAC,IER,IPVT,
     8   SAVE1,ISWFLG,BND,JSTATE)
C***BEGIN PROLOGUE  SDPST
C***REFER TO  SDRIV3
C  Subroutine SDPST is called to reevaluate the partials.
C  If MITER is 1, 2, 4, or 5, the matrix
C  P = I - L(0)*H*Jacobian is stored in DFDY and subjected to LU
C  decomposition, with the results also stored in DFDY.
C***ROUTINES CALLED  SGEFA,SGBFA,SNRM2
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870401   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDPST
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(MATDIM,*), DFDY(MATDIM,*), EL(13,12), FAC(*),
     8        SAVE1(*), SAVE2(*), Y(*), YH(N,*), YWT(*)
      INTEGER IPVT(*)
      LOGICAL IER
      PARAMETER(FACMAX = .5E0)
C***FIRST EXECUTABLE STATEMENT  SDPST
      do lk=1,7
      fac(lk)=fac(lk)+lk*1.0e-9
      enddo
      NJE = NJE + 1
      IER = .FALSE.
      IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
        IF (MITER .EQ. 1) THEN
          CALL JACOBN (N, T, Y, DFDY, MATDIM, ML, MU)
          IF (N .EQ. 0) THEN
            JSTATE = 8
            RETURN
          END IF
          IF (ISWFLG .EQ. 3) BND = SNRM2(N*N, DFDY, 1)
          FACTOR = -EL(1,NQ)*H
          DO 110 J = 1,N
            DO 110 I = 1,N
 110          DFDY(I,J) = FACTOR*DFDY(I,J)
        ELSE IF (MITER .EQ. 2) THEN
          BR = UROUND**(.875E0)
          BL = UROUND**(.75E0)
          BU = UROUND**(.25E0)
          BP = UROUND**(-.15E0)
          FACMIN = UROUND**(.78E0)
          DO 170 J = 1,N
            YS = DMAX1(DABS(YWT(J)), DABS(Y(J)))
 120        DY = FAC(J)*YS
            IF (DY .EQ. 0.E0) THEN
              IF (FAC(J) .LT. FACMAX) THEN
                FAC(J) = DMIN1(100.E0*FAC(J), FACMAX)
                GO TO 120
              ELSE
                DY = YS
              END IF
            END IF
            IF (NQ .EQ. 1) THEN
              DY = DSIGN(DY, SAVE2(J))
            ELSE
              DY = DSIGN(DY, YH(J,3))
            END IF
            DY = (Y(J) + DY) - Y(J)
            YJ = Y(J)
            Y(J) = Y(J) + DY
            CALL F (N, T, Y, SAVE1)
            IF (N .EQ. 0) THEN
              JSTATE = 6
              RETURN
            END IF
            Y(J) = YJ
            FACTOR = -EL(1,NQ)*H/DY
            DO 140 I = 1,N
 140          DFDY(I,J) = (SAVE1(I) - SAVE2(I))*FACTOR
C
```

```
          DIFF = DABS(SAVE2(1) - SAVE1(1))
          IMAX = 1
          DO 150 I = 2,N
            IF (DABS(SAVE2(I) - SAVE1(I)) .GT. DIFF) THEN
              IMAX = I
              DIFF = DABS(SAVE2(I) - SAVE1(I))
            END IF
 150        CONTINUE
C                                                                   Step 2
        IF (DMIN1(DABS(SAVE2(IMAX)), DABS(SAVE1(IMAX))) .GT. 0.E0)
     +        THEN
            SCALE = DMAX1(DABS(SAVE2(IMAX)), DABS(SAVE1(IMAX)))
C                                                                   Step 3
            IF (DIFF .GT. BU*SCALE) THEN
              FAC(J) = DMAX1(FACMIN, FAC(J)*.1E0)
            ELSE IF (BR*SCALE .LE. DIFF .AND. DIFF .LE. BL*SCALE) THEN
              FAC(J) = DMIN1(FAC(J)*10.E0, FACMAX)
C                                                                   Step 4
            ELSE IF (DIFF .LT. BR*SCALE) THEN
              FAC(J) = DMIN1(BP*FAC(J), FACMAX)
            END IF
          END IF
 170      CONTINUE
        IF (ISWFLG .EQ. 3) BND = SNRM2(N*N, DFDY, 1)/(-EL(1,NQ)*H)
        NFE = NFE + N
      END IF
      IF (IMPL .EQ. 0) THEN
        DO 190 I = 1,N
 190      DFDY(I,I) = DFDY(I,I) + 1.E0
      ELSE IF (IMPL .EQ. 1) THEN
        CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
        IF (N .EQ. 0) THEN
          JSTATE = 9
          RETURN
        END IF
        DO 210 J = 1,N
          DO 210 I = 1,N
 210        DFDY(I,J) = DFDY(I,J) + A(I,J)
      ELSE IF (IMPL .EQ. 2) THEN
        CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
        IF (N .EQ. 0) THEN
          JSTATE = 9
          RETURN
        END IF
        DO 230 I = 1,NDE
 230      DFDY(I,I) = DFDY(I,I) + A(I,1)
      END IF
      CALL SGEFA (DFDY, MATDIM, N, IPVT, INFO)
      IF (INFO .NE. 0) IER = .TRUE.
    ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
      IF (MITER .EQ. 4) THEN
        CALL JACOBN (N, T, Y, DFDY(ML+1,1), MATDIM, ML, MU)
        IF (N .EQ. 0) THEN
          JSTATE = 8
          RETURN
        END IF
        FACTOR = -EL(1,NQ)*H
```

```
            MW = ML + MU + 1
            DO 260 J = 1,N
              I1 = MAX(ML+1, MW+1-J)
              I2 = MIN(MW+N-J, MW+ML)
              DO 260 I = I1,I2
 260            DFDY(I,J) = FACTOR*DFDY(I,J)
          ELSE IF (MITER .EQ. 5) THEN
            BR = UROUND**(.875E0)
            BL = UROUND**(.75E0)
            BU = UROUND**(.25E0)
            BP = UROUND**(-.15E0)
            FACMIN = UROUND**(.78E0)
            MW = ML + MU + 1
            J2 = MIN(MW, N)
            DO 340 J = 1,J2
              DO 290 K = J,N,MW
                YS = DMAX1(DABS(YWT(K)), DABS(Y(K)))
 280            DY = FAC(K)*YS
                IF (DY .EQ. 0.E0) THEN
                  IF (FAC(K) .LT. FACMAX) THEN
                    FAC(K) = DMIN1(100.E0*FAC(K), FACMAX)
                    GO TO 280
                  ELSE
                    DY = YS
                  END IF
                END IF
                IF (NQ .EQ. 1) THEN
                  DY = DSIGN(DY, SAVE2(K))
                ELSE
                  DY = DSIGN(DY, YH(K,3))
                END IF
                DY = (Y(K) + DY) - Y(K)
                DFDY(MW,K) = Y(K)
 290            Y(K) = Y(K) + DY
              CALL F (N, T, Y, SAVE1)
              IF (N .EQ. 0) THEN
                JSTATE = 6
                RETURN
              END IF
              DO 330 K = J,N,MW
                Y(K) = DFDY(MW,K)
                YS = DMAX1(DABS(YWT(K)), DABS(Y(K)))
                DY = FAC(K)*YS
                IF (DY .EQ. 0.E0) DY = YS
                IF (NQ .EQ. 1) THEN
                  DY = DSIGN(DY, SAVE2(K))
                ELSE
                  DY = DSIGN(DY, YH(K,3))
                END IF
                DY = (Y(K) + DY) - Y(K)
                FACTOR = -EL(1,NQ)*H/DY
                I1 = MAX(ML+1, MW+1-K)
                I2 = MIN(MW+N-K, MW+ML)
                DO 300 I = I1,I2
                  I3 = K + I - MW
 300              DFDY(I,K) = FACTOR*(SAVE1(I3) - SAVE2(I3))
```

```
C                                                                     Step 1
                IMAX = MAX(1, K - MU)
                DIFF = DABS(SAVE2(IMAX) - SAVE1(IMAX))
                I1 = IMAX
                I2 = MIN(K + ML, N)
                DO 310 I = I1+1,I2
                  IF (DABS(SAVE2(I) - SAVE1(I)) .GT. DIFF) THEN
                    IMAX = I
                    DIFF = DABS(SAVE2(I) - SAVE1(I))
                  END IF
 310              CONTINUE
C                                                                     Step 2
             IF (DMIN1(DABS(SAVE2(IMAX)), DABS(SAVE1(IMAX))) .GT.0.E0)
     +            THEN
                  SCALE = DMAX1(DABS(SAVE2(IMAX)), DABS(SAVE1(IMAX)))
C                                                                     Step 3
                  IF (DIFF .GT. BU*SCALE) THEN
                    FAC(K) = DMAX1(FACMIN, FAC(K)*.1E0)
                  ELSE IF (BR*SCALE .LE.DIFF .AND. DIFF .LE.BL*SCALE) THEN
                    FAC(K) = DMIN1(FAC(K)*10.E0, FACMAX)
C                                                                     Step 4
                  ELSE IF (DIFF .LT. BR*SCALE) THEN
                    FAC(K) = DMIN1(BP*FAC(K), FACMAX)
                  END IF
                END IF
 330            CONTINUE
 340          CONTINUE
            NFE = NFE + J2
          END IF
          IF (ISWFLG .EQ. 3) THEN
            DFDYMX = 0.E0
            DO 345 J = 1,N
              I1 = MAX(ML+1, MW+1-J)
              I2 = MIN(MW+N-J, MW+ML)
              DO 345 I = I1,I2
 345            DFDYMX = DMAX1(DFDYMX, DABS(DFDY(I,J)))
            BND = 0.E0
            IF (DFDYMX .NE. 0.E0) THEN
              DO 350 J = 1,N
                I1 = MAX(ML+1, MW+1-J)
                I2 = MIN(MW+N-J, MW+ML)
                DO 350 I = I1,I2
 350              BND = BND + (DFDY(I,J)/DFDYMX)**2
              BND = DFDYMX*DSQRT(BND)/(-EL(1,NQ)*H)
            END IF
          END IF
          IF (IMPL .EQ. 0) THEN
            DO 360 J = 1,N
 360          DFDY(MW,J) = DFDY(MW,J) + 1.E0
          ELSE IF (IMPL .EQ. 1) THEN
            CALL FA (N, T, Y, A(ML+1,1), MATDIM, ML, MU, NDE)
            IF (N .EQ. 0) THEN
              JSTATE = 9
              RETURN
            END IF
            DO 380 J = 1,N
              I1 = MAX(ML+1, MW+1-J)
          I2 = MIN(MW+N-J, MW+ML)
          DO 380 I = I1,I2
 380        DFDY(I,J) = DFDY(I,J) + A(I,J)
        ELSE IF (IMPL .EQ. 2) THEN
          CALL FA (N, T, Y, A, MATDIM, ML, MU, NDE)
          IF (N .EQ. 0) THEN
            JSTATE = 9
            RETURN
          END IF
          DO 400 J = 1,NDE
 400        DFDY(MW,J) =  DFDY(MW,J) + A(J,1)
        END IF
        CALL SGBFA (DFDY, MATDIM, N, ML, MU, IPVT, INFO)
        IF (INFO .NE. 0) IER = .TRUE.
      ELSE IF (MITER .EQ. 3) THEN
        IFLAG = 1
        CALL USERS (Y, YH(1,2), YWT, SAVE1, SAVE2, T, H, EL(1,NQ), IMPL,
     8              N, NDE, IFLAG)
        IF (N .EQ. 0) THEN
          JSTATE = 10
          RETURN
        END IF
      END IF
      END

      SUBROUTINE SDSCL (HMAX,N,NQ,RMAX,H,RC,RH,YH)
C***BEGIN PROLOGUE  SDSCL
C***REFER TO  SDRIV3
C   This subroutine rescales the YH array whenever the step size
C   is changed.
C***ROUTINES CALLED  (NONE)
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  850319   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDSCL
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION  YH(N,*)
C***FIRST EXECUTABLE STATEMENT  SDSCL
      IF (H .LT. 1.E0) THEN
        RH = DMIN1(DABS(H)*RH, DABS(H)*RMAX, HMAX)/DABS(H)
      ELSE
        RH = DMIN1(RH, RMAX, HMAX/DABS(H))
      END IF
      R1 = 1.E0
      DO 10 J = 1,NQ
        R1 = R1*RH
        DO 10 I = 1,N
 10       YH(I,J+1) = YH(I,J+1)*R1
      H = H*RH
      RC = RC*RH
      END
      SUBROUTINE SDSTP (EPS,F,FA,HMAX,IMPL,JACOBN,MATDIM,MAXORD,MINT,
     8   MITER,ML,MU,N,NDE,YWT,UROUND,USERS,AVGH,AVGORD,H,HUSED,JTASK,
     8   MNTOLD,MTROLD,NFE,NJE,NQUSED,NSTEP,T,Y,YH,A,CONVRG,DFDY,EL,FAC,
     8   HOLD,IPVT,JSTATE,NQ,NWAIT,RC,RMAX,SAVE1,SAVE2,TQ,TREND,ISWFLG,
     8    MTRSV,MXRDSV)
C***BEGIN PROLOGUE  SDSTP
C***REFER TO  SDRIV3
C  SDSTP performs one step of the integration of an initial value
C  problem for a system of ordinary differential equations.
C  Communication with SDSTP is done with the following variables:
C
C    YH      An N by MAXORD+1 array containing the dependent variables
C              and their scaled derivatives.  MAXORD, the maximum order
C              used, is currently 12 for the Adams methods and 5 for the
C              Gear methods.  YH(I,J+1) contains the J-th derivative of
C              Y(I), scaled by H**J/factorial(J).  Only Y(I),
C              1 .LE. I .LE. N, need be set by the calling program on
C              the first entry.  The YH array should not be altered by
C              the calling program.  When referencing YH as a
C              2-dimensional array, use a column length of N, as this is
C              the value used in SDSTP.
C    DFDY    A block of locations used for partial derivatives if MITER
C              is not 0.  If MITER is 1 or 2 its length must be at least
C              N*N.  If MITER is 4 or 5 its length must be at least
C              (2*ML+MU+1)*N.
C    YWT     An array of N locations used in convergence and error tests
C    SAVE1
C    SAVE2   Arrays of length N used for temporary storage.
C    IPVT    An integer array of length N used by the linear system
C              solvers for the storage of row interchange information.
C    A       A block of locations used to store the matrix A, when using
C              the implicit method.  If IMPL is 1, A is a MATDIM by N
C              array.  If MITER is 1 or 2 MATDIM is N, and if MITER is 4
C              or 5 MATDIM is 2*ML+MU+1.  If IMPL is 2 its length is N.
C    JTASK   An integer used on input.
C              It has the following values and meanings:
C                 .EQ. 0  Perform the first step.  This value enables
C                         the subroutine to initialize itself.
C                .GT. 0  Take a new step continuing from the last.
C                         Assumes the last step was successful and
C                         user has not changed any parameters.
C                 .LT. 0  Take a new step with a new value of H and/or
C                         MINT and/or MITER.
C    JSTATE  A completion code with the following meanings:
C                1  The step was successful.
C                2  A solution could not be obtained with H .NE. 0.
C                3  A solution was not obtained in MXTRY attempts.
C                4  For IMPL .NE. 0, the matrix A is singular.
C             On a return with JSTATE .GT. 1, the values of T and
C             the YH array are as of the beginning of the last
C             step, and H is the last step size attempted.
C***ROUTINES CALLED  SDNTL,SDPST,SDCOR,SDPSC,SDSCL,SNRM2
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870810   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDSTP
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      EXTERNAL F, JACOBN, FA, USERS
      REAL*8 NUMER
      DIMENSION A(MATDIM,*), DFDY(MATDIM,*), EL(13,12), FAC(*),
     8     SAVE1(*), SAVE2(*), TQ(3,12), Y(*), YH(N,*), YWT(*)
      INTEGER IPVT(*)
      LOGICAL CONVRG, EVALFA, EVALJC, IER, SWITCH
      PARAMETER(BIAS1 = 1.3E0, BIAS2 = 1.2E0, BIAS3 = 1.4E0, MXFAIL = 3,
     8          MXITER = 3, MXTRY = 50, RCTEST = .3E0, RMFAIL = 2.E0,
     8          RMNORM = 10.E0, TRSHLD = 1.E0)
      DATA IER /.FALSE./
C***FIRST EXECUTABLE STATEMENT  SDSTP
      NSV = N
      BND = 0.E0
      SWITCH = .FALSE.
      NTRY = 0
      TOLD = T
      NFAIL = 0
      IF (JTASK .LE. 0) THEN
        CALL SDNTL (EPS, F, FA, HMAX, HOLD, IMPL, JTASK, MATDIM,
     8              MAXORD, MINT, MITER, ML, MU, N, NDE, SAVE1, T,
     8              UROUND, USERS, Y, YWT,  H, MNTOLD, MTROLD, NFE, RC,
     8              YH,  A, CONVRG, EL, FAC, IER, IPVT, NQ, NWAIT, RH,
     8              RMAX, SAVE2, TQ, TREND, ISWFLG, JSTATE)
        IF (N .EQ. 0) GO TO 440
        IF (H .EQ. 0.E0) GO TO 400
        IF (IER) GO TO 420
      END IF
 100  NTRY = NTRY + 1
      IF (NTRY .GT. MXTRY) GO TO 410
      T = T + H
      CALL SDPSC (1, N, NQ,  YH)
      EVALJC = ((DABS(RC - 1.E0) .GT. RCTEST) .AND. (MITER .NE. 0))
      EVALFA = .NOT. EVALJC
C
 110  ITER = 0
      DO 115 I = 1,N
 115    Y(I) = YH(I,1)
      CALL F (N, T, Y, SAVE2)
      IF (N .EQ. 0) THEN
        JSTATE = 6
        GO TO 430
      END IF
      NFE = NFE + 1
      IF (EVALJC .OR. IER) THEN
        CALL SDPST (EL, F, FA, H, IMPL, JACOBN, MATDIM, MITER, ML,
     8              MU, N, NDE, NQ, SAVE2, T, USERS, Y, YH, YWT, UROUND,
     8              NFE, NJE,  A, DFDY, FAC, IER, IPVT, SAVE1, ISWFLG,
     8              BND, JSTATE)
        IF (N .EQ. 0) GO TO 430
        IF (IER) GO TO 160
        CONVRG = .FALSE.
        RC = 1.E0
      END IF
      DO 125 I = 1,N
 125    SAVE1(I) = 0.E0
C                       Up to MXITER corrector iterations are taken.
C                       Convergence is tested by requiring the r.m.s.
C                       norm of changes to be less than EPS.  The sum of
```

