**Special Report 869**
March 1991

# User's Guide for the Mathematical Model LT3VSI Denitrification in Nonhomogeneous Laboratory Scale Aquifers


Agricultural Experiment Station
Oregon State University

Agricultural Experiment Station
Oregon State University
Special Report 869

# User's Guide for the Mathematical Model LT3VSI Denitrification in Nonhomogeneous Laboratory Scale Aquifers

G.A. Bachelor, D.E. Cawlfield, F.T. Lindstrom, and L. Boersma

This is the user's guide for a mathematical model for the steady state flow field and the transport and fate of chemicals in laboratory scale, nonhomogeneous aquifers. The model is intended to be used for evaluation of bioremediation of aquifers with high nitrate concentrations. It also can be used for the design of more general bioremediation processes. This model handles four chemicals and two microbial populations.

This guide is a companion document to the report that describes the mathematical model, LT3VSI (Lindstrom, et al, 1990).


**Authors**
G.A. Bachelor and D.E. Cawlfield, senior systems analysts,
F.T. Lindstrom, associate professor, and L. Boersma, professor,
Dept. of Crop and Soil Science, Oregon State University.

## FOREWORD

This report provides a user guide for a mathematical model describing transport and fate processes in laboratory scale model aquifers. Such model aquifers are constructed to evaluate strategies for the reclamation of groundwater contaminated with nitrates and other pollutants. The mathematical model can be used for evaluations of strategies for injecting chemicals, which could enhance pollutant degradation, into an aquifer. The model allows calculation of hydraulic pressure fields and perturbations of the field by injection and/or extraction wells. It uses the pressure field to compute the Darcy velocities, which in turn are used in computing the chemical transport and fate. This model is LT3VSI, which handles four chemicals and two microbial populations.

## NOTICE

Several changes have been made in the LT3VSI programs since this manual was written. See Section I.2 for a description of the revisions.

## ACKNOWLEDGMENT

This publication reports results of studies supported by Cooperative Agreement CR 814502-01-1 "Transport and Fate of Solutes in Unsaturated/Saturated Soils." This agreement is between the Robert S. Kerr Environmental Research Laboratory of the U.S. Environmental Protection Agency and Oregon State University.

CONTENTS

# DENITRIFICATION IN NONHOMOGENEOUS LABORATORY SCALE AQUIFERS: 5: USER'S MANUAL FOR THE MATHEMATICAL MODEL LT3VSI

## I. Introduction

This report is a companion document to the report (Volume 4) describing the mathematical model LT3VSI, (Lindstrom, et al, 1990). As the title implies, this volume is the user's guide for running the model on mainframe computers. Those readers, wishing to understand the basic science on which the model is built, must first read Volume 4, (Lindstrom, Boersma, Myrold, and Barlaz, 1990). Those readers wishing only to learn how to set up data files and run the computer programs need to read this document thoroughly. This document begins with a brief overview of the major sections of the computer code including the names of certain input and output files (Section I). The user is then walked through a basic installation procedure (Section II). A listing of the Fortran variables, with mathematical symbols, definitions, and units is given next (Section III). This is followed by a section on the preparation of input data files (Section IV). A section on the running of the programs and a discussion of the output files is next (Section V). This is followed by a section describing the programs (Section VI) and finally, the references (Section R). The Fortran 77 program listings end this volume (Appendices A and B).

### I.1 Program overview

The programs WATER, CHEMINIT, and CHEMLOOP have been coded in ANSI Standard Fortran 77. They implement the mathematical model described by Lindstrom, et al (1990). The programs use three data files, one of which is optional. One file specifies the

dimensions and parameters for the hydrology of the aquifer. The second file specifies the parameters for the chemistry. The third, optional, file specifies a schedule for changing concentrations of chemicals in injection wells.

The notation "xxxxxxxx" commonly used in this manual represents a "root" file name that is chosen by the user. The 3-character extensions such as ".WAD" must be exactly as shown here.

The water program WATER reads the hydrology file xxxxxxxx.WAD. It then computes the steady-state fluid flow in the aquifer, using Stone's Strongly Implicit System (SIP) method. The program produces four output files. The file xxxxxxxx.WAO contains the information read from file xxxxxxxx.WAD together with computed information. The file xxxxxxxx.WPV contains the hydraulic pressures and the flow velocities in the aquifer. The file xxxxxxxx.WDB contains "debugging" outputs selected by control information in the file xxxxxxxx.WAD. And finally, the water program writes an unformatted interface file xxxxxxxx.WIF containing the flow velocities and other information needed by the chemistry programs.

The chemistry initialization program CHEMINIT reads the chemistry data file xxxxxxxx.CHD and the interface file produced by the water program. It does some data processing in preparation for the loop program. It writes four output files. The file xxxxxxxx.CHO contains most of the information read from the file xxxxxxxx.CHD together with computed information. The file xxxxxxxx.CDB contains "debugging" outputs selected by control information in the file xxxxxxxx.CHD. The file LT3VSI.CLD contains data from the file xxxxxxxx.CHD that is not processed by CHEMINIT. And finally, the unformatted interface file LT3VSI.CIF contains the information needed by the loop program.

The chemistry loop program CHEMLOOP is run immediately after

CHEMINIT has finished. CHEMLOOP reads the interface file produced by CHEMINIT, the file LT3VSI.CLD, and a schedule file xxxxxxxx.SCH, if the user provided it. CHEMLOOP computes the transport and fate of four chemicals and the changing populations of two microbes in the aquifer for a specified period of time, using a modified Euler-LaGrange procedure (Lindstrom, et al, 1990). The program produces three output files. The file xxxxxxxx.CNC contains the four chemical concentration distributions in the aquifer at selected times, together with the total chemical mass. This file also contains the populations of the two microbes at the selected times. Data in the chemistry input file specify how many times to print this information during the run. Print times are at equally spaced intervals. CHEMLOOP also writes additional information on the files xxxxxxxx.CDB and xxxxxxxx.CHO.

By setting a certain "flag" in the chemistry file xxxxxxxx.CHD, you can cause CHEMLOOP to write an unformatted file xxxxxxxx.CUF containing the final chemical concentrations and microbe populations. This file can be read by CHEMINIT during a subsequent run of the chemistry programs. The final chemical concentrations of one run serve as the starting concentration distributions for the next run. One use of this feature is to continue a run, if it happens that the first run didn't go far enough to produce a desired result. A more interesting possibility is to change some of the chemical parameters and continue the run. In this case, one would use a different chemistry data file in the subsequent run or runs.

For example, a simulation run can be made using the $Q_{SO}$ slow leak source term. This can simulate a slowly leaking drum or canister of waste chemical. After a period of time, the leak is discovered, dug up, and removed from the aquifer. At this point in time $Q_{SO}$ is set to zero. The existing concentration plume constitutes the "initial data" for the remainder of the run.

It is also possible to change the hydraulic parameters, by using two or more different water data files and running the water program on each of them. A subsequent chemistry run can then use an interface file different from the one used in the earlier run.

## I.2 Update notice: LT3VSI Version 1.3

Some changes have been made in the LT3VSI programs since the other sections of this manual were written. The three programs WATER, CHEMINIT, and CHEMLOOP are now Version 1.3. The only change in the WATER program is the version number. The revisions in CHEMINIT and CHEMLOOP are described below.

Correction of an error. The principal reason for making a new version of LT3VSI was to correct an error that was discovered in the part of CHEMLOOP that computes the new chemical concentrations (CNEW) from the old concentrations. In Version 1.2, the MLOSS term, which represents the loss due to microbes, was placed in an expression with several other terms. The sum was multiplied by several factors, including the old chemical concentration (COLD). MLOSS should not be multiplied by COLD. In Version 1.3, this error has been corrected.

Improved time management. In the data file xxxxxxxx.CHD, the user is allowed to specify TMAX and DT0. DT0 is the desired time increment and TMAX is the length of time the program should run. The method used in CHEMLOOP is unstable if DT0 is too large. The CHEMINIT program computes bounds on the largest allowable DT0. There is one bound (DTMAX) for each of the four chemicals. DT0 must be less than the smallest of these four bounds. If the user-specified DT0 is larger than one-half of the minimum bound, the program sets DT0 to this value. In Version 1.2, the output

file xxxxxxxx.CHO showed the user-specified DT0, followed by the values of the four DTMAXes. The file did not show the actual DT0 to be used by the program. In Version 1.3, the file xxxxxxxx.CHO shows the actual DT0, and if the user's DT0 was reduced, also prints a message. If DT0 is reduced, Version 1.3 also reduces TMAX by the same ratio so that the number of iterations will be the same as would have been performed under the original values. This avoids the problem of having the program run much longer than anticipated.

For example, if part of the xxxxxxxx.CHD file is as follows:

```
NPRT  TMAX  DT0 (DAYS) (RUN CONTROL DATA)
2     4.0   0.001
```

and it happens that DT0 = 0.001 is too large, then part of the xxxxxxxx.CHO file may appear as follows:

```
       RUN CONTROL INFORMATION.

       TIMES IN DAYS:
  T0=  0.000E+00  TMAX=  6.186E-04  DT0=  1.547E-07
       DT0 AND TMAX HAVE BEEN REDUCED TO ENSURE STABILITY

  COMPUTED   NUTRIENT    SUBSTRATE   OXYGEN      NITRATE
    DTMAX=   4.722E-05   1.067E-04   3.093E-07   7.479E-07

  NPRT=     2;  PRINT TIMES (DAYS) ARE:

3.093E-04 6.186E-04
```

In most places where TIME is printed or displayed, Version 1.2 used fixed format output. Version 1.3 uses "G" format output, which automatically switches to exponential format if the value is too large or too small for fixed format.

Reduction of DT0 during run. CHEMINIT determines how large DT0 can be for stability under the initial conditions. However, as the microbial populations change during a run, the maximum allowable DT0 may change. During the computation of CNEW from COLD in CHEMLOOP, a certain part of the expression is computed first and tested. If this subexpression is negative, the program divides DT0 by 2.0 to make it smaller, and goes back to the

beginning of the loop. In Version 1.2, the program displayed a message and also printed a message on the output file xxxxxxxx.CDB. In Version 1.3, the program displays more information when DT0 is reduced, including the iteration count (IC), the current TIME, and the values of I, J, and K. I and J are the indices that specify a node point in the aquifer, and K selects one of the four chemicals. Version 1.3 also puts a limit of ten on the number of times that DT0 will be reduced. If this limit is exceeded, the program terminates with an error message.

Length of run. Version 1.2's CHEMLOOP program continued executing the loop until the elapsed time reached the TMAX specified by the user. In Version 1.3, the value of TMAX/DT0, using the user-specified values, determines the number of time steps that will be performed. This number is not affected by changes in TMAX or DT0 carried out by either CHEMINIT or CHEMLOOP. The reason for this change is to assure that when the program is run in "batch" mode, it will not run for a much longer CPU time than the user had planned.

Anisotropic dispersion. It was discovered that Version 1.2 could not handle anisotropic dispersion, which occurs if the X and Y components of dispersivity (DISPLX and DISPLY) are different. Anisotropic dispersion causes negative chemical concentrations, which causes CHEMLOOP to repeatedly reduce DT0. Two ad-hoc features were added to Version 1.3 to deal with this problem. The first feature is that the cross-component of dispersion (DCHLXY) is set to zero instead of the value specified by the formula. However, if NFLAG(18) is set to 1 in the xxxxxxxx.CHD file, the program will compute DCHLXY from the formula, producing the same results as Version 1.2. The second feature is that the reduction of DT0 can be disabled by setting NFLAG(19) to 1 in the xxxxxxxx.CHD file. This is risky and is

not recommended; if the computation in CHEMLOOP becomes unstable, the results may be wrong.

# II. Installing the programs

The programs, as distributed, can be run on an FPS Scientific computer attached to a VAX computer. They can be modified fairly easily to run on a VAX or on a PC (AT-class or better). In the following discussion, we are considering the VAX 780 computer running the VMS operating system, serving as the "host" computer for an FPS 264 computer. These computers are installed at the University Computing Services at Oregon State University (OSU). We assume the user has a PC or compatible computer running communication software that provides file transfer capability. The PC is connected to the VAX system by a communications link. We assume the user has accounts on the VAX and the FPS.

The programs were developed under DOS 3.10, using Microsoft Fortran Version 4.01, and initial testing was done under this system. They were then modified to run on the FPS computer. The distribution diskette contains an archive file which contains the Fortran source files for the FPS version. Other archive files contain the data files and result files for the small example that is used for illustration in this manual.

There are also files named READ.ME and LT3.HLP on the diskette that contain instructions for installing and running the programs. Those instructions are also included in this Guide.

## II.1. Installation

The user of LT3VSI should have a PC or compatible computer with a hard disk that has at least 650K bytes of free space. The PC should have communications software and a communications link to a VAX or other computer that is large enough and fast enough to run LT3VSI.

To begin the installation, create a subdirectory named LT3VSI on the hard disk. Then copy all of the files on the distribution

diskette to the subdirectory LT3VSI on the hard disk.

There is a "Browse" program on the diskette for viewing files. To get information about it, type:

```
BROWSE  BROWSE.DOC
```

Use PgDn to page through the file; press Esc to quit. Other file-viewing programs can be used, if preferred.

View or print the file LT3.HLP for further instructions and information. Or, follow the instructions in this manual.

The diskette contains the files listed below. The files whose extension is ".ARC" are Archive files. Each of them contains several files in compressed form. The program ARC.EXE is used to update ARC files and to list the names of files contained in them.

| File Name | Contents |
|---|---|
| ARC.EXE | The Archive program (see above and below). |
| BROWSE.COM | The Browse program, used to view files. |
| BROWSE.DOC | Documentation for Browse. |
| LT3.HLP | Help information for files on the distribution diskette. |
| LT3APV.ARC | Source code for the include files and the three programs of LT3VSI. |
| LT3DATA.ARC | Data files for a small example that was used to test LT3VSI. |
| LT3RESLT.ARC | Output files containing results of running LT3VSI on the data files in LT3DATA.ARC. |
| LTPREP.ARC | Include file and source code for LTPREP, a program that helps prepare data files for LT3VSI. |
| READ.ME | The file that should be read first. |

Using ARC.

To get a listing of the names of the files in an ARC file, type:

```
ARC  V  [path]filename
```

The extension can be omitted if it is ".ARC". For example, to list the names of the files in LT3APV.ARC, type:

```
ARC  V  LT3APV
```

To decompress and extract the files in an ARC file, type:

```
ARC  E  [path]filename
```

As with the "V" command, the extension can be omitted if it is ".ARC". For example, to extract all of the files in LT3APV.ARC, type:

```
ARC  E  LT3APV
```

Both the "V" and the "E" commands can be followed by a list of filenames to be listed or extracted from the ARC file. To extract all of the files from LT3RESLT.ARC whose extension is ".CNC" or ".LOG", type:

```
ARC  E  LT3RESLT  *.CNC  *.LOG
```

If all of the files are extracted from the ARC files, the extracted files will occupy approximately 400K bytes.

Description of the ARC files.

The file LT3APV.ARC contains the source code for the include files and the three programs of LT3VSI. The programs are the Water program WATER.APV, the Chemistry Initialization program CHEMINIT.APV, and the Chemistry Loop program CHEMLOOP.APV. They are coded to run on the FPS Scientific Computer with a VAX/VMS front-end computer. They can be modified fairly easily to run on a VAX or on a PC (AT-class or better).

The Water program is run by itself, reading a data file whose extension is ".WAD". It computes the steady state water pressure and velocities in the aquifer described by the data file. It produces printable output files.

The two chemistry programs are run in tandem. The Chemistry Initialization program is run first; it reads the interface file (.WIF) produced by the water program, and a chemistry data file (.CHD) and optionally a schedule file (.SCH). It produces an interface file (.CIF) and some other files. As soon as it has finished, the Chemistry Loop program is run. It reads files produced by the Initialization program and computes the time-varying concentrations of the four chemicals and the populations of the two microbes. It produces output files that can be printed.

The file LTPREP.ARC contains the source code for an include file and Fortran code for the program named LTPREP, which runs on PC's and helps prepare data files for LT3VSI. This program does not detect all of the possible data errors. See Section IV.1 for more information.

The file LT3DATA.ARC contains the data files for the small example used to test LT3VSI. The file LT3EXAM.WAD was generated by LTPREP from the files LT3EXAM.GEO and LT3EXAM.WAG. LT3EXAM.WAD is used for a water run.

The file LT3EXAM1.CHD was generated by LTPREP from the files LT3EXAM.GEO and LT3EXAM1.CHG. The file LT3EXAM2.CHD was created by using a text editor to make a few changes in LT3EXAM1.CHD. LT3EXAM2.CHD differs from LT3EXAM1.CHD in that the maxtime is 2.0 instead of 4.0, and NFLAG(5) is set to 1 to cause the program to read a .CUF file. LT3EXAM1.CHD is used for an "initial" run and LT3EXAM2.CHD is used for a "continuation" run. More information will be found below.

The file LT3EXAM.PIX is the picture file produced by LTPREP, renamed.

Also in LT3EXAM.ARC is the file LT3EXAM.SCH. This is a "schedule" file that specifies at what times the concentrations of the wells are to be changed.

The *.COM files in the LT3DATA.ARC file are "batch" files that run under VMS on a VAX and submit jobs to run on the FPS Scientific Computer attached to the VAX. WATRUN3.COM runs the Water program of LT3VSI. CHMRUN3.COM runs the chemistry programs (Chemistry Initialization followed by Chemistry Loop), in an "initial" run (starts at time 0.0), which produces a .CUF file to allow continuing the run. CHMGOON3.COM is similar to CHMRUN3.COM, but makes a "continuation" run, which reads the .CUF file produced by an earlier chemistry run and writes a new .CUF file when it is done.

The file RUNLT3VS.BAT in LT3DATA.ARC is a batch file that was used to run LT3VSI on an AT-class computer during development and early testing.

The file LT3RESLT.ARC contains the files that were produced by running LT3VSI on the example data files.

The *.LOG files show the "console" output from the runs of the three *.COM files. WATRUN3.LOG is the console output from the water run, producing the three files LT3EXAM.W*, and the interface file LT3EXAM.WIF which is not included on the diskette.

There were two chemistry runs. The file CHMRUN3.LOG contains the console output from the 4.0 day initial chemistry run. The files LT3EXAM1.* in LT3RESLT.ARC are the result files from this run.

The file CHMGOON3.LOG is the console output from the continuation chemistry run of 2.0 days. The files LT3EXAM2.* are the result files from this run.

Installing LT3VSI on an FPS computer.

First, install the files on the PC. Create a subdirectory named LT3VSI on the PC's hard disk, unless this has already been done. Then, copy the program and data files from the distribution diskette to the LT3VSI subdirectory on the hard disk. The files that are needed are:

ARC.EXE, LT3APV.ARC, LT3DATA.ARC, and LTPREP.ARC.

Next, use the ARC program to extract files. Type the following commands to extract the files that will be needed to run LT3VSI:

```
ARC  E  LT3APV
ARC  E  LT3DATA  LT3EXAM*.*  *.COM
ARC  E  LTPREP
```

Below is a list of the 30 files that should be extracted by the commands above:

| | | | |
|---|---|---|---|
| CBND.SIK | CHMRUN3.COM | CVELOC.SIB | LT3EXAM1.CHD |
| CCHEM.SIC | CNRK.SIL | CVELOC.SIK | LT3EXAM1.CHG |
| CCHEM.SIK | CPARAM.SIK | CWAT.SIW | LT3EXAM2.CHD |
| CCHEM.SIL | CPROP.SIB | LT3EXAM.GEO | LTPREP.FOR |
| CHEMINIT.APV | CRUNC.SIK | LT3EXAM.PIX | LTPREP.INC |
| CHEMLOOP.APV | CSIZE.SIB | LT3EXAM.SCH | WATER.APV |
| CHMAT.SIK | CSSIP.SIW | LT3EXAM.WAD | WATRUN3.COM |
| CHMGOON3.COM | | LT3EXAM.WAG | |

WARNING! The files whose extension is ".COM" are "batch" files for VMS. They are not executable files for the PC. Do not type the name of one of them at the DOS prompt on the PC, or undesirable results may occur!

Now upload the files. Log on to the VAX and use a file transfer protocol such as Kermit to upload the files described

below to the VAX. A total of 23 files should be uploaded.

```
*.APV   *.SI*   *.COM   *.WAD   *.CHD
```

Use the FPS software on the VAX to compile and link the three programs of LT3VSI. To compile the Water program, type the following command at the VMS prompt:

```
APFTN64/OPT=3/DEPCHK=WARN  WATER.APV
```

Other options can be used, if desired. There will be several "warnings" of "potential data dependencies" involving various arrays. It is not known what causes these warnings; the program apppears to work correctly. If the compilation is successful, except for these warnings, link the program by typing the following command:

```
APLINK64  WATER
```

To save file space, delete the object file:

```
DEL  WATER.AOB;*
```

The executable image file WATER.IMG is 717 blocks long. All files mentioned here are on the VAX, not the FPS.

Similarly, compile and link the Chemistry Initialization program:

```
APFTN64/OPT=3/DEPCHK=WARN  CHEMINIT.APV
APLINK64  CHEMINIT
DEL  CHEMINIT.AOB;*
```

There shouldn't be any warnings during compilation of CHEMINIT. The executable image file CHEMINIT.IMG is 777 blocks long.

Finally, compile and link the Chemistry Loop program:

```
APFTN64/OPT=3/DEPCHK=WARN  CHEMLOOP.APV
APLINK64  CHEMLOOP
DEL  CHEMLOOP.AOB;*
```

There will be several warnings of potential data dependencies during compilation of CHEMLOOP. There will also be a warning of

a label defined but not referenced. The executable image file CHEMLOOP.IMG is 756 blocks long.

See Section V for instructions on running the programs.

The distribution diskette includes the output files that were generated when the programs were run on an FPS computer with the given data files. It is possible to see that the programs have been installed and are working correctly by comparing their output with the output files supplied.

# III. List of variables

Section III.1 contains a list of the program variables that are used in input and output. These variables are listed in alphabetic order. The mathematical symbol corresponding to each variable from Volume 4 is shown, with a brief description of the purpose of the variable, and its units. The notation "dimless" means "dimensionless". There is a list of the meanings of the run control flags in Section III.2.

## III.1. Program variables involved in input/output

| Fortran variable | Math symbol | Meaning | Units |
|---|---|---|---|
| ALFNI1 | $\alpha_{ni}^{1}$ | nitrate maintenance rate coefficient for population 1 and anaerobic conditions | kg nitrate / kg cells |
| ALFO1 | $\alpha_{o}^{1}$ | oxygen maintenance rate coefficient for population 1 and aerobic conditions | kg oxygen / kg cells |
| ALFO2 | $\alpha_{o}^{2}$ | oxygen maintenance rate coefficient for population 2 and aerobic conditions | kg oxygen / kg cells |
| ALPH | $\alpha$ | current alpha parameter for SIP method | (dimless) |
| ALPHAS(I) | $\alpha_i$ | array containing values for ALPH | (dimless) |
| CO(K) | $C_o^*$ | constant concentration of chemical in feed stream entering inlet tank | ($kg/m^3$) |
| CHNAME(K) | | array that contains the names of the four chemicals: NUTRIENT, SUBSTRATE, OXYGEN, and NITRATE | |
| CIN(K) | $C_{in}^*$ | chemical concentration in inlet end mixing tank | ($kg/m^3$) |

| | | | |
|---|---|---|---|
| CNEW(I,J,K) | $C^*$ | free phase chemical concentration distribution (new values) | $(kg/m^3)$ |
| COLD(I,J,K) | $C^*$ | free phase chemical concentration distribution (current values) | $(kg/m^3)$ |
| CONC(K) | | new chemical concentrations for the injection wells at an event time | $\frac{\text{kg chem}}{\text{kg solution}}$ |
| COUT(K) | $C^*_{out}$ | chemical concentration in outlet end mixing tank | $(kg/m^3)$ |
| CSWELN(I,J,K) | $C^*_{sour}$ | injection well chemical concentrations | $\frac{\text{kg chem}}{\text{kg solution}}$ |
| DCHLX(I,J,K) | $D^*_{xx}$ | x-component of dispersion | $(m^2/day)$ |
| DCHLXY(I,J,K) | $D^*_{xy}$ | cross component of dispersion | $(m^2/day)$ |
| DCHLY(I,J,K) | $D^*_{yy}$ | y-component of dispersion | $(m^2/day)$ |
| DELTA1(I,J) | $\delta^1$ | constant for death rate of population 1 | (1/day) |
| DELTA2(I,J) | $\delta^2$ | constant for death rate of population 2 | (1/day) |
| DISPLX(I,J,K) | $\alpha^*_{dispx}$ | x-component of dispersivity | (m) |
| DISPLY(I,J,K) | $\alpha^*_{dispy}$ | y-component of dispersivity | (m) |
| DL0(K) | $D^*_{L_0}$ | molecular diffusion coefficient of compound in free solution | $(m^2/day)$ |
| DT0 | $\Delta t$ | time increment $t_{n+1} = t_n + \Delta t$ | (days) |
| DT1(K) | | maximum $\Delta t$ that satisfies stability criterion: $\Delta t \leq$ DT1(K) for all K | (days) |
| DX(I) | $\Delta x_i$ | node spacing in x-direction; DX(I) = XNODE(I+1)-XNODE(I) | (m) |

| | | | |
|---|---|---|---|
| DY(J) | $\Delta y_j$ | node spacing in y-direction; DY(J) = YNODE(J+1)-YNODE(J) | (m) |
| EPS(I,J) | $\epsilon$ | porosity of porous medium | $(m^3/m^3)$ |
| ETANI1 | $\eta_{ni}^1$ | nitrate use coefficient for population 1 and anaerobic conditions | $\frac{\text{kg nitrate}}{\text{kg substrate}}$ |
| ETIME | | an event time at which the chemical concentrations at an injection well are changed | (days) |
| GAMMO1 | $\gamma_o^1$ | oxygen use coefficient for population 1 and aerobic conditions | $\frac{\text{kg oxygen}}{\text{kg substrate}}$ |
| GAMMO2 | $\gamma_o^2$ | oxygen use coefficient for population 2 and aerobic conditions | $\frac{\text{kg oxygen}}{\text{kg substrate}}$ |
| HEAD1 | | first text line read from file xxxxxxxx.WAD or xxxxxxxx.CHD | |
| HEAD2 | | second text line read from file xxxxxxxx.WAD or xxxxxxxx.CHD | |
| HIN | $H_{in}$ | hydraulic head field at y = 0 | (m water) |
| HNEW(I,J) | H | hydraulic head field (new value) | (m water) |
| HOLD(I,J) | H | hydraulic head field (current value) | (m water) |
| HOUT | $H_{out}$ | hydraulic head field at y = Ly | (m water) |
| I | | index for arrays, x direction | |
| IC | | iteration count in the chemistry loop | |
| ICOUNT | | iteration count for SIP method | |
| IX | | constant = maximum value allowed for NSLXP1; used to dimension arrays | |
| IY | | constant = maximum value allowed for NSLYP1; used to dimension arrays | |

| | | | |
|---|---|---|---|
| J | | index for arrays, y direction | |
| K | | index for selecting one of the four chemicals; see CHNAME(K) | |
| KCLAY(K) | $K_s^*$ | linear equilibrium distribution constant for strongly sorbing surfaces (includes clay minerals) | ($m^3$/kg strongly sorbing particles) |
| KNI1 | $K_{ni}^1$ | one half maximum nitrate based saturation constant for population 1 using nitrate | (kg/$m^3$) |
| KNINU1 | $K_{ninu}^1$ | one half maximum nitrate based saturation constant for population 1 using nutrients | (kg/$m^3$) |
| KO1 | $K_o^1$ | one half maximum saturation constant for population 1 using oxygen | (kg/$m^3$) |
| KO2 | $K_o^2$ | one half maximum saturation constant for population 2 using oxygen | (kg/$m^3$) |
| KONI1 | $K_{oni}^1$ | one half maximum nitrate based inhibition constant for population 1 using nitrate | (kg/$m^3$) |
| KONU1 | $K_{onu}^1$ | one half maximum saturation constant for population 1 using nutrients | (kg/$m^3$) |
| KONU2 | $K_{onu}^2$ | one half maximum saturation constant for population 2 using nutrients | (kg/$m^3$) |
| KORG(K) | $K_{org}^*$ | linear equilibrium distribution constant for organics | ($m^3$/kg organics) |
| KSAND(K) | $K_{ws}^*$ | linear equilibrium distribution constant for mildly sorbing surfaces | ($m^3$/kg weakly sorbing particles) |
| KSATXX(I,J) | $K_{sxx}$ | x-component of saturated hydraulic conductivity tensor | (m/day) |

| | | | |
|---|---|---|---|
| KSATYY(I,J) | $K_{syy}$ | y-component of saturated hydraulic conductivity tensor | (m/day) |
| KSNI1 | $K^1_{sni}$ | one half maximum nitrate based saturation constant for population 1 using substrate | ($kg/m^3$) |
| KSO1 | $K^1_{so}$ | one half maximum saturation constant for population 1 using substrate | ($kg/m^3$) |
| KSO2 | $K^2_{so}$ | one half maximum saturation constant for population 2 using substrate | ($kg/m^3$) |
| KSOM1 | $K^1_{som}$ | one half saturation coefficient for maintenance in population 1 | ($kg/m^3$) |
| KSOM2 | $K^2_{som}$ | one half saturation coefficient for maintenance in population 2 | ($kg/m^3$) |
| LAMDA(I,J,K) | $\lambda^*$ | overall first order loss coefficient | (1/day) |
| MUNI1 | $\mu^1_{ni}$ | maximum specific nitrate based growth rate for heterotrophic population 1 | (1/day) |
| MUO1 | $\mu^1_o$ | maximum specific growth rate for heterotrophic population 1 | (1/day) |
| MUO2 | $\mu^2_o$ | maximum specific growth rate for heterotrophic population 2 | (1/day) |
| NALPH | | number of ALPHAS | |
| NBSOUR | | number of chemical sources in flow field | |
| NEXTW | | number of extraction wells | |
| NFLAG(I) | | run control flags (see section III.2) | |
| NINJW | | number of injection wells | |
| NLSOR | | maximum number of iterations allowed in SIP method | |
| NMOD | | specifies number of times to use each value of ALPH in SIP method | |

| | | | |
|---|---|---|---|
| NPRT | | number of times to print chemical data | |
| NSLXM1 | $N_{x-1}$ | number of internal nodes on transverse coordinate | |
| NSLXP1 | $N_{x+1}$ | total number of nodes on transverse coordinate | |
| NSLXXX | $N_x$ | number of subintervals in the interval $[0,L_x]$ | |
| NSLYM1 | $N_{y-1}$ | number of internal nodes on longitudinal coordinate | |
| NSLYP1 | $N_{y+1}$ | total number of nodes on longitudinal coordinate | |
| NSLYYY | $N_y$ | number of subintervals in the interval $[0,L_y]$ | |
| PCTCLA(I,J) | $M_s/M_t$ | mass fraction of strongly sorbing particles in porous medium (includes clay fraction) | $\frac{\text{kg strongly sorbing}}{\text{kg soil}}$ |
| PCTORG(I,J) | $M_o/M_t$ | mass fraction of organic matter in porous medium | $\frac{\text{kg organic matter}}{\text{kg soil}}$ |
| PCTSAN(I,J) | $M_{ws}/M_t$ | mass fraction of weakly sorbing particles in porous medium | $\frac{\text{kg weakly sorbing}}{\text{kg soil}}$ |
| POP1(I,J) | $N_1(t)$ | population 1 cell mass in the unit pore volume at time t | $\frac{\text{kg cell}}{\text{kg soil}}$ |
| POP2(I,J) | $N_2(t)$ | population 2 cell mass in the unit pore volume at time t | $\frac{\text{kg cell}}{\text{kg soil}}$ |
| PSIO1 | $\psi_o^1$ | nutrient use coefficient for population 1 and aerobic conditions | $\frac{\text{kg nutrient}}{\text{kg substrate}}$ |
| PSIO2 | $\psi_o^2$ | nutrient use coefficient for population 2 and aerobic conditions | $\frac{\text{kg nutrient}}{\text{kg substrate}}$ |

| | | | |
|---|---|---|---|
| QCHM1S(I,J,K) | $Q^{*}_{so}$ | source rate density for buried sources (The program reads kg chem/day; it divides the input values by the appropriate volumes) | $\frac{\text{kg chem}}{m^3\text{-day}}$ |
| QTEMP | | temporary variable used for reading QWELIN, QWELOT, CSWELN, and QCHM1S | |
| QWELIN(I,J) | $Q_{inj}$ | mass density rate of injection well (see note under QWELOT below) | $\frac{\text{kg water}}{m^3\text{-day}}$ |
| QWELOT(I,J) | $Q_{out}$ | mass density rate of extraction well (For both QWELIN and QWELOT, the program reads kg water/day; it divides the input values by the appropriate volumes) | $\frac{\text{kg water}}{m^3\text{-day}}$ |
| RETARD(I,J,K) | $1+R^{*}$ | retardation = 1 + retention | (dimless) |
| RHOBD | $\rho_b$ | average bulk density of porous medium | ($kg/m^3$) |
| RHOCLA | $\rho_s$ | average particle density of strongly sorbing fraction | ($kg/m^3$) |
| RHOORG | $\rho_{org}$ | average particle density of organic matter fraction | ($kg/m^3$) |
| RHOSND | $\rho_{ws}$ | average particle density of weakly sorbing fraction | ($kg/m^3$) |
| RHOWAT | $\rho_w$ | density of water | ($kg/m^3$) |
| RMSEA | | actual root mean square error in SIP method | |
| RMSER | | relative root mean square error in SIP method | |
| RSNINU1(I,J) | $r^{1}_{sninu}$ | nitrate utilization rate by microbial population 1 | $\frac{\text{kg substrate}}{\text{kg cell-day}}$ |
| RSONU1(I,J) | $r^{1}_{sonu}$ | oxygen utilization rate by microbial population 1 | $\frac{\text{kg substrate}}{\text{kg cell-day}}$ |
| RSONU2(I,J) | $r^{2}_{sonu}$ | oxygen utilization rate by microbial population 2 | $\frac{\text{kg substrate}}{\text{kg cell-day}}$ |

| | | | |
|---|---|---|---|
| STRING | | temporarily holds strings of text read from data files | |
| T0 | | starting time for chemical processing (days) (0.0 for an initial run) | |
| THENI1 | $\theta^{1}_{ni}$ | nutrient use coefficient for population 1 and nitrate conditions | $\frac{\text{kg nutrient}}{\text{kg substrate}}$ |
| TIME | t | elapsed time in days from commencing injection and/or pumping (days since beginning of initial run) | (days) |
| TIMEI | | cycle time in SIP method | (seconds CPU time) |
| TIMES | | startup time in SIP method | (seconds CPU time) |
| TLRNWA | | tolerance for actual root mean square error convergence criterion for SIP method | |
| TLRNWR | | tolerance for relative root mean square error convergence criterion for SIP method | |
| TMAX | | length of time for chemical processing | (days) |
| TORT(I,J) | $\alpha_{tort}$ | tortuosity factor (0.67) | (dimless) |
| VLXX(I,J) | $q_x$ | x-component of Darcy velocity | (m/day) |
| VLYY(I,J) | $q_y$ | y-component of Darcy velocity (When VLXX and VLYY are first computed, they represent $q_x$ and $q_y$) | (m/day) |
| VLXX(I,J) | $U_x$ | x-component of average seepage velocity vector | (m/day) |
| VLYY(I,J) | $U_y$ | y-component of average seepage velocity vector (VLXX and VLYY are subsequently changed to represent $U_x$ and $U_y$) | (m/day) |
| XI | $x_i$ | one of the x-nodes | (m) |
| YJ | $y_j$ | one of the y-nodes | (m) |

| | | | |
|---|---|---|---|
| XLAMIR(I,J,K) | $\lambda^{*}_{irr}$ | first order free phase loss rate constant for irreversible loss processes, e.g. chemical reactions | (1/day) |
| XLW | $L_w$ | vertical dimension of aquifer | (m) |
| XLYIN | $L_{in}$ | length of the inlet end mixing tank | (m) |
| XLYOUT | $L_{out}$ | length of the outlet end mixing tank | (m) |
| XMASIN(K) | $M^{*}_{Inlet(t)}$ | cumulative chemical mass inflow or outflow at inlet end | (kg) |
| XMASOT(K) | $M^{*}_{Outlet(t)}$ | cumulative chemical mass inflow or outflow at outlet end | (kg) |
| XMASS(K) | $M^{*}_{Aq(t)}$ | total chemical mass in aquifer | (kg) |
| XMFONW(K) | $M^{*}_{Lost(t)}$ | cumulative first order loss of chemical mass | (kg) |
| XMSOUR(K) | $M^{*}_{Source(t)}$ | cumulative chemical mass from buried sources | (kg) |
| XNODE(I) | x | transverse spatial component | (m) |
| XNODE(NSLXP1) | $L_x$ | transverse dimension (width) of aquifer | (m) |
| XSLMIR(I,J,K) | $\lambda^{s*}_{irr}$ | first order sorbed phase loss rate constant for irreversible processes | (1/day) |
| YNODE(J) | y | longitudinal spatial component | (m) |
| YNODE(NSLYP1) | $L_y$ | longitudinal dimension (length) of aquifer | (m) |
| YSNI1 | $Y^{1}_{sni}$ | yield coefficient for microbial population 1 using nitrate | $\frac{\text{kg cells}}{\text{kg substrate}}$ |
| YSO1 | $Y^{1}_{so}$ | yield coefficient for microbial population 1 using oxygen | $\frac{\text{kg cells}}{\text{kg substrate}}$ |
| YSO2 | $Y^{2}_{so}$ | yield coefficient for microbial population 2 using oxygen | $\frac{\text{kg cells}}{\text{kg substrate}}$ |

ZTHRSH — zero threshold: numbers in arrays whose magnitudes are less than ZTHRSH are printed as zero.

III.2. Meanings of run control flags NFLAG(I)

"Display" means that the information is shown on the screen and is not written on a file, unless the screen output is re-directed to a file. "Write", for NFLAGS 10, 11, 12, 13, 14, and 15, means that the information is written on one of the two files used for debugging output. See the notes below. Flags not mentioned, such as NFLAG(7), are not used in this version of the program. Flags that are used in both water and chemistry phases have the same function. The only such flag at present is NFLAG(8).

Flags used during the water phase

NFLAG(1)=0 means: compute the water pressure field only. The chemistry phases cannot be run in this case.

NFLAG(1)=1 means: compute the water pressure field and the velocity components.

NFLAG(2)=0 means: compute an initial guess for the water pressure.

NFLAG(2)=1 means: read an initial guess for the water pressure from the water data file.

NFLAG(3)=0 means: display ICOUNT, ALPH, RMSEA, RMSER, during the SIP process.

NFLAG(3)=1 means: display the above information, plus the values of XDUM and DELX at the four "interior" corners.

NFLAG(8)=0 means: write two-dimensional arrays in narrow (80 column) format.
NFLAG(8)=1 means: write two-dimensional arrays in wide (132 column) format.

NFLAG(10)=1 means: write the matrix elements which define the water pressure field.
NFLAG(10)=0 means: do not write the matrix elements.

NFLAG(11)=1 means: write the adjusted boundary matrix elements.
NFLAG(11)=0 means: do not write the adjusted elements.

NFLAG(12)=1 means: write the source and boundary component contributions to the over all "known vector".
NFLAG(12)=0 means: do not write the "known vector".

NFLAG(13)=1 means: write the max and min DELX's.
NFLAG(13)=0 means: do not write the DELX's.

Note: the outputs enabled by NFLAG's 10, 11, 12, and 13 are written on the file 'xxxxxxxx.WDB' during the water phase.

Flags used during the chemistry phases

NFLAG(4)=0 means: do not use a schedule file "xxxxxxxx.SCH". The injection well chemical concentrations specified in the chem data file "xxxxxxxx.CHD" will remain unchanged throughout the run.
NFLAG(4)=1 means: read a schedule file "xxxxxxxx.SCH" and alter the injection well chemical concentrations at the specified event times, as specified by the data in the file.

NFLAG(5)=0 means: read initial chemical concentrations from the chemical data file.

NFLAG(5)=1 means: read initial chemical concentrations from the unformatted file written by a previous run of the chemistry loop phase. See NFLAG(6) below.

NFLAG(6)=1 means: at the end of a chemistry loop phase run, write an unformatted file containing the final chemical concentrations. This file can be read in by a subsequent chemistry run as initial data. See NFLAG(5) above.

NFLAG(6)=0 means: do not write the unformatted file described above.

NFLAG(8)=0 means: write two-dimensional arrays in narrow (80 column) format.

NFLAG(8)=1 means: write two-dimensional arrays in wide (132 column) format.

NFLAG(9)=1 means: read a value for ZTHRSH from the chemical data file, during the chemistry phase.

NFLAG(9)=0 means: do not read ZTHRSH during the chemistry phase; use the value that was read in during the water phase.

NFLAG(14)=1 means: write the matrix elements defining the chemical field.

NFLAG(14)=0 means: do not write the matrix elements.

NFLAG(15)=1 means: write the coordinates of the P* points.

NFLAG(15)=0 means: do not write the above.

NFLAG(16)=0 means: use 4-point interpolation method in the CLAG function, which computes the chemical concentrations at the P* points.

NFLAG(16)=1 means: use 25-point Lagrange interpolation method in the CLAG function (this takes much more time).

NFLAG(17)=1 means: compute and write the cumulative chemical masses (XMASS, and so on) on the file 'xxxxxxxx.CNC'.

NFLAG(17)=0 means: do not compute and do not write the above.

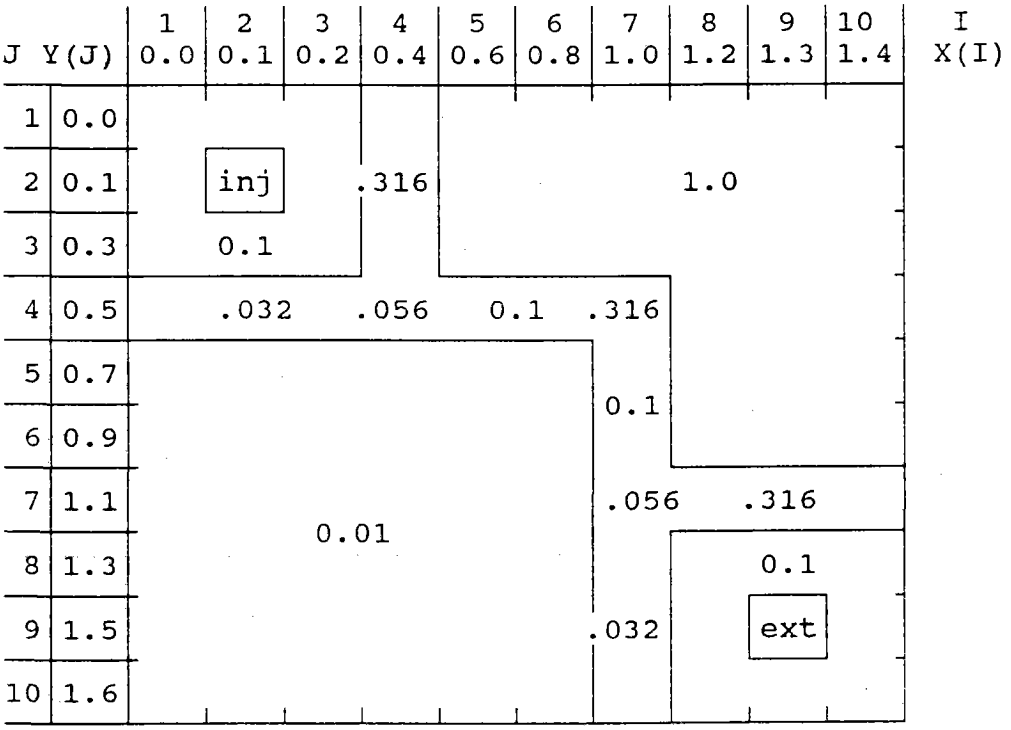
Note: the outputs enabled by NFLAG's 14 and 15 are written on the file 'xxxxxxxx.CDB' during the chemistry phases.

# IV. Preparation of data files

Example input data files are presented and described in this section. These example data files are included on the distribution diskette. It is recommended that these files be used as input to the program after installation, to verify that the program is working properly. The files can then be modified to describe an actual scenario for model simulation.

## IV.1. Preparation of nonhomogeneous data for an example aquifer

The following is a picture of the example aquifer used for illustration in this manual. Normally, one would use more nodes than this; the number of nodes was kept small so that the listings of the files would not take up too much space in the manual. It also takes less computer time to run a small model.



This aquifer has four regions and two wells. The value of a particular soil property may be different in different regions, but is constant within a region. This example shows how to set up the data for the hydraulic conductivity.

The illustration shows the geometry of the aquifer. The width is 1.4 meters with 10 nodes in the X direction; the length is 1.6 meters with 10 nodes in the Y direction. There is one injection well (inj) at X-Y location (0.1,0.1). There is one extraction well (ext) at X-Y location (1.3,1.5).

The aquifer has two regions of medium conductivity (0.1 m/day) at the upper left and the lower right of the picture, one region of low conductivity (0.01 m/day) at the lower left, and one region of high conductivity (1.0 m/day) at the upper right. The values (0.1, 0.01, and 1.0 m/day) are the values of the X component of the saturated hydraulic conductivity KSATXX(I,J). In this example, the Y component KSATYY(I,J) is the same as KSATXX(I,J).

The picture shows the boundaries between the regions. The value at a node on a boundary is the geometric mean of the values at the adjacent nodes on either side, either those on left and right, or those above and below. For example, along the boundary from (0.0,0.5) to (0.2,0.5), the value above the boundary is 0.1 and the value below is 0.01. The geometric mean is:

SQRT(0.1 * 0.01),

or 0.0316228, which has been rounded to 0.032. At the corners of regions, or where there are junctions between boundaries, the values of nearby boundary points must be computed first and then used in computing the special points. For example, at the point (0.4,0.5), the boundary value on the right is 0.1 and the boundary value on the left is 0.0316228. The geometric mean of these values is 0.0562341, rounded to 0.056.

The same geometry is used for each of the hydraulic and chemical parameters that can vary over the aquifer. The values

in the regions will be different for each parameter and the boundary values must be computed as described above. Then these values must be put into the two data files described in sections IV.2 and IV.3.

One can compute the values at the various nodes and create the data files xxxxxxxx.WAD and xxxxxxxx.CHD by using a text editor. However, this can be VERY tedious labor. To make it easier to create the data files, the program LTPREP has been coded. To use LTPREP, you create abbreviated forms of the data files, together with a "Geometry" file. In this example, the abbreviated data files have the extensions .WAG and .CHG. The geometry file has the extension .GEO. LTPREP allows any file names and extensions for its input and output files.

The geometry is specified in the LT3EXAM.GEO file as follows:

```
X  Y  (NO. OF INTERIOR NODES)
8  8
X-COORD POSITIONS (M)
 0.0   0.1   0.2   0.4   0.6   0.8   1.0   1.2
 1.3   1.4
Y-COORD POSITIONS (M)
 0.0   0.1   0.3   0.5   0.7   0.9   1.1
 1.3   1.5   1.6
BOUNDARIES
2  (0.0,0.5)  (1.0,0.5)  (1.0,1.6)
1  (0.4,0.0)  (0.4,0.5)
1  (1.4,1.1)  (1.0,1.1)
-1
```

The fourth section of the geometry file specifies internal boundaries, between regions of different values. The boundaries must be horizontal or vertical, and are specified by giving the coordinates of their endpoints. The data for each boundary line has the form:

(x1,y1) (x2,y2)

where x1, y1, x2, and y2 are real numbers representing the coordinates of nodes specified in the second and third sections of the file, and either x1 = x2, or y1 = y2. Data in the file has the form:

```
      n  (x1,y1)  (x2,y2)  ...  (xn+1,yn+1)
```

where n is an integer (0 <= n <= 10), followed by n+1 pairs of coordinates. This represents n connected boundary lines:

```
      (x1,y1)  (x2,y2)
      (x2,y2)  (x3,y3)
            ...
      (xn,yn)  (xn+1,yn+1)
```

The special case when n = 0:

```
      0  (x1,y1)
```

represents a single boundary point. The data is ended by a line containing a single -1:

```
      -1
```

In the example above, the data line:

```
      2  (0.0,0.5)  (1.0,0.5)  (1.0,1.6)
```

represents two of the boundary lines: the line from (0.0,0.5) to (1.0,0.5), and the line from (1.0,0.5) to (1.0,1.6).

In the abbreviated data files, one uses commands to specify: (1) where to include the first three parts of the geometry file, and (2) where to generate data for the LT3VSI program to read. Here is a listing of the file LT3EXAM.WAG:

```
  DATE IS 9/21/1990 - EXAMPLE DATA FOR MODEL LT3VSI V1.2 [4 chem]
  ONE INJECTION WELL; ONE EXTRACTION WELL
NLSOR  NMOD  NALPH  TLRNWA  TLRNWR  ZTHRSH  (RUN CONTROL DATA)
 3000    2     4    1.0E-8  1.0E-8  1.0E-8
DECISION FLAGS
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
ALPHAS
1.0  0.90  0.95  0.75
#GEOMETRY
AQUIFER THICKNESS (M)
0.3
WATER DENSITY (KG/M^3)
1000.0
X-COMPONENT OF HYDRAULIC CONDUCTIVITY (M/DAY)
#VALUES
(0.1,1.5) 0.010 [low]
(0.1,0.1) 0.100 [medium]
(1.3,1.5) 0.100 [medium]
(1.3,0.1) 1.000 [high]
(0,0) 0
Y-COMPONENT OF HYDRAULIC CONDUCTIVITY (M/DAY)
#VALUES
(0.1,1.5) 0.010 [low]
(0.1,0.1) 0.100 [medium]
(1.3,1.5) 0.100 [medium]
(1.3,0.1) 1.000 [high]
(0,0) 0
NO. OF INJECTION AND EXTRACTION WELLS
          1             1
 XI   YJ  STRENGTH OF INJECTION WELLS (KG WATER/DAY)
0.1  0.1    7.5
 XI   YJ  STRENGTH OF EXTRACTION WELLS (KG WATER/DAY)
1.3  1.5    4.0
INLET AND EXIT PORTS PRESSURE HEADS (M)
 1.0       0.0
```

LTPREP copies most of the abbreviated input file to the output file. Wherever there is an LTPREP command in the input file, LTPREP replaces the command by other information. LTPREP commands have a "#" in column 1, followed by a "G" or "V" in column 2. The rest of the line is ignored by LTPREP. In the example file above, the line

#GEOMETRY

causes LTPREP to copy the first three parts of the LT3EXAM.GEO file, up to the BOUNDARIES line, and write them out in place of the #G command.

The data for the x-component of the conductivity is specified as follows.

```
X-COMPONENT OF HYDRAULIC CONDUCTIVITY (M/DAY)
#VALUES
(0.1,1.5)  0.010 [low]
(0.1,0.1)  0.100 [medium]
(1.3,1.5)  0.100 [medium]
(1.3,0.1)  1.000 [high]
(0,0)  0
```

The line beginning "X-COMPONENT OF ...." is copied to the .WAD file. The #V command is followed by a variable number of lines, specifying values for interiors of regions. There should be one value in each region. The LTPREP program propagates the value throughout the region. Each line of data must have the form:

(x,y) val

where x and y are real numbers specifying the location of a nodal point inside one of the regions, and val is the value to be assigned to all points within that region. The data is ended by a line of the form:

(0,0) 0.0

The notations in brackets such as "[low]" are comments that are ignored by LTPREP. LTPREP is easy to use; simply type its name at the DOS prompt, and it will ask for the names of the geometry file, the input data file, and the output file. In addition to these files, LTPREP produces a file named LTPICT.XXX, which is a picture of the aquifer. For the geometry and LT3EXAM.WAG files shown above, LTPREP writes the following LTPICT.XXX file:

```
LTPREP Picture File for lt3exam.geo

++++++++++++
+:::#::::::+
+:::#::::::+
+:::#::::::+
+#######:::+
+::::::#:::+
+::::::#:::+
+::::::####+
+::::::#:::+
+::::::#:::+
+::::::#:::+
++++++++++++
```

LTPREP detects some errors during its processing and writes error messages. Other errors are not detected; some of them can be discovered by examining the picture file. The file shown above is correct. The "+" symbols around the outside represent the outside boundary. The "#" symbols represent internal boundaries between regions. Any "+" symbols inside the diagram represent unevaluated internal boundary points; these are caused by data errors. The ":" symbols represent points within regions whose values have been determined by propagating the values specified in the data file. Any "?" symbols represent points within a region which have not been evaluated because no value was specified in that region. LTPREP does not detect the situation where more than one value is specified in a region. Also, the picture file shows the situation resulting from the last #V command in the data file.

Below is a listing of part of the LT3EXAM.WAD file produced by LTPREP from the .GEO and .WAG files listed above.

```
 ...     ...     ...     ...
ALPHAS
1.0  0.90  0.95  0.75
X  Y  (NO. OF INTERIOR NODES)
8  8
X-COORD POSITIONS (M)
 0.0   0.1   0.2   0.4   0.6   0.8   1.0   1.2
 1.3   1.4
Y-COORD POSITIONS (M)
 0.0   0.1   0.3   0.5   0.7   0.9   1.1
 1.3   1.5   1.6
AQUIFER THICKNESS (M)
0.3
WATER DENSITY (KG/M^3)
1000.0
X-COMPONENT OF HYDRAULIC CONDUCTIVITY (M/DAY)
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*3.16E-02   1*5.62E-02   2*1.00E-01   1*3.16E-01   3*1.00E+00
   6*1.00E-02   1*1.00E-01   3*1.00E+00
   6*1.00E-02   1*1.00E-01   3*1.00E+00
   6*1.00E-02   1*5.62E-02   3*3.16E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
Y-COMPONENT OF HYDRAULIC CONDUCTIVITY (M/DAY)
 ...     ...     ...     ...
```

The portion of the file listed above shows the information

that replaced the #G command and the first #V command. LTPREP's output takes advantage of the repeat feature of Fortran's list-directed input. The complete file is included on the distribution diskette for LT3VSI and is listed in Section IV.2.

## IV.2. Example input hydrology data file: LT3EXAM.WAD

This section shows an example hydrology and aquifer geometry file. The file must have a name with the extension ".WAD"; it is read in by subroutine FLOREAD in module RWWSIW.FOR. A complete listing of the file is shown followed by a detailed discussion of the data.

```
    DATE IS 9/21/1990 - EXAMPLE DATA FOR MODEL LT3VSI V1.2 [4 chem]
    ONE INJECTION WELL; ONE EXTRACTION WELL
NLSOR  NMOD  NALPH  TLRNWA  TLRNWR  ZTHRSH  (RUN CONTROL DATA)
 3000    2     4    1.0E-8  1.0E-8  1.0E-8
DECISION FLAGS
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
ALPHAS
1.0  0.90  0.95  0.75
X  Y  (NO. OF INTERIOR NODES)
8  8
X-COORD POSITIONS (M)
 0.0   0.1   0.2   0.4   0.6   0.8   1.0   1.2
 1.3   1.4
Y-COORD POSITIONS (M)
 0.0   0.1   0.3   0.5   0.7   0.9   1.1
 1.3   1.5   1.6
AQUIFER THICKNESS (M)
0.3
WATER DENSITY (KG/M^3)
1000.0
X-COMPONENT OF HYDRAULIC CONDUCTIVITY (M/DAY)
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*3.16E-02   1*5.62E-02   2*1.00E-01   1*3.16E-01   3*1.00E+00
   6*1.00E-02   1*1.00E-01   3*1.00E+00
   6*1.00E-02   1*1.00E-01   3*1.00E+00
   6*1.00E-02   1*5.62E-02   3*3.16E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
Y-COMPONENT OF HYDRAULIC CONDUCTIVITY (M/DAY)
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*3.16E-02   1*5.62E-02   2*1.00E-01   1*3.16E-01   3*1.00E+00
   6*1.00E-02   1*1.00E-01   3*1.00E+00
   6*1.00E-02   1*1.00E-01   3*1.00E+00
   6*1.00E-02   1*5.62E-02   3*3.16E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
NO. OF INJECTION AND EXTRACTION WELLS
          1             1
 XI   YJ  STRENGTH (INJECTION WELLS) (KG WATER/DAY)
0.1  0.1    7.5
 XI   YJ  STRENGTH (EXTRACTION WELLS) (KG WATER/DAY)
1.3  1.5    4.0
INLET AND EXIT PORTS PRESSURE HEADS (M)
 1.0       0.0
```

As explained in Section IV.1, the LT3EXAM.WAD file above was prepared by creating files named LT3EXAM.GEO and LT3EXAM.WAG. These files were read by the program LTPREP, which produced the file above. In this small example, LTPREP does not save very much work. However, for larger data files, it makes data file preparation much easier than it would be without LTPREP.

