Transport and Fate of Water and Chemicals in Laboratory-Scale, Single-Layer Aquifers

Volume 2. User's Manual for Program LTRSKAQ2

Special Report 846
October 1989

Agricultural Experiment Station
Oregon State University

TRANSPORT AND FATE OF WATER AND CHEMICALS IN LABORATORY-SCALE, SINGLE-LAYER AQUIFERS

VOLUME 2. USER'S MANUAL FOR PROGRAM LTRSKAQ2

G. A. BACHELOR, D. E. CAWLFIELD, F. T. LINDSTROM, L. BOERSMA

This is one of two volumes describing a mathematical model for the steady state flow field, transport, and fate of chemicals in laboratory-scale, single-layer aquifers.

AUTHORS: G. A. Bachelor and D. E. Cawlfield are senior systems analysts, F. T. Lindstrom is associate professor, and L. Boersma is professor, Dept. of 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 to inject chemicals into an aquifer which would enhance pollutant degradation. The model allows calculation of hydraulic pressure fields and perturbations of the field by injection and/or extraction wells.

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 Environmental Protection Agency and Oregon State University.

CONTENTS

TRANSPORT AND FATE OF WATER AND CHEMICALS IN LABORATORY-SCALE, SINGLE-LAYER AQUIFERS

VOLUME 2. USER'S MANUAL FOR PROGRAM LTRSKAQ2

I. Introduction

This report (Volume 2) is a companion document to the report (Volume 1) describing the mathematical model, LTRSKAQ2, (Lindstrom et al, 1989). As the title implies, this volume is essentially the user's guide for running the model on desk top type computers. Those readers wishing to understand the basic science on which the model is built need to read Volume 1 first. Those readers wishing only to learn how to set up data files and run the computer program need to thoroughly read this document. This document begins with a brief overview of the major sections of the computer code including the names of certain input and output files. The user is then walked through a basic installation procedure. A listing of the Fortran variables, with mathematical symbols, meaning, and units is given next. This is followed by a section on the preparation of input data files. A section on the running of the program and a discussion of the output files is next. This is followed by a section which describes the program and finally the references. The Fortran 77 program listing ends Volume 2.

I.1 Program overview

The program LTRSKAQ2 has been coded in ANSI Standard Fortran 77. It implements the mathematical model described by Lindstrom, et al, 1989. The program uses two data files. One of these specifies the dimensions and parameters for the hydrology of the aquifer. The second file specifies the parameters for the

chemistry.

The program reads the hydrology file (WATLT2.DAT); then it computes the steady-state fluid flow in the aquifer, using the Peaceman-Rachford ADII method. Next, the program reads the chemistry file (CHMLT2.DAT) and then it computes the transport and fate of chemicals in the aquifer for a specified period of time, using a modified method of characteristics (MOC).

The program produces five output files. WATLT2.OUT and CHMLT2.OUT contain the information read from files WATLT2.DAT and CHMLT2.DAT, respectively, together with computed information. The file HYDROOUT.OUT contains the hydraulic pressures and the flow velocities in the aquifer, as computed by the first part of the program. The file CHEMFOUT.OUT contains the chemical concentration distribution in the aquifer at selected times, together with the total chemical mass. Data in the chemistry input file specify how many times to print this information during the run. Print times are at equally spaced intervals. The file DEBUGLT2.OUT contains "debugging" outputs selected by control information in the file WATLT2.DAT.

II. Installing the program

The program can be run on IBM PC, IBM XT, or IBM AT computers or their equivalent, which have 640K memory and an Intel 8087/80x87 math co-processor. A hard disk is desirable, but not absolutely necessary. The program was developed under DOS 3.10, using Microsoft Fortran Version 4.01. It would probably work under some earlier versions of these systems. The distribution diskette includes both the executable form of the program, and the Fortran source files. The latter are needed if the user wishes to change the program, or if the program is to be run on a different type of computer.

There are three Batch files on the distribution diskette. INSTALL.BAT is used to install the LTRSKAQ2 program on a diskette or in a subdirectory. The other two Batch files help to run the program and to manage the output files. Two of the Batch files contain instructions for using them. These are displayed by typing the name of the Batch file (either INSTALL or SAVLTAQ2) without the ".BAT" extension and without arguments.

There is a file named READ.ME on the diskette that contains instructions for installing and running LTRSKAQ2. Those instructions are also included in this Guide.

