1/70

SOFTWARE RELEASE NOTICE

01. SRN Number: **GHGC-SRN-167**

02. Project Title: **Near Field Environment KTI** | Project No.: **20-5708-562**

03. SRN Title: **MULTIFLO Version 1.2b**

04. Originator/Requestor: **Bruce Mabrito** | Date: **3/31/00**

05. Summary of Actions

- ☐ Release of new software
- ☐ Release of modified software:
  - ☐ Enhancements made
  - ☐ Corrections made
- ☐ Change of access software
- ■ Software Retirement

06. Persons Authorized Access

Name	RO/RW	A/C/D

07. Element Manager Approval: **English Pearcy** Date: 3/31/00

08. Remarks:

CNWRA Form TOP-6 (06/95)

2/10

SOFTWARE RELEASE NOTICE

01. SRN Number: GHGC-SRN-~~130~~ 167 BSM 3/24/98

02. Project Title: Near-Field Environment KTI	Project No. 20-5708-562
03. SRN Title: MULTIFLO Version 1.2b	
04. Originator/Requestor: Bruce Mabrito	Date: 03/03/98

05. Summary of Actions

- ☐ Release of new software
- ■ Release of modified software:
- ■ Enhancements made: DCM, ~~Unstructured Gind, MINC.~~ E.C. 3/24/98
- ■ Corrections made: Minor bugs corrected.
- ☐ Change of access software
- ☐ Software Retirement

06. Persons Authorized Access

Name	RO/RW	A/C/D
Peter Lichtner	RW	
Mohan Seth	RW	
Narasi Sridhar	RO	
Brett Leslie	RO	
English Pearcy	RO	

07. Element Manager Approval: E.C. Date: 3/24/98

08. Remarks:

CNWRA Form TOP-6 (06/95)

3/50

SOFTWARE SUMMARY FORM

01. Summary Date: 03/12/98	02. Summary prepared by (Name and phone) Peter C. Lichtner, 522-6084	03. Summary Action: New
04. Software Date: 03/12/98	05. Short Title: MULTIFLO Version 1.2b	

06. Software Title: MULTIFLO Version 1.2b	07. Internal Software ID: NONE

08. Software Type:	09. Processing Mode:	10. APPLICATION AREA
☐ Automated Data System	☐ Interactive	a. General: ■ Scientific/Engineering ■ Auxiliary Analyses ☐ Total System PA
■ Computer Program	☐ Batch	☐ Subsystem PA ☐ Other
☐ Subroutine/Module	■ Combination	b. Specific: Groundwater multiphase flow and reactive transport model

11. Submitting Organization and Address:	12. Technical Contact(s) and Phone:
CNWRA 6220 Culebra Road San Antonio, TX 78228	Peter Lichtner, (210) 522-6084 Mohan Seth, (972) 699-3610

13. Narrative:
The code is used to model multiphase groundwater flow and reactive transport.

14. Computer Platform	15. Computer Operating System:	16. Programming Language(s):	17. Number of Source Program Statements:
SUN	UNIX	Fortran 77	~64,000
18. Computer Memory Requirements:	**19. Tape Drives:**	**20. Disk/Drum Units:**	**21. Graphics:**
Problem Dependent	N/A	N/A	ASCII plot data files

22. Other Operational Requirements
Thermodynamic database required.

23. Software Availability:	24. Documentation Availability:
☐ Available ■ Limited ☐ In-House ONLY	☐ Available ☐ Inadequate ■ In-House ONLY DRAFT

Software Custodian: Date: 3/23/98

CNWRA Form TOP-4-1

SOFTWARE REQUIREMENTS DESCRIPTION

21/170

SOFTWARE REQUIREMENTS DESCRIPTION FOR THE COMPUTER CODE MULTIFLO VERSION 1.2

Prepared for

Nuclear Regulatory Commission
Contract NRC-02-97-009

Prepared by

Peter C. Lichtner

Center for Nuclear Waste Regulatory Analyses
San Antonio, Texas

March 1998

22/70

ABSTRACT

This Software Requirements Description presents proposed revisions to the computer code MULTIFLO to implement a dual continuum model (DCM) capability. The DCM will be used in place of the equivalent continuum model in thermal-hydrological-chemical modeling of the proposed high-level nuclear waste repository at Yucca Mountain, Nevada.

23/70

CONTENTS

24/30

TABLES

25/30

ACKNOWLEDGMENTS

This report was prepared to document work performed by the Center for Nuclear Waste Regulatory Analyses (CNWRA) for the Nuclear Regulatory Commission (NRC) under Contract No. NRC-02-97-009. The activities reported here were performed on behalf of the NRC Office of Nuclear Material Safety and Safeguards (NMSS), Division of Waste Management (DWM). The report is an independent product of the CNWRA and does not necessarily reflect the views or regulatory position of the NRC.

QUALITY OF DATA, ANALYSES, AND CODE DEVELOPMENT

DATA: CNWRA-generated original data contained in this report meets quality assurance requirements described in the CNWRA Quality Assurance Manual. Sources for other data should be consulted for determining the level of quality for those data.

ANALYSES AND CODES: No analyses work is reported in this document. This document describes planned changes to software.

1 INTRODUCTION

This Software Requirements Description (SRD) document describes proposed revision to the computer code MULTIFLO, a numerical model describing multiphase, multicomponent, reactive transport in a variably saturated porous medium. This software could be used in the high-level waste (HLW) repository license application review process for Yucca Mountain (YM).

The code can be used to address the very-near-field (drift scale), and near-field (repository scale) performance of the repository. The code can be applied to such processes as:

(i) isothermal and nonisothermal liquid and vapor phase movement of water through unsaturated rock at YM.

(ii) predicting the evolution of groundwater compositions near and within the engineered barrier system.

(iii) predicting changes in porosity and permeability of the host rock resulting from mineral alteration and their effect on fluid transport.

(iv) prediction of transport of aqueous and gaseous radionuclides from the waste package.

2 SOFTWARE REQUIREMENT DESCRIPTION: MULTIFLO, VERSION 1.2

This SRD briefly outlines the software function, technical basis, and computational approach, that are relevant to the proposed enhancements of the code MULTIFLO. Version 1.0 of MULTIFLO has been completed and satisfies TOP-018 QA requirements. A SRD was completed for Version 1.0 of MULTIFLO. A draft version of the User's Manual has been completed. The revised code will be issued as Version 1.2.

2.1 SOFTWARE FUNCTION

Planned change to the code MULTIFLO is to include a dual continuum model (DCM) capability. The DCM will be used both to replace and enhance the multiple interacting continua (MINC) approach for modeling highly fractured porous rock at the YM proposed repository site. The MINC method is currently being programmed into both METRA and GEM modules which will be released as Version 2.0 of MULTIFLO. The programming language used in MULTIFLO is FORTRAN. The code will be developed on a Sun-sparc workstation and PCs running NEXTSTEP and PC-UNIX.

2.2 BACKGROUND

The main purpose of this revision to MULTIFLO to incorporate the DCM is to aid in providing a detailed model of the near-field environment from which total performance assessment analyses may be abstracted. Both Lawrence Livermore National Laboratory (LLNL) and Lawrence Berkeley National Laboratory (LBNL) have begun using the DCM almost exclusively in place of the equivalent continuum model (ECM). The ECM is based on the assumption of capillary equilibrium between matrix and

fractures which is much too stringent to explain field observations of pore water chemistry at YM, including recent observations of ^{36}Cl and differences in matrix and fracture solution chemistry. Observations of ^{36}Cl at the proposed repository horizon indicate the existence of fast pathways from the ground surface to the watertable which are presumed related to flow through fractures. To describe such situations it is important to be able to distinguish between fracture and matrix flow systems.

Two available alternative approaches to the ECM, one the DCM and the other the MINC model (Pruess and Narisimhan, 1980), have been applied to YM. The DCM is applicable to the case where the matrix forms a connected flow region unobstructed by fractures. The MINC model on the other hand applies when matrix blocks are disconnected from one another by the presence of through-going fractures. Thus the two models are complementary to one another. In particular, the MINC model is not a generalization of the DCM, but is applicable to large-scale fractures in contrast to the DCM which is applicable to rocks with a high fracture density such as characterize parts of YM. Future models for YM could employ the MINC for large-scale fractures and use the DCM to represent matrix blocks within the MINC formulation.

The added capability of the DCM in MULTIFLO will enable evaluation of DOE's current DCM modeling effort. Because many of the thermal-hydrologic aspects of DOE's Total System Performance Assessment for the Viability Assessment will be based on the DCM rather than the ECM, it is important for the CNWRA to also have an independent capability to conduct effective reviews. This is especially true because of the greater flexibility and additional parameter requirements of the DCM.

2.3 CURRENT STATUS OF MULTIFLO

The current status of MULTIFLO is as follows:

- Unstructured grid has been completed in METRA and GEM but is not fully tested
- Programming MINC into METRA has been initiated, but not completed
- Operator splitting is not implemented with the unstructured grid version of GEM
- The MULTIFLO User's Manual is incomplete with respect to MINC and the unstructured grid

2.4 DUAL CONTINUUM MODEL IMPLEMENTATION IN MULTIFLO

The DCM represents a fractured porous medium as two interacting continua: one continuum represents the fracture network and the other the rock matrix. In the case of solute transport a linear coupling term describes mass transfer between the two continua. For partially saturated systems the coupling is a nonlinear function of the saturation and is linear in the pressure difference between matrix and fracture network. The DCM is presumed valid provided the rock mass contains fractures which are connected to form a continuous flow network, typical of rock with a high density of fractures which are closely spaced. The matrix must also form a connected flow regime. For a system with widely spaced continuous fractures which isolate matrix blocks thereby disrupting their continuity, the dual continuum approach is not valid and an explicit representation of each fracture or a multiple interacting continua model MINC approach may be necessary.

Flow equations for the DCM consist of separate mass conservation equations for the matrix and fracture. As a result it necessary to solve twice the number of equations compared to a single continuum model. The implementation of the DCM into MULTIFLO is relatively easy because the complete structure for a single continuum is already in place. Furthermore, the coupling terms are linear in pressure or concentration difference between matrix and fracture.