```
C                         the corrections is accumulated in the vector
C                         SAVE1(I).  It is approximately equal to the L-th
C                         derivative of Y multiplied by
C                         H**L/(factorial(L-1)*EL(L,NQ)), and is thus
C                         proportional to the actual errors to the lowest
C                         power of H present (H**L).  The YH array is not
C                         altered in the correction loop.  The norm of the
C                         iterate difference is stored in D.  If
C                         ITER .GT. 0, an estimate of the convergence rate
C                         constant is stored in TREND, and this is used in
C                         the convergence test.
C
 130  CALL SDCOR (DFDY, EL, FA, H, IMPL, IPVT, MATDIM, MITER, ML,
     8             MU, N, NDE, NQ, T, USERS, Y, YH, YWT,  EVALFA, SAVE1,
     8             SAVE2,  A, D, JSTATE)
        IF (N .EQ. 0) GO TO 430
      IF (ISWFLG .EQ. 3 .AND. MINT .EQ. 1) THEN
        IF (ITER .EQ. 0) THEN
          NUMER = SNRM2(N, SAVE1, 1)
          DO 132 I = 1,N
 132        DFDY(1,I) = SAVE1(I)
          Y0NRM = SNRM2(N, YH, 1)
        ELSE
          DENOM = NUMER
          DO 134 I = 1,N
 134        DFDY(1,I) = SAVE1(I) - DFDY(1,I)
          NUMER = SNRM2(N, DFDY, MATDIM)
          IF (EL(1,NQ)*NUMER .LE. 100.E0*UROUND*Y0NRM) THEN
            IF (RMAX .EQ. RMFAIL) THEN
              SWITCH = .TRUE.
              GO TO 170
            END IF
          END IF
          DO 136 I = 1,N
 136        DFDY(1,I) = SAVE1(I)
          IF (DENOM .NE. 0.E0)
     8    BND = DMAX1(BND, NUMER/(DENOM*DABS(H)*EL(1,NQ)))
        END IF
      END IF
      IF (ITER .GT. 0) TREND = DMAX1(.9E0*TREND, D/D1)
      D1 = D
      CTEST = DMIN1(2.D0*TREND, 1.D0)*D
      IF (CTEST .LE. EPS) GO TO 170
      ITER = ITER + 1
      IF (ITER .LT. MXITER) THEN
        DO 140 I = 1,N
 140      Y(I) = YH(I,1) + EL(1,NQ)*SAVE1(I)
        CALL F (N, T, Y, SAVE2)
        IF (N .EQ. 0) THEN
          JSTATE = 6
          GO TO 430
        END IF
        NFE = NFE + 1
        GO TO 130
      END IF
C                         The corrector iteration failed to converge in
C                         MXITER tries.  If partials are involved but are
```

```
C                       not up to date, they are reevaluated for the next
C                       try.  Otherwise the YH array is retracted to its
C                       values before prediction, and H is reduced, if
C                       possible.  If not, a no-convergence exit is taken.
      IF (CONVRG) THEN
        EVALJC = .TRUE.
        EVALFA = .FALSE.
        GO TO 110
      END IF
 160  T = TOLD
      CALL SDPSC (-1, N, NQ,  YH)
      NWAIT = NQ + 2
      IF (JTASK .NE. 0 .AND. JTASK .NE. 2) RMAX = RMFAIL
      IF (ITER .EQ. 0) THEN
        RH = .3E0
      ELSE
        RH = .9E0*(EPS/CTEST)**(.2E0)
      END IF
      IF (RH*H .EQ. 0.E0) GO TO 400
      CALL SDSCL (HMAX, N, NQ, RMAX,  H, RC, RH, YH)
      GO TO 100
C                            The corrector has converged.  CONVRG is set
C                            to .TRUE. if partial derivatives were used,
C                            to indicate that they may need updating on
C                            subsequent steps.  The error test is made.
 170  CONVRG = (MITER .NE. 0)
      DO 180 I = 1,NDE
 180    SAVE2(I) = SAVE1(I)/YWT(I)
      ETEST = SNRM2(NDE, SAVE2, 1)/(TQ(2,NQ)*DSQRT(DBLE(NDE)))
C
C                            The error test failed.  NFAIL keeps track of
C                            multiple failures.  Restore T and the YH
C                            array to their previous values, and prepare
C                            to try the step again.  Compute the optimum
C                            step size for this or one lower order.
      IF (ETEST .GT. EPS) THEN
        T = TOLD
        CALL SDPSC (-1, N, NQ,  YH)
        NFAIL = NFAIL + 1
        IF (NFAIL .LT. MXFAIL) THEN
          IF (JTASK .NE. 0 .AND. JTASK .NE. 2) RMAX = RMFAIL
          RH2 = 1.D0/(BIAS2*(ETEST/EPS)**(1.D0/DBLE(NQ+1)))
          IF (NQ .GT. 1) THEN
            DO 190 I = 1,NDE
 190          SAVE2(I) = YH(I,NQ+1)/YWT(I)
            ERDN = SNRM2(NDE, SAVE2, 1)/(TQ(1,NQ)*DSQRT(DBLE(NDE)))
            RH1 = 1.D0/DMAX1(1.D0, BIAS1*(ERDN/EPS)**(1.D0/DBLE(NQ)))
            IF (RH2 .LT. RH1) THEN
              NQ = NQ - 1
              RC = RC*EL(1,NQ)/EL(1,NQ+1)
              RH = RH1
            ELSE
              RH = RH2
            END IF
          ELSE
            RH = RH2
          END IF
          NWAIT = NQ + 2
          IF (RH*H .EQ. 0.E0) GO TO 400
          CALL SDSCL (HMAX, N, NQ, RMAX,  H, RC, RH, YH)
          GO TO 100
        END IF
C                  Control reaches this section if the error test has
C                  failed MXFAIL or more times.  It is assumed that the
C                  derivatives that have accumulated in the YH array have
C                  errors of the wrong order.  Hence the first derivative
C                  is recomputed, the order is set to 1, and the step is
C                  retried.
        NFAIL = 0
        JTASK = 2
        DO 215 I = 1,N
 215      Y(I) = YH(I,1)
        CALL SDNTL (EPS, F, FA, HMAX, HOLD, IMPL, JTASK, MATDIM,
     8              MAXORD, MINT, MITER, ML, MU, N, NDE, SAVE1, T,
     8              UROUND, USERS, Y, YWT,  H, MNTOLD, MTROLD, NFE, RC,
     8              YH,  A, CONVRG, EL, FAC, IER, IPVT, NQ, NWAIT, RH,
     8              RMAX, SAVE2, TQ, TREND, ISWFLG, JSTATE)
        RMAX = RMNORM
        IF (N .EQ. 0) GO TO 440
        IF (H .EQ. 0.E0) GO TO 400
        IF (IER) GO TO 420
        GO TO 100
      END IF
C                            After a successful step, update the YH array.
      NSTEP = NSTEP + 1
      HUSED = H
      NQUSED = NQ
      AVGH =  (DBLE(NSTEP-1)*AVGH + H)/DBLE(NSTEP)
      AVGORD =  (DBLE(NSTEP-1)*AVGORD + DBLE(NQ))/DBLE(NSTEP)
      DO 230 J = 1,NQ+1
        DO 230 I = 1,N
 230      YH(I,J) = YH(I,J)  + EL(J,NQ)*SAVE1(I)
      DO 235 I = 1,N
 235    Y(I) = YH(I,1)
C                                             If ISWFLG is 3, consider
C                                             changing integration methods.
      IF (ISWFLG .EQ. 3) THEN
        IF (BND .NE. 0.E0) THEN
          IF (MINT .EQ. 1 .AND. NQ .LE. 5) THEN
            HN = DABS(H)/DMAX1(UROUND,  (ETEST/EPS)**(1.E0/DBLE(NQ+1)))
            HN = DMIN1(HN,  1.E0/(2.E0*EL(1,NQ)*BND))
            HS = DABS(H)/DMAX1(UROUND,
     8      (ETEST/(EPS*EL(NQ+1,1)))**(1.E0/DBLE(NQ+1)))
            IF (HS .GT. 1.2E0*HN) THEN
              MINT = 2
              MNTOLD = MINT
              MITER = MTRSV
              MTROLD = MITER
              MAXORD = MIN(MXRDSV, 5)
              RC = 0.E0
              RMAX = RMNORM
              TREND = 1.E0
              CALL SDCST (MAXORD, MINT, ISWFLG, EL, TQ)
              NWAIT = NQ + 2
            END IF
          ELSE IF (MINT .EQ. 2) THEN
            HS = DABS(H)/DMAX1(UROUND,  (ETEST/EPS)**(1.E0/DBLE(NQ+1)))
            HN = DABS(H)/DMAX1(UROUND,
     8      (ETEST*EL(NQ+1,1)/EPS)**(1.E0/DBLE(NQ+1)))
            HN = DMIN1(HN, 1.E0/(2.E0*EL(1,NQ)*BND))
            IF (HN .GE. HS) THEN
              MINT = 1
              MNTOLD = MINT
              MITER = 0
              MTROLD = MITER
              MAXORD = MIN(MXRDSV, 12)
              RMAX = RMNORM
              TREND = 1.E0
              CONVRG = .FALSE.
              CALL SDCST (MAXORD, MINT, ISWFLG, EL, TQ)
              NWAIT = NQ + 2
            END IF
          END IF
        END IF
      END IF
      IF (SWITCH) THEN
        MINT = 2
        MNTOLD = MINT
        MITER = MTRSV
        MTROLD = MITER
        MAXORD = MIN(MXRDSV, 5)
        NQ = MIN(NQ, MAXORD)
        RC = 0.E0
        RMAX = RMNORM
        TREND = 1.E0
        CALL SDCST (MAXORD, MINT, ISWFLG, EL, TQ)
        NWAIT = NQ + 2
      END IF
C                               Consider changing H if NWAIT = 1.  Otherwise
C                               decrease NWAIT by 1.  If NWAIT is then 1 and
C                               NQ.LT.MAXORD, then SAVE1 is saved for use in
C                               a possible order increase on the next step.
C
      IF (JTASK .EQ. 0 .OR. JTASK .EQ. 2) THEN
        RH = 1.E0/DMAX1(UROUND, BIAS2*(ETEST/EPS)**(1.E0/DBLE(NQ+1)))
        IF (RH.GT.TRSHLD) CALL SDSCL (HMAX, N, NQ, RMAX, H, RC, RH, YH)
      ELSE IF (NWAIT .GT. 1) THEN
        NWAIT = NWAIT - 1
        IF (NWAIT .EQ. 1 .AND. NQ .LT. MAXORD) THEN
          DO 250 I = 1,NDE
 250        YH(I,MAXORD+1) = SAVE1(I)
        END IF
C             If a change in H is considered, an increase or decrease in
C             order by one is considered also.  A change in H is made
C             only if it is by a factor of at least TRSHLD.  Factors
C             RH1, RH2, and RH3 are computed, by which H could be
C             multiplied at order NQ - 1, order NQ, or order NQ + 1,
C             respectively.  The largest of these is determined and the
C             new order chosen accordingly.  If the order is to be
C             increased, we compute one additional scaled derivative.
```

```
C             If there is a change of order, reset NQ and the
C             coefficients.  In any case H is reset according to RH and
C             the YH array is rescaled.
      ELSE
        IF (NQ .EQ. 1) THEN
          RH1 = 0.E0
        ELSE
          DO 270 I = 1,NDE
 270        SAVE2(I) = YH(I,NQ+1)/YWT(I)
          ERDN = SNRM2(NDE, SAVE2, 1)/(TQ(1,NQ)*DSQRT(DBLE(NDE)))
          RH1 = 1.E0/DMAX1(UROUND, BIAS1*(ERDN/EPS)**(1.E0/DBLE(NQ)))
        END IF
        RH2 = 1.E0/DMAX1(UROUND, BIAS2*(ETEST/EPS)**(1.E0/DBLE(NQ+1)))
        IF (NQ .EQ. MAXORD) THEN
          RH3 = 0.E0
        ELSE
          DO 290 I = 1,NDE
 290        SAVE2(I) = (SAVE1(I) - YH(I,MAXORD+1))/YWT(I)
          ERUP = SNRM2(NDE, SAVE2, 1)/(TQ(3,NQ)*DSQRT(DBLE(NDE)))
          RH3 = 1.E0/DMAX1(UROUND, BIAS3*(ERUP/EPS)**(1.E0/DBLE(NQ+2)))
        END IF
        IF (RH1 .GT. RH2 .AND. RH1 .GE. RH3) THEN
          RH = RH1
          IF (RH .LE. TRSHLD) GO TO 380
          NQ = NQ - 1
          RC = RC*EL(1,NQ)/EL(1,NQ+1)
        ELSE IF (RH2 .GE. RH1 .AND. RH2 .GE. RH3) THEN
          RH = RH2
          IF (RH .LE. TRSHLD) GO TO 380
        ELSE
          RH = RH3
          IF (RH .LE. TRSHLD) GO TO 380
          DO 360 I = 1,N
 360        YH(I,NQ+2) = SAVE1(I)*EL(NQ+1,NQ)/DBLE(NQ+1)
          NQ = NQ + 1
          RC = RC*EL(1,NQ)/EL(1,NQ-1)
        END IF
        IF (ISWFLG .EQ. 3 .AND. MINT .EQ. 1) THEN
        IF (BND.NE.0.E0) RH = DMIN1(RH, 1.E0/(2.E0*EL(1,NQ)*BND*
     +      DABS(H)))
        END IF
        CALL SDSCL (HMAX, N, NQ, RMAX,  H, RC, RH, YH)
        RMAX = RMNORM
 380    NWAIT = NQ + 2
      END IF
C             All returns are made through this section.  H is saved
C             in HOLD to allow the caller to change H on the next step
      JSTATE = 1
      HOLD = H
      RETURN
C
 400  JSTATE = 2
      HOLD = H
      DO 405 I = 1,N
 405    Y(I) = YH(I,1)
      RETURN
```

```
C
 410  JSTATE = 3
      HOLD = H
      RETURN
C
 420  JSTATE = 4
      HOLD = H
      RETURN
C
 430  T = TOLD
      CALL SDPSC (-1, NSV, NQ,  YH)
      DO 435 I = 1,NSV
 435    Y(I) = YH(I,1)
 440  HOLD = H
      RETURN
      END
      SUBROUTINE SDZRO (AE,F,H,N,NQ,IROOT,RE,T,YH,UROUND,B,C,FB,FC,Y)
C***BEGIN PROLOGUE  SDZRO
C***REFER TO  SDRIV3
C     This is a special purpose version of ZEROIN, modified for use with
C     the SDRIV1 package.
C
C     Sandia Mathematical Program Library
C     Mathematical Computing Services Division 5422
C     Sandia Laboratories
C     P. O. Box 5800
C     Albuquerque, New Mexico  87115
C     Control Data 6600 Version 4.5, 1 November 1971
C
C     DABSTRACT
C        ZEROIN searches for a zero of a function F(N, T, Y, IROOT)
C        between the given values B and C until the width of the
C        interval (B, C) has collapsed to within a tolerance specified
C        by the stopping criterion, DABS(B - C) .LE. 2.*(RW*DABS(B) +
AE).
C
C     Description of parameters
C        F     - Name of the external function, which returns a
C                real result.  This name must be in an
C                EXTERNAL statement in the calling program.
C        B     - One end of the interval (B, C).  The value returned for
C                B usually is the better approximation to a zero of F.
C        C     - The other end of the interval (B, C).
C        RE    - Relative error used for RW in the stopping criterion.
C                If the requested RE is less than machine precision,
C                then RW is set to approximately machine precision.
C        AE    - Absolute error used in the stopping criterion.  If the
C                given interval (B, C) contains the origin, then a
C                nonzero value should be chosen for AE.
C
C     REFERENCES
C       1.  L F Shampine and H A Watts, ZEROIN, A Root-Solving Routine,
C           SC-TM-70-631, Sept 1970.
C       2.  T J Dekker, Finding a Zero by Means of Successive Linear
C           Interpolation, "Constructive Aspects of the Fundamental
C           Theorem of Algebra", edited by B Dejon and P Henrici, 1969.
C***ROUTINES CALLED  SDNTP
```