II.1. Installation

If a hard disk is available, LTRSKAQ2 should be installed in a subdirectory on that disk. If a hard disk is not available, or if preferred, the program can be installed on a diskette. Using the installation diskette to run the program is not advised.

First, insert the installation diskette in drive A. Then enter the command

A:

to enter the root directory on the installation diskette. The

install command must be followed by a drive letter, or a drive letter plus a directory name. To make a working diskette in drive B, insert a blank, formatted diskette in drive B and enter the command:

```
INSTALL B:\
```

To make a working directory named LTAQ2 on drive C, enter the commands:

```
MKDIR   C:\LTAQ2
INSTALL C:\LTAQ2\
```

NOTE the terminating backslash (\) after the directory path on the install command.

The install command copies the executable program file, the data files, and the Batch files onto the diskette or into the specified subdirectory. The installation diskette also includes the output files generated when the program is run with the given data files. It is possible to check that LTRSKAQ2 has been installed and is working correctly, by comparing its output with the output files supplied.

See section V for instructions on running the program.

III. List of variables

This section 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 1 is shown, with a brief description of the purpose of the variable, and its units. There is a list of the meanings of the run control flags, at the end of the list of variables.

Program variables involved in input/output

Fortran variable	Math symbol	Meaning	Units
ADIMAX		ADII method heuristic parameter	
ADIPRM		ADII method convergence parameter	
C0	C_o	free stream chemical concentration	(kg/m^3)
CHMCON(I)		chemical concentrations for injection wells; location indices are stored in INJI(I) and INJJ(I)	$(\frac{\text{kg chem}}{\text{kg solution}})$
CIN	C_{in}	chemical concentration in inlet end mixing tank	(kg/m^3)
CNEW(I,J)	C	free phase chemical concentration distribution (field)	(kg/m^3)
COLD(I,J)	C	free phase chemical concentration distribution (field)	(kg/m^3)
COUT	C_{out}	chemical concentration in outlet end mixing tank	(kg/m^3)
CSWELN(I,J)	C_s	injection well chemical concentrations while wells are on; zero while wells are off	$(\frac{\text{kg chem}}{\text{kg solution}})$
DELTA		ADII method heuristic parameter	

DELTPR		time increment for printing; computed by CHMREAD	(days)
DISPLX	α_{dispx}	x-component of dispersivity	(m)
DISPLY	α_{dispy}	y-component of dispersivity	(m)
DL0	D_{L0}	free solution molecular diffusion coefficient of compound	(m^2/day)
DT0	Δt	time increment $t_{n+1} = t_n + \Delta t$	(days)
DT1		maximum t that satisfies stability criterion: $t \leq$ DT1	(days)
DX(I)	Δx_i	node spacing in x-direction; DX(I) = XNODE(I+1)-XNODE(I)	(m)
DY(J)	Δy_i	node spacing in y-direction; DY(J) = YNODE(J+1)-YNODE(J)	(m)
EPS	ϵ	volume porosity of porous medium $(\frac{m^3 \text{ voids}}{m^3 \text{ porous medium}})$	
HEAD1		first text line read from file WATLT2.DAT	
HEAD2		second text line read from file WATLT2.DAT	
HIN	H_{in}	hydraulic head field at y = 0	(m water)
HOLD(I,J)	H	hydraulic head field	(m water)
HNEW(I,J)	H	hydraulic head field	(m water)
HOUT	H_{out}	hydraulic head field at y = Ly	(m water)
I		index for arrays, x direction	
ICADII		iteration count for ADII method	
INJI(I)		I-indices for locations of injection wells	
INJJ(I)		J-indices for locations of injection wells	