2.5 RELATED CODES

The code DCM3D (Updegraff et al., 1991) applies the DCM model to unsaturated flow. However, the code applies only to isothermal conditions and uses incorrect coupling terms between fracture and matrix which depend only on matrix and not fracture properties. Codes used by LBNL which incorporate the DCM, such as TOUGH and its derivatives (Pruess, 1989), are not currently available. The NUFT code (Nitao, 1996) incorporates the same general approach to the DCM as envisaged for MULTIFLO. This code may be available in the future for comparison and benchmarking with MULTIFLO.

3 IMPLEMENTATION

It is proposed to begin work on the DCM immediately, postponing further work on MINC until the DCM is completed and implemented for the following reasons:

- An immediate need exists for an alternative model to replace the ECM
- The DCM requires far less programming effort compared to the MINC implementation
- MINC may not be applicable to small-scale fractures at YM which constitute the bulk of the rock mass. This type of geometry may be better described by the DCM

An estimate of time and effort involved in the planned developmental work and the order in which the work will be performed is provided in table 3-1. Although, the time for some tasks may exceed the individual estimates, the total time should represent a good estimate. This time includes debugging time which adds additional uncertainty. Work will be carried out by M. Seth under supervision by P. Lichtner.

Table 3-1. Estimation of DCM implementation time

Task	Description	Time (hrs)	
		METRA	GEM
I	DCM Coding	60	60
II	Testing	40	40
III	Revise User's Manual	16	16
Total		116	116

29/70

4 REFERENCES

Nitao, J.J. 1996. *Reference Manual for the NUFT Flow and Transport Code, Version 1.0.* UCRL-ID-113520. Lawrence Livermore National Laboratory: Lawrence Livermore, CA.

Pruess, K., and T.N. Narisimhan. 1985. A practical method for modeling fluid and heat flow in fractured porous media. Society of Petroleum Engineers 25(1): 14–27.

Pruess, K. 1991. TOUGH2: *A General—Purpose Numerical Simulator for Multiphase Fluid and Heat Flow.* LBL-29400. Lawrence Berkeley Laboratory: Berkeley, CA.

Updegraff, C.D., C.E. Lee, D.P. Gallego. 1991. *DCM3D: A Dual-Continuum, Three-Dimensional, Groundwater Flow Code for Unsaturated, Fractured Porous Media.* NUREG/CR-5536. Washington, DC: Nuclear Regulatory Commission.

CENTER FOR NUCLEAR WASTE REGULATORY ANALYSES

DESIGN VERIFICATION REPORT FOR CNWRA SOFTWARE: MULTIFLO Version 1.2b

---

March 3, 1998

MULTIFLO (Scientific and Engineering Software) Version 1.2b

NOTE: This version of the MULTIFLO Software contains changes from the previous version 1.0 released April 4, 1997. There was a MULTIFLO Version 1.1 that was put on 8 mm tape and retained in the QA Records Room, however there was no formal release of that version. This Version 1.2b incorporates the modifications for the dual continuum model in the GEM and METRA modules, along with minor "bug fixes."

1. Scientific Notebook Documentation: Changes to the MULTIFLO code have been made utilizing input from P. Lichtner and M. Seth, and the CNWRA electronic scientific notebook No. 095, assigned to P. Lichtner, was used to document the changes.

2. Programming Language: ANSI Standard FORTRAN 77 confirmed by the Software Custodian. See attached hard copies of MULTIFLO Version 1.2b pages as examples.

3. Internal Documentation: On 3/3/98, B. Mabrito reviewed portions of the MULTIFLO Version 1.2b scientific and engineering software on the "Gravenstein" workstation (PC, Next Step OS Version 3.3, 133 MHz) in P. Lichtner's office. The MULTIFLO Version 1.2b software is located at /home/skippy/Lichtner/masstrans/mflo_dcm on the "Skippy" server. This version of the MULTIFLO code contained substantial comments and a hard copy page showing representative comments is attached to this Design Verification Report. The internal documentation comments in MULTIFLO Version 1.2b meet the requirements of TOP-018, Section 5.4.4.

4. Software Labels and Data

a. Header Data and Format: MULTIFLO Version 1.2b header data and the format were compared against TOP-018 Section 5.4.6 and found generally acceptable (see several of the attached hard copy pages), with the exception of there not being a list of the Software Problem Change Request numbers. This is because the scientific notebook method of documentation alone was utilized in modification of this software, which is more appropriate since the new version of the code represents substantial new development as outlined in the SRD, rather than just fixes to the original code.

b. NRC Data: MULTIFLO Version 1.2b NRC data and the format were compared against TOP-018, Section 5.4.6, third bullet and found to be acceptable, except that no specific NRC staff member is identified in the main METRA or GEM headers. Such a contact is identified in the main MULTIFLO program as B. Leslie at the NRC.

c. Source Code Header: MULTIFLO Version 1.2b header data was compared to TOP-018 Section 5.4.6, fourth bullet, and found to meet the requirements, except where noted above that no SPCRs were utilized in the modification of this version and changes are documented in an electronic scientific notebook.

5. Unique Run Identification: Output runs of MULTIFLO Version 1.2b show a unique identifier on the top of each output file when printed out, however the plot files of MULTIFLO do not have such an identifier since this is a beta version.

6. Software Analysis and Results: Software analysis tools (FOR_STUDY) were run by P. Lichtner on the MULTIFLO Version 1.2b code and changes were made as the beta version was developed by the software developer.

3/3/98	3/3/98
Peter Lichtner Date	**Bruce Mabrito Date**
CNWRA MULTIFLO Software Developer	**CNWRA Software Custodian**