In the rest of this section, the data read in by subroutine FLOREAD and the computations it performs are described. For input data, there is a brief description of the information being read, as well as the names of the variables being read in. This is followed by the example data, enclosed in a box. A brief description of computations is given, along with a description of the variables involved. The List of Variables in Section III provides more information about the variables.

There are two kinds of data in the file. The data used by the program is numerical, either real or integer form. Real numbers are handled by the program as double-precision. To help users understand and modify the data file, there is also text data. The first two lines of text are read by the program and printed out on the files xxxxxxxx.WAO and xxxxxxxx.WPV. The rest of the text lines are read and discarded. They may contain any desired information or may be left blank, but they must be there.

The numerical data is read using Fortran's "list-directed" input format. This is a "free-form" input: numbers may occupy any number of positions; they are separated by spaces or commas or ends-of-lines. However, each Fortran READ statement begins reading a new line. Any numbers or other data left on a line when the READ finishes are discarded. Here is the description of the actions of FLOREAD as it reads the data file xxxxxxxx.WAD:

First, FLOREAD reads a two-line heading HEAD1, HEAD2 that will be printed out on files xxxxxxxx.WAO and xxxxxxxx.WPV:

```
DATE IS 9/21/1990 - EXAMPLE DATA FOR MODEL LT3VSI V1.2 [4 chem]
ONE INJECTION WELL; ONE EXTRACTION WELL
```

Next, FLOREAD reads the run control information:

```
NLSOR  NMOD  NALPH  TLRNWA  TLRNWR  ZTHRSH  (RUN CONTROL DATA)
 3000     2      4  1.0E-8  1.0E-8  1.0E-8
```

NLSOR, NMOD, NALPH, TLRNWA, and TLRNWR are the SIP method parameters; they are discussed in Section IV.4. ZTHRSH is the "zero threshold"; when arrays are printed out, numbers whose absolute values are less than ZTHRSH are printed as ".00". If the value specified for ZTHRSH is larger than 0.99, FLOREAD sets it to 0.99.

FLOREAD next reads the 20 control flags NFLAG(I):

```
DECISION FLAGS
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

In this case:

NFLAG(1) = 1 means compute the water pressure field and the velocity components. Choose NFLAG(1) = 0 if computation of the water pressure only is desired. This would be appropriate while attempting to find suitable values for the SIP method variables. When NFLAG(1) = 0, the chemistry phase cannot be executed.

NFLAG(8) = 0 means write two-dimensional arrays in narrow (80 column) format. With a printer that can print 132 or more characters per line, choose NFLAG(8) = 1; with a printer which can only print 80 characters per line, choose 0. The wider format allows the program to print more numbers per line. If NFLAG(8) were changed to 1, the data above would appear as follows:

```
DECISION FLAGS
1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
```

Some of the other NFLAG's cause various items of information to be written on the debug file xxxxxxxx.WDB, to aid in testing and debugging the program. Some of the NFLAG's are not used; the meanings of those that are used are described in Section III.2.

FLOREAD reads ALPHAS, an array of values for ALPH. NALPH, which must be in the range 1 to 10, specifies the number of ALPH's to be read. See Section IV.4 for a discussion of the ALPH's.

```
ALPHAS
1.0  0.90  0.95  0.75
```

The grid geometry is read next, and some related variables are computed. FLOREAD reads the number of interior x and y nodes NSLXM1, NSLYM1:

```
X  Y  (NO. OF INTERIOR NODES)
8  8
```

In this example, the number of nodes is the same in both the X and Y directions. The program allows these numbers to be different.

FLOREAD computes the number of intervals:

NSLXXX = NSLXM1+1

NSLYYY = NSLYM1+1

and the total number of nodes, including boundary nodes:

NSLXP1 = NSLXXX+1

NSLYP1 = NSLYYY+1

If NSLXP1 > IX or NSLYP1 > IY then FLOREAD prints the error message:

```
Too many grid points for fixed-size arrays.
Change IX and/or IY, re-compile, & re-link.
```

and stops.

IX and IY are symbolic constants (parameters) in the Fortran program that are used to dimension the arrays. The size of the aquifer grid must not exceed these dimensions. IX and IY can be increased to accommodate larger grids, by editing the "include"

file CSIZE.SIB, re-compiling all modules of the program, and re-linking. This causes the program to require more computer memory; if it needs more memory than is available, the program cannot be run.

FLOREAD sets the arrays QWELIN(I,J) and QWELOT(I,J) to zero.

The nodal positions XNODE(1) through XNODE(NSLXP1) are read in, remembering that NSLXP1 is two (2) greater than the value NSLXM1 that was read in:

```
X-COORD POSITIONS (M)
 0.0   0.1   0.2   0.4   0.6   0.8   1.0   1.2
 1.3   1.4
```

The nodes do not have to be equally spaced. They should be closely spaced next to the boundaries and on all sides of each well and each buried source. The first node XNODE(1) should be zero.

The nodal positions YNODE(1) through YNODE(NSLYP1) are read in, remembering that NSLYP1 is two (2) greater than the value NSLYM1 that was read in:

```
Y-COORD POSITIONS (M)
 0.0   0.1   0.3   0.5   0.7   0.9   1.1
 1.3   1.5   1.6
```

See the note above under X-COORD POSITIONS. The first node YNODE(1) should be zero.

FLOREAD reads the aquifer vertical dimension XLW (M):

```
AQUIFER THICKNESS (M)
0.3
```

FLOREAD calculates the spacing between the nodes:

DY(J) = YNODE(J+1)-YNODE(J) for J=1 to NSLYYY

DX(I) = XNODE(I+1)-XNODE(I) for I=1 to NSLXXX

FLOREAD reads the water density RHOWAT:

```
WATER DENSITY (KG/M^3)
1000.0
```

The X-component of the saturated hydraulic conductivity is read next. NSLYP1 rows are read, with row J containing values KSATXX(1,J) through KSATXX(NSLXP1,J):

```
X-COMPONENT OF HYDRAULIC CONDUCTIVITY (M/DAY)
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*3.16E-02   1*5.62E-02   2*1.00E-01   1*3.16E-01   3*1.00E+00
   6*1.00E-02   1*1.00E-01   3*1.00E+00
   6*1.00E-02   1*1.00E-01   3*1.00E+00
   6*1.00E-02   1*5.62E-02   3*3.16E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
```

There must be NSLYP1 sets of numbers, with each set beginning on a new line, and each set containing NSLXP1 numbers. In the data shown above, the "repeat" feature of Fortran list-directed input has been used. An integer, followed by an asterisk (*), followed by a data value, represents as many copies of that data value as the value of the integer. Thus, 6*1.00E-02 represents 6 values of 1.00E-02. As an example, if one wanted to make the third number in line 8 have the value 1.23, line 8 above would be changed to:

```
2*1.00E-02  1.23  3*1.00E-02  1*3.16E-02  3*1.00E-01
```

This represents 2 values of 1.00E-02, followed by 1.23, followed by 3 values of 1.00E-02, followed by 3.16E-02, followed by 3 values of 1.00E-01; a total of 10 numbers. The 1* in front of 3.16E-02 can be omitted; the data file was prepared by the program LTPREP, which always puts a repeat factor in front of each value.

Remember that NSLXP1 and NSLYP1 are the total numbers of nodes in the X and Y directions. They were computed by subroutine FLOREAD from the values of NSLXM1 and NSLYM1 which

were read in from the xxxxxxxx.WAD file.

The Y-component of the saturated hydraulic conductivity is read next. NSLYP1 rows are read, with row J containing values KSATYY(1,J) through KSATYY(NSLXP1,J):

```
Y-COMPONENT OF HYDRAULIC CONDUCTIVITY (M/DAY)
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*1.00E-01   1*3.16E-01   6*1.00E+00
   3*3.16E-02   1*5.62E-02   2*1.00E-01   1*3.16E-01   3*1.00E+00
   6*1.00E-02   1*1.00E-01   3*1.00E+00
   6*1.00E-02   1*1.00E-01   3*1.00E+00
   6*1.00E-02   1*5.62E-02   3*3.16E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
   6*1.00E-02   1*3.16E-02   3*1.00E-01
```

FLOREAD reads the number of injection and extraction well positions NINJW, NEXTW. A minimum of one of each kind is required, but their strengths may be zero.

```
NO. OF INJECTION AND EXTRACTION WELLS
          1             1
```

Next, FLOREAD reads the injection and extraction well strengths. The minimum is one well of each kind, which may have zero strength (KG WATER/DAY).

FLOREAD reads NINJW lines containing XI, YJ, and QWELIN(I,J):

```
 XI   YJ  STRENGTH (INJECTION WELLS) (KG WATER/DAY)
0.1  0.1    7.5
```

There must be NINJW data lines, following the ONE text line. The positions of wells are specified by the X and Y coordinates, NOT by indices. The values of XI and YJ must be the coordinate values for XNODE(I) and YNODE(J). The input strength data has units of (kg water/day). The program divides the data by the appropriate volumes to get units of (kg water/m^3-day) for QWELIN.

FLOREAD reads NEXTW lines containing XI, YJ, and QWELOT(I,J):

```
 XI   YJ  STRENGTH (EXTRACTION WELLS) (KG WATER/DAY)
1.3  1.5    4.0
```

There must be NEXTW data lines, following the ONE text line. See the notes above under INJECTION WELLS concerning the coordinates and the strength data.

If there were no extraction wells, one well of strength 0.0 would be placed at position (0.1,0.1), or any other interior position. Wells must be at interior nodes, and cannot be on the boundaries, which is why there could not be a well at position (0.0,0.0).

FLOREAD reads the inlet and outlet boundary hydraulic heads HIN, HOUT (Meters):

```
INLET AND EXIT PORTS PRESSURE HEADS (M)
 1.0       0.0
```

In the example data file, NFLAG(2) is 0, which causes the program to compute an initial guess for the water pressure. If NFLAG(2) = 1, FLOREAD will read an initial guess for the water pressure, following the HIN and HOUT data shown above. For example, to make the program begin with an initial guess of 0.0 for the pressure, set NFLAG(2) to 1 and add the following to the xxxxxxxx.WAD file:

```
INITIAL WATER PRESSURE (M WATER)
 10*0.0
 10*0.0
 10*0.0
 10*0.0
 10*0.0
 10*0.0
 10*0.0
 10*0.0
 10*0.0
 10*0.0
```

Like the other arrays that are read in by the program, this data must consist of NSLYP1 rows with NSLXP1 values per row.

## IV.3. Example input chemistry data file: LT3EXAM1.CHD

This section shows an example of the chemical parameter data file. The file must have a name with extension ".CHD"; it is read in by subroutine CHMREAD in module RWCSIC.FOR. Both the abbreviated version of the file and the complete version are listed below. The listings are followed by a detailed discussion of the data.

The first listing is the abbreviated file LT3EXAM1.CHG, which is one of the files to be read by LTPREP:

```
     DATE IS 9/21/1990 - EXAMPLE DATA FOR MODEL LT3VSI V1.2 [4 chem]
     CHEM DATA FILE; ONE INJECTION WELL; ONE EXTRACTION WELL
NPRT  TMAX   DTO  (DAYS)  (RUN CONTROL DATA)
  2   4.0    0.001
DECISION FLAGS
1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0
INLET AND EXIT PORT TANK LENGTHS (M)
.1        .1
 SAND    CLAY   ORGANICS  (SOIL PARTICLE DENSITIES KG/M^3)
2660.0  2650.0   1300.0
TORTUOSITY (DIMLESS)
#VALUES
(0.1,1.5) 0.500 [low]
(0.1,0.1) 0.670 [medium]
(1.3,1.5) 0.670 [medium]
(1.3,0.1) 0.900 [high]
(0,0) 0
POROSITY (EPS) (M^3 VOIDS/M^3 POROUS MEDIUM)
#VALUES
(0.1,1.5) 0.465 [low]
(0.1,0.1) 0.365 [medium]
(1.3,1.5) 0.365 [medium]
(1.3,0.1) 0.285 [high]
(0,0) 0
PERCENT SAND (DIMLESS)
#VALUES
(0.1,1.5) 0.800 [low]
(0.1,0.1) 0.950 [medium]
(1.3,1.5) 0.950 [medium]
(1.3,0.1) 1.000 [high]
(0,0) 0
PERCENT CLAY & SILT (DIMLESS)
#VALUES
(0.1,1.5) 0.190 [low]
(0.1,0.1) 0.050 [medium]
(1.3,1.5) 0.050 [medium]
(1.3,0.1) 0.000 [high]
(0,0) 0
PERCENT ORGANICS (DIMLESS)
#VALUES
(0.1,1.5) 0.010 [low]
(0.1,0.1) 0.000 [medium]
(1.3,1.5) 0.000 [medium]
(1.3,0.1) 0.000 [high]
(0,0) 0
```

"Sand" stands for the weakly sorbing fraction, which includes sand and large silt.

"Clay" stands for the strongly sorbing fraction, which includes clay and small silt.

See "sand" note above.

See "clay" note above.

```
X-COMPONENT OF DISPERSIVITY FOR NUTRIENT (M)
#VALUES
(0.1,1.5) 1.0e-2 [low]
(0.1,0.1) 3.5e-3 [medium]
(1.3,1.5) 3.5e-3 [medium]
(1.3,0.1) 1.2e-3 [high]
(0,0) 0
Y-COMPONENT OF DISPERSIVITY FOR NUTRIENT (M)
#VALUES
(0.1,1.5) 1.0e-2 [low]
(0.1,0.1) 3.5e-3 [medium]
(1.3,1.5) 3.5e-3 [medium]
(1.3,0.1) 1.2e-3 [high]
(0,0) 0
X-COMPONENT OF DISPERSIVITY FOR SUBSTRATE (M)
#VALUES
(0.1,1.5) 1.0e-2 [low]
(0.1,0.1) 3.5e-3 [medium]
(1.3,1.5) 3.5e-3 [medium]
(1.3,0.1) 1.2e-3 [high]
(0,0) 0
Y-COMPONENT OF DISPERSIVITY FOR SUBSTRATE (M)
#VALUES
(0.1,1.5) 1.0e-2 [low]
(0.1,0.1) 3.5e-3 [medium]
(1.3,1.5) 3.5e-3 [medium]
(1.3,0.1) 1.2e-3 [high]
(0,0) 0
X-COMPONENT OF DISPERSIVITY FOR OXYGEN (M)
#VALUES
(0.1,1.5) 1.0e-2 [low]
(0.1,0.1) 3.5e-3 [medium]
(1.3,1.5) 3.5e-3 [medium]
(1.3,0.1) 1.2e-3 [high]
(0,0) 0
Y-COMPONENT OF DISPERSIVITY FOR OXYGEN (M)
#VALUES
(0.1,1.5) 1.0e-2 [low]
(0.1,0.1) 3.5e-3 [medium]
(1.3,1.5) 3.5e-3 [medium]
(1.3,0.1) 1.2e-3 [high]
(0,0) 0
X-COMPONENT OF DISPERSIVITY FOR NITRATE (M)
#VALUES
(0.1,1.5) 1.0e-2 [low]
(0.1,0.1) 3.5e-3 [medium]
(1.3,1.5) 3.5e-3 [medium]
(1.3,0.1) 1.2e-3 [high]
(0,0) 0
Y-COMPONENT OF DISPERSIVITY FOR NITRATE (M)
#VALUES
(0.1,1.5) 1.0e-2 [low]
(0.1,0.1) 3.5e-3 [medium]
(1.3,1.5) 3.5e-3 [medium]
(1.3,0.1) 1.2e-3 [high]
(0,0) 0
NUTRIENT   SUBSTRATE  OXYGEN    NITRATE
0.000143   0.000143   0.000143  0.000143  DLO (DIFFUSION) (M^2/DAY)
0.0        0.0        0.0       0.0       KSAND   (M^3/KG)
0.0        0.0        0.0       0.0       KCLAY   (M^3/KG)
0.0        0.0        0.0       0.0       KORG    (M^3/KG)
NUTRIENT   SUBSTRATE  OXYGEN    NITRATE
3.0E-3     1.5E-2     5.0E-3    5.0E-3    INLET TANK  (KG CHEM/M^3)
0.0        0.0        0.0       0.0       EXIT TANK   (KG CHEM/M^3)
3.0E-3     1.5E-2     5.0E-3    5.0E-3    STREAM      (KG CHEM/M^3)
IRREVERSIBLE LOSS IN FREE PHASE FOR NUTRIENT  (1/DAY)
```

```
#VALUES
(0.1,1.5) 1.0e-2 [low]
(0.1,0.1) 1.0e-4 [medium]
(1.3,1.5) 1.0e-4 [medium]
(1.3,0.1) 0.000  [high]
(0,0) 0
IRREVERSIBLE LOSS IN FREE PHASE FOR SUBSTRATE  (1/DAY)
#VALUES
(0.1,1.5) 1.0e-2 [low]
(0.1,0.1) 1.0e-4 [medium]
(1.3,1.5) 1.0e-4 [medium]
(1.3,0.1) 0.000  [high]
(0,0) 0
IRREVERSIBLE LOSS IN FREE PHASE FOR OXYGEN  (1/DAY)
#VALUES
(0.1,1.5) 1.0e-2 [low]
(0.1,0.1) 1.0e-4 [medium]
(1.3,1.5) 1.0e-4 [medium]
(1.3,0.1) 0.000  [high]
(0,0) 0
IRREVERSIBLE LOSS IN FREE PHASE FOR NITRATE  (1/DAY)
#VALUES
(0.1,1.5) 1.0e-2 [low]
(0.1,0.1) 1.0e-4 [medium]
(1.3,1.5) 1.0e-4 [medium]
(1.3,0.1) 0.000  [high]
(0,0) 0
IRREVERSIBLE LOSS IN SORBED PHASE FOR NUTRIENT  (1/DAY)
#VALUES
(0.1,1.5) 0.000 [low]
(0.1,0.1) 0.000 [medium]
(1.3,1.5) 0.000 [medium]
(1.3,0.1) 0.000 [high]
(0,0) 0
IRREVERSIBLE LOSS IN SORBED PHASE FOR SUBSTRATE  (1/DAY)
#VALUES
(0.1,1.5) 0.000 [low]
(0.1,0.1) 0.000 [medium]
(1.3,1.5) 0.000 [medium]
(1.3,0.1) 0.000 [high]
(0,0) 0
IRREVERSIBLE LOSS IN SORBED PHASE FOR OXYGEN  (1/DAY)
#VALUES
(0.1,1.5) 0.000 [low]
(0.1,0.1) 0.000 [medium]
(1.3,1.5) 0.000 [medium]
(1.3,0.1) 0.000 [high]
(0,0) 0
IRREVERSIBLE LOSS IN SORBED PHASE FOR NITRATE  (1/DAY)
#VALUES
(0.1,1.5) 0.000 [low]
(0.1,0.1) 0.000 [medium]
(1.3,1.5) 0.000 [medium]
(1.3,0.1) 0.000 [high]
(0,0) 0
INITIAL DISTRIBUTION OF NUTRIENT  (KG CHEM/M^3)
#VALUES
(0.1,1.5) 3.0E-3 [low]
(0.1,0.1) 3.0E-3 [medium]
(1.3,1.5) 3.0E-3 [medium]
(1.3,0.1) 3.0E-3 [high]
(0,0) 0
INITIAL DISTRIBUTION OF SUBSTRATE  (KG CHEM/M^3)
#VALUES
(0.1,1.5) 5.0E-3 [low]
(0.1,0.1) 5.0E-3 [medium]
```

```
(1.3,1.5) 5.0E-3 [medium]
(1.3,0.1) 5.0E-3 [high]
(0,0) 0
INITIAL DISTRIBUTION OF OXYGEN  (KG CHEM/M^3)
#VALUES
(0.1,1.5) 5.0E-3 [low]
(0.1,0.1) 5.0E-3 [medium]
(1.3,1.5) 5.0E-3 [medium]
(1.3,0.1) 5.0E-3 [high]
(0,0) 0
INITIAL DISTRIBUTION OF NITRATE  (KG CHEM/M^3)
#VALUES
(0.1,1.5) 5.0E-3 [low]
(0.1,0.1) 5.0E-3 [medium]
(1.3,1.5) 5.0E-3 [medium]
(1.3,0.1) 5.0E-3 [high]
(0,0) 0
POPULATION 1 INITIAL VALUE (KG CELLS/KG SOIL)
#VALUES
(0.1,1.5) 1.0E-7 [low]
(0.1,0.1) 1.0E-7 [medium]
(1.3,1.5) 1.0E-7 [medium]
(1.3,0.1) 1.0E-7 [high]
(0,0) 0
POPULATION 2 INITIAL VALUE (KG CELLS/KG SOIL)
#VALUES
(0.1,1.5) 1.0E-7 [low]
(0.1,0.1) 1.0E-7 [medium]
(1.3,1.5) 1.0E-7 [medium]
(1.3,0.1) 1.0E-7 [high]
(0,0) 0
CHEMISTRY USAGE PARAMETERS
 1.8E-2     KSO1   = 1/2 MAX SAT CONST POP 1 USING SUB (KG/M^3)
 1.8E-2     KSO2   = 1/2 MAX SAT CONST POP 2 USING SUB (KG/M^3)
 3.0E-5     KO1    = 1/2 MAX SAT CONST POP 1 USING OXY (KG/M^3)
 3.0E-5     KO2    = 1/2 MAX SAT CONST POP 2 USING OXY (KG/M^3)
 3.0E-4     KONU1  = 1/2 MAX SAT CONST POP 1 USING NUT (KG/M^3)
 3.0E-4     KONU2  = 1/2 MAX SAT CONST POP 2 USING NUT (KG/M^3)
 1.8E-2     KSNI1  = 1/2 MAX NIT BASE SATCON POP 1 USING SUB (KG/M^3)
 2.0E-5     KNI1   = 1/2 MAX NIT BASE SATCON POP 1 USING NIT (KG/M^3)
 3.0E-4     KNINU1 = 1/2 MAX NIT BASE SATCON POP 1 USING NUT (KG/M^3)
 1.1E-4     KONI1  = 1/2 MAX NIT BASE INH CON POP 1 USING NIT (KG/M^3)
 3.0E-5     KSOM1  = 1/2 SAT COEF MAINT POP 1 (KG/M^3)
 3.0E-5     KSOM2  = 1/2 SAT COEF MAINT POP 2 (KG/M^3)
 0.4        YSO1   = YIELD COEF POP 1 USING OXY (KG CELLS/KG SUB)
 0.4        YSO2   = YIELD COEF POP 2 USING OXY (KG CELLS/KG SUB)
 0.17       YSNI1  = YIELD COEF POP 1 USING NIT (KG CELLS/KG SUB)
 0.004      ALFO1  = OXY MAINT COEF POP 1 AER COND (KG OXY/KG CELLS)
 0.004      ALFO2  = OXY MAINT COEF POP 2 AER COND (KG OXY/KG CELLS)
 0.002      ALFNI1 = NIT MAINT COEF POP 1 ANAER COND (KG NIT/KG CELLS)
 0.375      ETANI1 = NIT USE COEF POP 1 ANAER COND (KG NIT/KG SUB)
 1.0        GAMMO1 = OXY USE COEF POP 1 AER COND (KG OXY/KG SUB)
 1.0        GAMMO2 = OXY USE COEF POP 2 AER COND (KG OXY/KG SUB)
 0.05       PSIO1  = NUT USE COEF POP 1 AER COND (KG NUT/KG SUB)
 0.05       PSIO2  = NUT USE COEF POP 2 AER COND (KG NUT/KG SUB)
 0.021      THENI1 = NUT USE COEF POP 1 NIT COND (KG NUT/KG SUB)
 4.0        MUO1   = MAX SPEC GROWTH RATE HETERO POP 1 (1/DAY)
 4.0        MUO2   = MAX SPEC GROWTH RATE HETERO POP 2 (1/DAY)
 2.5        MUNI1  = MAX SPEC NIT BASE GROWTH RATE HETERO POP 1 (1/DAY)
 1770.0     RHOBD  = AVE BULK DENSITY OF POROUS MEDIUM (KG/M^3)
CONCENTRATION OF INJECTION WELL (KG CHEM/KG SOLUTION)
 XI   YJ  NUTRIENT  SUBSTRATE  OXYGEN  NITRATE
0.1  0.1  0.0       0.0        0.0     0.0
NUMBER OF BURIED SOURCES
  1
CONCENTRATION OF BURIED SOURCE (KG CHEM/DAY)
```

```
 XI   YJ  NUTRIENT  SUBSTRATE  OXYGEN  NITRATE
0.1  0.1  0.0       0.0        0.0     0.0
END OF CHEM DATA FILE
```

LTPREP reads the geometry file LT3EXAM.GEO and LT3EXAM1.CHG listed above, and writes the complete data file LT3EXAM1.CHD, which is listed below. See Section IV.1 for more information about geometry files and LTPREP. In this case, LTPREP saves a considerable amount of work in preparing the data file. Below is the listing of LT3EXAM1.CHD, which is read by the chemistry phase of LT3VSI:

```
     DATE IS 9/21/1990 - EXAMPLE DATA FOR MODEL LT3VSI V1.2 [4 chem]
     CHEM DATA FILE; ONE INJECTION WELL; ONE EXTRACTION WELL
NPRT  TMAX   DTO  (DAYS)  (RUN CONTROL DATA)
  2   4.0    0.001
DECISION FLAGS
1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0
INLET AND EXIT PORT TANK LENGTHS (M)
.1        .1
 SAND    CLAY   ORGANICS  (SOIL PARTICLE DENSITIES KG/M^3)
2660.0  2650.0   1300.0
TORTUOSITY (DIMLESS)
   3*6.70E-01   1*7.77E-01   6*9.00E-01
   3*6.70E-01   1*7.77E-01   6*9.00E-01
   3*6.70E-01   1*7.77E-01   6*9.00E-01
   3*5.79E-01   1*6.23E-01   2*6.71E-01   1*7.77E-01   3*9.00E-01
   6*5.00E-01   1*6.71E-01   3*9.00E-01
   6*5.00E-01   1*6.71E-01   3*9.00E-01
   6*5.00E-01   1*6.23E-01   3*7.77E-01
   6*5.00E-01   1*5.79E-01   3*6.70E-01
   6*5.00E-01   1*5.79E-01   3*6.70E-01
   6*5.00E-01   1*5.79E-01   3*6.70E-01
POROSITY (EPS) (M^3 VOIDS/M^3 POROUS MEDIUM)
   3*3.65E-01   1*3.23E-01   6*2.85E-01
   3*3.65E-01   1*3.23E-01   6*2.85E-01
   3*3.65E-01   1*3.23E-01   6*2.85E-01
   3*4.12E-01   1*3.87E-01   2*3.64E-01   1*3.22E-01   3*2.85E-01
   6*4.65E-01   1*3.64E-01   3*2.85E-01
   6*4.65E-01   1*3.64E-01   3*2.85E-01
   6*4.65E-01   1*3.87E-01   3*3.23E-01
   6*4.65E-01   1*4.12E-01   3*3.65E-01
   6*4.65E-01   1*4.12E-01   3*3.65E-01
   6*4.65E-01   1*4.12E-01   3*3.65E-01
PERCENT SAND (DIMLESS)
   3*9.50E-01   1*9.75E-01   6*1.00E+00
   3*9.50E-01   1*9.75E-01   6*1.00E+00
   3*9.50E-01   1*9.75E-01   6*1.00E+00
   3*8.72E-01   1*8.83E-01   2*8.94E-01   1*9.46E-01   3*1.00E+00
   6*8.00E-01   1*8.94E-01   3*1.00E+00
   6*8.00E-01   1*8.94E-01   3*1.00E+00
   6*8.00E-01   1*8.83E-01   3*9.75E-01
   6*8.00E-01   1*8.72E-01   3*9.50E-01
   6*8.00E-01   1*8.72E-01   3*9.50E-01
   6*8.00E-01   1*8.72E-01   3*9.50E-01
PERCENT CLAY & SILT (DIMLESS)
   3*5.00E-02   7*0.00E+00
   3*5.00E-02   7*0.00E+00
   3*5.00E-02   7*0.00E+00
```

```
   3*9.75E-02   7*0.00E+00
   6*1.90E-01   4*0.00E+00
   6*1.90E-01   4*0.00E+00
   6*1.90E-01   4*0.00E+00
   6*1.90E-01   1*9.75E-02   3*5.00E-02
   6*1.90E-01   1*9.75E-02   3*5.00E-02
   6*1.90E-01   1*9.75E-02   3*5.00E-02
PERCENT ORGANICS (DIMLESS)
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   4*0.00E+00
X-COMPONENT OF DISPERSIVITY FOR NUTRIENT (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*5.92E-03   1*4.53E-03   2*3.46E-03   1*2.04E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*4.53E-03   3*2.05E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
Y-COMPONENT OF DISPERSIVITY FOR NUTRIENT (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*5.92E-03   1*4.53E-03   2*3.46E-03   1*2.04E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*4.53E-03   3*2.05E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
X-COMPONENT OF DISPERSIVITY FOR SUBSTRATE (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*5.92E-03   1*4.53E-03   2*3.46E-03   1*2.04E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*4.53E-03   3*2.05E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
Y-COMPONENT OF DISPERSIVITY FOR SUBSTRATE (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*5.92E-03   1*4.53E-03   2*3.46E-03   1*2.04E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*4.53E-03   3*2.05E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
X-COMPONENT OF DISPERSIVITY FOR OXYGEN (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
```

```
   3*5.92E-03   1*4.53E-03   2*3.46E-03   1*2.04E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*4.53E-03   3*2.05E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
Y-COMPONENT OF DISPERSIVITY FOR OXYGEN (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*5.92E-03   1*4.53E-03   2*3.46E-03   1*2.04E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*4.53E-03   3*2.05E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
X-COMPONENT OF DISPERSIVITY FOR NITRATE (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*5.92E-03   1*4.53E-03   2*3.46E-03   1*2.04E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*4.53E-03   3*2.05E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
Y-COMPONENT OF DISPERSIVITY FOR NITRATE (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*3.50E-03   1*2.05E-03   6*1.20E-03
   3*5.92E-03   1*4.53E-03   2*3.46E-03   1*2.04E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*3.46E-03   3*1.20E-03
   6*1.00E-02   1*4.53E-03   3*2.05E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
   6*1.00E-02   1*5.92E-03   3*3.50E-03
NUTRIENT   SUBSTRATE  OXYGEN    NITRATE
0.000143   0.000143   0.000143  0.000143  DLO (DIFFUSION) (M^2/DAY)
0.0        0.0        0.0       0.0       KSAND   (M^3/KG)
0.0        0.0        0.0       0.0       KCLAY   (M^3/KG)
0.0        0.0        0.0       0.0       KORG    (M^3/KG)
NUTRIENT   SUBSTRATE  OXYGEN    NITRATE
3.0E-3     1.5E-2     5.0E-3    5.0E-3    INLET TANK  (KG CHEM/M^3)
0.0        0.0        0.0       0.0       EXIT TANK   (KG CHEM/M^3)
3.0E-3     1.5E-2     5.0E-3    5.0E-3    STREAM      (KG CHEM/M^3)
IRREVERSIBLE LOSS IN FREE PHASE FOR NUTRIENT  (1/DAY)
   3*1.00E-04   7*0.00E+00
   3*1.00E-04   7*0.00E+00
   3*1.00E-04   7*0.00E+00
   3*1.00E-03   7*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   1*1.00E-03   3*1.00E-04
   6*1.00E-02   1*1.00E-03   3*1.00E-04
   6*1.00E-02   1*1.00E-03   3*1.00E-04
IRREVERSIBLE LOSS IN FREE PHASE FOR SUBSTRATE  (1/DAY)
   3*1.00E-04   7*0.00E+00
   3*1.00E-04   7*0.00E+00
   3*1.00E-04   7*0.00E+00
   3*1.00E-03   7*0.00E+00
   6*1.00E-02   4*0.00E+00
```

```
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   1*1.00E-03   3*1.00E-04
   6*1.00E-02   1*1.00E-03   3*1.00E-04
   6*1.00E-02   1*1.00E-03   3*1.00E-04
IRREVERSIBLE LOSS IN FREE PHASE FOR OXYGEN  (1/DAY)
   3*1.00E-04   7*0.00E+00
   3*1.00E-04   7*0.00E+00
   3*1.00E-04   7*0.00E+00
   3*1.00E-03   7*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   1*1.00E-03   3*1.00E-04
   6*1.00E-02   1*1.00E-03   3*1.00E-04
   6*1.00E-02   1*1.00E-03   3*1.00E-04
IRREVERSIBLE LOSS IN FREE PHASE FOR NITRATE  (1/DAY)
   3*1.00E-04   7*0.00E+00
   3*1.00E-04   7*0.00E+00
   3*1.00E-04   7*0.00E+00
   3*1.00E-03   7*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   4*0.00E+00
   6*1.00E-02   1*1.00E-03   3*1.00E-04
   6*1.00E-02   1*1.00E-03   3*1.00E-04
   6*1.00E-02   1*1.00E-03   3*1.00E-04
IRREVERSIBLE LOSS IN SORBED PHASE FOR NUTRIENT  (1/DAY)
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
IRREVERSIBLE LOSS IN SORBED PHASE FOR SUBSTRATE  (1/DAY)
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
IRREVERSIBLE LOSS IN SORBED PHASE FOR OXYGEN  (1/DAY)
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
IRREVERSIBLE LOSS IN SORBED PHASE FOR NITRATE  (1/DAY)
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
```

```
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
  10*0.00E+00
INITIAL DISTRIBUTION OF NUTRIENT  (KG CHEM/M^3)
  10*3.00E-03
  10*3.00E-03
  10*3.00E-03
  10*3.00E-03
  10*3.00E-03
  10*3.00E-03
  10*3.00E-03
  10*3.00E-03
  10*3.00E-03
  10*3.00E-03
INITIAL DISTRIBUTION OF SUBSTRATE  (KG CHEM/M^3)
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
INITIAL DISTRIBUTION OF OXYGEN  (KG CHEM/M^3)
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
INITIAL DISTRIBUTION OF NITRATE  (KG CHEM/M^3)
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
  10*5.00E-03
POPULATION 1 INITIAL VALUE (KG CELLS/KG SOIL)
  10*1.00E-07
  10*1.00E-07
  10*1.00E-07
  10*1.00E-07
  10*1.00E-07
  10*1.00E-07
  10*1.00E-07
  10*1.00E-07
  10*1.00E-07
  10*1.00E-07
POPULATION 2 INITIAL VALUE (KG CELLS/KG SOIL)
  10*1.00E-07
  10*1.00E-07
  10*1.00E-07
  10*1.00E-07
  10*1.00E-07
```

```
  10*1.00E-07
  10*1.00E-07
  10*1.00E-07
  10*1.00E-07
  10*1.00E-07
CHEMISTRY USAGE PARAMETERS
 1.8E-2      KSO1    = 1/2 MAX SAT CONST POP 1 USING SUB (KG/M^3)
 1.8E-2      KSO2    = 1/2 MAX SAT CONST POP 2 USING SUB (KG/M^3)
 3.0E-5      KO1     = 1/2 MAX SAT CONST POP 1 USING OXY (KG/M^3)
 3.0E-5      KO2     = 1/2 MAX SAT CONST POP 2 USING OXY (KG/M^3)
 3.0E-4      KONU1   = 1/2 MAX SAT CONST POP 1 USING NUT (KG/M^3)
 3.0E-4      KONU2   = 1/2 MAX SAT CONST POP 2 USING NUT (KG/M^3)
 1.8E-2      KSNI1   = 1/2 MAX NIT BASE SATCON POP 1 USING SUB (KG/M^3)
 2.0E-5      KNI1    = 1/2 MAX NIT BASE SATCON POP 1 USING NIT (KG/M^3)
 3.0E-4      KNINU1  = 1/2 MAX NIT BASE SATCON POP 1 USING NUT (KG/M^3)
 1.1E-4      KONI1   = 1/2 MAX NIT BASE INH CON POP 1 USING NIT (KG/M^3)
 3.0E-5      KSOM1   = 1/2 SAT COEF MAINT POP 1 (KG/M^3)
 3.0E-5      KSOM2   = 1/2 SAT COEF MAINT POP 2 (KG/M^3)
 0.4         YSO1    = YIELD COEF POP 1 USING OXY (KG CELLS/KG SUB)
 0.4         YSO2    = YIELD COEF POP 2 USING OXY (KG CELLS/KG SUB)
 0.17        YSNI1   = YIELD COEF POP 1 USING NIT (KG CELLS/KG SUB)
 0.004       ALFO1   = OXY MAINT COEF POP 1 AER COND (KG OXY/KG CELLS)
 0.004       ALFO2   = OXY MAINT COEF POP 2 AER COND (KG OXY/KG CELLS)
 0.002       ALFNI1  = NIT MAINT COEF POP 1 ANAER COND (KG NIT/KG CELLS)
 0.375       ETANI1  = NIT USE COEF POP 1 ANAER COND (KG NIT/KG SUB)
 1.0         GAMMO1  = OXY USE COEF POP 1 AER COND (KG OXY/KG SUB)
 1.0         GAMMO2  = OXY USE COEF POP 2 AER COND (KG OXY/KG SUB)
 0.05        PSIO1   = NUT USE COEF POP 1 AER COND (KG NUT/KG SUB)
 0.05        PSIO2   = NUT USE COEF POP 2 AER COND (KG NUT/KG SUB)
 0.021       THENI1  = NUT USE COEF POP 1 NIT COND (KG NUT/KG SUB)
 4.0         MUO1    = MAX SPEC GROWTH RATE HETERO POP 1 (1/DAY)
 4.0         MUO2    = MAX SPEC GROWTH RATE HETERO POP 2 (1/DAY)
 2.5         MUNI1   = MAX SPEC NIT BASE GROWTH RATE HETERO POP 1 (1/DAY)
 1770.0      RHOBD   = AVE BULK DENSITY OF POROUS MEDIUM (KG/M^3)
CONCENTRATION OF INJECTION WELL (KG CHEM/KG SOLUTION)
 XI   YJ  NUTRIENT  SUBSTRATE  OXYGEN  NITRATE
0.1  0.1  0.0       0.0        0.0     0.0
NUMBER OF BURIED SOURCES
  1
CONCENTRATION OF BURIED SOURCE (KG CHEM/DAY)
 XI   YJ  NUTRIENT  SUBSTRATE  OXYGEN  NITRATE
0.1  0.1  0.0       0.0        0.0     0.0
END OF CHEM DATA FILE
```

In the rest of this section, the data read in by subroutine CHMREAD and the computations that it performs are described. The description is similar to that used in describing the hydrology data file (LT3EXAM.WAD), except that most of the values will be omitted.

CHMREAD reads from data file xxxxxxxx.CHD. It first reads a two-line heading, using variables HEAD1 and HEAD2, which will be printed out on files xxxxxxxx.CHO and xxxxxxxx.CNC:

```
DATE IS 9/21/1990 - EXAMPLE DATA FOR MODEL LT3VSI V1.2 [4 chem]
CHEM DATA FILE; ONE INJECTION WELL; ONE EXTRACTION WELL
```

Next, CHMREAD reads the run control data NPRT, TMAX, DT0:

```
NPRT  TMAX   DT0  (DAYS)  (RUN CONTROL DATA)
  2   4.0    0.001
```

NPRT is the number of times to print the distribution of chemical during the run; TMAX is the amount of simulated time that the program should run, computing chemical concentrations; and DT0 is the desired time increment. All times are in days.

DT0 is the "delta-T"; the increment for TIME in the chemical processing. As discussed in Volume 4 (Section 5.6), delta-T must satisfy a certain stability criterion in order for the method to be stable. The program computes this criterion (DT1) for each of the four chemicals, and if the delta-T chosen by the user is greater than one-half of the minimum DT1, the program sets DT0 to one-half of the minimum DT1, thus ensuring that the process will be stable. The values of the four DT1's are printed out on the file xxxxxxxx.CHO, as "Computed DTMAX."

CHMREAD reads the 20 control flags NFLAG(1) through NFLAG(20):

```
DECISION FLAGS
0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0
```

The control flags are read by both the water and chemistry programs; their values are usually different in the two phases. Most flags are used in either the water or the chemical phase. Only NFLAG(8) is used in both phases. In the chemistry phase NFLAG(5) and NFLAG(6) are used to control "continuation" runs; see the discussion near the end of this section, and also Section V.7. NFLAG(4) = 1 causes the program to read a schedule file

xxxxxxxx.SCH, which specifies at which times the concentrations of the chemicals in injection wells are changed. See Section IV.3.1. NFLAG(17) = 1 above, which causes the program to compute and print the cumulative masses XMASS, etc. See the discussion of flags in Section III.2.

If NFLAG(9) = 0, as it is in this example, the chemistry phase uses the value of ZTHRSH that was read in by FLOREAD in the water phase. If it is desired to use a different value for ZTHRSH in the chemistry phase, set NFLAG(9) to 1, and insert two lines at this point in the xxxxxxxx.CHD file, of the following form:

```
ZTHRSH
1.0E-9
```

XLYIN and XLYOUT, the lengths (in meters) of the inlet and outlet end tanks are read:

```
INLET AND EXIT PORT TANK LENGTHS (M)
.1        .1
```

These are the lengths of the well-stirred inlet and outlet mixing tanks.

CHMREAD reads the average particle densities of the soil components RHOSND, RHOCLA, RHOORG:

```
 SAND     CLAY    ORGANICS  (SOIL PARTICLE DENSITIES KG/M^3)
2660.0  2650.0    1300.0
```

The tortuosity factor is read in. The data consists of NSLYP1 rows, with row J containing values for TORT(1,J) through TORT(NSLXP1,J):

```
TORTUOSITY (DIMLESS)
   3*6.70E-01   1*7.77E-01   6*9.00E-01
    ...    ...    ...    ...    ...
```

For this array and the following two-dimensional arrays, only

the first line of data will be shown. The complete arrays are shown in the listing of LT3EXAM1.CHD above.

For each two-dimensional array to be read in, there must be NSLYP1 sets of numbers, with each set beginning on a new line, and each set containing NSLXP1 numbers. See Section IV.2 for a discussion of Fortran's "repeat" feature in list-directed input.

Remember that NSLXP1 and NSLYP1 are the total numbers of nodes in the X and Y directions. They were computed by subroutine FLOREAD from the values of NSLXM1 and NSLYM1 which were read in from the xxxxxxxx.WAD file.

The porosity EPS(I,J) is read next:

```
POROSITY (EPS) (M^3 VOIDS/M^3 POROUS MEDIUM)
   3*3.65E-01   1*3.23E-01   6*2.85E-01
    ...    ...    ...    ...    ...
```

CHMREAD reads PCTSAN(I,J), which means "percent sand"; however, the correct terminology is "mass fraction of weakly sorbing particles," which includes sand and large silt.

```
PERCENT SAND (DIMLESS)
   3*9.50E-01   1*9.75E-01   6*1.00E+00
    ...    ...    ...    ...    ...
```

PCTCLA(I,J) is read next. The correct terminology is "mass fraction of strongly sorbing particles," which includes clay and small silt.

```
PERCENT CLAY & SILT (DIMLESS)
   3*5.00E-02   7*0.00E+00
    ...    ...    ...    ...    ...
```

PCTORG(I,J) is read in. The correct terminology is "mass fraction of organic matter."

```
PERCENT ORGANICS (DIMLESS)
  10*0.00E+00
    ...    ...    ...    ...    ...
```

CHMREAD reads the X-components and Y-components of dispersivity for the four chemicals. DISPLX(I,J,K) is the X-component of dispersivity for chemical K, K=1 to 4; DISPLY(I,J,K) is the Y-component of dispersivity for chemical K.

```
X-COMPONENT OF DISPERSIVITY FOR NUTRIENT (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
    ...     ...    ...     ...     ...
Y-COMPONENT OF DISPERSIVITY FOR NUTRIENT (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
    ...     ...    ...     ...     ...
```

```
X-COMPONENT OF DISPERSIVITY FOR SUBSTRATE (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
    ...     ...    ...     ...     ...
Y-COMPONENT OF DISPERSIVITY FOR SUBSTRATE (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
    ...     ...    ...     ...     ...
```

```
X-COMPONENT OF DISPERSIVITY FOR OXYGEN (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
    ...     ...    ...     ...     ...
Y-COMPONENT OF DISPERSIVITY FOR OXYGEN (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
    ...     ...    ...     ...     ...
```

```
X-COMPONENT OF DISPERSIVITY FOR NITRATE (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
    ...     ...    ...     ...     ...
Y-COMPONENT OF DISPERSIVITY FOR NITRATE (M)
   3*3.50E-03   1*2.05E-03   6*1.20E-03
    ...     ...    ...     ...     ...
```

CHMREAD next reads the chemical parameters; each parameter has four values, one for each of the four chemicals:

```
NUTRIENT   SUBSTRATE  OXYGEN    NITRATE
0.000143   0.000143   0.000143  0.000143  DL0 (DIFFUSION) (M^2/DAY)
0.0        0.0        0.0       0.0       KSAND   (M^3/KG)
0.0        0.0        0.0       0.0       KCLAY   (M^3/KG)
0.0        0.0        0.0       0.0       KORG    (M^3/KG)
```

DL0(K) is the molecular diffusion coefficient for chemical K; KSAND(K), KCLAY(K), and KORG(K) are the linear equilibrium distribution constants. The text at the right side of the second through fifth lines above is ignored by the program, because it is excess data.

CIN(K), COUT(K), and C0(K) are read next, four values for each:

```
NUTRIENT   SUBSTRATE  OXYGEN    NITRATE
3.0E-3     1.5E-2     5.0E-3    5.0E-3    INLET TANK   (KG CHEM/M^3)
0.0        0.0        0.0       0.0       EXIT TANK    (KG CHEM/M^3)
3.0E-3     1.5E-2     5.0E-3    5.0E-3    STREAM       (KG CHEM/M^3)
```

CIN(K) and COUT(K) are the initial chemical concentrations in the inlet and outlet end tanks, respectively. C0(K) is the constant concentration of chemical in the feed stream entering the inlet tank.

CHMREAD next reads the first order loss rate constants XLAMIR(I,J,K) and XSLMIR(I,J,K).

XLAMIR(I,J,K) is the first order free phase loss rate constant for irreversible loss processes for chemical K; XSLMIR(I,J,K) is the first order sorbed phase loss rate constant for irreversible processes for chemical K.

The free phase constants XLAMIR(I,J,K) are read first:

```
IRREVERSIBLE LOSS IN FREE PHASE FOR NUTRIENT  (1/DAY)
   3*1.00E-04   7*0.00E+00
    ...     ...    ...     ...     ...
IRREVERSIBLE LOSS IN FREE PHASE FOR SUBSTRATE  (1/DAY)
   3*1.00E-04   7*0.00E+00
    ...     ...    ...     ...     ...
IRREVERSIBLE LOSS IN FREE PHASE FOR OXYGEN  (1/DAY)
   3*1.00E-04   7*0.00E+00
    ...     ...    ...     ...     ...
IRREVERSIBLE LOSS IN FREE PHASE FOR NITRATE  (1/DAY)
   3*1.00E-04   7*0.00E+00
    ...     ...    ...     ...     ...
```

The sorbed phase constants XSLMIR(I,J,K) are read next:

```
IRREVERSIBLE LOSS IN SORBED PHASE FOR NUTRIENT  (1/DAY)
  10*0.00E+00
    ...     ...     ...     ...     ...
IRREVERSIBLE LOSS IN SORBED PHASE FOR SUBSTRATE  (1/DAY)
  10*0.00E+00
    ...     ...     ...     ...     ...
IRREVERSIBLE LOSS IN SORBED PHASE FOR OXYGEN  (1/DAY)
  10*0.00E+00
    ...     ...     ...     ...     ...
IRREVERSIBLE LOSS IN SORBED PHASE FOR NITRATE  (1/DAY)
  10*0.00E+00
    ...     ...     ...     ...     ...
```

At this point, CHMREAD sets T0 = 0.0 as the initial TIME and also sets the cumulative mass arrays XMASS(K), XMFONW(K), XMSOUR(K), XMASIN(K), and XMASOT(K) to zero. Then it reads the initial chemical distributions for the four chemicals, using array COLD(I,J,K):

```
INITIAL DISTRIBUTION OF NUTRIENT  (KG CHEM/M^3)
  10*3.00E-03
    ...     ...     ...     ...     ...
INITIAL DISTRIBUTION OF SUBSTRATE  (KG CHEM/M^3)
  10*5.00E-03
    ...     ...     ...     ...     ...
INITIAL DISTRIBUTION OF OXYGEN  (KG CHEM/M^3)
  10*5.00E-03
    ...     ...     ...     ...     ...
INITIAL DISTRIBUTION OF NITRATE  (KG CHEM/M^3)
  10*5.00E-03
    ...     ...     ...     ...     ...
```

The microbial populations POP1(I,J) and POP2(I,J) are read next:

```
POPULATION 1 INITIAL VALUE (KG CELLS/KG SOIL)
  10*1.00E-07
    ...     ...     ...     ...     ...
POPULATION 2 INITIAL VALUE (KG CELLS/KG SOIL)
  10*1.00E-07
    ...     ...     ...     ...     ...
```

If NFLAG(5) = 0, as it is in this example file, the computations and input data described following the line above that begins "At this point," remain as is. This run is an "initial run".

However, if NFLAG(5) = 1, this is a "continuation run". CHMREAD reads from the unformatted chemical file (extension .CUF)

that was written by a previous chemical run that had NFLAG(6) = 1. T0 for this run is set equal to the TMAX for the previous run and the TMAX for the previous run is added to the TMAX that was read in for this run. The values of CIN(K), COUT(K), COLD(I,J,K), POP1(I,J), and POP2(I,J) at the end of the previous run are used as the initial values for this run instead of the values that were read in from the file xxxxxxxx.CHD.

In other words, the values for these variables in the file are overwritten by the values from the previous run. It would not be necessary for the values to be present in the xxxxxxxx.CHD file, but it is simpler to have all .CHD files have the same structure. See Section V.7 for detailed information on "continuation" runs.

CHMREAD now reads the chemistry usage parameters, one per line. As usual, the text following the value on each line is ignored by the program. Its purpose is to help users who prepare or read the file to understand which variables are being defined. See Section III for a more readable description of the variables.

```
CHEMISTRY USAGE PARAMETERS
 1.8E-2     KSO1   = 1/2 MAX SAT CONST POP 1 USING SUB (KG/M^3)
 1.8E-2     KSO2   = 1/2 MAX SAT CONST POP 2 USING SUB (KG/M^3)
 3.0E-5     KO1    = 1/2 MAX SAT CONST POP 1 USING OXY (KG/M^3)
 3.0E-5     KO2    = 1/2 MAX SAT CONST POP 2 USING OXY (KG/M^3)
 3.0E-4     KONU1  = 1/2 MAX SAT CONST POP 1 USING NUT (KG/M^3)
 3.0E-4     KONU2  = 1/2 MAX SAT CONST POP 2 USING NUT (KG/M^3)
 1.8E-2     KSNI1  = 1/2 MAX NIT BASE SATCON POP 1 USING SUB (KG/M^3)
 2.0E-5     KNI1   = 1/2 MAX NIT BASE SATCON POP 1 USING NIT (KG/M^3)
 3.0E-4     KNINU1 = 1/2 MAX NIT BASE SATCON POP 1 USING NUT (KG/M^3)
 1.1E-4     KONI1  = 1/2 MAX NIT BASE INH CON POP 1 USING NIT (KG/M^3)
 3.0E-5     KSOM1  = 1/2 SAT COEF MAINT POP 1 (KG/M^3)
 3.0E-5     KSOM2  = 1/2 SAT COEF MAINT POP 2 (KG/M^3)
 0.4        YSO1   = YIELD COEF POP 1 USING OXY (KG CELLS/KG SUB)
 0.4        YSO2   = YIELD COEF POP 2 USING OXY (KG CELLS/KG SUB)
 0.17       YSNI1  = YIELD COEF POP 1 USING NIT (KG CELLS/KG SUB)
 0.004      ALFO1  = OXY MAINT COEF POP 1 AER COND (KG OXY/KG CELLS)
 0.004      ALFO2  = OXY MAINT COEF POP 2 AER COND (KG OXY/KG CELLS)
 0.002      ALFNI1 = NIT MAINT COEF POP 1 ANAER COND (KG NIT/KG CELLS)
 0.375      ETANI1 = NIT USE COEF POP 1 ANAER COND (KG NIT/KG SUB)
 1.0        GAMMO1 = OXY USE COEF POP 1 AER COND (KG OXY/KG SUB)
 1.0        GAMMO2 = OXY USE COEF POP 2 AER COND (KG OXY/KG SUB)
 0.05       PSIO1  = NUT USE COEF POP 1 AER COND (KG NUT/KG SUB)
 0.05       PSIO2  = NUT USE COEF POP 2 AER COND (KG NUT/KG SUB)
 0.021      THENI1 = NUT USE COEF POP 1 NIT COND (KG NUT/KG SUB)
 4.0        MUO1   = MAX SPEC GROWTH RATE HETERO POP 1 (1/DAY)
 4.0        MUO2   = MAX SPEC GROWTH RATE HETERO POP 2 (1/DAY)
 2.5        MUNI1  = MAX SPEC NIT BASE GROWTH RATE HETERO POP 1 (1/DAY)
 1770.0     RHOBD  = AVE BULK DENSITY OF POROUS MEDIUM (KG/M^3)
```

The last activity that CHMREAD performs is to compute the print times and display them on the console. NPRT, which was read in near the beginning of the file, specifies how many times the chemical concentrations and the microbe populations are to be printed on the file xxxxxxxx.CNC. The print times are equally spaced during the run.

CHMREAD returns control to the main program of the Chemistry initialization program, which copies the rest of the xxxxxxxx.CHD file to a file named LT3VSI.CLD.

Later, during the run of the Chemistry loop program, the subroutine LOOPIO reads the file LT3VSI.CLD, thereby reading the last of the data that was in the .CHD file. The following describes the actions of LOOPIO.

The first thing that LOOPIO does is to set the arrays

QCHM1S(I,J,K) and CSWELN(I,J,K) to zero. Then it reads the injection well chemical concentrations (kg chem/kg solution). The minimum number is one well, which may have strength zero.

NINJW (number of injection wells) lines are read. Each line specifies the location of a well and the concentrations of the four chemicals for that well.

```
CONCENTRATION OF INJECTION WELL (KG CHEM/KG SOLUTION)
 XI   YJ  NUTRIENT  SUBSTRATE  OXYGEN  NITRATE
0.1  0.1  0.0       0.0        0.0     0.0
```

There must be NINJW data lines, following the TWO text lines. Remember that NINJW and the node coordinates XNODE(I), YNODE(J) were read from the xxxxxxxx.WAD file. XI and YJ must be the coordinates of one of the injection wells specified in the .WAD file. The concentrations specified here are the initial values. They are stored in the array CSWELN(I,J,K). Concentrations can be changed at any time by using a schedule file. See Section IV.3.1.

Next, LOOPIO reads the number of buried chemical sources NBSOUR. The minimum is one source, which may have strength zero:

```
NUMBER OF BURIED SOURCES
  1
```

LOOPIO reads buried source positions and strengths for the NBSOUR sources:

```
CONCENTRATION OF BURIED SOURCE (KG CHEM/DAY)
 XI   YJ  NUTRIENT  SUBSTRATE  OXYGEN  NITRATE
0.1  0.1  0.0       0.0        0.0     0.0
```

There must be NBSOUR data lines, following the TWO text lines. Note that the location is given by coordinates, not by indices. The source can not be on the boundary but must be at an interior point. In this example, there are no buried sources; therefore a source of strength 0.0 is placed at position (0.1,0.1). Also

note that the input data gives the concentrations in units of (kg chem/day); the program divides the data by the appropriate volumes to get units of (kg chem/m^3 day). The concentrations are stored in the array QCHM1S(I,J,K).

The last line of the data file is not supposed to be read by the program. If it is read, there is an error in the data.

```
END OF CHEM DATA FILE
```

IV.3.1. Example schedule data file: LT3EXAM.SCH

If NFLAG(4) = 1 in the xxxxxxxx.CHD file, the Chemistry programs read and process a schedule data file, which specifies times at which the concentrations of chemicals at injection wells are to be changed. If NFLAG(4) = 0, no schedule file is read; the concentrations of chemicals at injection wells remain as specified by the initial data in the .CHD file.

This section shows an example of a schedule data file.