```
C***DATE WRITTEN   790601   (YYMMDD)
C***REVISION DATE  870511   (YYMMDD)
C***CATEGORY NO.  I1A2,I1A1B
C***AUTHOR  KAHANER, D. K., NATIONAL BUREAU OF STANDARDS,
C           SUTHERLAND, C. D., LOS ALAMOS NATIONAL LABORATORY
C***END PROLOGUE  SDZRO
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION  Y(*), YH(N,*)
C***FIRST EXECUTABLE STATEMENT  SDZRO
      ER = 4.E0*UROUND
      RW = DMAX1(RE, ER)
      IC = 0
      ACBS = DABS(B - C)
      A = C
      FA = FC
      KOUNT = 0
C                                                    Perform interchange
 10   IF (DABS(FC) .LT. DABS(FB)) THEN
        A = B
        FA = FB
        B = C
        FB = FC
        C = A
        FC = FA
      END IF
      CMB = 0.5E0*(C - B)
      ACMB = DABS(CMB)
      TOL = RW*DABS(B) + AE
C                                                 Test stopping criterion
      IF (ACMB .LE. TOL) RETURN
      IF (KOUNT .GT. 50) RETURN
C                                    Calculate new iterate implicitly as
C                                    B + P/Q, where we arrange P .GE. 0.
C                           The implicit form is used to prevent overflow.
      P = (B - A)*FB
      Q = FA - FB
      IF (P .LT. 0.E0) THEN
        P = -P
        Q = -Q
      END IF
C                           Update A and check for satisfactory reduction
C                           in the size of our bounding interval.
      A = B
      FA = FB
      IC = IC + 1
      IF (IC .GE. 4) THEN
        IF (8.E0*ACMB .GE. ACBS) THEN
C                                                                  Bisect
          B = 0.5E0*(C + B)
          GO TO 20
        END IF
        IC = 0
      END IF
      ACBS = ACMB
C                                             Test for too small a change
      IF (P .LE. DABS(Q)*TOL) THEN
C                                                 Increment by tolerance
```

```
          B = B + DSIGN(TOL, CMB)
 C                                                        Root ought to be between
 C                                                        B and (C + B)/2.
        ELSE IF (P .LT. CMB*Q) THEN
 C                                                                     Interpolate
          B = B + P/Q
        ELSE
 C                                                                          Bisect
          B = 0.5E0*(C + B)
        END IF
 C                                                      Have completed computation
 C                                                      for new iterate B.
  20    CALL SDNTP (H, 0, N, NQ, T, B, YH,  Y)
        FB = F(N, B, Y, IROOT)
        IF (N .EQ. 0) RETURN
        IF (FB .EQ. 0.E0) RETURN
        KOUNT = KOUNT + 1
 C
 C              Decide whether next step is interpolation or extrapolation
 C
        IF (DSIGN(1.0D0, FB ) .EQ. DSIGN(1.0D0, FC)) THEN
          C = A
          FC = FA
        END IF
        GO TO 10
        END
        SUBROUTINE SDRIV3 (N,T,Y,F,NSTATE,TOUT,NTASK,NROOT,EPS,EWT,IERROR,
      8   MINT,MITER,IMPL,ML,MU,MXORD,HMAX,WORK,LENW,IWORK,LENIW,JACOBN,
      8   FA,NDE,MXSTEP,G,USERS)
C***BEGIN PROLOGUE  SDRIV3
C      THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C      FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C
C     From the book "Numerical Methods and Software"
C        by  D. Kahaner, C. Moler, S. Nash
C            Prentice Hall 1988
C
C***END PROLOGUE  SDRIV3
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       EXTERNAL F, JACOBN, FA, G, USERS
       REAL*8 NROUND
       DIMENSION  EWT(*), WORK(*), Y(*)
       INTEGER IWORK(*)
       LOGICAL CONVRG
       CHARACTER MSG*205
       PARAMETER(NROUND = 20.E0)
       PARAMETER(IAVGH = 1, IHUSED = 2, IAVGRD = 3,
     8           IEL = 4, IH = 160, IHMAX = 161, IHOLD = 162,
     8           IHSIGN = 163, IRC = 164, IRMAX = 165, IT = 166,
     8           ITOUT = 167, ITQ = 168, ITREND = 204, IYH = 205,
     8           INDMXR = 1, INQUSD = 2, INSTEP = 3, INFE = 4, INJE = 5,
     8           INROOT = 6, ICNVRG = 7, IJROOT = 8, IJTASK = 9,
     8           IMNTLD = 10, IMTRLD = 11, INQ = 12, INRTLD = 13,
     8           INDTRT = 14, INWAIT = 15, IMNT = 16, IMTRSV = 17,
     8           IMTR = 18, IMXRDS = 19, IMXORD = 20, INDPRT = 21,
     8           INDPVT = 22)
```

```
C***FIRST EXECUTABLE STATEMENT  SDRIV3
      NPAR = N
      UROUND = D1MACH (4)
      IF (NROOT .NE. 0) THEN
        AE = D1MACH(1)
        RE = UROUND
      END IF
      IF (EPS .LT. 0.E0) THEN
        WRITE(MSG, '(''SDRIV36FE Illegal input.  EPS,'', E16.8,
     8  '', is negative.'')') EPS
        CALL XERROR(MSG(1:60), 60, 6, 2)
        RETURN
      END IF
      IF (N .LE. 0) THEN
        WRITE(MSG, '(''SDRIV37FE Illegal input.  Number of equations,'',
     8  I8, '', is not positive.'')') N
        CALL XERROR(MSG(1:72), 72, 7, 2)
        RETURN
      END IF
      IF (MXORD .LE. 0) THEN
        WRITE(MSG, '(''SDRIV314FE Illegal input.  Maximum order,'', I8,
     8  '', is not positive.'')') MXORD
        CALL XERROR(MSG(1:67), 67, 14, 2)
        RETURN
      END IF
      IF ((MINT .LT. 1 .OR. MINT .GT. 3) .OR. (MINT .EQ. 3 .AND.
     8  (MITER .EQ. 0 .OR. MITER .EQ. 3 .OR. IMPL .NE. 0))
     8  .OR. (MITER .LT. 0 .OR. MITER .GT. 5) .OR.
     8  (IMPL .NE. 0 .AND. IMPL .NE. 1 .AND. IMPL .NE. 2) .OR.
     8  ((IMPL .EQ. 1 .OR. IMPL .EQ. 2) .AND. MITER .EQ. 0) .OR.
     8  (IMPL .EQ. 2 .AND. MINT .EQ. 1) .OR.
     8  (NSTATE .LT. 1 .OR. NSTATE .GT. 10)) THEN
        WRITE(MSG, '(''SDRIV39FE Illegal input.  Improper value for '',
     8  ''NSTATE(MSTATE), MINT, MITER or IMPL.'')')
        CALL XERROR(MSG(1:81), 81, 9, 2)
        RETURN
      END IF
      IF (MITER .EQ. 0 .OR. MITER .EQ. 3) THEN
        LIWCHK = INDPVT - 1
      ELSE IF (MITER .EQ. 1 .OR. MITER .EQ. 2 .OR. MITER .EQ. 4 .OR.
     8  MITER .EQ. 5) THEN
        LIWCHK = INDPVT + N - 1
      END IF
      IF (LENIW .LT. LIWCHK) THEN
        WRITE(MSG, '(''SDRIV310FE Illegal input.  Insufficient '',
     8  ''storage allocated for the IWORK array.  Based on the '')')
        WRITE(MSG(94:), '(''value of the input parameters involved, '',
     8  ''the required storage is'', I8)') LIWCHK
        CALL XERROR(MSG(1:164), 164, 10, 2)
        RETURN
      END IF
C                                                   Allocate the WORK array
C                                            IYH is the index of YH in WORK
      IF (MINT .EQ. 1 .OR. MINT .EQ. 3) THEN
        MAXORD = MIN(MXORD, 12)
      ELSE IF (MINT .EQ. 2) THEN
        MAXORD = MIN(MXORD, 5)
      END IF
      IDFDY = IYH + (MAXORD + 1)*N
C                                                 IDFDY is the index of DFDY
C
      IF (MITER .EQ. 0 .OR. MITER .EQ. 3)  THEN
        IYWT = IDFDY
      ELSE IF (MITER .EQ. 1 .OR. MITER .EQ. 2)  THEN
        IYWT = IDFDY + N*N
      ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5)  THEN
        IYWT = IDFDY + (2*ML + MU + 1)*N
      END IF
C                                                   IYWT is the index of YWT
      ISAVE1 = IYWT + N
C                                               ISAVE1 is the index of SAVE1
      ISAVE2 = ISAVE1 + N
C                                               ISAVE2 is the index of SAVE2
      IGNOW = ISAVE2 + N
C                                                 IGNOW is the index of GNOW
      ITROOT = IGNOW + NROOT
C                                               ITROOT is the index of TROOT
      IFAC = ITROOT + NROOT
C                                                   IFAC is the index of FAC
      IF (MITER .EQ. 2 .OR. MITER .EQ. 5 .OR. MINT .EQ. 3) THEN
        IA = IFAC + N
      ELSE
        IA = IFAC
      END IF
C                                                       IA is the index of A
      IF (IMPL .EQ. 0 .OR. MITER .EQ. 3) THEN
        LENCHK = IA - 1
      ELSE IF (IMPL .EQ. 1 .AND. (MITER .EQ. 1 .OR. MITER .EQ. 2)) THEN
        LENCHK = IA - 1 + N*N
      ELSE IF (IMPL .EQ. 1 .AND. (MITER .EQ. 4 .OR. MITER .EQ. 5)) THEN
        LENCHK = IA - 1 + (2*ML + MU + 1)*N
      ELSE IF (IMPL .EQ. 2 .AND. MITER .NE. 3) THEN
        LENCHK = IA - 1 + N
      END IF
      IF (LENW .LT. LENCHK) THEN
        WRITE(MSG, '(''SDRIV38FE Illegal input.  Insufficient '',
     8  ''storage allocated for the WORK array.  Based on the '')')
        WRITE(MSG(92:), '(''value of the input parameters involved, '',
     8  ''the required storage is'', I8)') LENCHK
        CALL XERROR(MSG(1:162), 162, 8, 2)
        RETURN
      END IF
      IF (MITER .EQ. 0 .OR. MITER .EQ. 3) THEN
        MATDIM = 1
      ELSE IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
        MATDIM = N
      ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
        MATDIM = 2*ML + MU + 1
      END IF
      IF (IMPL .EQ. 0 .OR. IMPL .EQ. 1) THEN
        NDECOM = N
      ELSE IF (IMPL .EQ. 2) THEN
        NDECOM = NDE
      END IF
      IF (NSTATE .EQ. 1) THEN
C                                                   Initialize parameters
        IF (MINT .EQ. 1 .OR. MINT .EQ. 3) THEN
          IWORK(IMXORD) = MIN(MXORD, 12)
        ELSE IF (MINT .EQ. 2) THEN
          IWORK(IMXORD) = MIN(MXORD, 5)
        END IF
        IWORK(IMXRDS) = MXORD
        IF (MINT .EQ. 1 .OR. MINT .EQ. 2) THEN
          IWORK(IMNT) = MINT
          IWORK(IMTR) = MITER
          IWORK(IMNTLD) = MINT
          IWORK(IMTRLD) = MITER
        ELSE IF (MINT .EQ. 3) THEN
          IWORK(IMNT) = 1
          IWORK(IMTR) = 0
          IWORK(IMNTLD) = IWORK(IMNT)
          IWORK(IMTRLD) = IWORK(IMTR)
          IWORK(IMTRSV) = MITER
        END IF
        WORK(IHMAX) = HMAX
        H = (TOUT - T)*(1.D0 - 4.D0*UROUND)
        H = DSIGN(DMIN1(DABS(H), HMAX), H)
        WORK(IH) = H
        HSIGN = DSIGN(1.D0, H)
        WORK(IHSIGN) = HSIGN
        IWORK(IJTASK) = 0
        WORK(IAVGH) = 0.D0
        WORK(IHUSED) = 0.D0
        WORK(IAVGRD) = 0.D0
        IWORK(INDMXR) = 0
        IWORK(INQUSD) = 0
        IWORK(INSTEP) = 0
        IWORK(INFE) = 0
        IWORK(INJE) = 0
        IWORK(INROOT) = 0
        WORK(IT) = T
        IWORK(ICNVRG) = 0
        IWORK(INDPRT) = 0
C                                                   Set initial conditions
        DO 30 I = 1,N
          JYH = I + IYH - 1
 30       WORK(JYH) = Y(I)
        IF (T .EQ. TOUT) RETURN
        GO TO 180
      END IF
C                                                   On a continuation, check
C                                                   that output points have
C                                                   been or will be overtaken.
      IF (IWORK(ICNVRG) .EQ. 1) THEN
        CONVRG = .TRUE.
      ELSE
        CONVRG = .FALSE.
      END IF
      T = WORK(IT)
      H = WORK(IH)
      HSIGN = WORK(IHSIGN)
      IF (IWORK(IJTASK) .EQ. 0) GO TO 180
C
C                                        IWORK(IJROOT) flags unreported
C                                        roots, and is set to the value of
C                                        NTASK when a root was last selected.
C                                        It is set to zero when all roots
C                                        have been reported.  IWORK(INROOT)
C                                        contains the index and WORK(ITOUT)
C                                        contains the value of the root last
C                                        selected to be reported.
C                                        IWORK(INRTLD) contains the value of
C                                        NROOT and IWORK(INDTRT) contains
C                                        the value of ITROOT when the array
C                                        of roots was last calculated.
      IF (NROOT .NE. 0) THEN
        JROOT = IWORK(IJROOT)
        IF (JROOT .GT. 0) THEN
C                                          TOUT has just been reported.
C                                          If TROOT .LE. TOUT, report TROOT.
          IF (NSTATE .NE. 5) THEN
            IF (TOUT*HSIGN .GE. WORK(ITOUT)*HSIGN) THEN
              TROOT = WORK(ITOUT)
              CALL SDNTP(H, 0, N, IWORK(INQ), T, TROOT, WORK(IYH),  Y)
              T = TROOT
              NSTATE = 5
              GO TO 580
            END IF
C                                            A root has just been reported.
C                                            Select the next root.
          ELSE
            TROOT = T
            IROOT = 0
            DO 50 I = 1,IWORK(INRTLD)
              JTROOT = IWORK(INDTRT) + I - 1
              IF (WORK(JTROOT)*HSIGN .LE. TROOT*HSIGN) THEN
C
C                                                Check for multiple roots.
C
                IF (WORK(JTROOT) .EQ. WORK(ITOUT) .AND.
     8          I .GT. IWORK(INROOT)) THEN
                  IROOT = I
                  TROOT = WORK(JTROOT)
                  GO TO 60
                END IF
                IF (WORK(JTROOT)*HSIGN .GT. WORK(ITOUT)*HSIGN) THEN
                  IROOT = I
                  TROOT = WORK(JTROOT)
                END IF
              END IF
 50           CONTINUE
 60         IWORK(INROOT) = IROOT
            WORK(ITOUT) = TROOT
            IWORK(IJROOT) = NTASK
            IF (NTASK .EQ. 1) THEN
              IF (IROOT .EQ. 0) THEN
                IWORK(IJROOT) = 0
              ELSE
                IF (TOUT*HSIGN .GE. TROOT*HSIGN) THEN
                  CALL SDNTP(H, 0, N, IWORK(INQ), T, TROOT,WORK(IYH),Y)
                  NSTATE = 5
                  T = TROOT
                  GO TO 580
                END IF
              END IF
            ELSE IF (NTASK .EQ. 2 .OR. NTASK .EQ. 3) THEN
C
C                                       If there are no more roots, or the
C                                       user has altered TOUT to be less
C                                       than a root, set IJROOT to zero.
C
              IF (IROOT .EQ. 0 .OR. (TOUT*HSIGN .LT. TROOT*HSIGN)) THEN
                IWORK(IJROOT) = 0
              ELSE
                CALL SDNTP(H, 0, N, IWORK(INQ), T, TROOT, WORK(IYH), Y)
                NSTATE = 5
                T = TROOT
                GO TO 580
              END IF
            END IF
          END IF
        END IF
      END IF
C
      IF (NTASK .EQ. 1) THEN
        NSTATE = 2
        IF (T*HSIGN .GE. TOUT*HSIGN) THEN
          CALL SDNTP (H, 0, N, IWORK(INQ), T, TOUT, WORK(IYH),  Y)
          T = TOUT
          GO TO 580
        END IF
      ELSE IF (NTASK .EQ. 2) THEN
C                                                     Check if TOUT has
C                                                     been reset .LT. T
        IF (T*HSIGN .GT. TOUT*HSIGN) THEN
          WRITE(MSG, '(''SDRIV32WRN With NTASK='', I1, '' on input, '',
     8    ''T,'', E16.8, '', was beyond TOUT,'', E16.8, ''.  Solution'',
     8    '' obtained by interpolation.'')') NTASK, T, TOUT
          CALL XERROR(MSG(1:124), 124, 2, 0)
          CALL SDNTP (H, 0, N, IWORK(INQ), T, TOUT, WORK(IYH),  Y)
          T = TOUT
          NSTATE = 2
          GO TO 580
        END IF
C                                  Determine if TOUT has been overtaken
C
      IF (DABS(TOUT - T).LE.NROUND*UROUND*DMAX1(DABS(T),DABS(TOUT)))
     +     THEN
          T = TOUT
          NSTATE = 2
          GO TO 560
        END IF
C                                               If there are no more roots
C                                               to report, report T.
```