Attachments/
Original to: Software Folder
cc: CNWRA Software Developer
Cognizant EM

```
c*file  mainmet.f

c Program Name:          MULTIFLO/METRA
c File/Subroutine Names: mainmet.f/mainmet.f frfmt.f cputim.f second.f
c Release Date:          March, 1998
c Release Version:       1.2 Beta
c Client Name:           USNRC
c Contract Number:       NRC 02-93-005
c CNWRA Contact:         Peter C. Lichtner (210-522-6084)
c                        Center for Nuclear Waste Regulatory Analyses
c                        San Antonio, Texas 78238-5166
c                        lichtner@swri.edu
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c VERSION/REVISION HISTORY

c $Id$
c $Log$

c-------------------------------------------------------------------------
c   Date            Author(s)            Comments/Modifications
c-------------------------------------------------------------------------

c  February  97     Mohan S. Seth        Initial Implementation
c                   Peter C. Lichtner

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c DISCLAIMER/NOTICE

c This computer code/material was prepared as an account of work
c performed by the Center for Nuclear Waste Regulatory Analyses (CNWRA)
c for the Division of Waste Management of the Nuclear Regulatory
c Commission (NRC), an independent agency of the United States
c Government. The developer(s) of the code nor any of their sponsors
c make any warranty, expressed or implied, or assume any legal
c liability or responsibility for the accuracy, completeness, or
c usefulness of any information, apparatus, product or process
c disclosed, or represent that its use would not infringe on
c privately-owned rights.

c IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW WILL THE SPONSORS
c OR THOSE WHO HAVE WRITTEN OR MODIFIED THIS CODE, BE LIABLE FOR
c DAMAGES, INCLUDING ANY LOST PROFITS, LOST MONIES, OR OTHER SPECIAL,
c INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR
c INABILITY TO USE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA
c BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY THIRD PARTIES OR A
c FAILURE OF THE PROGRAM TO OPERATE WITH OTHER PROGRAMS) THE PROGRAM,
c EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES,
c OR FOR ANY CLAIM BY ANY OTHER PARTY.

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  PURPOSE:

c     This is main driver program for the entire code. Essentially
c     no computations are performed by this module, but it controls
c     the flow of computations.

c     This module is replaced my metra.f when run in coupled mode with
c     multiflo.

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c    PROGRAMMING LANGUAGE

c    ANSI Standard Fortran - 77
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c    PRECISION

c    Double precison for real varaibles (8 bytes)
c    Integers (4 bytes)

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  INTERFACING ARGUMENTS:

c  Variable name       Type                  Description
c  =============       =====                 ============

c    None                -
c-----------------------------------------------------------------------
c   Externals
c   =========

c   accm        computes accumulatin terms in the jacoby in the absence
c               of vapor pressure lowering
c   accmvp      computes accumulatin terms in the jacoby in the presence
c               of vapor pressure lowering
c   coefs       computes jacobain coefs arising from flux terms in the
c               absence of vapor pressure lowering
c   coefsvp     computes jacobain coefs arising from flux terms in the
c               presence of vapor pressure lowering
c   updtpsk     update primary variables after each newtonian iteration
c               in the absence of vapor pressure lowering
c   updtvpk     update primary variables after each newtonian iteration
c               in the presence of vapor pressure lowering
c   pvth2ox     pvth2o, This is made external to be able to
c                       use different pvt packages. Currently
c                       only one is used.
c   pvt         calc pvt properties fo fluid in the absence of vapor
c               pressure lowering.
c   pvtvp       calc pvt properties fo fluid in the presence of vapor
c               pressure lowering.

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   INTERFACING ROUTINES

c     Calling routines
c     ================

c      None

c   Called routines       Function
c   ================   ===============

c   allot.f      set pointers for dynamic memory management
c   autodt.f     calculates the time step size
c   autostep     calculates the tiem step size using diff. algorithm
c   cond.f       computes temp. for pure conduction system
c   convert.f    converts small case letters to capitals
c   cputim.f     caputres cpu-times
c   frfmt.f      converts free-format to a defined fixed format.
c   iter.f       solve one time step
c   init.f       calculates initial conditions and time invariant coefs
c   inpmetra.f   reads all time-invariant data
c   openfls.f    open files
c   outmetra.f   output the results
c   recdat.f     reads time variant data
c   rstart.f     read/write data for a restart run
c   setbc.f      set boundary conditions at each tiem step
c   slv1p.f      computes pressure distribution for isothermal 1-phase liquid
c   update.f     update the primary variables for the enxt time step
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   INCLUDE FILES

c      Name                              Description
c      ====                              ============

c   'add.h'         addresses (pointers) for dynamic memory allocation.
c   'impl.h'        Declares real variables to real*8 and sets
c                   frequently used constants in common.
c   'com.h'         arrays in common
c   'frfmt.h'       free-format variables
c   'metragem.h'    variables  common to both metra and gem codes.
c   'paramtrs.h'    sets dimension limits for all variables.
c   'pckr.h'        set dimensions for rel. perms and capillary press
c   'scalars.h'     scalars in common
c   'units.h'       input/output unit variables

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   SYSTEM LIBRARY ROUTINES

c      Name                             Description
c      ====                             ===========

c      None

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   OUTPUT UNIT(s)

c   Unit Name(Number)       file name            Description

c      iferr (53)           lfn_err           error message file
c      ifout (52)           lfn_out            normal run output

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   REFERENCES

c   None

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   PROGRAM FLOW


c                                 Begin
c                                   |
c                                   |
c                  ---------------------------------------
c                           print the banner and
c                           read 2 lines of title
c                       not counting any comment lines
c                     or lines skipped by 'skip' keyword
c                  ---------------------------------------
c                                   |
c                                   |
c                  -------------------------------------------
c                  call openfls.f (open the required files)
c                  -------------------------------------------
c                                   |
c                                   |
c                  -------------------------------------------
c                  call cputim (initialize cpu time for the run)
c                  -------------------------------------------
c                                   |
```

```
c                              |
c                  ---------------------------------------------
c                   read restart record number 'nnrst' to resume
c                             the  run
c
c                   if nnrst > 0  call rstart.f (read restart
c                                 data and go to 40
c                           else continue below
c                  ---------------------------------------------
c                              |
c                              |
c                  ---------------------------------------------
c                    call inpmetra.f (read initialization data)
c                  ---------------------------------------------
c                              |
c                              |
c                  ---------------------------------------------
c                   call allot.f (set pointers for work array aa)
c                  ---------------------------------------------
c                              |
c                              |
c                  ---------------------------------------------
c                   call init.f (calc initialization and time
c                               time invariant coefs)
c                  ---------------------------------------------
c                              |
c                              |
c                  ---------------------------------------------
c                   call cputim.f (capture the intialization cpu)
c                  ---------------------------------------------
c                              |
c     40 ----------------------> |
c                              |
c                  ---------------------------------------------
c                      if iend = 4, (run completed) go to 500
c                             else continue below
c                  ---------------------------------------------
c                              |
c     100---------------------> |
c                              |
c                  ---------------------------------------------
c                       call recdat.f (read recurrent data)
c                      if iend = 4, (run completed) go to 500
c                             else continue below
c                  ---------------------------------------------
c                              |
c     200---------------------> |
c                              |
c                  ---------------------------------------------
c                       call autodt.f (calc time step size,
c                           and increment the current time)
c                  ---------------------------------------------
c                              |
c                              |
c                  ---------------------------------------------
c                       call setbc.f (set boundary conditions)
c                  ---------------------------------------------
c                              |
c                              |
c                  ---------------------------------------------
c                   if not a conduction or single phase isothermal
c                   liquid  run then call iter.f (calculates the
c                   solution for one time step)

c                      all update.f (update the primary variables
c                       corresponding to the end of the time step)
c
c                   else if conduction run, call cond.f (calc temp
```

```
c                       distribution at the end of one time step
c                       for a pure heat conduction system)

c                    else if a single phase isothermal liquid run,
c                    call slv1p.f (calc pressure distribution at
c                    the end of one time step for such system

c                    if iautodt > 0, call autostep.f (calculates
c                              the next time step size
c                   ---------------------------------------------
c                                       |
c                                       |
c                   ---------------------------------------------
c                    call outmetra.f (prints the specified output
c                                     results)
c                   ---------------------------------------------