```
   DATE IS 9/21/1990 - EXAMPLE DATA FOR MODEL LT3VSI V1.2 [4 chem]
   SCHEDULE DATA FILE; ONE INJECTION WELL; ONE EXTRACTION WELL
ETIME  XI   YJ   NUT      SUB    OXY    NIT
0.0    0.1  0.1  0.0      0.0    0.0    0.0      [1]
4.5    0.1  0.1  0.0002   0.0    0.0    0.0      [2]
```

The file must have a name with extension ".SCH"; it is first read in by subroutine VALSCH in module LT3VSIC.FOR of the Chemistry initialization program. This subroutine "validates" the file. If it detects any errors, it writes messages describing them on the file xxxxxxxx.CDB; writes a warning message on the console; and stops. This avoids the problem of having a long time-consuming chemistry run aborted because of a schedule error.

During the run of the Chemistry loop program, the schedule file is read by subroutine PREVNT in module LOOPSIL.FOR. This subroutine processes events in the schedule file.

The first three lines of the file (see example above) are read and ignored by the programs. Following that come one or

more lines specifying events. On each line, the first value ETIME is the time in days from the beginning of the run at which the event is to occur. ETIME is the total time from the beginning of the initial run. In the example above, the initial run is for 4.0 days. The second event (ETIME = 4.5) occurs 0.5 days into the continuation run.

The second and third values (XI and YJ) on a data line specify the coordinates of an injection well at which the concentrations of one or more of the four chemicals are to be changed. The four remaining values on the line specify the new concentrations of the chemicals (Kg chemical/Kg solution). The values are stored in the array CSWELN(I,J,K). The notations such as [1] are excess data and are not read in by the programs.

There are several requirements on the data in a schedule file. Subroutine VALSCH verifies these requirements:

1. The data must be valid real numbers.

2. The ETIME in the first line must be non-negative, and the ETIME in each following line must not be less than that in the previous line. In other words, ETIME's must be non-decreasing.

3. XI and YJ must be the coordinates of an interior point; that is, XI must be equal to one the XNODE(I)'s and YJ must be equal to one of the YNODE(J)'s. Neither of them can be on a boundary of the aquifer.

4. The concentrations for the four chemicals must be non-negative.

5. There must be a non-zero injection well at point (XI,YJ).

The amount of fluid being injected at each well must remain constant throughout the run, because a change in injection rate would change the hydraulic pressure distribution. Only the concentrations of the chemicals can be changed.

The actual times at which events are processed will be at the TIME points (T0, T0+DT0, T0+2*DT0, ...) closest to the specified

times. DTO is "small" for the method used in this program, so that changing the concentrations at the nearest time points is sufficiently close to the specified times. TO is 0.0 for an "initial" chemical run. Also see Section V.7.

IV.3.2. LaGrangian interpolation.

If NFLAG(16) = 1 in the chemistry data file xxxxxxxx.CHD, the chemistry loop program will use a 25-point LaGrangian interpolation method; if NFLAG(16) = 0, a simpler 4-point method is used. The 25-point method takes about three times as much computing time and requires careful preparation of the data, but should have less numerical dispersion. If the 25-point method is used, the initial chemical concentrations, wells, and buried sources should have a Gaussian or Normal Surface distribution. Even with this form of data, small negative values of chemical concentration will appear in the results. The formula for a Gaussian surface is:

$$G(x,y) = \frac{A}{2\pi\sigma_x\sigma_y} e^{-(x-x_s)^2/(2\sigma_x^2)-(y-y_s)^2/(2\sigma_y^2)}$$

A is a measure of the peak amplitude of the surface, which occurs at the point $(x_s,y_s)$. $\sigma_x$ and $\sigma_y$ are the standard deviations in the x and y directions.

IV.4. Choosing the SIP method parameters

The program uses an iterative method, called the SIP method (Stone, 1968), to compute the steady-state hydraulic head at the nodal points in the aquifer. The following data (except ZTHRSH) from the file LT3EXAM.WAD is used to control the SIP process:

| NLSOR | NMOD | NALPH | TLRNWA | TLRNWR | ZTHRSH | (RUN CONTROL DATA) |
|---|---|---|---|---|---|---|
| 3000 | 2 | 4 | 1.0E-8 | 1.0E-8 | 1.0E-8 | |

| ALPHAS | | | |
|---|---|---|---|
| 1.0 | 0.90 | 0.95 | 0.75 |

NLSOR, NMOD, NALPH, TLRNWA, TLRNWR, and ALPHAS are the SIP method parameters; ZTHRSH is the "zero threshold" (see Section IV.2). The user must choose appropriate values for these parameters.

At each step in the iteration, the SIP method computes an improved approximation to the hydraulic head. One heuristic parameter, called ALPH, is used in the process. ALPH must be in the range 0.0 to 1.0 inclusive. It may happen that using one value for ALPH will result in convergence of the method. It is usually better to select several values for ALPH, using one value for several iterations, then the next value, etc. If convergence has not been achieved after using the last value, the first one is used again. In the xxxxxxxx.WAD data file, ALPHAS is an array of values for ALPH. NALPH specifies the number of ALPHS to be read; the program allows NALPH to be in the range from 1 to 10. The value of NMOD specifies how many times to use each value of ALPH before changing to the next one.

In the example data, NALPH = 4, NMOD = 2, and the ALPHS array consists of the 4 values 1.0, 0.90, 0.95, 0.75. This means that each of the ALPH values will be used twice; the sequence that will be used is: 1.0, 1.0, 0.90, 0.90, 0.95, 0.95, 0.75, 0.75, 1.0, 1.0, 0.90, etc. For the example data, this sequence results in convergence after 22 iterations. The sequence of ALPHS used in the example data was developed by a series of trials to solve a particularly difficult scenario. It is recommended that this sequence be used as a starting point in attempting to solve a problem. If the program fails to converge, or converges too slowly, then other values and/or a different number of values can

be tried. Stone (1968) gives a method for choosing parameters in a highly idealized case. A nonhomogeneous problem is not idealized; all that can be done is to try various sequences.

At each step, the program computes two measures of convergence. RMSEA is the Root Mean Square (RMS) of the residuals; RMSER is the RMS of the relative change in the approximate hydraulic head. The user specifies two tolerances TLRNWA and TLRNWR. When RMSEA becomes less than TLRNWA, or RMSER becomes less than TLRNWR, the computed head values are accepted. In the example data, both TLRNWA and TLRNWR are set to 1.0E-8. The two tolerances do not have to be the same; the user can select any appropriate positive values for them.

In addition to the variables mentioned above, the user must also choose the iteration limit NLSOR. If the number of iterations ICOUNT exceeds NSLOR, the program stops and displays an error message. NLSOR = 3000 is a very generous limit; it should not take that many iterations to reach convergence.

During the SIP process, the program will display the values of ICOUNT, ALPH, RMSEA, and RMSER after each iteration. RMSEA and RMSER should continually decrease until one of them becomes less than its corresponding tolerance TLRNWA or TLRNWR. Sometimes one or both of them will increase, especially when a new value is selected for ALPH. Such increases should be only temporary.

In typical program runs, the time to compute the hydraulic head by the SIP method is much shorter than the time to do the chemistry processing; therefore, one should not spend too much time trying to find the "best" values for the SIP parameters.

# V. Running the programs and output files

## V.1. Instructions for running the programs

(1) LT3VSI must have been installed on a VAX/FPS computer system, as described in Section II. The files are stored on the VAX and copied to the FPS when a run is made. The executable image files for the three programs have the names WATER.IMG. CHEMINIT.IMG, and CHEMLOOP.IMG. The batch files for running the programs have the names WATRUN3.COM, CHMRUN3.COM, and CHMGOON3.COM. These files will probably need to be modified. See Section V.2.

(2) The two required data files xxxxxxxx.WAD and xxxxxxxx.CHD must be prepared as described in Section IV. If a schedule file xxxxxxxx.SCH is to be used, it must also be prepared. The data files for the example shown in this manual are LT3EXAM.WAD, LT3EXAM1.CHD, LT3EXAM2.CHD, and LT3EXAM.SCH.

(3) The batch files are used to run the programs. The VMS "SUBMIT" command is used to put a .COM file into an appropriate batch queue for execution. The batch file submits a job to the FPS computer to make a water or chemistry run. The form of the SUBMIT command used on the Oregon State University system is:

```
SUBMIT/NOPRINT/QUEUE=FPS264  filename
```

The "NOPRINT" option is used because the VAX at Oregon State University does not have a printer connected. The queue named "FPS264" is used for submitting jobs to be run on the FPS computer. The options may need to be changed if the program is run on a different VAX/FPS pair of computers.

The "filename" is the name of one of the batch files without the .COM extension. The job will create a file with the same base name but an extension of .LOG that will contain the "console" output from the job. For example, if "filename" in the

SUBMIT command is "WATRUN3", the batch file WATRUN3.COM is submitted for execution and the console output is written on a file named WATRUN3.LOG.

The job will read one or more data files and write several output files. The batch job includes commands to copy the output files from the FPS computer to the VAX computer.

See sections V.3 through V.6 for details.

## V.2. Modifying the batch files

The *.COM batch files as distributed were used to run LT3VSI on the O.S.U. VAX/FPS system, using a certain account name. Some changes will be needed if they are to be run under other circumstances. This section contains listings of the batch files with explanations of the commands and data in them. It also points out what changes will be needed.

The batch file WATRUN3.COM is listed below, with comments at the right for explanation.

```
$SJE/ECHO/TIME/CONTINUE ! LT3VSI WATER RUN   Initiates FPS run
   ATT/W/PRIOR=5                             Attaches to FPS
   ACC :BACHELOR                             Accesses directory
   COPYIN/B WATER.IMG                        Copies in the water image
      WATER.IMG                                file and executes it
LT3EXAM                                      Base name for .WAD file
   COPYOUT LT3EXAM.WDB                       Copies out the
   COPYOUT LT3EXAM.WAO                         three result
   COPYOUT LT3EXAM.WPV                           files
   DEL LT3EXAM.WAO                           Deletes the two large
   DEL LT3EXAM.WPV                             result files and
   DEL WATER.IMG                               the image file
QUIT                                         Detaches from FPS and quits
```

The spaces at the left of most of the lines in the file are placed there to make the file more readable. The spaces are ignored by the computers. The line "LT3EXAM" is data to be read by the WATER program; there must NOT be leading spaces in this line. This data is the base name of the water data file whose extension is .WAD. If the xxxxxxxx.WAD file has a base name other than LT3EXAM, then the data line must be changed, and all

commands where the base name appears must also be changed. In each case, replace "LT3EXAM" by the base name of the .WAD file.

The first line "$SJE/ ..." is a VMS command that calls the FPS system. All of the other lines are FPS commands, except the data line "LT3EXAM".

The "ATT" command in the second line attaches the job to an FPS processor. This command specifies a priority of 5, which is suitable for running the small example on the O.S.U. system. The priority number may need to be changed if a larger problem is run, or if the job is run on another system.

The "ACC" command in the third line selects the user's file directory on the FPS system. The name in this line will need to be changed if anyone other than "BACHELOR" is running the job.

The chemistry batch files are listed below. They will need changes similar to the water batch file above.

The batch file CHMRUN3.COM is listed below. The comment on the first line has been truncated. Its complete form is:

```
! LT3VSI INITIAL CHEM RUN; CUF OUTPUT FILE.
```

```
$SJE/ECHO/TIME/CONTINUE ! LT3VSI ...   Initiates FPS run
   ATT/W/PRIOR=5                       Attaches to FPS
   ACC :BACHELOR                       Accesses directory
   COPYIN/B CHEMINIT.IMG               Copies in the chem init
   COPYIN LT3EXAM.SCH                    image file and sched file
      CHEMINIT.IMG
LT3EXAM                                Executes chem init file
LT3EXAM1                               .WIF file base name
LT3EXAMA                               .CHD file base name
LT3EXAM                                .CUF output file base name
                                       .SCH file base name
   DEL CHEMINIT.IMG                    Deletes chem init image
   COPYIN/B CHEMLOOP.IMG               Copies in the chem loop
      CHEMLOOP.IMG                       image file and executes it
   COPYOUT LT3EXAM1.CDB                Copies out the
   COPYOUT LT3EXAM1.CNC                  three result
   COPYOUT LT3EXAM1.CHO                    files
   DEL LT3EXAM1.CHO                    Deletes the two large
   DEL LT3EXAM1.CNC                      result files,
   DEL LT3EXAM.SCH                       the schedule file,
   DEL LT3VSI.CLD                        the chem loop data file,
   DEL LT3VSI.CIF                        the chem interface file,
   DEL CHEMLOOP.IMG                      and the image file
QUIT                                   Detaches from FPS and quits
```

This batch file is used to make an "initial" chemistry run, with provision for continuing it later. It runs both the CHEMINIT

and CHEMLOOP programs. The second and third lines may need changes similar to those described above for WATRUN3.COM.

There are four data lines in this file:

LT3EXAM
LT3EXAM1
LT3EXAMA
LT3EXAM

These lines specify the base file names for the data and interface files. In order, they are the base names for: (1) the .WIF water interface file, which is the same as the base name for the .WAD water data file used in the water run; (2) the .CHD chemistry data file; (3) the .CUF unformatted chemistry output file, which is written only if NFLAG(6) in the .CHD file is 1; (4) the .SCH schedule data file, which is used only if NFLAG(4) is 1. If NFLAG(6) or NFLAG(4) is 0, the corresponding base file name must be omitted from the batch file. In other words, if the program does not read or write a certain file, it does not read a name for it.

If the base file names are different from those used in this example, then the data lines and the corresponding commands in the .COM file must be changed.

The batch file CHMGOON3.COM is listed below. The comment on the first line has been truncated. Its complete form is:

```
! LT3VSI CONTINUATION RUN; CUF INPUT/OUTPUT.
```

```
$SJE/ECHO/TIME/CONTINUE ! LT3VSI ...      Initiates FPS run
   ATT/W/PRIOR=5                          Attaches to FPS
   ACC :BACHELOR                          Accesses directory
   COPYIN/B CHEMINIT.IMG                  Copies in the chem init
   COPYIN LT3EXAM.SCH                       image file and sched file
      CHEMINIT.IMG
LT3EXAM                                   Executes chem init file
LT3EXAM2                                  .WIF file base name
LT3EXAMA                                  .CHD file base name
LT3EXAMB                                  .CUF input file base name
LT3EXAM                                   .CUF output file base name
                                          .SCH file base name
   DEL CHEMINIT.IMG                       Deletes chem init image
   COPYIN/B CHEMLOOP.IMG                  Copies in the chem loop
      CHEMLOOP.IMG                          image file and executes it
   COPYOUT LT3EXAM2.CDB                   Copies out the
   COPYOUT LT3EXAM2.CNC                     three result
   COPYOUT LT3EXAM2.CHO                       files
   DEL LT3EXAM2.CHO                       Deletes the two large
   DEL LT3EXAM2.CNC                         result files,
   DEL LT3EXAM.SCH                          the sched file,
   DEL LT3VSI.CLD                           the chem loop data file,
   DEL LT3VSI.CIF                           the chem interface file,
   DEL CHEMLOOP.IMG                         and the image file
QUIT                                      Detaches from FPS and quits
```

CHMGOON3.COM is very similar to CHMRUN3.COM. It performs a "continuation" run, in which the chemicals and microbe populations are initialized to the state they had at the end of the previous run. The .CHD file is different; specifically, it has NFLAG(5) set to 1, to cause the program to read an input .CUF file. It may also have different time limits and other parameters. The data lines in the .COM file above are:

LT3EXAM
LT3EXAM2
LT3EXAMA
LT3EXAMB
LT3EXAM

The data lines are the base file names for the .WIF file, the .CHD file, the input .CUF file, the output .CUF file, and the .SCH file.

The following sections explain how to run the example problem that is used for illustration in this manual. To run other problems, change the data files and batch files.

V.3. Running the water program.

The file WATRUN3.COM may need to be modified before running it as a batch job. See Section V.2 for information on the changes needed. To run the water program, type the following command at the VMS prompt:

```
SUBMIT/NOPRINT/QUEUE=FPS264  WATRUN3
```

The SUBMIT command places the batch file WATRUN3.COM into the FPS264 queue to be executed. The commands in the .COM file submit a job to the FPS computer to run the program WATER.IMG with data file LT3EXAM.WAD as input. As the program executes, its "console" output is written on a file named WATRUN3.LOG. After the job has finished, the contents of the .LOG file must be examined to see whether the job ran without errors. If all goes well, the WATRUN3.LOG file should be very similar to the file of the same name in the file LT3RESLT.ARC from the distribution diskette.

Shown below is part of the file WATRUN3.LOG, the console output from running WATRUN3.COM with the file LT3EXAM.WAD as input data:

```
      TWO-DIMENSIONAL STEADY WATER FLOW IN
      THE LONG THIN RSKERL PHYSICAL AQUIFER.
    ==========================================

     Models a 2 dimensional (horizontal) flow field
 for a single layer porous medium.  The medium can
 be anisotropic as well as nonhomogeneous.  Pressure
 and velocity components are calculated at each
 nodal point.  The flow field is confined to
 the interior of the rectangular boundaries.
 A Finite Difference (space centered) method
 is used.  An interface file is written which is
 used to pass information to the chemistry-
 processing program.

 ++++++++++++++++++++++++++++++++++++++++++++++++++++
 G. A. Bachelor,  Sr. Systems Analyst,
 D. E. Cawlfield, Sr. Systems Analyst,
 F. T. Lindstrom, Assoc. Prof.,
 Soil Science Dept. Oregon State Univ.,
 Corvallis, OR., 97331,   (503) 737-2441
 ++++++++++++++++++++++++++++++++++++++++++++++++++++
 Enter base name of WATER data file.
Do not enter the extension, which is ".WAD":
LT3EXAM

 Input  file 1 is:  LT3EXAM.WAD
 Output file 2 is:  LT3EXAM.WAO
 Output file 3 is:  LT3EXAM.WPV
 Output file 8 is:  LT3EXAM.WIF
 Output file 9 is:  LT3EXAM.WDB
```

This output is followed by messages telling what the program is doing: calling subroutines, reading data, writing outputs, etc. If the program terminates because of an error, the last message displayed before the error messages will indicate where the error occurred.

Shown below is the display during the SIP method that indicates how rapidly the method is converging. If the method does not converge, the program will usually detect that and terminate with an error message. If this happens, the SIP parameters in the .WAD file must be changed and the job re-submitted. See Section IV.4.

```
 Beginning master loop for the hydraulic field

ICOUNT      ALPH          RMSEA          RMSER
Initial    1.0000       0.59332
     1     1.0000     8.08087-002    0.41638
     2     1.0000     4.52632-002    6.68348-002
     3    0.90000     1.27992-002    3.44458-002
     4    0.90000     5.06561-003    7.04382-003
     5    0.95000     2.42803-003    2.44346-003
     6    0.95000     1.21483-003    9.18589-004
     7    0.75000     6.41884-004    3.48462-004
     8    0.75000     3.98969-004    2.00459-004
     9     1.0000     1.65868-004    2.99693-004
    10     1.0000     8.18173-005    6.46390-005
    11    0.90000     3.54592-005    2.03161-005
    12    0.90000     1.67697-005    9.86753-006
    13    0.95000     8.24437-006    6.02614-006
    14    0.95000     3.99255-006    3.00095-006
    15    0.75000     1.87433-006    1.03432-006
    16    0.75000     1.11224-006    5.85792-007
    17     1.0000     4.77407-007    8.18155-007
    18     1.0000     2.39171-007    1.79081-007
    19    0.90000     1.03913-007    5.81736-008
    20    0.90000     4.96330-008    3.01280-008
    21    0.95000     2.47757-008    1.84186-008

  Convergence Achieved -- Final Values:
ICOUNT=   22, RMSEA=  1.225534-008, RMSER=  9.291242-009
  Cycle Time =     0.03 (Startup Time =     0.01)
  Total Time =     0.04
```

More messages are displayed, and the program finishes with the following:

```
Total integration time was 00 days, 00 hours, 00 minutes, and 00 sec.

Total CPU  Clock  time was 00 days, 00 hours, 00 minutes, and 00 sec.

STOP   Normal Fortran Termination
```

For this small problem, the FPS 264 computer consumes less than 1 second of CPU time. The .LOG file also contains echoed commands and messages from the operating system, before and after the console output from the WATER program.

## V.4. Output files for water phase

If the job runs correctly, it will "copy out" the three result files from the FPS to the VAX and delete the two larger

files from the FPS. The files copied out are LT3EXAM.WDB, LT3EXAM.WAO, and LT3EXAM.WPV. They should be very similar to the files with the same names in LT3RESLT.ARC on the distribution diskette.

File xxxxxxxx.WAO shows the information read from xxxxxxxx.WAD, together with computed data. This output file should be examined to be sure that the data were read in correctly. All of the numbers in this file are labeled to indicate what they represent.

The file xxxxxxxx.WPV shows the water pressure and Darcy velocity fields at all of the node points in the aquifer.

The file xxxxxxxx.WDB shows "debugging" information, if any of this information was selected by setting certain NFLAG's. See the discussion of NFLAG's in Section III.2. If none of these NFLAG's were set to 1, xxxxxxxx.WDB will show only the final convergence data from the water run and the timing for the run.

NFLAG(8) affects the output as follows: if this flag is zero, arrays are printed in a format that uses no more than 80 columns. If NFLAG(8) is 1, a wider format, but no wider than 132 columns, is used. The wider format is better, if a printer capable of printing that many columns is available. "Columns" here means "print positions". NFLAG(8) was set to 0 in the example file, so that portions of the output files could be shown in this manual.

When the water program is run with the data file LT3EXAM.WAD as input, it produces an output file LT3EXAM.WAO, part of which is shown below. The format of the print-out for the "x-comp hydraulic conductivity KSATXX(i,j)" is the narrow format, produced when NFLAG(8) = 0. If NFLAG(8) = 1 in the xxxxxxxx.WAD data file, a wider format is printed, with 12 columns of the matrix printed on each "page", instead of 7. Also, when several consecutive lines of numbers are the same, only the first such

line is shown, followed by a message of the form "Line above repeated 5 times."

Most of the print-outs of matrices are in the same format as shown below.

```
     DATE IS 9/21/1990 - EXAMPLE DATA FOR MODEL LT3VSI V1.2 [4 chem]
     ONE INJECTION WELL; ONE EXTRACTION WELL

    ** NOTE!  SI units are indicated, but any units  **
    ** can be used, as long as they are CONSISTENT.  **

    RUN CONTROL INFORMATION.
    NLSOR=          3000  NMOD=              2  NALPH=              4
    TLRNWA=  1.00000-008  TLRNWR=  1.00000-008  ZTHRSH=  1.00000-008

          I   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
    NFLAG(I)  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

ALPHAS (array of ALPH's)
  1.0000       0.90000       0.95000       0.75000

        NODE COORDINATES (M)

    XNODE(I)
    1         2         3         4         5         6         7         8
 0.       0.10000   0.20000   0.40000   0.60000   0.80000    1.0000    1.2000

    9        10
 1.3000    1.4000


    YNODE(J)
    1         2         3         4         5         6         7         8
 0.       0.10000   0.30000   0.50000   0.70000   0.90000    1.1000    1.3000

    9        10
 1.5000    1.6000


    DX(I)
    1         2         3         4         5         6         7         8
0.10000   0.10000   0.20000   0.20000   0.20000   0.20000   0.20000   0.10000

    9
0.10000


    DY(J)
    1         2         3         4         5         6         7         8
0.10000   0.20000   0.20000   0.20000   0.20000   0.20000   0.20000   0.20000

    9
0.10000


    WIDTH     OF AQUIFER (M)=  1.40
    LENGTH    OF AQUIFER (M)=  1.60
    THICKNESS OF AQUIFER (M)= 0.300

    INLET/OUTLET HYDRAULIC PRESSURES (M WATER)
```

```
    HIN=   1.0000      HOUT=  0.00000

         BASIC SOIL CHARACTERIZING PARAMETERS (KG/M^3)

    RHOWAT=  1.000+003

         TABLE OF SOIL PROPERTIES


         X-COMP HYDRAULIC CONDUCTIVITY KSATXX(I,J) (M/DAY)

    PAGE  1
       X   0.         .1000      .2000      .4000      .6000      .8000      1.000
  Y            1          2          3          4          5          6          7
0.      1 0.10000    0.10000    0.10000    0.31600    1.0000     1.0000     1.0000
          Line above repeated   2 times.
.5000   4 0.03160    0.03160    0.03160    0.05620    0.10000    0.10000    0.31600
.7000   5 0.01000    0.01000    0.01000    0.01000    0.01000    0.01000    0.10000
          Line above repeated   1 times.
1.100   7 0.01000    0.01000    0.01000    0.01000    0.01000    0.01000    0.05620
1.300   8 0.01000    0.01000    0.01000    0.01000    0.01000    0.01000    0.03160
          Line above repeated   2 times.

    PAGE  2
       X   1.200      1.300      1.400
  Y            8          9         10
0.      1  1.0000     1.0000     1.0000
          Line above repeated   5 times.
1.100   7 0.31600    0.31600    0.31600
1.300   8 0.10000    0.10000    0.10000
          Line above repeated   2 times.

         Y-COMP HYDRAULIC CONDUCTIVITY KSATYY(I,J) (M/DAY)

...     ...     ...     ...     ...     ...     ...     ...     ...     ...
(Listing of the y-component of hydraulic conductivity has been omitted)
...     ...     ...     ...     ...     ...     ...     ...     ...     ...

    TABLE OF INJECTION WELLS (KG WATER/M^3 DAY)

        XI          YJ     QWELIN(XI,YJ)
    ------------------------------------
     0.100       0.100        1666.7


    TABLE OF EXTRACTION WELLS (KG WATER/M^3 DAY)

        XI          YJ     QWELOT(XI,YJ)
    ------------------------------------
      1.30        1.50         888.89
```

The water program, when run with the data file LT3EXAM.WAD as input, also produces an output file LT3EXAM.WPV, shown below. The first part of the output is the pressure field, printed in the same format as other arrays.

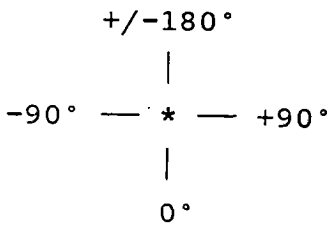
The second part shows the Darcy velocity field, in polar coordinate form. For each point in the aquifer grid, a pair of

numbers is printed. The first number is the magnitude of the Darcy velocity in meters per day; the second number is the angle in degrees, as follows:

.08098,-16.6o  .09058, .000o  .05954, 19.5o

where the "o" means degrees.

The x coordinate runs across the page, and the y coordinate runs vertically down the page. The inlet end is at the top; the outlet end is at the bottom of the page. The fluid flow thus runs from top to bottom. The convention for the angle is as follows: zero degrees means parallel to the y coordinate; negative angles mean that the fluid is flowing toward the left side (smaller x values); positive angles mean that the fluid is flowing toward the right side (larger x values), as follows.




DATE IS 9/21/1990 - EXAMPLE DATA FOR MODEL LT3VSI V1.2 [4 chem]
ONE INJECTION WELL; ONE EXTRACTION WELL

** NOTE! SI units are indicated, but any units **
** can be used, as long as they are CONSISTENT. **

OUTPUT DATA FOR FLOW SYSTEM

HYDRAULIC PRESSURE FIELD (M WATER)

PAGE 1

| | X | 0. | .1000 | .2000 | .4000 | .6000 | .8000 | 1.000 |
|---|---|---|---|---|---|---|---|---|
| Y | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0. | 1 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| .1000 | 2 | 1.1521 | 1.1237 | 1.0385 | 1.0025 | 0.99724 | 0.99447 | 0.99099 |
| .3000 | 3 | 1.0488 | 1.0415 | 1.0199 | 0.99783 | 0.99120 | 0.98396 | 0.97285 |
| .5000 | 4 | 1.0069 | 1.0044 | 0.99701 | 0.98867 | 0.98547 | 0.97191 | 0.94577 |
| .7000 | 5 | 0.91655 | 0.91692 | 0.91804 | 0.93372 | 0.94826 | 0.92550 | 0.89421 |
| .9000 | 6 | 0.73066 | 0.73196 | 0.73587 | 0.75298 | 0.78089 | 0.81959 | 0.83182 |
| 1.100 | 7 | 0.53500 | 0.53657 | 0.54129 | 0.56145 | 0.60273 | 0.68513 | 0.73982 |
| 1.300 | 8 | 0.32733 | 0.32861 | 0.33242 | 0.34880 | 0.38346 | 0.45202 | 0.51688 |
| 1.500 | 9 | 0.11000 | 0.11047 | 0.11187 | 0.11787 | 0.13029 | 0.15256 | 0.16691 |
| 1.600 | 10 | 0. | 0. | 0. | 0. | 0. | 0. | 0. |

```
    PAGE  2
       X   1.200      1.300      1.400
  Y            8          9         10
0.      1  1.0000     1.0000     1.0000
.1000   2 0.98768    0.98683    0.98655
.3000   3 0.96140    0.95881    0.95795
.5000   4 0.92674    0.92391    0.92297
.7000   5 0.88293    0.88146    0.88097
.9000   6 0.83476    0.83509    0.83519
1.100   7 0.76353    0.76593    0.76673
1.300   8 0.55672    0.55688    0.55693
1.500   9 0.16110    0.12248    0.10961
1.600  10  0.          0.         0.

        DARCY VELOCITY FIELD
    (MAGNITUDE IN M/D,ANGLE IN DEGREES)

    PAGE  1
       X      0.              .1000            .2000            .4000
  Y                1                2                3                4
0.      1       0.0,0.000o   0.12370, 180.o   0.03848,-180.o   0.00782,-180.o
.1000   2       0.0,0.000o   0.08921, 140.o   0.06674, 110.o   0.03270, 94.9o
.3000   3       0.0,0.000o   0.03313, 25.8o   0.02087, 60.2o   0.02516, 64.3o
.5000   4       0.0,0.000o   0.00997, 9.00o   0.00829, 13.9o   0.00915, 10.2o
.7000   5       0.0,0.000o   0.00681,-.628o   0.00654,-2.95o   0.00594,-7.31o
.9000   6       0.0,0.000o   0.00951,-1.57o   0.00943,-3.32o   0.00937,-6.90o
1.100   7       0.0,0.000o   0.01009,-1.79o   0.01011,-3.69o   0.01022,-8.64o
1.300   8       0.0,0.000o   0.01066,-1.37o   0.01075,-2.81o   0.01116,-6.56o
1.500   9       0.0,0.000o   0.01100,-.487o   0.01114,-.995o   0.01172,-2.25o
1.600  10       0.0,0.000o   0.01105,0.000o   0.01119,0.000o   0.01179,0.000o

    PAGE  2
       X      .6000            .8000            1.000            1.200
  Y                5                6                7                8
0.      1   0.02762, .000o   0.05525,0.000o   0.09009,0.000o   0.12324,0.000o
.1000   2   0.03479, 35.1o   0.05655, 16.0o   0.09189, 10.7o   0.12645, 5.06o
.3000   3   0.04546, 49.7o   0.07273, 39.1o   0.12634, 26.5o   0.15661, 13.4o
.5000   4   0.01153, 21.3o   0.01767, 34.2o   0.07164, 29.9o   0.20259, 14.5o
.7000   5   0.00512, 2.30o   0.00404, 19.5o   0.03041, 20.5o   0.23172, 7.09o
.9000   6   0.00880,-10.9o   0.00614,-12.0o   0.03878,-5.61o   0.29859,-1.36o
1.100   7   0.01041,-17.3o   0.00981,-20.5o   0.04560,-14.0o   0.22035,-4.57o
1.300   8   0.01209,-12.3o   0.01373,-14.1o   0.04601,-10.4o   0.15076,-2.56o
1.500   9   0.01293,-3.84o   0.01519,-3.46o   0.05360,-.721o   0.17538, 8.76o
1.600  10   0.01303,0.000o   0.01526,0.000o   0.05274,0.000o   0.16110,0.000o

    PAGE  3
       X      1.300            1.400
  Y                9               10
0.      1   0.13167,0.000o      0.0,0.000o
.1000   2   0.13459, 2.39o      0.0,0.000o
.3000   3   0.15825, 6.25o      0.0,0.000o
.5000   4   0.19431, 5.57o      0.0,0.000o
.7000   5   0.22227, 2.53o      0.0,0.000o
.9000   6   0.28883,-.429o      0.0,0.000o
1.100   7   0.21984,-1.32o      0.0,0.000o
1.300   8   0.16086,-.038o      0.0,0.000o
1.500   9   0.15619, 9.49o      0.0,0.000o
1.600  10   0.12248,0.000o      0.0,0.000o
```

## V.5. Making an initial chemistry run.

The batch file CHMRUN3.COM is used to make an initial

chemistry run, with a .CUF file output to allow continuation of the run. The CHMRUN3.COM file may need to be modified, as explained in Section V.2. When the changes have been made, type the VMS command:

```
SUBMIT/NOPRINT/QUEUE=FPS264  CHMRUN3
```

This submits a job to the FPS computer to run CHEMINIT.IMG and CHEMLOOP.IMG using the .WIF file produced by the water run, the data file LT3EXAM1.CHD, and the schedule file LT3EXAM.SCH. The console output is written on CHMRUN3.LOG. If the job runs correctly, the three result files LT3EXAM1.CDB, LT3EXAM1.CNC, and LT3EXAM1.CHO are copied out to the VAX. All of these files should be very similar to files of the same names in LT3RESLT.ARC.

Shown below is part of the file CHMRUN3.LOG, the console output from running CHMRUN3.COM.

```
        TWO-DIMENSIONAL DYNAMIC CHEMICAL
        TRANSPORT AND FATE IN THE LONG THIN
        RSKERL PHYSICAL AQUIFER.
    ========================================

     Models a 2 dimensional (horizontal) flow field
 for a single layer porous medium.  The medium can
 be anisotropic as well as nonhomogeneous.  The fluid
 velocity components must have been calculated by
 the water-processing program and written to an
 interface file.  This program calculates the
 dispersion coefficients and other variables at each
 nodal point.  Chemical concentrations and microbial
 populations are computed at each nodal point at
 specified time intervals.  They are printed at
 selected times.  A Finite Difference (space
 centered) method is used.

 ++++++++++++++++++++++++++++++++++++++++++++++++++
 G. A. Bachelor,  Sr. Systems Analyst,
 D. E. Cawlfield, Sr. Systems Analyst,
 F. T. Lindstrom, Assoc. Prof.,
 Soil Science Dept. Oregon State Univ.,
 Corvallis, OR., 97331,   (503) 737-2441
 ++++++++++++++++++++++++++++++++++++++++++++++++++
Enter base name of INTERFACE file, from Water run.
Do not enter the extension, which is ".WIF":
LT3EXAM
Enter base name of CHEMISTRY data file.
Do not enter the extension, which is ".CHD":
LT3EXAM1

 Input  file  1 is: LT3EXAM1.CHD
 Output file  2 is: LT3EXAM1.CHO
 Input  file  8 is:  LT3EXAM.WIF
 Output file  9 is: LT3EXAM1.CDB
 Output file 15 is:   LT3VSI.CIF
 Output file 16 is:   LT3VSI.CLD
```

The portion of CHMRUN3.LOG shown above is the initial output from the CHEMINIT program. This output is followed by messages telling what the program is doing. If an error occurs, the messages will give an indication of where the error happened. More messages appear between the portions of the .LOG file shown below.

```
Enter base name of UNFORMATTED chem OUTPUT file.
Do not enter the extension, which is ".CUF":
LT3EXAMA
```

The messages above appear if NFLAG(6) = 1; this causes CHEMINIT to read a base name for an output .CUF file, to allow continuing the run later.

```
The print times are as follows:

   2.0000     4.0000
```

This message displays the times at which the chemical concentrations and the microbial populations will be printed on the xxxxxxxx.CNC file.

```
Enter base name of SCHEDULE data file.
Do not enter the extension, which is ".SCH":
LT3EXAM
 Input  file  4 is:  LT3EXAM.SCH
```

If NFLAG(4) = 1, these messages appear and the program reads the base name of the schedule file.

```
CPU time for Chem Init was 00 days, 00 hours, 00 minutes, and 03 sec.
 End of chemical initialization phase.
STOP   Normal Fortran Termination
```

These messages appear at the end of the CHEMINIT run. They are followed by messages from the operating system. Then comes the first output from the CHEMLOOP program:

```
 Input  file 15 is:   LT3VSI.CIF

 Reading interface file from LT3VSI.CIF

 Interface file successfully read in.

 Input  file  1 is:   LT3VSI.CLD
 Output file  2 is: LT3EXAM1.CHO
 Output file  3 is: LT3EXAM1.CNC
 Input  file  4 is:  LT3EXAM.SCH
 Output file  9 is: LT3EXAM1.CDB
 Output file 11 is: LT3EXAMA.CUF
```

This output is followed by messages telling what the CHEMLOOP program is doing. When it enters the loop, the following display appears:

```
 Beginning master loop for the dynamic chemical field

 Computing the P* points.

IC =       1 Time =     0.0010
IC =       2 Time =     0.0020
IC =       3 Time =     0.0030
IC =       4 Time =     0.0040
IC =       5 Time =     0.0050
IC =       6 Time =     0.0060
IC =       7 Time =     0.0070
IC =       8 Time =     0.0080
IC =       9 Time =     0.0090
IC =      10 Time =     0.0100
IC =      20 Time =     0.0200
IC =      30 Time =     0.0300
IC =      40 Time =     0.0400
IC =      50 Time =     0.0500
IC =      60 Time =     0.0600
IC =      70 Time =     0.0700
IC =      80 Time =     0.0800
IC =      90 Time =     0.0900
IC =     100 Time =     0.1000
IC =     200 Time =     0.2000
IC =     300 Time =     0.3000
IC =     400 Time =     0.4000
IC =     500 Time =     0.5000
IC =     600 Time =     0.6000
IC =     700 Time =     0.7000
IC =     800 Time =     0.8000
IC =     900 Time =     0.9000
IC =    1000 Time =     1.0000
IC =    2000 Time =     2.0000
IC =    3000 Time =     3.0000
IC =    4000 Time =     4.0000
IC =    4000 Time =     4.0000

 Time t meets or exceeds TMAX! CEASE COMPUTING!
```

IC is the iteration count; time is the time in days since the beginning of the run. As can be seen, the program displays IC and time less frequently as time goes along; the purpose of this is avoid large amounts of unnecessary output.

```
(Integration time was 00 days, 00 hours, 00 minutes, and 53 sec.)

CPU time for Chem Loop was 00 days, 00 hours, 00 minutes, and 54 sec.
 End of this simulation run.
STOP   Normal Fortran Termination
```

The messages above appear at the end of the CHEMLOOP run. Messages from the operating system appear before the first program output and also after the last program output.

## V.6. Output files for chemistry phase

File xxxxxxxx.CHO shows the information read from xxxxxxxx.CHD, together with computed data. This file should be examined to be sure that the data were read in correctly. The numbers in this file are labeled to indicate what they represent.

The file xxxxxxxx.CNC shows the computed chemical concentrations and microbial populations at selected print times.

The file xxxxxxxx.CDB shows "debugging" information, if any of this information was selected by setting certain NFLAG's. See the discussion of NFLAG's in Section III.2. If none of these NFLAG's were set to 1, xxxxxxxx.CDB will contain only the timing data.

When the chemistry programs are run with the data file LT3EXAM1.CHD as input, they produce an output file LT3EXAM1.CHO, part of which is shown below. The format of the print-out for the matrices is the narrow format, produced when NFLAG(8) = 0. If NFLAG(8) = 1 in the xxxxxxxx.CHD data file, a wider format is printed, with 12 columns of the matrix printed on each "page", instead of 7.

```
     DATE IS 9/21/1990 - EXAMPLE DATA FOR MODEL LT3VSI V1.2 [4 chem]
     CHEM DATA FILE; ONE INJECTION WELL; ONE EXTRACTION WELL

** NOTE!  SI units are indicated, but any units  **
** can be used, as long as they are CONSISTENT.  **

     RUN CONTROL INFORMATION.

     TIMES IN DAYS:
TO=  0.000      TMAX=   4.00      DTO=  1.000-003
COMPUTED   NUTRIENT    SUBSTRATE   OXYGEN      NITRATE
  DTMAX=   7.130-002   7.932-002   2.350-003   5.478-003

NPRT=     2;  PRINT TIMES (DAYS) ARE:

2.0000    4.0000

ZTHRSH=  0.10000-007

      I   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
NFLAG(I)  1  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  1  0  0  0


     LISTING OF SCHEDULE FILE.
```

```
        DATE IS 9/21/1990 - EXAMPLE DATA FOR MODEL LT3VSI V1.2 [4 chem]
        SCHEDULE DATA FILE; ONE INJECTION WELL; ONE EXTRACTION WELL

      ETIME      XI         YJ        NUTRIENT  SUBSTRATE OXYGEN    NITRATE

     0.        0.10000    0.10000    0.         0.        0.        0.
     4.5000    0.10000    0.10000    2.00-004   0.        0.        0.

    INLET PORT TANK LENGTH (M)=  0.100
    EXIT  PORT TANK LENGTH (M)=  0.100


        BASIC SOIL CHARACTERIZING PARAMETERS (KG/M^3)

    RHOSND=  2.660+003  RHOCLA=  2.650+003  RHOORG= 1.300+003


        TORTUOSITY FACTOR TORT(I,J) (DIMLESS)

    PAGE  1
       X   0.          .1000      .2000      .4000      .6000      .8000      1.000
  Y            1          2          3          4          5          6          7
0.      1 0.67000    0.67000    0.67000    0.77700    0.90000    0.90000    0.90000
          Line above repeated   2 times.
.5000   4 0.57900    0.57900    0.57900    0.62300    0.67100    0.67100    0.77700
.7000   5 0.50000    0.50000    0.50000    0.50000    0.50000    0.50000    0.67100
          Line above repeated   1 times.
1.100   7 0.50000    0.50000    0.50000    0.50000    0.50000    0.50000    0.62300
1.300   8 0.50000    0.50000    0.50000    0.50000    0.50000    0.50000    0.57900
          Line above repeated   2 times.

    PAGE  2
       X   1.200      1.300      1.400
  Y            8          9         10
0.      1 0.90000    0.90000    0.90000
          Line above repeated   5 times.
1.100   7 0.77700    0.77700    0.77700
1.300   8 0.67000    0.67000    0.67000
          Line above repeated   2 times.

   ...  (listings of EPS, PCTSAN, PCTCLA, PCTORG omitted)   ...      ...

        DISPERSIVITY COMPONENTS DISPLX(I,J,K) AND DISPLY(I,J,K)

        X-COMP DISPERSIVITY FOR NUTRIENT  (M)

    PAGE  1
       X   0.          .1000      .2000      .4000      .6000      .8000      1.000
  Y            1          2          3          4          5          6          7
0.      1 0.00350    0.00350    0.00350    0.00205    0.00120    0.00120    0.00120
          Line above repeated   2 times.
.5000   4 0.00592    0.00592    0.00592    0.00453    0.00346    0.00346    0.00204
.7000   5 0.01000    0.01000    0.01000    0.01000    0.01000    0.01000    0.00346
          Line above repeated   1 times.
1.100   7 0.01000    0.01000    0.01000    0.01000    0.01000    0.01000    0.00453
1.300   8 0.01000    0.01000    0.01000    0.01000    0.01000    0.01000    0.00592
          Line above repeated   2 times.

    PAGE  2
       X   1.200      1.300      1.400
  Y            8          9         10
0.      1 0.00120    0.00120    0.00120
          Line above repeated   5 times.
1.100   7 0.00205    0.00205    0.00205
1.300   8 0.00350    0.00350    0.00350
          Line above repeated   2 times.
```

```
...      (Y-Comp Dispersivity for Nutrient omitted)      ...      ...

...  (Dispersivities for Substrate, Oxygen, & Nitrate omitted)   ...

      CHEMICAL PARAMETERS

        NUTRIENT    SUBSTRATE   OXYGEN      NITRATE

DLO=  1.430-004   1.430-004   1.430-004   1.430-004  (M^2/DAY)

KSAND= 0.000        0.000       0.000       0.000     (M^3/KG SAND)
KCLAY= 0.000        0.000       0.000       0.000     (M^3/KG SILT)
KORG=  0.000        0.000       0.000       0.000     (M^3/KG ORGANICS)

CIN=   3.000-003  1.500-002   5.000-003   5.000-003 (KG/M^3)
COUT=  0.000      0.000       0.000       0.000     (KG/M^3)
CO=    3.000-003  1.500-002   5.000-003   5.000-003 (KG/M^3)


      FIRST ORDER LOSS COEFS XLAMIR(I,J,K) & XSLMIR(I,J,K)


      IRREVERSIBLE LOSS IN FREE PHASE FOR NUTRIENT  (1/DAY)

...      ...  (Listing of data omitted)  ...      ...      ...      ...

...  (Free phase losses for Substrate, Oxygen, Nitrate omitted)  ...

      IRREVERSIBLE LOSS IN SORBED PHASE FOR NUTRIENT  (1/DAY)

   PAGE  1
      X   0.        .1000     .2000     .4000     .6000     .8000     1.000
  Y           1         2         3         4         5         6         7
0.    1  0.        0.        0.        0.        0.        0.        0.
       Line above repeated   9 times.

   PAGE  2
      X   1.200     1.300     1.400
  Y           8         9        10
0.    1  0.        0.        0.
       Line above repeated   9 times.

... (Sorbed phase losses for Substrate, Oxygen, Nitrate omitted) ...

      INITIAL CHEMICAL DISTRIBUTION OF NUTRIENT  (KG/M^3)

   PAGE  1
      X   0.        .1000     .2000     .4000     .6000     .8000     1.000
  Y           1         2         3         4         5         6         7
0.    1 0.00300   0.00300   0.00300   0.00300   0.00300   0.00300   0.00300
       Line above repeated   9 times.

   PAGE  2
      X   1.200     1.300     1.400
  Y           8         9        10
0.    1 0.00300   0.00300   0.00300
       Line above repeated   9 times.

...   (Chemical distr of substrate, oxygen, nitrate omitted)      ...

      1+RETENTION(I,J,K) FOR NUTRIENT  (DIMLESS)

...      ...  (Listing of data omitted)  ...      ...      ...      ...

... (1+Retention for Substrate, Oxygen, Nitrate omitted) ...      ...
```

```
        OVER ALL FIRST ORDER LOSS COEF LAMDA(I,J,K) FOR NUTRIENT  (1/DAY)

   ...     ... (Listing of data omitted)   ...     ...     ...     ...

   ... (Over all loss coef for Substrate, Oxygen, Nitrate omitted) ...

        INITIAL DISTRIBUTION OF MICROBE POPULATION #1 (KG CELLS/KG SOIL)

    PAGE  1
       X  0.        .1000     .2000     .4000     .6000     .8000     1.000
  Y          1         2         3         4         5         6         7
0.     1  1.00-007  1.00-007  1.00-007  1.00-007  1.00-007  1.00-007  1.00-007
         Line above repeated   9 times.

    PAGE  2
       X  1.200     1.300     1.400
  Y          8         9        10
0.     1  1.00-007  1.00-007  1.00-007
         Line above repeated   9 times.

        INITIAL DISTRIBUTION OF MICROBE POPULATION #2 (KG CELLS/KG SOIL)

   ...     ... (Listing of data omitted)   ...     ...     ...     ...

        CHEMISTRY USAGE PARAMETERS

    KSO1   =  1.8000-002 (KG/M^3)
    KSO2   =  1.8000-002 (KG/M^3)
    KO1    =  3.0000-005 (KG/M^3)

   ...     ... (Part of listing omitted)   ...     ...     ...     ...

    YSO1   =  0.4000     (KG CELLS/KG SUB)
    YSO2   =  0.4000     (KG CELLS/KG SUB)

   ...     ... (Part of listing omitted)   ...     ...     ...     ...

    MUNI1  =   2.500     (1/DAY)
    RHOBD  =   1770.     (KG/M^3)


    INJECTION WELLS CHEMICAL CONC. CSWELN(XI,YJ,K) (KG CHEM/KG SOLUTION)

       XI         YJ       NUTRIENT    SUBSTRATE  OXYGEN     NITRATE
    ------------------------------------------------------------------
    NO NON-ZERO INJECTION WELLS.

    BURIED CHEMICAL SOURCE CONC. QCHM1S(XI,YJ,K) (KG CHEM/M^3 DAY)

       XI         YJ       NUTRIENT    SUBSTRATE  OXYGEN     NITRATE
    ------------------------------------------------------------------
    NO NON-ZERO BURIED SOURCES.
```

The chemistry programs, when run with the data file LT3EXAM1.CHD as input, also produce an output file LT3EXAM1.CNC, part of which is shown below. Most of the output is devoted to showing the chemical concentrations and the microbial populations at the nodal points of the aquifer, at each of the print times

listed near the beginning of the xxxxxxxx.CHO file. In two-dimensional arrays, as printed in this file and the files mentioned previously, the numbers are printed in a variable format. In the first row below are examples from the PC/Microsoft Fortran version; in the second row are examples from the FPS APFTN64 version. The FORMAT statements are identical in the two versions, but the support libraries for the two systems produce different forms:

```
0.     .00    8.65E-05     .00189     .99447    1.1521
0.    0.00    8.65-005    0.00189    0.99447    1.1521
```

When a number is within a certain range, it is printed in fixed-point format, with the decimal point in its proper position. Numbers outside this range are printed in exponential format, with two exceptions. Numbers that are actually zero are printed as "0.". Numbers that are non-zero, but whose magnitudes are smaller than the value of ZTHRSH in the xxxxxxxx.WAD data file or the xxxxxxxx.CHD data file, are printed as ".00" or "0.00". In this manner, the format of the output helps to distinguish between regions in the aquifer: usually, fixed-point numbers show where the value is relatively large; exponential numbers show where it is smaller; and zeroes show where the value is very small.

Other output in the file includes messages telling when the chemical concentrations at the wells are changed by scheduled "events". If NFLAG(17) = 1, the cumulative masses are also printed.

```
     DATE IS 9/21/1990 - EXAMPLE DATA FOR MODEL LT3VSI V1.2 [4 chem]
     CHEM DATA FILE; ONE INJECTION WELL; ONE EXTRACTION WELL

** NOTE!  SI units are indicated, but any units  **
** can be used, as long as they are CONSISTENT.  **

     OUTPUT DATA FOR CHEMICAL SYSTEM

ZTHRSH=  0.10000-007

NFLAG(16) = 0; USING  4-POINT INTERPOLATION METHOD.
```

AT TIME T= 0.0000 (DAYS), CONCENTRATIONS OF THE
INJECTION WELL AT ( 0.100 , 0.100 ) WERE CHANGED TO:

| NUTRIENT | SUBSTRATE | OXYGEN | NITRATE | |
|---|---|---|---|---|
| 0.000 | 0.000 | 0.000 | 0.000 | (KG CHEM/KG SOLN) |

TIME T= 2.0000 (DAYS)

CUMULATIVE CHEMICAL MASSES (KG) FOR NUTRIENT

XMASS= 7.30283-004 XMFONW= 7.30081-006 XMSOUR= 0.00000
XMASIN= 8.45006-005 XMASOT= 1.05283-004

CHEMICAL CONCENTRATION DISTRIBUTION (KG/M^3) FOR NUTRIENT

PAGE 1

| | X | 0. | .1000 | .2000 | .4000 | .6000 | .8000 | 1.000 |
|---|---|---|---|---|---|---|---|---|
| Y | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0. | 1 | 0.00298 | 0.00298 | 0.00298 | 0.00298 | 0.00298 | 0.00298 | 0.00298 |
| .1000 | 2 | 1.22-004 | 2.78-004 | 7.48-004 | 0.00195 | 0.00282 | 0.00297 | 0.00298 |
| .3000 | 3 | 0.00129 | 0.00152 | 0.00218 | 0.00271 | 0.00290 | 0.00297 | 0.00299 |
| .5000 | 4 | 0.00278 | 0.00282 | 0.00293 | 0.00298 | 0.00299 | 0.00300 | 0.00299 |
| .7000 | 5 | 0.00293 | 0.00294 | 0.00294 | 0.00294 | 0.00294 | 0.00294 | 0.00299 |
| .9000 | 6 | 0.00294 | 0.00294 | 0.00294 | 0.00294 | 0.00294 | 0.00294 | 0.00300 |
| 1.100 | 7 | 0.00294 | 0.00294 | 0.00294 | 0.00294 | 0.00294 | 0.00294 | 0.00300 |
| 1.300 | 8 | 0.00294 | 0.00294 | 0.00294 | 0.00294 | 0.00294 | 0.00294 | 0.00299 |
| 1.500 | 9 | 0.00287 | 0.00287 | 0.00287 | 0.00287 | 0.00286 | 0.00286 | 0.00289 |
| 1.600 | 10 | 0.00170 | 0.00170 | 0.00170 | 0.00170 | 0.00170 | 0.00170 | 0.00170 |

PAGE 2

| | X | 1.200 | 1.300 | 1.400 |
|---|---|---|---|---|
| Y | | 8 | 9 | 10 |
| 0. | 1 | 0.00298 | 0.00298 | 0.00298 |
| .1000 | 2 | 0.00298 | 0.00298 | 0.00298 |
| .3000 | 3 | 0.00299 | 0.00299 | 0.00299 |
| .5000 | 4 | 0.00299 | 0.00299 | 0.00299 |
| .7000 | 5 | 0.00299 | 0.00299 | 0.00299 |
| .9000 | 6 | 0.00299 | 0.00299 | 0.00299 |
| 1.100 | 7 | 0.00300 | 0.00300 | 0.00300 |
| 1.300 | 8 | 0.00300 | 0.00300 | 0.00300 |
| 1.500 | 9 | 0.00292 | 0.00293 | 0.00293 |
| 1.600 | 10 | 0.00170 | 0.00170 | 0.00170 |

(Listings of chem concentrations of Substrate, Oxygen and Nitrate omitted)

DISTRIBUTION OF MICROBE POPULATION #1 (KG CELLS/KG SOIL)

PAGE 1

| | X | 0. | .1000 | .2000 | .4000 | .6000 | .8000 | 1.000 |
|---|---|---|---|---|---|---|---|---|
| Y | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0. | 1 | 5.62-007 | 5.62-007 | 5.62-007 | 5.62-007 | 5.62-007 | 5.62-007 | 5.62-007 |
| .1000 | 2 | 2.34-008 | 2.78-008 | 4.83-008 | 8.51-008 | 2.60-007 | 3.76-007 | 4.43-007 |
| .3000 | 3 | 6.44-008 | 6.96-008 | 8.67-008 | 9.68-008 | 1.30-007 | 1.87-007 | 2.73-007 |
| .5000 | 4 | 9.66-008 | 9.72-008 | 9.90-008 | 9.98-008 | 1.02-007 | 1.09-007 | 1.55-007 |
| .7000 | 5 | 9.85-008 | 9.85-008 | 9.86-008 | 9.86-008 | 9.87-008 | 9.88-008 | 1.10-007 |
| .9000 | 6 | 9.85-008 | 9.85-008 | 9.86-008 | 9.86-008 | 9.86-008 | 9.86-008 | 1.04-007 |
| 1.100 | 7 | 9.86-008 | 9.86-008 | 9.86-008 | 9.86-008 | 9.86-008 | 9.87-008 | 1.04-007 |
| 1.300 | 8 | 9.85-008 | 9.85-008 | 9.85-008 | 9.85-008 | 9.85-008 | 9.86-008 | 1.01-007 |
| 1.500 | 9 | 9.67-008 | 9.67-008 | 9.66-008 | 9.66-008 | 9.64-008 | 9.62-008 | 9.65-008 |
| 1.600 | 10 | 3.46-008 | 3.46-008 | 3.46-008 | 3.46-008 | 3.46-008 | 3.46-008 | 3.46-008 |

PAGE 2

| | X | 1.200 | 1.300 | 1.400 |
|---|---|---|---|---|
| Y | | 8 | 9 | 10 |
| 0. | 1 | 5.62-007 | 5.62-007 | 5.62-007 |
| .1000 | 2 | 4.76-007 | 4.82-007 | 4.84-007 |

```
.3000  3  3.44-007  3.62-007  3.69-007
.5000  4  2.53-007  2.83-007  2.93-007
.7000  5  2.03-007  2.31-007  2.41-007
.9000  6  1.81-007  2.00-007  2.07-007
1.100  7  1.55-007  1.66-007  1.70-007
1.300  8  1.28-007  1.34-007  1.36-007
1.500  9  1.09-007  1.12-007  1.13-007
1.600 10  3.46-008  3.46-008  3.46-008

 ... ...   (Listing of Microbe Population #2 omitted) ...     ...     ...