```
          IF (NSTATE .EQ. 5) THEN
            NSTATE = 2
            GO TO 560
          END IF
          NSTATE = 2
C                                                              See if TOUT will
C                                                              be overtaken.
          IF ((T + H)*HSIGN .GT. TOUT*HSIGN) THEN
            H = TOUT - T
            IF ((T + H)*HSIGN .GT. TOUT*HSIGN) H = H*(1.E0 - 4.E0*UROUND)
            WORK(IH) = H
            IF (H .EQ. 0.E0) GO TO 670
            IWORK(IJTASK) = -1
          END IF
        ELSE IF (NTASK .EQ. 3) THEN
          NSTATE = 2
          IF (T*HSIGN .GT. TOUT*HSIGN) THEN
            WRITE(MSG, '(''SDRIV32WRN With NTASK='', I1, '' on input, '',
     8      ''T,'', E16.8, '', was beyond TOUT,'', E16.8, ''.  Solution'',
     8      '' obtained by interpolation.'')') NTASK, T, TOUT
            CALL XERROR(MSG(1:124), 124, 2, 0)
            CALL SDNTP (H, 0, N, IWORK(INQ), T, TOUT, WORK(IYH),  Y)
            T = TOUT
            GO TO 580
          END IF
          IF (DABS(TOUT - T).LE.NROUND*UROUND*DMAX1(DABS(T),DABS(TOUT)))
     +      THEN
            T = TOUT
            GO TO 560
          END IF
          IF ((T + H)*HSIGN .GT. TOUT*HSIGN) THEN
            H = TOUT - T
            IF ((T + H)*HSIGN .GT. TOUT*HSIGN) H = H*(1.E0 - 4.E0*UROUND)
            WORK(IH) = H
            IF (H .EQ. 0.D0) GO TO 670
            IWORK(IJTASK) = -1
          END IF
        END IF
C                              Implement changes in MINT, MITER, and/or HMAX.
C
        IF ((MINT .NE. IWORK(IMNTLD) .OR. MITER .NE. IWORK(IMTRLD)) .AND.
     8    MINT .NE. 3 .AND. IWORK(IMNTLD) .NE. 3) IWORK(IJTASK) = -1
        IF (HMAX .NE. WORK(IHMAX)) THEN
          H = DSIGN(DMIN1(DABS(H), HMAX), H)
          IF (H .NE. WORK(IH)) THEN
            IWORK(IJTASK) = -1
            WORK(IH) = H
          END IF
          WORK(IHMAX) = HMAX
        END IF
C
 180    NSTEPL = IWORK(INSTEP)
        DO 190 I = 1,N
          JYH = IYH + I - 1
 190      Y(I) = WORK(JYH)
        IF (NROOT .NE. 0) THEN
```

```
          DO 200 I = 1,NROOT
            JGNOW = IGNOW + I - 1
            WORK(JGNOW) = G (NPAR, T, Y, I)
            IF (NPAR .EQ. 0) THEN
              IWORK(INROOT) = I
              NSTATE = 7
              RETURN
            END IF
 200      CONTINUE
        END IF
        IF (IERROR .EQ. 1) THEN
          DO 230 I = 1,N
            JYWT = I + IYWT - 1
 230        WORK(JYWT) = 1.E0
          GO TO 410
        ELSE IF (IERROR .EQ. 5) THEN
          DO 250 I = 1,N
            JYWT = I + IYWT - 1
 250        WORK(JYWT) = EWT(I)
          GO TO 410
        END IF
C                                                   Reset YWT array.  Looping point.
 260    IF (IERROR .EQ. 2) THEN
          DO 280 I = 1,N
            IF (Y(I) .EQ. 0.E0) GO TO 290
            JYWT = I + IYWT - 1
 280        WORK(JYWT) = DABS(Y(I))
          GO TO 410
 290      IF (IWORK(IJTASK) .EQ. 0) THEN
            CALL F (NPAR, T, Y, WORK(ISAVE2))
            IF (NPAR .EQ. 0) THEN
              NSTATE = 6
              RETURN
            END IF
            IWORK(INFE) = IWORK(INFE) + 1
            IF (MITER .EQ. 3 .AND. IMPL .NE. 0) THEN
              IFLAG = 0
              CALL USERS(Y, WORK(IYH), WORK(IYWT), WORK(ISAVE1),
     8                   WORK(ISAVE2), T, H, WORK(IEL), IMPL, NPAR,
     8                   NDECOM, IFLAG)
              IF (NPAR .EQ. 0) THEN
                NSTATE = 10
                RETURN
              END IF
            ELSE IF (IMPL .EQ. 1) THEN
              IF (MITER .EQ. 1 .OR. MITER .EQ. 2) THEN
                CALL FA (NPAR, T, Y, WORK(IA), MATDIM, ML, MU, NDECOM)
                IF (NPAR .EQ. 0) THEN
                  NSTATE = 9
                  RETURN
                END IF
                CALL SGEFA (WORK(IA), MATDIM, N, IWORK(INDPVT), INFO)
                IF (INFO .NE. 0) GO TO 690
                CALL SGESL(WORK(IA),MATDIM,N,IWORK(INDPVT),WORK(ISAVE2),0)
              ELSE IF (MITER .EQ. 4 .OR. MITER .EQ. 5) THEN
                JAML = IA + ML
                CALL FA (NPAR, T, Y, WORK(JAML), MATDIM, ML, MU, NDECOM)
              IF (NPAR .EQ. 0) THEN
                NSTATE = 9
                RETURN
              END IF
              CALL SGBFA (WORK(IA),MATDIM,N,ML,MU,IWORK(INDPVT),INFO)
              IF (INFO .NE. 0) GO TO 690
              CALL SGBSL (WORK(IA), MATDIM, N, ML, MU, IWORK(INDPVT),
     8                    WORK(ISAVE2), 0)
            END IF
          ELSE IF (IMPL .EQ. 2) THEN
            CALL FA (NPAR, T, Y, WORK(IA), MATDIM, ML, MU, NDECOM)
            IF (NPAR .EQ. 0) THEN
              NSTATE = 9
              RETURN
            END IF
            DO 340 I = 1,NDECOM
              JA = I + IA - 1
              JSAVE2 = I + ISAVE2 - 1
              IF (WORK(JA) .EQ. 0.E0) GO TO 690
 340          WORK(JSAVE2) = WORK(JSAVE2)/WORK(JA)
          END IF
        END IF
        DO 360 J = I,N
          JYWT = J + IYWT - 1
          IF (Y(J) .NE. 0.E0) THEN
            WORK(JYWT) = DABS(Y(J))
          ELSE
            IF (IWORK(IJTASK) .EQ. 0) THEN
              JSAVE2 = J + ISAVE2 - 1
              WORK(JYWT) = DABS(H*WORK(JSAVE2))
            ELSE
              JHYP = J + IYH + N - 1
              WORK(JYWT) = DABS(WORK(JHYP))
            END IF
          END IF
          IF (WORK(JYWT) .EQ. 0.E0) WORK(JYWT) = UROUND
 360      CONTINUE
      ELSE IF (IERROR .EQ. 3) THEN
        DO 380 I = 1,N
          JYWT = I + IYWT - 1
 380      WORK(JYWT) = DMAX1(EWT(1), DABS(Y(I)))
      ELSE IF (IERROR .EQ. 4) THEN
        DO 400 I = 1,N
          JYWT = I + IYWT - 1
 400      WORK(JYWT) = DMAX1(EWT(I), DABS(Y(I)))
      END IF
C
 410  DO 420 I = 1,N
        JYWT = I + IYWT - 1
        JSAVE2 = I + ISAVE2 - 1
 420    WORK(JSAVE2) = Y(I)/WORK(JYWT)
      SUM = SNRM2(N, WORK(ISAVE2), 1)/DSQRT(DBLE(N))
      IF (EPS .LT. SUM*UROUND) THEN
        EPS = SUM*UROUND*(1.E0 + 10.E0*UROUND)
        WRITE(MSG, '(''SDRIV34REC At T,'', E16.8, '', the requested '',
     8  ''accuracy, EPS, was not obtainable with the machine '',
     8  ''precision.  EPS has been increased to'')') T
```

```
          WRITE(MSG(137:), '(E16.8)') EPS
          CALL XERROR(MSG(1:152), 152, 4, 1)
          NSTATE = 4
          GO TO 560
        END IF
        IF (DABS(H) .GE. UROUND*DABS(T)) THEN
          IWORK(INDPRT) = 0
        ELSE IF (IWORK(INDPRT) .EQ. 0) THEN
          WRITE(MSG, '(''SDRIV35WRN At T,'', E16.8, '', the step size,'',
     8    E16.8, '', is smaller than the roundoff level of T.  '')') T, H
          WRITE(MSG(109:), '(''This may occur if there is an abrupt '',
     8    ''change in the right hand side of the differential '',
     8    ''equations.'')')
          CALL XERROR(MSG(1:205), 205, 5, 0)
          IWORK(INDPRT) = 1
        END IF
        IF (NTASK.NE.2) THEN
          IF ((IWORK(INSTEP)-NSTEPL) .GT. MXSTEP) THEN
            WRITE(MSG, '(''SDRIV33WRN At T,'', E16.8, '', '', I8,
     8      '' steps have been taken without reaching TOUT,'', E16.8)')
     8      T, MXSTEP, TOUT
            CALL XERROR(MSG(1:103), 103, 3, 0)
            NSTATE = 3
            GO TO 560
          END IF
        END IF
C
C       CALL SDSTP (EPS, F, FA, HMAX, IMPL, JACOBN, MATDIM, MAXORD,
C    8              MINT, MITER, ML, MU, N, NDE, YWT, UROUND, USERS,
C    8              AVGH, AVGORD, H, HUSED, JTASK, MNTOLD, MTROLD,
C    8              NFE, NJE, NQUSED, NSTEP, T, Y, YH,  A, CONVRG,
C    8              DFDY, EL, FAC, HOLD, IPVT, JSTATE, NQ, NWAIT, RC,
C    8              RMAX, SAVE1, SAVE2, TQ, TREND, ISWFLG, MTRSV, MXRDSV)
C
        CALL SDSTP (EPS, F, FA, WORK(IHMAX), IMPL, JACOBN, MATDIM,
     8              IWORK(IMXORD), IWORK(IMNT), IWORK(IMTR), ML, MU, NPAR,
     8             NDECOM, WORK(IYWT), UROUND, USERS,  WORK(IAVGH),
     8             WORK(IAVGRD), WORK(IH), WORK(IHUSED), IWORK(IJTASK),
     8             IWORK(IMNTLD), IWORK(IMTRLD), IWORK(INFE), IWORK(INJE),
     8              IWORK(INQUSD), IWORK(INSTEP), WORK(IT), Y, WORK(IYH),
     8              WORK(IA), CONVRG, WORK(IDFDY), WORK(IEL), WORK(IFAC),
     8              WORK(IHOLD), IWORK(INDPVT), JSTATE, IWORK(INQ),
     8              IWORK(INWAIT), WORK(IRC), WORK(IRMAX), WORK(ISAVE1),
     8              WORK(ISAVE2), WORK(ITQ), WORK(ITREND), MINT,
     8              IWORK(IMTRSV), IWORK(IMXRDS))
        T = WORK(IT)
        H = WORK(IH)
        GO TO (470, 670, 680, 690, 690, 660, 660, 660, 660, 660), JSTATE
 470    IWORK(IJTASK) = 1
C                                 Determine if a root has been overtaken
        IF (NROOT .NE. 0) THEN
          IROOT = 0
          DO 500 I = 1,NROOT
            JTROOT = ITROOT + I - 1
            JGNOW = IGNOW + I - 1
            GLAST = WORK(JGNOW)
```

```
             WORK(JGNOW) = G (NPAR, T, Y, I)
             IF (NPAR .EQ. 0) THEN
               IWORK(INROOT) = I
               NSTATE = 7
               RETURN
             END IF
             IF (GLAST*WORK(JGNOW) .GT. 0.E0) THEN
               WORK(JTROOT) = T + H
             ELSE
               IF (WORK(JGNOW) .EQ. 0.E0) THEN
                 WORK(JTROOT) = T
                 IROOT = I
               ELSE
                 IF (GLAST .EQ. 0.E0) THEN
                   WORK(JTROOT) = T + H
                 ELSE
                   IF (DABS(WORK(IHUSED)) .GE. UROUND*DABS(T)) THEN
                     TLAST = T - WORK(IHUSED)
                     IROOT = I
                     TROOT = T
                     CALL SDZRO (AE, G, H, NPAR, IWORK(INQ), IROOT, RE, T,
     8                           WORK(IYH), UROUND,  TROOT, TLAST,
     8                           WORK(JGNOW), GLAST,  Y)
                     DO 480 J = 1,N
 480                   Y(J) = WORK(IYH + J -1)
                     IF (NPAR .EQ. 0) THEN
                       IWORK(INROOT) = I
                       NSTATE = 7
                       RETURN
                     END IF
                     WORK(JTROOT) = TROOT
                   ELSE
                     WORK(JTROOT) = T
                     IROOT = I
                   END IF
                 END IF
               END IF
             END IF
 500         CONTINUE
           IF (IROOT .EQ. 0) THEN
             IWORK(IJROOT) = 0
C                                                      Select the first root
           ELSE
             IWORK(IJROOT) = NTASK
             IWORK(INRTLD) = NROOT
             IWORK(INDTRT) = ITROOT
             TROOT = T + H
             DO 510 I = 1,NROOT
               JTROOT = ITROOT + I - 1
               IF (WORK(JTROOT)*HSIGN .LT. TROOT*HSIGN) THEN
                 TROOT = WORK(JTROOT)
                 IROOT = I
               END IF
 510           CONTINUE
             IWORK(INROOT) = IROOT
             WORK(ITOUT) = TROOT
             IF (TROOT*HSIGN .LE. TOUT*HSIGN) THEN
```

```
              CALL SDNTP (H, 0, N, IWORK(INQ), T, TROOT, WORK(IYH),  Y)
              NSTATE = 5
              T = TROOT
              GO TO 580
            END IF
          END IF
        END IF
C                                   Test for NTASK condition to be satisfied
        NSTATE = 2
        IF (NTASK .EQ. 1) THEN
          IF (T*HSIGN .LT. TOUT*HSIGN) GO TO 260
          CALL SDNTP (H, 0, N, IWORK(INQ), T, TOUT, WORK(IYH),  Y)
          T = TOUT
          GO TO 580
C                                   TOUT is assumed to have been attained
C                                   exactly if T is within twenty roundoff
C                                   units of TOUT, relative to max(TOUT, T).
        ELSE IF (NTASK .EQ. 2) THEN
        IF (DABS(TOUT - T).LE.NROUND*UROUND*DMAX1(DABS(T),DABS(TOUT)))
     +    THEN
            T = TOUT
          ELSE
            IF ((T + H)*HSIGN .GT. TOUT*HSIGN) THEN
              H = TOUT - T
              IF ((T + H)*HSIGN.GT.TOUT*HSIGN) H = H*(1.E0 - 4.E0*UROUND)
              WORK(IH) = H
              IF (H .EQ. 0.E0) GO TO 670
              IWORK(IJTASK) = -1
            END IF
          END IF
        ELSE IF (NTASK .EQ. 3) THEN
        IF (DABS(TOUT - T).LE.NROUND*UROUND*DMAX1(DABS(T),DABS(TOUT)))
     +    THEN
            T = TOUT
          ELSE
            IF ((T + H)*HSIGN .GT. TOUT*HSIGN) THEN
              H = TOUT - T
              IF ((T + H)*HSIGN.GT.TOUT*HSIGN) H = H*(1.E0 - 4.E0*UROUND)
              WORK(IH) = H
              IF (H .EQ. 0.E0) GO TO 670
              IWORK(IJTASK) = -1
            END IF
            GO TO 260
          END IF
        END IF
C                                        All returns are made through this
C                                        section.  IMXERR is determined.
 560  DO 570 I = 1,N
        JYH = I + IYH - 1
 570    Y(I) = WORK(JYH)
 580  IF (CONVRG) THEN
        IWORK(ICNVRG) = 1
      ELSE
        IWORK(ICNVRG) = 0
      END IF
      IF (IWORK(IJTASK) .EQ. 0) RETURN
      BIG = 0.E0
```