c                                       |
c                                       |
c                   ---------------------------------------------
c                     if iend = 5 (corresponding to the attainment
c                                 of steady state, if specified)
c                      then reset the time, step number etc. zero
c                      and go to 100
c                   ---------------------------------------------
c                                       |
c                                       |
c                   ---------------------------------------------
c                    if iend < 5  then
c                       if time < target time, go to 200
c                       if time = target time, go to 100
c                   ---------------------------------------------
c                                       |
c   500 ---------------------------->   |
c                                       |
c                   ---------------------------------------------
c                       run completed successfully
c                       call cputim (prints the cpu-usage)
c                       close all files
c                   ---------------------------------------------
c                                       |
c                                       |
c                                     stop

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c     program mainmet

      include 'impl.h'
      include 'paramtrs.h'
      include 'metragem.h'
      include 'com.h'
      include 'scalars.h'
      include 'frfmt.h'
      include 'add.h'
      include 'units.h'

      character*4 word

      external  pvt  ,  updtpsk, pvtmnc, pvth2o,
     *          pvtvp,  updtvpk, pvtvpmnc

      save nnxp,nnyp,nnzp
      common/dynamic/aa(1200000)

      data maxax/1200000/

c pvth2o is declared external for allowing other pvt-h2o packages
c in place of pvth2o.f. To fully activate other such routines, simply
```

```
c declare them as externals and pass through the argument list for
c various subroutine calls in this routine (as done for vapor pressure
c lowering.). Of course, add the new routine in the makefile. Ensure
c that the new pvt routine has the same order and number of arguments.

c=======================================================================

      maxaa = maxax

      do i = 1,10
        iopen(i) = 0
      end do

c     open input/output files

      call openfls (1)

      ifbug = ifout    !  special---set ifbug = ifout

      include 'title.h'

      write (ifout,15)

      do k = 1,2
2       read  (ifinp,10) (title(i:i),i=1,80)

        if(title(1:1).eq.':') go to 2
        if(title(1:1).eq.'s'.or.title(1:1).eq.'S'.and.
     .  title(2:2).eq.'k'.or.title(2:2).eq.'K') then

          do i = 1,4
            word(i:i) = title(i:i)
          end do

          call convert(word,4)

          if(word.eq.SKIP) then
            i1 = -1
25          read (itp,60) word
            i1 =i1+1
            call convert(word,4)
            if(word.ne.NOSKIP) go to 25
            write (ifout,50) i1
            go to 2
          else if(word.eq.NOSKIP) then
            go to 2
          endif
        endif
        write (ifout,15) (title(i:i),i=1,80)
      end do
10    format(80a1)
15    format(10x,80a1)
50    format(/1x,30(1h=)/1x,i6,'  Data Lines Skipped'/1x,30(1h=)/)
60    format(a4)
c-----------------------------------------------------------------
c                                  call cpu time at run-start
      call cputim (-2)
      call cputim (-1)
c-----------------------------------------------------------------
c                                 read run-type & restart information

      icode = 1
      ipor  = 0
      call frfmt (ione,itwnty,ione,ifive,izro,izro,itp)
      read (image,20) nnrst

20    format(20x,i5)
      write (ifout,30) nnrst
```

```
30    format (//' *RSTART',10x,'Run started from Restart File #',i3/)

      if (nnrst.gt.0) then

        call rstart (aa,nnrst,1)

        nrst = 0
        iend = 0
        go to 40

      end if

c-------------------------------------------------------------
c                              read initialization input data

      call inpmetra (pvth2o,aa)
      if(igeom.ge.0) call pproc (aa,aa(nbpor+1),aa(nb2+1))

      isolve = 0
c-------------------------------------------------------------
c    calculate the initial distribution and time-invariant quantities.

c .. aa is temporary used here for generating steam-tables, if opted.

        call allot (aa,1)

        call init (aa(iprmx),aa(iprmr),aa(itx),aa(itdx),aa(iarex),
     *  krwk,aa(imaxnc),aa(indcon),aa(incdiag),aa(ir),aa(ir+201),aa,
     *  pvth2o,nb2,nerow)

      call cputim (1)

40    continue


c     do m = ivlx2,ivlx2+nconn-1              ! zero velocity arrays
c       aa(m) = zero
c     enddo

      if(igeom.ge.0) then
        if(nx.eq.1) then
          nnxp = 1
        else
          nnxp = nxp1
        endif
        if(ny.eq.1) then
          nnyp = 1
        else
          nnyp = nyp1
        endif
        if(nz.eq.1) then
          nnzp = 1
        else
          nnzp = nzp1
        endif
      else
        nnxp = nb+1
        nnyp = 1
        nnzp = 1
        nx = nb
        ny = 1
        nz = 1
      endif
c-------------------------------------------------------------
      if(iend.eq.4) go to 500

c             iend = 4 run ends normally
c                  = 1 read new set of recurrent data
```

```
c                  = 0 advance in time steps with no data-change.

c-----------------------------------------------------------------
c                   read time-variant data including time steps

100   call recdat (aa)

c-----------------------------------------------------------------

      if(iend.eq.4) go to 500

c-----------------------------------------------------------------
c                   increment time
c       if (itime .eq. 0) then
c         dtnew = dtt
c         dt    = dtt
c       endif

200   continue
      if(iautodt.gt.0) dtt = dtnew
      call autodt(1,itime)

c-----------------------------------------------------------------
c                                       set boundary conditions
      if(nbc.gt.0) call setbc (pvth2o)

c-----------------------------------------------------------------
c                   get the solution for a time step

      if(ndf.gt.1) then
        if(ivplwr.eq.0) then
c                                  no-vapor pressure lowering case
          call iter (aa(idsol),aa(ir),aa(iaa),aa,aa(ivlx2),aa(ivgx2),
     *         pvt,updtpsk,pvth2o,pvtmnc)
        else
c                                  vapor pressure lowering case

          call iter (aa(idsol),aa(ir),aa(iaa),aa,aa(ivlx2),aa(ivgx2),
     *         pvtvp,updtvpk,pvth2o,pvtvpmnc)
        endif
      else
        if(icond.gt.0) then

c                                pure heat conduction case
          call cond (aa(iaa),aa(ic),aa(ir),aa(idsol),aa(itx),
     *    aa(iarex),aa(incdiag),aa(imaxnc),aa(indcon),aa,nerow)
        else

c                        case of single phase isothermal fluid

          call slv1p (aa(iaa),aa(ic),aa(ir),aa(idsol),aa(itx),
     *               aa(iprmx),aa(ivlx2),aa(iarex),aa(incdiag),
     *               aa(imaxnc),aa(indcon),aa,nb,nerow)
        endif

        if (iautodt.gt.0..and.itime.gt.1)
     *  call autostep (dtnew,aa(ipo),aa(ito),aa(isgo))

        go to 400

      endif
c-----------------------------------------------------------------
c                   calculate auto-time-step size

      if (iautodt.gt.0..and.itime.gt.1) then
        call autostep (dtnew,aa(ipo),aa(ito),aa(isgo))
      endif
c-----------------------------------------------------------------
```

```
c                  update the solution vector

      call update (aa(ipo),aa(ito),aa(isgo))

c-----------------------------------------------------------------
c                   print the desired results

400   call cputim (0)

      call outmetra (aa,aa(ivlx2),aa(ivgx2))

      call cputim (7)

c-----------------------------------------------------------------
      if(itime.ge.maxstep) go to 500

      if(iend.eq.5) then
c...                       steady state run--steady state reached
        time = zero
        dt = zero
        dtnew = zero
        ttarget = zero
        itime = 0
        iend = 0
        dtmndf = dtmn
        dtmxdf = dtmx
      endif
      if(time.lt.ttarget) go to 200
      go to 100

500   continue

c-----------------------------------------------------------------
c                              call cpu time at run-end
      call cputim (10)
      call cputim (20)
c-----------------------------------------------------------------
c     close all files

      do i = 51,60
        close(i)
      end do

      stop
      end

c*file frfmt

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  PURPOSE:

c  This routine is used for free-format reading. It sets a free
c  format input-line into a prespecified format which is then
c  re-read by 'internal read' statement in a given format.

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  INTERFACING ARGUMENTS:

c  Variable name       Type                  Description
c  =============       =====                 ============

c   n1,n2,n3      Scalars, Integer*4      numbers of variables for
c                                         a given field

c   ifld1,ifld2,  Scalars, Integer*4      field-widths for variables
c   and ifld3                             corresponding to n1,n2,n3.
```

```
c
c   itpp          Scalars, Integer*4      unit number for reading
c                                         the input data.
c-----------------------------------------------------------------------
c   Externals
c   =========

c    NONE

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   INTERFACING ROUTINES

c     Calling routines
c     ================

c      metra.f
c  (mainmet.f for standalone metra)
c   inpmetra.f   reads initialization data
c   recdat.f     reads recurrent data

c   Called routines       Function
c   ================   ===============

c   convert.f

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   INCLUDE FILES

c      Name                             Description
c      ====                             ============

c   'impl.h'       Declares real variables to real*8 and sets
c                  frequently used constants in common.
c   'frfmt.h'      free-format variables
c   'units.h'      input/output unit variables

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   SYSTEM LIBRARY ROUTINES

c     Name                            Description
c     ====                            ===========

c     none

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   OUTPUT UNIT(s)

c   Unit Name(Number)       file name            Description

c     iferr (53)            lfn_errs              error msgs
c     itpp  (51)            lfn_runname           input data
c       *                    -                    prints on screen

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   REFERENCES

c   Incorporated by MSS from TSE's program library

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine frfmt (n1,ifld1,n2,ifld2,n3,ifld3,itpp)

      include 'impl.h'
```

```
      include 'frfmt.h'
      include 'units.h'

      dimension images(60)
      character kard1*141,kard*141,word*4

c=======================================================================

10    read (itpp,200) (kard(i:i),i=1,ncol)

      icrd = icrd+1
      if (list.eq.0) go to 30
      write (iferr,20) icrd,(kard(i:i),i=1,80)
20    format (1x,i8,2h $,80a1,1h$)
30    continue

      if (kard(1:1).eq.icmt) go to 10

      do i = 1,4
        word(i:i) = kard(i:i)
      end do

      call convert(word,4)

      if(word.eq.SKIP) then
        i1 = -1
40      read (itpp,100) word
        i1 =i1+1
        call convert(word,4)
        if(word.ne.NOSKIP) go to 40
        write (ifout,50) i1
        go to 10
      else if(word.eq.NOSKIP) then
        go to 10
      endif