 ...   (Listings of chemicals & microbes for Time = 4.0 days omitted)   ...
```

## V.7. Continuation runs

Runs of the chemistry programs can take considerable amounts of computer time. It is desirable to be able to stop such runs "temporarily" and resume them later. The "continuation" feature is provided to make this possible. As will be discussed below, this feature also makes evaluation of other interesting scenarios possible.

If NFLAG(6) is set to 1 in the chemistry data file xxxxxxxx.CHD, the CHEMLOOP program will write an unformatted file xxxxxxxx.CUF containing the final chemical concentration at the end of the run. If NFLAG(5) = 1 in the xxxxxxxx.CHD file for a subsequent run of the chemistry programs, the xxxxxxxx.CUF file will be read in and the programs will use the final chemical concentration of the previous run as the starting concentration of the current run.

One use of this feature is to continue a run, if it happens that the first run didn't go far enough to produce a desired result. Another possibility is to change some of the chemical parameters and continue the run; in this case, one would use a different chemistry data file in the subsequent run or runs. It is also possible to change the hydraulic parameters, by using two or more different water data files with the same node coordinates and running the water program on each of them. A subsequent chemistry run can then use an interface file xxxxxxxx.WIF that is different from the one used in the earlier run.

The "initial" run for a sequence of chemistry runs must have NFLAG(5) = 0 and NFLAG(6) = 1 in the chemistry data file xxxxxxxx.CHD. The two differences between this run and a "normal" run with both of these NFLAGs = 0 are:

(1) After reading the base names for the water interface file xxxxxxxx.WIF and the chemistry data file, the CHEMINIT program reads a base name for the xxxxxxxx.CUF output file, which can be the same as the base names for other files, or can be different.

(2) At the end of the run, the CHEMLOOP program will write the xxxxxxxx.CUF file, which contains the final chemical concentrations in the aquifer, the final microbial populations, the final simulated time TMAX, the final chemical concentrations in the inlet and outlet tanks (CIN and COUT), and the final cumulative masses (XMASS, XMFONW, XMSOUR, XMASIN, and XMASOT). All of this information except CIN and COUT is also shown in printable form in the xxxxxxxx.CNC file.

A "continuation" run for a sequence of chemistry runs must have NFLAG(5) = 1 in the chemistry data file xxxxxxxx.CHD. It may have NFLAG(6) set to either 0 or 1. If this flag is 1, this run will write a xxxxxxxx.CUF file when it finishes, and a further continuation is possible. If NFLAG(6) = 0, no xxxxxxxx.CUF file will be written and no continuation is possible.

The three differences between a continuation run and a normal run are:

(1) After reading the base names for the interface file and the chemistry data file, the CHEMINIT program reads the base name for the xxxxxxxx.CUF file that was written by the previous chemistry run. If NFLAG(6) in the xxxxxxxx.CHD file is also set to 1, the program reads a base name for the xxxxxxxx.CUF output file, which must be different from the base name for the xxxxxxxx.CUF input file.

(2) The CHEMINIT program reads the chemistry data file xxxxxxxx.CHD as usual, until it has read the initial microbial populations. At this point, an initial run sets the initial time T0 to 0.0 and sets the cumulative masses XMASS, XMFONW, XMSOUR, XMASIN, XMASOT to 0.0. It then goes on to read the rest of the data from the xxxxxxxx.CHD file. A continuation run reads the chemical concentrations and microbial populations from the xxxxxxxx.CHD file, just like the normal run. However, because NFLAG(5) =1, it then reads the xxxxxxxx.CUF input file specified by the user. It sets CIN and COUT to the values in this file, replacing the values that were read in from the xxxxxxxx.CHD file; it sets the initial time T0 for this run equal to the TMAX from the xxxxxxxx.CUF file; it adds the TMAX read from the xxxxxxxx.CUF file to the TMAX read from the xxxxxxxx.CHD file; it sets the cumulative masses XMASS etc. to the values in the xxxxxxxx.CUF file; it sets the initial chemical concentrations to the values in the xxxxxxxx.CUF file; and it sets the initial microbial populations to the values in the xxxxxxxx.CUF file. The values that were read from the xxxxxxxx.CHD file are overwritten by the data from the xxxxxxxx.CUF file. Then the program reads the rest of the xxxxxxxx.CHD file. If an initial run and a continuation run read the same interface file and read xxxxxxxx.CHD files that are the same except for the NFLAGs, NPRT, and/or TMAX, then the final results will be the same as a single run for the total time specified by the sum of the TMAX'es in the xxxxxxxx.CHD files.

(3) If NFLAG(6) = 1, the CHEMLOOP program will write the xxxxxxxx.CUF output file at the end of the run, as described above for an initial run. This file can be used as a xxxxxxxx.CUF input file for another continuation run.

The small example problem that is included on the distribution diskette and is used as an example in this manual

includes a continuation run. The water data file is the file LT3EXAM.WAD shown in Section IV.2. The chemistry data file for the initial run is the file LT3EXAM1.CHD of Section IV.3. The name of the unformatted chemical concentration output file is LT3EXAMA.CUF. For the continuation run, the file named LT3EXAM2.CHD is used. It is like LT3EXAM1.CHD, except that NFLAG(5) is 1 instead of 0, and TMAX is 2.0 instead of 4.0. The schedule file LT3EXAM.SCH specifies that the injection well has zero concentrations for the four chemicals at time 0.0; at time 4.5 days the concentration of nutrient in the well is changed to 2.0E-4 (kg chem/kg soln), with the other chemicals remaining at zero concentration. Thus, the initial run computes for 4 days with no chemical being injected. The continuation run computes for 0.5 days with no chemical being injected. Then it computes for 1.5 days with nutrient being injected. The chemical concentrations and microbial populations are printed at 2, 4, 5, and 6 days.

The input and output files for each run are shown below. The description lists all files used, except the "debug" files xxxxxxxx.WDB and xxxxxxxx.CDB, and the files LT3VSI.CIF and LT3VSI.CLD used as interfaces between CHEMINIT and CHEMLOOP.

Water run

| Input file | Output files |
|---|---|
| LT3EXAM.WAD | LT3EXAM.WAO |
| | LT3EXAM.WPV |
| | LT3EXAM.WIF |

Initial chemistry run

NFLAG(4)=1, NFLAG(5)=0, NFLAG(6)=1, NPRT=2, TMAX=4.0

| Input files | Output files |
|---|---|
| LT3EXAM.WIF | LT3EXAM1.CHO |
| LT3EXAM1.CHD | LT3EXAM1.CNC |
| LT3EXAM.SCH | LT3EXAMA.CUF |

Time: 0 to 4 days
No chemical injected

Continuation chemistry run
NFLAG(4)=1, NFLAG(5)=1, NFLAG(6)=1, NPRT=2, TMAX=2.0

| Input files | Output files |
|---|---|
| LT3EXAM.WIF | LT3EXAM2.CHO |
| LT3EXAM2.CHD | LT3EXAM2.CNC |
| LT3EXAMA.CUF | LT3EXAMB.CUF |
| LT3EXAM.SCH | |

| Time: 4 to 6 days |
|---|
| Nutrient injected |
| 4.5 to 6 days |

Because each continuation run reads a xxxxxxxx.CHD file, it is possible to change the chemistry parameters included in this file, by creating different xxxxxxxx.CHD files for each run. To change the parameters in the xxxxxxxx.WAD file for a continuation run, one must create a new xxxxxxxx.WAD file and make another hydraulic head field run on it. When the continuation run reads the base name of the interface file, enter the base name of the new xxxxxxxx.WAD file. The various water data files in a sequence of runs must all have the same number and values of node coordinates. There may, however, be a different number and placement of wells.

To make the example continuation run, modify the batch file CHMGOON3.COM if necessary, and type the VMS command:

```
SUBMIT/NOPRINT/QUEUE=FPS264  CHMGOON3
```

This submits a job to the FPS computer that is very similar to the CHMRUN3 job. However, it reads the file LT3EXAMA.CUF that was written by the CHMRUN3 job so that it can continue the processing from the point where the previous job quit. It reads the data file LT3EXAM2.CHD and the schedule file LT3EXAM.SCH. CHMGOON3 writes a new file LT3EXAMB.CUF to allow another continuation run to be made. The output files CHMGOON3.LOG, LT3EXAM2.CDB, LT3EXAM2.CNC, and LT3EXAM2.CHO should be very similar to the files in LT3RESLT.ARC.

To make other runs of LT3VSI, it will be necessary to modify

or make new versions of the .COM files and the data files .WAD, .CHD, and .SCH. The LTPREP program can save a considerable amount of editing in creating new versions of the data files .WAD and .CHD. To run LTPREP, compile and link LTPREP.FOR with LTPREP.INC in the same subdirectory, using Microsoft Fortran Version 4.01 or later. Prepare a geometry file .GEO and abbreviated data files .WAG and .CHG. Then type LTPREP at the MSDOS prompt. The program will ask for the names of the geometry file and an abbreviated data file. Then it asks for the name of the output file. Run LTPREP twice, once with the .GEO and .WAG files as input, and .WAD as output file; then with .GEO and .CHG as input and .CHD as output. See Section IV.1 for more information on using LTPREP.

# VI. Description of the programs

LT3VSI is the LT (Long Thin aquifer) series 3 program that:

(1) Allows spatially variable hydraulic conductivity, porosity, etc.

(2) Uses the SIP method to solve the steady-state hydraulic problem.

(3) Is split into three programs, one that computes the hydraulic pressures & velocities, one that initializes the chemistry data, and one that computes the dynamic chemical concentrations.

(4) Handles four chemicals and two microbe populations.

## VI.1. Source files for LT3VSI.

Common block definitions and a few constants (parameters) are defined in include files; there are 14 include files. The Fortran code was divided into 22 separate files during development of the programs. Each of these original files is called a "module" and may contain one or more subroutines or functions. Three of the modules contain the main programs for the three phases of LT3VSI. For convenience in handling the source code on the mainframe computers, the modules were combined into three large files, one for each of the three programs.

The chart below shows the include files and Fortran modules used in the three programs of LT3VSI, Ver. 1.2. Some include files and some modules are used in more than one phase.

| | Water phase | Chem init phase | Chem loop phase |
|---|---|---|---|
| Include Files (14 distinct files) | CSIZE.SIB<br>CPROP.SIB<br>CVELOC.SIB<br>CSSIP.SIW<br>CWAT.SIW | CSIZE.SIB<br>CPROP.SIB<br>CVELOC.SIB<br>CCHEM.SIC<br>CBND.SIK<br>CCHEM.SIK<br>CHMAT.SIK<br>CPARAM.SIK<br>CRUNC.SIK<br>CVELOC.SIK | CSIZE.SIB<br>CPROP.SIB<br>CVELOC.SIB<br>CBND.SIK<br>CCHEM.SIK<br>CHMAT.SIK<br>CPARAM.SIK<br>CRUNC.SIK<br>CVELOC.SIK<br>CCHEM.SIL<br>CNRK.SIL |
| Program Name | WATER.APV | CHEMINIT.APV | CHEMLOOP.APV |
| Fortran Modules (22 distinct modules) | LT3VSIW.FOR<br>FSECOND.FOR<br>ITGRSIW.FOR<br>LUFSIW.FOR<br>OUTSIB.FOR<br>RWWSIW.FOR<br>SIPSIW.FOR<br>SUBSIB.FOR<br>WATSIW.FOR | LT3VSIC.FOR<br>BDATSIK.FOR<br>CHMSIC.FOR<br>FSECOND.FOR<br>INITSIC.FOR<br>OUTSIB.FOR<br>RATSIK.FOR<br>RWCSIC.FOR<br>SUBSIB.FOR | LT3VSIL.FOR<br>BDATSIK.FOR<br>CSUBSIL.FOR<br>FSECOND.FOR<br>LOOPSIL.FOR<br>NEWTWO.FOR<br>OUTSIB.FOR<br>PSTARSIL.FOR<br>RATSIK.FOR<br>RWCSIL.FOR<br>SUBSIB.FOR<br>TCMSIL.FOR |

On the distribution diskette, the Archive file LT3APV.ARC contains the source code for the include files and the three programs of LT3VSI. The programs are the Water program WATER.APV, the Chemistry Initialization program CHEMINIT.APF, and the Chemistry Loop program CHEMLOOP.APV. They are coded to run on the FPS Scientific Computer with a VAX/VMS front-end computer. They can be modified fairly easily to run on a VAX or on a PC (AT-class or better).

The Water program is run by itself, reading a data file whose extension is ".WAD". It computes the steady state water pressure and velocities in the aquifer described by the data file. It produces printable output files and an interface file for the

chemistry programs.

The two chemistry programs are run in tandem. The Chemistry Initialization program is run first; it reads the interface file (.WIF) produced by the water program and a chemistry data file (.CHD), and optionally a schedule file (.SCH). It produces an interface file (.CIF) as well as other files. As soon as it has finished, the Chemistry Loop program is run. It reads files produced by the Initialization program and computes the time-varying concentrations of the 4 chemicals and the populations of the 2 microbes. It produces output files that can be printed.

### VI.1.1. Source files for LTPREP.

The file LTPREP.ARC on the distribution diskette contains the source code for an include file and Fortran code for the program named LTPREP, which runs on PC's and helps prepare data files for LT3VSI. This program does not detect all of the possible data errors. See Section IV.1 for more information.

## VI.2. Running LT3VSI on other computers.

If LT3VSI is to be run on computers other than the FPS, it will be necessary to make changes in the source files before compiling them. Two kinds of changes may be needed.

(1) Source language changes. The programs are written in Standard Fortran 77, with a few exceptions. The 3 programs use "Include" statements to cause the compiler to read the "include" files that contain common block declarations. These statements may have to be changed to the form required by another Fortran compiler. If the other compiler does not have an "include" feature, then a text editor must be used to substitute the include files in place of the include statements.

Standard Fortran 77 allows only upper case letters in Fortran programs. The LT3VSI programs have lower case letters in comments and in character strings. These can be changed to

uppper case, if needed.

Standard Fortran 77 allows a maximum of 6 characters in programmer-chosen names. In the LT3VSI programs, some subroutine names are more than 6 characters long. These names can be truncated to 6 characters if needed.

(2) Run-time changes. The programs may compile without errors on another compiler, but refuse to run because of incompatibility with the other system's run-time support library. One change that will be surely be needed is to remove the feature that allows a program on the FPS computer to read a file directly from the host VAX computer. This feature is used in the WATER and CHEMINIT programs, but not in the CHEMLOOP program. In the WATER program, change the line

```
      OPEN(UNIT=1,FILE=':HOSTCHAR:' // FILBAS // '.WAD',
```

to

```
      OPEN(UNIT=1,FILE=FILBAS // '.WAD',
```

In the CHEMINIT program, change the line

```
      OPEN(UNIT=1,FILE=':HOSTCHAR:' // FILBAS // '.CHD',
```

to

```
      OPEN(UNIT=1,FILE=FILBAS // '.CHD',
```

Another change that will be needed is in the SECOND function in the FSECOND module used in all three programs. The SECOND function returns the number of seconds of CPU time used by the program, as a real number. This function is used only to display the amount of CPU time used by the programs, so it is not critical to the computations carried out. The code for SECOND contains three sections for getting the CPU time from the operating system. One section is for use under Microsoft Fortran 4.01 or later on PC, XT, AT, etc., computers; another section is for use under VMS Fortran on a VAX computer; and the third section is for use under APFTN64 on an FPS computer. In the

programs as distributed, the first two sections are "commented out"; only the FPS code is compiled. If the programs are to be run on a PC, etc., or on a VAX running VMS, all that is necessary is to "comment out" the FPS code and remove the comment flags (C in column 1) from the appropriate section of code. If the programs are to be run on other compilers or computers, it will be necessary to comment out or delete the existing sections of code and insert appropriate code. If the system to be used does not have any way for a Fortran program to obtain the CPU time, simply insert code to set SECOND to zero.

If the system to be used has different file name conventions from those used on PC's, VAX'es, and FPS'es, then all references to file names in OPEN statements will have to be changed. If the file name formats are VERY different, it may also be necessary to change some of the code.

## VI.3. Files used by LT3VSI

The tables below show the input and output files used by the three programs of LT3VSI. The notations "filbas", "wifbas", etc., are "file base names" selected and typed in by the user. The extensions "WAD", "WAO", etc., are required by the programs.

Water Phase

| File name | Unit no. | I/O | Purpose |
|---|---|---|---|
| filbas.WAD | 1 | in | Water data |
| filbas.WAO | 2 | out | Water listing |
| filbas.WPV | 3 | out | Pressure & velocity |
| filbas.WIF | 8 | out | Water-chem interface |
| filbas.WDB | 9 | out | Water debug |

Chem Initialization Phase

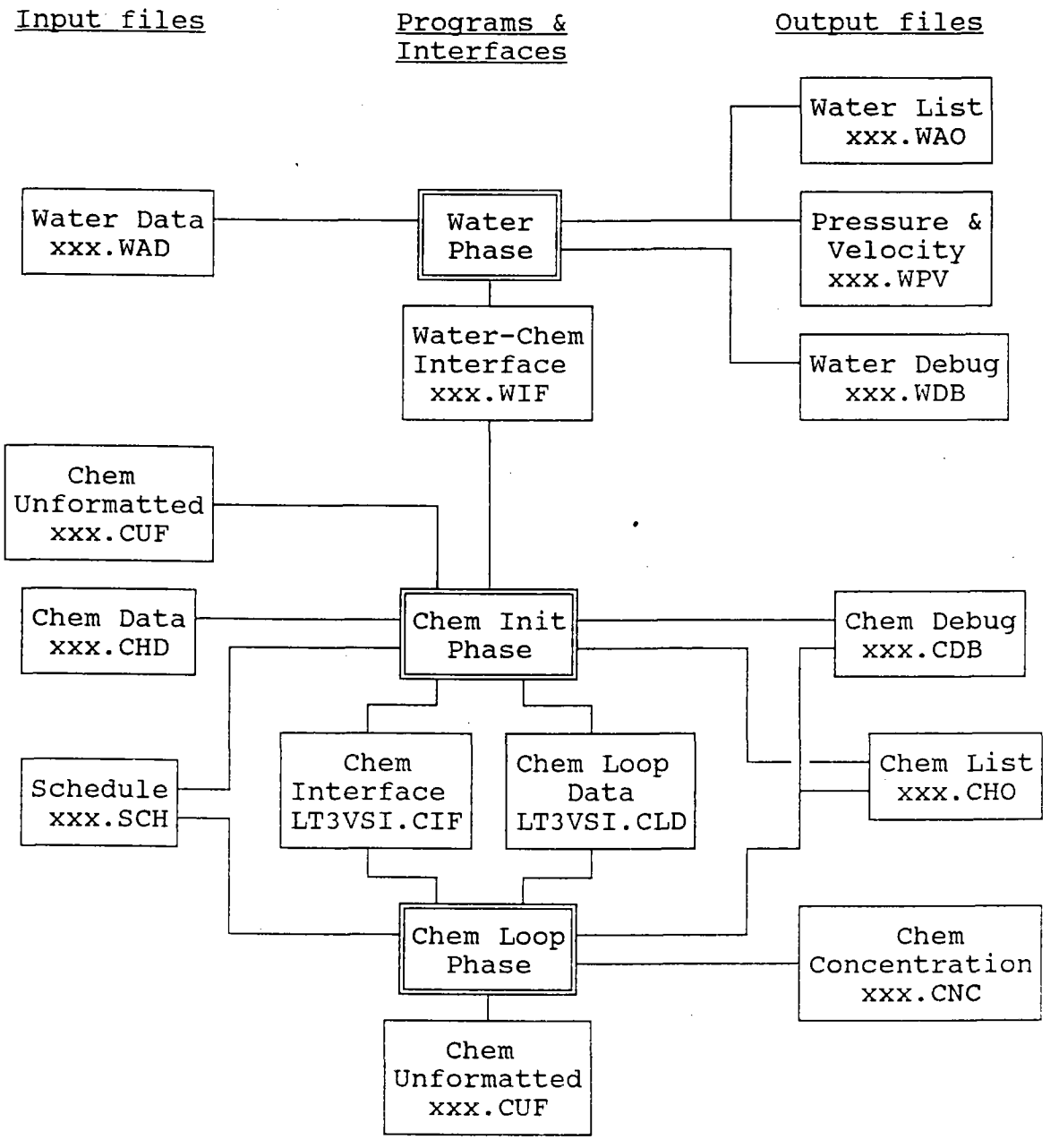
| File name | Unit no. | I/O | Purpose |
|---|---|---|---|
| filbas.CHD | 1 | in | Chemistry data |
| filbas.CHO | 2 | out | Chemistry listing |
| schbas.SCH | 4 | in | Schedule |
| wifbas.WIF | 8 | in | Water-chem interface |
| filbas.CDB | 9 | out | Chemistry debug |
| cufin .CUF | 10 | in | Chemistry unformatted |
| LT3VSI.CIF | 15 | out | Chemistry interface |
| LT3VSI.CLD | 16 | out | Chemistry loop data |

Chemistry Loop Phase

| File name | Unit no. | I/O | Purpose |
|---|---|---|---|
| LT3VSI.CLD | 1 | in | Chemistry loop data |
| filbas.CHO | 2 | out | Chemistry listing |
| filbas.CNC | 3 | out | Chemical concentrations |
| schbas.SCH | 4 | in | Schedule |
| filbas.CDB | 9 | out | Chemistry debug |
| cufout.CUF | 11 | out | Chemistry unformatted |
| LT3VSI.CIF | 15 | in | Chemistry interface |

The following diagram depicts graphically the relationships between the programs and files.

Data flow diagram for LT3VSI




VI.4. General flow diagrams of LT3VSI programs

These diagrams show the two major loops, one in the water-processing program and one in the chemistry loop program. There

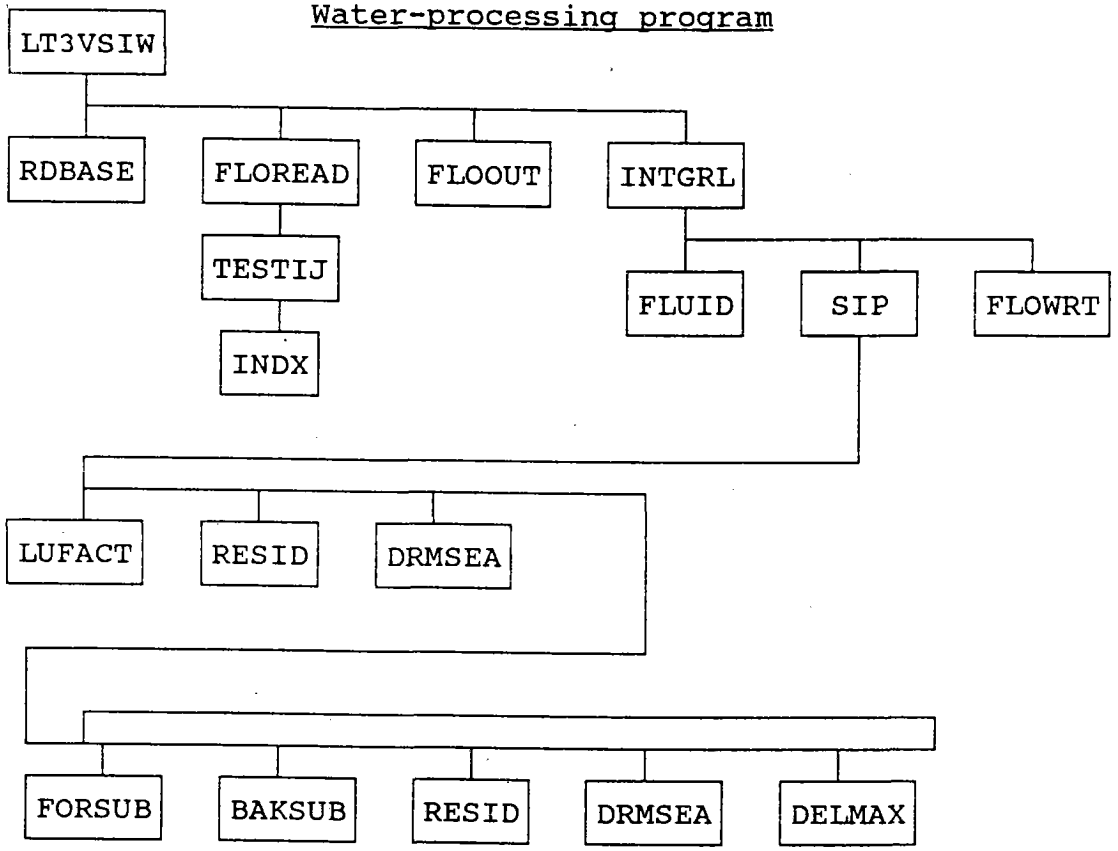
are many small loops that are not shown. When the same subroutine is called more than once in a "short" section of code, only one call is shown. Calls to the subroutines DATERR and SCHERR, to the function SECOND, and to subroutines in the module OUTSIB.FOR are not shown.

LT3VSIW is the main program for the water phase. LT3VSIC is the main program for the chemistry initialization phase. LT3VSIL is the main program for the chemistry loop phase. The other boxes represent subroutines and functions. The flow is: first downward, then to the right. Thus, LT3VSIW calls RDBASE, then FLOREAD. FLOREAD calls TESTIJ, which calls INDX. When FLOREAD returns, LT3VSIW calls FLOOUT, then INTGRL. INTGRL calls FLUID, then SIP. SIP calls LUFACT, RESID, and DRMSEA. Then it enters a loop during which it calls FORSUB, BAKSUB, RESID, DRMSEA, and DELMAX. This loop computes the steady-state hydraulic pressure field. When the loop terminates, INTGRL calls FLOWRT.

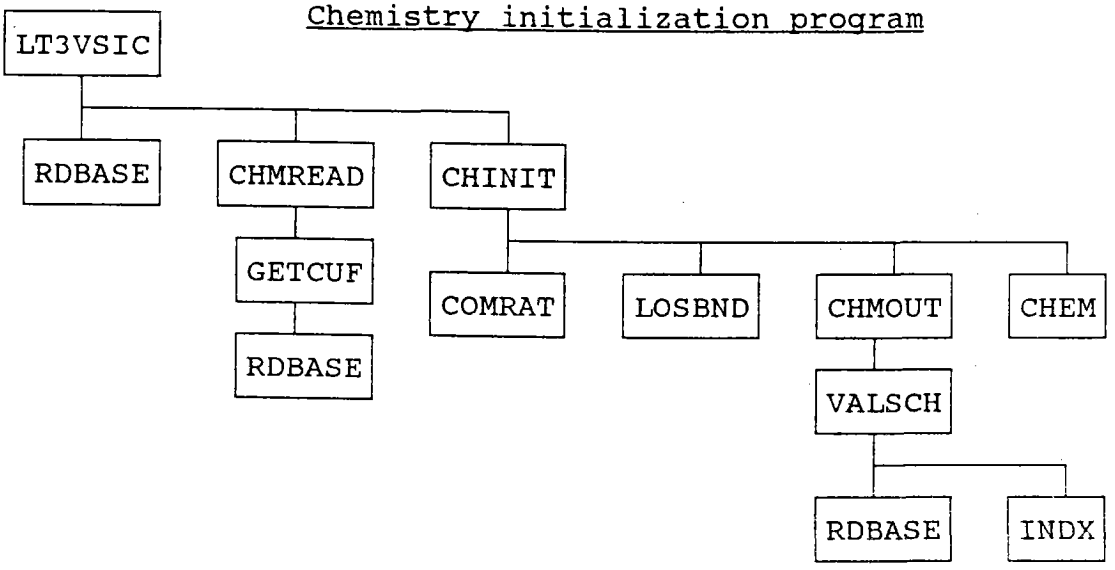
In the second diagram, LT3VSIC calls RDBASE, then CHMREAD. CHMREAD calls GETCUF, which calls RDBASE. When CHMREAD returns, LT3VSIC calls CHINIT, which calls COMRAT, etc.

In the third diagram, LT3VSIL calls LOOPIO, which calls TESTIJ, etc. Then LT3VSIL calls CHLOOP. Here a choice is made: either CHLOOP calls PREVNT, which calls INDX and COMSRC, or CHLOOP calls COMSRC. CHLOOP then enters the major loop, which computes the time-varying chemical concentrations and microbial populations. Normally, PSTAR is called once, and CHLOOP calls PREVNT, COMRAT, CLAG, MLOSS, TCMIA, and CHMWRT for each time step. However, a test for stability is made after the call to MLOSS; if necessary, delta T is reduced, COMSRC is called, and CHLOOP goes back to call PSTAR again.

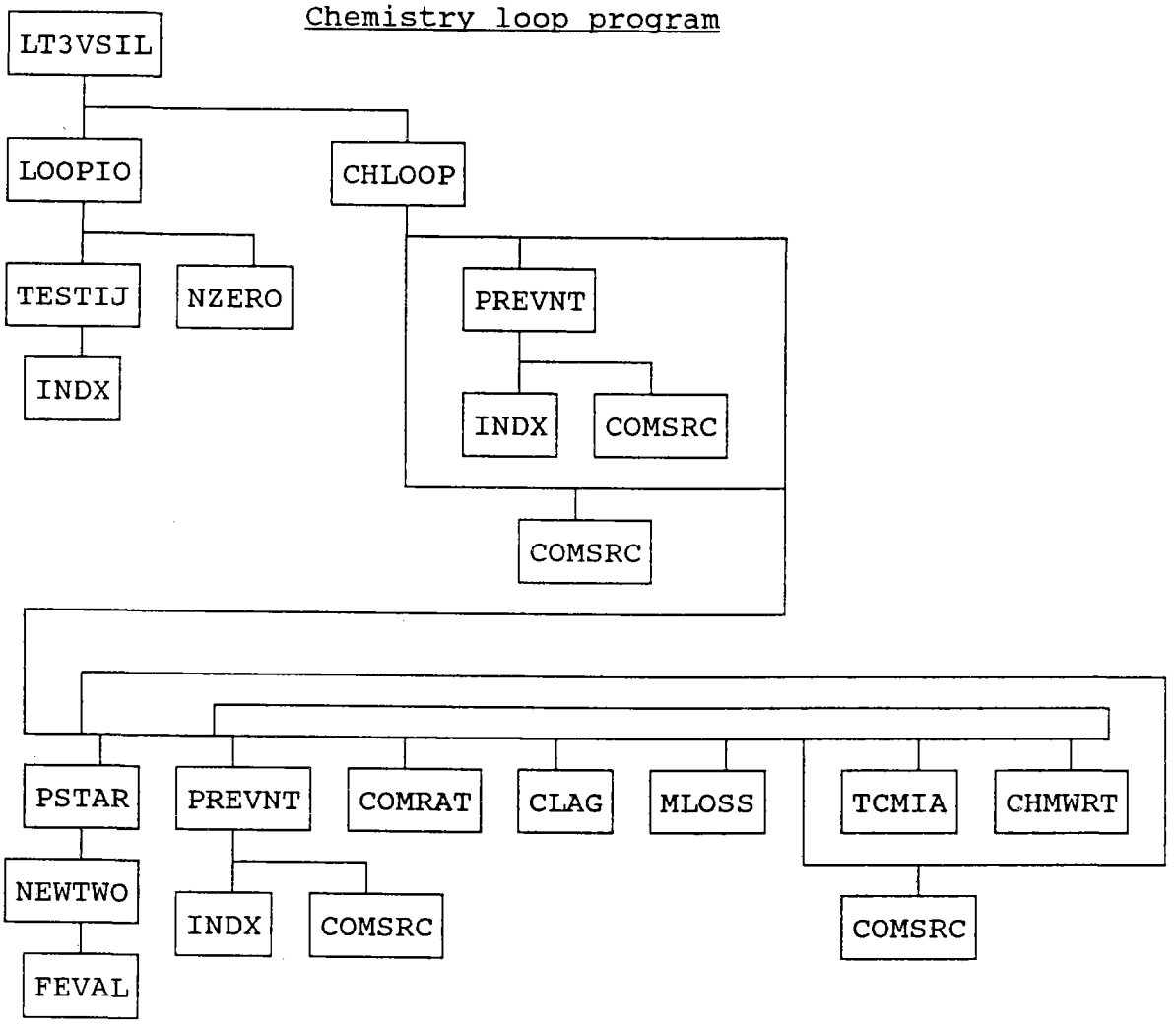
Water-processing program




Chemistry initialization program



Chemistry loop program



# REFERENCES

Lindstrom, F. T., L. Boersma, D. Myrold, and M. Barlaz. 1990. Denitrification in nonhomogeneous laboratory scale aquifers: 4. Hydraulics, nitrogen chemistry, and microbiology in a single layer. To be published by U.S. EPA.

Stone, Herbert L. 1968. Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations. Soc. Ind. Appl. Math. Journal Numer. Anal. 5, 3 (Sept. 1968) pp. 530-558.

# Appendix A. Listing of LT3VSI programs

Notice: this listing shows Version 1.3; see Section I.2 for information about the changes that were made in Version 1.2 to produce this version.

The source code for LT3VSI is on the installation diskette, in the file LT3APV.ARC. There are files named READ.ME and LT3.HLP that tell how to install and run the programs. The listings below are divided into two groups: (1) the "include" files are listed in alphabetic order; (2) the Fortran modules are listed in alphabetic order. See Section VI for more information.

Section VI.1 contains a chart showing which include files and modules are used in each of the three programs. The chart below is a brief description of the file naming convention. For example, files whose names contain the string "SIB", either in the base name or in the extension, are used in all 3 programs. Files whose names contain "SIW" are used only in the water program, and so on for the other strings. There are two exceptions to this convention: the module FSECOND.APV is used in all three programs, and the module NEWTWO.APV is used only in the chemistry loop program.

| Filename contains | File is used in WATER | CHEMINIT | CHEMLOOP |
|---|---|---|---|
| ...SIB... | X | X | X |
| ...SIW... | X | | |
| ...SIC... | | X | |
| ...SIK... | | X | X |
| ...SIL... | | | X |

The table below shows which Module contains each of the subroutines, functions, or main programs used in LT3VSI. "subr" means SUBROUTINE, "func" means FUNCTION, "d.p." means DOUBLE PRECISION, "int" means INTEGER, "logic" means LOGICAL, "real" means REAL, and "program" means PROGRAM. As mentioned above, the source listings of the Modules are in alphabetic order to make them easy to find.

| Subroutine or Function | Module |
|---|---|
| subr ARYPOL | OUTSIB.APV |
| subr ARYSTR | OUTSIB.APV |
| subr BAKSUB | SIPSIW.APV |
| subr CHEM | CHMSIC.APV |
| subr CHINIT | INITSIC.APV |
| subr CHLOOP | LOOPSIL.APV |
| subr CHMOUT | RWCSIC.APV |
| subr CHMREAD | RWCSIC.APV |
| subr CHMWRT | CSUBSIL.APV |
| d.p. func CLAG | CSUBSIL.APV |
| subr COMRAT | RATSIK.APV |
| subr COMSRC | LOOPSIL.APV |
| subr CONVTIM | FSECOND.APV |
| subr DATERR | SUBSIB.APV |
| subr DELMAX | SIPSIW.APV |
| d.p. func DRMSEA | SIPSIW.APV |
| subr FEVAL | PSTARSIL.APV |
| subr FLOOUT | RWWSIW.APV |
| subr FLOREAD | RWWSIW.APV |
| subr FLOWRT | RWWSIW.APV |
| subr FLUID | WATSIW.APV |
| subr FORSUB | SIPSIW.APV |
| subr GETCUF | LT3VSIC.APV |
| int func INDX | SUBSIB.APV |
| subr INTGRL | ITGRSIW.APV |
| subr LOOPIO | RWCSIL.APV |

| Subroutine or Function | Module |
|---|---|
| d.p. func LOSBND | INITSIC.APV |
| program LT3VSIC | LT3VSIC.APV |
| program LT3VSIL | LT3VSIL.APV |
| program LT3VSIW | LT3VSIW.APV |
| subr LUFACT | LUFSIW.APV |
| d.p. func MLOSS | LOOPSIL.APV |
| subr NEWTWO | NEWTWO.APV |
| subr NUMFIX | OUTSIB.APV |
| subr NUMSTR | OUTSIB.APV |
| logic func NZERO | RWCSIL.APV |
| subr POLAR | OUTSIB.APV |
| subr PREVNT | LOOPSIL.APV |
| subr PRIN3S | OUTSIB.APV |
| subr PRINT1 | OUTSIB.APV |
| subr PRINT2 | OUTSIB.APV |
| subr PRINT3 | OUTSIB.APV |
| subr PRINT4 | OUTSIB.APV |
| subr PSTAR | PSTARSIL.APV |
| subr RDBASE | SUBSIB.APV |
| subr RESID | SIPSIW.APV |
| subr SCHERR | LT3VSIC.APV |
| real func SECOND | FSECOND.APV |
| subr SIP | SIPSIW.APV |
| subr TCMIA | TCMSIL.APV |
| subr TESTIJ | SUBSIB.APV |
| subr VALSCH | LT3VSIC.APV |

## I N C L U D E  F I L E S

```
* Include file: CBND.SIK              Last revision: October 9, 1990
*                       For both chem init & chem loop phases of LT3VSI.
C
      DOUBLE PRECISION ECIN,      ECOUT,     EEIN,     EEOUT,
     &                 UIN1,      UIN2,      UIN3,
     &                 UOUT1,     UOUT2
      COMMON /BOUND/   ECIN(NC),  ECOUT(NC), EEIN(NC), EEOUT(NC),
     &                 UIN1,      UIN2(NC),  UIN3(NC),
     &                 UOUT1(NC), UOUT2(NC)
C
      DOUBLE PRECISION XLYIN, XLYOUT
      COMMON /GEOM3 /  XLYIN, XLYOUT
```

```
* Include file: CCHEM.SIC             Last revision: March 14, 1990
*                                      For chem init phase of LT3VSI.
```

```
C
      DOUBLE PRECISION DL0,     KSAND,     KCLAY,     KORG,
     &                 RHOSND,RHOORG,RHOCLA
      COMMON /CHEM3 /  DL0(NC),KSAND(NC),KCLAY(NC),KORG(NC),
     &                 RHOSND,RHOORG,RHOCLA
C
      DOUBLE PRECISION DCHLX,           DCHLXY
      COMMON /CHEM5 /  DCHLX(IX,IY,NC),DCHLXY(IX,IY,NC)
C
      DOUBLE PRECISION XLAMIR,            XSLMIR
      COMMON /CHEM7 /  XLAMIR(IX,IY,NC), XSLMIR(IX,IY,NC)
C
      DOUBLE PRECISION TORT,
     &                 PCTSAN,        PCTCLA,        PCTORG
      COMMON /SOIL1 /  TORT(IX,IY),
     &                 PCTSAN(IX,IY),PCTCLA(IX,IY),PCTORG(IX,IY)
C
      DOUBLE PRECISION DISPLX,           DISPLY
      COMMON /SOIL3 /  DISPLX(IX,IY,NC),DISPLY(IX,IY,NC)
```

```
* Include file: CCHEM.SIK            Last revision: October 9, 1990
*                      For both chem init & chem loop phases of LT3VSI.
C
      DOUBLE PRECISION COLD
      COMMON /CHEM1 /  COLD(IX,IY,NC)
C
      DOUBLE PRECISION DCHLY
      COMMON /CHEM6 /  DCHLY(IX,IY,NC)
C
      DOUBLE PRECISION RETARD,           LAMDA
      COMMON /CHEM10/  RETARD(IX,IY,NC),LAMDA(IX,IY,NC)
C
      DOUBLE PRECISION CIN,     COUT,     C0
      COMMON /CHEM11/  CIN(NC),COUT(NC),C0(NC)
C
      DOUBLE PRECISION EPS
      COMMON /SOIL2 /  EPS(IX,IY)
```

```
* Include file: CCHEM.SIL              Last revision: March 14, 1990
*                                      For chem loop phase of LT3VSI.
C
      DOUBLE PRECISION CNEW
      COMMON /CHEM2 /  CNEW(IX,IY,NC)
C
      DOUBLE PRECISION CSWELN,           QCHM1S
      COMMON /CHEM4 /  CSWELN(IX,IY,NC),QCHM1S(IX,IY,NC)
C
      LOGICAL          SCHED
      DOUBLE PRECISION        SOURC
      COMMON /CHEM12/  SCHED, SOURC(IX,IY,NC)
```

---

```
* Include file: CHMAT.SIK              Last revision: October 9, 1990
*                       For both chem init & chem loop phases of LT3VSI.
C
      DOUBLE PRECISION ALT1,           ALT2,           ALT3
      COMMON /CMAT1 /  ALT1(IX,IY,NC),ALT2(IX,IY,NC),ALT3(IX,IY,NC)
C
      DOUBLE PRECISION ADT1,           ADT2,           ADT3
      COMMON /CMAT2 /  ADT1(IX,IY,NC),ADT2(IX,IY,NC),ADT3(IX,IY,NC)
C
      DOUBLE PRECISION AUT1,           AUT2,           AUT3
      COMMON /CMAT3 /  AUT1(IX,IY,NC),AUT2(IX,IY,NC),AUT3(IX,IY,NC)
```

---

```
* Include file: CNRK.SIL               Last revision: April 10, 1990
*                                      For chem loop phase of LT3VSI.
C
      DOUBLE PRECISION DT, XI, YJ, COEF
      COMMON /NRK/     DT, XI, YJ, COEF(8)
```

---

```
* Include file: CPARAM.SIK             Last revision: October 9, 1990
```

```
*                     For both chem init & chem loop phases of LT3VSI.
C
      DOUBLE PRECISION ALFNI1, ALFO1,  ALFO2,
     &                 ETANI1, GAMMO1, GAMMO2,
     &                 KNI1,   KNINU1, KO1,   KO2,
     &                 KONI1,  KONU1,  KONU2,
     &                 KSNI1,  KSO1,   KSO2,  KSOM1,  KSOM2,
     &                 MUNI1,  MUO1,   MUO2,  PSIO1,  PSIO2,
     &                 RHOBD,  THENI1, YSNI1, YSO1,   YSO2
      COMMON /CPARAM/  ALFNI1, ALFO1,  ALFO2,
     &                 ETANI1, GAMMO1, GAMMO2,
     &                 KNI1,   KNINU1, KO1,   KO2,
     &                 KONI1,  KONU1,  KONU2,
     &                 KSNI1,  KSO1,   KSO2,  KSOM1,  KSOM2,
     &                 MUNI1,  MUO1,   MUO2,  PSIO1,  PSIO2,
     &                 RHOBD,  THENI1, YSNI1, YSO1,   YSO2
```

```
* Include file: CPROP.SIB           Last revision: January 31, 1990
*                                   For all three phases of LT3VSI.
C
C  PROPERTIES OF THE AQUIFER AND THE SOIL.
C
      DOUBLE PRECISION XNODE,    YNODE,    DX,    DY,
     &                 CONST1,CONST2,CONST3,CONST4,
     &                 RHOWAT, XLW
      COMMON /GEOM1 /  XNODE(IX),YNODE(IY),DX(IX),DY(IY),
     &                 CONST1,CONST2,CONST3,CONST4,
     &                 RHOWAT, XLW
C
      INTEGER          NSLXM1,NSLYM1,NSLXXX,NSLYYY,NSLXP1,NSLYP1
      COMMON /INDEX2/  NSLXM1,NSLYM1,NSLXXX,NSLYYY,NSLXP1,NSLYP1
C
      INTEGER          NINJW
      DOUBLE PRECISION       QWELIN
      COMMON /WELLS1/  NINJW,QWELIN(IX,IY)
```

```
* Include file: CRUNC.SIK           Last revision: October 9, 1990
*                     For both chem init & chem loop phases of LT3VSI.
```

```
C
      INTEGER    NC
      PARAMETER (NC=4)
C
      CHARACTER*8       CUFIN,CUFOUT,FILBAS,SCHBAS,WIFBAS,CHNAME(NC)*9
      COMMON /CHARS /   CUFIN,CUFOUT,FILBAS,SCHBAS,WIFBAS,CHNAME
C
      DOUBLE PRECISION PSTARX,           PSTARY
      COMMON /GEOM4 /  PSTARX(IX,IY,NC),PSTARY(IX,IY,NC)
C
      DOUBLE PRECISION XMASS,     XMFONW,     XMSOUR,
     &                 XMASIN,    XMASOT
      COMMON /MASSES/  XMASS(NC), XMFONW(NC),XMSOUR(NC),
     &                 XMASIN(NC),XMASOT(NC)
C
      DOUBLE PRECISION DELTA1,         DELTA2,
     &                 POP1,           POP2
      COMMON /POPS  /  DELTA1(IX,IY), DELTA2(IX,IY),
     &                 POP1(IX,IY),   POP2(IX,IY)
C
      DOUBLE PRECISION RSNINU1,         RSONU1,          RSONU2
      COMMON /RATES /  RSNINU1(IX,IY), RSONU1(IX,IY),  RSONU2(IX,IY)
C
      INTEGER          NPRT
      DOUBLE PRECISION      T0,TMAX,DT0,DT1
      COMMON /RUNCT3/  NPRT,T0,TMAX,DT0,DT1(NC)
```

---

```
* Include file: CSIZE.SIB           Last revision: August 29, 1990
*                                   For all three phases of LT3VSI.
C
      INTEGER    IX,   IY,   JX,     JY
      PARAMETER (IX=50,IY=50,JX=IX-1,JY=IY-1)
C
      INTEGER          NFLAG
      DOUBLE PRECISION           ZTHRSH
      COMMON /RUNCT1/  NFLAG(20),ZTHRSH
```

---

```
* Include file: CSSIP.SIW           Last revision: January 4, 1990
```

```
*                                    For Water phase of LT3VSI.
C
      DOUBLE PRECISION ALT2,            AUT2
      COMMON /SSIP1 /  ALT2(2:JX,2:JY),AUT2(2:JX,2:JY)
C
      DOUBLE PRECISION ADT1,            ADT3
      COMMON /SSIP2 /  ADT1(2:JX,2:JY),ADT3(2:JX,2:JY)
C
      DOUBLE PRECISION ADT2X,            ADT2Y
      COMMON /SSIP3 /  ADT2X(2:JX,2:JY),ADT2Y(2:JX,2:JY)
C
      DOUBLE PRECISION B,           C,           D
      COMMON /SSIP4 /  B(2:JX,2:JY),C(2:JX,2:JY),D(2:JX,2:JY)
C
      DOUBLE PRECISION E,           F
      COMMON /SSIP5 /  E(2:JX,2:JY),F(2:JX,2:JY)
C
      DOUBLE PRECISION XDUM,            YDUM
      COMMON /SSIP6 /  XDUM(2:JX,2:JY),YDUM(2:JX,2:JY)
C
      DOUBLE PRECISION DELX,            DELY
      COMMON /SSIP7 /  DELX(2:JX,2:JY),DELY(2:JX,2:JY)
C
      INTEGER          NLSOR,NMOD,NALPH
      DOUBLE PRECISION                   TLRNWA,TLRNWR,ALPH,ALPHAS
      COMMON /RUNCT2/  NLSOR,NMOD,NALPH,TLRNWA,TLRNWR,ALPH,ALPHAS(0:9)
```

---

```
* Include file: CVELOC.SIB           Last revision: January 31, 1990
*                                    For all three phases of LT3VSI.
C
      INTEGER          NFUNC
      DOUBLE PRECISION              VLXX,       VLYY
      COMMON /VELCT1/  NFUNC(IX,IY),VLXX(IX,IY),VLYY(IX,IY)
```

---

```
* Include file: CVELOC.SIK           Last revision: October 9, 1990
*                      For both chem init & chem loop phases of LT3VSI.
C
      DOUBLE PRECISION UX,          UY
```

```
      COMMON /VELCT2/  UX(IX,IY,NC),UY(IX,IY,NC)
```

```
* Include file: CWAT.SIW                Last revision: January 4, 1990
*                                       For water phase of LT3VSI.
C
      CHARACTER*80     HEAD1, HEAD2, FILBAS*8
      COMMON /CHARS/   HEAD1, HEAD2, FILBAS
C
      DOUBLE PRECISION HIN,HOUT,PHNEWX,        PHNEWY
      COMMON /PRESS1/  HIN,HOUT,PHNEWX(IX,IY),PHNEWY(IX,IY)
C
      DOUBLE PRECISION HOLD,        HNEW
      COMMON /STATV1/  HOLD(IX,IY),HNEW(IX,IY)
C
      DOUBLE PRECISION KSATXX,        KSATYY
      COMMON /SOIL4 /  KSATXX(IX,IY),KSATYY(IX,IY)
C
      INTEGER          NEXTW
      DOUBLE PRECISION       QWELOT
      COMMON /WELLS2/  NEXTW,QWELOT(IX,IY)
```

# F O R T R A N  M O D U L E S

```
*         File: BDATSIK.FOR          Last revision: October 9, 1990
*                        For both chem init & chem loop phases of LT3VSI.
      BLOCK DATA
C
      include 'CSIZE.SIB'
      include 'CRUNC.SIK'
C
C     DEFINE NAMES OF CHEMICALS IN CHNAME ARRAY.
C
      DATA CHNAME/'NUTRIENT','SUBSTRATE','OXYGEN','NITRATE'/
C
      END
```

```
*         File: CHMSIC.FOR           Last revision:  August 29, 1990
C                                    For chem init phase of LT3VSI.
      SUBROUTINE CHEM
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CRUNC.SIK'
      include 'CCHEM.SIC'
      include 'CCHEM.SIK'
      include 'CHMAT.SIK'
C
      INTEGER I, J, K
      DOUBLE PRECISION CONSTX, CONSTY
      DOUBLE PRECISION DXYIM, DXYIP, DXYJM, DXYJP
      DOUBLE PRECISION DXXIM, DXXIP, DYYJM, DYYJP
      DOUBLE PRECISION FACTOR
C
C     DEFINE MATRIX ELEMENTS FOR CHEMICAL FIELD TIME MARCHING MATRIX.
C
      DO 30, K=1,NC
        DO 30, I=2,NSLXXX
          CONSTX=(DX(I-1)+DX(I))/2.0D0
          DO 30, J=2,NSLYYY
            CONSTY=(DY(J-1)+DY(J))/2.0D0
C
            DXYIM=(EPS(I-1,J)*DCHLXY(I-1,J,K)+
```

```
     &              EPS(I,J)*  DCHLXY(I,J,K)  )/(8.0D0)
           DXYIP=(EPS(I,J)*  DCHLXY(I,J,K)  +
     &              EPS(I+1,J)*DCHLXY(I+1,J,K))/(8.0D0)
           DXYJM=(EPS(I,J-1)*DCHLXY(I,J-1,K)+
     &              EPS(I,J)*  DCHLXY(I,J,K)  )/(8.0D0)
           DXYJP=(EPS(I,J)*  DCHLXY(I,J,K)  +
     &              EPS(I,J+1)*DCHLXY(I,J+1,K))/(8.0D0)
C
           DXXIM=(EPS(I-1,J)*DCHLX(I-1,J,K)+
     &              EPS(I,J)*  DCHLX(I,J,K)  )/(2.0D0*DX(I-1))
           DXXIP=(EPS(I,J)*  DCHLX(I,J,K)  +
     &              EPS(I+1,J)*DCHLX(I+1,J,K))/(2.0D0*DX(I))
C
           DYYJM=(EPS(I,J-1)*DCHLY(I,J-1,K)+
     &              EPS(I,J)*  DCHLY(I,J,K)  )/(2.0D0*DY(J-1))
           DYYJP=(EPS(I,J)*  DCHLY(I,J,K)  +
     &              EPS(I,J+1)*DCHLY(I,J+1,K))/(2.0D0*DY(J))
C
           FACTOR=1.0D0/(EPS(I,J)*CONSTX*CONSTY)
C
           ALT1(I,J,K)= FACTOR*(DXYIM+DXYJM)
           ALT2(I,J,K)= FACTOR*(CONSTY*DXXIM+(DXYJM-DXYJP))
           ALT3(I,J,K)=-FACTOR*(DXYJP+DXYIM)
C
           ADT1(I,J,K)= FACTOR*(CONSTX*DYYJM+(DXYIM-DXYIP))
           ADT2(I,J,K)=-FACTOR*(CONSTY*(DXXIP+DXXIM)+
     &                          CONSTX*(DYYJP+DYYJM))
           ADT3(I,J,K)= FACTOR*(CONSTX*DYYJP+(DXYIP-DXYIM))
C
           AUT1(I,J,K)=-FACTOR*(DXYIP+DXYJM)
           AUT2(I,J,K)= FACTOR*(CONSTY*DXXIP+(DXYJP-DXYJM))
           AUT3(I,J,K)= FACTOR*(DXYIP+DXYJP)
C
   30 CONTINUE
C
      IF(NFLAG(14).EQ.0) GO TO 950
C
C+++++++++++++++++++++++++++++++++++++++++
      WRITE(9,*)
      WRITE(9,*)'Raw matrix elements'
      WRITE(9,*)
      DO 800, K=1,NC
        WRITE(9,803) K
        DO 800, I=2,NSLXXX
          DO 800, J=2,NSLYYY
            WRITE(9,802) I,J,
     &         ALT1(I,J,K),ALT2(I,J,K),ALT3(I,J,K),
     &         ADT1(I,J,K),ADT2(I,J,K),ADT3(I,J,K),
     &         AUT1(I,J,K),AUT2(I,J,K),AUT3(I,J,K)
  800 CONTINUE
```

```
  802 FORMAT(1X,2I3,9E12.6)
  803 FORMAT(1X,'K= ',I2)
      WRITE(9,*)
C+++++++++++++++++++++++++++++++++++++++++
C
  950 CONTINUE
C
      RETURN
      END
```

---

```
*         File: CSUBSIL.FOR          Last revision:  December 17, 1990
*                          Subroutines for chem loop phase of LT3VSI.
C
      SUBROUTINE CHMWRT(TIME)
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DOUBLE PRECISION TIME
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CRUNC.SIK'
      include 'CCHEM.SIL'
C
      INTEGER K
C
 2000 FORMAT(/,5X,'TIME T=',1P,G11.4,' (DAYS)',/)
 2500 FORMAT(5X,'CUMULATIVE CHEMICAL MASSES (KG)  FOR ',A,/
     &       1P,/,5X,'XMASS=  ',G12.5,2X,'XMFONW= ',G12.5,2X,
     &              'XMSOUR= ',G12.5,
     &         /,5X,'XMASIN= ',G12.5,2X,'XMASOT= ',G12.5,/)
 3000 FORMAT(10X,'CHEMICAL CONCENTRATION DISTRIBUTION (KG/M^3)  FOR ',
     &       A,/)
C
      DO 40, K=1,NC
        WRITE(3,2000) TIME
        IF (NFLAG(17).NE.0)
     &    WRITE(3,2500) CHNAME(K),XMASS(K),XMFONW(K),
     &                  XMSOUR(K),XMASIN(K),XMASOT(K)
        WRITE(3,3000) CHNAME(K)
        CALL PRINT3(3,NSLXP1,NSLYP1,CNEW(1,1,K))
   40 CONTINUE
C
 3500 FORMAT(/,10X,'DISTRIBUTION OF MICROBE POPULATION #',I1,
     &       ' (KG CELLS/KG SOIL)',/)
C
      WRITE(3,3500) 1
```

```
      CALL PRINT3(3,NSLXP1,NSLYP1,POP1)
      WRITE(3,3500) 2
      CALL PRINT3(3,NSLXP1,NSLYP1,POP2)
C
      RETURN
      END
C
C *****************************************************
C
C  CLAG function computes COLD(P*(i,j,n))
C  Coded by Gilbert A. Bachelor, Dec. 1988.
C  Modified by Gilbert A. Bachelor, Apr. 1990; Aug. 1990.
C
      DOUBLE PRECISION FUNCTION CLAG(I,J,K)
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      INTEGER I, J, K
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CVELOC.SIB'
      include 'CRUNC.SIK'
      include 'CCHEM.SIK'
      DOUBLE PRECISION ALF,BET,GAM,DEL
      DOUBLE PRECISION DDX,DDY
      DOUBLE PRECISION P1X, P2X, P3X, P4X, P5X
      DOUBLE PRECISION P1Y, P2Y, P3Y, P4Y, P5Y
      DOUBLE PRECISION X, Y
      DOUBLE PRECISION XX(5), YY(5)
      INTEGER P, Q
C
C  DEFINITIONS OF STATEMENT FUNCTIONS FOR 25-POINT
C  LAGRANGE INTERPOLATION:
C
      P1X(X)=(X-XNODE(I-1))*(X-XNODE(I))*(X-XNODE(I+1))*(X-XNODE(I+2))
      P2X(X)=(X-XNODE(I-2))*(X-XNODE(I))*(X-XNODE(I+1))*(X-XNODE(I+2))
      P3X(X)=(X-XNODE(I-2))*(X-XNODE(I-1))*(X-XNODE(I+1))*(X-XNODE(I+2))
      P4X(X)=(X-XNODE(I-2))*(X-XNODE(I-1))*(X-XNODE(I))*(X-XNODE(I+2))
      P5X(X)=(X-XNODE(I-2))*(X-XNODE(I-1))*(X-XNODE(I))*(X-XNODE(I+1))
C
      P1Y(Y)=(Y-YNODE(J-1))*(Y-YNODE(J))*(Y-YNODE(J+1))*(Y-YNODE(J+2))
      P2Y(Y)=(Y-YNODE(J-2))*(Y-YNODE(J))*(Y-YNODE(J+1))*(Y-YNODE(J+2))
      P3Y(Y)=(Y-YNODE(J-2))*(Y-YNODE(J-1))*(Y-YNODE(J+1))*(Y-YNODE(J+2))
      P4Y(Y)=(Y-YNODE(J-2))*(Y-YNODE(J-1))*(Y-YNODE(J))*(Y-YNODE(J+2))
      P5Y(Y)=(Y-YNODE(J-2))*(Y-YNODE(J-1))*(Y-YNODE(J))*(Y-Y NODE(J+1))
C
C  IF POINT (I,J) IS JUST INSIDE THE BOUNDARY, USE A 4-POINT METHOD
C  OF INTERPOLATION; OTHERWISE, USE A 25-POINT LAGRANGE INTERPOLATION
C  METHOD.  HOWEVER, IF NFLAG(16) = 0, USE THE 4-POINT METHOD FOR
C  ALL POINTS.
C
```