```
       IMXERR = 1
       IWORK(INDMXR) = IMXERR
       DO  590 I = 1,N
C                                                  SIZE =
DABS(ERROR(I)/YWT(I))
         JYWT = I + IYWT - 1
         JERROR = I + ISAVE1 - 1
         SIZE = DABS(WORK(JERROR)/WORK(JYWT))
         IF (BIG .LT. SIZE) THEN
           BIG = SIZE
           IMXERR = I
           IWORK(INDMXR) = IMXERR
         END IF
 590     CONTINUE
       RETURN
C
 660   NSTATE = JSTATE
       RETURN
C                                              Fatal errors are processed here
C
 670   WRITE(MSG, '(''SDRIV311FE At T,'', E16.8, '', the attempted '',
     8   ''step size has gone to zero.  Often this occurs if the '',
     8   ''problem setup is incorrect.'')') T
       CALL XERROR(MSG(1:129), 129, 11, 2)
       RETURN
C
 680   WRITE(MSG, '(''SDRIV312FE At T,'', E16.8, '', the step size has'',
     8   '' been reduced about 50 times without advancing the '')') T
       WRITE(MSG(103:), '(''solution.  Often this occurs if the '',
     8   ''problem setup is incorrect.'')')
       CALL XERROR(MSG(1:165), 165, 12, 2)
       RETURN
C
 690   WRITE(MSG, '(''SDRIV313FE At T,'', E16.8, '', while solving'',
     8   '' A*YDOT = F, A is singular.'')') T
       CALL XERROR(MSG(1:74), 74, 13, 2)
       RETURN
       END
       SUBROUTINE SGBFA(ABD,LDA,N,ML,MU,IPVT,INFO)
C***BEGIN PROLOGUE  SGBFA
C      THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C      FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C
C    From the book "Numerical Methods and Software"
C       by  D. Kahaner, C. Moler, S. Nash
C           Prentice Hall 1988
C
C***END PROLOGUE  SGBFA
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER LDA,N,ML,MU,IPVT(*),INFO
      DIMENSION ABD(LDA,*)
C
      INTEGER I,ISAMAX,I0,J,JU,JZ,J0,J1,K,KP1,L,LM,M,MM,NM1
C
C***FIRST EXECUTABLE STATEMENT  SGBFA
      M = ML + MU + 1
      INFO = 0
C
C     ZERO INITIAL FILL-IN COLUMNS
C
      J0 = MU + 2
      J1 = MIN0(N,M) - 1
      IF (J1 .LT. J0) GO TO 30
      DO 20 JZ = J0, J1
         I0 = M + 1 - JZ
         DO 10 I = I0, ML
            ABD(I,JZ) = 0.0E0
   10    CONTINUE
   20 CONTINUE
   30 CONTINUE
      JZ = J1
      JU = 0
C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C
      NM1 = N - 1
      IF (NM1 .LT. 1) GO TO 130
      DO 120 K = 1, NM1
         KP1 = K + 1
C
C        ZERO NEXT FILL-IN COLUMN
C
         JZ = JZ + 1
         IF (JZ .GT. N) GO TO 50
         IF (ML .LT. 1) GO TO 50
            DO 40 I = 1, ML
               ABD(I,JZ) = 0.0E0
   40       CONTINUE
   50    CONTINUE
C
C        FIND L = PIVOT INDEX
C
         LM = MIN0(ML,N-K)
         L = ISAMAX(LM+1,ABD(M,K),1) + M - 1
         IPVT(K) = L + K - M
C
C        ZERO PIVOT IMPLIES THIS COLUMN ALREADY TRIANGULARIZED
C
         IF (ABD(L,K) .EQ. 0.0E0) GO TO 100
C
C           INTERCHANGE IF NECESSARY
C
            IF (L .EQ. M) GO TO 60
               T = ABD(L,K)
               ABD(L,K) = ABD(M,K)
               ABD(M,K) = T
   60       CONTINUE
C
C           COMPUTE MULTIPLIERS
C
            T = -1.0E0/ABD(M,K)
            CALL SSCAL(LM,T,ABD(M+1,K),1)
C
```

```
C           ROW ELIMINATION WITH COLUMN INDEXING
C
            JU = MIN0(MAX0(JU,MU+IPVT(K)),N)
            MM = M
            IF (JU .LT. KP1) GO TO 90
            DO 80 J = KP1, JU
               L = L - 1
               MM = MM - 1
               T = ABD(L,J)
               IF (L .EQ. MM) GO TO 70
                  ABD(L,J) = ABD(MM,J)
                  ABD(MM,J) = T
   70          CONTINUE
               CALL SAXPY(LM,T,ABD(M+1,K),1,ABD(MM+1,J),1)
   80       CONTINUE
   90       CONTINUE
         GO TO 110
  100    CONTINUE
            INFO = K
  110    CONTINUE
  120 CONTINUE
  130 CONTINUE
      IPVT(N) = N
      IF (ABD(M,N) .EQ. 0.0E0) INFO = N
      RETURN
      END
      SUBROUTINE SGBSL(ABD,LDA,N,ML,MU,IPVT,B,JOB)
C***BEGIN PROLOGUE  SGBSL
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C
C    From the book "Numerical Methods and Software"
C       by  D. Kahaner, C. Moler, S. Nash
C           Prentice Hall 1988
C
C***END PROLOGUE  SGBSL
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER LDA,N,ML,MU,IPVT(*),JOB
      DIMENSION ABD(LDA,*),B(*)
C
      INTEGER K,KB,L,LA,LB,LM,M,NM1
C***FIRST EXECUTABLE STATEMENT  SGBSL
      M = MU + ML + 1
      NM1 = N - 1
      IF (JOB .NE. 0) GO TO 50
C
C        JOB = 0 , SOLVE  A * X = B
C        FIRST SOLVE L*Y = B
C
         IF (ML .EQ. 0) GO TO 30
         IF (NM1 .LT. 1) GO TO 30
            DO 20 K = 1, NM1
               LM = MIN0(ML,N-K)
               L = IPVT(K)
               T = B(L)
               IF (L .EQ. K) GO TO 10
                  B(L) = B(K)
                  B(K) = T
   10          CONTINUE
               CALL SAXPY(LM,T,ABD(M+1,K),1,B(K+1),1)
   20       CONTINUE
   30    CONTINUE
C
C        NOW SOLVE  U*X = Y
C
         DO 40 KB = 1, N
            K = N + 1 - KB
            B(K) = B(K)/ABD(M,K)
            LM = MIN0(K,M) - 1
            LA = M - LM
            LB = K - LM
            T = -B(K)
            CALL SAXPY(LM,T,ABD(LA,K),1,B(LB),1)
   40    CONTINUE
      GO TO 100
   50 CONTINUE
C
C        JOB = NONZERO, SOLVE  TRANS(A) * X = B
C        FIRST SOLVE  TRANS(U)*Y = B
C
         DO 60 K = 1, N
            LM = MIN0(K,M) - 1
            LA = M - LM
            LB = K - LM
            T = SDOT(LM,ABD(LA,K),1,B(LB),1)
            B(K) = (B(K) - T)/ABD(M,K)
   60    CONTINUE
C
C        NOW SOLVE TRANS(L)*X = Y
C
         IF (ML .EQ. 0) GO TO 90
         IF (NM1 .LT. 1) GO TO 90
            DO 80 KB = 1, NM1
               K = N - KB
               LM = MIN0(ML,N-K)
               B(K) = B(K) + SDOT(LM,ABD(M+1,K),1,B(K+1),1)
               L = IPVT(K)
               IF (L .EQ. K) GO TO 70
                  T = B(L)
                  B(L) = B(K)
                  B(K) = T
   70          CONTINUE
   80       CONTINUE
   90    CONTINUE
  100 CONTINUE
      RETURN
      END
C
      SUBROUTINE SGEFS(A,LDA,N,V,ITASK,IND,WORK,IWORK,RCOND)
C***BEGIN PROLOGUE  SGEFS
C***DATE WRITTEN   800317   (YYMMDD)
C***REVISION DATE  870916   (YYMMDD)
C***CATEGORY NO.  D2A1
```

```
C***KEYWORDS  GENERAL SYSTEM OF LINEAR EQUATIONS,LINEAR EQUATIONS
C***AUTHOR  VOORHEES, E., (LOS ALAMOS NATIONAL LABORATORY)
C***PURPOSE  SGEFS solves a GENERAL single precision real
C            NXN system of linear equations.
C***DESCRIPTION
C
C    From the book "Numerical Methods and Software"
C       by  D. Kahaner, C. Moler, S. Nash
C           Prentice Hall 1988
C
C    Subroutine SGEFS solves a general NxN system of single
C    precision linear equations using LINPACK subroutines SGECO
C    and SGESL.  That is, if A is an NxN real matrix and if X
C    and B are real N-vectors, then SGEFS solves the equation
C
C                          A*X=B.
C
C    The matrix A is first factored into upper and lower tri-
C    angular matrices U and L using partial pivoting.  These
C    factors and the pivoting information are used to find the
C    solution vector X.  An approximate condition number is
C    calculated to provide a rough estimate of the number of
C    digits of accuracy in the computed solution.
C
C    If the equation A*X=B is to be solved for more than one vector
C    B, the factoring of A does not need to be performed again and
C    the option to only solve (ITASK .EQ. 2) will be faster for
C    the succeeding solutions.  In this case, the contents of A,
C    LDA, N and IWORK must not have been altered by the user follow-
C    ing factorization (ITASK=1).  IND will not be changed by SGEFS
C    in this case.  Other settings of ITASK are used to solve linear
C    systems involving the transpose of A.
C
C  Argument Description ***
C
C    A      REAL(LDA,N)
C             on entry, the doubly subscripted array with dimension
C               (LDA,N) which contains the coefficient matrix.
C             on return, an upper triangular matrix U and the
C               multipliers necessary to construct a matrix L
C               so that A=L*U.
C    LDA    INTEGER
C             the leading dimension of the array A.  LDA must be great-
C             er than or equal to N.  (terminal error message IND=-1)
C    N      INTEGER
C             the order of the matrix A.  The first N elements of
C             the array A are the elements of the first column of
C             the  matrix A.  N must be greater than or equal to 1.
C             (terminal error message IND=-2)
C    V      REAL(N)
C             on entry, the singly subscripted array(vector) of di-
C               mension N which contains the right hand side B of a
C               system of simultaneous linear equations A*X=B.
C             on return, V contains the solution vector, X .
C    ITASK  INTEGER
C             If ITASK=1, the matrix A is factored and then the
C               linear equation is solved.
```

```
 C              If ITASK=2, the equation is solved using the existing
 C                factored matrix A and IWORK.
 C              If ITASK=3, the matrix is factored and A'x=b is solved
 C              If ITASK=4, the transposed equation is solved using the
 C                existing factored matrix A and IWORK.
 C              If ITASK .LT. 1 or ITASK .GT. 4, then the terminal error
 C                message IND=-3 is printed.
 C    IND     INTEGER
 C              GT. 0  IND is a rough estimate of the number of digits
 C                      of accuracy in the solution, X.
 C              LT. 0  see error message corresponding to IND below.
 C    WORK    REAL(N)
 C              a singly subscripted array of dimension at least N.
 C    IWORK   INTEGER(N)
 C              a singly subscripted array of dimension at least N.
 C    RCOND   REAL
 C              estimate of 1.0/cond(A)
 C
 C  Error Messages Printed ***
 C
 C    IND=-1  fatal   N is greater than LDA.
 C    IND=-2  fatal   N is less than 1.
 C    IND=-3  fatal   ITASK is less than 1 or greater than 4.
 C    IND=-4  fatal   The matrix A is computationally singular.
 C                          A solution has not been computed.
 C    IND=-10 warning    The solution has no apparent significance.
 C                          The solution may be inaccurate or the matrix
 C                          A may be poorly scaled.
 C
 C***REFERENCES  SUBROUTINE SGEFS WAS DEVELOPED BY GROUP C-3, LOS ALAMOS
 C                 SCIENTIFIC LABORATORY, LOS ALAMOS, NM 87545.
 C                 THE LINPACK SUBROUTINES USED BY SGEFS ARE DESCRIBED IN
 C                 DETAIL IN THE *LINPACK USERS GUIDE* PUBLISHED BY
 C                 THE SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS
 C                 (SIAM) DATED 1979.
 C***ROUTINES CALLED  D1MACH,SGECO,SGESL,XERROR
 C***END PROLOGUE  SGEFS
 C
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       INTEGER LDA,N,ITASK,IND,IWORK(*)
       DIMENSION A(LDA,*),V(*),WORK(*)
       CHARACTER MSG*54
 C***FIRST EXECUTABLE STATEMENT  SGEFS
       IF (LDA.LT.N)  GO TO 101
       IF (N.LE.0)  GO TO 102
       IF (ITASK.LT.1) GO TO 103
       IF (ITASK.GT.4) GO TO 103
       IF (ITASK.EQ.2 .OR. ITASK.GT.3) GO TO 20
 C
 C     FACTOR MATRIX A INTO LU
       CALL SGECO(A,LDA,N,IWORK,RCOND,WORK)
 C
 C     CHECK FOR COMPUTATIONALLY SINGULAR MATRIX
       IF (RCOND.EQ.0.0)  GO TO 104
 C
 C     COMPUTE IND (ESTIMATE OF NO. OF SIGNIFICANT DIGITS)
       IND=-INT(DLOG10(D1MACH(4)/RCOND))
```

```
C
C     CHECK FOR IND GREATER THAN ZERO
      IF (IND.GT.0)  GO TO 20
      IND=-10
      CALL XERROR( 'SGEFS ERROR (IND=-10) -- SOLUTION MAY HAVE NO SIGNIF
     1ICANCE',58,-10,0)
C
C     SOLVE AFTER FACTORING
   20 JOB=0
      IF (ITASK.GT.2) JOB=1
      CALL SGESL(A,LDA,N,IWORK,V,JOB)
      RETURN
C
C     IF LDA.LT.N, IND=-1, FATAL XERROR MESSAGE
  101 IND=-1
      WRITE(MSG, '(
     * ''SGEFS ERROR (IND=-1) -- LDA='', I5, '' IS LESS THAN N='',
     *     I5       )'   ) LDA, N
      CALL XERROR(MSG(1:54), 54, -1, 0)
      RETURN
C
C     IF N.LT.1, IND=-2, FATAL XERROR MESSAGE
  102 IND=-2
      WRITE(MSG, '(
     * ''SGEFS ERROR (IND=-2) -- N='', I5, '' IS LESS THAN 1.'') ')N
      CALL XERROR(MSG(1:47), 47, -2, 0)
      RETURN
C
C     IF ITASK.LT.1, IND=-3, FATAL XERROR MESSAGE
  103 IND=-3
      WRITE(MSG, '(
     * ''SGEFS ERROR (IND=-3) -- ITASK='', I5, '' IS LT 1 OR GT 4.'')
     *                ') ITASK
      CALL XERROR(MSG(1:52), 52, -3, 0)
      RETURN
C
C     IF SINGULAR MATRIX, IND=-4, FATAL XERROR MESSAGE
  104 IND=-4
      CALL XERROR( 'SGEFS ERROR (IND=-4) -- SINGULAR MATRIX A - NO SOLUT
     1ION',55,-4,0)
      RETURN
C
      END
      SUBROUTINE SGECO(A,LDA,N,IPVT,RCOND,Z)
C***BEGIN PROLOGUE  SGECO
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C               Prentice Hall 1988
C***ROUTINES CALLED  SASUM,SAXPY,SDOT,SGEFA,SSCAL
C***END PROLOGUE  SGECO
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER LDA,N,IPVT(*)
      DIMENSION A(LDA,*),Z(*)
C
```