50    format(/1x,30(1h=)/1x,i6,'  Data Lines Skipped'/1x,30(1h=)/)
100   format(a4)


 60   nvar = n1+n2+n3
      do i=1,n1
        images(i) = ifld1
      end do

      nfld = ifld1*n1
      if (n2.eq.0) go to 110
      i1 = n1+1
      i2 = n1+n2
      do i=i1,i2
        images(i) = ifld2
      end do

      nfld = nfld+ifld2*n2
      if (n3.eq.0) go to 110
      i1 = i2+1
      i2 = i2+n3
      do i=i1,i2
        images(i) = ifld3
      end do

      nfld = nfld+ifld3*n3
110   continue

      do i=1,nfld
        kard1(i:i) = iblank
      end do

      kard(ncol+1:ncol+1) = iblank
```

```
      i2 = 1
      ll = 0
      i1 = 1

      do 120 m=1,nvar
      nsize = 0
      do i=i1,ncol
        if(kard(i:i).eq.icmt.or.kard(i:i).eq.'!') go to 150
        if (kard(i:i).ne.iblank) then
          nsize = nsize+1
          if(kard(i+1:i+1).eq.iblank) go to 130
        end if
      end do
      go to 150
130   i3 = images(m)
      if(nsize.gt.i3) then
        write (*,132) nsize,i3,(kard(mm:mm),mm=1,ncol)
        write (iferr,132) nsize,i3,(kard(mm:mm),mm=1,ncol)
        stop 'reading error in frfmt.f'
      end if
      i1 = i+1
      l = i1-i2
      if (l.lt.i3) go to 140
      i2 = i1-i3
      do k=i2,i
        ll = ll+1
        kard1(ll:ll) = kard(k:k)
      end do
      go to 120

140   ll = ll+i3-1
      do k=i2,i
        ll = ll+1
        kard1(ll:ll) = kard(k:k)
      end do
120   i2 = i1
150   continue

      write (image,200) (kard1(i:i),i=1,nfld)
200   format (140a1)

132     format(1x,'field used for the variable > internally specified',/
     *  1x,'field used = ',i3,'  specified =',i3,'  on the following-',
     *  '  reduce field'/(1x,121a1))
c1790  format(//1x,5hcnwra,36x,37(1h=),44x,4hpage,i4/42x,1h=,35x,1h=,/
c     +37h=  i n p u t  d a t a  i m a g e s  =,/42x,1h=,35x,1h=,/42x
c     +,37(1h=)//5x,4hc no,3h  1,i9,7i10,/5x,4(1h=),2h +,16(4h====,1h+),
c     +10(1h-),24herrors/warnings detected,9(1h-))
c
      return
      end

c*file  convert

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  PURPOSE:

c     This routine is really cute! It converts all small case letters
c     to capitals in a character string 'key' of length nchar. It
c     is currently set to a max of 10 which can be changed to larger
c     value if desired.

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  INTERFACING ARGUMENTS:
```

```
c  Variable name       Type         Description
c  =============       =====        ============

c   key        character*10          keyword name
c   nchar      scalar, integer*4     no of characters for conversion

c-----------------------------------------------------------------------
c   Externals
c   =========

c    NONE

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   INTERFACING ROUTINES

c     Calling routines
c     ================

c      inpmetra.f
c      recdat.f

c   Called routines       Function
c   ================    ==============

c   None

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   INCLUDE FILES

c      Name                           Description
c      ====                           ============

c     None

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   SYSTEM LIBRARY ROUTINES

c     Name                           Description
c     ====                           ===========

c     None

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   OUTPUT UNIT(s)

c   Unit Name(Number)       file name            Description

c     None

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   REFERENCES

c   None

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine convert (key,nchar)

      save ich1,ich2
      character*26 ich1,ich2
      character*10 key

      data ich1,ich2/'abcdefghijklmnopqrstuvwxyz',
```

```
     *                    'ABCDEFGHIJKLMNOPQRSTUVWXYZ'/

c=====================================================================

      do 20 i = 1,nchar
        do 10 m = 1,26
          if(key(i:i).ne.ich1(m:m)) go to 10
            key(i:i)=ich2(m:m)
            go to 20
 10       continue
 20     continue
c       print *,'key    = ',(key(i:i),i=1,nchar)
       return
       end

c*file cputim.f

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  PURPOSE:

c     This subroutine captures computing time for different
c     components of the total code. Subroutine named 'seconds'
c     (appended below this routine) MUST BE MODIFIED for a
c     particular computer and the operating system.

c     Usage: to measure cpu time of subroutine/process
c            call cputim (0)
c            call subroutine/process
c            call cputim (n)
c     where n is a unique integer assigned to subroutine/process.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  INTERFACING ARGUMENTS:

c  Variable name       Type                    Description
c  =============       =====                   ============

c   ipgm       scalar, integer*4   gets the cpu for designated
c                                  subroutines as defined below

c---------------------------------------------------------------------
c    ipgm       subroutine/process
c---------------------------------------------------------------------
c     10 . . . . total metra.f
c      2 . . . . pvtx.f
c      3 . . . . pckr.f
c      4 . . . . compute coefficients
c      5 . . . . solver
c      6 . . . . updtkx.f
c      7 . . . . outmetra.f
c     20 . . . . total run time

c---------------------------------------------------------------------
c   Externals
c   =========

c    NONE

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   INTERFACING ROUTINES

c     Calling routines
c     ================

c      metra.f
```

```
c  (mainmet.f for standalone metra)
c      iter.f
c      cond.f

c   Called routines       Function
c   ================   ===============

c   seconds.f    get cpu-time in seconds

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   INCLUDE FILES

c      Name                              Description
c      ====                              ============

c   'impl.h'       Declares real variables to real*8 and sets
c                  frequently used constants in common.
c   'metragem.h'   variables  common to both metra and gem codes.
c   'paramtrs.h'   sets dimension limits for all variables.
c   'scalars.h'    scalars in common
c   'units.h'      input/output unit variables

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   SYSTEM LIBRARY ROUTINES

c      Name                             Description
c      ====                             ===========

c      None

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   OUTPUT UNIT(s)

c   Unit Name(Number)       file name       Description

c      ifout (52)           lfn_out         normal output
c        *                     -            output on screen

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c   REFERENCES

c   None

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine cputim (ipgm)

      include 'impl.h'
      include 'paramtrs.h'
      include 'units.h'
      include 'metragem.h'
      include 'scalars.h'

      common/cpus/ cpusub(20),tim1,timmetra,timgem

c=====================================================================

      if(ipgm.le.0) then                ! set the initial cpu-time
        call seconds(tim1)
        if(ipgm.eq.-1) timmetra = tim1
        if(ipgm.eq.-2) timgem = tim1
        return
c
      else
```

```
c
        call seconds(tim2)
        if(tim2.lt.tim1) tim2 = tim2+86400.
        sec = tim2-tim1
        if(ipgm.ne.20) then
          if(ipgm.ne.10) then
            cpusub(ipgm) = cpusub(ipgm)+sec  ! get cpu time for a processor
          else
            cpusub(ipgm) = cpusub(ipgm)+tim2-timmetra ! total metra time
          endif
        else
          totsec =(tim2-timgem)
          cpusub(20) = totsec
           deltmin = deltmin*utcnvf
           deltmax = deltmax*utcnvf

          write (ifout,9)
          write (ifout,10) (cpusub(i),i=1,7),(cpusub(i),i=10,13),
     *                 cpusub(20),itime,nnewton,inritrt,ncuts,deltmin,
     *                 tunits,deltmax,tunits
          write (*,9)
          write (*,10) (cpusub(i),i=1,7),(cpusub(i),i=10,13),
     *                 cpusub(20),itime,nnewton,inritrt,ncuts,deltmin,
     *                 tunits,deltmax,tunits
        endif

 9      format(/'======================================='/
     *          '      Routines          cpu-seconds' /
     *          '======================================')
10      format( ' read/initialization  = ',f10.2,/
     *          '        pvt           = ',f10.2,/
     *          '        pckr          = ',f10.2,/
     *          '     accm-coefs       = ',f10.2,/
     *          '       solve          = ',f10.2,/
     *          '   updt(psk)(vpk)     = ',f10.2,/
     *          '       output         = ',f10.2,/
     *          '  Total Metra Exec.   = ',f10.2,/
     *          '  Gem  read/initial   = ',f10.2,/
     *          '    Gem masstran      = ',f10.2,/
     *          '     Gem-Update       = ',f10.2,/
     *          '  Total Metra + Gem   = ',f10.2,//
     *          'Total Number of Steps =',i10,/
     *          'Total Newtonian Iters =',i10,/
     *          'Cumulative Inner Itrs =',i10,/
     *          'Total Time-Step Cuts  =',i10,/
     *          'Min Time Step Size    =',e10.3,1x,a2,/
     *          'Max Time Step Size    =',e10.3,1x,a2,/
     *          '======================================='/)
       endif

       return
       end

c*file seconds

c ============================================================

c PURPOSE

c     This subroutine returns cpu-seconds in variable 'sec'.
c     This routine  MUST BE MODIFIED  for a particular
c     operating system and the computer. If a suitable
c     system routine is not available, simply set 'sec'
c     = 0.d0  and 'return'.

c     Note that 'sec' is a double precision variable.

c============================================================
```

```
      subroutine seconds (sec)

      character*8 tim1,image
      real*8 sec
      dimension tyme(2)

c================================================
csun                       for sun  work station
csun        insert here the call for sun timing routine

        sec = etime(tyme)
c       sec = dtime(tyme)

c================================================
c                          for pc/lahey
cpc        call timer (isec)
cpc        sec = .01d0*isec

c================================================
c                          for pc/interactive-unix
c     call time (tim1)

c     write(image,10) tim1
c     read (image,20) ihr1,min1,isec1

c     sec = ihr1*3600+min1*60+isec1

c10   format(a8)
c20   format(3(i2,1x))

      return
      end

c*file lnblnk

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c PURPOSE

c  get the number of characters in a string removing the blanks at the
c  end of the string.

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      function lnblnk(b)
      character*(*) b

      lnblnk=len(b)
      do j=len(b),1,-1
        if(b(j:j).eq.' ') lnblnk=lnblnk-1
      end do

      return
      end

c*datablock

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  PURPOSE

c  This block data sets miscellaneous constants.

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      block data metragem
```

```
      include 'impl.h'
      include 'frfmt.h'

      data izro,ione,itwo,ithr,ifour,ifive,isix,isvn,ieight,inine,iten,
     +     ififtn,itwnty/0,1,2,3,4,5,6,7,8,9,10,15,20/

      data zero,quarter,half,one,two,three,four,five,six,seven/
     *     0.d0,0.25d0,0.5d0,1.d0,2.d0,3.d0,4.d0,5.d0,6.d0,7.d0/
      data eight, fnine, ten,    onesixth,          twthrds/
     *      8.d0, 9.d0, 10.d0, .1666666666667d0,.6666666666667d0/

      data ncol,nvalue,echo,   echono,iblank, iplus,iminus/
     +     140,   10,  'LIST', 'NOLI',  ' ',   '+',  '-'/
      data iqual,islash,istar,icmt, master,  skip,  noskip/
     +     '=',   '/',  '*',   ':', 'MAST', 'SKIP', 'NOSK'/

      end
CEND
```