```
      IF (NFLAG(16).EQ.0 .OR.
     &    I.EQ.2 .OR. I.EQ.NSLXXX .OR. J.EQ.2 .OR. J.EQ.NSLYYY) THEN
C
C  COMPUTE THE COEFFICIENTS FOR THE 4-POINT METHOD, ACCORDING TO WHICH
C  QUADRANT P*(I,J) IS IN.
C
      ALF=COLD(I,J,K)
C
      GOTO (100,200,300,400), NFUNC(I,J)
      PRINT 1300,NFUNC(I,J),I,J
 1300 FORMAT(1X,'NFUNC =',I4,'at I=',I3,', J=',I3)
      STOP ' NFUNC OUT OF RANGE!'
C
C  QUADRANT 1.
C
  100 CONTINUE
      BET=(COLD(I,J,K)-COLD(I-1,J,K))/DX(I-1)
      GAM=(COLD(I,J,K)-COLD(I,J-1,K))/DY(J-1)
      DEL=(COLD(I,J,K)+COLD(I-1,J-1,K)-COLD(I-1,J,K)-COLD(I,J-1,K))/
     &    (DX(I-1)*DY(J-1))
      GOTO 500
C
C  QUADRANT 2.
C
  200 CONTINUE
      BET=(COLD(I+1,J,K)-COLD(I,J,K))/DX(I)
      GAM=(COLD(I,J,K)-COLD(I,J-1,K))/DY(J-1)
      DEL=(COLD(I,J-1,K)+COLD(I+1,J,K)-COLD(I,J,K)-COLD(I+1,J-1,K))/
     &    (DX(I)*DY(J-1))
      GOTO 500
C
C  QUADRANT 3.
C
  300 CONTINUE
      BET=(COLD(I,J,K)-COLD(I-1,J,K))/DX(I-1)
      GAM=(COLD(I,J+1,K)-COLD(I,J,K))/DY(J)
      DEL=(COLD(I-1,J,K)+COLD(I,J+1,K)-COLD(I,J,K)-COLD(I-1,J+1,K))/
     &    (DX(I-1)*DY(J))
      GOTO 500
C
C  QUADRANT 4.
C
  400 CONTINUE
      BET=(COLD(I+1,J,K)-COLD(I,J,K))/DX(I)
      GAM=(COLD(I,J+1,K)-COLD(I,J,K))/DY(J)
      DEL=(COLD(I+1,J+1,K)+COLD(I,J,K)-COLD(I+1,J,K)-COLD(I,J+1,K))/
     &    (DX(I)*DY(J))
C
C  ALL FOUR BRANCHES COME TOGETHER HERE.
C
```

```
  500 CONTINUE
C
C  NOW COMPUTE THE CLAG FUNCTION.
C
      DDX = PSTARX(I,J,K) - XNODE(I)
      DDY = PSTARY(I,J,K) - YNODE(J)
C
      CLAG = ALF + BET*DDX + GAM*DDY + DEL*DDX*DDY
C
      ELSE
C
C  COMPUTE QUOTIENTS USED IN COEFFICIENTS FOR 25-POINT METHOD.
C
      XX(1)=P1X(PSTARX(I,J,K))/P1X(XNODE(I-2))
      XX(2)=P2X(PSTARX(I,J,K))/P2X(XNODE(I-1))
      XX(3)=P3X(PSTARX(I,J,K))/P3X(XNODE(I))
      XX(4)=P4X(PSTARX(I,J,K))/P4X(XNODE(I+1))
      XX(5)=P5X(PSTARX(I,J,K))/P5X(XNODE(I+2))
C
      YY(1)=P1Y(PSTARY(I,J,K))/P1Y(YNODE(J-2))
      YY(2)=P2Y(PSTARY(I,J,K))/P2Y(YNODE(J-1))
      YY(3)=P3Y(PSTARY(I,J,K))/P3Y(YNODE(J))
      YY(4)=P4Y(PSTARY(I,J,K))/P4Y(YNODE(J+1))
      YY(5)=P5Y(PSTARY(I,J,K))/P5Y(YNODE(J+2))
C
C  COMPUTE CLAG.
C
      CLAG=0.0D0
      DO 900, P=1,5
        DO 900, Q=1,5
          CLAG=CLAG+COLD(I-3+P,J-3+Q,K)*XX(P)*YY(Q)
  900 CONTINUE
C
      ENDIF
C
      RETURN
      END
```

```
*         File:  FSECOND.FOR              Last Revision:  August 29, 1990
C                                         For all three phases of LT3VSI
      REAL FUNCTION SECOND()
*-------------------------------------------------------------------------*
*  Interface to the appropriate system timer.  At present, there is no    *
*  method of getting timing information which is uniform across the       *
*  various Fortran dialects.  This is to be remedied in Fortran '90.      *
```

```
*  This routine returns the current CPU time, in seconds.  This value  *
*  is not critical, as it is used only in an informative timing        *
*  message.  This routine must be modified to conform with whatever    *
*  system dependent clock/timer calls are available.  We have included *
*  those calls which work on several systems --                        *
*----------------------------------------------------------------------*
*  If you are using Microsoft Optimizing Fortran 4.xx + invoke the     *
*  following lines ------------     MICROSOFT     --------------------*
C**********************************************************************
C  Returns the number of seconds and hundredths of seconds elapsed
C  since midnight.                          D. E. Cawlfield, July '89.
C**********************************************************************
C     INTEGER*2 IH, IM, IS, IHU
C     REAL      START, DAY
C     SAVE      START, DAY
C     DATA START, DAY/ 2*0.0 /
C     CALL GETTIM(IH, IM, IS, IHU)
C     SECOND = 3600.*IH + 60.*IM + IS + 0.01*IHU
C     IF (SECOND+DAY+1.0 .LT. START) DAY = DAY + 86400.
C     SECOND = SECOND + DAY
C     START = SECOND
C     RETURN
C----------------------------------------------------------------------*
*  If you are using VMS Fortran on a VAX (or a MicroVax), invoke the   *
*  following lines ------------     VAX/VMS      --------------------*
**********************************************************************
*  Returns the current seconds since midnight.  The Vax call to        *
*  SECNDS(X) returns the seconds since midnight minus X.               *
**********************************************************************
C     REAL  X, SECNDS
C     X = 0.0
C     SECOND = SECNDS(X)
C     RETURN
C----------------------------------------------------------------------*
*  If you are using a Floating Point Systems Scientific Computer (FPS) *
*  invoke the following lines --------------  FPS   ------------------*
C----------------------------------------------------------------------*
**********************************************************************
*  Returns the number of current CPU seconds (the current Wall Clock   *
*  seconds are also available.          D. E. Cawlfield, July '89.      *
**********************************************************************
      REAL      CPUTIME, WCLTIME
      CALL SYS$GETTIME (CPUTIME, WCLTIME)
      SECOND = CPUTIME
      RETURN
C----------------------------------------------------------------------*
      END
C
C**********************************************
C
```

```
      SUBROUTINE CONVTIM(TOOK, IDAY, IHRS, IMIN, ISEC)
      REAL      TOOK
      INTEGER   IDAY, IHRS, IMIN, ISEC
**********************************************************************
* Conversion routine to change real TOOK into integer IDAY, IHRS, etc. *
**********************************************************************
      IDAY = TOOK / 86400.
      TOOK = MOD(TOOK, 86400.)
      IHRS = TOOK / 3600.
      TOOK = MOD(TOOK, 3600.)
      IMIN = TOOK / 60.
      ISEC = MOD(TOOK, 60.)
      RETURN
      END
```

---

```
*         File: INITSIC.FOR          Last revision:  January 30, 1991
C                                    For chem init phase of LT3VSI.
      SUBROUTINE CHINIT
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CVELOC.SIB'
      include 'CRUNC.SIK'
      include 'CBND.SIK'
      include 'CCHEM.SIC'
      include 'CCHEM.SIK'
      include 'CPARAM.SIK'
      include 'CVELOC.SIK'
C
      INTEGER I, J, K
      DOUBLE PRECISION LOSBND
      EXTERNAL LOSBND
      DOUBLE PRECISION OLDDT0, RETENT
      DOUBLE PRECISION SUIN1, SUIN2(NC), SUOUT1(NC)
      DOUBLE PRECISION TEMP
      DOUBLE PRECISION XMAG
C
C  SECTION 3: DYNAMIC CHEMICAL FIELD DISTRIBUTION.
C
      WRITE(*,*)' A dynamic modeling of the chemical field follows.'
      WRITE(*,*)
C
C  SUB SECTION 3.A: COMPUTATION OF EFFECTIVE CHEMICAL-POROUS
C          MEDIUM PROPERTIES.
```

```
C  RETARDATION AND FIRST ORDER LOSS PROCESS COEFFICIENTS.
C
      DO 302, K=1,NC
        DO 302, I=1,NSLXP1
          DO 302, J=1,NSLYP1
            RETENT=(1.0D0-EPS(I,J))/EPS(I,J)*
     &             (PCTSAN(I,J)*RHOSND*KSAND(K)+
     &              PCTCLA(I,J)*RHOCLA*KCLAY(K)+
     &              PCTORG(I,J)*RHOORG*KORG(K) )
            LAMDA(I,J,K)=RETENT*XSLMIR(I,J,K) + XLAMIR(I,J,K)
            RETARD(I,J,K)=1.0D0+RETENT
  302   CONTINUE
C
C  CHANGE VLXX AND VLYY TO REPRESENT Ux AND Uy.
C
      DO 311, I=1,NSLXP1
       DO 310, J=1,NSLYP1
          VLXX(I,J)=VLXX(I,J)/EPS(I,J)
          VLYY(I,J)=VLYY(I,J)/EPS(I,J)
  310   CONTINUE
  311 CONTINUE
C
C  DISPERSION COEFFICIENTS.
C
      DO 300, K=1,NC
        DO 300, I=2,NSLXXX
         DO 300, J=1,NSLYP1
            TEMP=TORT(I,J)*DL0(K)
            XMAG=DSQRT(VLXX(I,J)*VLXX(I,J)+VLYY(I,J)*VLYY(I,J))
            DCHLX(I,J,K)=TEMP+(DISPLY(I,J,K)*VLXX(I,J)*VLXX(I,J)+
     &                   DISPLX(I,J,K)*VLYY(I,J)*VLYY(I,J))/XMAG
            DCHLY(I,J,K)=TEMP+(DISPLX(I,J,K)*VLXX(I,J)*VLXX(I,J)+
     &                   DISPLY(I,J,K)*VLYY(I,J)*VLYY(I,J))/XMAG
            IF (NFLAG(18) .NE. 0) THEN
              DCHLXY(I,J,K)=(DISPLY(I,J,K)-DISPLX(I,J,K))*
     &                      VLXX(I,J)*VLYY(I,J)/XMAG
            ELSE
              DCHLXY(I,J,K)=0.0D0
            ENDIF
  300 CONTINUE
C
      DO 305, K=1,NC
      DO 305, J=1,NSLYP1
        DCHLX(1,J,K)=TORT(1,J)*DL0(K)
        DCHLY(1,J,K)=TORT(1,J)*DL0(K)
        DCHLXY(1,J,K)=0.0D0
        DCHLX(NSLXP1,J,K)=TORT(NSLXP1,J)*DL0(K)
        DCHLY(NSLXP1,J,K)=TORT(NSLXP1,J)*DL0(K)
        DCHLXY(NSLXP1,J,K)=0.0D0
  305 CONTINUE
```

```
C
C  COMPUTE RATES AND DELTAS.
C
      CALL COMRAT
C
      DO 320, I=1,NSLXP1
        DO 320, J=1,NSLYP1
          DELTA1(I,J)=YSO1*RSONU1(I,J) + YSNI1*RSNINU1(I,J)
          DELTA2(I,J)=YSO2*RSONU2(I,J)
  320  CONTINUE
C
C  COMPUTE DT1'S (MAXIMUM DELTA-T); STABILITY CRITERION.
C
      DO 341, K=1,NC
        DT1(K) = 0.0D0
        DO 340, I=1,NSLXXX
          DO 340, J=1,NSLYYY
            TEMP = (LAMDA(I,J,K)+LOSBND(I,J,K)+QWELIN(I,J)/
     &                                        (EPS(I,J)*RHOWAT) +
     &              2.0D0*(ABS(VLXX(I,J))/DX(I) +
     &                     ABS(VLYY(I,J))/DY(J) +
     &                     DCHLX(I,J,K)/DX(I)**2 +
     &                     DCHLY(I,J,K)/DY(J)**2 ))/RETARD(I,J,K)
            IF (TEMP .GT. DT1(K)) DT1(K) = TEMP
  340   CONTINUE
        IF (DT1(K) .NE. 0.0D0) THEN
          DT1(K) = 1.0D0 / DT1(K)
        ELSE
          WRITE(*,*) ' DT1(K) = INFINITY???'
          STOP 1
        ENDIF
  341 CONTINUE
C
C  TO ENSURE STABILITY, SET DT0 TO MINIMUM OF DT0 AND
C  DT1(K)/2 FOR K=1,...,NC.  ADJUST TMAX CORRESPONDINGLY.
C
      OLDDT0=DT0
      DO 342, K=1,NC
        IF (DT1(K)*0.5D0 .LT. DT0) DT0=DT1(K)*0.5D0
  342 CONTINUE
      TMAX=(TMAX-T0)*DT0/OLDDT0 + T0
C
C  PRINT CHEMICAL PROPERTIES ON FILE 'xxxxxxxx.CHO'
C
      CALL CHMOUT(DT0/OLDDT0)
C
C  COMPUTE Ux*(.) AND Uy*(.).
C
      DO 318, K=1,NC
        DO 318, I=1,NSLXP1
```

```
          DO 318, J=1,NSLYP1
            UX(I,J,K)=VLXX(I,J)/RETARD(I,J,K)
            UY(I,J,K)=VLYY(I,J)/RETARD(I,J,K)
  318   CONTINUE
C
      WRITE(*,*)' Setting up inlet/outlet chemical field boundary data'
      WRITE(*,*)
C
C  SUB SECTION 3.B: INLET AND OUTLET BOUNDARY DATA
C
      SUIN1 =0.0D0
      DO 350, K=1,NC
        SUIN2(K) =0.0D0
        SUOUT1(K)=0.0D0
  350 CONTINUE
C
      DO 352 I=1,NSLXXX
        SUIN1=SUIN1+
     &        DX(I)*(EPS(I,1)*VLYY(I,1)+EPS(I+1,1)*VLYY(I+1,1))/2.0D0
        DO 352, K=1,NC
          SUIN2(K)=SUIN2(K)+DX(I)*
     &           (EPS(I,1)*(DCHLY(I,1,K)/DY(1)+VLYY(I,1)) +
     &            EPS(I+1,1)*(DCHLY(I+1,1,K)/DY(1)+VLYY(I+1,1))
     &           )/2.0D0
          SUOUT1(K)=SUOUT1(K)+DX(I)*
     &           (EPS(I,NSLYP1)*
     &              (DCHLY(I,NSLYP1,K)/DY(NSLYYY)+VLYY(I,NSLYP1)) +
     &            EPS(I+1,NSLYP1)*
     &              (DCHLY(I+1,NSLYP1,K)/DY(NSLYYY)+VLYY(I+1,NSLYP1))
     &           )/2.0D0
  352 CONTINUE
C
      UIN1 =SUIN1 /XNODE(NSLXP1)
      DO 354, K=1,NC
        UIN2(K) =SUIN2(K) /XNODE(NSLXP1)
        UOUT1(K)=SUOUT1(K)/XNODE(NSLXP1)
  354 CONTINUE
C
      CALL CHEM
C
      RETURN
      END
C
C ***********************************************
C
      DOUBLE PRECISION FUNCTION LOSBND(I,J,K)
C
C  COMPUTES BOUND ON MICROBIAL LOSS MLOSS AT POINT (I,J) FOR
C  CHEMICAL K.
C
```

```
      include 'CSIZE.SIB'
      include 'CRUNC.SIK'
      include 'CCHEM.SIK'
      include 'CPARAM.SIK'
C
      INTEGER I, J, K
C
C  USE ONE OF FOUR FORMULAS TO COMPUTE LOSBND, DEPENDING ON WHICH
C  CHEMICAL IS INVOLVED.
C
C  NUTRIENT (K=1)
      IF (K .EQ. 1) THEN
        LOSBND=(RHOBD/EPS(I,J))*
     &      ((PSIO1*MUO1/(YSO1*KONU1) + THENI1*MUNI1/
     &                                  (YSNI1*KNINU1) )*POP1(I,J) +
     &        PSIO2*MUO2/(YSO2*KONU2)*POP2(I,J) )
C  SUBSTRATE (K=2)
      ELSEIF (K .EQ. 2) THEN
        LOSBND=(RHOBD/EPS(I,J))*
     &      ((MUO1/(YSO1*KSO1) + MUNI1/(YSNI1*KSNI1) )*POP1(I,J) +
     &        MUO2/(YSO2*KSO2)*POP2(I,J) )
C  OXYGEN (K=3)
      ELSEIF (K .EQ. 3) THEN
        LOSBND=(RHOBD/EPS(I,J))*
     &      ((GAMMO1*MUO1/(YSO1*KO1) + ALFO1*DELTA1(I,J)/
     &                                 KSOM1 )*POP1(I,J) +
     &       (GAMMO2*MUO2/(YSO2*KO2) + ALFO2*DELTA2(I,J)/
     &                                 KSOM2 )*POP2(I,J) )
C  NITRATE (K=4)
      ELSEIF (K .EQ. 4) THEN
        LOSBND=(RHOBD/EPS(I,J))*
     &      ((ETANI1*MUNI1/(YSNI1*KNI1) + ALFNI1*DELTA1(I,J)/
     &                                    KSNI1 )*POP1(I,J) )
C  OTHERWISE, ERROR.
      ELSE
        WRITE(*,*) ' Error in LOSBND: K=',K
        STOP 1
      ENDIF
C
      RETURN
      END
```

---

```
*         File: ITGRSIW.FOR          Last revision:  August 29, 1990
C                                    For water phase of LT3VSI.
      SUBROUTINE INTGRL
```

```
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CVELOC.SIB'
      include 'CSSIP.SIW'
      include 'CWAT.SIW'
C
      INTEGER  I, J, IALPH
      DOUBLE PRECISION CON1, CON2, CON3, CON4, CON5, CON6
      LOGICAL  DONE
C
      WRITE(*,*)' Setting up inlet and exit flow field boundary data'
      WRITE(*,*)
C
C  SECTION 1: INLET AND EXIT PORTS (BOUNDARY DATA).
C
      DO 100, I=1,NSLXP1
         HOLD(I,1)=HIN
         HOLD(I,NSLYP1)=HOUT
         HNEW(I,1)=HIN
         HNEW(I,NSLYP1)=HOUT
  100 CONTINUE
C
      WRITE(*,*)' Beginning master loop for the hydraulic field'
      WRITE(*,*)
C
C  SECTION 2: MASTER LOOP FOR THE HYDRAULIC FIELD.
C
C  SUB SECTION 2A: CALL TO SUBROUTINE FLUID.
C
      CALL FLUID
C
C  DONE IS TRUE WHEN THE METHOD HAS CONVERGED.  IALPH IS USED TO
C  SELECT THE NEXT VALUE OF ALPH FROM THE ARRAY ALPHAS.  THIS IS
C  ONLY DONE WHEN NALPH IS > 1.  THAT IS, SIP SHOULD NEVER
C  RETURN WITH DONE = FALSE IF NALPH = 1.
C
      IF(NFLAG(3).EQ.0) WRITE(*,2000)
 2000 FORMAT(1X,'ICOUNT',T14,'ALPH',T27,'RMSEA',T41,'RMSER')
      IALPH = 0
      ALPH  = ALPHAS(IALPH)
      CALL SIP(DONE)
   50 IF (.NOT. DONE) THEN
         IALPH = MOD(IALPH+1, NALPH)
         ALPH  = ALPHAS(IALPH)
         CALL SIP(DONE)
         GO TO 50
      ENDIF
C
```

```
      WRITE(*,*)
      WRITE(*,*)' Redefining hydraulic head variables'
      WRITE(*,*)
C
C  SUB SECTION 2B: REDEFINITION OF HYDRAULIC HEAD VARIABLES.
C
      DO 201, I=2,NSLXXX
       DO 200, J=2,NSLYYY
          HNEW(I,J)=XDUM(I,J)
  200  CONTINUE
  201 CONTINUE
C
      DO 210, J=2,NSLYYY
       HNEW(1,J)=CONST1*HNEW(2,J)-CONST2*HNEW(3,J)
       HNEW(NSLXP1,J)=-CONST4*HNEW(NSLXM1,J)+CONST3*HNEW(NSLXXX,J)
  210 CONTINUE
C
      WRITE(*,*)' Steady state hydraulic head calculated'
      WRITE(*,*)
C
C  SUBSECTION 2C: SELECT OTHER COMPONENTS AND/OR FIELDS FOR
C       COMPUTING.
C
      IF(NFLAG(1).EQ.0) GO TO 1000
C
      WRITE(*,*)' Calculating required hydraulic field gradients'
      WRITE(*,*)
C  SUBSECTION 2D: CALCULATE REQUIRED HYDRAULIC FIELD GRADIENTS.
C
C  TOP-CENTRAL-BOTTOM BOUNDARY NODES.
C
      DO 231, I=2,NSLXXX
C
C  TOP BOUNDARY NODES
C
       CON1=-DX(I)/(DX(I-1)*(DX(I-1)+DX(I)))
       CON2=1.0D0/DX(I-1)-1.0D0/DX(I)
       CON3=DX(I-1)/(DX(I)*(DX(I-1)+DX(I)))
       PHNEWX(I,1)=CON1*HNEW(I-1,1)+CON2*HNEW(I,1)+CON3*HNEW(I+1,1)
       PHNEWY(I,1)=(HNEW(I,2)-HNEW(I,1))/DY(1)
C
C  CENTRAL NODES.
C
       DO 230, J=2,NSLYYY
          PHNEWX(I,J)=CON1*HNEW(I-1,J)+CON2*HNEW(I,J)+CON3*
     &              HNEW(I+1,J)
          CON4=-DY(J)/(DY(J-1)*(DY(J-1)+DY(J)))
          CON5=1.0D0/DY(J-1)-1.0D0/DY(J)
          CON6=DY(J-1)/(DY(J)*(DY(J-1)+DY(J)))
          PHNEWY(I,J)=CON4*HNEW(I,J-1)+CON5*HNEW(I,J)+CON6*
```

```
     &             HNEW(I,J+1)
  230  CONTINUE
C
C  BOTTOM BOUNDARY NODES.
C
       PHNEWX(I,NSLYP1)=CON1*HNEW(I-1,NSLYP1)+CON2*HNEW(I,NSLYP1)
     &          +CON3*HNEW(I+1,NSLYP1)
       PHNEWY(I,NSLYP1)=(HNEW(I,NSLYP1)-HNEW(I,NSLYYY))/DY(NSLYYY)
  231 CONTINUE
C
      DO 240, J=1,NSLYP1
       PHNEWX(1,J)=0.0D0
       PHNEWY(1,J)=0.0D0
       PHNEWX(NSLXP1,J)=0.0D0
       PHNEWY(NSLXP1,J)=0.0D0
  240 CONTINUE
C
C  SUB SECTION 2E: CALCULATE ALL THE VELOCITY COMPONENTS
C
      WRITE(*,*)' Calculating Darcy velocity components'
      WRITE(*,*)
C
C  DARCY VELOCITY COMPONENTS.
C
      DO 251, I=1,NSLXP1
       DO 250, J=1,NSLYP1
          VLXX(I,J)=-KSATXX(I,J)*PHNEWX(I,J)
          VLYY(I,J)=-KSATYY(I,J)*PHNEWY(I,J)
  250  CONTINUE
  251 CONTINUE
C
C COMPUTE APPROPRIATE VELOCITY FIELD FLAG AT EACH INTERIOR
C NODAL POINT; DEFINES NFUNC ARRAY, AS FOLLOWS:
C     VLXX(I,J) > 0  AND VLYY(I,J) > 0 :  NFUNC(I,J) = 1
C     VLXX(I,J) < 0  AND VLYY(I,J) > 0 :  NFUNC(I,J) = 2
C     VLXX(I,J) > 0  AND VLYY(I,J) < 0 :  NFUNC(I,J) = 3
C     VLXX(I,J) < 0  AND VLYY(I,J) < 0 :  NFUNC(I,J) = 4
C
      DO 261, I=2,NSLXXX
       DO 260, J=2,NSLYYY
          NFUNC(I,J)=1
          IF (VLXX(I,J).LT.0.0D0) NFUNC(I,J)=2
          IF (VLYY(I,J).LT.0.0D0) NFUNC(I,J)=NFUNC(I,J)+2
  260  CONTINUE
  261 CONTINUE
C
C  WRITE OUT THE STEADY STATE FLOW FIELD VELOCITY COMPONENTS
C
      WRITE(*,*)' Printing the steady state flow field velocity comps.'
      WRITE(*,*)
```

```
C
 1000 CALL FLOWRT
C
      RETURN
      END
```

---

```
*          File: LOOPSIL.FOR          Last revision:  January 30, 1991
C                                     For chem loop phase of LT3VSI.
      SUBROUTINE CHLOOP
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CVELOC.SIB'
      include 'CRUNC.SIK'
      include 'CBND.SIK'
      include 'CCHEM.SIK'
      include 'CCHEM.SIL'
      include 'CHMAT.SIK'
      include 'CPARAM.SIK'
C
      DOUBLE PRECISION TINY
      PARAMETER (TINY=1.0D-60)
C
      INTEGER DIVLIM
      INTEGER I, J, K, KPRT
      INTEGER IC, ICMAX, ICMOD, ICPRT, ICUP
      DOUBLE PRECISION SUIN3(NC), SUOUT2(NC)
      DOUBLE PRECISION CTEMP, TEMP, TIME
C
      DOUBLE PRECISION CLAG, MLOSS
      EXTERNAL         CLAG, MLOSS
C
C  DEFINE CHEMICAL CONCENTRATION ALONG SIDES.
C
      DO 355, K=1,NC
        DO 355, J=2,NSLYYY
          COLD(1,J,K)=CONST1*COLD(2,J,K)-CONST2*COLD(3,J,K)
          COLD(NSLXP1,J,K)=-CONST4*COLD(NSLXM1,J,K)+
     &                          CONST3*COLD(NSLXXX,J,K)
  355 CONTINUE
C
C  SUB SECTION 3.C: MASTER LOOP FOR THE DYNAMIC CHEMICAL FIELD.
C
C  LOOP INITIALIZATION
```

```
C
      WRITE(*,*)' Beginning master loop for the dynamic chemical field'
      WRITE(*,*)
      TIME=T0
      KPRT=1
      DIVLIM=10
C
C  IC = ITERATION COUNT.
C  ICMOD = MODULUS TELLING HOW OFTEN TO DISPLAY IC & TIME.
C  ICUP = VALUE OF IC WHEN ICMOD SHOULD BE INCREASED.
C  ICMAX = NUMBER OF ITERATIONS TO BE PERFORMED.
C  ICPRT = VALUE OF IC FOR NEXT PRINT TIME.
C
      IC=1
      ICMOD=1
      ICUP=10
      ICMAX=(TMAX-T0)/DT0 + 0.5
      ICPRT=(ICMAX*KPRT + NPRT/2)/NPRT
C
      IF (SCHED) THEN
         CALL PREVNT(.TRUE.,TIME)
      ELSE
         CALL COMSRC
      ENDIF
C
C  INITIALIZATION THAT MUST BE REPEATED WHEN DT0 IS CHANGED.
C
 2900 CALL PSTAR
      DO 360, K=1,NC
        EEIN(K) =DEXP(-DT0*UIN2(K) /XLYIN)
        EEOUT(K)=DEXP(-DT0*UOUT1(K)/XLYOUT)
        ECIN(K) =1.0D0-EEIN(K)
        ECOUT(K)=1.0D0-EEOUT(K)
  360 CONTINUE
C
C  BEGIN LOOP
C
 3000 CONTINUE
C
      IF (SCHED) CALL PREVNT(.FALSE.,TIME)
C
      CALL COMRAT
C
C  SUB SECTION 3.D: COMPUTE CHEMICAL FIELD VARIABLES.
C
      DO 370, K=1,NC
        DO 370, I=2,NSLXXX
          DO 370, J=2,NSLYYY
            TEMP=(LAMDA(I,J,K)-ADT2(I,J,K)+QWELIN(I,J)/
     &                                    (RHOWAT*EPS(I,J))
```

```
     &             )*COLD(I,J,K)
               CTEMP=CLAG(I,J,K)
               TEMP=CTEMP-(TEMP+MLOSS(I,J,K))*DT0/RETARD(I,J,K)
               IF (TEMP.LT.0.0D0 .AND. NFLAG(19).EQ.0) THEN
                 DT0=DT0*0.5D0
                 WRITE(*,9800) DT0,IC,TIME,I,J,K
                 WRITE(3,9800) DT0,IC,TIME,I,J,K
 9800 FORMAT(1X,'DT0 REDUCED TO ',1P,G11.4,' AT IC =',I6,', TIME=',
     &        G11.4,/,1X,'WHERE I=',I5,' J=',I5,' K=',I5)
                 CALL COMSRC
                 DIVLIM=DIVLIM-1
                 IF (DIVLIM.LT.0) THEN
                   WRITE(*,*) 'DT0 reduced too many times!'
                   WRITE(3,*) 'DT0 reduced too many times!'
                   STOP 1
                 ENDIF
                 GOTO 2900
               ENDIF
               CNEW(I,J,K)=TEMP+(DT0/RETARD(I,J,K))*
     &           (ALT1(I,J,K)*COLD(I-1,J-1,K)+ALT2(I,J,K)*COLD(I-1,J,K)+
     &            ALT3(I,J,K)*COLD(I-1,J+1,K)+ADT1(I,J,K)*COLD(I,J-1,K)+
     &            ADT3(I,J,K)*COLD(I,J+1,K)+AUT1(I,J,K)*COLD(I+1,J-1,K)+
     &            AUT2(I,J,K)*COLD(I+1,J,K)+AUT3(I,J,K)*COLD(I+1,J+1,K)+
     &            SOURC(I,J,K) )
               IF (ABS(CNEW(I,J,K)) .LT. TINY) CNEW(I,J,K)=0.0D0
  370 CONTINUE
C
      DO 379, K=1,NC
        SUIN3(K) =0.0D0
        SUOUT2(K)=0.0D0
  379 CONTINUE
C
      DO 380, K=1,NC
        DO 380, I=1,NSLXXX
          SUIN3(K)=SUIN3(K)+DX(I)*
     &       (EPS(I,1)*(DCHLY(I,1,K)/DY(1))*COLD(I,2,K)+
     &        EPS(I+1,1)*(DCHLY(I+1,1,K)/DY(1))*COLD(I+1,2,K))/2.0D0
          SUOUT2(K)=SUOUT2(K)+DX(I)*
     &       (EPS(I,NSLYP1)*
     &          (DCHLY(I,NSLYP1,K)/DY(NSLYYY)+VLYY(I,NSLYP1))*
     &           COLD(I,NSLYYY,K)+
     &        EPS(I+1,NSLYP1)*
     &          (DCHLY(I+1,NSLYP1,K)/DY(NSLYYY)+VLYY(I+1,NSLYP1))*
     &           COLD(I+1,NSLYYY,K)
     &      )/2.0D0
  380 CONTINUE
C
      DO 381, K=1,NC
        UIN3(K) =SUIN3(K) /XNODE(NSLXP1)
        UOUT2(K)=SUOUT2(K)/XNODE(NSLXP1)
```

```
        CIN(K)= CIN(K)*EEIN(K)+((C0(K)*UIN1+UIN3(K))/UIN2(K))*ECIN(K)
        IF (ABS(CIN(K)) .LT. TINY) CIN(K)=0.0D0
        COUT(K)=COUT(K)*EEOUT(K)+(UOUT2(K)/UOUT1(K))*ECOUT(K)
        IF (ABS(COUT(K)) .LT. TINY) COUT(K)=0.0D0
  381 CONTINUE
C
      DO 388, K=1,NC
        DO 385 I=1,NSLXP1
          CNEW(I,1,K)=CIN(K)
          CNEW(I,NSLYP1,K)=COUT(K)
  385   CONTINUE
C
        DO 388, J=1,NSLYP1
          CNEW(1,J,K)=CONST1*CNEW(2,J,K)-CONST2*CNEW(3,J,K)
          IF (ABS(CNEW(1,J,K)) .LT. TINY) CNEW(1,J,K)=0.0D0
          CNEW(NSLXP1,J,K)=-CONST4*CNEW(NSLXM1,J,K)+
     &       CONST3*CNEW(NSLXXX,J,K)
          IF (ABS(CNEW(NSLXP1,J,K)) .LT. TINY) CNEW(NSLXP1,J,K)=0.0D0
  388 CONTINUE
C
C  COMPUTE NEW MICROBIAL POPULATIONS.
C
      DO 400, I=1,NSLXP1
        DO 400, J=1,NSLYP1
          POP1(I,J)=POP1(I,J)*EXP(DT0*(YSO1*RSONU1(I,J)-DELTA1(I,J)+
     &      YSNI1*RSNINU1(I,J)))
          POP2(I,J)=POP2(I,J)*EXP(DT0*(YSO2*RSONU2(I,J)-DELTA2(I,J)))
  400 CONTINUE
C
      TIME=TIME+DT0
C
      IF (NFLAG(17).NE.0) CALL TCMIA
C
      IF (MOD(IC,ICMOD) .EQ. 0) WRITE(*,9900) IC,TIME
 9900 FORMAT(1X,'IC = ',I7,' Time = ',1P,G11.4)
      IC=IC+1
      IF (IC .EQ. ICUP) THEN
        ICMOD=ICUP
        ICUP=10*ICUP
      ENDIF
C
      IF(IC.GT.ICPRT) THEN
         CALL CHMWRT(TIME)
         KPRT=KPRT+1
         ICPRT=(ICMAX*KPRT + NPRT/2)/NPRT
      ENDIF
C
      DO 410, K=1,NC
        DO 410, I=1,NSLXP1
          DO 410, J=1,NSLYP1
```

```
            COLD(I,J,K)=CNEW(I,J,K)
  410 CONTINUE
C
      IF(KPRT.LE.NPRT) GO TO 3000
C
      TMAX=TIME
      WRITE(*,9900) IC-1,TIME
      WRITE(*,*)
      WRITE(*,*)' Time t meets or exceeds TMAX! CEASE COMPUTING! '
      WRITE(*,*)
C
      RETURN
      END
C
C *********************************************
C
      SUBROUTINE COMSRC
C
C  COMPUTES ELEMENTS OF SOURC ARRAY.  CALL COMSRC AT BEGINNING OF
C  CHEMICAL PROCESSING, AND WHENEVER THE CHEMICAL CONCENTRATIONS
C  OF THE WELLS ARE CHANGED.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CRUNC.SIK'
      include 'CCHEM.SIK'
      include 'CCHEM.SIL'
C
      INTEGER I, J, K
C
      DO 20, K=1,NC
        DO 20, I=2,NSLXXX
          DO 20, J=2,NSLYYY
            SOURC(I,J,K)=(QCHM1S(I,J,K)+QWELIN(I,J)*CSWELN(I,J,K))/
     &                        EPS(I,J)
   20 CONTINUE
C
      RETURN
      END
C
C *********************************************
C
      SUBROUTINE PREVNT(FIRST,TIME)
C
C  PROCESSES EVENTS IN SCHEDULE FILE.   ALL EVENTS (IF ANY) WHOSE
C  EVENT TIMES ARE LESS THAN HALF OF DELTA-T (DT0/2) AHEAD OR
C  WITHIN HALF OF DELTA-T IN THE PAST ARE PROCESSED.  ARGUMENT
C  FIRST MUST BE .TRUE. ON THE FIRST CALL TO PREVNT, AND .FALSE.
C  ON SUBSEQUENT CALLS.  COMSRC IS CALLED IF FIRST IS TRUE ON
```

```
C  ENTRY, OR IF ANY EVENTS ARE PROCESSED.
C  THIS SUBROUTINE MUST BE CALLED *ONLY* IF THERE IS A SCHEDULE FILE.
C  NO ERROR CHECKING IS DONE; VALSCH IN THE INITIALIZATION PHASE
C  HAS ALREADY VALIDATED THE SCHEDULE FILE.
C
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CRUNC.SIK'
      include 'CCHEM.SIL'
C
      DOUBLE PRECISION CONC(NC), ETIME, XI, YJ
      SAVE             CONC,     ETIME, XI, YJ
      DOUBLE PRECISION TIME
      INTEGER I, J, K, INDX
      EXTERNAL         INDX
      LOGICAL CHANGE, FIRST
C
10000 FORMAT(/,5X,'AT TIME T= ',1P,G11.4,
     &   ' (DAYS), CONCENTRATIONS OF THE',/,5X,1P,
     &   ' INJECTION WELL AT (',G10.3,',',G10.3,') WERE CHANGED TO:',
     &   /,3X,4(3X,A9),/,5X,4(G10.3,2X),'(KG CHEM/KG SOLN)')
C
C  READ FIRST EVENT DESCRIPTION IF "FIRST" IS TRUE ON ENTRY.
C
      CHANGE=FIRST
      IF (FIRST) READ(4,*,END=99) ETIME,XI,YJ,(CONC(K),K=1,NC)
C
C  LOOP TO PROCESS ALL ELIGIBLE EVENTS.
C
   30 IF ((ETIME-TIME) .LE. (DT0*0.5D0)) THEN
         I=INDX(NSLXP1,XNODE,XI)
         J=INDX(NSLYP1,YNODE,YJ)
         DO 40, K=1,NC
           CSWELN(I,J,K)=CONC(K)
   40    CONTINUE
         WRITE(3,10000) TIME, XI, YJ, (CHNAME(K),K=1,NC),
     &     (CONC(K),K=1,NC)
         CHANGE=.TRUE.
         READ(4,*,END=99) ETIME,XI,YJ,(CONC(K),K=1,NC)
         GOTO 30
      ENDIF
      GOTO 120
C
C  AT END OF FILE, SET NEXT EVENT TIME BEYOND THE END OF RUN (TMAX).
C
   99 ETIME=TMAX+DT0
C
  120 IF (CHANGE) CALL COMSRC
C
      RETURN
```

```
      END
C
C *****************************************
C
      DOUBLE PRECISION FUNCTION MLOSS(I,J,K)
C
C  COMPUTES MICROBIAL LOSS AT POINT (I,J) FOR CHEMICAL K.
C  MATHEMATICAL FUNCTION NAME IS LAMBDA*(x,y,t).
C
      include 'CSIZE.SIB'
      include 'CRUNC.SIK'
      include 'CCHEM.SIK'
      include 'CPARAM.SIK'
C
      INTEGER I, J, K, OXY
      PARAMETER (OXY=3)
C
C  USE ONE OF FOUR FORMULAS TO COMPUTE MLOSS, DEPENDING ON WHICH
C  CHEMICAL IS INVOLVED.
C
C  NUTRIENT (K=1)
      IF (K .EQ. 1) THEN
         MLOSS=RHOBD*(
     &       (PSIO1*RSONU1(I,J)+THENI1*RSNINU1(I,J))*POP1(I,J)+
     &        PSIO2*RSONU2(I,J)*POP2(I,J) )
C  SUBSTRATE (K=2)
      ELSEIF (K .EQ. 2) THEN
         MLOSS=RHOBD*(
     &       (RSONU1(I,J)+RSNINU1(I,J))*POP1(I,J)+
     &        RSONU2(I,J)*POP2(I,J) )
C  OXYGEN (K=3)
      ELSEIF (K .EQ. 3) THEN
         MLOSS=RHOBD*(
     &       (GAMMO1*RSONU1(I,J)+ALFO1*DELTA1(I,J)*COLD(I,J,K)/
     &                                (KSOM1+COLD(I,J,K))  )*POP1(I,J)+
     &       (GAMMO2*RSONU2(I,J)+ALFO2*DELTA2(I,J)*COLD(I,J,K)/
     &                                (KSOM2+COLD(I,J,K))  )*POP2(I,J) )
C  NITRATE (K=4)
      ELSEIF (K .EQ. 4) THEN
         MLOSS=RHOBD*(
     &       (ETANI1*RSNINU1(I,J)+ALFNI1*DELTA1(I,J)*
     &                          (COLD(I,J,K)/(KNI1+COLD(I,J,K))) /
     &                     (1.0D0+COLD(I,J,OXY)/KONI1) )*POP1(I,J) )
C  OTHERWISE, ERROR.
      ELSE
         WRITE(*,*) ' Error in MLOSS: K=', K
         STOP 1
      ENDIF
C
      RETURN
```

```
      END
```

```
*         File: LT3VSIC.FOR          Last revision:  December 14, 1990
C                                    For chem init phase of LT3VSI.
      PROGRAM LT3VSIC
C
C  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  LT3VSI : Two Dimensional Water and Chemical Transport in the
C  long thin RSKERL physical aquifer.  (Version 1.3)
C      In this version, water & chemistry processing are split
C  into three programs: LT3VSIW, LT3VSIC, and LT3VSIL.  These
C  programs allow hydraulic conductivity, porosity, and other soil
C  parameters to vary in space.
C
C  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C   Nonhomogeneous and anisotropic confined aquifer - continuously
C   differentiable saturated water conductivity function - no flow
C   boundaries as shown in the figure (see diagram in header
C   comment of the water-processing program).
C
C  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C      THIS IS A MATHEMATICAL MODEL OF THE TWO-DIMENSIONAL
C  (HORIZONTAL) TRANSPORT AND FATE OF LOW WATER SOLUBILITY CHEMICALS
C  IN AN AQUIFER.
C      THESE THREE PROGRAMS ARE MODULAR IN DESIGN.  COMMON BLOCKS
C  AND SOME CONSTANTS (PARAMETERS) ARE DEFINED IN "INCLUDE" FILES.
C  INCLUDE FILES THAT ARE USED BY ALL THREE PROGRAMS HAVE FILENAMES
C  WITH THE EXTENSION "SIB".  THOSE USED ONLY BY THE WATER-PROCESSING
C  PROGRAM HAVE THE EXTENSION "SIW".  THOSE USED ONLY BY THE
C  CHEMISTRY INITIALIZATION PROGRAM HAVE THE EXTENSION "SIC".
C  THOSE USED ONLY BY THE CHEMISTRY LOOP PROGRAM HAVE THE EXTENSION
C  "SIL", AND THOSE USED BY BOTH CHEMISTRY PROGRAMS HAVE THE
C  EXTENSION "SIK".
C      THE MAJOR SECTIONS OF THIS, THE CHEMISTRY INITIALIZATION
C  PROGRAM ARE:
C     1)  PROGRAM LT3VSIC.  THE MAIN PROGRAM, WHICH DEFINES INPUT AND
C  OUTPUT UNITS, READS THE "WATER INTERFACE" FILE PRODUCED BY THE
C  WATER-PROCESSING PROGRAM, CALLS THE PROCESSING SUBROUTINES, AND
C  WRITES THE "CHEMISTRY INTERFACE" FILE.
C     2)  CHMREAD.  INPUT OF ALL CHEMICAL SYSTEM PARAMETERS.
C     3)  CHMOUT.  OUTPUT OF ALL CHEMICAL SYSTEM PARAMETERS.
C     4)  CHINIT.  INITIALIZES VARIABLES AND ARRAYS FOR COMPUTING
C  THE TIME INTEGRATION OF THE DYNAMIC CHEMICAL FIELD DISTRIBUTION.
```

```
C     5)  COMRAT.  COMPUTES THE UTILIZATION RATES.
C     6)  CHEM.  COMPUTES MATRICES USED IN SOLVING THE CHEMICAL SYSTEM.
C     7)  VALSCH.  READS AND VALIDATES THE SCHEDULE FILE.
C     8)  PRINT1, PRINT2, PRINT3, PRINT4, ARYSTR, ARYPOL, NUMSTR,
C  NUMFIX, POLAR.  SUBROUTINES TO PRODUCE VARIABLE-FORMAT OUTPUT.
C  THESE SUBROUTINES ARE USED ALL THREE PROGRAMS.
C     9)  RDBASE, TESTIJ, INDX, DATERR.  SUBROUTINES AND FUNCTIONS
C  USED BY ALL THREE PROGRAMS.
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CVELOC.SIB'
      include 'CRUNC.SIK'
      include 'CBND.SIK'
      include 'CCHEM.SIK'
      include 'CHMAT.SIK'
      include 'CPARAM.SIK'
      include 'CVELOC.SIK'
C
      INTEGER LINWID
      PARAMETER (LINWID=120)
      CHARACTER HEAD1*75, HEAD2*75, IDMESS*33, LINE*(LINWID)
      INTEGER I, J, K, IOERR, LENCIF
      REAL CIFVER, WIFVER, IWIFVR
C..The following are for Timing purposes
      INTEGER  IDAY, IHRS, IMIN, ISEC
      REAL     TIMET, SECOND
      EXTERNAL SECOND
C
C     WELCOME MESSAGE
C
      TIMET = SECOND()
      WRITE(*,*)'          TWO-DIMENSIONAL DYNAMIC CHEMICAL'
      WRITE(*,*)'          TRANSPORT AND FATE IN THE LONG THIN'
      WRITE(*,*)'          RSKERL PHYSICAL AQUIFER.'
      WRITE(*,*)'     ========================================='
      WRITE(*,*)
      WRITE(*,*)'      Models a 2 dimensional (horizontal) flow field'
     WRITE(*,*)' for a single layer porous medium.  The medium can'
     WRITE(*,*)' be anisotropic as well as nonhomogeneous.  The fluid'
     WRITE(*,*)' velocity components must have been calculated by'
     WRITE(*,*)' the water-processing program and written to an'
     WRITE(*,*)' interface file.  This program calculates the'
     WRITE(*,*)' dispersion coefficients and other variables at each'
     WRITE(*,*)' nodal point.  Chemical concentrations and microbial'
     WRITE(*,*)' populations are computed at each nodal point at'
     WRITE(*,*)' specified time intervals.  They are printed at'
     WRITE(*,*)' selected times.  A Finite Difference (space'
```

```
      WRITE(*,*)' centered) method is used.'
      WRITE(*,*)
      WRITE(*,*)' ++++++++++++++++++++++++++++++++++++++++++++++++++'
      WRITE(*,*)' G. A. Bachelor,  Sr. Systems Analyst,'
      WRITE(*,*)' D. E. Cawlfield, Sr. Systems Analyst,'
      WRITE(*,*)' F. T. Lindstrom, Assoc. Prof.,'
      WRITE(*,*)' Soil Science Dept. Oregon State Univ.,'
      WRITE(*,*)' Corvallis, OR., 97331,   (503) 737-2441'
      WRITE(*,*)' ++++++++++++++++++++++++++++++++++++++++++++++++++'
C
      WRITE(*,*) 'Enter base name of INTERFACE file, from Water run.'
      CALL RDBASE(WIFBAS,'WIF')
C
      WRITE(*,*) 'Enter base name of CHEMISTRY data file.'
      CALL RDBASE(FILBAS,'CHD')
C
      WRITE(*,*)
C
C   UNITS 2, 9, 15, AND 16 USED FOR OUTPUT, UNITS 1 AND 8 FOR INPUT
C
      WRITE(*,*) ' Input  file  1 is: ',FILBAS,'.CHD'
      WRITE(*,*) ' Output file  2 is: ',FILBAS,'.CHO'
      WRITE(*,*) ' Input  file  8 is: ',WIFBAS,'.WIF'
      WRITE(*,*) ' Output file  9 is: ',FILBAS,'.CDB'
      WRITE(*,*) ' Output file 15 is:   LT3VSI.CIF'
      WRITE(*,*) ' Output file 16 is:   LT3VSI.CLD'
      WRITE(*,*)
C
      OPEN(UNIT=1,FILE=':HOSTCHAR:' // FILBAS // '.CHD',
     &     STATUS='OLD')
      OPEN(UNIT=2,FILE=FILBAS//'.CHO',STATUS='UNKNOWN')
      OPEN(UNIT=8,FILE=WIFBAS//'.WIF',FORM='UNFORMATTED',
     &     ACCESS='SEQUENTIAL',STATUS='OLD',IOSTAT=IOERR)
      IF (IOERR .NE. 0) THEN
         WRITE(*,*) ' Trouble Re-Opening WIF file!'
         STOP  50
      ENDIF
      OPEN(UNIT=9,FILE=FILBAS//'.CDB',STATUS='UNKNOWN')
      OPEN(UNIT=16,FILE='LT3VSI.CLD',STATUS='UNKNOWN')
C
      WRITE(*,*)' Reading interface file from ',WIFBAS,'.WIF'
      WRITE(*,*)
C
      WIFVER=31.0
      READ(8) IWIFVR
      IF (WIFVER .NE. IWIFVR) THEN
         WRITE(*,*) 'Interface file is version ',IWIFVR
         WRITE(*,*) 'It should be version ',WIFVER
         STOP 1
      ENDIF
```

```
      READ(8) IDMESS
      READ(8) NSLXM1,NSLYM1,NSLXXX,NSLYYY,NSLXP1,NSLYP1,NINJW
      IF (NSLXP1.GT.IX .OR. NSLYP1.GT.IY) THEN
         WRITE(*,*) 'Size of grid in Interface file is'
         WRITE(*,*) NSLXP1,' by ',NSLYP1,', which is larger than'
         WRITE(*,*) 'the maximum dimensions ',IX,' by ',IY
         STOP 1
      ENDIF
      READ(8) NFLAG,ZTHRSH
      IF (NFLAG(1) .EQ. 0) THEN
         WRITE(*,*) 'NFLAG(1) = 0 in Interface file; hydraulic'
         WRITE(*,*) 'velocities were not computed, so chemistry'
         WRITE(*,*) 'phase cannot be run.'
         STOP 1
      ENDIF
      READ(8) CONST1,CONST2,CONST3,CONST4,RHOWAT,XLW
      READ(8) (XNODE(I),I=1,NSLXP1)
      READ(8) (YNODE(J),J=1,NSLYP1)
      READ(8) (DX(I),I=1,NSLXP1)
      READ(8) (DY(J),J=1,NSLYP1)
      DO 40, I=1,NSLXP1
         READ(8) (QWELIN(I,J),J=1,NSLYP1)
         READ(8) (NFUNC(I,J),J=1,NSLYP1)
         READ(8) (VLXX(I,J),J=1,NSLYP1)
         READ(8) (VLYY(I,J),J=1,NSLYP1)
   40 CONTINUE
      CLOSE(UNIT=8)
C
      WRITE(*,*)' Interface file successfully read in.'
      WRITE(*,*)
C
      LENCIF=MAX(NSLXP1,NSLYP1,30)*8 + 4
      OPEN(UNIT=15,FILE='LT3VSI.CIF',FORM='UNFORMATTED',
     &     ACCESS='SEQUENTIAL',STATUS='UNKNOWN',
     &     RECL=LENCIF, IOSTAT=IOERR)
      IF (IOERR .NE. 0) THEN
         WRITE(*,*) ' Trouble opening CIF file!'
         STOP 50
      ENDIF
C
      WRITE(*,*)' Calling subroutine CHMREAD'
      WRITE(*,*)
      WRITE(*,*)' Reading from data file ',FILBAS,'.CHD'
      WRITE(*,*)
C
      CALL CHMREAD(HEAD1,HEAD2)
      WRITE(*,*)' Chemical initialization data successfully read in'
      WRITE(*,*)
C
      WRITE(*,*)' Copying chem loop data to file LT3VSI.CLD'
```

```
      WRITE(*,*)
1000  FORMAT(A)
50    CONTINUE
         READ(1,1000,END=56) LINE
         DO 52, I=LINWID,2,-1
            IF (LINE(I:I) .NE. ' ') GOTO 54
52       CONTINUE
54       WRITE(16,1000) LINE(1:I)
      GOTO 50
56    WRITE(*,*)' Chem loop data copied to file LT3VSI.CLD'
      WRITE(*,*)
      CLOSE(UNIT=1)
      CLOSE(UNIT=16)
C
      WRITE(*,*)' Writing input and calculated params. to ',
     &          FILBAS,'.CHO'
      WRITE(*,*)
      CALL CHINIT
      WRITE(2,'(/)')
      CLOSE(UNIT=2)
C
      WRITE(*,*)' Writing interface file to  LT3VSI.CIF'
      WRITE(*,*)
C
      CIFVER=83.0
      WRITE(15) CIFVER
      WRITE(15) ' LT3VSI.CIF CHEM INTERFACE FILE VER. 83.0 '
      WRITE(15) NSLXM1,NSLYM1,NSLXXX,NSLYYY,NSLXP1,NSLYP1,NINJW
      WRITE(15) NFLAG,ZTHRSH
      WRITE(15) CONST1,CONST2,CONST3,CONST4,RHOWAT,XLW
      WRITE(15) (XNODE(I),I=1,NSLXP1)
      WRITE(15) (YNODE(J),J=1,NSLYP1)
      WRITE(15) (DX(I),I=1,NSLXP1)
      WRITE(15) (DY(J),J=1,NSLYP1)
      WRITE(15) CUFIN,CUFOUT,FILBAS,SCHBAS,WIFBAS
      WRITE(15) HEAD1,HEAD2
      WRITE(15) NPRT,T0,TMAX,DT0,UIN1,XLYIN,XLYOUT
      WRITE(15) ALFNI1, ALFO1,  ALFO2,
     &          ETANI1, GAMMO1, GAMMO2,
     &          KNI1,   KNINU1, KO1,   KO2
      WRITE(15) KONI1,  KONU1,  KONU2,
     &          KSNI1,  KSO1,   KSO2,  KSOM1,  KSOM2
      WRITE(15) MUNI1,  MUO1,   MUO2,  PSIO1,  PSIO2,
     &          RHOBD,  THENI1, YSNI1, YSO1,   YSO2
C
      DO 60, I=1,NSLXP1
        WRITE(15) (QWELIN(I,J),J=1,NSLYP1)
        WRITE(15) (NFUNC(I,J),J=1,NSLYP1)
        WRITE(15) (VLXX(I,J),J=1,NSLYP1)
        WRITE(15) (VLYY(I,J),J=1,NSLYP1)
```