```
      INTEGER INFO,J,K,KB,KP1,L
C
C     COMPUTE 1-NORM OF A
C
C***FIRST EXECUTABLE STATEMENT  SGECO
      ANORM = 0.0E0
      DO 10 J = 1, N
         ANORM = DMAX1(ANORM,SASUM(N,A(1,J),1))
   10 CONTINUE
C
C     FACTOR
C
      CALL SGEFA(A,LDA,N,IPVT,INFO)
C
C     RCOND = 1/(NORM(A)*(ESTIMATE OF NORM(INVERSE(A)))) .
C     ESTIMATE = NORM(Z)/NORM(Y) WHERE  A*Z = Y  AND  TRANS(A)*Y = E .
C     TRANS(A)  IS THE TRANSPOSE OF A .  THE COMPONENTS OF  E  ARE
C     CHOSEN TO CAUSE MAXIMUM LOCAL GROWTH IN THE ELEMENTS OF W  WHERE
C     TRANS(U)*W = E .  THE VECTORS ARE FREQUENTLY RESCALED TO AVOID
C     OVERFLOW.
C
C     SOLVE TRANS(U)*W = E
C
      EK = 1.0E0
      DO 20 J = 1, N
         Z(J) = 0.0E0
   20 CONTINUE
      DO 100 K = 1, N
         IF (Z(K) .NE. 0.0E0) EK = DSIGN(EK,-Z(K))
         IF (DABS(EK-Z(K)) .LE. DABS(A(K,K))) GO TO 30
            S = DABS(A(K,K))/DABS(EK-Z(K))
            CALL SSCAL(N,S,Z,1)
            EK = S*EK
   30    CONTINUE
         WK = EK - Z(K)
         WKM = -EK - Z(K)
         S = DABS(WK)
         SM = DABS(WKM)
         IF (A(K,K) .EQ. 0.0E0) GO TO 40
            WK = WK/A(K,K)
            WKM = WKM/A(K,K)
         GO TO 50
   40    CONTINUE
            WK = 1.0E0
            WKM = 1.0E0
   50    CONTINUE
         KP1 = K + 1
         IF (KP1 .GT. N) GO TO 90
            DO 60 J = KP1, N
               SM = SM + DABS(Z(J)+WKM*A(K,J))
               Z(J) = Z(J) + WK*A(K,J)
               S = S + DABS(Z(J))
   60       CONTINUE
            IF (S .GE. SM) GO TO 80
               T = WKM - WK
               WK = WKM
               DO 70 J = KP1, N
```

```
                    Z(J) = Z(J) + T*A(K,J)
   70             CONTINUE
   80          CONTINUE
   90       CONTINUE
            Z(K) = WK
  100 CONTINUE
      S = 1.0E0/SASUM(N,Z,1)
      CALL SSCAL(N,S,Z,1)
C
C     SOLVE TRANS(L)*Y = W
C
      DO 120 KB = 1, N
         K = N + 1 - KB
         IF (K .LT. N) Z(K) = Z(K) + SDOT(N-K,A(K+1,K),1,Z(K+1),1)
         IF (DABS(Z(K)) .LE. 1.0E0) GO TO 110
            S = 1.0E0/DABS(Z(K))
            CALL SSCAL(N,S,Z,1)
  110    CONTINUE
         L = IPVT(K)
         T = Z(L)
         Z(L) = Z(K)
         Z(K) = T
  120 CONTINUE
      S = 1.0E0/SASUM(N,Z,1)
      CALL SSCAL(N,S,Z,1)
C
      YNORM = 1.0E0
C
C     SOLVE L*V = Y
C
      DO 140 K = 1, N
         L = IPVT(K)
         T = Z(L)
         Z(L) = Z(K)
         Z(K) = T
         IF (K .LT. N) CALL SAXPY(N-K,T,A(K+1,K),1,Z(K+1),1)
         IF (DABS(Z(K)) .LE. 1.0E0) GO TO 130
            S = 1.0E0/DABS(Z(K))
            CALL SSCAL(N,S,Z,1)
            YNORM = S*YNORM
  130    CONTINUE
  140 CONTINUE
      S = 1.0E0/SASUM(N,Z,1)
      CALL SSCAL(N,S,Z,1)
      YNORM = S*YNORM
C
C     SOLVE  U*Z = V
C
      DO 160 KB = 1, N
         K = N + 1 - KB
         IF (DABS(Z(K)) .LE. DABS(A(K,K))) GO TO 150
            S = DABS(A(K,K))/DABS(Z(K))
            CALL SSCAL(N,S,Z,1)
            YNORM = S*YNORM
  150    CONTINUE
         IF (A(K,K) .NE. 0.0E0) Z(K) = Z(K)/A(K,K)
         IF (A(K,K) .EQ. 0.0E0) Z(K) = 1.0E0
         T = -Z(K)
         CALL SAXPY(K-1,T,A(1,K),1,Z(1),1)
  160 CONTINUE
C     MAKE ZNORM = 1.0
      S = 1.0E0/SASUM(N,Z,1)
      CALL SSCAL(N,S,Z,1)
      YNORM = S*YNORM
C
      IF (ANORM .NE. 0.0E0) RCOND = YNORM/ANORM
      IF (ANORM .EQ. 0.0E0) RCOND = 0.0E0
      RETURN
      END
      SUBROUTINE SGEFA(A,LDA,N,IPVT,INFO)
C***BEGIN PROLOGUE  SGEFA
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C                 Prentice Hall 1988
C***END PROLOGUE  SGEFA
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER LDA,N,IPVT(*),INFO
      DIMENSION A(LDA,*)
C
      INTEGER ISAMAX,J,K,KP1,L,NM1
C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C
C***FIRST EXECUTABLE STATEMENT  SGEFA
      INFO = 0
      NM1 = N - 1
      IF (NM1 .LT. 1) GO TO 70
      DO 60 K = 1, NM1
         KP1 = K + 1
C
C        FIND L = PIVOT INDEX
C
         L = ISAMAX(N-K+1,A(K,K),1) + K - 1
         IPVT(K) = L
C
C        ZERO PIVOT IMPLIES THIS COLUMN ALREADY TRIANGULARIZED
C
         IF (A(L,K) .EQ. 0.0E0) GO TO 40
C
C           INTERCHANGE IF NECESSARY
C
            IF (L .EQ. K) GO TO 10
               T = A(L,K)
               A(L,K) = A(K,K)
               A(K,K) = T
   10       CONTINUE
C
C           COMPUTE MULTIPLIERS
C
            T = -1.0E0/A(K,K)
            CALL SSCAL(N-K,T,A(K+1,K),1)
```

```
C
C           ROW ELIMINATION WITH COLUMN INDEXING
C
            DO 30 J = KP1, N
               T = A(L,J)
               IF (L .EQ. K) GO TO 20
                  A(L,J) = A(K,J)
                  A(K,J) = T
   20          CONTINUE
               CALL SAXPY(N-K,T,A(K+1,K),1,A(K+1,J),1)
   30       CONTINUE
         GO TO 50
   40    CONTINUE
            INFO = K
   50    CONTINUE
   60 CONTINUE
   70 CONTINUE
      IPVT(N) = N
      IF (A(N,N) .EQ. 0.0E0) INFO = N
      RETURN
      END
      SUBROUTINE SGESL(A,LDA,N,IPVT,B,JOB)
C***BEGIN PROLOGUE  SGESL
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C               Prentice Hall 1988
C***END PROLOGUE  SGESL
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER LDA,N,IPVT(*),JOB
      DIMENSION A(LDA,*),B(*)
C
      INTEGER K,KB,L,NM1
C***FIRST EXECUTABLE STATEMENT  SGESL
      NM1 = N - 1
      IF (JOB .NE. 0) GO TO 50
C
C        JOB = 0 , SOLVE  A * X = B
C        FIRST SOLVE  L*Y = B
C
         IF (NM1 .LT. 1) GO TO 30
         DO 20 K = 1, NM1
            L = IPVT(K)
            T = B(L)
            IF (L .EQ. K) GO TO 10
               B(L) = B(K)
               B(K) = T
   10       CONTINUE
            CALL SAXPY(N-K,T,A(K+1,K),1,B(K+1),1)
   20    CONTINUE
   30    CONTINUE
C
C        NOW SOLVE  U*X = Y
C
         DO 40 KB = 1, N
            K = N + 1 - KB
            B(K) = B(K)/A(K,K)
            T = -B(K)
            CALL SAXPY(K-1,T,A(1,K),1,B(1),1)
   40    CONTINUE
      GO TO 100
   50 CONTINUE
C
C        JOB = NONZERO, SOLVE  TRANS(A) * X = B
C        FIRST SOLVE  TRANS(U)*Y = B
C
         DO 60 K = 1, N
            T = SDOT(K-1,A(1,K),1,B(1),1)
            B(K) = (B(K) - T)/A(K,K)
   60    CONTINUE
C
C        NOW SOLVE TRANS(L)*X = Y
C
         IF (NM1 .LT. 1) GO TO 90
         DO 80 KB = 1, NM1
            K = N - KB
            B(K) = B(K) + SDOT(N-K,A(K+1,K),1,B(K+1),1)
            L = IPVT(K)
            IF (L .EQ. K) GO TO 70
               T = B(L)
               B(L) = B(K)
               B(K) = T
   70       CONTINUE
   80    CONTINUE
   90    CONTINUE
  100 CONTINUE
      RETURN
      END
C
      INTEGER FUNCTION ISAMAX(N,SX,INCX)
C***BEGIN PROLOGUE  ISAMAX
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C               Prentice Hall 1988
C***END PROLOGUE  ISAMAX
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION SX(*)
C***FIRST EXECUTABLE STATEMENT  ISAMAX
      ISAMAX = 0
      IF(N.LE.0) RETURN
      ISAMAX = 1
      IF(N.LE.1)RETURN
      IF(INCX.EQ.1)GOTO 20
C
C        CODE FOR INCREMENTS NOT EQUAL TO 1.
C
      SMAX = DABS(SX(1))
      NS = N*INCX
      II = 1
          DO 10 I=1,NS,INCX
```

```
           XMAG = DABS(SX(I))
           IF(XMAG.LE.SMAX) GO TO 5
           ISAMAX = II
           SMAX = XMAG
    5      II = II + 1
   10      CONTINUE
      RETURN
C
C        CODE FOR INCREMENTS EQUAL TO 1.
C
   20 SMAX = DABS(SX(1))
      DO 30 I = 2,N
         XMAG = DABS(SX(I))
         IF(XMAG.LE.SMAX) GO TO 30
         ISAMAX = I
         SMAX = XMAG
   30 CONTINUE
      RETURN
      END
      REAL*8 FUNCTION SASUM(N,SX,INCX)
C***BEGIN PROLOGUE  SASUM
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C               Prentice Hall 1988
C***END PROLOGUE  SASUM
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION SX(*)
C***FIRST EXECUTABLE STATEMENT  SASUM
      SASUM = 0.0E0
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GOTO 20
C
C        CODE FOR INCREMENTS NOT EQUAL TO 1.
C
      NS = N*INCX
          DO 10 I=1,NS,INCX
          SASUM = SASUM + DABS(SX(I))
   10     CONTINUE
      RETURN
C
C        CODE FOR INCREMENTS EQUAL TO 1.
C
C
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 6.
C
   20 M = MOD(N,6)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SASUM = SASUM + DABS(SX(I))
   30 CONTINUE
      IF( N .LT. 6 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,6
```

```
         SASUM = SASUM + DABS(SX(I)) + DABS(SX(I + 1)) + DABS(SX(I + 2))
     1   + DABS(SX(I + 3)) + DABS(SX(I + 4)) + DABS(SX(I + 5))
   50 CONTINUE
      RETURN
      END
      SUBROUTINE SAXPY(N,SA,SX,INCX,SY,INCY)
C***BEGIN PROLOGUE  SAXPY
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C                Prentice Hall 1988
C***END PROLOGUE  SAXPY
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION SX(*),SY(*)
C***FIRST EXECUTABLE STATEMENT  SAXPY
      IF(N.LE.0.OR.SA.EQ.0.E0) RETURN
      IF(INCX.EQ.INCY) IF(INCX-1) 5,20,60
    5 CONTINUE
C
C        CODE FOR NONEQUAL OR NONPOSITIVE INCREMENTS.
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        SY(IY) = SY(IY) + SA*SX(IX)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 4.
C
   20 M = MOD(N,4)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SY(I) = SY(I) + SA*SX(I)
   30 CONTINUE
      IF( N .LT. 4 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,4
        SY(I) = SY(I) + SA*SX(I)
        SY(I + 1) = SY(I + 1) + SA*SX(I + 1)
        SY(I + 2) = SY(I + 2) + SA*SX(I + 2)
        SY(I + 3) = SY(I + 3) + SA*SX(I + 3)
   50 CONTINUE
      RETURN
C
C        CODE FOR EQUAL, POSITIVE, NONUNIT INCREMENTS.
C
   60 CONTINUE
```

```
      NS = N*INCX
          DO 70 I=1,NS,INCX
          SY(I) = SA*SX(I) + SY(I)
   70     CONTINUE
      RETURN
      END
      SUBROUTINE SCOPY(N,SX,INCX,SY,INCY)
C***BEGIN PROLOGUE  SCOPY
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C                 Prentice Hall 1988
C***END PROLOGUE  SCOPY
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION SX(*),SY(*)
C***FIRST EXECUTABLE STATEMENT  SCOPY
      IF(N.LE.0)RETURN
      IF(INCX.EQ.INCY) IF(INCX-1) 5,20,60
    5 CONTINUE
C
C        CODE FOR UNEQUAL OR NONPOSITIVE INCREMENTS.
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        SY(IY) = SX(IX)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 7.
C
   20 M = MOD(N,7)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SY(I) = SX(I)
   30 CONTINUE
      IF( N .LT. 7 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,7
        SY(I) = SX(I)
        SY(I + 1) = SX(I + 1)
        SY(I + 2) = SX(I + 2)
        SY(I + 3) = SX(I + 3)
        SY(I + 4) = SX(I + 4)
        SY(I + 5) = SX(I + 5)
        SY(I + 6) = SX(I + 6)
   50 CONTINUE
      RETURN
```

```
C
C        CODE FOR EQUAL, POSITIVE, NONUNIT INCREMENTS.
C
   60 CONTINUE
      NS = N*INCX
          DO 70 I=1,NS,INCX
          SY(I) = SX(I)
   70     CONTINUE
      RETURN
      END
      REAL*8 FUNCTION SDOT(N,SX,INCX,SY,INCY)
C***BEGIN PROLOGUE  SDOT
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C                 Prentice Hall 1988
C***END PROLOGUE  SDOT
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION SX(*),SY(*)
C***FIRST EXECUTABLE STATEMENT  SDOT
      SDOT = 0.0E0
      IF(N.LE.0)RETURN
      IF(INCX.EQ.INCY) IF(INCX-1)5,20,60
    5 CONTINUE
C
C        CODE FOR UNEQUAL INCREMENTS OR NONPOSITIVE INCREMENTS.
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        SDOT = SDOT + SX(IX)*SY(IY)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 5.
C
   20 M = MOD(N,5)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SDOT = SDOT + SX(I)*SY(I)
   30 CONTINUE
      IF( N .LT. 5 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
        SDOT = SDOT + SX(I)*SY(I) + SX(I + 1)*SY(I + 1) +
     1   SX(I + 2)*SY(I + 2) + SX(I + 3)*SY(I + 3) + SX(I + 4)*SY(I + 4)
   50 CONTINUE
```

```
      RETURN
C
C        CODE FOR POSITIVE EQUAL INCREMENTS .NE.1.
C
   60 CONTINUE
      NS=N*INCX
      DO 70 I=1,NS,INCX
        SDOT = SDOT + SX(I)*SY(I)
   70   CONTINUE
      RETURN
      END
      REAL*8 FUNCTION SNRM2(N,SX,INCX)
C***BEGIN PROLOGUE  SNRM2
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C                 Prentice Hall 1988
C***END PROLOGUE  SNRM2
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER          NEXT
      DIMENSION   SX(*)
      DATA   ZERO, ONE /0.0E0, 1.0E0/
C
      DATA CUTLO, CUTHI / 4.441E-16,  1.304E19 /
C***FIRST EXECUTABLE STATEMENT  SNRM2
      IF(N .GT. 0) GO TO 10
         SNRM2  = ZERO
         GO TO 300
C
   10 ASSIGN 30 TO NEXT
      SUM = ZERO
      NN = N * INCX
C                                                     BEGIN MAIN LOOP
      I = 1
   20    GO TO NEXT,(30, 50, 70, 110)
   30 IF( DABS(SX(I)) .GT. CUTLO) GO TO 85
      ASSIGN 50 TO NEXT
      XMAX = ZERO
C
C                        PHASE 1.  SUM IS ZERO
C
   50 IF( SX(I) .EQ. ZERO) GO TO 200
      IF( DABS(SX(I)) .GT. CUTLO) GO TO 85
C
C                                PREPARE FOR PHASE 2.
      ASSIGN 70 TO NEXT
      GO TO 105
C
C                                PREPARE FOR PHASE 4.
C
  100 I = J
      ASSIGN 110 TO NEXT
      SUM = (SUM / SX(I)) / SX(I)
  105 XMAX = DABS(SX(I))
      GO TO 115
C
```