postscript

JOB 1749

/sparc20/lichtner/masstrans/multiflo/listing/dcm.dvi

For: **lichtner**
Date: **Tue Mar 3 18:09:44 CST 1998**
Creator: **dvipsk 5.58f Copyright 1986, 1994 Radical Eye Software**

Submit queue: IF 1 / Ethernet / UHSW
Submitted: Mon Mar 02 19:43:39 1998
Started: Mon Mar 02 19:43:39 1998

QMS 3825 Print System

```
c    qf
c    qb
c    affin
c    rrkin
c    cc
c    phik
c    surf
c    gam
c    rkin

c-----------------------------------------------------------------------
c  Externals
c  =========

c  none

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  INTERFACING ROUTINES

c    Calling routines
c    ================

c    hybrid.f

c    Called routines        Function
c    ===============        =========

c    modbnd.f
c    ionexc.f
c    kinrxn.f
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  INCLUDE FILES

c      Name                            Description
c      ====                            ============

c      impl.h        -Declares real variables to real*8 and sets
c                       frequently used constants in common.
c      metragem.h    -Variables which are common to both metra and gem
c                       codes.
c      paramtrs.h    -Sets dimension limits for all variables.
c      scalgem.h     -Scalars in common.
c      kinetic.h     -Common block for kinetic rate parameters.
c      comgem.h      -General common block.
c      iounits.h     -Input/output units.

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  SYSTEM LIBRARY ROUTINES

c      Name                           Description
c      ====                           ===========

c      none

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  OUTPUT UNIT(s)

c  Unit Name(Number)              Description          file name

c    iunit2 (8)                   normal run output    masout

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  REFERENCES
```

```
c  none

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine kinrxns (nc,ncx,nk,np,r,qk,qf,qb,affin,rrkin,rkin,
     .           cc,cx,phik,surf,gam,gamx,cdl,eqkinr,iflgerr,index,aa)

      include 'impl.h'
      include 'addgem.h'
      include 'paramtrs.h'
      include 'metragem.h'
      include 'kinetic.h'
      include 'iounits.h'
      include 'scalgem.h'
      include 'comgem.h'

      dimension r(nc,*),qk(nk,*),qf(nk,*),qb(nk,*),affin(nk,*),
     .          rrkin(np,*),rkin(np,*),phik(nk,*),surf(nk,*),
     .          cc(nc,*),cx(ncx,*),gam(nc,*),gamx(ncx,*),cdl(nc,nc,*),
     .          eqkinr(nk,*),skpri(ncmx),sksec(ncxmx),aa(*)

      dimension ajnlog(ncmx)

c-----index = 0 - normal call (all reaction rates)
c             1 - electrochemical reaction rates only

      iflgerr = 0
      rgast0 = one/(rgaskj*tk0)

      do n = 1, nmax

        sum = zero

        do j=1,ncomp
          u4 = gam(j,n)*cc(j,n)
          if(u4.gt.zero) then
            ajnlog(j)= log(u4)
          else
            ajnlog(j) = zero
          endif
        enddo

        if (isothrm.eq.1) then
          tk = temp(n)
          uu1 = tk/tk0
          uu2 = one/(tk*rgaskj)-rgast0
          do nr = 1, nkin
            u3 = uu1*exp(-delh(nr)*uu2)
            do l = npar1(nr), npar2(nr)
              rkf(l) = rkf0(l)*u3
            end do
            eqkin(nr) = eqkinr(nr,n)
          end do
        endif

        do nr = 1, nkin

          if (index.eq.1 .and. ze(nr).eq.zero) goto 100

c---------compute forward and backward ion activity product
          qrf=zero
          qrb=zero
          do j=1,ncomp
            ajn=ajnlog(j)*skin(j,nr)
            if(skin(j,nr) .lt. zero) then
              qrb=qrb-ajn
            else
```

```
            qrf=qrf+ajn
          endif
        enddo

        dlnqk = qrb-qrf-eqkin(nr)*aln10

c       if (ze(nr).ne.0 .and. n.le.3) then
c         zpot0 = rgaskj*temp(n)*eqkin(nr)/(ze(nr)*faraday)*1.e3*
c    .    log(ten)
c         write(*,*) 'kinrxns: ',nr,eqkin(nr),zpot0,ze(nr),dlnqk
c       endif

        if (-dlnqk .gt. qkmax) then
          write(*,'(''WARNING!--affinity too large in kinrxn: '',
     .    a12,2i4,1p4e10.3)') namk(nr),n,j,dlnqk,qrb,qrf,qkmax
          write(*,'(3(2x,a8,1pe12.4))') (nam(j),cc(j,n),j=1,ncomp)
          iflgerr = 1
          return
        endif
        affin(nr,n) = rgaskj*tk*dlnqk
        qk(nr,n) = exp(-dlnqk)
        qrf      = exp(qrf)
        qrb      = exp(qrb)
        qf(nr,n) = qrf
        qb(nr,n) = qrb

c       rate sign convention chosen so that reaction rate is positive
c       for precipitation and negative for dissolution with mineral
c       species on left hand side of reaction.

        totrate = zero
        do lp = npar1(nr), npar2(nr)
          rrkin(lp,n) = zero
        enddo

        if ((qk(nr,n).gt.one.and.phik(nr,n).le.zero) .or.
     .    phik(nr,n).gt.zero) then

          if ((qk(nr,n).gt.one .and. qk(nr,n).ge.fkin(nr))
     .      .or. qk(nr,n).le.one) then

            do lp = npar1(nr), npar2(nr)

              ityprxn = itypkin(lp)
              rk  = rkf(lp)*surf(nr,n)
              sig = one/sigma(lp)

              npri = nkinpri(lp)
              if (npri .gt. 0) then
                do j = 1, ncomp
                  skpri(j) = skinpri(j,lp)
                enddo
              endif

              nsec = nkinsec(lp)
              if (nsec .gt. 0) then
                do i = 1, ncmplx
                  sksec(i) = skinsec(i,lp)
                enddo
              endif

              call rate (nr,lp,n,nc,ncx,nk,np,qk,dlnqk,rrkn,cc,cx,
     .          gam,gamx,ityprxn,rk,sig,skpri,sksec,npri,nsec,
     .          aa(iprod),aa(ipot))

              rrkin(lp,n) = rrkn
              totrate = totrate + wlam*rrkin(lp,n)+wlam1*rkin(lp,n)
            enddo
```

```
              totrate = voln(n)*delt*totrate
              do j=1,ncomp
                r(j,n) = r(j,n) + skin(j,nr)*totrate
              enddo
            endif
          endif

c---------sum contribution of reaction rates producing water
          if (jh2o .gt. 0) then
            sum = sum + skin(jh2o,nr)*totrate
          endif

c---------compute hydration/dehydration rate for use in metra
          rtot(n) = -sum*voln(n)

c**********************************************************************
c                add heterogeneous equilibria    (coefrxn.f-routine)
c                          kinetics
c**********************************************************************

          do lp = npar1(nr), npar2(nr)

            if (rrkin(lp,n) .ne. zero) then

              ityprxn = itypkin(lp)
              rk  = rkf(lp)*surf(nr,n)
              sig = one/sigma(lp)

              npri = nkinpri(lp)
              if (npri .gt. 0) then
                do j = 1, ncomp
                  skpri(j) = skinpri(j,lp)
                enddo
              endif

              nsec = nkinsec(lp)
              if (nsec .gt. 0) then
                do i = 1, ncmplx
                  sksec(i) = skinsec(i,lp)
                enddo
              endif

              call ratejac (nr,lp,n,nc,ncx,nk,np,qk,dlnqk,cdl,cc,cx,
     .        gam,gamx,ityprxn,rk,sig,skpri,sksec,npri,nsec,
     .        aa(iprod),aa(ipot))
            endif
          enddo
  100     continue
        enddo
      enddo

      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine rate (nr,lp,n,nc,ncx,nk,np,qk,dlnqk,rrkin,cc,cx,
     .           gam,gamx,ityprxn,rk,sig,skpri,sksec,npri,nsec,prod,
     .           pot)

      include 'impl.h'
      include 'paramtrs.h'
      include 'metragem.h'
      include 'kinetic.h'
      include 'iounits.h'
      include 'scalgem.h'
      include 'comgem.h'

      dimension qk(nk,*),cc(nc,*),cx(ncx,*),gam(nc,*),
```

```
     .          gamx(ncx,*),skpri(*),sksec(*),prod(np,*),pot(*)

c---------------------------------------
c-----reaction rate types
c         0        TST, pH-independent, rk (1-KQ)
c         1        TST, pH-dependent,   (rk0 + rk aH) (1-KQ)
c         2        TST, pH-dependent,
c         3        TST, pH-dependent,
c         5
c        10
c        11
c        12
c        20     TST general, rk prod_j a_j^vj prod_i a_i^vi (1-KQ^sigma)
c---------------------------------------

c-----compute prefactor
      if (npri.gt.0 .or. nsec.gt.0) then
        prefac = zero
        if (npri .gt. 0) then
          do j = 1,ncomp
            if(skpri(j) .ne. zero) then
              ccjn = cc(j,n)*gam(j,n)
              prefac = prefac+log(ccjn)*skpri(j)
            endif
          enddo
        endif
        if (nsec .gt. 0) then
          do i = 1,ncmplx
            if(sksec(i) .ne. zero) then
              ccxin = cx(i,n)*gamx(i,n)
              prefac = prefac+log(ccxin)*sksec(i)
            endif
          enddo
        endif
        prefac = exp(prefac)
      else
        prefac = one
      endif

c-----store prefactor for use in jacobian calculation
      prod(lp,n) = prefac

      if (ityprxn.eq.20) then

c-------transition state rate law
        rrkin = -rk*prefac*(one-qk(nr,n)**sig)

      else if (ityprxn.eq.21) then

c-------general butler-volmer rate law
c       note that dlnqk = -ln[Q_m K_m]

        fac   = (ze(nr)*faraday)/(rgaskj*temp(n))
        fpotn = fac*pot(n)
        rrkin = rk*prefac*(exp(-acorr(nr)*(fpotn+dlnqk))
     .          -exp(bcorr(nr)*(fpotn+dlnqk)))

      else if (ityprxn.eq.22) then

c-------iron oxidation - tafel rate law

        fac   = (rgaskj*temp(n))/(ze(nr)*faraday)
        potn  = pot(n)

c       expf = (potn+fac*dlnqk)/tafel(nr)
c       write(*,*) 'rate: ',n,nr,potn,fac,dlnqk,prefac,tafel(nr),expf

        rrkin = -rk*prefac*exp((potn+fac*dlnqk)/tafel(nr))
```

```
      else if (ityprxn.eq.30) then

c-------transport-limited Butler-Volmer rate law

c       note that dlnqk = -ln[Q_m K_m]

        fac  = (ze(nr)*faraday)/(rgaskj*temp(n))*1.d-3
        ffac = fac*pot(n)+dlnqk
        if (ffac .gt. 5.d0) then
          rrkin = -rk*prefac/gfac
        else if (ffac .lt. -5.d0) then
          rrkin =  rk*prefac/gfac
        else
          argp  =  bcorr(nr)*ffac
          argm  = -acorr(nr)*ffac
          expp  =  exp(argp)
          expm  =  exp(argm)
          rrkin =  rk*prefac*(expm-expp)/(one+gfac*(expm+expp))
        endif

        idebug = 0
        if (idebug .eq. 1 .and. n.le.3) then
          ecorr = -dlnqk/fac
          ecorr0 = eqkin(nr)/fac*aln10
          write(*,*) 'rate: ',n,nr,ecorr0,ecorr,pot(n),ffac,rrkin
        endif
      endif

      return
      end

      subroutine ratejac (nr,lp,n,nc,ncx,nk,np,qk,dlnqk,cdl,cc,cx,
     .                    gam,gamx,ityprxn,rk,sig,skpri,sksec,npri,
     .                    nsec,prod,pot)

      include 'impl.h'
      include 'paramtrs.h'
      include 'metragem.h'
      include 'kinetic.h'
      include 'scalgem.h'
      include 'comgem.h'

      dimension  qk(nk,*),cdl(nc,nc,*),
     .           cc(*),cx(ncx,*),gam(nc,*),gamx(ncx,*),
     .           skpri(*),sksec(*),prod(np,*),pot(*)

c-----get prefactor
      prefac = prod(lp,n)

      dtvol = delt*wlam*voln(n)

      if (ityprxn .eq. 20) then

c-------transition state rate law

        fk = rk*prefac*dtvol
        do l=1,ncomp

          dprfacdl = zero
          if (npri .gt. 0) then
            dprfacdl = skpri(l)
          endif
          if (nsec .gt. 0) then
            do i = 1, ncmplx
              dprfacdl = dprfacdl + shom(l,i)*sksec(i)
            enddo
          endif
```

```
          if (loglin .eq. 0) then
            fl = skin(l,nr)
          else
            ccl = cc(l+(n-1)*nc)
            fl = skin(l,nr)/ccl
            dprfacdl = dprfacdl/ccl
          endif

          pwrqk = qk(nr,n)**sig
          do j=1,ncomp
            if (skin(j,nr).ne.zero) then
              cdl(j,l,n) = cdl(j,l,n)+skin(j,nr)*fk*
     .        (sig*fl*pwrqk+dprfacdl*(one-pwrqk))
            endif
          enddo
        enddo

      else if (ityprxn .eq. 21) then

c-------general butler-volmer rate law

        fk = rk*prefac*dtvol
        fac   = (ze(nr)*faraday)/(rgaskj*temp(n))
        fpotn = fac*pot(n)

        do l=1,ncomp

          if (loglin .eq. 0) then
            fl = skin(l,nr)
          else
            ccl = cc(l+(n-1)*nc)
            fl = skin(l,nr)/ccl
          endif

          do j=1,ncomp
            if (skin(j,nr).ne.zero) then
              cdl(j,l,n) = cdl(j,l,n)+skin(j,nr)*fk*fl*(
     .        acorr(nr)*exp(-acorr(nr)*(fpotn+dlnqk))+
     .        bcorr(nr)*exp(bcorr(nr)*(fpotn+dlnqk)))
            endif
          enddo
        enddo

      else if (ityprxn .eq. 22) then

c-------iron oxidation - tafel rate law

        fac = (rgaskj*temp(n))/(ze(nr)*faraday)
        fk = rk*prefac*dtvol*fac
        potn = pot(n)

        do l=1,ncomp

          if (loglin .eq. 0) then
            fl=skin(l,nr)
          else
            ccl = cc(l+(n-1)*nc)
            fl = skin(l,nr)/ccl
          endif

          do j=1,ncomp
            if (skin(j,nr).ne.zero) then
              cdl(j,l,n) = cdl(j,l,n)+skin(j,nr)*fk*fl*
     .        exp((potn+fac*dlnqk)/tafel(nr))/tafel(nr)
            endif
          enddo
        enddo
```

```
      else if (ityprxn.eq.30) then

c-------transport-limited Butler-Volmer rate law

        fac   =  (ze(nr)*faraday)/(rgaskj*temp(n))*1.d-3
        ffac  =  fac*pot(n)+dlnqk

        if (ffac.le.5.d0 .and. ffac.ge.-5.d0) then
          do l=1,ncomp

            if (loglin .eq. 0) then
              fl=skin(l,nr)
            else
              ccl = cc(l+(n-1)*nc)
              fl = skin(l,nr)/ccl
            endif

            do j=1,ncomp
              if (skin(j,nr).ne.zero) then
                argp  =  bcorr(nr)*ffac
                argm  = -acorr(nr)*ffac
                expp  =  exp(argp)
                expm  =  exp(argm)
                sum   =  expm+expp
                den   =  one+gfac*sum
                cdl(j,l,n) = cdl(j,l,n)+skin(j,nr)*fk*fl*(
     .          acorr(nr)*expm+bcorr(nr)*expp
     .          +gfac*sum*(-acorr(nr)*expm+bcorr(nr)*expp)/den)/den
              endif
            enddo
          enddo
        endif

      endif

      return
      end
```