```
      WRITE(15) (EPS(I,J),J=1,NSLYP1)
      WRITE(15) (POP1(I,J),J=1,NSLYP1)
      WRITE(15) (POP2(I,J),J=1,NSLYP1)
      WRITE(15) (DELTA1(I,J),J=1,NSLYP1)
      WRITE(15) (DELTA2(I,J),J=1,NSLYP1)
      WRITE(15) (RSNINU1(I,J),J=1,NSLYP1)
      WRITE(15) (RSONU1(I,J),J=1,NSLYP1)
      WRITE(15) (RSONU2(I,J),J=1,NSLYP1)
   60 CONTINUE
C
      DO 70, K=1,NC
        WRITE(15) DT1(K)
        WRITE(15) XMASS(K),XMFONW(K),XMSOUR(K),XMASIN(K),XMASOT(K)
        WRITE(15) ECIN(K),ECOUT(K),EEIN(K),EEOUT(K),
     &            UIN2(K),UIN3(K),UOUT1(K),UOUT2(K)
        WRITE(15) CIN(K),COUT(K),C0(K)
        DO 70, I=1,NSLXP1
          WRITE(15) (PSTARX(I,J,K),J=1,NSLYP1)
          WRITE(15) (PSTARY(I,J,K),J=1,NSLYP1)
          WRITE(15) (COLD(I,J,K),J=1,NSLYP1)
          WRITE(15) (DCHLY(I,J,K),J=1,NSLYP1)
          WRITE(15) (RETARD(I,J,K),J=1,NSLYP1)
          WRITE(15) (LAMDA(I,J,K),J=1,NSLYP1)
          WRITE(15) (ALT1(I,J,K),J=1,NSLYP1)
          WRITE(15) (ALT2(I,J,K),J=1,NSLYP1)
          WRITE(15) (ALT3(I,J,K),J=1,NSLYP1)
          WRITE(15) (ADT1(I,J,K),J=1,NSLYP1)
          WRITE(15) (ADT2(I,J,K),J=1,NSLYP1)
          WRITE(15) (ADT3(I,J,K),J=1,NSLYP1)
          WRITE(15) (AUT1(I,J,K),J=1,NSLYP1)
          WRITE(15) (AUT2(I,J,K),J=1,NSLYP1)
          WRITE(15) (AUT3(I,J,K),J=1,NSLYP1)
          WRITE(15) (UX(I,J,K),J=1,NSLYP1)
          WRITE(15) (UY(I,J,K),J=1,NSLYP1)
   70 CONTINUE
      CLOSE(UNIT=15)
C
      TIMET = SECOND() - TIMET
      CALL CONVTIM(TIMET, IDAY, IHRS, IMIN, ISEC)
      WRITE(*,2210) IDAY, IHRS, IMIN, ISEC
      WRITE(9,2210) IDAY, IHRS, IMIN, ISEC
      WRITE(9,'(/)')
      CLOSE(UNIT=9)
      WRITE(*,*)' End of chemical initialization phase.'
      STOP' Normal Fortran Termination'
C
 2210 FORMAT(/,1X,'CPU time for Chem Init was ', I2.2, ' days, ',
     &  I2.2, ' hours, ', I2.2, ' minutes, and ', I2.2, ' sec. ')
      END
C
```

```
C ****************************************
C
      SUBROUTINE GETCUF
C
C  READ BASE NAMES FOR INPUT AND/OR OUTPUT UNFORMATTED CHEMICAL
C  CONCENTRATION FILES, IF SPECIFIED BY NFLAGS.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CRUNC.SIK'
C
      INTEGER IOERR
C
      IF(NFLAG(5).NE.0) THEN
         WRITE(*,*) 'Enter base name of UNFORMATTED chem INPUT file.'
         CALL RDBASE(CUFIN,'CUF')
         WRITE(*,*) ' Input file 10 is: ',CUFIN,'.CUF'
         OPEN(UNIT=10,FILE=CUFIN//'.CUF',FORM='UNFORMATTED',
     &        ACCESS='SEQUENTIAL',STATUS='OLD',IOSTAT=IOERR)
         IF (IOERR .NE. 0) THEN
            WRITE(*,*) ' Cannot re-open input CUF file?'
            STOP 50
         ENDIF
      ENDIF
C
      IF(NFLAG(6).NE.0) THEN
         WRITE(*,*) 'Enter base name of UNFORMATTED chem OUTPUT file.'
         CALL RDBASE(CUFOUT,'CUF')
      ENDIF
C
      RETURN
      END
C
C ****************************************
C
      SUBROUTINE VALSCH
C
C  READ AND VALIDATE THE SCHEDULE FILE; PRINT LISTING ON .CHO FILE.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CRUNC.SIK'
      CHARACTER STRING*75
      DOUBLE PRECISION DATA(NC+3), ETIME, PTIME, XI, YJ
      EQUIVALENCE (ETIME,DATA(1)), (XI,DATA(2)), (YJ,DATA(3))
      INTEGER I, J, INDX, K
      EXTERNAL INDX
      LOGICAL ERR, NEG
C
```

```
1000  FORMAT(A)
1010  FORMAT(5X,A)
C
      WRITE(*,*) 'Enter base name of SCHEDULE data file.'
      CALL RDBASE(SCHBAS,'SCH')
      WRITE(*,*) ' Input  file  4 is: ',SCHBAS,'.SCH'
      OPEN(UNIT=4,FILE=SCHBAS//'.SCH',STATUS='OLD')
C
      WRITE(*,*) ' Validating the Schedule file.'
      WRITE(*,*)
C
      WRITE(2,2000)
2000  FORMAT(/,10X,'LISTING OF SCHEDULE FILE.',/)
C
      READ (4,1000) STRING
      WRITE(2,1010) STRING
      READ (4,1000) STRING
      WRITE(2,1010) STRING
C
      READ (4,1000) STRING
C
      WRITE(2,2010) (CHNAME(K),K=1,NC)
2010  FORMAT(/,T8,'ETIME',T18,'XI',T28,'YJ',T36,4(1X,A9),/)
      PTIME=0.0D0
      ERR=.FALSE.
C
C  LOOP UNTIL END-OF-FILE
C
20    CONTINUE
         READ(4,*,END=99,ERR=50) (DATA(K),K=1,NC+3)
         CALL ARYSTR(DATA,1,NC+3,1,10,STRING)
         WRITE(2,1010) STRING
         IF (ETIME.LT.PTIME)
     &      CALL SCHERR('*** Event time less than previous event time.',
     &                  STRING,ERR)
         PTIME=ETIME
         I=INDX(NSLXP1,XNODE,XI)
         J=INDX(NSLYP1,YNODE,YJ)
         IF (I.LT.1 .OR. J.LT.1)
     &      CALL SCHERR('*** Invalid coordinates (XI and/or YJ).',
     &                  STRING,ERR)
         NEG=.FALSE.
         DO 30, K=4,NC
            IF (DATA(K).LT.0.0D0) NEG=.TRUE.
30       CONTINUE
         IF (NEG)
     &      CALL SCHERR('*** Negative Concentration(s).',
     &                  STRING,ERR)
         IF (I.GE.1 .AND. J.GE.1 .AND. QWELIN(I,J).EQ.0.0D0)
     &      CALL SCHERR('*** No injection well at location (XI,YJ).',
```

```
     &                    STRING,ERR)
      GOTO 20
C
C FORMAT ERROR IN DATA.
C
50       IF (.NOT. ERR) WRITE(9,2030)
2030  FORMAT(1X,'*** Errors in Schedule File ***')
         BACKSPACE(UNIT=4)
         READ (4,1000) STRING
         WRITE(2,1010) STRING
         WRITE(9,1010) STRING
         WRITE(2,2040) NC+3
         WRITE(9,2040) NC+3
2040  FORMAT('*** Invalid data: should be ',I2,' numbers.')
         ERR=.TRUE.
      GOTO 20
C
C END OF FILE
C
99    IF (ERR) THEN
         WRITE(*,*) ' *** Errors in Schedule File.  See ',
     &      FILBAS,'.CDB file. ***'
         STOP 1
      ENDIF
C
      CLOSE(UNIT=4)
      RETURN
      END
C
C ****************************************
C
      SUBROUTINE SCHERR(MESS,STRING,ERR)
C
C  WRITE ERROR MESSAGE ON .CHO FILE (UNIT 2) AND ON .CDB FILE (UNIT 9),
C  AND SET ERR FLAG.
C
      CHARACTER MESS*(*), STRING*(*)
      LOGICAL ERR
C
      IF (.NOT. ERR) WRITE(9,4000)
4000  FORMAT(1X,'*** Errors in Schedule File ***')
      WRITE(9,4010) STRING
4010  FORMAT(5X,A)
      WRITE(9,*) MESS
      WRITE(2,*) MESS
      ERR=.TRUE.
C
      RETURN
      END
```

```
*         File: LT3VSIL.FOR          Last revision:  December 17, 1990
C                                    For chem loop phase of LT3VSI.
      PROGRAM LT3VSIL
C
C  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  LT3VSI : Two Dimensional Water and Chemical Transport in the
C  long thin RSKERL physical aquifer.  (Version 1.3)
C      In this version, water & chemistry processing are split
C  into three programs: LT3VSIW, LT3VSIC, and LT3VSIL.  These
C  programs allow hydraulic conductivity, porosity, and other soil
C  parameters to vary in space.
C
C  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C   Nonhomogeneous and anisotropic confined aquifer - continuously
C   differentiable saturated water conductivity function - no flow
C   boundaries as shown in the figure (see diagram in header
C   comment of the water-processing program).
C
C  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C      THIS IS A MATHEMATICAL MODEL OF THE TWO-DIMENSIONAL
C  (HORIZONTAL) TRANSPORT AND FATE OF LOW WATER SOLUBILITY CHEMICALS
C  IN AN AQUIFER.
C      THESE THREE PROGRAMS ARE MODULAR IN DESIGN.  COMMON BLOCKS
C  AND SOME CONSTANTS (PARAMETERS) ARE DEFINED IN "INCLUDE" FILES.
C  INCLUDE FILES THAT ARE USED BY ALL THREE PROGRAMS HAVE FILENAMES
C  WITH THE EXTENSION "SIB".  THOSE USED ONLY BY THE WATER-PROCESSING
C  PROGRAM HAVE THE EXTENSION "SIW".  THOSE USED ONLY BY THE
C  CHEMISTRY INITIALIZATION PROGRAM HAVE THE EXTENSION "SIC".
C  THOSE USED ONLY BY THE CHEMISTRY LOOP PROGRAM HAVE THE EXTENSION
C  "SIL", AND THOSE USED BY BOTH CHEMISTRY PROGRAMS HAVE THE
C  EXTENSION "SIK".
C      THE MAJOR SECTIONS OF THIS, THE CHEMISTRY LOOP PROGRAM ARE:
C     1)  PROGRAM LT3VSIL.  THE MAIN PROGRAM, WHICH DEFINES INPUT AND
C  OUTPUT UNITS, READS THE "CHEMISTRY INTERFACE" FILE PRODUCED BY THE
C  CHEMISTRY INITIALIZATION PROGRAM, AND CALLS THE CHLOOP SUBROUTINE.
C     2)  CHLOOP.  COMPUTES TIME INTEGRATION OF THE DYNAMIC CHEMICAL
C  FIELD DISTRIBUTION.
C     3)  LOOPIO.  READS DATA FOR THE CHEMICAL LOOP AND LISTS IT.
C     4)  CHMWRT.  OUTPUT OF THE DYNAMIC CHEMICAL DISTRIBUTION AT
C  SELECTED VALUES OF TIME.
C     5)  COMSRC.  COMPUTES ELEMENTS OF SOURC ARRAY.
C     6)  PREVNT.  PROCESSES "EVENTS" IN THE SCHEDULE FILE, TO CHANGE
C  CONCENTRATIONS OF CHEMICALS AT THE INJECTION WELLS.
C     7)  PSTAR.  COMPUTES COORDINATES OF P* POINTS FOR CHEMICAL
```

```
C  CONCENTRATION.
C     8)  NEWTWO.  SOLVES TWO-DIMENSIONAL NON-LINEAR SYSTEM BY
C  NEWTON-RAPHSON METHOD.
C     9)  FEVAL.  COMPUTES VALUES OF FUNCTIONS AND DERIVATIVES FOR
C  NEWTWO.
C    10)  COMRAT.  COMPUTES THE UTILIZATION RATES.
C    11)  MLOSS.  COMPUTES MICROBIAL LOSSES.
C    12)  CLAG.  A FUNCTION USED BY CHLOOP TO COMPUTE CHEMICAL
C  CONCENTRATIONS AT THE P* POINTS.
C    13)  TCMIA.  COMPUTES TOTAL CHEMICAL MASS IN AQUIFER, ETC.
C    14)  PRINT1, PRINT2, PRINT3, PRINT4, ARYSTR, ARYPOL, NUMSTR,
C  NUMFIX, POLAR.  SUBROUTINES TO PRODUCE VARIABLE-FORMAT OUTPUT.
C  THESE SUBROUTINES ARE USED BY BOTH PROGRAMS.
C    15)  RDBASE, TESTIJ, INDX, DATERR.  SUBROUTINES AND FUNCTIONS
C  USED BY ALL THREE PROGRAMS.
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CVELOC.SIB'
      include 'CRUNC.SIK'
      include 'CBND.SIK'
      include 'CCHEM.SIK'
      include 'CCHEM.SIL'
      include 'CHMAT.SIK'
      include 'CPARAM.SIK'
      include 'CVELOC.SIK'
C
      CHARACTER HEAD1*75, HEAD2*75, IDMESS*42
      INTEGER I, J, K, IOERR, LENCUF
      REAL CIFVER, CUFVER, ICIFVR
C..The following are for Timing purposes
      INTEGER  IDAY, IHRS, IMIN, ISEC
      REAL     TIMEI, TIMET, SECOND
      EXTERNAL SECOND
C
      TIMET = SECOND()
C
      WRITE(*,*)
C
C   UNIT 15 USED TO READ INTERFACE FILE
C
      WRITE(*,*) ' Input  file 15 is:   LT3VSI.CIF'
      WRITE(*,*)
C
      OPEN(UNIT=15,FILE='LT3VSI.CIF',FORM='UNFORMATTED',
     &     ACCESS='SEQUENTIAL',STATUS='OLD',IOSTAT=IOERR)
      IF (IOERR .NE. 0) THEN
         WRITE(*,*) ' Trouble Re-Opening CIF file!'
```

```
         STOP  50
      ENDIF
C
      WRITE(*,*)' Reading interface file from LT3VSI.CIF'
      WRITE(*,*)
C
      CIFVER=83.0
      READ(15) ICIFVR
      IF (CIFVER .NE. ICIFVR) THEN
         WRITE(*,*) 'Interface file is version ',ICIFVR
         WRITE(*,*) 'It should be version ',CIFVER
         STOP 1
      ENDIF
      READ(15) IDMESS
      READ(15) NSLXM1,NSLYM1,NSLXXX,NSLYYY,NSLXP1,NSLYP1,NINJW
      IF (NSLXP1.GT.IX .OR. NSLYP1.GT.IY) THEN
         WRITE(*,*) 'Size of grid in Interface file is'
         WRITE(*,*) NSLXP1,' by ',NSLYP1,', which is larger than'
         WRITE(*,*) 'the maximum dimensions ',IX,' by ',IY
         STOP 1
      ENDIF
      READ(15) NFLAG,ZTHRSH
      READ(15) CONST1,CONST2,CONST3,CONST4,RHOWAT,XLW
      READ(15) (XNODE(I),I=1,NSLXP1)
      READ(15) (YNODE(J),J=1,NSLYP1)
      READ(15) (DX(I),I=1,NSLXP1)
      READ(15) (DY(J),J=1,NSLYP1)
      READ(15) CUFIN,CUFOUT,FILBAS,SCHBAS,WIFBAS
      READ(15) HEAD1,HEAD2
      READ(15) NPRT,T0,TMAX,DT0,UIN1,XLYIN,XLYOUT
      READ(15) ALFNI1, ALFO1,  ALFO2,
     &         ETANI1, GAMMO1, GAMMO2,
     &         KNI1,   KNINU1, KO1,   KO2
      READ(15) KONI1,  KONU1,  KONU2,
     &         KSNI1,  KSO1,   KSO2,  KSOM1,  KSOM2
      READ(15) MUNI1,  MUO1,   MUO2,  PSIO1,  PSIO2,
     &         RHOBD,  THENI1, YSNI1, YSO1,   YSO2
C
      DO 60, I=1,NSLXP1
        READ(15) (QWELIN(I,J),J=1,NSLYP1)
        READ(15) (NFUNC(I,J),J=1,NSLYP1)
        READ(15) (VLXX(I,J),J=1,NSLYP1)
        READ(15) (VLYY(I,J),J=1,NSLYP1)
        READ(15) (EPS(I,J),J=1,NSLYP1)
        READ(15) (POP1(I,J),J=1,NSLYP1)
        READ(15) (POP2(I,J),J=1,NSLYP1)
        READ(15) (DELTA1(I,J),J=1,NSLYP1)
        READ(15) (DELTA2(I,J),J=1,NSLYP1)
        READ(15) (RSNINU1(I,J),J=1,NSLYP1)
        READ(15) (RSONU1(I,J),J=1,NSLYP1)
```

```
      READ(15) (RSONU2(I,J),J=1,NSLYP1)
   60 CONTINUE
C
      DO 70, K=1,NC
        READ(15) DT1(K)
        READ(15) XMASS(K),XMFONW(K),XMSOUR(K),XMASIN(K),XMASOT(K)
        READ(15) ECIN(K),ECOUT(K),EEIN(K),EEOUT(K),
     &           UIN2(K),UIN3(K),UOUT1(K),UOUT2(K)
        READ(15) CIN(K),COUT(K),C0(K)
        DO 70, I=1,NSLXP1
          READ(15) (PSTARX(I,J,K),J=1,NSLYP1)
          READ(15) (PSTARY(I,J,K),J=1,NSLYP1)
          READ(15) (COLD(I,J,K),J=1,NSLYP1)
          READ(15) (DCHLY(I,J,K),J=1,NSLYP1)
          READ(15) (RETARD(I,J,K),J=1,NSLYP1)
          READ(15) (LAMDA(I,J,K),J=1,NSLYP1)
          READ(15) (ALT1(I,J,K),J=1,NSLYP1)
          READ(15) (ALT2(I,J,K),J=1,NSLYP1)
          READ(15) (ALT3(I,J,K),J=1,NSLYP1)
          READ(15) (ADT1(I,J,K),J=1,NSLYP1)
          READ(15) (ADT2(I,J,K),J=1,NSLYP1)
          READ(15) (ADT3(I,J,K),J=1,NSLYP1)
          READ(15) (AUT1(I,J,K),J=1,NSLYP1)
          READ(15) (AUT2(I,J,K),J=1,NSLYP1)
          READ(15) (AUT3(I,J,K),J=1,NSLYP1)
          READ(15) (UX(I,J,K),J=1,NSLYP1)
          READ(15) (UY(I,J,K),J=1,NSLYP1)
   70 CONTINUE
      CLOSE(UNIT=15)
C
      WRITE(*,*)' Interface file successfully read in.'
      WRITE(*,*)
C
C   UNITS 2, 3, 9, AND 11 USED FOR OUTPUT, UNITS 1 AND 4 FOR INPUT
C
      WRITE(*,*) ' Input  file  1 is:   LT3VSI.CLD'
      OPEN(UNIT=1,FILE='LT3VSI.CLD',STATUS='OLD')
C
      WRITE(*,*) ' Output file  2 is: ',FILBAS,'.CHO'
      OPEN(UNIT=2,FILE=FILBAS//'.CHO',STATUS='OLD')
   40 CONTINUE
         READ(2,'()',END=44)
      GOTO 40
   44 BACKSPACE(UNIT=2)
C
      WRITE(*,*) ' Output file  3 is: ',FILBAS,'.CNC'
      OPEN(UNIT=3,FILE=FILBAS//'.CNC',STATUS='UNKNOWN')
C
      IF (NFLAG(4).NE.0 .AND. NINJW.NE.0) THEN
         WRITE(*,*) ' Input  file  4 is: ',SCHBAS,'.SCH'
```

```
         OPEN(UNIT=4,FILE=SCHBAS//'.SCH',STATUS='OLD')
         READ(4,'(//)')
         SCHED=.TRUE.
      ELSE
         SCHED=.FALSE.
      ENDIF
C
      WRITE(*,*) ' Output file  9 is: ',FILBAS,'.CDB'
      OPEN(UNIT=9,FILE=FILBAS//'.CDB',STATUS='OLD')
   50 CONTINUE
         READ(9,'()',END=54)
      GOTO 50
   54 BACKSPACE(UNIT=9)
C
      IF (NFLAG(6).NE.0) THEN
         WRITE(*,*) ' Output file 11 is: ',CUFOUT,'.CUF'
         LENCUF = MAX(NSLXP1,20)*8 + 4
         OPEN(UNIT=11,FILE=CUFOUT//'.CUF',FORM='UNFORMATTED',
     &         ACCESS='SEQUENTIAL',STATUS='UNKNOWN',
     &         RECL=LENCUF, IOSTAT=IOERR)
         IF (IOERR .NE. 0) THEN
            WRITE(*,*) ' Cannot open output CUF file?'
            STOP 60
         ENDIF
      ENDIF
      WRITE(*,*)
C
C  READ CHEM LOOP DATA AND PRINT IT.
C
      WRITE(*,*) ' Calling subroutine LOOPIO to read from data file'
      WRITE(*,*) ' LT3VSI.CLD and write on output file ',FILBAS,'.CHO'
      WRITE(*,*)
      CALL LOOPIO
      CLOSE(UNIT=1)
      CLOSE(UNIT=2)
      WRITE(*,*) ' Chemical loop data successfully read in'
      WRITE(*,*)
C
C  WRITE HEADINGS ON FILE 'xxxxxxxx.CNC'
C
      WRITE(3,2300) HEAD1
      WRITE(3,2300) HEAD2
 2300 FORMAT(5X,A)
      WRITE(3,98765)
98765 FORMAT(/,
     &  5X,'** NOTE!  SI units are indicated, but any units  ** ',
     &  /,5X,'** can be used, as long as they are CONSISTENT.  **')
      WRITE(3,9400)
 9400 FORMAT(/,10X,'OUTPUT DATA FOR CHEMICAL SYSTEM')
      WRITE(3,9410) ZTHRSH
```

```
 9410 FORMAT(/,5X,'ZTHRSH= ',G12.5)
      IF (NFLAG(16).EQ.0) J=4
      IF (NFLAG(16).NE.0) J=25
      WRITE(3,9500) NFLAG(16), J
 9500 FORMAT(/,5X,'NFLAG(16) =',I2,'; USING ',I2,
     &       '-POINT INTERPOLATION METHOD.')
      IF (NFLAG(17).EQ.0) WRITE(3,9600)
 9600 FORMAT(/,5X,'NFLAG(17) = 0; XMASS ET AL WILL NOT ',
     &       'BE COMPUTED OR PRINTED.')
C
C  CALL CHLOOP SUBROUTINE TO COMPUTE THE CHEMICAL CONCENTRATIONS
C  AT TIME STEPS SPECIFIED BY T0, DT0, TMAX.
C
      WRITE(*,*)' Calling CHLOOP subroutine.'
      WRITE(*,*)
C..Time integration time separately
      TIMEI = SECOND()
      CALL CHLOOP
      TIMEI = SECOND() - TIMEI
C
      WRITE(*,*)' Chemical conc. distrib. written to ',FILBAS,'.CNC'
      WRITE(*,*)
      CUFVER=132.0
      IF (NFLAG(6).NE.0) THEN
         WRITE(*,*)' Writing final chem. conc. distr. to ',
     &             CUFOUT,'.CUF'
         WRITE(11) CUFVER
         WRITE(11) NSLXP1,NSLYP1,TMAX
         DO 80, K=1,NC
            WRITE(11) CIN(K),COUT(K),XMASS(K),XMFONW(K),XMSOUR(K),
     &                XMASIN(K),XMASOT(K)
            DO 80, J=1,NSLYP1
               WRITE(11) (CNEW(I,J,K),I=1,NSLXP1)
   80    CONTINUE
         DO 90, J=1,NSLYP1
            WRITE(11) (POP1(I,J),I=1,NSLXP1)
            WRITE(11) (POP2(I,J),I=1,NSLXP1)
   90    CONTINUE
         WRITE(*,*)
      ENDIF
C
      TIMET = SECOND() - TIMET
      CALL CONVTIM(TIMEI, IDAY, IHRS, IMIN, ISEC)
      WRITE(*,2200) IDAY, IHRS, IMIN, ISEC
      WRITE(9,2200) IDAY, IHRS, IMIN, ISEC
      CALL CONVTIM(TIMET, IDAY, IHRS, IMIN, ISEC)
      WRITE(*,2210) IDAY, IHRS, IMIN, ISEC
      WRITE(9,2210) IDAY, IHRS, IMIN, ISEC
      WRITE(*,*)' End of this simulation run.'
      STOP' Normal Fortran Termination'
```

```
C
 2200 FORMAT(/,1X,'(Integration time was ', I2.2, ' days, ',
     &  I2.2, ' hours, ', I2.2, ' minutes, and ', I2.2, ' sec.)')
 2210 FORMAT(/,1X,'CPU time for Chem Loop was ', I2.2, ' days, ',
     &  I2.2, ' hours, ', I2.2, ' minutes, and ', I2.2, ' sec.')
      END
```

---

```
*         File: LT3VSIW.FOR          Last revision:  December 17, 1990
C                                    For water phase of LT3VSI.
      PROGRAM LT3VSIW
C
C  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  LT3VSI : Two Dimensional Water and Chemical Transport in the
C  long thin RSKERL physical aquifer.  (Version 1.3)
C      In this version, water & chemistry processing are split
C  into three programs: LT3VSIW, LT3VSIC, and LT3VSIL.  These
C  programs allow hydraulic conductivity, porosity, and other soil
C  parameters to vary in space.  This version uses the Strongly
C  Implicit (SIP) method to solve the steady-state hydraulic
C  problem.
C
C  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C   Nonhomogeneous and anisotropic confined aquifer - continuously
C   differentiable saturated water conductivity function - no flow
C   boundaries as shown in the figure:
C
C  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C              no flux (right hand) boundary
C     x=Lx  ____________________________________________________
C          .                                                    .
C          .                                                    .
C          .                                                    .
C          .          injection wells and extraction            .
C          .             wells (if any present)                 .
C          .                                                    .
C   prescribed                                                  prescribed
C   entrance             high permeability layer                exit
C   hydraulic                                                   hydraulic
C   head   .                                                    head
C          .                                                     .
C          .                                                     .
C          .                                                     .
C          .                                                     .
```

```
C    x=0  .-------------------------------------------------------.
C           no flux (left hand) boundary
C         |                                                       |
C        y=0                                                    y=Ly
C
C
C  ========================================================
C  Finite difference- linear equilibrium sorption.
C  ========================================================
C
C       THIS IS A MATHEMATICAL MODEL OF THE TWO-DIMENSIONAL
C  (HORIZONTAL) TRANSPORT AND FATE OF LOW WATER SOLUBILITY CHEMICALS
C  IN AN AQUIFER.
C       THESE THREE PROGRAMS ARE MODULAR IN DESIGN.  COMMON BLOCKS
C  AND SOME CONSTANTS (PARAMETERS) ARE DEFINED IN "INCLUDE" FILES.
C  INCLUDE FILES THAT ARE USED BY ALL THREE PROGRAMS HAVE FILENAMES
C  WITH THE EXTENSION "SIB".  THOSE USED ONLY BY THE WATER-PROCESSING
C  PROGRAM HAVE THE EXTENSION "SIW", AND THOSE USED ONLY BY THE
C  CHEMISTRY-PROCESSING PROGRAMS HAVE EXTENSIONS "SIC", "SIK", AND
C  "SIL".
C       THE MAJOR SECTIONS OF THIS, THE WATER-PROCESSING PROGRAM ARE:
C     1)  PROGRAM LT3VSIW.  THE MAIN PROGRAM, WHICH DEFINES THE INPUT
C  AND OUTPUT FILES, CALLS THE PROCESSING SUBROUTINES, AND WRITES
C  THE "WATER INTERFACE" FILE WHICH PASSES INFORMATION TO THE
C  CHEMISTRY-PROCESSING PROGRAMS.
C     2)  FLOREAD.  INPUT OF ALL FLOW SYSTEM PARAMETERS, VARIABLE
C  INITIALIZATION, ETC...
C     3)  FLOOUT.  OUTPUT OF ALL FLOW SYSTEM PARAMETERS.
C     4)  FLOWRT.  OUTPUT OF PRESSURE FIELD; ALSO VELOCITY COMPONENTS
C  IF REQUESTED.
C     5)  INTGRL.  ORGANIZES THE SYSTEM FOR SOLVING THE STEADY STATE
C  FLOW SYSTEM.  COMPUTES VELOCITY COMPONENTS IF REQUESTED.
C     6)  FLUID.  DEFINES ELEMENTS OF WATER PRESSURE FIELD MATRIX,
C  AND CALCULATES THE "KNOWN" VECTOR FOR THE WATER PRESSURE
C  DISTRIBUTION.
C     7)  SIP.  SOLVES WATER PRESSURE HEAD SYSTEM IN THE STEADY
C  STATE, USING THE STRONGLY IMPLICIT METHOD (SIP).
C     8)  LUFACT.  PERFORMS AN INCOMPLETE LU FACTORIZATION.
C     9)  RESID, DRMSEA, FORSUB, BAKSUB, DELMAX.  SUBROUTINES AND
C  FUNCTIONS USED BY SIP.
C    10)  PRINT1, PRINT2, PRINT3, PRIN3S, PRINT4, ARYSTR, ARYPOL,
C  NUMSTR, NUMFIX, POLAR.  SUBROUTINES TO PRODUCE VARIABLE-FORMAT
C  OUTPUT.  THESE SUBROUTINES ARE USED BY ALL THREE PROGRAMS.
C    11)  RDBASE, TESTIJ, INDX, DATERR.  SUBROUTINES AND FUNCTIONS
C  USED BY ALL THREE PROGRAMS.
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
```

```
      include 'CVELOC.SIB'
      include 'CWAT.SIW'
C
      INTEGER I, J, LENWIF, WIFERR
      REAL WIFVER
C..The following variables are for Timing purposes . . .
      INTEGER  IDAY, IHRS, IMIN, ISEC
      REAL     TIMEI, TIMET, SECOND
      EXTERNAL SECOND
C
C     WELCOME MESSAGE
C
      TIMET = SECOND()
      WRITE(*,*)'         TWO-DIMENSIONAL STEADY WATER FLOW IN'
      WRITE(*,*)'         THE LONG THIN RSKERL PHYSICAL AQUIFER.'
      WRITE(*,*)'     ========================================='
      WRITE(*,*)
      WRITE(*,*)'      Models a 2 dimensional (horizontal) flow field'
      WRITE(*,*)' for a single layer porous medium.  The medium can'
      WRITE(*,*)' be anisotropic as well as nonhomogeneous.  Pressure '
      WRITE(*,*)' and velocity components are calculated at each '
      WRITE(*,*)' nodal point.  The flow field is confined to '
      WRITE(*,*)' the interior of the rectangular boundaries.  '
      WRITE(*,*)' A Finite Difference (space centered) method'
      WRITE(*,*)' is used.  An interface file is written which is'
      WRITE(*,*)' used to pass information to the chemistry-'
      WRITE(*,*)' processing programs.'
      WRITE(*,*)
      WRITE(*,*)' ++++++++++++++++++++++++++++++++++++++++++++++++++++'
      WRITE(*,*)' G. A. Bachelor,  Sr. Systems Analyst,'
      WRITE(*,*)' D. E. Cawlfield, Sr. Systems Analyst,'
      WRITE(*,*)' F. T. Lindstrom, Assoc. Prof.,'
      WRITE(*,*)' Soil Science Dept. Oregon State Univ.,'
      WRITE(*,*)' Corvallis, OR., 97331,   (503) 737-2441'
      WRITE(*,*)' ++++++++++++++++++++++++++++++++++++++++++++++++++++'
C
      WRITE(*,*)' Enter base name of WATER data file.'
      CALL RDBASE(FILBAS,'WAD')
      WRITE(*,*)
C
C   UNITS 2, 3, 8, AND 9 USED FOR OUTPUT, UNIT 1 FOR INPUT
C
      WRITE(*,*) ' Input  file 1 is: ',FILBAS,'.WAD'
      WRITE(*,*) ' Output file 2 is: ',FILBAS,'.WAO'
      WRITE(*,*) ' Output file 3 is: ',FILBAS,'.WPV'
      WRITE(*,*) ' Output file 8 is: ',FILBAS,'.WIF'
      WRITE(*,*) ' Output file 9 is: ',FILBAS,'.WDB'
      WRITE(*,*)
C
      OPEN(UNIT=1,FILE=':HOSTCHAR:' // FILBAS // '.WAD',
```

```
     &     STATUS='OLD')
      OPEN(UNIT=2,FILE=FILBAS//'.WAO',STATUS='UNKNOWN')
      OPEN(UNIT=3,FILE=FILBAS//'.WPV',STATUS='UNKNOWN')
      OPEN(UNIT=9,FILE=FILBAS//'.WDB',STATUS='UNKNOWN')
C
      WRITE(*,*)' Calling subroutine FLOREAD'
      WRITE(*,*)
      WRITE(*,*)' Reading from data file ',FILBAS,'.WAD'
      CALL FLOREAD
      WRITE(*,*)' All flow field data successfully read in.'
      WRITE(*,*)
      LENWIF=MAX(NSLXP1,NSLYP1,30)*8 + 4
      OPEN(UNIT=8,FILE=FILBAS//'.WIF',FORM='UNFORMATTED',
     &     ACCESS='SEQUENTIAL',STATUS='UNKNOWN',
     &     RECL=LENWIF, IOSTAT=WIFERR)
      IF (WIFERR .NE. 0) THEN
         WRITE(*,*) ' Trouble opening WIF file!'
         STOP 50
      ENDIF
C
      WRITE(*,*)' Writing input and calculated params. to ',
     &          FILBAS,'.WAO'
      CALL FLOOUT
      WRITE(*,*)
C
C  BEGIN WATER PROCESSING.
C
      WRITE(*,*)' Beginning water processing.'
      WRITE(*,*)
C
C..Compute the *integration time* (startup is ignored). . .
C
      TIMEI = SECOND()
      CALL INTGRL
      TIMEI = SECOND() - TIMEI
C
      WRITE(*,*)' Pressure and velocity fields written to ',
     &          FILBAS,'.WPV'
      WRITE(*,*)
C
      WRITE(*,*)' Writing interface file to ',FILBAS,'.WIF'
      WRITE(*,*)
C
      WIFVER=31.0
      WRITE(8) WIFVER
      WRITE(8) ' LT3VSI INTERFACE FILE VER. 31.0 '
      WRITE(8) NSLXM1,NSLYM1,NSLXXX,NSLYYY,NSLXP1,NSLYP1,NINJW
      WRITE(8) NFLAG,ZTHRSH
      WRITE(8) CONST1,CONST2,CONST3,CONST4,RHOWAT,XLW
      WRITE(8) (XNODE(I),I=1,NSLXP1)
```

```
      WRITE(8) (YNODE(J),J=1,NSLYP1)
      WRITE(8) (DX(I),I=1,NSLXP1)
      WRITE(8) (DY(J),J=1,NSLYP1)
      DO 40, I=1,NSLXP1
         WRITE(8) (QWELIN(I,J),J=1,NSLYP1)
         WRITE(8) (NFUNC(I,J),J=1,NSLYP1)
         WRITE(8) (VLXX(I,J),J=1,NSLYP1)
         WRITE(8) (VLYY(I,J),J=1,NSLYP1)
   40 CONTINUE
C
      TIMET = SECOND() - TIMET
      CALL CONVTIM(TIMEI, IDAY, IHRS, IMIN, ISEC)
      WRITE(*,2200) IDAY, IHRS, IMIN, ISEC
      WRITE(9,2200) IDAY, IHRS, IMIN, ISEC
      CALL CONVTIM(TIMET, IDAY, IHRS, IMIN, ISEC)
      WRITE(*,2210) IDAY, IHRS, IMIN, ISEC
      WRITE(9,2210) IDAY, IHRS, IMIN, ISEC
      STOP ' Normal Fortran Termination'
C
 2200 FORMAT(  1X,'Total integration time was ', I2.2, ' days, ',
     &  I2.2, ' hours, ', I2.2, ' minutes, and ', I2.2, ' sec. ')
 2210 FORMAT(/,1X,'Total CPU  Clock  time was ', I2.2, ' days, ',
     &  I2.2, ' hours, ', I2.2, ' minutes, and ', I2.2, ' sec. ',/)
      END
```

---

```
*         File: LUFSIW.FOR              Last revision:  August 29, 1990
C                                       For water phase of LT3VSI.
      SUBROUTINE LUFACT
C
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CSSIP.SIW'
C
      INTEGER           I, J, II, JJ
      DOUBLE PRECISION  SAXY
C
C  The following is a STATEMENT FUNCTION.
C
      SAXY(II,JJ) = ADT2X(II,JJ) + ADT2Y(II,JJ)
C
*****************************************************************
*  Perform a one-pass incomplete LU factorization (Stone, 1968).
*  Variables --
*     B(i,j)    -  small b-ij, lower band
*     C(i,j)    -    "   c-ij, lower off-diagonal
```

```
*     D(i,j)    -     "   d-ij, diagonal
*     E(i,j)    -     "   e-ij, upper off-diagonal
*     F(i,j)    -     "   f-ij, upper band
*     ALPH      -   The Stone "weighting parameter", 0 <= ALPH <= 1
*
*  The original LHS array ("A") is stored in banded form in the
*  ADT1, ADT3, ALT2 and AUT2 arrays.
*                     David E. Cawlfield,  July '89
**********************************************************************
C
C  FIRST ROW
C
      B(2,2) = 0.D0
      C(2,2) = 0.D0
      D(2,2) = SAXY(2,2)
      E(2,2) = ADT3(2,2) / D(2,2)
      F(2,2) = AUT2(2,2) / D(2,2)
C
C  REST OF LEFT CENTRAL BLOCK
C
      DO 10, J = 3, NSLYM1
         B(2,J) = 0.D0
         C(2,J) = ADT1(2,J) / (1.D0 + ALPH*F(2,J-1))
         D(2,J) = SAXY(2,J) + C(2,J)*(ALPH*F(2,J-1) - E(2,J-1))
         E(2,J) = ADT3(2,J) / D(2,J)
         F(2,J) = (AUT2(2,J) - ALPH*C(2,J)*F(2,J-1)) / D(2,J)
   10 CONTINUE
C
C  REST OF LEFT BOTTOM
C
      B(2,NSLYYY) = 0.D0
      C(2,NSLYYY) = ADT1(2,NSLYYY) / (1.D0 + ALPH*F(2,NSLYM1))
      D(2,NSLYYY) = SAXY(2,NSLYYY) +
     &    C(2,NSLYYY)*(ALPH*F(2,NSLYM1) - E(2,NSLYM1))
      E(2,NSLYYY) = 0.D0
      F(2,NSLYYY) = (AUT2(2,NSLYYY) - ALPH*C(2,NSLYYY)*F(2,NSLYM1)) /
     &              D(2,NSLYYY)
C
C  INSIDE TOP
C
      DO 30, I = 3, NSLXM1
         B(I,2) = ALT2(I,2) / (1.D0 + ALPH*E(I-1,2))
         C(I,2) = 0.D0
         D(I,2) = SAXY(I,2) + B(I,2)*(ALPH*E(I-1,2) - F(I-1,2))
         E(I,2) = (ADT3(I,2) - ALPH*B(I,2)*E(I-1,2)) / D(I,2)
         F(I,2) = AUT2(I,2) / D(I,2)
C
C  CENTRAL
C
         DO 20, J = 3, NSLYM1
```

```
            B(I,J) = ALT2(I,J) / (1.D0 + ALPH*E(I-1,J))
            C(I,J) = ADT1(I,J) / (1.D0 + ALPH*F(I,J-1))
            D(I,J) = SAXY(I,J) +
     &         B(I,J)*(ALPH*E(I-1,J) - F(I-1,J)) +
     &         C(I,J)*(ALPH*F(I,J-1) - E(I,J-1))
            E(I,J) = (ADT3(I,J) - ALPH*B(I,J)*E(I-1,J)) / D(I,J)
            F(I,J) = (AUT2(I,J) - ALPH*C(I,J)*F(I,J-1)) / D(I,J)
   20     CONTINUE
          B(I,NSLYYY) = ALT2(I,NSLYYY)
          C(I,NSLYYY) = ADT1(I,NSLYYY) / (1.D0 + ALPH*F(I,NSLYM1))
          D(I,NSLYYY) = SAXY(I,NSLYYY) +
     &     C(I,NSLYYY)*(ALPH*F(I,NSLYM1) - E(I,NSLYM1)) -
     &     B(I,NSLYYY)*F(I-1,NSLYYY)
          E(I,NSLYYY) = 0.D0
          F(I,NSLYYY) = (AUT2(I,NSLYYY) - ALPH*C(I,NSLYYY)*F(I,NSLYM1)) /
     &                  D(I,NSLYYY)
   30 CONTINUE
C
C  RIGHT HAND TOP
C
          B(NSLXXX,2) = ALT2(NSLXXX,2) / (1.D0 + ALPH*E(NSLXM1,2))
          C(NSLXXX,2) = 0.D0
          D(NSLXXX,2) = SAXY(NSLXXX,2) +
     &                  B(NSLXXX,2)*(ALPH*E(NSLXM1,2) - F(NSLXM1,2))
          E(NSLXXX,2) = (ADT3(NSLXXX,2) - ALPH*B(NSLXXX,2)*E(NSLXM1,2)) /
     &                   D(NSLXXX,2)
          F(NSLXXX,2) = 0.D0
C
C  RIGHT CENTRAL
C
          DO 40, J = 3, NSLYM1
             B(NSLXXX,J) = ALT2(NSLXXX,J) / (1.D0 + ALPH*E(NSLXM1,J))
             C(NSLXXX,J) = ADT1(NSLXXX,J)
             D(NSLXXX,J) = SAXY(NSLXXX,J) +
     &                     B(NSLXXX,J)*(ALPH*E(NSLXM1,J) - F(NSLXM1,J)) -
     &                     C(NSLXXX,J)*E(NSLXXX,J-1)
             E(NSLXXX,J) = (ADT3(NSLXXX,J) - ALPH*B(NSLXXX,J)*E(NSLXM1,J)) /
     &                      D(NSLXXX,J)
             F(NSLXXX,J) = 0.D0
   40 CONTINUE
C
C  BOTTOM ROW OF BOTTOM BLOCK
C
          B(NSLXXX,NSLYYY) = ALT2(NSLXXX,NSLYYY)
          C(NSLXXX,NSLYYY) = ADT1(NSLXXX,NSLYYY)
          D(NSLXXX,NSLYYY) = SAXY(NSLXXX,NSLYYY) -
     &                  C(NSLXXX,NSLYYY)*E(NSLXXX,NSLYM1) -
     &                  B(NSLXXX,NSLYYY)*F(NSLXM1,NSLYYY)
          E(NSLXXX,NSLYYY) = 0.D0
          F(NSLXXX,NSLYYY) = 0.D0
```

```
C
      RETURN
      END
```

```
*         File: NEWTWO.FOR              Last revision:  August 29, 1990
C                                       For chem loop phase of LT3VSI.
C
      SUBROUTINE NEWTWO(NDIM,X,RHS,AJAC,FCT,PAR,TOLER,MITER,IERR)
      INTEGER   NDIM, MITER, IERR
      DOUBLE PRECISION X(NDIM), RHS(NDIM), AJAC(NDIM,NDIM), PAR(*)
      DOUBLE PRECISION TOLER, TOLRAN
      INTEGER    MDIM, ITER
      PARAMETER (MDIM = 2)
      DOUBLE PRECISION XO(MDIM), DELX(MDIM), TEST, DENOM, EPS
      PARAMETER (EPS = 1.D-10)
      EXTERNAL FCT
*-----------------------------------------------------------------------*
*     NEWTWO  --  Two-Dimensional Newton-Raphson Routine                *
*                                                                       *
*       The two-dimensional sub-set of the GENEWT (generalized)         *
*  Newton-Raphson Routine.                                              *
*                                                                       *
*  Parameters:                                                          *
*     NDIM       -  Size of problem; number of variables & functions.   *
*     X          -  Array of independent variables, size NDIM.          *
*     RHS        -  Array of dependent function values (the right hand  *
*                   side).                                              *
*     AJAC       -  Array of Jacobian elements, dimension NDIM by NDIM. *
*     FCT        -  A user supplied external subroutine which evaluates *
*                   both RHS and AJAC.  Called with                     *
*                     CALL   FCT(NDIM, X, RHS, AJAC, PAR)               *
*     PAR        -  An array for passing parameters into FCT, if        *
*                   needed.  Should be dimensioned (1) or (*) in FCT.   *
*                   Not used in this version.                           *
*     TOLER      -  The tolerance for convergence.  A local variable,   *
*                   TOLRAN, is set equal to the square of TOLER.        *
*                   TOLRAN is compared with the square of the distance  *
*                   between the old and new values of X.                *
*     MITER      -  The maximum number of iterations allowed.  If       *
*                   this limit is reached, the process is terminated,   *
*                   and an error is indicated.                          *
*     IERR       -  An error flag:                                      *
*                     + n  ==  No error, number of iterations taken     *
*                     + 0  ==  NDIM was <= zero.                        *
*                     - 1  ==  Singular Jacobian.  (not used here)      *
```

```
*                    - 2  ==  Too many iterations.                      *
*                    - 3  ==  NDIM > MDIM (up MDIM & re-compile).       *
*                                                                       *
*  Externals:                                                           *
*                                                                       *
*     FCT(NDIM, X, RHS, AJAC, PAR)                                      *
*     DIMENSION X(NDIM), RHS(NDIM), AJAC(NDIM,NDIM), PAR(*)             *
*     A user supplied external subroutine (FCT) is required to fill in  *
*  the RHS and Jacobian elements, as described above.                   *
*                                                                       *
************************************************************************
      IF (NDIM .LE. 0) THEN
           WRITE(*,2000)
           IERR = 0
           RETURN
      ENDIF
      IF (NDIM .GT. MDIM) THEN
           WRITE(*,2001)
           IERR = -3
      ENDIF
      TOLRAN = TOLER * TOLER
C     WRITE(*,2050)
C     WRITE(*,2060)   0 , 0.D0, (X(I), I=1, NDIM)
C
*-----------------------------------------------------------------------*
*  Two variable case.                                                   *
*-----------------------------------------------------------------------*
*  Iterate:                                                             *
*     X{n+1} = F(X{n},Y{n}) = X{n} + d(1), and                          *
*     Y{n+1} = G(X{n},Y{n}) = Y{n} + d(2).                              *
*  d(1) and d(2) are found by solving the system:                       *
*          Fx * d(1) +  Fy * d(2) = -F                                  *
*          Gx * d(1) +  Gy * d(2) = -G                                  *
*  which arises from the Taylor series expansion of F(X+d(1),Y+d(2)),   *
*  etc.  See any numerical analysis text.                               *
*-----------------------------------------------------------------------*
           XO(1) = X(1)
           XO(2) = X(2)
           DO 200, ITER = 1, MITER
                CALL FCT(NDIM, X, RHS, AJAC, PAR)
                DENOM = (AJAC(1,1)*AJAC(2,2) - AJAC(1,2)*AJAC(2,1))
                IF (ABS(DENOM) .LT. EPS) THEN
                     WRITE(*,2020) DENOM
                     IERR = -1
                     GO TO 999
                ENDIF
                DELX(1) = (RHS(2)*AJAC(1,2) - RHS(1)*AJAC(2,2)) / DENOM
                DELX(2) = (RHS(1)*AJAC(2,1) - RHS(2)*AJAC(1,1)) / DENOM
C Need a -RHS and *add* DELX, or will +RHS let us *sub* DELX?
                X(1) = XO(1) + DELX(1)
```

```
                X(2) = XO(2) + DELX(2)
                TEST = (X(1) - XO(1))**2 +
     &                 (X(2) - XO(2))**2
C               WRITE(*,2060) ITER, TEST, X(1), X(2)
                XO(1) = X(1)
                XO(2) = X(2)
                IF (TEST .LT. TOLRAN) GO TO 250
  200       CONTINUE
            WRITE(*,2010) MITER
            IERR = -2
            GO TO 999
  250       IERR = ITER
C
  999 RETURN
C
 2000 FORMAT(1X,'*NEWTWO* - NDIM was <= zero.  Iterate a constant?')
 2001 FORMAT(1X,'*NEWTWO* - NDIM > MDIM.  Fix & re-compile.')
 2010 FORMAT(1X,'*NEWTWO* -- Caution.  Max iterations (',
     &  I3, ') exceeded.',/,
     &  T10,'Solution may be unstable or TOLER too small.')
 2020 FORMAT(1X,'*NEWTWO* -- The denominator was small enough (',
     &  1P,G13.6,' to be zero.',/,
     &  T10,'This may or may not be the solution.')
C2030 FORMAT(1X,'*NEWTWO* - Singular Jacobian Matrix.  Bye.')
C2040 FORMAT(1X,'*NEWTWO* - The impossible has happened (case).')
C2050 FORMAT(/,1X,'ITER ', '          TEST', '          X(N)')
C2060 FORMAT(1X,I3, 2X, 1P, 4(1X,G13.6), /, (T20,3(1X,G13.6)) )
      END
```

---

```
*         File: OUTSIB.FOR          Last revision: August 29, 1990
C                                   For all three phases of LT3VSI.
C
C  These subroutines provide a variable-format output that is clearer
C  than the G-format of *certain* versions of Fortran.
C  Coded by Gilbert A. Bachelor,  Febr..Aug 1989
C
      SUBROUTINE PRINT1(LUN,N,ARY)
      INTEGER C1, C2, LUN, N
      DOUBLE PRECISION ARY(*)
      CHARACTER LINE*81
C
C  PRINTS ELEMENTS 1 THRU N OF ARRAY ARY ON UNIT LUN, USING
C  VARIABLE FORMAT; 8 ELEMENTS PER LINE.
C
1000  FORMAT(A)
```

```
      C1=1
      C2=MIN(8,N)
10    CONTINUE
         CALL ARYSTR(ARY,C1,C2,2,10,LINE)
         WRITE(LUN,1000) LINE(1:(C2-C1+1)*10)
         IF (C2.LT.N) THEN
            C1=C2+1
            C2=MIN(C2+8,N)
            GOTO 10
         ENDIF
      RETURN
      END
C
C  ************************************************
C
      SUBROUTINE PRINT2(LUN,N,ARY)
      INTEGER C1, C2, I, LUN, N
      DOUBLE PRECISION ARY(*)
      CHARACTER LINE*81
C
C  PRINTS ELEMENTS 1 THRU N OF ARRAY ARY ON UNIT LUN, USING
C  VARIABLE FORMAT; 8 ELEMENTS PER LINE.  ALSO PRINTS INDEX
C  ON LINE ABOVE.
C
1000  FORMAT(1X,8(2X,I3,5X))
1001  FORMAT(A,/)
      C1=1
      C2=MIN(8,N)
10    CONTINUE
         WRITE(LUN,1000) (I,I=C1,C2)
         CALL ARYSTR(ARY,C1,C2,2,10,LINE)
         WRITE(LUN,1001) LINE(1:(C2-C1+1)*10)
         IF (C2.LT.N) THEN
            C1=C2+1
            C2=MIN(C2+8,N)
            GOTO 10
         ENDIF
      RETURN
      END
C
C  ************************************************
C
      SUBROUTINE PRINT3(LUN,NX,NY,ARY)
C
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
C
      INTEGER C1,C2,I,J,LUN,NCP,NX,NY,PAGE
      CHARACTER CHARY(12)*5, LINE*121, TEMP*5
      DOUBLE PRECISION ARY(IX,IY)
```

```
C
C  PRINTS (ON UNIT LUN) NX ROWS BY NY COLUMNS OF ARRAY ARY, WHOSE
C  DECLARED SIZE IS IX BY IY.  PRINTS I AND XNODE(I) AS HORIZONTAL
C  LABEL, AND J AND YNODE(J) AS VERTICAL LABEL.
C
1000  FORMAT(5X,'PAGE ',I2)
1010  FORMAT(8X,'X',12(3X,A,2X))
1020  FORMAT(3X,'Y',2X,12(7X,I3))
1030  FORMAT(1X,A,I3,A)
      IF (NFLAG(8).NE.0) THEN
         NCP=12
      ELSE
         NCP=7
      ENDIF
      PAGE=1
      C1=1
      C2=MIN(NCP,NX)
10    CONTINUE
         WRITE(LUN,1000) PAGE
         DO 15, I=C1,C2
            CALL NUMFIX(XNODE(I),CHARY(I-C1+1))
15       CONTINUE
         WRITE(LUN,1010) (CHARY(I-C1+1),I=C1,C2)
         WRITE(LUN,1020) (I,I=C1,C2)
         DO 20, J=1,NY
            CALL NUMFIX(YNODE(J),TEMP)
            CALL ARYSTR(ARY(1,J),C1,C2,2,10,LINE)
            WRITE(LUN,1030) TEMP,J,LINE(1:(C2-C1+1)*10)
20       CONTINUE
         IF (C2.LT.NX) THEN
            WRITE(LUN,*)
            C1=C2+1
            C2=MIN(C2+NCP,NX)
            PAGE=PAGE+1
            GOTO 10
         ENDIF
      RETURN
      END
C
C  ************************************************
C
      SUBROUTINE PRIN3S(LUN,NX,NY,ARY)
C
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
C
      INTEGER C1, C2, I, J, LUN, NCP, NX, NY, PAGE, RPT, SAVJ
      CHARACTER CHARY(12)*5, LINE*121, SAVLIN*121, TEMP*5
      DOUBLE PRECISION ARY(IX,IY)
C
```

```
C  PRINTS (ON UNIT LUN) NX ROWS BY NY COLUMNS OF ARRAY ARY, WHOSE
C  DECLARED SIZE IS IX BY IY.  PRINTS I AND XNODE(I) AS HORIZONTAL
C  LABEL, AND J AND YNODE(J) AS VERTICAL LABEL.  ** THIS SUBROUTINE
C  IS THE SAME AS PRINT3, EXCEPT THAT IT SUPPRESSES DUPLICATE LINES.
C
1000  FORMAT(5X,'PAGE ',I2)
1010  FORMAT(8X,'X',12(3X,A,2X))
1020  FORMAT(3X,'Y',2X,12(7X,I3))
1030  FORMAT(1X,A,I3,A)
1040  FORMAT(10X,'Line above repeated ',I3,' times.')
      IF (NFLAG(8).NE.0) THEN
         NCP=12
      ELSE
         NCP=7
      ENDIF
      PAGE=1
      C1=1
      C2=MIN(NCP,NX)
10    CONTINUE
         WRITE(LUN,1000) PAGE
         DO 15, I=C1,C2
            CALL NUMFIX(XNODE(I),CHARY(I-C1+1))
15       CONTINUE
         WRITE(LUN,1010) (CHARY(I-C1+1),I=C1,C2)
         WRITE(LUN,1020) (I,I=C1,C2)
         RPT=1
         SAVJ=1
         CALL ARYSTR(ARY(1,1),C1,C2,2,10,SAVLIN)
         DO 20, J=2,NY
            CALL ARYSTR(ARY(1,J),C1,C2,2,10,LINE)
            IF (LINE .EQ. SAVLIN) THEN
               RPT=RPT+1
            ELSE
               CALL NUMFIX(YNODE(SAVJ),TEMP)
               WRITE(LUN,1030) TEMP,SAVJ,SAVLIN(1:(C2-C1+1)*10)
               IF (RPT .GT. 1) WRITE(LUN,1040) RPT-1
               RPT=1
               SAVJ=J
               SAVLIN=LINE
            ENDIF
20       CONTINUE
         CALL NUMFIX(YNODE(SAVJ),TEMP)
         WRITE(LUN,1030) TEMP,SAVJ,SAVLIN(1:(C2-C1+1)*10)
         IF (RPT .GT. 1) WRITE(LUN,1040) RPT-1
         IF (C2.LT.NX) THEN
            WRITE(LUN,*)
            C1=C2+1
            C2=MIN(C2+NCP,NX)
            PAGE=PAGE+1
            GOTO 10
```

```
         ENDIF
      RETURN
      END
C
C  ************************************************
C
      SUBROUTINE PRINT4(LUN,NX,NY,ARX,ARY)
C
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
C
      INTEGER C1,C2,I,J,LUN,NCP,NX,NY,PAGE
      CHARACTER CHARY(12)*5, LINE*121, TEMP*5
      DOUBLE PRECISION ARX(IX,IY),ARY(IX,IY)
C
C  PRINTS (ON UNIT LUN) NX ROWS BY NY COLUMNS OF ARRAYS ARX AND ARY,
C  BOTH OF WHICH HAVE A DECLARED SIZE OF IX BY IY.  CORRESPONDING
C  ELEMENTS OF ARX AND ARY REPRESENT THE X AND Y COMPONENTS OF A
C  VECTOR QUANTITY.  THIS SUBROUTINE CONVERTS THE (X,Y) VALUES TO
C  MAGNITUTE AND ANGLE (IN DEGREES) AND PRINTS THE POLAR FORM.
C  IT ALSO PRINTS I AND XNODE(I) AS HORIZONTAL LABEL, AND J AND
C  YNODE(J) AS VERTICAL LABEL.
C
1000  FORMAT(5X,'PAGE ',I2)
1010  FORMAT(8X,'X',7(6X,A,6X))
1020  FORMAT(3X,'Y',7(12X,I3,2X))
1030  FORMAT(1X,A,I3,A)
      IF (NFLAG(8).NE.0) THEN
         NCP=7
      ELSE
         NCP=4
      ENDIF
      PAGE=1
      C1=1
      C2=MIN(NCP,NX)
10    CONTINUE
         WRITE(LUN,1000) PAGE
         DO 15, I=C1,C2
            CALL NUMFIX(XNODE(I),CHARY(I-C1+1))
15       CONTINUE
         WRITE(LUN,1010) (CHARY(I-C1+1),I=C1,C2)
         WRITE(LUN,1020) (I,I=C1,C2)
         DO 30, J=1,NY
            CALL NUMFIX(YNODE(J),TEMP)
            CALL ARYPOL(ARX(1,J),ARY(1,J),C1,C2,2,17,LINE)
            WRITE(LUN,1030) TEMP,J,LINE(1:(C2-C1+1)*17)
30       CONTINUE
         IF (C2.LT.NX) THEN
            WRITE(LUN,*)
            C1=C2+1
```