```
C                    PHASE 2.  SUM IS SMALL.
C                              SCALE TO AVOID DESTRUCTIVE UNDERFLOW.
C
   70 IF( DABS(SX(I)) .GT. CUTLO ) GO TO 75
C
C                      COMMON CODE FOR PHASES 2 AND 4.
C                      IN PHASE 4 SUM IS LARGE.  SCALE TO AVOID OVERFLOW.
C
  110 IF( DABS(SX(I)) .LE. XMAX ) GO TO 115
         SUM = ONE + SUM * (XMAX / SX(I))**2
         XMAX = DABS(SX(I))
         GO TO 200
C
  115 SUM = SUM + (SX(I)/XMAX)**2
      GO TO 200
C
C
C                    PREPARE FOR PHASE 3.
C
   75 SUM = (SUM * XMAX) * XMAX
C
C
C     FOR REAL OR D.P. SET HITEST = CUTHI/N
C     FOR COMPLEX      SET HITEST = CUTHI/(2*N)
C
   85 HITEST = CUTHI/FLOAT( N )
C
C                    PHASE 3.  SUM IS MID-RANGE.  NO SCALING.
C
      DO 95 J =I,NN,INCX
      IF(DABS(SX(J)) .GE. HITEST) GO TO 100
   95    SUM = SUM + SX(J)**2
      SNRM2 = DSQRT( SUM )
      GO TO 300
C
  200 CONTINUE
      I = I + INCX
      IF ( I .LE. NN ) GO TO 20
C
C              END OF MAIN LOOP.
C
C              COMPUTE SQUARE ROOT AND ADJUST FOR SCALING.
C
      SNRM2 = XMAX * DSQRT(SUM)
  300 CONTINUE
      RETURN
      END
      SUBROUTINE SSCAL(N,SA,SX,INCX)
C***BEGIN PROLOGUE  SSCAL
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C               Prentice Hall 1988
C***END PROLOGUE  SSCAL
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION SX(*)
C***FIRST EXECUTABLE STATEMENT  SSCAL
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GOTO 20
C
C        CODE FOR INCREMENTS NOT EQUAL TO 1.
C
      NS = N*INCX
          DO 10 I = 1,NS,INCX
          SX(I) = SA*SX(I)
   10     CONTINUE
      RETURN
C
C        CODE FOR INCREMENTS EQUAL TO 1.
C
C
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 5.
C
   20 M = MOD(N,5)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SX(I) = SA*SX(I)
   30 CONTINUE
      IF( N .LT. 5 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
        SX(I) = SA*SX(I)
        SX(I + 1) = SA*SX(I + 1)
        SX(I + 2) = SA*SX(I + 2)
        SX(I + 3) = SA*SX(I + 3)
        SX(I + 4) = SA*SX(I + 4)
   50 CONTINUE
      RETURN
      END
      SUBROUTINE SSWAP(N,SX,INCX,SY,INCY)
C***BEGIN PROLOGUE  SSWAP
C     THIS PROLOGUE HAS BEEN REMOVED FOR REASONS OF SPACE
C     FOR A COMPLETE COPY OF THIS ROUTINE CONTACT THE AUTHORS
C     From the book "Numerical Methods and Software"
C          by  D. Kahaner, C. Moler, S. Nash
C               Prentice Hall 1988
C***END PROLOGUE  SSWAP
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION SX(*),SY(*)
C***FIRST EXECUTABLE STATEMENT  SSWAP
      IF(N.LE.0)RETURN
      IF(INCX.EQ.INCY) IF(INCX-1) 5,20,60
    5 CONTINUE
C
C       CODE FOR UNEQUAL OR NONPOSITIVE INCREMENTS.
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        STEMP1 = SX(IX)
        SX(IX) = SY(IY)
        SY(IY) = STEMP1
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C       CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C       CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 3.
C
   20 M = MOD(N,3)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        STEMP1 = SX(I)
        SX(I) = SY(I)
        SY(I) = STEMP1
   30 CONTINUE
      IF( N .LT. 3 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,3
        STEMP1 = SX(I)
        STEMP2 = SX(I+1)
        STEMP3 = SX(I+2)
        SX(I) = SY(I)
        SX(I+1) = SY(I+1)
        SX(I+2) = SY(I+2)
        SY(I) = STEMP1
        SY(I+1) = STEMP2
        SY(I+2) = STEMP3
   50 CONTINUE
      RETURN
   60 CONTINUE
C
C     CODE FOR EQUAL, POSITIVE, NONUNIT INCREMENTS.
C
      NS = N*INCX
        DO 70 I=1,NS,INCX
        STEMP1 = SX(I)
        SX(I) = SY(I)
        SY(I) = STEMP1
   70   CONTINUE
      RETURN
      END
C
      DOUBLE PRECISION FUNCTION D1MACH(I)
C***BEGIN PROLOGUE  D1MACH
C***DATE WRITTEN   750101   (YYMMDD)
C***REVISION DATE  831014   (YYMMDD)
C***CATEGORY NO.  R1
C***KEYWORDS  MACHINE CONSTANTS
C***AUTHOR  FOX, P. A., (BELL LABS)
C           HALL, A. D., (BELL LABS)
C           SCHRYER, N. L., (BELL LABS)
```

```
C***PURPOSE  Returns double precision machine dependent constants
C***DESCRIPTION
C     From the book, "Numerical Methods and Software" by
C                D. Kahaner, C. Moler, S. Nash
C                Prentice Hall, 1988
C
C
C     D1MACH can be used to obtain machine-dependent parameters
C     for the local machine environment.  It is a function
C     subprogram with one (input) argument, and can be called
C     as follows, for example
C
C          D = D1MACH(I)
C
C     where I=1,...,5.  The (output) value of D above is
C     determined by the (input) value of I.  The results for
C     various values of I are discussed below.
C
C  Double-precision machine constants
C  D1MACH( 1) = B**(EMIN-1), the smallest positive magnitude.
C  D1MACH( 2) = B**EMAX*(1 - B**(-T)), the largest magnitude.
C  D1MACH( 3) = B**(-T), the smallest relative spacing.
C  D1MACH( 4) = B**(1-T), the largest relative spacing.
C  D1MACH( 5) = LOG10(B)
C***REFERENCES  FOX P.A., HALL A.D., SCHRYER N.L.,*FRAMEWORK FOR A
C                 PORTABLE LIBRARY*, ACM TRANSACTIONS ON MATHEMATICAL
C                 SOFTWARE, VOL. 4, NO. 2, JUNE 1978, PP. 177-188.
C***ROUTINES CALLED  XERROR
C***END PROLOGUE  D1MACH
C
      INTEGER SMALL(4)
      INTEGER LARGE(4)
      INTEGER RIGHT(4)
      INTEGER DIVER(4)
      INTEGER LOG10(4)
C
      DOUBLE PRECISION DMACH(5)
C
      EQUIVALENCE (DMACH(1),SMALL(1))
      EQUIVALENCE (DMACH(2),LARGE(1))
      EQUIVALENCE (DMACH(3),RIGHT(1))
      EQUIVALENCE (DMACH(4),DIVER(1))
      EQUIVALENCE (DMACH(5),LOG10(1))
C
C
C     MACHINE CONSTANTS FOR THE CDC CYBER 170 SERIES (FTN5).
C
C      DATA SMALL(1) / O"00604000000000000000" /
C      DATA SMALL(2) / O"00000000000000000000" /
C
C      DATA LARGE(1) / O"37767777777777777777" /
C      DATA LARGE(2) / O"37167777777777777777" /
C
C      DATA RIGHT(1) / O"15604000000000000000" /
C      DATA RIGHT(2) / O"15000000000000000000" /
C
C      DATA DIVER(1) / O"15614000000000000000" /
```

```
C       DATA DIVER(2) / O"15010000000000000000" /
C
C       DATA LOG10(1) / O"17164642023241175717" /
C       DATA LOG10(2) / O"16367571421742254654" /
C
C     MACHINE CONSTANTS FOR THE CDC CYBER 200 SERIES
C
C     DATA SMALL(1) / X'9000400000000000' /
C     DATA SMALL(2) / X'8FD1000000000000' /
C
C     DATA LARGE(1) / X'6FFF7FFFFFFFFFFF' /
C     DATA LARGE(2) / X'6FD07FFFFFFFFFFF' /
C
C     DATA RIGHT(1) / X'FF74400000000000' /
C     DATA RIGHT(2) / X'FF45000000000000' /
C
C     DATA DIVER(1) / X'FF75400000000000' /
C     DATA DIVER(2) / X'FF46000000000000' /
C
C     DATA LOG10(1) / X'FFD04D104D427DE7' /
C     DATA LOG10(2) / X'FFA17DE623E2566A' /
C
C
C     MACHINE CONSTANTS FOR THE CDC 6000/7000 SERIES.
C
C     DATA SMALL(1) / 00564000000000000000B /
C     DATA SMALL(2) / 00000000000000000000B /
C
C     DATA LARGE(1) / 37757777777777777777B /
C     DATA LARGE(2) / 37157777777777777777B /
C
C     DATA RIGHT(1) / 15624000000000000000B /
C     DATA RIGHT(2) / 00000000000000000000B /
C
C     DATA DIVER(1) / 15634000000000000000B /
C     DATA DIVER(2) / 00000000000000000000B /
C
C     DATA LOG10(1) / 17164642023241175717B /
C     DATA LOG10(2) / 16367571421742254654B /
C
C     MACHINE CONSTANTS FOR THE CRAY 1
C
C     DATA SMALL(1) / 201354000000000000000B /
C     DATA SMALL(2) / 000000000000000000000B /
C
C     DATA LARGE(1) / 577767777777777777777B /
C     DATA LARGE(2) / 000007777777777777774B /
C
C     DATA RIGHT(1) / 376434000000000000000B /
C     DATA RIGHT(2) / 000000000000000000000B /
C
C     DATA DIVER(1) / 376444000000000000000B /
C     DATA DIVER(2) / 000000000000000000000B /
C
C     DATA LOG10(1) / 377774642023241175717B /
C     DATA LOG10(2) / 000007571421742254654B /
```

```
C
C
C     MACHINE CONSTANTS FOR THE IBM 360/370 SERIES,
C     THE XEROX SIGMA 5/7/9, THE SEL SYSTEMS 85/86, AND
C     THE PERKIN ELMER (INTERDATA) 7/32.
C
C     DATA SMALL(1),SMALL(2) / Z00100000, Z00000000 /
C     DATA LARGE(1),LARGE(2) / Z7FFFFFFF, ZFFFFFFFF /
C     DATA RIGHT(1),RIGHT(2) / Z33100000, Z00000000 /
C     DATA DIVER(1),DIVER(2) / Z34100000, Z00000000 /
C     DATA LOG10(1),LOG10(2) / Z41134413, Z509F79FF /
C
C     MACHINE CONSTATNS FOR THE IBM PC FAMILY (D. KAHANER NBS)
C
      DATA DMACH/2.23D-250,1.79D+250,1.11D-16,2.22D-16,
     *  0.301029995663981195D0/
C
C     MACHINE CONSTANTS FOR THE PDP-10 (KA PROCESSOR).
C
C     DATA SMALL(1),SMALL(2) / "033400000000, "000000000000 /
C     DATA LARGE(1),LARGE(2) / "377777777777, "344777777777 /
C     DATA RIGHT(1),RIGHT(2) / "113400000000, "000000000000 /
C     DATA DIVER(1),DIVER(2) / "114400000000, "000000000000 /
C     DATA LOG10(1),LOG10(2) / "177464202324, "144117571776 /
C
C     MACHINE CONSTANTS FOR THE PDP-10 (KI PROCESSOR).
C
C     DATA SMALL(1),SMALL(2) / "000400000000, "000000000000 /
C     DATA LARGE(1),LARGE(2) / "377777777777, "377777777777 /
C     DATA RIGHT(1),RIGHT(2) / "103400000000, "000000000000 /
C     DATA DIVER(1),DIVER(2) / "104400000000, "000000000000 /
C     DATA LOG10(1),LOG10(2) / "177464202324, "476747767461 /
C
C
C     MACHINE CONSTANTS FOR THE SUN-3 (INCLUDES THOSE WITH 68881 CHIP,
C        OR WITH FPA BOARD. ALSO INCLUDES SUN-2 WITH SKY BOARD. MAY ALSO
C        WORK WITH SOFTWARE FLOATING POINT ON EITHER SYSTEM.)
C
C     DATA SMALL(1),SMALL(2) / X'00100000', X'00000000' /
C     DATA LARGE(1),LARGE(2) / X'7FEFFFFF', X'FFFFFFFF' /
C     DATA RIGHT(1),RIGHT(2) / X'3CA00000', X'00000000' /
C     DATA DIVER(1),DIVER(2) / X'3CB00000', X'00000000' /
C     DATA LOG10(1),LOG10(2) / X'3FD34413', X'509F79FF' /
C
C
C     MACHINE CONSTANTS FOR VAX 11/780
C     (EXPRESSED IN INTEGER AND HEXADECIMAL)
C    *** THE INTEGER FORMAT SHOULD BE OK FOR UNIX SYSTEMS***
C
C     DATA SMALL(1), SMALL(2) /       128,           0 /
C     DATA LARGE(1), LARGE(2) /    -32769,          -1 /
C     DATA RIGHT(1), RIGHT(2) /      9344,           0 /
C     DATA DIVER(1), DIVER(2) /      9472,           0 /
C     DATA LOG10(1), LOG10(2) / 546979738,  -805796613 /
C
C    ***THE HEX FORMAT BELOW MAY NOT BE SUITABLE FOR UNIX SYSYEMS***
C     DATA SMALL(1), SMALL(2) / Z00000080, Z00000000 /
```

```
C     DATA LARGE(1), LARGE(2) / ZFFFF7FFF, ZFFFFFFFF /
C     DATA RIGHT(1), RIGHT(2) / Z00002480, Z00000000 /
C     DATA DIVER(1), DIVER(2) / Z00002500, Z00000000 /
C     DATA LOG10(1), LOG10(2) / Z209A3F9A, ZCFF884FB /
C
C   MACHINE CONSTANTS FOR VAX 11/780 (G-FLOATING)
C     (EXPRESSED IN INTEGER AND HEXADECIMAL)
C    *** THE INTEGER FORMAT SHOULD BE OK FOR UNIX SYSTEMS***
C
C     DATA SMALL(1), SMALL(2) /        16,           0 /
C     DATA LARGE(1), LARGE(2) /    -32769,          -1 /
C     DATA RIGHT(1), RIGHT(2) /     15552,           0 /
C     DATA DIVER(1), DIVER(2) /     15568,           0 /
C     DATA LOG10(1), LOG10(2) /  1142112243, 2046775455 /
C
C    ***THE HEX FORMAT BELOW MAY NOT BE SUITABLE FOR UNIX SYSYEMS***
C     DATA SMALL(1), SMALL(2) / Z00000010, Z00000000 /
C     DATA LARGE(1), LARGE(2) / ZFFFF7FFF, ZFFFFFFFF /
C     DATA RIGHT(1), RIGHT(2) / Z00003CC0, Z00000000 /
C     DATA DIVER(1), DIVER(2) / Z00003CD0, Z00000000 /
C     DATA LOG10(1), LOG10(2) / Z44133FF3, Z79FF509F /
C
C
C***FIRST EXECUTABLE STATEMENT  D1MACH
      IF (I .LT. 1  .OR.  I .GT. 5)
     1   CALL XERROR( 'D1MACH -- I OUT OF BOUNDS',25,1,2)
C
      D1MACH = DMACH(I)
      RETURN
C
      END
      INTEGER FUNCTION I1MACH(I)
C***BEGIN PROLOGUE  I1MACH
C***DATE WRITTEN   750101   (YYMMDD)
C***REVISION DATE  840405   (YYMMDD)
C***CATEGORY NO.  R1
C***KEYWORDS  MACHINE CONSTANTS
C***AUTHOR  FOX, P. A., (BELL LABS)
C           HALL, A. D., (BELL LABS)
C           SCHRYER, N. L., (BELL LABS)
C***PURPOSE  Returns integer machine dependent constants
C***DESCRIPTION
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C   These machine constant routines must be activated for
C   a particular environment.
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     I1MACH can be used to obtain machine-dependent parameters
C     for the local machine environment.  It is a function
C     subroutine with one (input) argument, and can be called
C     as follows, for example
C
C          K = I1MACH(I)
C
C     where I=1,...,16.  The (output) value of K above is
C     determined by the (input) value of I.  The results for
```

```
C     various values of I are discussed below.
C
C  I/O unit numbers.
C    I1MACH( 1) = the standard input unit.
C    I1MACH( 2) = the standard output unit.
C    I1MACH( 3) = the standard punch unit.
C    I1MACH( 4) = the standard error message unit.
C
C  Words.
C    I1MACH( 5) = the number of bits per integer storage unit.
C    I1MACH( 6) = the number of characters per integer storage unit.
C
C  Integers.
C    assume integers are represented in the S-digit, base-A form
C
C               sign ( X(S-1)*A**(S-1) + ... + X(1)*A + X(0) )
C
C               where 0 .LE. X(I) .LT. A for I=0,...,S-1.
C    I1MACH( 7) = A, the base.
C    I1MACH( 8) = S, the number of base-A digits.
C    I1MACH( 9) = A**S - 1, the largest magnitude.
C
C  Floating-Point Numbers.
C    Assume floating-point numbers are represented in the T-digit,
C    base-B form
C               sign (B**E)*( (X(1)/B) + ... + (X(T)/B**T) )
C
C               where 0 .LE. X(I) .LT. B for I=1,...,T,
C               0 .LT. X(1), and EMIN .LE. E .LE. EMAX.
C    I1MACH(10) = B, the base.
C
C  Single-Precision
C    I1MACH(11) = T, the number of base-B digits.
C    I1MACH(12) = EMIN, the smallest exponent E.
C    I1MACH(13) = EMAX, the largest exponent E.
C
C  Double-Precision
C    I1MACH(14) = T, the number of base-B digits.
C    I1MACH(15) = EMIN, the smallest exponent E.
C    I1MACH(16) = EMAX, the largest exponent E.
C
C  To alter this function for a particular environment,
C  the desired set of DATA statements should be activated by
C  removing the C from column 1.  Also, the values of
C  I1MACH(1) - I1MACH(4) should be checked for consistency
C  with the local operating system.
C***REFERENCES  FOX P.A., HALL A.D., SCHRYER N.L.,*FRAMEWORK FOR A
C                 PORTABLE LIBRARY*, ACM TRANSACTIONS ON MATHEMATICAL
C                 SOFTWARE, VOL. 4, NO. 2, JUNE 1978, PP. 177-188.
C***ROUTINES CALLED  (NONE)
C***END PROLOGUE  I1MACH
C
      INTEGER IMACH(16),OUTPUT
      EQUIVALENCE (IMACH(4),OUTPUT)
C
C
```