61/70

```
c*file kinrxnaq.f

c Program Name:          MULTIFLO/GEM
c File/Subroutine Name: kinrxnaq.f
c Release Date:          April 11, 1997
c Release Version:       1.0
c Client Name:           USNRC
c Contract Number:       NRC 02-93-005
c CNWRA Contact:         Peter C. Lichtner (210-522-6084)
c                        Center for Nuclear Waste Regulatory Analyses
c                        San Antonio, Texas 78238-5166
c                        lichtner@swri.edu
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c VERSION/REVISION HISTORY

c $Id$
c $Log$

c-------------------------------------------------------------------------
c   Date            Author(s)            Comments/Modifications
c-------------------------------------------------------------------------

c  April  97        Peter C. Lichtner        Initial Implementation
c                   Mohan S. Seth
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c DISCLAIMER/NOTICE

c This computer code/material was prepared as an account of work
c performed by the Center for Nuclear Waste Regulatory Analyses (CNWRA)
c for the Division of Waste Management of the Nuclear Regulatory
c Commission (NRC), an independent agency of the United States
c Government. The developer(s) of the code nor any of their sponsors
c make any warranty, expressed or implied, or assume any legal
c liability or responsibility for the accuracy, completeness, or
c usefulness of any information, apparatus, product or process
c disclosed, or represent that its use would not infringe on
c privately-owned rights.

c IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW WILL THE SPONSORS
c OR THOSE WHO HAVE WRITTEN OR MODIFIED THIS CODE, BE LIABLE FOR
c DAMAGES, INCLUDING ANY LOST PROFITS, LOST MONIES, OR OTHER SPECIAL,
c INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR
c INABILITY TO USE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA
c BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY THIRD PARTIES OR A
c FAILURE OF THE PROGRAM TO OPERATE WITH OTHER PROGRAMS) THE PROGRAM,
c EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES,
c OR FOR ANY CLAIM BY ANY OTHER PARTY.

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  PURPOSE:

c  This routine computes the kinetic reaction rates of minerals, and
c  computes the diagonal jacobian coefficients for kinetic rate terms.

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  INTERFACING ARGUMENTS:

c  Variable name       Type                  Description
c  =============       =====                 ===========

c    nc          Integer               -number of primary species
c    nk
c    r
c    qk
```