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
KCLAY	K_{silt}	linear equilibrium distribution constant for silt (m^3/kg silt)
KORG	K_{org}	linear equilibrium distribution constant for organics (m^3/kg organics)
KSAND	K_{sand}	linear equilibrium distribution constant for sand (m^3/kg sand)
KTHSTS	K_s	saturated hydraulic conductivity (m/day)
LAMDA	Λ	overall first order loss coefficient (1/day)
MAXPRT		constant = maximum number of printing times allowed
MAXSWT		constant = maximum number of switching times allowed
NBSOUR		number of chemical sources in flow field
NEXTW		number of extraction wells
NFLAG(I)		run control flags (see list below)
NFUNC(I,J)		indicates quadrant location (1, 2, 3, or 4) of P* point, for each I and J
NINJW		number of injection wells
NLSOR		maximum number of sweeps allowed for any given local cycle in ADII method
NMOD		specifies how often to modify and display ADII method convergence data
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,Lx]	
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,Ly]	
NSWT		number of times injection wells are switched on or off	
PCTCLA	$\%_{silt}$	decimal percent silt in porous medium; included clay fraction	(no dim)
PCTORG	$\%_{org}$	decimal percent organics in porous medium	(no dim)
PCTSAN	$\%_{sand}$	decimal percent sand in porous medium	(no dim)
PRTIME(I)		times to print chemical data computed by CHMREAD	(days)
QCHM1S(I,J)	Q_{so}	chemical source strength of buried sources (The program reads kg chem/day; it divides the input values by the appropriate volumes)	$\left(\frac{\text{kg chem}}{m^3 \text{ day}}\right)$
QWELIN(I,J)	Q_{inj}	injection rate for wells (see note under QWELOT below)	$\left(\frac{\text{kg water}}{m^3 \text{ day}}\right)$
QWELOT(I,J)	Q_{out}	extraction rate for wells (For both QWELIN and QWELOT, the program reads kg water/day; it divides the input values by the appropriate volumes)	$\left(\frac{\text{kg water}}{m^3 \text{ day}}\right)$
RATIO		ratio "previous/current" values of RMSE in ADII method	
RETARD	R	retardation parameter	(no dim)

RHOCLA	ρ_{silt}	average particle density of silt plus clay fraction	(kg/m^3)
RHOORG	ρ_{org}	average particle density of organics	(kg/m^3)
RHOSND	ρ_{sand}	average particle density of sand	(kg/m^3)
RHOWAT	ρ_w	density of water	(kg/m^3)
RMSE		root mean square error in ADII method	
STRING		temporarily holds strings of text read from data files	
SWTIME(I)		times to switch injection wells on or off	(days)
TO		starting time for chemical processing (usually 0.0)	(days)
TIME	t	elapsed time in days from commencing injection and/or pumping	(days)
TLRNWA		tolerance for root mean square error convergence criterion for ADII method	
TMAX		ending time for chemical processing	(days)
TORT	α_{tort}	tortuosity factor (0.67)	(no dim)
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 "effective" average inter-void velocity	(m/day)
VLYY(I,J)	U_y^*	y-component of "effective" average inter-void velocity (VLXX and VLYY are subsequently changed to represent U_x^* and U_y^*)	(m/day)

XLAM1O	λ_{met}	rate constant for first order loss in free phase due to microbial action	(1/day)
XLAMIR	λ_{irr}	rate constant for first order loss in free phase due to all irreversible loss processes, e.g. pumping, chemical reaction	(1/day)
XLAMRA	λ_{rad}	rate constant for first order loss in free phase due to radioactive decay	(1/day)
XLW	L_w	thickness of the aquifer	(m)
XLYIN	L_{in}	length of the inlet end mixing tank	(m)
XLYOUT	L_{out}	length of the exit end mixing tank	(m)
XMASIN		cumulative chemical mass inflow or outflow at inlet end	(kg)
XMASOT		cumulative chemical mass inflow or outflow at exit end	(kg)
XMASS		total chemical mass in aquifer	(kg)
XMFONW		cumulative first order loss of chemical mass	(kg)
XMSOUR		cumulative chemical mass from buried sources	(kg)
XNODE(I)	x	transverse component	(m)
XNODE(NSLXP1)	Lx	transverse coordinate width of aquifer	(m)
XSLM1O	λ^s_{met}	rate constant for first order loss in the sorbed phase due to microbial action	(1/day)
XSLMRA	λ^s_{rad}	rate constant for first order loss in the sorbed phase due to radioactive decay	(1/day)
YNODE(I)	y	longitudinal component	(m)
YNODE(NSLYP1)	Ly	longitudinal coordinate length of aquifer	(m)

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

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 7, 10, 11, 12, 13, and 14, means that the information is written on the file DEBUGLT2.OUT. Flags not mentioned, such as NFLAG(2), are not used in this version of the program.

NFLAG(1)=0 means: compute the water pressure field only.
NFLAG(1)=1 means: compute the water pressure field and the velocity components.
NFLAG(1)=2 means: compute the water pressure field, the velocity components, the dispersion components, and the dynamic chemical field distribution.