```
C      MACHINE CONSTANTS FOR THE CDC CYBER 170 SERIES (FTN5).
C
C       DATA IMACH( 1) /    5 /
C       DATA IMACH( 2) /    6 /
C       DATA IMACH( 3) /    7 /
C       DATA IMACH( 4) /    6 /
C       DATA IMACH( 5) /   60 /
C       DATA IMACH( 6) /   10 /
C       DATA IMACH( 7) /    2 /
C       DATA IMACH( 8) /   48 /
C       DATA IMACH( 9) / O"00007777777777777777" /
C       DATA IMACH(10) /    2 /
C       DATA IMACH(11) /   48 /
C       DATA IMACH(12) / -974 /
C       DATA IMACH(13) / 1070 /
C       DATA IMACH(14) /   96 /
C       DATA IMACH(15) / -927 /
C       DATA IMACH(16) / 1070 /
C
C      MACHINE CONSTANTS FOR THE CDC CYBER 200 SERIES
C
C      DATA IMACH( 1) /       5 /
C      DATA IMACH( 2) /       6 /
C      DATA IMACH( 3) /       7 /
C      DATA IMACH( 4) /       6 /
C      DATA IMACH( 5) /      64 /
C      DATA IMACH( 6) /       8 /
C      DATA IMACH( 7) /       2 /
C      DATA IMACH( 8) /      47 /
C      DATA IMACH( 9) / X'00007FFFFFFFFFFF' /
C      DATA IMACH(10) /       2 /
C      DATA IMACH(11) /      47 /
C      DATA IMACH(12) / -28625 /
C      DATA IMACH(13) /  28718 /
C      DATA IMACH(14) /      94 /
C      DATA IMACH(15) / -28625 /
C      DATA IMACH(16) /  28718 /
C
C
C      MACHINE CONSTANTS FOR THE CDC 6000/7000 SERIES.
C
C      DATA IMACH( 1) /    5 /
C      DATA IMACH( 2) /    6 /
C      DATA IMACH( 3) /    7 /
C      DATA IMACH( 4) /6LOUTPUT/
C      DATA IMACH( 5) /   60 /
C      DATA IMACH( 6) /   10 /
C      DATA IMACH( 7) /    2 /
C      DATA IMACH( 8) /   48 /
C      DATA IMACH( 9) / 00007777777777777777B /
C      DATA IMACH(10) /    2 /
C      DATA IMACH(11) /   47 /
C      DATA IMACH(12) / -929 /
C      DATA IMACH(13) / 1070 /
C      DATA IMACH(14) /   94 /
C      DATA IMACH(15) / -929 /
C      DATA IMACH(16) / 1069 /
```

```
C
C     MACHINE CONSTANTS FOR THE CRAY 1
C
C     DATA IMACH( 1) /   100 /
C     DATA IMACH( 2) /   101 /
C     DATA IMACH( 3) /   102 /
C     DATA IMACH( 4) /   101 /
C     DATA IMACH( 5) /    64 /
C     DATA IMACH( 6) /     8 /
C     DATA IMACH( 7) /     2 /
C     DATA IMACH( 8) /    63 /
C     DATA IMACH( 9) /  777777777777777777777B /
C     DATA IMACH(10) /     2 /
C     DATA IMACH(11) /    47 /
C     DATA IMACH(12) / -8189 /
C     DATA IMACH(13) /  8190 /
C     DATA IMACH(14) /    94 /
C     DATA IMACH(15) / -8099 /
C     DATA IMACH(16) /  8190 /
C
C
C     MACHINE CONSTANTS FOR THE IBM 360/370 SERIES,
C     THE XEROX SIGMA 5/7/9, THE SEL SYSTEMS 85/86, AND
C     THE PERKIN ELMER (INTERDATA) 7/32.
C
C     DATA IMACH( 1) /   5 /
C     DATA IMACH( 2) /   6 /
C     DATA IMACH( 3) /   7 /
C     DATA IMACH( 4) /   6 /
C     DATA IMACH( 5) /  32 /
C     DATA IMACH( 6) /   4 /
C     DATA IMACH( 7) /  16 /
C     DATA IMACH( 8) /  31 /
C     DATA IMACH( 9) / Z7FFFFFFF /
C     DATA IMACH(10) /  16 /
C     DATA IMACH(11) /   6 /
C     DATA IMACH(12) / -64 /
C     DATA IMACH(13) /  63 /
C     DATA IMACH(14) /  14 /
C     DATA IMACH(15) / -64 /
C     DATA IMACH(16) /  63 /
C
C     MACHINE CONSTANTS FOR THE IBM PC FAMILY (D. KAHANER NBS)
C
      DATA IMACH/5,6,0,6,32,4,2,31,2147483647,2,24,
     * -125,127,53,-1021,1023/
C                 NOTE! I1MACH(3) IS NOT WELL DEFINED AND IS SET TO ZERO.
C
C
C     MACHINE CONSTANTS FOR THE PDP-10 (KA PROCESSOR).
C
C     DATA IMACH( 1) /    5 /
C     DATA IMACH( 2) /    6 /
C     DATA IMACH( 3) /    5 /
C     DATA IMACH( 4) /    6 /
C     DATA IMACH( 5) /   36 /
C     DATA IMACH( 6) /    5 /
```

```
C     DATA IMACH( 7) /    2 /
C     DATA IMACH( 8) /   35 /
C     DATA IMACH( 9) / "377777777777 /
C     DATA IMACH(10) /    2 /
C     DATA IMACH(11) /   27 /
C     DATA IMACH(12) / -128 /
C     DATA IMACH(13) /  127 /
C     DATA IMACH(14) /   54 /
C     DATA IMACH(15) / -101 /
C     DATA IMACH(16) /  127 /
C
C     MACHINE CONSTANTS FOR THE PDP-10 (KI PROCESSOR).
C
C     DATA IMACH( 1) /    5 /
C     DATA IMACH( 2) /    6 /
C     DATA IMACH( 3) /    5 /
C     DATA IMACH( 4) /    6 /
C     DATA IMACH( 5) /   36 /
C     DATA IMACH( 6) /    5 /
C     DATA IMACH( 7) /    2 /
C     DATA IMACH( 8) /   35 /
C     DATA IMACH( 9) / "377777777777 /
C     DATA IMACH(10) /    2 /
C     DATA IMACH(11) /   27 /
C     DATA IMACH(12) / -128 /
C     DATA IMACH(13) /  127 /
C     DATA IMACH(14) /   62 /
C     DATA IMACH(15) / -128 /
C     DATA IMACH(16) /  127 /
C
C
C     MACHINE CONSTANTS FOR THE SUN-3 (INCLUDES THOSE WITH 68881 CHIP,
C       OR WITH FPA BOARD. ALSO INCLUDES SUN-2 WITH SKY BOARD. MAY ALSO
C       WORK WITH SOFTWARE FLOATING POINT ON EITHER SYSTEM.)
C
C     DATA IMACH( 1) /    5 /
C     DATA IMACH( 2) /    6 /
C     DATA IMACH( 3) /    6 /
C     DATA IMACH( 4) /    0 /
C     DATA IMACH( 5) /   32 /
C     DATA IMACH( 6) /    4 /
C     DATA IMACH( 7) /    2 /
C     DATA IMACH( 8) /   31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /    2 /
C     DATA IMACH(11) /   24 /
C     DATA IMACH(12) / -125 /
C     DATA IMACH(13) /  128 /
C     DATA IMACH(14) /   53 /
C     DATA IMACH(15) / -1021 /
C     DATA IMACH(16) /  1024 /
C
C
C     MACHINE CONSTANTS FOR THE VAX 11/780
C
C     DATA IMACH(1) /    5 /
```

```
C     DATA IMACH(2) /    6 /
C     DATA IMACH(3) /    5 /
C     DATA IMACH(4) /    6 /
C     DATA IMACH(5) /   32 /
C     DATA IMACH(6) /    4 /
C     DATA IMACH(7) /    2 /
C     DATA IMACH(8) /   31 /
C     DATA IMACH(9) /2147483647 /
C     DATA IMACH(10)/    2 /
C     DATA IMACH(11)/   24 /
C     DATA IMACH(12)/ -127 /
C     DATA IMACH(13)/  127 /
C     DATA IMACH(14)/   56 /
C     DATA IMACH(15)/ -127 /
C     DATA IMACH(16)/  127 /
C
C***FIRST EXECUTABLE STATEMENT  I1MACH
      IF (I .LT. 1  .OR.  I .GT. 16)
     1   CALL XERROR ( 'I1MACH -- I OUT OF BOUNDS',25,1,2)
C
      I1MACH=IMACH(I)
      RETURN
C
      END
C
      SUBROUTINE XERROR(MESSG,NMESSG,NERR,LEVEL)
C***BEGIN PROLOGUE  XERROR
C***DATE WRITTEN   790801   (YYMMDD)
C***REVISION DATE  870930   (YYMMDD)
C***CATEGORY NO.  R3C
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Processes an error (diagnostic) message.
C***DESCRIPTION
C    From the book "Numerical Methods and Software"
C       by  D. Kahaner, C. Moler, S. Nash
C           Prentice Hall 1988
C     Abstract
C        XERROR processes a diagnostic message. It is a stub routine
C        written for the book above. Actually, XERROR is a sophisticated
C        error handling package with many options, and is described
C        in the reference below. Our version has the same calling
sequence
C        but only prints an error message and either returns (if the
C        input value of DABS(LEVEL) is less than 2) or stops (if the
C        input value of DABS(LEVEL) equals 2).
C
C     Description of Parameters
C      --Input--
C        MESSG - the Hollerith message to be processed.
C        NMESSG- the actual number of characters in MESSG.
C                (this is ignored in this stub routine)
C        NERR  - the error number associated with this message.
C                NERR must not be zero.
C                (this is ignored in this stub routine)
C        LEVEL - error category.
C                =2 means this is an unconditionally fatal error.
```

```
C                  =1 means this is a recoverable error.  (I.e., it is
C                     non-fatal if XSETF has been appropriately called.)
C                  =0 means this is a warning message only.
C                  =-1 means this is a warning message which is to be
C                     printed at most once, regardless of how many
C                     times this call is executed.
C                   (in this stub routine
C                         LEVEL=2 causes a message to be printed and then
a
C                                              stop.
C                         LEVEL<2 causes a message to be printed and then
a
C                                              return.
C
C     Examples
C        CALL XERROR('SMOOTH -- NUM WAS ZERO.',23,1,2)
C        CALL XERROR('INTEG  -- LESS THAN FULL ACCURACY ACHIEVED.',
C                    43,2,1)
C        CALL XERROR('ROOTER -- ACTUAL ZERO OF F FOUND BEFORE INTERVAL F
C    1ULLY COLLAPSED.',65,3,0)
C        CALL XERROR('EXP    -- UNDERFLOWS BEING SET TO ZERO.',39,1,-1)
C
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  XERRWV
C***END PROLOGUE  XERROR
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      CHARACTER*(*) MESSG
C***FIRST EXECUTABLE STATEMENT  XERROR
      CALL XERRWV(MESSG,NMESSG,NERR,LEVEL,0,0,0,0,0.,0.)
      RETURN
      END
      SUBROUTINE XERRWV(MESSG,NMESSG,NERR,LEVEL,NI,I1,I2,NR,R1,R2)
C***BEGIN PROLOGUE  XERRWV
C***DATE WRITTEN   800319   (YYMMDD)
C***REVISION DATE  870930   (YYMMDD)
C***CATEGORY NO.  R3C
C***KEYWORDS  ERROR,XERROR PACKAGE
C***AUTHOR  JONES, R. E., (SNLA)
C***PURPOSE  Processes error message allowing 2 integer and two real
C             values to be  included in the message.
C***DESCRIPTION
C    From the book "Numerical Methods and Software"
C       by  D. Kahaner, C. Moler, S. Nash
C           Prentice Hall 1988
C     Abstract
C        XERRWV prints a diagnostic error message.
C        In addition, up to two integer values and two real
C        values may be printed along with the message.
C        A stub routine for the book above. The actual XERRWV is
described
C        in the reference below and contains many other options.
C
C     Description of Parameters
C      --Input--
C        MESSG - the Hollerith message to be processed.
```

```
C        NMESSG- the actual number of characters in MESSG.
C                (ignored in this stub)
C        NERR  - the error number associated with this message.
C                NERR must not be zero.
C                (ignored in this stub)
C        LEVEL - error category.
C                =2 means this is an unconditionally fatal error.
C                =1 means this is a recoverable error.  (I.e., it is
C                   non-fatal if XSETF has been appropriately called.)
C                =0 means this is a warning message only.
C                =-1 means this is a warning message which is to be
C                   printed at most once, regardless of how many
C                   times this call is executed.
C                  (in this stub LEVEL=2 causes an error message to be
C                                          printed followed by a stop,
C                               LEVEL<2 causes an error message to be
C                                          printed followed by a
return.)
C        NI    - number of integer values to be printed. (0 to 2)
C        I1    - first integer value.
C        I2    - second integer value.
C        NR    - number of real values to be printed. (0 to 2)
C        R1    - first real value.
C        R2    - second real value.
C
C     Examples
C        CALL XERRWV('SMOOTH -- NUM (=I1) WAS ZERO.',29,1,2,
C    1   1,NUM,0,0,0.,0.)
C        CALL XERRWV('QUADXY -- REQUESTED ERROR (R1) LESS THAN MINIMUM (
C    1R2).,54,77,1,0,0,0,2,ERRREQ,ERRMIN)
C
C***REFERENCES  JONES R.E., KAHANER D.K., "XERROR, THE SLATEC ERROR-
C                 HANDLING PACKAGE", SAND82-0800, SANDIA LABORATORIES,
C                 1982.
C***ROUTINES CALLED  (NONE)
C***END PROLOGUE  XERRWV
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      CHARACTER*(*) MESSG
C***FIRST EXECUTABLE STATEMENT  XERRWV
      WRITE(*,*) MESSG
      IF(NI.EQ.2)THEN
        WRITE(*,*) I1,I2
      ELSEIF(NI.EQ.1) THEN
        WRITE(*,*) I1
      ENDIF
      IF(NR.EQ.2) THEN
        WRITE(*,*) R1,R2
      ELSEIF(NR.EQ.1) THEN
        WRITE(*,*) R1
      ENDIF
      IF(ABS(LEVEL).LT.2)RETURN
      STOP
      END
```

Technical Reports
List of SKB Technical Reports 1993

TR 93-14
Simulation of the European ice sheet trough the last glacial cycle and prediction of future glaciation
G S Boulton, A Payne
Department of Geology and Geophysics, Edinburgh Uneversity, Grant Institute, Edinburgh, United Kingdom
December 1992

TR 93-15
Analysis of the regional groundwater flow in the Finnsjön area
Anders Boghammar, Bertil Grundfelt, Hans Widén
Kemakta Konsult AB
June 1993

TR 93-16
Kinetic modelling of bentonite - canister interaction.
Implications for Cu, Fe, and Pb corrosion in a repository for spent nuclear fuel
Paul Wersin, Jordi Bruno, Kastriot Spahiu
MTB Tecnologia Ambiental, Cerdanyola, Spain
June 1993

TR 93-17
Oxidation of uraninite
Janusz Janeczek, Rodney C Ewing
Department of Earth & Planetary Science, University of New Mexico, Albuquerque, NM, USA
June 1993

TR 93-18
Solubility of the redox-sensitive radionuclides ^{99}Tc and ^{237}Np under reducing conditions in neutral to alkaline solutions. Effect of carbonate
Trygve E Eriksen[1], Pierre Ndalamba[1], Daqing Cui[1], Jordi Bruno[2], Marco Caceci[2], Kastriot Spahiu[2]
[1] Dept. of Nuclear Chemistry, Royal Institute of Technology, Stockholm, Sweden
[2] MBT Tecnologia Ambiental, Cerdanyola, Spain
September 1993

TR 93-19
Mechanical properties of fracture zones
Bengt Lejon
Conterra AB
May 1993

TR 93-20
The Fracture Zone Project - Final report
Peter Andersson (ed.)
Geosigma AB, Uppsala, Sweden
September 1993