62/70

```
c    qf
c    qb
c    affin
c    rrkin
c    cc
c    phik
c    gam
c    rkin

c-----------------------------------------------------------------------
c  Externals
c  =========

c  none

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  INTERFACING ROUTINES

c    Calling routines
c    ================

c    hybrid.f

c    Called routines        Function
c    ===============        =========

c    modbnd.f
c    ionexc.f
c    kinrxn.f
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  INCLUDE FILES

c      Name                          Description
c      ====                          ============

c      impl.h        -Declares real variables to real*8 and sets
c                       frequently used constants in common.
c      metragem.h    -Variables which are common to both metra and gem
c                       codes.
c      paramtrs.h    -Sets dimension limits for all variables.
c      scalgem.h     -Scalars in common.
c      kinetic.h     -Common block for kinetic rate parameters.
c      comgem.h      -General common block.
c      iounits.h     -Input/output units.

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  SYSTEM LIBRARY ROUTINES

c      Name                          Description
c      ====                          ===========

c      none

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  OUTPUT UNIT(s)

c  Unit Name(Number)              Description          file name

c    iunit2 (8)                   normal run output    masout

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c  REFERENCES
```

```
c  none

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine kinrxnaq (nc,ncx,nk,np,r,rrkinaq,rkinaq,cc,cx,
     .                     gam,gamx,cdl,iflgerr,index,aa)

      include 'impl.h'
      include 'addgem.h'
      include 'paramtrs.h'
      include 'metragem.h'
      include 'kinetic.h'
      include 'cxkin.h'
      include 'iounits.h'
      include 'scalgem.h'
      include 'comgem.h'

      dimension r(nc,*),qk(nxkmx),rrkinaq(np,*),rkinaq(np,*),
     .          cc(nc,*),cx(ncx,*),gam(nc,*),gamx(ncx,*),cdl(nc,nc,*),
     .          skpri0(ncmx),sksec0(ncxmx),aa(*)

      dimension ajnlog(ncmx)

c-----index = 0 - normal call (all reaction rates)
c             1 - electrochemical reaction rates

      iflgerr = 0
      rgast0 = one/(rgaskj*tk0)

      do n = 1, nmax

        sum = zero

        do j=1,ncomp
          u4 = gam(j,n)*cc(j,n)
          if(u4.gt.zero) then
            ajnlog(j)= log(u4)
          else
            ajnlog(j) = zero
          endif
        enddo

        if (isothrm.eq.1) then
          tk = temp(n)
          uu1 = tk/tk0
          uu2 = one/(tk*rgaskj)-rgast0
          do nr = 1, ncxkin
            u3 = uu1*exp(-delhcxk(nr)*uu2)
            do l = nparcxk1(nr), nparcxk2(nr)
              rcxkf(l) = rcxkf0(l)*u3
            enddo
          enddo
        endif

        do nr = 1, ncxkin

          if (index.eq.1 .and. zeaq(nr).eq.zero) goto 100

c---------compute forward and backward ion activity product

          qrf=zero
          qrb=zero
          do j=1,ncomp
            ajn=ajnlog(j)*skpri(j,nr)
            if(skpri(j,nr) .lt. zero) then
              qrb=qrb-ajn
            else
              qrf=qrf+ajn
```

```
          endif
        enddo

        dlnqk = qrb-qrf-eqkin(nr)*aln10
        if (-dlnqk .gt. qkmax) then
           write(*,'(''WARNING!--affinity too large in kinrxn: '',
     .     a12,2i4,1p4e10.3)')
     .     namk(nr),n,j,dlnqk,qrb,qrf,qkmax
           write(*,'(3(2x,a8,1pe12.4))')
     .     (nam(j),cc(j,n),j=1,ncomp)
           iflgerr = 1
           return
        endif
        qk(nr)   = exp(-dlnqk)
        qrf      = exp(qrf)
        qrb      = exp(qrb)

c       rate sign convention chosen so that reaction rate is positive
c       for precipitation and negative for dissolution with mineral
c       species on left hand side of reaction.

        totrate = zero

        do lp = nparcxk1(nr), nparcxk2(nr)

          rrkinaq(l,n) = zero

          ityprxn = itypcxk(lp)
          rk = rcxkf(lp)

          npri = nkpriaq(lp)
          if (npri .gt. 0) then
            do j = 1, ncomp
              skpri0(j) = skpriaq(j,lp)
            enddo
          endif

          nsec = nkinsec(lp)
          if (nsec .gt. 0) then
            do i = 1, ncmplx
              sksec0(i) = sksecaq(i,lp)
            enddo
          endif

          call rateaq (nr,lp,n,nc,ncx,nk,np,qk,dlnqk,rrkn,cc,cx,
     .     gam,gamx,ityprxn,rk,skpri0,sksec0,npri,nsec,
     .     aa(iprod),aa(ipot))

          rrkinaq(lp,n) = rrkn
          totrate = totrate + wlam*rrkinaq(lp,n)+wlam1*rkinaq(lp,n)

        enddo

        totrate = voln(n)*delt*totrate
        do j=1,ncomp-ncxkin
          r(j,n) = r(j,n) + skpri(j,nr)*totrate
        enddo
        do j=ncomp-ncxkin+1,ncomp
          r(j,n) = r(j,n) - totrate
        enddo

c---------sum contribution of reaction rates producing water
c       if (jh2o .gt. 0) then
c         sum = sum + skpri(jh2o,nr)*totrate
c       endif

c---------compute hydration/dehydration rate for use in metra
c       rtot(n) = -sum*voln(n)
```

```
c**********************************************************************
c                add homogeneous equilibria    (coefrxn.f-routine)
c                         kinetics
c**********************************************************************

          do lp = nparcxk1(nr), nparcxk2(nr)

            if (rrkinaq(lp,n) .ne. zero) then

              ityprxn = itypcxk(lp)
              rk  = rkf(lp)

              npri = nkpriaq(lp)
              if (npri .gt. 0) then
                do j = 1, ncomp
                  skpri0(j) = skpriaq(j,lp)
                enddo
              endif

              nsec = nksecaq(l)
              if (nsec .gt. 0) then
                do i = 1, ncmplx
                  sksec0(i) = sksecaq(i,lp)
                enddo
              endif

              call ratejaq (nr,lp,n,nc,ncx,nk,np,qk,dlnqk,cdl,cc,cx,
     .        gam,gamx,ityprxn,rk,skpri0,sksec0,npri,nsec,
     .        aa(iprod),aa(ipot))
            endif
          enddo
  100     continue
        enddo
      enddo

      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine rateaq (nr,lp,n,nc,ncx,nk,np,qk,dlnqk,rrkin,cc,cx,
     .            gam,gamx,ityprxn,rk,skpri,sksec,npri,nsec,prod,pot)

      include 'impl.h'
      include 'paramtrs.h'
      include 'metragem.h'
      include 'kinetic.h'
      include 'iounits.h'
      include 'scalgem.h'
      include 'comgem.h'

      dimension qk(*),cc(nc,*),cx(ncx,*),gam(nc,*),gamx(ncx,*),
     .          skpri(*),sksec(*),prod(np,*),pot(*)

c---------------------------------------
c-----reaction rate types
c         0        TST, pH-independent, rk (1-KQ)
c         1        TST, pH-dependent,   (rk0 + rk aH) (1-KQ)
c         2        TST, pH-dependent,
c         3        TST, pH-dependent,
c         5
c        10
c        11
c        12
c        20        TST general, sum rk prod aj^bj prod ai^bi (1-KQ)
c        21        B-V, sum rk prod aj^bj prod ai^bi (1-KQ)
c        22        Tafel, sum rk prod aj^bj prod ai^bi (1-KQ)
c        23        Tafel, sum rk prod aj^bj prod ai^bi (1-KQ)
c---------------------------------------
```

```
      if (npri.gt.0 .or. nsec.gt.0) then
        prefac = zero
        if (npri .gt. 0) then
          do j = 1,ncomp
            if(skpri(j) .ne. zero) then
              ccjn = cc(j,n)*gam(j,n)
              prefac = prefac+log(ccjn)*skpri(j)
            endif
          enddo
        endif
        if (nsec .gt. 0) then
          do i = 1,ncmplx
            if(sksec(i) .ne. zero) then
              ccxin = cx(i,n)*gamx(i,n)
              prefac = prefac+log(ccxin)*sksec(i)
            endif
          enddo
        endif
        prefac = exp(prefac)
      else
        prefac = one
      endif

c-----store prefactor for use in jacobian calculation
      prod(lp,n) = prefac

      if (ityprxn.eq.20) then

        rrkin = -rk*prefac*(one-qk(nr))

      else if (ityprxn.eq.21) then

c-------general butler-volmer rate law

        fac = (zeaq(nr)*faraday)/rgas*temp(n)
        fpotn = fac*pot(n)
        rrkin = rk*prefac*(exp(-acorraq(nr)*(fpotn+dlnqk))
     .  -exp(bcorraq(nr)*(fpotn+dlnqk)))

      else if (ityprxn.eq.22) then

c-------oxygen reduction - tafel rate law - diffusion limited

c-------set temporarily
        diff02 = 1.e-9 ! m^2/s
        bndlayer = 5.d-4 ! m
        fac = rgas*temp(n)/(zeaq(nr)*faraday)
        potn = pot(n)
        fexp = exp(-(potn+fac*dlnqk)/tafelaq(nr))
        rfac  = rk*zeaq(nr)*bndlayer/(four*diff02*cc(jo2,n))

c*********check sign!
        rrkin = rk*prefac*fexp/(one+rfac*fexp)

      else if (ityprxn.eq.23) then

c-------water reduction - tafel rate law

        fac = rgas*temp(n)/(zeaq(nr)*faraday)
        potn = pot(n)

c*********check sign!
        rrkin = -rk*prefac*exp((potn+fac*dlnqk)/tafelaq(nr))

      else if (ityprxn.eq.30) then

c-------transport-limited Butler-Volmer rate law
```

```
c       note that dlnqk = -ln[Q_m K_m]

        fac  = (zeaq(nr)*faraday)/(rgaskj*temp(n))*1.d-3
        ffac = fac*pot(n)+dlnqk
        if (ffac .gt. 5.d0) then
          rrkin = -rk*prefac/gfac
        else if (ffac .lt. -5.d0) then
          rrkin =  rk*prefac/gfac
        else
          argp  =  bcorraq(nr)*ffac
          argm  = -acorraq(nr)*ffac
          expp  =  exp(argp)
          expm  =  exp(argm)
          rrkin =  rk*prefac*(expm-expp)/(one+gfac*(expm+expp))
        endif

        idebug = 0
        if (idebug .eq. 1 .and. n.le.3) then
          ecorr = -dlnqk/fac
          ecorr0 = eqkin(nr)/fac*aln10
          write(*,*) 'rate: ',n,nr,ecorr0,ecorr,pot(n),ffac,rrkin
        endif

      endif

      return
      end

      subroutine ratejaq (nr,lp,n,nc,ncx,nk,npaq,qk,dlnqk,cdl,cc,cx,
     .            gam,gamx,ityprxn,rk,skpri0,sksec0,npri,nsec,prod,pot)

      include 'impl.h'
      include 'paramtrs.h'
      include 'metragem.h'
      include 'kinetic.h'
      include 'cxkin.h'
      include 'scalgem.h'
      include 'comgem.h'

      dimension qk(*),cdl(nc,nc,*),cc(*),cx(ncx,*),prod(npaq,*),
     .          gam(nc,*),gamx(ncx,*),skpri0(*),sksec0(*),pot(*)

c-----get prefactor
      prefac = prod(lp,n)

      dtvol = delt*wlam*voln(n)

      if (ityprxn .eq. 20) then

c-------TST rate law

        fqk = rk*prefac*dtvol
        do l=1,ncomp

          dprfacdl = zero
          if (npri .gt. 0) then
            dprfacdl = skpri0(l)
          endif
          if (nsec .gt. 0) then
            do i = 1, ncmplx
              dprfacdl = dprfacdl + shom(l,i)*sksec0(i)
            enddo
          endif

          if (loglin .eq. 0) then
            fl=skpri(l,nr)
          else
```

```
            ccl = cc(1+(n-1)*nc)
            fl = skpri(l,nr)/ccl
            dprfacdl = dprfacdl/ccl
          endif

          pwrqk = qk(nr)
          do j=1,ncomp-ncxkin
            if (skpri(j,nr).ne.zero) then
              cdl(j,l,n) = cdl(j,l,n)+skpri(j,nr)*fqk*
     .        dprfacdl*(one-pwrqk)
            endif
          enddo
          do j=1,ncomp-ncxkin+1,ncomp
            if (skpri(j,nr).ne.zero) then
              cdl(j,l,n) = cdl(j,l,n)-fqk*dprfacdl*(one-pwrqk)
            endif
          enddo
        enddo

      else if (ityprxn .eq. 21) then

c-------general butler-volmer rate law

        fk = rk*prefac*dtvol
        fac   = (zeaq(nr)*faraday)/(rgas*temp(n))
        fpotn = fac*pot(n)

        do l=1,ncomp

          if (loglin .eq. 0) then
            fl = skpri(l,nr)
          else
            ccl = cc(1+(n-1)*nc)
            fl = skpri(l,nr)/ccl
          endif

          do j=1,ncomp
            if (skpri(j,nr).ne.zero) then
              cdl(j,l,n) = cdl(j,l,n)+skpri(j,nr)*fk*fl*(
     .        acorraq(nr)*exp(-acorraq(nr)*(fpotn+dlnqk))+
     .        bcorraq(nr)*exp(bcorraq(nr)*(fpotn+dlnqk)))
            endif
          enddo
        enddo

      else if (ityprxn .eq. 22) then

c-------oxygen reduction - tafel rate law - diffusion limited

c-------set temporarily
        diff02 = 1.e-9 ! m^2/s
        bndlayer = 5.d-4 ! m

        c02aq = cc(jo2+(n-1)*nmax)
        fac  = rgas*temp(n)/(zeaq(nr)*faraday)
        potn = pot(n)
        fexp = exp(-(potn+fac*dlnqk)/tafelaq(nr))
        rfac = rk*zeaq(nr)*bndlayer/(four*diff02*c02aq)
        fk   = rk*prefac*dtvol/(one+rfac*fexp)**2

        do l=1,ncomp

          if (loglin .eq. 0) then
            fl=skpri(l,nr)
          else
            ccl = cc(1+(n-1)*nc)
            fl = skpri(l,nr)/ccl
          endif
```

```
          if (l .ne. jo2) then
            fll = fac*fexp/tafelaq(nr)*fl
          else
            fll = fexp*(fac/tafelaq(nr)*fl-fexp*rfac/c02aq)
          endif

          do j=1,ncomp
            if (skpri(j,nr).ne.zero) then
              cdl(j,l,n) = cdl(j,l,n)+skpri(j,nr)*fk*fll
            endif
          enddo
        enddo

      else if (ityprxn .eq. 23) then

c-------water reduction - tafel rate law

        fac = (rgas*temp(n))/(zeaq(nr)*faraday)
        fk = rk*prefac*dtvol*fac
        potn = pot(n)

        do l=1,ncomp

          if (loglin .eq. 0) then
            fl=skpri(l,nr)
          else
            ccl = cc(l+(n-1)*nc)
            fl = skpri(l,nr)/ccl
          endif

          do j=1,ncomp
            if (skpri(j,nr).ne.zero) then
              cdl(j,l,n) = cdl(j,l,n)+skpri(j,nr)*fk*fl*
     .        exp((potn+fac*dlnqk)/tafelaq(nr))/tafelaq(nr)
            endif
          enddo
        enddo

      else if (ityprxn.eq.30) then

c-------transport-limited Butler-Volmer rate law

        fac   =  (zeaq(nr)*faraday)/(rgaskj*temp(n))*1.d-3
        ffac  =  fac*pot(n)+dlnqk

        if (ffac.le.5.d0 .and. ffac.ge.-5.d0) then
          do l=1,ncomp

            if (loglin .eq. 0) then
              fl=skpri(l,nr)
            else
              ccl = cc(l+(n-1)*nc)
              fl = skpri(l,nr)/ccl
            endif

            do j=1,ncomp
              if (skpri(j,nr).ne.zero) then
                argp  =  bcorraq(nr)*ffac
                argm  = -acorraq(nr)*ffac
                expp  =  exp(argp)
                expm  =  exp(argm)
                sum   =  expm+expp
                den   =  one+gfac*sum
                cdl(j,l,n) = cdl(j,l,n)+skpri(j,nr)*fk*fl*(
     .          acorraq(nr)*expm+bcorraq(nr)*expp
     .          +gfac*sum*(-acorraq(nr)*expm+bcorraq(nr)*expp)/den)/den
              endif
```

```
          enddo
        enddo
      endif

    endif

    return
    end
```