```
            C2=MIN(C2+NCP,NX)
            PAGE=PAGE+1
            GOTO 10
         ENDIF
      RETURN
      END
C
C  ***********************************************
C
      SUBROUTINE ARYSTR(ARY,N1,N2,M,W,STR)
      INTEGER K, M, N, N1, N2, W
      DOUBLE PRECISION ARY(*)
      CHARACTER STR*(*)
C
C  CONVERTS ELEMENTS ARY(N1) THRU ARY(N2) OF ARRAY ARY INTO
C  STRING FORM (PRINTABLE) AND STORES THEM IN STRING STR,
C  BEGINNING AT POSITION M>=1, USING WIDTH W>=9 POSITIONS PER
C  NUMBER.  ALL UNUSED POSITIONS ARE SET TO BLANK.
C
      STR=' '
      K=M
      DO 20, N=N1,N2
         CALL NUMSTR(ARY(N),STR(K:K+8))
         K=K+W
20    CONTINUE
      RETURN
      END
C
C  ***********************************************
C
      SUBROUTINE ARYPOL(ARX,ARY,N1,N2,M,W,STR)
      INTEGER K, M, N, N1, N2, W
      DOUBLE PRECISION ARX(*),ARY(*)
      CHARACTER STR*(*)
C
C  CONVERTS ELEMENTS N1 THRU N2 OF ARRAYS ARX AND ARY INTO
C  POLAR FORM (PRINTABLE STRING) AND STORES THEM IN STRING STR,
C  BEGINNING AT POSITION M>=1, USING WIDTH W>=16 POSITIONS PER
C  NUMBER.  ALL UNUSED POSITIONS ARE SET TO BLANK.
C
      STR=' '
      K=M
      DO 20, N=N1,N2
         CALL POLAR(ARX(N),ARY(N),STR(K:K+15))
         K=K+W
20    CONTINUE
      RETURN
      END
C
C  ***********************************************
```

```
C
      SUBROUTINE NUMSTR(X,ALF)
C
      include 'CSIZE.SIB'
C
      INTEGER K
      DOUBLE PRECISION X
      CHARACTER ALF*9, FORM(1:11)*9
      DATA FORM/'(F7.5)','(F7.5)','(F7.5)','(F7.4)','(F7.3)',
     &   '(F7.2)','(F7.1)','(F7.0)','(1P,E9.2)','(F3.0)','(F4.2)'/
C
C  CONVERTS DOUBLE PRECISION NUMBER X INTO PRINTABLE STRING FORM
C  AND STORES IT IN STRING ALF OF WIDTH 9, USING VARIABLE FORMAT.
C
      IF (X.EQ.0.0D0) THEN
         K=10
      ELSEIF (ABS(X).LT.ZTHRSH) THEN
         K=11
      ELSE
         K=INT(LOG10(ABS(X))+4)
         IF (K.LT.1 .OR. K.GT.8) K=9
      ENDIF
40    WRITE(ALF,FORM(K)) X
      IF (ALF(2:2).EQ.'*') THEN
         K=MIN(K+1,9)
         GOTO 40
      ENDIF
      RETURN
      END
C
C  ***********************************************
C
      SUBROUTINE NUMFIX(X,ALF)
      INTEGER K
      DOUBLE PRECISION X
      CHARACTER ALF*5, FORM(1:6)*6
      DATA FORM/'(F5.4)','(F5.3)','(F5.2)','(F5.1)',
     &   '(F5.0)','(F2.0)'/
C
C  CONVERTS NON-NEGATIVE DOUBLE PRECISION NUMBER X INTO PRINTABLE
C  STRING FORM (FIXED POINT FORMAT), AND STORES IT IN STRING ALF
C  OF WIDTH 5, USING VARIABLE FORMAT.
C
      IF (X.NE.0.0) THEN
         K=INT(LOG10(ABS(X))+2)
         IF (K.LT.1) THEN
            K=1
         ELSEIF (K.GT.5) THEN
            K=5
         ENDIF
```

```
      ELSE
         K=6
      ENDIF
      WRITE(ALF,FORM(K)) X
      RETURN
      END
C
C  ***********************************************
C
      SUBROUTINE POLAR(X,Y,ALF)
      INTEGER K
      DOUBLE PRECISION ANGLE, MAG, X, Y
      CHARACTER ALF*16, FORM(1:10)*9, AFORM(1:4)*6
      DATA FORM/'(2X,F7.5)','(2X,F7.5)','(2X,F7.5)','(2X,F7.4)',
     &   '(2X,F7.3)','(2X,F7.2)','(2X,F7.1)','(2X,F7.0)',
     &   '(1P,E9.2)','(5X,F4.1)'/
      DATA AFORM/'(F5.3)','(F5.2)','(F5.1)','(F5.0)'/
C
C  DOUBLE PRECISION ARGUMENTS X AND Y ARE THE X AND Y COMPONENTS
C  OF A VECTOR QUANTITY.  THIS SUBROUTINE CONVERTS (X,Y) INTO
C  POLAR FORM (MAGNITUDE AND ANGLE IN DEGREES) AND STORES A
C  PRINTABLE FORM IN STRING ARGUMENT ALF OF WIDTH 16, USING
C  VARIABLE FORMAT.  **NOTE: THE POSITIVE X-AXIS IS 90 DEGREES;
C  THE POSITIVE Y-AXIS IS 0 DEGREES.
C
      MAG=SQRT(X**2 + Y**2)
      IF (MAG.NE.0.0) THEN
         ANGLE=ATAN2(X,Y)*180.0D0/3.1415926536D0
         K=INT(LOG10(ABS(MAG))+4)
         IF (K.LT.1 .OR. K.GT.8) K=9
      ELSE
         ANGLE=0.0
         K=10
      ENDIF
40    WRITE(ALF(1:9),FORM(K)) MAG
      IF (ALF(2:2).EQ.'*') THEN
         K=MIN(K+1,9)
         GOTO 40
      ENDIF
      ALF(10:10)=','
      IF (ANGLE.NE.0.0) THEN
         K=INT(LOG10(ABS(ANGLE))+2)
         IF (K.LT.1) THEN
            K=1
         ELSEIF (K.GT.4) THEN
            K=4
         ENDIF
      ELSE
         K=1
      ENDIF
```

```
60     WRITE(ALF(11:15),AFORM(K)) ANGLE
       IF (ALF(12:12).EQ.'*') THEN
          K=K+1
          GOTO 60
       ENDIF
       ALF(16:16)='o'
       RETURN
       END
```

---

```
*          File: PSTARSIL.FOR          Last revision:  August 29, 1990
C                                      For chem loop phase of LT3VSI.
C
C  PSTAR computes the coordinates of the points P(.)(i,j,n)
C  (called P* for short), with some help from subroutines
C  NEWTWO  and FEVAL.
C  By Gilbert A. Bachelor, Dec. 1988; Apr. 1990.
C
       SUBROUTINE PSTAR
C
       IMPLICIT DOUBLE PRECISION(A-H,O-Z)
       INTEGER MITER, NDIM
       DOUBLE PRECISION TOLER
       PARAMETER(MITER=10,NDIM=2,TOLER=1.0D-4)
       include 'CSIZE.SIB'
       include 'CPROP.SIB'
       include 'CVELOC.SIB'
       include 'CRUNC.SIK'
       include 'CNRK.SIL'
       include 'CCHEM.SIK'
       include 'CVELOC.SIK'
C
       INTEGER I, J, K, IERR
       LOGICAL PSERR
       DOUBLE PRECISION AJAC(NDIM,NDIM), RHS(NDIM)
       DOUBLE PRECISION X(NDIM), PAR
       EXTERNAL FEVAL
C
C  COMPUTE P*(i,j,n) FOR INTERIOR NODES.
C
       WRITE(*,*) ' Computing the P* points.'
       WRITE(*,*)
       PSERR=.FALSE.
       DO 600, K=1,NC
        DO 600, I=2,NSLXXX
         DO 600, J=2,NSLYYY
```

```
C
C  Compute the coefficients A1, B1, C1, D1, A2, B2, C2, D2, and
C  store them in COEF(1) through COEF(8), respectively.
C  The formulas are different, depending on which of the 4 quadrants
C  P* should lie in.  The NFUNC array tells which quadrant to use.
C
      COEF(1)=UX(I,J,K)
      COEF(5)=UY(I,J,K)
C
      GOTO (100,200,300,400), NFUNC(I,J)
      PRINT 1300,NFUNC(I,J),I,J
1300  FORMAT(1X,'NFUNC =',I4,'at I=',I3,', J=',I3)
      WRITE(*,*) ' NFUNC OUT OF RANGE!'
      STOP 2
C
C  Quadrant 1.
C
100   CONTINUE
      COEF(2)=(UX(I,J,K)-UX(I-1,J,K))/DX(I-1)
      COEF(6)=(UY(I,J,K)-UY(I-1,J,K))/DX(I-1)
      COEF(3)=(UX(I,J,K)-UX(I,J-1,K))/DY(J-1)
      COEF(7)=(UY(I,J,K)-UY(I,J-1,K))/DY(J-1)
      COEF(4)=(UX(I,J,K)+UX(I-1,J-1,K)-UX(I-1,J,K)-UX(I,J-1,K))/
     &   (DX(I-1)*DY(J-1))
      COEF(8)=(UY(I,J,K)+UY(I-1,J-1,K)-UY(I-1,J,K)-UY(I,J-1,K))/
     &   (DX(I-1)*DY(J-1))
      GOTO 500
C
C  Quadrant 2.
C
200   CONTINUE
      COEF(2)=(UX(I+1,J,K)-UX(I,J,K))/DX(I)
      COEF(6)=(UY(I+1,J,K)-UY(I,J,K))/DX(I)
      COEF(3)=(UX(I,J,K)-UX(I,J-1,K))/DY(J-1)
      COEF(7)=(UY(I,J,K)-UY(I,J-1,K))/DY(J-1)
      COEF(4)=(UX(I,J-1,K)+UX(I+1,J,K)-UX(I,J,K)-UX(I+1,J-1,K))/
     &   (DX(I)*DY(J-1))
      COEF(8)=(UY(I,J-1,K)+UY(I+1,J,K)-UY(I,J,K)-UY(I+1,J-1,K))/
     &   (DX(I)*DY(J-1))
      GOTO 500
C
C  Quadrant 3.
C
300   CONTINUE
      COEF(2)=(UX(I,J,K)-UX(I-1,J,K))/DX(I-1)
      COEF(6)=(UY(I,J,K)-UY(I-1,J,K))/DX(I-1)
      COEF(3)=(UX(I,J+1,K)-UX(I,J,K))/DY(J)
      COEF(7)=(UY(I,J+1,K)-UY(I,J,K))/DY(J)
      COEF(4)=(UX(I-1,J,K)+UX(I,J+1,K)-UX(I,J,K)-UX(I-1,J+1,K))/
     &   (DX(I-1)*DY(J))
```

```
      COEF(8)=(UY(I-1,J,K)+UY(I,J+1,K)-UY(I,J,K)-UY(I-1,J+1,K))/
     &    (DX(I-1)*DY(J))
      GOTO 500
C
C  Quadrant 4.
C
400   CONTINUE
      COEF(2)=(UX(I+1,J,K)-UX(I,J,K))/DX(I)
      COEF(6)=(UY(I+1,J,K)-UY(I,J,K))/DX(I)
      COEF(3)=(UX(I,J+1,K)-UX(I,J,K))/DY(J)
      COEF(7)=(UY(I,J+1,K)-UY(I,J,K))/DY(J)
      COEF(4)=(UX(I+1,J+1,K)+UX(I,J,K)-UX(I+1,J,K)-UX(I,J+1,K))/
     &    (DX(I)*DY(J))
      COEF(8)=(UY(I+1,J+1,K)+UY(I,J,K)-UY(I+1,J,K)-UY(I,J+1,K))/
     &    (DX(I)*DY(J))
C
C  All four branches come together here.
C
500   CONTINUE
C
C  Initialization for FEVAL: copy data to vars in common block
C  NRK.
C
      DT=DT0
      XI=XNODE(I)
      YJ=YNODE(J)
C
C  Set X array to initial guess and call NEWTWO subroutine.
C
      X(1)=XI
      X(2)=YJ
      CALL NEWTWO(NDIM,X,RHS,AJAC,FEVAL,PAR,TOLER,MITER,IERR)
C
C  Retrieve coordinates of P*, as computed by NEWTWO.
C
      PSTARX(I,J,K)=X(1)
      PSTARY(I,J,K)=X(2)
C
C  Print error messages if anything is wrong.
C
      IF (IERR.LT.1) THEN
         PSERR=.TRUE.
         PRINT 1000,IERR,I,J,K
1000     FORMAT(1X,'Error #',I3,' in computing PSTAR for I=',
     &       I3,', J=',I3,', K=',I3)
      ELSEIF (X(1).LT.XNODE(I-1) .OR. X(1).GT.XNODE(I+1) .OR.
     &        X(2).LT.YNODE(J-1) .OR. X(2).GT.YNODE(J+1)) THEN
         PSERR=.TRUE.
         PRINT 1100,I,J,K
1100     FORMAT(1X,'Error in computing PSTAR for I=',I3,', J=',I3,
```

```
     &      ', K=',I3,/,1X,'Point not in correct region.')
       ENDIF
  600 CONTINUE
C
      IF (PSERR) THEN
         WRITE(*,*) ' *** Errors in computing P* points! ***'
         STOP 2
      ENDIF
C
C  PRINT THE COORDINATES OF THE P* POINTS ON FILE 'xxxxxxxx.CDB'
C
      IF(NFLAG(15).NE.0) THEN
        DO 331, K=1,NC
          WRITE(9,9500) CHNAME(K)
 9500     FORMAT(/,6X,'COORDINATES OF THE P* POINTS FOR ',A,//,
     &           25X,'P-STAR',17X,'P-STAR',/,
     &           4X,'I',4X,'J',7X,'XN(M)',5X,'XN(M)',8X,'YN(M)',
     &           5X,'YN(M)',1X,'NFUNC',/)
          DO 330, I=2,NSLXXX
           DO 330, J=2,NSLYYY
              WRITE(9,9600)I,J,XNODE(I),PSTARX(I,J,K),
     &                YNODE(J),PSTARY(I,J,K),NFUNC(I,J)
 9600         FORMAT(2(1X,I4),2X,2F10.4,3X,2F10.4,2X,I3)
  330     CONTINUE
  331   CONTINUE
      ENDIF
C
      RETURN
      END
C ***********************************
*FEVAL
      SUBROUTINE FEVAL(NDIM, X, RHS, AJAC, PAR)
      INTEGER   NDIM
      DOUBLE PRECISION    X(NDIM), RHS(NDIM), AJAC(NDIM,NDIM), PAR(*)
      DOUBLE PRECISION    DX, DY, XS, YS
C     (common /nrk/   dt, xi, yj, coef(8)
      include 'CNRK.SIL'
*****************************************************************
*  Subroutine used by NEWTWO to compute the right-hand-side (RHS) and  *
*  Jacobian elements (AJAC) for the NRK method.                        *
*     Parameters:                                                      *
*          NDIM      -  The number of variables being iterated by      *
*                       NEWTWO.  This version assumes NDIM=2.          *
*          X         -  The array of current independent variables     *
*                       being solved by NEWTWO.  Size NDIM.            *
*          RHS       -  An array which will return the right-hand-side *
*                       of the system.                                 *
*          AJAC      -  The NDIM by NDIM Jacobian array returned.      *
*          PAR       -  An array for auxiliary variables, if needed.   *
*                       Not used in this version.                      *
```

```
*                                        D. E. Cawlfield, Winter '88        *
**************************************************************************
      XS = X(1)
      YS = X(2)
      DX = XS - XI
      DY = YS - YJ
C
C  The RHS is simply . . .
C
      RHS(1) = DX + DT * (COEF(1) + DX*COEF(2) + DY*COEF(3) +
     &          DX*DY*COEF(4))
      RHS(2) = DY + DT * (COEF(5) + DX*COEF(6) + DY*COEF(7) +
     &          DX*DY*COEF(8))
C
C  Now the four Jacobian elements . . .
C
      AJAC(1,1) = 1.D0 + DT * (COEF(2) + DY*COEF(4))
      AJAC(1,2) =        DT * (COEF(3) + DX*COEF(4))
      AJAC(2,1) =        DT * (COEF(6) + DY*COEF(8))
      AJAC(2,2) = 1.D0 + DT * (COEF(7) + DX*COEF(8))
      RETURN
      END
```

---

```
*         File: RATSIK.FOR           Last revision:  October 9, 1990
C                    For both chem init and chem loop phases of LT3VSI.
      SUBROUTINE COMRAT
C
C  COMPUTE THE UTILIZATION RATES.
C
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CRUNC.SIK'
      include 'CCHEM.SIK'
      include 'CPARAM.SIK'
C
      INTEGER I, J, NUT, SUB, OXY, NIT
      PARAMETER (NUT=1, SUB=2, OXY=3, NIT=4)
C
      DO 40, I=1,NSLXP1
       DO 40, J=1,NSLYP1
C
        RSONU1(I,J)=(MUO1/YSO1)*
     &              (COLD(I,J,SUB)/(KSO1 +COLD(I,J,SUB)))*
     &              (COLD(I,J,OXY)/(KO1  +COLD(I,J,OXY)))*
     &              (COLD(I,J,NUT)/(KONU1+COLD(I,J,NUT)))
```

```
C
      RSONU2(I,J)=(MUO2/YSO2)*
     &              (COLD(I,J,SUB)/(KSO2 +COLD(I,J,SUB)))*
     &              (COLD(I,J,OXY)/(KO2  +COLD(I,J,OXY)))*
     &              (COLD(I,J,NUT)/(KONU2+COLD(I,J,NUT)))
C
      RSNINU1(I,J)=(MUNI1/YSNI1)*
     &               (COLD(I,J,SUB)/(KSNI1 +COLD(I,J,SUB)))*
     &               (COLD(I,J,NIT)/(KNI1  +COLD(I,J,NIT)))*
     &               (COLD(I,J,NUT)/(KNINU1+COLD(I,J,NUT)))/
     &               (1.0D0 + COLD(I,J,OXY)/KONI1)
C
   40 CONTINUE
C
      RETURN
      END
```

---

```
*         File: RWCSIC.FOR           Last revision:  December 14, 1990
C                                    For chem init phase of LT3VSI.
      SUBROUTINE CHMREAD(HEAD1,HEAD2)
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CRUNC.SIK'
      include 'CBND.SIK'
      include 'CCHEM.SIC'
      include 'CCHEM.SIK'
      include 'CPARAM.SIK'
C
      CHARACTER HEAD1*(*), HEAD2*(*), STRING*75
      INTEGER I, J, K
      REAL CUFVER, XCUFVR
C
C  FORMATS
C
 1000 FORMAT(1X,1P,7G10.3)
 2000 FORMAT(A)
 2300 FORMAT(5X,A)
C
C  R/W TWO HEADER LINES, THEN READ RUN CONTROL INFORMATION
C
      WRITE(*,*)' Reading run control data. '
      WRITE(*,*)
C
```

```
      READ (1,2000) HEAD1
      WRITE(2,2300) HEAD1
C
      READ (1,2000) HEAD2
      WRITE(2,2300) HEAD2
C
      READ (1,2000) STRING
      READ(1,*,ERR=999) NPRT,TMAX,DT0
C
C  READ IN CONTROL FLAGS FOR BOTH CHEM INIT AND CHEM LOOP PHASES.
C
C*****************************************************************
C
C  NFLAG(4)=0 MEANS: DO NOT USE A SCHEDULE FILE "xxxxxxxx.SCH".
C                    THE INJECTION WELL CHEMICAL CONCENTRATIONS
C                    SPECIFIED IN THE CHEM DATA FILE "xxxxxxxx.CHD"
C                    WILL REMAIN UNCHANGED THROUGHOUT THE RUN.
C  NFLAG(4)=1 MEANS: READ A SCHEDULE FILE "xxxxxxxx.SCH" AND ALTER
C                    THE INJECTION WELL CHEMICAL CONCENTRATIONS AT
C                    THE SPECIFIED EVENT TIMES, AS SPECIFIED BY THE
C                    DATA IN THE FILE.
C
C  NFLAG(5)=0 MEANS: READ INITIAL CHEMICAL CONCENTRATION FROM THE
C                    CHEMICAL DATA FILE.
C  NFLAG(5)=1 MEANS: READ INITIAL CHEMICAL CONCENTRATION FROM THE
C                    UNFORMATTED FILE WRITTEN BY A PREVIOUS RUN
C                    OF THE CHEMISTRY LOOP PHASE.  SEE NFLAG(6) BELOW.
C
C  NFLAG(6)=1 MEANS: AT THE END OF A CHEMISTRY LOOP PHASE RUN, WRITE
C                    AN UNFORMATTED FILE CONTAINING THE FINAL CHEMICAL
C                    CONCENTRATION.  THIS FILE CAN BE READ IN BY
C                    A SUBSEQUENT CHEMISTRY RUN AS INITIAL DATA.
C                    SEE NFLAG(5) ABOVE.
C  NFLAG(6)=0 MEANS: DO NOT WRITE THE UNFORMATTED FILE DESCRIBED ABOVE.
C
C  NFLAG(8)=0 MEANS: WRITE TWO-DIMENSIONAL ARRAYS IN NARROW
C         (80 COLUMN) FORMAT.
C  NFLAG(8)=1 MEANS: WRITE TWO-DIMENSIONAL ARRAYS IN WIDE
C         (132 COLUMN) FORMAT.
C
C  NFLAG(9)=1 MEANS: READ A VALUE FOR ZTHRSH FROM THE CHEMICAL
C         DATA FILE, DURING THE CHEMISTRY PHASE.
C  NFLAG(9)=0 MEANS: DO NOT READ ZTHRSH DURING THE CHEMISTRY PHASE;
C         USE THE VALUE THAT WAS READ IN DURING THE WATER PHASE.
C
C  NFLAG(14)=1 MEANS: WRITE THE CHEMICAL FIELD DEFINING
C          MATRIX ELEMENTS.
C  NFLAG(14)=0 MEANS: DO NOT WRITE THE MATRIX ELEMENTS.
C
C  NFLAG(15)=1 MEANS: WRITE THE COORDINATES OF THE P* POINTS.
```

```
C  NFLAG(15)=0 MEANS: DO NOT WRITE THE ABOVE.
C
C  NFLAG(16)=0 MEANS: USE 4-POINT INTERPOLATION METHOD IN THE
C          CLAG FUNCTION, WHICH COMPUTES THE CHEMICAL
C          CONCENTRATION AT THE P* POINTS.
C  NFLAG(16)=1 MEANS: USE 25-POINT LAGRANGE INTERPOLATION METHOD
C          IN THE CLAG FUNCTION (THIS TAKES MUCH MORE TIME).
C
C  NFLAG(17)=1 MEANS: COMPUTE AND WRITE THE CUMULATIVE CHEMICAL
C          MASSES (XMASS, ET AL) ON THE FILE 'xxxxxxxx.CNC'.
C  NFLAG(17)=0 MEANS: DO NOT COMPUTE AND DO NOT WRITE THE ABOVE.
C
C  NOTE: THE OUTPUTS ENABLED BY NFLAGS 14 AND 15 ARE WRITTEN ON THE
C  FILE 'xxxxxxxx.CDB' DURING THE CHEMISTRY INITIALIZATION PHASE.
C
C*********************************************************************
C
      READ(1,2000) STRING
      READ(1,*,ERR=999) (NFLAG(I),I=1,20)
C
      IF (NFLAG(9) .NE. 0) THEN
        READ(1,2000) STRING
        READ(1,*,ERR=999) ZTHRSH
        ZTHRSH=MIN(ZTHRSH,0.99D0)
      ENDIF
C
      CALL GETCUF
C
C  READ THE INLET AND EXIT PORT TANK LENGTHS (M)
C
      WRITE(*,*)' Reading inlet and exit port tank lengths.'
      WRITE(*,*)
      READ(1,2000) STRING
      READ(1,*,ERR=999) XLYIN,XLYOUT
C
      WRITE(*,*)' Reading soil particle density.'
      WRITE(*,*)
C
C  READ SOIL PARTICLE DENSITY
C
      READ(1,2000) STRING
      READ(1,*,ERR=999) RHOSND,RHOCLA,RHOORG
C
C  READ POROUS MEDIUM CHARACTERIZING PARAMETERS.
C
      WRITE(*,*)' Reading porous medium characterizing parameters such '
      WRITE(*,*)' as tortuosity, porosity, etc.'
      WRITE(*,*)
C
      READ(1,2000) STRING
```

```
      DO 25, J=1,NSLYP1
         READ(1,*,ERR=999) (TORT(I,J),I=1,NSLXP1)
   25 CONTINUE
C
      READ(1,2000) STRING
      DO 26, J=1,NSLYP1
         READ(1,*,ERR=999) (EPS(I,J),I=1,NSLXP1)
   26 CONTINUE
C
      READ(1,2000) STRING
      DO 27, J=1,NSLYP1
         READ(1,*,ERR=999) (PCTSAN(I,J),I=1,NSLXP1)
   27 CONTINUE
C
      READ(1,2000) STRING
      DO 28, J=1,NSLYP1
         READ(1,*,ERR=999) (PCTCLA(I,J),I=1,NSLXP1)
   28 CONTINUE
C
      READ(1,2000) STRING
      DO 29, J=1,NSLYP1
         READ(1,*,ERR=999) (PCTORG(I,J),I=1,NSLXP1)
   29 CONTINUE
C
C  READ DISPERSIVITIES
C
      WRITE(*,*)' Reading dispersivities.'
      WRITE(*,*)
      DO 32, K=1,NC
         READ(1,2000) STRING
         DO 30, J=1,NSLYP1
            READ(1,*,ERR=999) (DISPLX(I,J,K),I=1,NSLXP1)
   30    CONTINUE
C
         READ(1,2000) STRING
         DO 31, J=1,NSLYP1
            READ(1,*,ERR=999) (DISPLY(I,J,K),I=1,NSLXP1)
   31    CONTINUE
   32 CONTINUE
C
C  READ INPUT CHEMICAL PARAMETERS
C
      WRITE(*,*)' Reading chemical parameters.'
      WRITE(*,*)
      READ(1,2000) STRING
      READ(1,*,ERR=999) (DL0(K),  K=1,NC)
      READ(1,*,ERR=999) (KSAND(K),K=1,NC)
      READ(1,*,ERR=999) (KCLAY(K),K=1,NC)
      READ(1,*,ERR=999) (KORG(K), K=1,NC)
C
```

```
C  READ INLET & EXIT PORT BOUNDARY CHEMICAL CONCENTRATIONS.
C
      WRITE(*,*)' Reading inlet and exit port boundary data.'
      WRITE(*,*)
      READ(1,2000) STRING
      READ(1,*,ERR=999) (CIN(K), K=1,NC)
      READ(1,*,ERR=999) (COUT(K),K=1,NC)
      READ(1,*,ERR=999) (C0(K),  K=1,NC)
C
C  READ FIRST ORDER LOSS PARAMETERS.
C
      WRITE(*,*)' Reading first order loss parameters.'
      WRITE(*,*)
      DO 42, K=1,NC
         READ(1,2000) STRING
         DO 42, J=1,NSLYP1
            READ(1,*,ERR=999) (XLAMIR(I,J,K),I=1,NSLXP1)
   42 CONTINUE
C
      DO 48, K=1,NC
         READ(1,2000) STRING
         DO 48, J=1,NSLYP1
            READ(1,*,ERR=999) (XSLMIR(I,J,K),I=1,NSLXP1)
   48 CONTINUE
C
C  READ INITIAL CHEMICAL DISTRIBUTION AT TIME ZERO.
C
      WRITE(*,*)' Reading initial chem. distribution. '
      WRITE(*,*)
      T0=0.0D0
      DO 50, K=1,NC
         XMASS(K) =0.0D0
         XMFONW(K)=0.0D0
         XMSOUR(K)=0.0D0
         XMASIN(K)=0.0D0
         XMASOT(K)=0.0D0
   50 CONTINUE
      DO 54, K=1,NC
         READ(1,2000) STRING
         DO 54, J=1,NSLYP1
            READ(1,*,ERR=999) (COLD(I,J,K),I=1,NSLXP1)
   54 CONTINUE
C
C  READ INITIAL MICROBIAL POPULATIONS.
C
      WRITE(*,*)' Reading initial microbial populations.'
      WRITE(*,*)
      READ(1,2000) STRING
      DO 58, J=1,NSLYP1
         READ(1,*,ERR=999) (POP1(I,J),I=1,NSLXP1)
```

```
   58 CONTINUE
C
      READ(1,2000) STRING
      DO 62, J=1,NSLYP1
         READ(1,*,ERR=999) (POP2(I,J),I=1,NSLXP1)
   62 CONTINUE
C
C  IF NFLAG(5) <> 0, READ CHEMICAL CONCENTRATIONS AND MICROBIAL
C  POPULATIONS FROM CUF FILE, OVERWRITING VALUES FROM CHD FILE.
C
      IF (NFLAG(5).NE.0) THEN
         WRITE(*,*)' Reading unformatted chem file (CUF).'
         WRITE(*,*)
         CUFVER=132.0
         READ(10) XCUFVR
         IF (XCUFVR.NE.CUFVER) THEN
            WRITE(*,*) 'Unformatted chem file is version ',XCUFVR
            WRITE(*,*) 'It should be version ',CUFVER
            STOP 1
         ENDIF
         READ(10) I,J,T0
         IF (I.NE.NSLXP1 .OR. J.NE.NSLYP1) THEN
            WRITE(*,3000) I-2,J-2,NSLXM1,NSLYM1
            STOP1
         ENDIF
         TMAX=TMAX+T0
         DO 80, K=1,NC
            READ(10) CIN(K),COUT(K),XMASS(K),XMFONW(K),XMSOUR(K),
     &                  XMASIN(K),XMASOT(K)
            DO 80, J=1,NSLYP1
               READ(10) (COLD(I,J,K),I=1,NSLXP1)
   80    CONTINUE
         DO 90, J=1,NSLYP1
            READ(10) (POP1(I,J),I=1,NSLXP1)
            READ(10) (POP2(I,J),I=1,NSLXP1)
   90    CONTINUE
         CLOSE(UNIT=10)
      ENDIF
C
C  READ CHEMISTRY USAGE PARAMETERS.
C
      WRITE(*,*)' Reading chemistry usage parameters.'
      WRITE(*,*)
      READ(1,2000) STRING
      READ(1,*,ERR=999) KSO1
      READ(1,*,ERR=999) KSO2
      READ(1,*,ERR=999) KO1
      READ(1,*,ERR=999) KO2
      READ(1,*,ERR=999) KONU1
      READ(1,*,ERR=999) KONU2
```

```
      READ(1,*,ERR=999) KSNI1
      READ(1,*,ERR=999) KNI1
      READ(1,*,ERR=999) KNINU1
      READ(1,*,ERR=999) KONI1
      READ(1,*,ERR=999) KSOM1
      READ(1,*,ERR=999) KSOM2
      READ(1,*,ERR=999) YSO1
      READ(1,*,ERR=999) YSO2
      READ(1,*,ERR=999) YSNI1
      READ(1,*,ERR=999) ALFO1
      READ(1,*,ERR=999) ALFO2
      READ(1,*,ERR=999) ALFNI1
      READ(1,*,ERR=999) ETANI1
      READ(1,*,ERR=999) GAMMO1
      READ(1,*,ERR=999) GAMMO2
      READ(1,*,ERR=999) PSIO1
      READ(1,*,ERR=999) PSIO2
      READ(1,*,ERR=999) THENI1
      READ(1,*,ERR=999) MUO1
      READ(1,*,ERR=999) MUO2
      READ(1,*,ERR=999) MUNI1
      READ(1,*,ERR=999) RHOBD
C
C  COMPUTE AND DISPLAY PRINT TIMES
C
      WRITE(*,*)' Estimated print times are as follows:'
      WRITE(*,*)
      WRITE(*,1000)(((TMAX-T0)/NPRT)*I + T0 - DT0*1.0D-2,I=1,NPRT)
      WRITE(*,*)
C
      RETURN
C
C  HANDLE DATA ERRORS.
C
  999 CALL DATERR(STRING,FILBAS,'CHD')
      STOP 1
C
 3000 FORMAT(' Aquifer dimensions in unformatted chem file are ',/,
     &   1X,I3,' by ',I3,' (# interior nodes); they should be',/,
     &   I3,' by ',I3)
      END
C
C ****************************************
C
      SUBROUTINE CHMOUT(DTRAT)
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CRUNC.SIK'
```

```
      include 'CBND.SIK'
      include 'CCHEM.SIC'
      include 'CCHEM.SIK'
      include 'CPARAM.SIK'
C
      INTEGER I, K
      DOUBLE PRECISION DTRAT
C
C  WRITE OUT RUN CONTROL INFORMATION.
C
      WRITE(2,98765)
98765 FORMAT(/,
     &  5X,'** NOTE!  SI units are indicated, but any units  ** ',
     &  /,5X,'** can be used, as long as they are CONSISTENT.  **')
C
      WRITE(2,3)
    3 FORMAT(/,10X,'RUN CONTROL INFORMATION.',/)
C
      WRITE(2,4) T0,TMAX,DT0
    4 FORMAT(10X,'TIMES IN DAYS:',/,
     &   1P,5X,'T0= ',G10.3,2X,'TMAX= ',G10.3,2X,'DT0= ',G10.3)
C
      IF (DTRAT.LT.1.0D0) WRITE(2,14)
   14 FORMAT(10X,'DT0 AND TMAX HAVE BEEN REDUCED TO ENSURE STABILITY')
C
      WRITE(2,5) (CHNAME(K),K=1,NC), (DT1(K),K=1,NC)
    5 FORMAT(/,5X,'COMPUTED',4(3X,A9),/,
     &         5X,'  DTMAX=',1P,4(2X,G10.3),/)
C
      WRITE(2,6) NPRT
    6 FORMAT(5X,'NPRT= ',I5,';  PRINT TIMES (DAYS) ARE:',/)
C
      WRITE(2,7)(((TMAX-T0)/NPRT)*I + T0 - DT0*1.0D-2,I=1,NPRT)
    7 FORMAT(1X,1P,7G10.3)
C
      WRITE(2,8) ZTHRSH
    8 FORMAT(/,5X,'ZTHRSH= ',G12.5)
C
      WRITE(2,9) (I,I=1,20),(NFLAG(I),I=1,20)
    9 FORMAT(/,11X,'I ',20I3,/,5X,'NFLAG(I)',20I3,/)
C
C  VALIDATE AND LIST SCHEDULE FILE
C
      IF (NFLAG(4).NE.0 .AND. NINJW.NE.0) CALL VALSCH
C
C  WRITE OUT INLET & EXIT PORT TANK LENGTHS.
C
      WRITE(2,20) XLYIN,XLYOUT
   20 FORMAT(/,1P,5X,'INLET PORT TANK LENGTH (M)= ',G10.3,/,
     &           5X,'EXIT  PORT TANK LENGTH (M)= ',G10.3,/)
```

```
C
C  WRITE OUT SOIL CHARACTERIZING PARAMETERS.
C
      WRITE(2,30)
   30 FORMAT(/,10X,'BASIC SOIL CHARACTERIZING PARAMETERS (KG/M^3)',/)
      WRITE(2,43) RHOSND,RHOCLA,RHOORG
   43 FORMAT(1P,5X,'RHOSND= ',G10.3,2X,'RHOCLA= ',G10.3,2X,
     & 'RHOORG=',G10.3,/)
C
      WRITE(2,220)
  220 FORMAT(/,10X,'TORTUOSITY FACTOR TORT(I,J) (DIMLESS)',/)
      CALL PRIN3S(2,NSLXP1,NSLYP1,TORT)
C
      WRITE(2,230)
  230 FORMAT(/,10X,'POROSITY EPS(I,J) (M^3/M^3)',/)
      CALL PRIN3S(2,NSLXP1,NSLYP1,EPS)
C
      WRITE(2,240)
  240 FORMAT(/,10X,'PERCENT SAND PCTSAN(I,J) (DIMLESS)',/)
      CALL PRIN3S(2,NSLXP1,NSLYP1,PCTSAN)
C
      WRITE(2,250)
  250 FORMAT(/,10X,'PERCENT SILT (CLAY) PCTCLA(I,J) (DIMLESS)',/)
      CALL PRIN3S(2,NSLXP1,NSLYP1,PCTCLA)
C
      WRITE(2,260)
  260 FORMAT(/,10X,'PERCENT ORGANICS PCTORG(I,J) (DIMLESS)',/)
      CALL PRIN3S(2,NSLXP1,NSLYP1,PCTORG)
C
      WRITE(2,266)
  266 FORMAT(/,10X,'DISPERSIVITY COMPONENTS DISPLX(I,J,K) ',
     &       'AND DISPLY(I,J,K)')
      DO 284, K=1,NC
         WRITE(2,270) CHNAME(K)
  270    FORMAT(/,10X,'X-COMP DISPERSIVITY FOR ',A,' (M)',/)
         CALL PRIN3S(2,NSLXP1,NSLYP1,DISPLX(1,1,K))
C
         WRITE(2,280) CHNAME(K)
  280    FORMAT(/,10X,'Y-COMP DISPERSIVITY FOR ',A,' (M)',/)
         CALL PRIN3S(2,NSLXP1,NSLYP1,DISPLY(1,1,K))
  284 CONTINUE
C
C  WRITE OUT CHEMICAL PARAMETERS.
C
      WRITE(2,300)
  300 FORMAT(/,10X,'CHEMICAL PARAMETERS',/)
      WRITE(2,302) (CHNAME(K),K=1,NC)
  302 FORMAT(9X,4(3X,A9),/)
      WRITE(2,304) (DL0(K),K=1,NC)
  304 FORMAT(1P,5X,'DL0= ',4(1X,G10.3,1X),' (M^2/DAY)',/)
```

```
      WRITE(2,306) 'KSAND=',(KSAND(K),K=1,NC),'(M^3/KG SAND)',
     &             'KCLAY=',(KCLAY(K),K=1,NC),'(M^3/KG SILT)',
     &             'KORG= ',(KORG(K) ,K=1,NC),'(M^3/KG ORGANICS)'
  306 FORMAT(3(1P,5X,A,4(1X,G10.3,1X),A,/))
C
C  WRITE THE INLET AND EXIT BOUNDARY CHEMICAL CONCENTRATIONS.
C
      WRITE(2,306) 'CIN=  ',(CIN(K), K=1,NC),'(KG/M^3)',
     &             'COUT= ',(COUT(K),K=1,NC),'(KG/M^3)',
     &             'C0=   ',(C0(K),  K=1,NC),'(KG/M^3)'
C
C  WRITE FIRST ORDER LOSSES.
C
      WRITE(2,*)
      WRITE(2,320)
  320 FORMAT(10X,'FIRST ORDER LOSS COEFS ',
     & 'XLAMIR(I,J,K) & XSLMIR(I,J,K)',/)
C
      DO 326, K=1,NC
         WRITE(2,324) CHNAME(K)
  324    FORMAT(/,10X,'IRREVERSIBLE LOSS IN FREE PHASE FOR ',A,
     &          ' (1/DAY)',/)
         CALL PRIN3S(2,NSLXP1,NSLYP1,XLAMIR(1,1,K))
  326 CONTINUE
C
      DO 333, K=1,NC
         WRITE(2,331) CHNAME(K)
  331    FORMAT(/,10X,'IRREVERSIBLE LOSS IN SORBED PHASE FOR ',A,
     &          ' (1/DAY)',/)
         CALL PRIN3S(2,NSLXP1,NSLYP1,XSLMIR(1,1,K))
  333 CONTINUE
C
C  WRITE INITIAL DISTRIBUTION OF CHEMICALS.
C
      DO 347, K=1,NC
         WRITE(2,345) CHNAME(K)
  345    FORMAT(/,10X,'INITIAL CHEMICAL DISTRIBUTION OF ',A,
     &          ' (KG/M^3)',/)
         CALL PRIN3S(2,NSLXP1,NSLYP1,COLD(1,1,K))
  347 CONTINUE
C
C  WRITE RETARDATION AND OVER ALL LOSS.
C
      DO 350, K=1,NC
         WRITE(2,9100) CHNAME(K)
 9100    FORMAT(/,10X, '1+RETENTION(I,J,K) FOR ',A,' (DIMLESS)',/)
         CALL PRIN3S(2,NSLXP1,NSLYP1,RETARD(1,1,K))
  350 CONTINUE
C
      DO 354, K=1,NC
```

```
          WRITE(2,9200) CHNAME(K)
 9200     FORMAT(/,10X, 'OVER ALL FIRST ORDER LOSS',
     &           ' COEF LAMDA(I,J,K) FOR ',A,' (1/DAY)',/)
          CALL PRIN3S(2,NSLXP1,NSLYP1,LAMDA(1,1,K))
  354 CONTINUE
C
C  WRITE INITIAL MICROBIAL POPULATIONS.
C
      WRITE(2,9300)
 9300 FORMAT(/,10X, 'INITIAL DISTRIBUTION OF MICROBE POPULATION #1 ',
     &       '(KG CELLS/KG SOIL)',/)
      CALL PRIN3S(2,NSLXP1,NSLYP1,POP1)
C
      WRITE(2,9400)
 9400 FORMAT(/,10X, 'INITIAL DISTRIBUTION OF MICROBE POPULATION #2 ',
     &       '(KG CELLS/KG SOIL)',/)
      CALL PRIN3S(2,NSLXP1,NSLYP1,POP2)
C
C  WRITE CHEMISTRY USAGE PARAMETERS.
C
      WRITE(2,4300)
 4300 FORMAT(/,10X,'CHEMISTRY USAGE PARAMETERS',/)
 4400 FORMAT(5X,1P,A6,' = ',G11.4,1X,A)
      WRITE(2,4400) 'KSO1  ',KSO1  ,'(KG/M^3)'
      WRITE(2,4400) 'KSO2  ',KSO2  ,'(KG/M^3)'
      WRITE(2,4400) 'KO1   ',KO1   ,'(KG/M^3)'
      WRITE(2,4400) 'KO2   ',KO2   ,'(KG/M^3)'
      WRITE(2,4400) 'KONU1 ',KONU1 ,'(KG/M^3)'
      WRITE(2,4400) 'KONU2 ',KONU2 ,'(KG/M^3)'
      WRITE(2,4400) 'KSNI1 ',KSNI1 ,'(KG/M^3)'
      WRITE(2,4400) 'KNI1  ',KNI1  ,'(KG/M^3)'
      WRITE(2,4400) 'KNINU1',KNINU1,'(KG/M^3)'
      WRITE(2,4400) 'KONI1 ',KONI1 ,'(KG/M^3)'
      WRITE(2,4400) 'KSOM1 ',KSOM1 ,'(KG/M^3)'
      WRITE(2,4400) 'KSOM2 ',KSOM2 ,'(KG/M^3)'
      WRITE(2,4400) 'YSO1  ',YSO1  ,'(KG CELLS/KG SUB)'
      WRITE(2,4400) 'YSO2  ',YSO2  ,'(KG CELLS/KG SUB)'
      WRITE(2,4400) 'YSNI1 ',YSNI1 ,'(KG CELLS/KG SUB)'
      WRITE(2,4400) 'ALFO1 ',ALFO1 ,'(KG OXY/KG CELLS)'
      WRITE(2,4400) 'ALFO2 ',ALFO2 ,'(KG OXY/KG CELLS)'
      WRITE(2,4400) 'ALFNI1',ALFNI1,'(KG NIT/KG CELLS)'
      WRITE(2,4400) 'ETANI1',ETANI1,'(KG NIT/KG SUB)'
      WRITE(2,4400) 'GAMMO1',GAMMO1,'(KG OXY/KG SUB)'
      WRITE(2,4400) 'GAMMO2',GAMMO2,'(KG OXY/KG SUB)'
      WRITE(2,4400) 'PSIO1 ',PSIO1 ,'(KG NUT/KG SUB)'
      WRITE(2,4400) 'PSIO2 ',PSIO2 ,'(KG NUT/KG SUB)'
      WRITE(2,4400) 'THENI1',THENI1,'(KG NUT/KG SUB)'
      WRITE(2,4400) 'MUO1  ',MUO1  ,'(1/DAY)'
      WRITE(2,4400) 'MUO2  ',MUO2  ,'(1/DAY)'
      WRITE(2,4400) 'MUNI1 ',MUNI1 ,'(1/DAY)'
```

```
      WRITE(2,4400) 'RHOBD ',RHOBD ,'(KG/M^3)'
C
      RETURN
      END
```

---

```
*         File: RWCSIL.FOR              Last revision:  August 29, 1990
C                                       For chem loop phase of LT3VSI.
      SUBROUTINE LOOPIO
C
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CRUNC.SIK'
      include 'CCHEM.SIL'
C
      CHARACTER STRING*75
      DOUBLE PRECISION DUMMY, QTEMP(NC), XI, YJ
      INTEGER I, J, K, KKK, NBSOUR
      LOGICAL NONE, NZERO
      EXTERNAL NZERO
C
C  FORMATS
C
 2000 FORMAT(A)
 2300 FORMAT(5X,A)
C
C  INITIALIZE ARRAYS FOR CHEMICAL SOURCES
C
      DO 10, I = 1, NSLXP1
         DO 10, J = 1, NSLYP1
            DO 10, K=1, NC
               QCHM1S(I,J,K) = 0.D0
               CSWELN(I,J,K) = 0.D0
   10 CONTINUE
C
C  READ AND WRITE INJECTION WELL CHEMICAL CONCENTRATIONS (KG/KG).
C
      WRITE(*,*)' Reading injection wells chem. conc. '
**      WRITE(*,*)' Minimum number is one well of strength zero.'
      WRITE(2,4000)
 4000 FORMAT(/,5X,'INJECTION WELLS CHEMICAL CONC. CSWELN(XI,YJ,K) ',
     &    '(KG CHEM/KG SOLUTION)',/)
      WRITE(2,4010) (CHNAME(K),K=1,NC)
 4010 FORMAT(T10,'XI',T21,'YJ',T30,A,T42,A,T53,A,T64,A,/,5X,70('-'))
      NONE=.TRUE.
      READ(1,2000) STRING
```

```
      READ(1,2000) STRING
      DUMMY = 0.0D0
      DO 121, KKK = 1, NINJW
         READ (1,*,ERR=999) XI, YJ, (QTEMP(K),K=1,NC)
         CALL TESTIJ(STRING,FILBAS,'CHD',XI,YJ,DUMMY,I,J)
         IF (NZERO(NC,QTEMP)) THEN
            NONE=.FALSE.
            WRITE(2,509) XI, YJ, (QTEMP(K),K=1,NC)
         ENDIF
         DO 121, K=1,NC
            CSWELN(I,J,K) = QTEMP(K)
  121 CONTINUE
      IF (NONE) THEN
         WRITE(2,511)
      ENDIF
  509 FORMAT(5X,1P,2(G10.3,1X),4(1X,G11.4))
  511 FORMAT(5X,'NO NON-ZERO INJECTION WELLS.')
C
C  READ NUMBER OF BURIED CHEMICAL SOURCES;
C  THEN READ AND WRITE THEIR CONCENTRATIONS.
C
      READ(1,2000) STRING
      READ(1,*,ERR=999)  NBSOUR
C
      WRITE(*,*)' Reading buried source positions and strengths- '
      WRITE(*,*)' Minimum is one source of strength zero.'
      WRITE(*,*)
      WRITE(2,400)
  400 FORMAT(/,5X,'BURIED CHEMICAL SOURCE CONC. QCHM1S(XI,YJ,K) ',
     &    '(KG CHEM/M^3 DAY)',/)
      WRITE(2,4010) (CHNAME(K),K=1,NC)
      NONE=.TRUE.
      READ (1,2000) STRING
      READ (1,2000) STRING
      DO 305, KKK = 1, NBSOUR
         READ(1,*,ERR=999) XI, YJ, (QTEMP(K),K=1,NC)
         DO 300, K=1, NC
            CALL TESTIJ(STRING,FILBAS,'CHD',XI,YJ,QTEMP(K),I,J)
  300    CONTINUE
         IF (NZERO(NC,QTEMP)) THEN
            NONE=.FALSE.
            WRITE(2,509) XI, YJ, (QTEMP(K),K=1,NC)
         ENDIF
         DO 305, K=1,NC
            QCHM1S(I,J,K) = QTEMP(K)
  305 CONTINUE
      IF (NONE) WRITE(2,510)
  510 FORMAT(5X,'NO NON-ZERO BURIED SOURCES.')
C
      RETURN
```

```
C
C  HANDLE DATA ERRORS.
C
  999 CALL DATERR(STRING,FILBAS,'CHD')
      STOP 1
C
      END
C
C ****************************************
C
      LOGICAL FUNCTION NZERO(N,ARY)
C
C  RETURNS .TRUE. IF AT LEAST ONE ELEMENT OF ARRAY ARY OF SIZE N IS
C  NON-ZERO.  RETURNS .FALSE. IF ALL ELEMENTS ARE ZERO.
C
      INTEGER I,N
      DOUBLE PRECISION ARY(N)
C
      NZERO=.FALSE.
      DO 20, I=1,N
         IF (ARY(I) .NE. 0D0) NZERO=.TRUE.
20    CONTINUE
      RETURN
      END
```

---

```
*         File: RWWSIW.FOR           Last revision:  August 29, 1990
C                                    For water phase of LT3VSI.
      SUBROUTINE FLOREAD
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CSSIP.SIW'
      include 'CWAT.SIW'
C
      CHARACTER*80 STRING
      INTEGER I, J, KKK
      DOUBLE PRECISION QTEMP, RAT, XI, YJ
C
C  FORMATS
C
 2000 FORMAT(A)
C
C  READ RUN CONTROL INFORMATION
C
```

```
      WRITE(*,*)' Reading run control information.'
      WRITE(*,*)
      READ(1,2000) HEAD1,HEAD2
      READ(1,2000) STRING
      READ(1,*,ERR=999) NLSOR,NMOD,NALPH,TLRNWA,TLRNWR,ZTHRSH
      ZTHRSH=MIN(ZTHRSH,0.99D0)
C
C  READ IN CONTROL FLAGS FOR WATER PHASE.
C
C*****************************************************************
C
C  NFLAG(1)=0 MEANS: COMPUTE THE HYDRAULIC PRESSURE FIELD ONLY.
C         NOTE THAT THE CHEMISTRY PHASE CANNOT BE RUN IN THIS CASE.
C  NFLAG(1)=1 MEANS: COMPUTE THE HYDRAULIC PRESSURE FIELD AND THE
C         VELOCITY COMPONENTS.
C
C  NFLAG(2)=0 MEANS: COMPUTE AN INITIAL GUESS FOR THE HYDRAULIC
C         PRESSURE.
C  NFLAG(2)=1 MEANS: READ AN INITIAL GUESS FOR THE HYDRAULIC PRESSURE
C         FROM THE WATER DATA FILE.
C
C  NFLAG(3)=0 MEANS: DISPLAY ICOUNT, ALPH, RMSEA, RMSER, DURING THE
C         SIP PROCESS.
C  NFLAG(3)=1 MEANS: DISPLAY THE ABOVE INFORMATION, PLUS THE VALUES
C         OF XDUM AND DELX AT THE FOUR "INTERIOR" CORNERS.
C
C  NFLAG(8)=0 MEANS: WRITE TWO-DIMENSIONAL ARRAYS IN NARROW
C         (80 COLUMN) FORMAT.
C  NFLAG(8)=1 MEANS: WRITE TWO-DIMENSIONAL ARRAYS IN WIDE
C         (132 COLUMN) FORMAT.
C
C  NFLAG(10)=1 MEANS: WRITE THE MATRIX ELEMENTS WHICH DEFINE THE
C         HYDRAULIC PRESSURE FIELD.
C  NFLAG(10)=0 MEANS: DO NOT WRITE THE MATRIX ELEMENTS.
C
C  NFLAG(11)=1 MEANS: WRITE THE ADJUSTED BOUNDARY MATRIX ELEMENTS.
C  NFLAG(11)=0 MEANS: DO NOT WRITE THE ADJUSTED ELEMENTS.
C
C  NFLAG(12)=1 MEANS: WRITE THE SOURCE AND BOUNDARY COMPONENT
C          CONTRIBUTIONS TO THE OVER ALL "KNOWN VECTOR".
C  NFLAG(12)=0 MEANS: DO NOT WRITE THE "KNOWN VECTOR".
C
C  NFLAG(13)=1 MEANS: WRITE THE MAX AND MIN DELX'S.
C  NFLAG(13)=0 MEANS: DO NOT WRITE THE DELX'S.
C
C  NOTE: THE OUTPUTS ENABLED BY NFLAGS 10, 11, 12, AND 13 ARE
C  WRITTEN ON THE FILE 'xxxxxxxx.WDB' DURING THE WATER PHASE.
C
C*****************************************************************
C
```

```
      READ(1,2000) STRING
      READ(1,*,ERR=999) (NFLAG(I),I=1,20)
C
C  READ IN ALPHAS (AN ARRAY OF ALPH).  MUST HAVE 1 <= NALPH <= 10
C
      NALPH = MIN(NALPH,10)
      NALPH = MAX(NALPH, 1)
      READ(1,2000) STRING
      READ(1,*,ERR=999) (ALPHAS(I), I = 0, NALPH-1)
C
C  READ IN GRID GEOMETRY
C
      WRITE(*,*) ' Reading grid geometry.'
      WRITE(*,*)
C
C  FIRST, READ IN NUMBER OF INTERIOR X AND Y NODES.
C
      READ(1,2000) STRING
      READ(1,*,ERR=999) NSLXM1,NSLYM1
      NSLXXX=NSLXM1+1
      NSLYYY=NSLYM1+1
      NSLXP1 = NSLXXX+1
      NSLYP1 = NSLYYY+1
C
C  MAKE SURE GRID FITS WITHIN FIXED ARRAY SIZES.
C
      IF(NSLXP1.GT.IX .OR. NSLYP1.GT.IY) THEN
        WRITE(*,*) ' ************************************'
        WRITE(*,*) ' Too many grid points for fixed-size arrays.'
        WRITE(*,*) ' Change IX and/or IY, re-compile, & re-link.'
        STOP 1
      ENDIF
C
C  INITIALIZATION OF ARRAYS (NOW THAT WE HAVE SIZE OF PROBLEM)...
C
      DO 118, I = 1, NSLXP1
         DO 119, J = 1, NSLYP1
            QWELIN(I,J) = 0.D0
            QWELOT(I,J) = 0.D0
  119    CONTINUE
  118 CONTINUE
C
C  NEXT, READ IN NODAL POSITIONS
C
      READ(1,2000) STRING
      READ(1,*,ERR=999) (XNODE(I),I=1,NSLXP1)
      READ(1,2000) STRING
      READ(1,*,ERR=999) (YNODE(I),I=1,NSLYP1)
C
C  NEXT, READ IN THE AQUIFER VERTICAL THICKNESS (M)
```