NFLAG(3)=1 means: display the hydraulic field convergence data.
NFLAG(3)=0 means: skip the above display.

NFLAG(4)=0 means: modify ADIPRM every NMOD'th iteration, using DELTA and ADIMAX.
NFLAG(4)=1 means: read in a new value, from the keyboard, for ADIPRM every NMOD'th iteration.

NFLAG(7)=1 means: write the coordinates of the P* points.
NFLAG(7)=0 means: do not write the 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(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 Thomas algorithm matrix elements and the complete "known vector" components, every time through the algorithm.
NFLAG(13)=0 means: do not write any Thomas algorithm matrix and/or "known vector" components.

NFLAG(14)=1 means: write the matrix elements defining the chemical field.
NFLAG(14)=0 means: do not write the matrix elements.

NOTE: the outputs enabled by NFLAGS 7, 10, 11, 12, 13, and 14 are written on the file DEBUGLT2.OUT.

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. Example input hydrology data file: WATLT2.DAT

This section shows an example hydrology and aquifer geometry file. The file must be named WATLT2.DAT; it is read in by subroutine FLOREAD in module RWWLT2.FOR. A complete listing of the file is shown followed by a detailed discussion of the data.

```
   EXAMPLE WATLT2.DAT DATA FILE FOR MODEL LTRSKAQ2.  8/16/1989
   TEST RUN NUMBER 1 - SINGLE INJECTION WELL
ADIPRM  NLSOR  TOLERANCE  DELTA  ADIMAX  NMOD  ZTHRSH  (RUN CONTROL DATA)
 100.0   3000    1.0E-8     0.2   200.0    20   1.0E-9
DECISION FLAGS
2 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
X  Y  (NO. OF INTERIOR NODES)
13 25
X-COORD POSITIONS (M)
0.0      0.0453   0.136    0.273    0.41     0.547    0.578
0.61     0.6465   0.682    0.819    0.956    1.084    1.1747
1.22
Y-COORD POSITIONS (M)
0.0      0.05     0.1      0.25     0.40     0.55     0.70
0.85     1.0      1.07     1.14     1.22     1.32     1.42
1.52     1.604    1.6733   1.75     1.833    1.917    2.00
2.125    2.25     2.375    2.5      2.75     3.00
AQUIFER THICKNESS (M)
0.3
INLET AND EXIT PORT TANK LENGTHS (M)
.1        .1
 SAND    CLAY   ORGANICS  (SOIL PARTICLE DENSITIES KG/M^3)
2660.0  2650.0   1300.0
WATER DENSITY (KG/M^3)
1000.0
KTHSTS   TORT    EPS    %SAND   %CLAY   %ORG   DISPLX   DISPLY
5.616    0.67   0.365   0.996   0.004    0.0   0.0035   0.0035
NO. OF INJECTION AND EXTRACTION WELLS
          1              1
I  J  STRENGTH (INJECTION WELLS) (KG WATER/DAY)
8 10   9.008
I  J  STRENGTH (EXTRACTION WELLS) (KG WATER/DAY)
2  2   0.0
INLET AND EXIT PORTS PRESSURE HEADS (M)
0.148     0.09
```

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 WATLT2.OUT and HYDROOUT.OUT. 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:

Reading from data file WATLT2.DAT.

First, read two-line heading HEAD1, HEAD2 that will be printed out on files WATLT2.OUT and HYDROOUT.OUT:

```
EXAMPLE WATLT2.DAT DATA FILE FOR MODEL LTRSKAQ2.  8/16/1989
TEST RUN NUMBER 1 - SINGLE INJECTION WELL
```

Read run control information:

```
ADIPRM  NLSOR  TOLERANCE  DELTA  ADIMAX  NMOD  ZTHRSH  (RUN CONTROL DATA)
 100.0   3000    1.0E-8     0.2   200.0    20  1.0E-9
```

ADIPRM, NLSOR, TLRNWA, DELTA, ADIMAX, and NMOD are the ADII method parameters; they are discussed in section IV.3. ZTHRSH is the "zero threshold"; when arrays are printed out, numbers whose absolute values are less than ZTHRSH are printed as ".00".