```
C
      READ(1,2000) STRING
      READ(1,*,ERR=999) XLW
C
C  CALCULATE THE SPACING BETWEEN THE NODES
C
      DO 5, J=1,NSLYYY
         DY(J)=YNODE(J+1)-YNODE(J)
    5 CONTINUE
      DO 6, I=1,NSLXXX
         DX(I)=XNODE(I+1)-XNODE(I)
    6 CONTINUE
C
C  CALCULATE CONSTANTS USED FOR LAGRANGE INTERPOLATION ALONG THE
C  SIDES OF THE AQUIFER.
C
      RAT=DX(2)/DX(1)
      CONST2=1.0D0/(RAT*(2.0D0+RAT))
      CONST1=1.0D0+CONST2
      RAT=DX(NSLXM1)/DX(NSLXXX)
      CONST4=1.0D0/(RAT*(2.0D0+RAT))
      CONST3=1.0D0+CONST4
C
      WRITE(*,*)' Reading water density.'
      WRITE(*,*)
C
C  READ WATER DENSITY
C
      READ(1,2000) STRING
      READ(1,*,ERR=999) RHOWAT
C
C  READ HYDRAULIC CONDUCTIVITY.
C
      WRITE(*,*)' Reading hydraulic conductivity.'
      WRITE(*,*)
      READ(1,2000) STRING
      DO 13, J=1,NSLYP1
         READ(1,*,ERR=999) (KSATXX(I,J),I=1,NSLXP1)
   13 CONTINUE
C
      READ(1,2000) STRING
      DO 14, J=1,NSLYP1
         READ(1,*,ERR=999) (KSATYY(I,J),I=1,NSLXP1)
   14 CONTINUE
C
C  READ IN NUMBER OF INJECTION AND EXTRACTION WELL POSITIONS.
C  A MINIMUM OF ONE IS REQUIRED, BUT ITS STRENGTH MAY BE ZERO.
C
      WRITE(*,*) ' Reading number of injection and extraction'
      WRITE(*,*) ' well positions, NINJW and NEXTW ...'
```

```
      WRITE(*,*)
      READ (1,2000) STRING
      READ (1,*,ERR=999) NINJW, NEXTW
C
C  READ IN INJECTION AND EXTRACTION WELLS STRENGTH.
C
      WRITE(*,*) ' Reading injection and extraction well strengths -'
      WRITE(*,*) ' minimum is one well of zero strength (KG WATER/DAY)'
      WRITE(*,*)
      READ (1,2000) STRING
      DO 121, KKK = 1, NINJW
         READ (1,*,ERR=999) XI, YJ, QTEMP
         CALL TESTIJ(STRING,FILBAS,'WAD',XI,YJ,QTEMP,I,J)
         QWELIN(I,J) = QTEMP
  121 CONTINUE
      READ (1,2000) STRING
      DO 122, KKK = 1, NEXTW
         READ (1,*,ERR=999) XI, YJ, QTEMP
         CALL TESTIJ(STRING,FILBAS,'WAD',XI,YJ,QTEMP,I,J)
         QWELOT(I,J) = QTEMP
  122 CONTINUE
C
C  READ IN INLET AND OUTLET BOUNDARY HYDRAULIC HEADS
C
      WRITE(*,*)' Reading inlet & outlet hydraulic pressures (METERS).'
      WRITE(*,*)
      READ(1,2000) STRING
      READ(1,*,ERR=999) HIN,HOUT
C
C  READ INITIAL HYDRAULIC PRESSURE, OR COMPUTE AN INITIAL GUESS.
C
      IF(NFLAG(2).NE.0)THEN
         WRITE(*,*) 'Reading initial hydraulic pressure.'
         READ(1,2000) STRING
         DO 30, J=1,NSLYP1
            READ(1,*,ERR=999) (HOLD(I,J),I=1,NSLXP1)
   30    CONTINUE
      ELSE
         DO 32, I=1,NSLXP1
            DO 32, J=1,NSLYP1
               HOLD(I,J)=HIN+(HOUT-HIN)*(YNODE(J)/YNODE(NSLYP1))
   32    CONTINUE
      ENDIF
C
      RETURN
C
C  HANDLE DATA ERRORS.
C
  999 CALL DATERR(STRING,FILBAS,'WAD')
      STOP 1
```

```
C
      END
C
C ***************************************
C
      SUBROUTINE FLOOUT
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CSSIP.SIW'
      include 'CWAT.SIW'
C
      INTEGER I, J
      LOGICAL NONE
C
C  WRITE OUT RUN CONTROL INFORMATION.
C
      WRITE(2,2) HEAD1,HEAD2
      WRITE(3,2) HEAD1,HEAD2
    2 FORMAT(2(1X,A,/))
      WRITE(2,98765)
98765 FORMAT(5X,'** NOTE!  SI units are indicated, but any units  ** ',
     &  /,5X,'** can be used, as long as they are CONSISTENT.  **',/)
C
      WRITE(2,3) NLSOR,NMOD,NALPH,TLRNWA,TLRNWR,ZTHRSH
    3 FORMAT(1P,5X,'RUN CONTROL INFORMATION.',/,
     & T6,'NLSOR=  ',I12, T28, 'NMOD=   ', I12, T50, 'NALPH=  ', I12,/,
     & T6,1P,'TLRNWA= ',G12.5, T28,'TLRNWR= ',G12.5,
     &   T50,'ZTHRSH= ',G12.5,/)
C
      WRITE(2,7) (I,I=1,20),(NFLAG(I),I=1,20)
    7 FORMAT(11X,'I ',20I3,/,5X,'NFLAG(I)',20I3,/)
      WRITE(2,77) (ALPHAS(I), I = 0, NALPH-1)
   77 FORMAT(1X,'ALPHAS (array of ALPH''s)',/,(1X,1P,5(G12.5,2X)))
C
C  GEOMETRICAL PARAMETERS.
C
      WRITE(2,8)
    8 FORMAT(/,10X,'NODE COORDINATES (M)',/)
C
      WRITE(2,9)
    9 FORMAT(5X,'XNODE(I)')
      CALL PRINT2(2,NSLXP1,XNODE)
C
      WRITE(2,11)
   11 FORMAT(/,5X,'YNODE(J)')
      CALL PRINT2(2,NSLYP1,YNODE)
C
      WRITE(2,15)
```

```
   15 FORMAT(/,5X,'DX(I)')
      CALL PRINT2(2,NSLXXX,DX)
C
      WRITE(2,17)
   17 FORMAT(/,5X,'DY(J)')
      CALL PRINT2(2,NSLYYY,DY)
C
      WRITE(2,18) XNODE(NSLXP1),YNODE(NSLYP1),XLW
   18 FORMAT(1P,/,5X,'WIDTH     OF AQUIFER (M)=',G10.3,/,
     &           5X,'LENGTH    OF AQUIFER (M)=',G10.3,/,
     &           5X,'THICKNESS OF AQUIFER (M)=',G10.3,/)
C
C  WRITE OUT THE INLET AND EXIT BOUNDARY HYDRAULIC HEADS
C
      WRITE(2,507) HIN,HOUT
  507 FORMAT(5X,'INLET/OUTLET HYDRAULIC PRESSURES (M WATER)',/
     &   1P,5X,'HIN= ',G12.5,2X,'HOUT= ',G12.5)
C
C  WRITE OUT SOIL CHARACTERIZING PARAMETERS.
C
      WRITE(2,30)
   30 FORMAT(/,10X,'BASIC SOIL CHARACTERIZING PARAMETERS (KG/M^3)',/)
      WRITE(2,53) RHOWAT
   53 FORMAT(1P,5X,'RHOWAT= ',G10.3,/)
      WRITE(2,3095)
 3095 FORMAT(10X,'TABLE OF SOIL PROPERTIES',/)
C
      WRITE(2,300)
  300 FORMAT(/,10X,'X-COMP HYDRAULIC CONDUCTIVITY KSATXX(I,J) (M/DAY)',
     &       /)
      CALL PRIN3S(2,NSLXP1,NSLYP1,KSATXX)
C
      WRITE(2,310)
  310 FORMAT(/,10X,'Y-COMP HYDRAULIC CONDUCTIVITY KSATYY(I,J) (M/DAY)',
     &       /)
      CALL PRIN3S(2,NSLXP1,NSLYP1,KSATYY)
C
      WRITE(2,210)
  210 FORMAT(/,5X,'TABLE OF INJECTION WELLS (KG WATER/M^3 DAY)',/)
      WRITE(2,211)
  211 FORMAT(5X,'    XI          YJ     QWELIN(XI,YJ)',/,5X,36('-'))
      NONE=.TRUE.
      DO 212, J=1,NSLYP1
         DO 212, I=1,NSLXP1
            IF (QWELIN(I,J) .NE. 0.0D0) THEN
               NONE=.FALSE.
               WRITE(2,509) XNODE(I),YNODE(J),QWELIN(I,J)
            ENDIF
  212 CONTINUE
      IF (NONE) THEN
```

```
         WRITE(2,510) 'INJECTION'
         NINJW=0
      ENDIF
C
      WRITE(2,215)
  215 FORMAT(//,5X,'TABLE OF EXTRACTION WELLS (KG WATER/M^3 DAY)',/)
      WRITE(2,213)
  213 FORMAT(5X,'    XI          YJ     QWELOT(XI,YJ)',/,5X,36('-'))
      NONE=.TRUE.
      DO 214, J=1,NSLYP1
         DO 214, I=1,NSLXP1
            IF (QWELOT(I,J) .NE. 0.0) THEN
               NONE=.FALSE.
               WRITE(2,509) XNODE(I),YNODE(J),QWELOT(I,J)
            ENDIF
  214 CONTINUE
      IF (NONE) WRITE(2,510) 'EXTRACTION'
C
      WRITE(3,98765)
      WRITE(3,120)
  120 FORMAT(5X,' OUTPUT DATA FOR FLOW SYSTEM ',/)
C
      RETURN
  509 FORMAT(5X,1P,2(G10.3,2X),G12.5)
  510 FORMAT(5X,'NO NON-ZERO ',A,' WELLS.')
      END
C
C ****************************************
C
      SUBROUTINE FLOWRT
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CVELOC.SIB'
      include 'CWAT.SIW'
C
c     integer i, j
C
      WRITE(3,6)
    6 FORMAT(10X,' HYDRAULIC PRESSURE FIELD (M WATER)',/)
      CALL PRINT3(3,NSLXP1,NSLYP1,HNEW)
C
      IF(NFLAG(1).EQ.0) GO TO 501
C
      WRITE(3,7)
    7 FORMAT(/,10X,'DARCY VELOCITY FIELD',/,
     &   5X,'(MAGNITUDE IN M/D,ANGLE IN DEGREES)',/)
      CALL PRINT4(3,NSLXP1,NSLYP1,VLXX,VLYY)
C
```

```
c  A temporary way to drive surfer
c
c     do 10, i = 1, nslxp1
c     do 10, j = 1, nslyp1
c        write(61,2000) xnode(i), ynode(j), hnew(i,j)
c  10 continue
c2000 format(1P,3G15.6)
C
  501 RETURN
      END
```

---

```
*         File: SIPSIW.FOR              Last revision:  August 29, 1990
C                                       For water phase of LT3VSI.
      SUBROUTINE SIP(DONE)
C
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CSSIP.SIW'
C
      LOGICAL           DONE
      INTEGER           I, J, ICOUNT, IDMPDEL, NINTNO
      PARAMETER         (IDMPDEL=8)
      DOUBLE PRECISION  RESTOT(2:JX, 2:JY)
      DOUBLE PRECISION  DRMSEA, HRMSEA, RMSEA, RMSER
      EXTERNAL DRMSEA
C  The following *really* are REALS
      REAL  SECOND, TIMES, TIMEI, TIME1, START
      EXTERNAL SECOND
C
*****************************************************************
*   Stone's Strongly Implicit System (SIP) method for solving
*   linear systems arising from groundwater modeling:  July '89
*   Uses an incomplete LU factorization, LUFACT.
*                    David E. Cawlfield,  July '89
*****************************************************************
C
      DATA ICOUNT, TIMES, TIMEI/ 0, 0.0, 0.0/
C
C  0)  Entry conditions --
C          XDUM <-- HOLD in WATSIW
C          YDUM <--      in WATSIW
C  1)  Approximate the (M+N) matrix by (incomplete)
C      LU-factorization.
C
      NINTNO = NSLXM1*NSLYM1
```

```
      DONE = .FALSE.
      START = SECOND()
      CALL LUFACT
C
C  2)  Find initial residuals
C
      CALL RESID(RESTOT)
      RMSEA = DRMSEA(RESTOT)
      IF (NFLAG(3).EQ.0) THEN
         IF (ICOUNT.EQ.0) WRITE(*,3020) ALPH, RMSEA
      ELSE
         WRITE(*,2000) ALPH, RMSEA
      ENDIF
      HRMSEA = RMSEA
C
C  Main loop begins here (REPEAT until Converged)
C
      TIME1 = SECOND()
      TIMES = TIMES + TIME1 - START
      START = TIME1
C
C  ICOUNT is initialized in DATA so we can do re-starts
Cold  ICOUNT = 0
C
 1000 CONTINUE
         ICOUNT = ICOUNT+1
C
C  3)  Find DelY[k+1] = L-inv * r[k] by *forward* substitution
C
         CALL FORSUB(RESTOT)
C
C  4)  Form DelX[k+1] = U-inv * DelY[k] by *backward* substitution
C
         CALL BAKSUB
C
C  5)  Update the X's:
C
         RMSER = 0.D0
         DO 20, I = 2, NSLXXX
            DO 10, J = 2, NSLYYY
               XDUM(I,J) = XDUM(I,J) + DELX(I,J)
               IF (XDUM(I,J) .NE. 0.D0) THEN
                  RMSER = RMSER + (DELX(I,J) / XDUM(I,J))**2
               ENDIF
   10       CONTINUE
   20    CONTINUE
         RMSER = SQRT(RMSER/DBLE(NINTNO))
C
C  6)  Update the residuals, then test for convergance.
C
```

```
      CALL RESID(RESTOT)
      RMSEA = DRMSEA(RESTOT)
c  Do we really need two different tolerances, here?
      IF (RMSER .LE. TLRNWR .OR. RMSEA .LE. TLRNWA) THEN
         TIMEI = TIMEI + SECOND() - START
         WRITE(*,*)
         WRITE(*,*)'  Convergence Achieved -- Final Values: '
         WRITE(*,2032) ICOUNT, RMSEA, RMSER
         IF (NFLAG(3).NE.0) WRITE(*,2030) XDUM(2,2), XDUM(2,NSLYYY),
     &                      XDUM(NSLXXX,2), XDUM(NSLXXX,NSLYYY)
         WRITE(*,2200) TIMEI, TIMES, TIMEI+TIMES
         WRITE(9,*)'  Convergence Achieved -- Final Values: '
         WRITE(9,2032) ICOUNT, RMSEA, RMSER
         WRITE(9,2030) XDUM(2,2), XDUM(2,NSLYYY),
     &                      XDUM(NSLXXX,2), XDUM(NSLXXX,NSLYYY)
         WRITE(9,2200) TIMEI, TIMES, TIMEI+TIMES
         DONE = .TRUE.
C  *** R E T U R N S *** iff converged.
         RETURN
      ENDIF
C
C  Print current values if not converged
C
      IF (NFLAG(3).EQ.0) THEN
         WRITE(*,3000) ICOUNT, ALPH, RMSEA, RMSER
      ELSE
         WRITE(*,*)
         WRITE(*,2032) ICOUNT, RMSEA, RMSER
         WRITE(*,2030) XDUM(2,2), XDUM(2,NSLYYY),
     &                      XDUM(NSLXXX,2), XDUM(NSLXXX,NSLYYY)
         WRITE(*,2030) DELX(2,2), DELX(2,NSLYYY),
     &                      DELX(NSLXXX,2), DELX(NSLXXX,NSLYYY)
      ENDIF
C  Hold last used rmse
      HRMSEA = RMSEA
C
C  DELMAX finds and prints the max and min DELX's.
C
      IF (NFLAG(13) .EQ. 1 .AND. MOD(ICOUNT,IDMPDEL) .EQ. 0)
     &    CALL DELMAX(ICOUNT, RMSEA, RMSER)
C
C  Re-start every NMOD cycles, iff NMOD > 0
C
      IF (NMOD .GT. 0 .AND. MOD(ICOUNT,NMOD) .EQ. 0) THEN
         TIMEI = TIMEI + SECOND() - START
         RETURN
      ENDIF
      IF ((RMSEA .GT. 1.0D4 .OR. RMSER .GT. 1.0D0) .AND.
     &    ICOUNT .GT. 1) THEN
         WRITE(*,*) ' Impending Overflow (Divergence?) '
```

```
         STOP 2000
      ENDIF
      IF (ICOUNT .LT. NLSOR) GO TO 1000
C
C  Abnormal Exit
C
      WRITE(*,*) ' Failure to Converge in ', NLSOR, ' loops.'
      WRITE(*,*) ' Iteration Limit Exceeded.'
      STOP 1
C
 2000 FORMAT(/,' ALPH = ',F7.4,' Initial Residuals = ',1P,G13.6,/)
 2030 FORMAT(1X,1P,10(G15.8,1X))
 2032 FORMAT(' ICOUNT=',I5,1P,', RMSEA= ',G13.6,:,', RMSER= ',G13.6)
C2080 format(t3,'??? Oscillating ??? ', 1P, G13.6, ' < ', G13.6)
 2200 FORMAT(T4,'Cycle Time = ',F7.2,' (Startup Time = ',F7.2,')',/,
     &       T4,'Total Time = ',F7.2)
 3000 FORMAT(1X,I6,1P,3(2X,G12.5))
 3020 FORMAT(1X,'Initial',1X,1P,G12.5,2X,G12.5)
      END
C
C ************************************
C
      SUBROUTINE RESID(RESTOT)
C
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CSSIP.SIW'
C
      INTEGER           I, J, II, JJ
      DOUBLE PRECISION  RESTOT(2:JX, 2:JY), SAXY
C
C  The following is a STATEMENT FUNCTION
C
      SAXY(II,JJ) = ADT2X(II,JJ) + ADT2Y(II,JJ)
C
************************
*  Compute the residuals.
************************
C
C  TOP-LEFT HAND NODE.
C
      RESTOT(2,2) = YDUM(2,2) - SAXY(2,2)*XDUM(2,2) -
     &    ADT3(2,2)*XDUM(2,3) - AUT2(2,2)*XDUM(3,2)
C
C  LEFT HAND CENTRAL NODES.
C
      DO 169, J = 3, NSLYM1
      RESTOT(2,J) = YDUM(2,J) - SAXY(2,J)*XDUM(2,J) -
     &  ADT3(2,J)*XDUM(2,J+1) - AUT2(2,J)*XDUM(3,J) -
     &  ADT1(2,J)*XDUM(2,J-1)
```

```
  169 CONTINUE
C
C  BOTTOM-LEFT HAND NODE.
C
      RESTOT(2,NSLYYY) = YDUM(2,NSLYYY) -
     &  SAXY(2,NSLYYY)*XDUM(2,NSLYYY)   -
     &  ADT1(2,NSLYYY)*XDUM(2,NSLYM1)-AUT2(2,NSLYYY)*XDUM(3,NSLYYY)
C
C  CENTRAL BLOCK OF NODES.
C
      DO 170, I = 3, NSLXM1
C
C  TOP NODES.
C
      RESTOT(I,2) = YDUM(I,2) - SAXY(I,2)*XDUM(I,2) -
     &    ADT3(I,2)*XDUM(I,3) - AUT2(I,2)*XDUM(I+1,2) -
     &    ALT2(I,2)*XDUM(I-1,2)
C
C  INTERIOR BLOCK OF NODES
C
      DO 171, J = 3, NSLYM1
      RESTOT(I,J) = YDUM(I,J) - SAXY(I,J)*XDUM(I,J)   -
     &  ADT3(I,J)*XDUM(I,J+1) - AUT2(I,J)*XDUM(I+1,J) -
     &  ADT1(I,J)*XDUM(I,J-1) - ALT2(I,J)*XDUM(I-1,J)
  171 CONTINUE
C
C  BOTTOM NODES
C
      RESTOT(I,NSLYYY) = YDUM(I,NSLYYY) -
     &  SAXY(I,NSLYYY)*XDUM(I,NSLYYY)   -
     &  ADT1(I,NSLYYY)*XDUM(I,NSLYM1)   -
     &  AUT2(I,NSLYYY)*XDUM(I+1,NSLYYY) -
     &  ALT2(I,NSLYYY)*XDUM(I-1,NSLYYY)
  170 CONTINUE
C
C
C  TOP-RIGHT HAND NODE.
C
      RESTOT(NSLXXX,2) = YDUM(NSLXXX,2) -
     &  SAXY(NSLXXX,2)*XDUM(NSLXXX,2)   -
     &  ADT3(NSLXXX,2)*XDUM(NSLXXX,3)   -
     &  ALT2(NSLXXX,2)*XDUM(NSLXM1,2)
C
C  RIGHT HAND CENTRAL NODES.
C
      DO 177, J = 3, NSLYM1
      RESTOT(NSLXXX,J) = YDUM(NSLXXX,J) -
     &  SAXY(NSLXXX,J)*XDUM(NSLXXX,J)   -
     &  ADT3(NSLXXX,J)*XDUM(NSLXXX,J+1) -
     &  ADT1(NSLXXX,J)*XDUM(NSLXXX,J-1) -
```

```
     &   ALT2(NSLXXX,J)*XDUM(NSLXM1,J)
  177 CONTINUE
C
C  BOTTOM RIGHT HAND NODE.
C
      RESTOT(NSLXXX,NSLYYY) = YDUM(NSLXXX,NSLYYY) -
     &   SAXY(NSLXXX,NSLYYY)*XDUM(NSLXXX,NSLYYY)   -
     &   ADT1(NSLXXX,NSLYYY)*XDUM(NSLXXX,NSLYM1)   -
     &   ALT2(NSLXXX,NSLYYY)*XDUM(NSLXM1,NSLYYY)
C
      RETURN
      END
C
C ************************************
C
      DOUBLE PRECISION FUNCTION DRMSEA(RESTOT)
C
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
C
      DOUBLE PRECISION    RESTOT(2:JX, 2:JY)
      INTEGER             I, J, NINTNO
C
*****************************************************************
*  Simple function to compute the current Residual Sum of Squares.
*****************************************************************
C
      NINTNO = NSLXM1*NSLYM1
      DRMSEA = 0.0D0
      DO 20, I = 2, NSLXXX
         DO 10, J = 2, NSLYYY
            DRMSEA = DRMSEA + RESTOT(I,J)*RESTOT(I,J)
   10    CONTINUE
   20 CONTINUE
      DRMSEA  = DRMSEA / DBLE(NINTNO)
      DRMSEA  = DSQRT(DRMSEA)
C
      RETURN
      END
C
C ************************************
C
      SUBROUTINE FORSUB(R)
C
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CSSIP.SIW'
C
      INTEGER             I, J
      DOUBLE PRECISION    R(2:JX, 2:JY)
```

```
C
*********************************************************
*  Solve the following system by forward substitution --
*
*                  (    )-1
*     DelY[k+1] = ( L  )    *  r[k]
*                  (    )
*********************************************************
C
C  Compute left hand block at i == 2
C
      DELY(2,2) = R(2,2) / D(2,2)
      DO 10, J = 3, NSLYYY
         DELY(2,J) = (R(2,J) - C(2,J)*DELY(2,J-1)) / D(2,J)
   10 CONTINUE
C
C  Now, do remaining interior blocks
C
      DO 20, I = 3, NSLXXX
         DELY(I,2) = (R(I,2) - B(I,2)*DELY(I-1,2)) / D(I,2)
         DO 30, J = 3, NSLYYY
            DELY(I,J) = (R(I,J) - B(I,J)*DELY(I-1,J) -
     &                   C(I,J)*DELY(I,J-1)) / D(I,J)
   30    CONTINUE
   20 CONTINUE
C
      RETURN
      END
C
C ************************************
C
      SUBROUTINE BAKSUB
C
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CSSIP.SIW'
C
      INTEGER             I, J
C
*********************************************************
*  Solve the following system by backward substitution --
*
*                  (   )-1
*     DelX[k+1] = ( U )    *  DelY[k]
*                  (   )
*********************************************************
C
C  Lower block first
C
      DELX(NSLXXX,NSLYYY) = DELY(NSLXXX,NSLYYY)
```

```
      DO 60, J = NSLYYY-1, 2, -1
         DELX(NSLXXX,J) = DELY(NSLXXX,J) -
     &                    E(NSLXXX,J)*DELX(NSLXXX,J+1)
   60 CONTINUE
C
C  Now, do interior blocks
C
      DO 80, I = NSLXXX-1, 2, -1
         DELX(I,NSLYYY) = DELY(I,NSLYYY) -
     &                    F(I,NSLYYY)*DELX(I+1,NSLYYY)
         DO 70, J = NSLYYY-1, 2, -1
            DELX(I,J) = DELY(I,J) -
     &                  E(I,J)*DELX(I,J+1)  -
     &                  F(I,J)*DELX(I+1,J)
   70    CONTINUE
   80 CONTINUE
C
      RETURN
      END
C
C ************************************
C
      SUBROUTINE DELMAX(ICOUNT, RMSEA, RMSER)
C
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CSSIP.SIW'
C
      INTEGER           I, J, IMAX, JMAX, IMIN, JMIN, ICOUNT
      DOUBLE PRECISION  DELTMP, RMAX, RMIN
      DOUBLE PRECISION  RMSEA, RMSER
C
*****************************************************************
*  Search for max and min DELX and write to file.  This is an aid for
*  adjusting ALPH.
*****************************************************************
C
      RMAX = DELX(2,2)
      RMIN = DELX(2,2)
      DO 10, I = 2, NSLXXX
         DO 20, J = 2, NSLYYY
            DELTMP = DELX(I,J)
            IF (DELTMP .GT. RMAX) THEN
               RMAX = DELTMP
               IMAX = I
               JMAX = J
            ENDIF
            IF (DELTMP .LT. RMIN) THEN
               RMIN = DELTMP
               IMIN = I
```

```
               JMIN = J
             ENDIF
   20 CONTINUE
   10 CONTINUE
      WRITE(9,2000) ICOUNT, RMSEA, RMSER,
     &   IMAX, JMAX, RMAX, IMIN, JMIN, RMIN
      RETURN
C
 2000 FORMAT(I4, ' RMSEA = ', G13.6, ', RMSER = ', G13.6,/,
     &   '  MAX DELX(', I2.2, ',', I2.2, ') = ', G13.6,
     &   ', MIN DELX(', I2.2, ',', I2.2, ') = ', G13.6 )
      END
```

---

```
*         File: SUBSIB.FOR              Last revision: August 29, 1990
C                                       For all three phases of LT3VSI.
C
C  This module contains some subroutines that are used in both phases
C  of LT3VSI.  Coded and revised by Gilbert A. Bachelor,
C  Aug 1989 .. Jan 1990
C
      SUBROUTINE DATERR(STRING,FILBAS,EXT)
C
C  DISPLAY A MESSAGE CONCERNING AN ERROR FOLLOWING LINE STRING IN A
C  FILE WITH BASE NAME FILBAS AND EXTENSION EXT.  THIS IS USED BY
C  SUBROUTINES FLOREAD, CHMREAD, AND TESTIJ.
C
      CHARACTER EXT*3, FILBAS*8, STRING*(*)
C
 1000 FORMAT(' A data error was found in the file named ',A,'.',A,/,
     &       ' The error occurred after the line:',/,1X,A)
      WRITE(*,1000) FILBAS,EXT,STRING
      RETURN
      END
C
C  ************************************************
C
      SUBROUTINE RDBASE(FILBAS,EXT)
C
C  READ, FROM THE KEYBOARD, THE BASE NAME FOR A FILE WHOSE EXTENSION
C  WILL BE EXT.  RETURN THE BASE NAME, RIGHT-JUSTIFIED, IN FILBAS.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      CHARACTER CHRTMP*8, EXT*3, FILBAS*8
C
 1001 FORMAT(' Do not enter the extension, which is ".',A,'": ')
```

```
 1002 FORMAT(A)
C
      WRITE(*,1001) EXT
      READ(*,1002) FILBAS
      IF (FILBAS .EQ. ' ') THEN
         WRITE(*,*) ' ** File name must not be blank! **'
         STOP 763
      ENDIF
C
C  RIGHT-JUSTIFY BASE FILE NAME, SO THAT WHEN FILE NAMES ARE PRINTED
C  OUT, THERE WON'T BE SPACES BETWEEN BASE AND EXTENSION.  USE A
C  TEMPORARY VARIABLE BECAUSE SOME VERSIONS OF FORTRAN DO NOT
C  IMPLEMENT CHARACTER ASSIGNMENTS PROPERLY WHEN THE SAME VARIABLE
C  APPEARS ON BOTH SIDES OF ASSIGNMENT.
C
   20 IF (FILBAS(8:8) .EQ. ' ') THEN
         CHRTMP=' '//FILBAS(1:7)
         FILBAS=CHRTMP
         GOTO 20
      ENDIF
C
      RETURN
      END
C
C  ************************************************
C
      SUBROUTINE TESTIJ(STRING,FILBAS,EXT,XI,YJ,QTEMP,I,J)
C
C  THIS SUBROUTINE TESTS XI AND YJ TO MAKE SURE THEY ARE THE COORDINATES
C  OF AN INTERIOR NODE.  IF THEY ARE, IT DIVIDES QTEMP BY THE
C  APPROPRIATE VOLUME, SETS I AND J TO THE INDICES OF THE NODE, AND
C  RETURNS.  IF NOT, IT PRINTS AN ERROR MESSAGE AND STOPS.
C  STRING AND THE FILE NAME FILBAS WITH EXTENSION EXT ARE INCLUDED
C  IN THE MESSAGE.
C  THE OPERATION ON QTEMP CONVERTS IT FROM UNITS OF (KG/DAY) TO UNITS
C  OF (KG/M^3 DAY); THIS IS NEEDED IN SUBROUTINES FLOREAD AND CHMREAD.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      CHARACTER EXT*3,FILBAS*8,STRING*(*)
      DOUBLE PRECISION QTEMP, XI, YJ
      INTEGER I, J, INDX
      EXTERNAL INDX
C
1000  FORMAT(1P,' Data error: ',A,'=',G10.3,' is not a',
     &          ' valid coordinate.')
1001  FORMAT(1P,' Data error: XI=',G10.3,', YJ=',G10.3,
     &          ' is not an interior node!')
C
```

```
      I=INDX(NSLXP1,XNODE,XI)
      J=INDX(NSLYP1,YNODE,YJ)
      IF (I.LT.1 .OR. J.LT.1) THEN
         CALL DATERR(STRING,FILBAS,EXT)
         IF (I.LT.1) WRITE(*,1000) 'XI',XI
         IF (J.LT.1) WRITE(*,1000) 'YJ',YJ
         STOP 1
      ENDIF
      IF (2.LE.I .AND. I.LE.NSLXXX .AND.
     &    2.LE.J .AND. J.LE.NSLYYY) THEN
         QTEMP=QTEMP/(XLW*(DX(I-1)+DX(I))*(DY(J)+DY(J-1))/4.0D0)
      ELSE
         CALL DATERR(STRING,FILBAS,EXT)
         WRITE(*,1001) XI,YJ
         STOP 1
      ENDIF
      RETURN
      END
C
C  ************************************************
C
      INTEGER FUNCTION INDX(N,ARY,VAL)
C
C  SEARCHES ARRAY ARY FROM ARY(1) TO ARY(N) FOR VAL.
C  IF VAL IS FOUND, RETURNS INDEX OF VAL IN ARY.
C  RETURNS -1 IF VAL NOT FOUND.
C
      INTEGER I, N
      DOUBLE PRECISION ARY(N), VAL
C
      DO 20, I=1,N
         IF (VAL .EQ. ARY(I)) THEN
            INDX=I
            RETURN
         ENDIF
20    CONTINUE
      INDX=-1
      RETURN
      END
```

---

```
*         File: TCMSIL.FOR           Last revision:  August 29, 1990
C                                    For chem loop phase of LT3VSI.
      SUBROUTINE TCMIA
C
C  CALCULATE THE TOTAL CHEMICAL MASS IN AQUIFER, CUMULATIVE FIRST ORDER
```

```
C  LOSSES, AND THE CUMULATIVE ZERO ORDER BURIED SOURCES CONTRIBUTION.
C
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CVELOC.SIB'
      include 'CRUNC.SIK'
      include 'CCHEM.SIK'
      include 'CCHEM.SIL'
C
      INTEGER I, J, K
      DOUBLE PRECISION CFLXIN(IX),CFLXIO(IX)
      DOUBLE PRECISION CFLXON(IX),CFLXOO(IX)
      DOUBLE PRECISION ERTEMP
      DOUBLE PRECISION F1, F2, F3, F4
      DOUBLE PRECISION F5N, F6N, F7N, F8N
      DOUBLE PRECISION F5O, F6O, F7O, F8O
      DOUBLE PRECISION FS1, FS2, FS3, FS4, FS5, FS6, FS7, FS8
      DOUBLE PRECISION SUMM1, SUMM2, SUMM3, SUMM4, SUMM5
      DOUBLE PRECISION SUMM6, SUMM7
C
      DO 1002, K=1,NC
        SUMM1=0.0D0
        SUMM2=0.0D0
        SUMM3=0.0D0
C
        DO 1000, I=1,NSLXXX
          DO 1000, J=1,NSLYYY
            F1=EPS(I,J)*RETARD(I,J,K)*CNEW(I,J,K)
            F2=EPS(I+1,J)*RETARD(I+1,J,K)*CNEW(I+1,J,K)
            F3=EPS(I,J+1)*RETARD(I,J+1,K)*CNEW(I,J+1,K)
            F4=EPS(I+1,J+1)*RETARD(I+1,J+1,K)*CNEW(I+1,J+1,K)
            F5N=EPS(I,J)*LAMDA(I,J,K)*CNEW(I,J,K)
            F6N=EPS(I+1,J)*LAMDA(I+1,J,K)*CNEW(I+1,J,K)
            F7N=EPS(I,J+1)*LAMDA(I,J+1,K)*CNEW(I,J+1,K)
            F8N=EPS(I+1,J+1)*LAMDA(I+1,J+1,K)*CNEW(I+1,J+1,K)
            F5O=EPS(I,J)*LAMDA(I,J,K)*COLD(I,J,K)
            F6O=EPS(I+1,J)*LAMDA(I+1,J,K)*COLD(I+1,J,K)
            F7O=EPS(I,J+1)*LAMDA(I,J+1,K)*COLD(I,J+1,K)
            F8O=EPS(I+1,J+1)*LAMDA(I+1,J+1,K)*COLD(I+1,J+1,K)
            FS1=QCHM1S(I,J,K)
            FS2=QCHM1S(I+1,J,K)
            FS3=QCHM1S(I,J+1,K)
            FS4=QCHM1S(I+1,J+1,K)
            FS5=QWELIN(I,J)*CSWELN(I,J,K)
            FS6=QWELIN(I+1,J)*CSWELN(I+1,J,K)
            FS7=QWELIN(I,J+1)*CSWELN(I,J+1,K)
            FS8=QWELIN(I+1,J+1)*CSWELN(I+1,J+1,K)
            SUMM1=SUMM1+(DX(I)*DY(J))*(F1+F2+F3+F4)/4.0D0
            SUMM2=SUMM2+DX(I)*DY(J)*(F5N+F6N+F7N+F8N)/4.0D0+
     &          DX(I)*DY(J)*(F5O+F6O+F7O+F8O)/4.0D0
```

```
            SUMM3=SUMM3+DX(I)*DY(J)*(FS1+FS2+FS3+FS4+FS5+FS6+FS7+FS8)/
     &          4.0D0
 1000    CONTINUE
C
         XMASS(K)=XLW*SUMM1
         XMFONW(K)=XMFONW(K)+XLW*DT0*SUMM2/2.0D0
         XMSOUR(K)=XMSOUR(K)+XLW*DT0*SUMM3
 1002 CONTINUE
C
       DO 1020, K=1,NC
         SUMM4=0.0D0
         SUMM5=0.0D0
         SUMM6=0.0D0
         SUMM7=0.0D0
C
         DO 1010, I=1,NSLXP1
           ERTEMP=EPS(I,1)*RETARD(I,1,K)
           CFLXIO(I)=-ERTEMP*DCHLY(I,1,K)*(COLD(I,2,K)-
     &        COLD(I,1,K))/DY(1)+
     &        VLYY(I,1)*ERTEMP*COLD(I,1,K)
           CFLXIN(I)=-ERTEMP*DCHLY(I,1,K)*(CNEW(I,2,K)-
     &        CNEW(I,1,K))/DY(1)+
     &        VLYY(I,1)*ERTEMP*CNEW(I,1,K)
           ERTEMP=EPS(I,NSLYP1)*RETARD(I,NSLYP1,K)
           CFLXOO(I)=-ERTEMP*DCHLY(I,NSLYP1,K)*(COLD(I,NSLYP1,K)-
     &        COLD(I,NSLYYY,K))/DY(NSLYYY)+
     &        VLYY(I,NSLYP1)*ERTEMP*COLD(I,NSLYYY,K)
           CFLXON(I)=-ERTEMP*DCHLY(I,NSLYP1,K)*(CNEW(I,NSLYP1,K)-
     &        CNEW(I,NSLYYY,K))/DY(NSLYYY)+
     &        VLYY(I,NSLYP1)*ERTEMP*CNEW(I,NSLYYY,K)
 1010    CONTINUE
C
         DO 1015, I=1,NSLXXX
           SUMM4=SUMM4+DX(I)*(CFLXIO(I)+CFLXIO(I+1))/2.0D0
           SUMM5=SUMM5+DX(I)*(CFLXIN(I)+CFLXIN(I+1))/2.0D0
           SUMM6=SUMM6+DX(I)*(CFLXOO(I)+CFLXOO(I+1))/2.0D0
           SUMM7=SUMM7+DX(I)*(CFLXON(I)+CFLXON(I+1))/2.0D0
 1015    CONTINUE
C
         XMASIN(K)=XMASIN(K)+DT0*(SUMM4+SUMM5)*XLW/2.0D0
         XMASOT(K)=XMASOT(K)+DT0*(SUMM6+SUMM7)*XLW/2.0D0
 1020 CONTINUE
C
       RETURN
       END
```

```
*          File: WATSIW.FOR              Last revision:  August 29, 1990
C                                        For water phase of LT3VSI.
      SUBROUTINE FLUID
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      include 'CSIZE.SIB'
      include 'CPROP.SIB'
      include 'CSSIP.SIW'
      include 'CWAT.SIW'
C
      INTEGER I, J
      DOUBLE PRECISION CONSTX, CONSTY
C
C     DEFINE MATRIX ELEMENTS FOR WATER PRESSURE FIELD MATRIX.
C
      DO 30, I=2,NSLXXX
      CONSTX=2.0D0/(DX(I-1)+DX(I))
      DO 31, J=2,NSLYYY
      CONSTY=2.0D0/(DY(J-1)+DY(J))
C
      ADT1(I,J)= -(KSATYY(I,J-1)+KSATYY(I,J))*CONSTY/(2.0D0*DY(J-1))
      ADT2Y(I,J)=CONSTY*((KSATYY(I,J-1)+KSATYY(I,J))/(2.0D0*DY(J-1))+
     &                   (KSATYY(I,J)+KSATYY(I,J+1))/(2.0D0*DY(J)))
      ADT2X(I,J)=CONSTX*((KSATXX(I-1,J)+KSATXX(I,J))/(2.0D0*DX(I-1))+
     &                   (KSATXX(I,J)+KSATXX(I+1,J))/(2.0D0*DX(I)))
      ADT3(I,J)= -(KSATYY(I,J)+KSATYY(I,J+1))*CONSTY/(2.0D0*DY(J))
      ALT2(I,J)= -(KSATXX(I-1,J)+KSATXX(I,J))*CONSTX/(2.0D0*DX(I-1))
      AUT2(I,J)= -(KSATXX(I,J)+KSATXX(I+1,J))*CONSTX/(2.0D0*DX(I))
C
   31 CONTINUE
   30 CONTINUE
      IF(NFLAG(10).EQ.0) GO TO 950
C
C++++++++++++++++++++++++++++++++++++++++++++
      WRITE(9,*)
      WRITE(9,*)'Raw matrix elements'
      WRITE(9,*)
      DO 800, I=2,NSLXXX
      DO 801, J=2,NSLYYY
      WRITE(9,802) I,J,ALT2(I,J),ADT1(I,J),ADT2X(I,J),ADT2Y(I,J),
     &      ADT3(I,J),AUT2(I,J)
  801 CONTINUE
  800 CONTINUE
  802 FORMAT(1X,2I3,6E12.6)
      WRITE(9,*)
C++++++++++++++++++++++++++++++++++++++++++++
C
  950 CONTINUE
C
      DO 20, J=2,NSLYYY
```

```
      ADT2X(2,J)=ADT2X(2,J)+ALT2(2,J)*CONST1
      AUT2(2,J)=AUT2(2,J)-ALT2(2,J)*CONST2
      ADT2X(NSLXXX,J)=ADT2X(NSLXXX,J)+AUT2(NSLXXX,J)*CONST3
      ALT2(NSLXXX,J)=ALT2(NSLXXX,J)-AUT2(NSLXXX,J)*CONST4
   20 CONTINUE
C
      IF(NFLAG(11).EQ.0) GO TO 951
C
C++++++++++++++++++++++++++++++++++++++++++
      WRITE(9,*)
      WRITE(9,*)'Adjusted boundary matrix elements(Y-DIRECTION)'
      WRITE(9,*)
      DO 803, J=2,NSLYYY
      WRITE(9,804) J,ADT2X(2,J),AUT2(2,J),ADT2X(NSLXXX,J),
     &      ALT2(NSLXXX,J)
  803 CONTINUE
  804 FORMAT(1X,I3,4E12.6)
      WRITE(9,*)
C++++++++++++++++++++++++++++++++++++++++++
C
C     CALCULATE THE "KNOWN" VECTOR FOR THE WATER PRESSURE DISTRIBUTION.
C
C     THE KNOWN VECTOR IS FILLED WHERE EACH POSITION HAS CONTRIBUTIONS
C     FROM
C  1. ANY SOURCES
C  2. BOUNDARY TERMS OF MOISTURE FIELD
C
  951 DO 80, I=2,NSLXXX
C
      YDUM(I,2)=(QWELIN(I,2)-QWELOT(I,2))/RHOWAT
      YDUM(I,2)=YDUM(I,2)-ADT1(I,2)*HIN
C
      YDUM(I,NSLYYY)=(QWELIN(I,NSLYYY)-QWELOT(I,NSLYYY))/RHOWAT
      YDUM(I,NSLYYY)=YDUM(I,NSLYYY)-ADT3(I,NSLYYY)*HOUT
C
   80 CONTINUE
C
      DO 90, J=3,NSLYM1
C
      YDUM(2,J)=(QWELIN(2,J)-QWELOT(2,J))/RHOWAT
C
      YDUM(NSLXXX,J)=(QWELIN(NSLXXX,J)
     &     -QWELOT(NSLXXX,J))/RHOWAT
C
   90 CONTINUE
C
      DO 100, I=3,NSLXM1
      DO 101, J=3,NSLYM1
      YDUM(I,J)=(QWELIN(I,J)-QWELOT(I,J))/RHOWAT
  101 CONTINUE
```

```
  100 CONTINUE
C
      IF(NFLAG(12).EQ.0) GO TO 952
C
C+++++++++++++++++++++++++++++++++++++++++++
      WRITE(9,*)
      WRITE(9,*)' "Known vector" '
      WRITE(9,*)
      DO 805, I=2,NSLXXX
      DO 806, J=2,NSLYYY
      WRITE(9,807) I,J,YDUM(I,J)
  806 CONTINUE
  805 CONTINUE
  807 FORMAT(1X,2I3,E12.6)
      WRITE(9,*)
C+++++++++++++++++++++++++++++++++++++++++++
C
  952 CONTINUE
C
C  INITIAL GUESS FOR THE STEADY STATE FLUID HYDRAULIC FIELD.
C
      DO 301, I=2,NSLXXX
      DO 302, J=2,NSLYYY
      XDUM(I,J)=HOLD(I,J)
  302 CONTINUE
  301 CONTINUE
C
      RETURN
      END
```

## Appendix B. Listing of LTPREP program

The source code for LTPREP is on the installation diskette, in the file LTPREP.ARC. The two files are listed below: the "include" file LTPREP.INC, and the Fortran source file LTPREP.FOR. See section IV.1 for information on using LTPREP.

### I N C L U D E  F I L E

```
* Include file: LTPREP.INC (Ver 2.2)  Last revision:   November 9, 1989
* Limits on array sizes.
      integer limx,limy
      parameter(limx=50,limy=50)
* Common variables and arrays.
      integer nx, ny, state, svstat
      real value, xnode, ynode
      common /sense/ nx, ny, xnode(limx), ynode(limy),
     &   state(0:limx+1,0:limy+1), value(0:limx+1,0:limy+1),
     &   svstat(0:limx+1,0:limy+1)
*
      character svline*80
      common /tater/ svline
```

### F O R T R A N  C O D E

```
* LTPREP.FOR (Ver. 2.2)                Last revision: November 9, 1989
      program ltprep
*  Helps prepare data for "ltaq" programs that allow data
*  which varies from place to place within the aquifer.
*
$include:'LTPREP.INC'
      character*12 geofil, datfil, outfil
*  Unit numbers for files:
*    datfil: unit 1
*    outfil: unit 2
*    geofil: unit 3
*    picture file: unit 4
      character line*80
      integer length
      external length
*  Read names of files and open them.
1000  format(a)
      print 1010
1010  format(1x,'Enter name of GEOMETRY file: ',$)
      read(*,1000) geofil
      open(unit=3,file=geofil,status='OLD')
```

```
      print 1020
1020  format(1x,'Enter name of INPUT DATA file: ',$)
      read(*,1000) datfil
      open(unit=1,file=datfil,status='OLD')
      print 1030
1030  format(1x,'Enter name of OUTPUT file: ',$)
      read(*,1000) outfil
      open(unit=2,file=outfil)
*  Open picture file.
      open(unit=4,file='LTPICT.XXX')
*  Read geometry file & store data.
      call rdgeom(3)
*  Read data file, checking for commands.  Process commands
*  and write processed output; copy other lines to output file.
10    continue
      read(1,1000,end=99) line
*  Test for #G command (copy Geometry file)
      if (line(1:2) .eq. '#G') then
         call cpygeo(3,2)
*  Test for #V command (convert values from simple form to form needed
*  by "ltaq"-like programs).
      elseif (line(1:2) .eq. '#V') then
*  Set up the state array.
         call setup
*  Read values for interiors of regions.
         call rdvals(1)
*  Compute values on boundaries.
         call cmpbnd
*  Write the results on output file.
         call output(2)
      else
*  Copy non-command line to output.
         write(2,1000) line(1:length(line))
      endif
*  Loop back.
      goto 10
*  Write picture file and quit.
99    call wrtpic(4,geofil)
      print *, 'Picture has been written on file LTPICT.XXX'
      stop
      end
**********************
      subroutine rdgeom(unit)
$include:'LTPREP.INC'
      character string*80
      integer i, j, unit
*  Read the geometry of the aquifer from unit.
*  The data in the file must have the following form:
*     String
*     nx, ny
```

```
*     String
*     xnode(1), xnode(2), ..., xnode(nx+2)
*     String
*     ynode(1), ynode(2), ..., ynode(ny+2)
*     String
*     Boundary data - see subroutine rdbnds.
*  The "String"s can be any data, such that the fourth "String"
*  is NOT the same as any previous line in the file.
*  nx and ny are integers; the xnode's and ynode's are real
*  numbers.  See comments below.
*
*  Read the dimensions (no. of interior nodes in x & y directions).
      read (unit,2000) string
      read (unit,*) nx, ny
*  Make sure dimensions are within maximum limits.
      if (nx+2.gt.limx .or. ny+2.gt.limy) then
         print *, 'Nx = ',nx,', Ny = ',ny
         print *, 'Limits are: ',limx-2,', ',limy-2
         stop 1
      endif
*  Change dimensions to number of nodes including outside boundaries.
      nx=nx+2
      ny=ny+2
*  Read the node positions for the x-direction.
      read (unit,2000) string
      read (unit,*) (xnode(i),i=1,nx)
*  Read the node positions for the y-direction.
      read (unit,2000) string
      read (unit,*) (ynode(j),j=1,ny)
*  Initialize svstat array.
      call init
*  Read internal boundary data.
      read (unit,2000) svline
      call rdbnds(unit)
2000  format(a)
      return
      end
**********************
      subroutine init
$include:'LTPREP.INC'
      integer i, j
*  Initialize svstat array to undefined (0), except for a boundary (-1)
*  around the outside.
*************
*  Definition of values in state array and in svstat array:
*    state(i,j) = -1  boundary node whose value hasn't been computed.
*    state(i,j) =  0  interior node whose value hasn't been computed.
*    state(i,j) =  1  interior node whose value has been computed.
*    state(i,j) =  2  boundary node whose value has been computed.
*************
```

```
      do 20, j=1,ny
         do 20, i=1,nx
            svstat(i,j)=0
20    continue
      do 24, i=0,nx+1
         svstat(i,0)=-1
         svstat(i,ny+1)=-1
24    continue
      do 26, j=0,ny+1
         svstat(0,j)=-1
         svstat(nx+1,j)=-1
26    continue
      return
      end
**********************
      subroutine rdbnds(unit)
$include:'LTPREP.INC'
      complex z(11)
      integer i, j, k, nlines, nx1, nx2, ny1, ny2, unit
      real x1, x2, y1, y2
      integer indx
      external indx
*************
*  Read data specifying internal boundaries, between regions of
*  different values.  Boundaries must be horizontal or vertical,
*  and are specified by giving the coordinates of their endpoints.
*  The data for each boundary line has the form:
*        (x1,y1)  (x2,y2)
*  where x1, y1, x2, and y2 are real numbers, and either x1 = x2,
*  or y1 = y2.  Input data has the form:
*        n  (x1,y1)  (x2,y2)  ...  (xn+1,yn+1)
*  where n is an integer (0 <= n <= 10), followed by n+1 pairs of
*  coordinates.  This represents n connected boundary lines:
*        (x1,y1)  (x2,y2)
*        (x2,y2)  (x3,y3)
*                 ...
*        (xn,yn)  (xn+1,yn+1)
*  The special case when n = 0:
*        0  (x1,y1)
*  represents a single boundary point.
*  The data is ended by a line containing a single -1:
*        -1
*************
*  Read a list of endpoints, using list-directed complex number
*  input for the points.
10    read (unit,*) nlines, (z(i),i=1,nlines+1)
      if (nlines .lt. 0) then
         return
      elseif (nlines .eq. 0) then
         nlines=1
```

```
         z(2)=z(1)
      endif
      do 40, k=1,nlines
         x1=real(z(k))
         y1=aimag(z(k))
         x2=real(z(k+1))
         y2=aimag(z(k+1))
         if (x1.ne.x2 .and. y1.ne.y2) then
            print *, 'Bad boundary data; line is neither horizontal ',
     &         'nor vertical: ',z(k),z(k+1)
            stop 1
         endif
         nx1=indx(nx,xnode,x1)
         ny1=indx(ny,ynode,y1)
         if (x1.eq.x2) then
            ny2=indx(ny,ynode,y2)
            do 20, j=ny1,ny2,sign(1,ny2-ny1)
               svstat(nx1,j)=-1
20          continue
         elseif (y1.eq.y2) then
            nx2=indx(nx,xnode,x2)
            do 30, i=nx1,nx2,sign(1,nx2-nx1)
               svstat(i,ny1)=-1
30          continue
         else
            print *, 'Impossible error in RDBNDS!'
            stop 1
         endif
40    continue
      goto 10
      end
**********************
      subroutine cpygeo(geo,out)
$include:'LTPREP.INC'
      character line*80
      integer geo, out, length
      external length
*  Copy the geometry file to the output file, up to the
*  boundary data.
      rewind(geo)
10    continue
         read (geo,1000) line
         if (line .eq. svline) return
         write (out,1000) line(1:length(line))
      goto 10
1000  format(a)
      end
**********************
      subroutine setup
$include:'LTPREP.INC'
```

```
      integer i, j
*  Set up the state array, by copying the values from
*  the svstat array.
      do 20, i=0,nx+1
         do 20, j=0,ny+1
            state(i,j)=svstat(i,j)
20    continue
      return
      end
**********************
      subroutine rdvals(unit)
$include:'LTPREP.INC'
      complex z
      integer i, ii, j, jj, vx, vy, unit
      logical change
      real val
      integer indx
      external indx
*  Reads values for interiors of regions.  There should be one
*  value in each region.  The program propagates the value throughout
*  the region.  Each line of data must have the form:
*     (x,y)  val
*  where x and y are real numbers specifying the location of a
*  point inside one of the regions, and val is the value to be
*  assigned to all points within that region.  The data is ended by a
*  line of the form:
*     (0,0)  0.0
*
*  Read lines of data.
10    read (unit,*) z,val
      if (z .eq. 0.0) goto 20
      vx=indx(nx,xnode,real(z))
      vy=indx(ny,ynode,aimag(z))
      if (state(vx,vy).lt.0) then
         print *,'Attempt to assign point at ', z, ' with val ',
     &      val
         print *,'It is a boundary point.'
         stop 1
      endif
      value(vx,vy)=val
      state(vx,vy)=1
      goto 10
*  Propagate each value throughout the region in which it occurs.
20    change=.false.
*  First, propagate horizontally.
      do 30, i=1,nx
         j=1
32       if (state(i,j) .eq. 1) then
            jj=j-1
35          if (state(i,jj) .eq. 0) then
```

```
                  value(i,jj)=value(i,jj+1)
                  state(i,jj)=1
                  change=.true.
                  jj=jj-1
                  goto 35
               endif
37             if (state(i,j+1) .eq. 0) then
                  j=j+1
                  value(i,j)=value(i,j-1)
                  state(i,j)=1
                  change=.true.
                  goto 37
               endif
            endif
            j=j+1
            if (j .le. ny) goto 32
30       continue
*  Then, propagate vertically.
         do 40, j=1,ny
            i=1
42          if (state(i,j) .eq. 1) then
               ii=i-1
45             if (state(ii,j) .eq. 0) then
                  value(ii,j)=value(ii+1,j)
                  state(ii,j)=1
                  change=.true.
                  ii=ii-1
                  goto 45
               endif
47             if (state(i+1,j) .eq. 0) then
                  i=i+1
                  value(i,j)=value(i-1,j)
                  state(i,j)=1
                  change=.true.
                  goto 47
               endif
            endif
            i=i+1
            if (i .le. nx) goto 42
40       continue
*  If any change occurred, loop back and propagate some more.
         if (change) goto 20
         return
         end
**********************
         subroutine cmpbnd
$include:'LTPREP.INC'
*  Compute values of nodes on the internal boundaries.
*  The value of a boundary node is the geometric mean of the
*  values of the nodes on opposite sides of it.
```

```
      integer i, j
      logical change
*  Compute as many nodes as possible; repeat until no change
*  occurs.
20    change=.false.
      do 30, i=1,nx
         do 30, j=1,ny
            if (state(i,j).eq.-1) then
               if (state(i,j-1).gt.0 .and. state(i,j+1).gt.0) then
                  value(i,j)=sqrt(value(i,j-1)*value(i,j+1))
                  state(i,j)=2
                  change=.true.
               elseif (state(i-1,j).gt.0 .and. state(i+1,j).gt.0) then
                  value(i,j)=sqrt(value(i-1,j)*value(i+1,j))
                  state(i,j)=2
                  change=.true.
               endif
            endif
30    continue
*  If any changes occurred, loop back to see if more nodes can be
*  computed.
      if (change) goto 20
      return
      end
**********************
      subroutine output(unit)
*  Writes value array, using repeat feature of List-Directed input.
$include:'LTPREP.INC'
      integer i, j, nv, rpt(limx), unit
      real val(limx)
      do 30, j=1,ny
         nv=1
         rpt(1)=1
         val(1)=value(1,j)
         do 20, i=2,nx
            if (value(i,j) .eq. val(nv)) then
               rpt(nv)=rpt(nv)+1
            else
               nv=nv+1
               rpt(nv)=1
               val(nv)=value(i,j)
            endif
20       continue
         write(unit,3000) (rpt(i),val(i),i=1,nv)
3000     format(1p,6(1x,i3,'*',e8.2))
30    continue
      return
      end
**********************
      subroutine wrtpic(unit,string)
```

```
*  Writes state array in character form representing a picture
*  of the aquifer.
$include:'LTPREP.INC'
      character string*(*), symb(-1:2)*1
      integer i, j, unit
      data symb/'+', '?', ':', '#'/
      write(unit,4000) string
4000  format(1x,'LTPREP Picture File for ',a,/)
      do 20, j=0,ny+1
         write(unit,4100) (symb(state(i,j)),i=0,nx+1)
20    continue
4100  format(1x,79a)
      return
      end
**********************
      integer function indx(n,ary,val)
      integer i, n
      real ary(n), val
*  Returns index of val in array ary, of n elements.
      do 20, i=1,n
         if (val .eq. ary(i)) then
            indx=i
            return
         endif
20    continue
      print *, 'Value ',val, ' not found in INDX.'
      stop 1
      end
**********************
      integer function length(string)
*  Returns length of string, not including trailing blanks.
      character string*(*)
      integer i
      do 20, i=len(string),1,-1
         if (string(i:i) .ne. ' ') goto 40
20    continue
      length=1
      return
40    length=i
      return
      end
```