Read control flags NFLAG(I) for I=1 to 20:

```
DECISION FLAGS
2 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

In this case:

NFLAG(1) = 2 means compute the water pressure field, the velocity components, the dispersion components, and the dynamic chemical field distribution. Choose NFLAG(1) = 2 if computation of both the hydrology and the chemistry is desired. Choose 0 or 1 if computation of the hydrology only is desired. This would be appropriate while attempting to find suitable values for the ADII method variables.

NFLAG(3) = 1 means display the hydraulic field convergence data. Choosing NFLAG(3) = 1 is recommended; this causes the program to display the ADII method convergence data every NMODth iteration. This way, it can be ascertained whether the method is converging.

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
2 0 1 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, 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 at the end of section III.

Read grid geometry:

First, read number of interior x and y nodes NSLXM1, NSLYM1:

```
X  Y  (NO. OF INTERIOR NODES)
13 25
```

Compute number of intervals:

```
  NSLXXX = NSLXM1+1
  NSLYYY = NSLYM1+1
```

Compute number of nodes, including boundary nodes:

```
  NSLXP1 = NSLXXX+1
  NSLYP1 = NSLYYY+1
```

Make sure grid fits within fixed array sizes:

```
  If NSLXP1>IX or NSLYP1>IY then
    print ' Too many grid points for fixed arrays.'
          ' Change IX and/or IY, re-compile, & re-link.'
    and stop.
```

Note: 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.LT2, 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.

Initialize arrays, now that size of problem is established.

```
For I = 1 to NSLXP1
  For J = 1 to NSLYP1
    Set arrays QWELIN(I,J), QWELOT(I,J), QCHM1S(I,J),
       HOLD(I,J), COLD(I,J), CSWELN(I,J) to zero.
```

Read nodal positions XNODE(I) for I=1 to NSLXP1, remembering that NSLXP1 is two (2) greater than the value NSLXM1 that was read in:

```
X-COORD POSITIONS (M)
0.0     0.0453  0.136   0.273   0.41    0.547   0.578
0.61    0.6465  0.682   0.819   0.956   1.084   1.1747
1.22
```

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.

Read nodal positions YNODE(I) for I=1 to NSLYP1, remembering that NSLYP1 is two (2) greater than the value NSLYM1 that was read in:

```
Y-COORD POSITIONS (M)
0.0     0.05    0.1     0.25    0.40    0.55    0.70
0.85    1.0     1.07    1.14    1.22    1.32    1.42
1.52    1.604   1.6733  1.75    1.833   1.917   2.00
2.125   2.25    2.375   2.5     2.75    3.00
```

See the note above under X-COORD POSITIONS.

Read the aquifer thickness XLW (M):

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

Read the inlet and exit port tank lengths XLYIN, XLYOUT (M):

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

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

Calculate 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

Read particle density of soil components RHOSND, RHOCLA, RHOORG:

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

Read water density RHOWAT:

WATER DENSITY (KG/M^3)
1000.0

Read porous medium characterizing parameters:

KTHSTS	TORT	EPS	%SAND	%CLAY	%ORG	DISPLX	DISPLY
5.616	0.67	0.365	0.996	0.004	0.0	0.0035	0.0035

KTHSTS is the hydraulic conductivity; TORT is the tortuosity; EPS is the porosity; PCTSAN, PCTCLA, and PCTORG are the percentages of sand, silt, and organics in the medium; DISPLX and DISPLY are the x and y components of the dispersivity. KTHSTS has units of M/DAY; DISPLX and DISPLY have units of M; the other variables are dimensionless.

Read 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

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

Read I, J, QWELIN(I,J) for NINJW times:

```
I  J  STRENGTH (INJECTION WELLS) (KG WATER/DAY)
8 10   9.008
```

There must be NINJW data lines, following the ONE text line. The values of I and J for the injection wells are saved in the arrays INJI(K) and INJJ(K), so that the chemical flow from these wells can be switched on or off. The positions of wells are specified by indices, NOT by the X and Y coordinates. 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.

Read I, J, QWELOT(I,J) for NEXTW times:

```
I  J  STRENGTH (EXTRACTION WELLS) (KG WATER/DAY)
2  2   0.0
```

There must be NEXTW data lines, following the ONE text line. See the notes above under INJECTION WELLS concerning the indices and the strength data. Here there are no extraction wells, so a well of strength 0 is placed at position (2,2). Wells must be at interior nodes, and cannot be on the boundaries, which is why there cannot be a well at position (1,1).

Read inlet and outlet boundary hydraulic heads HIN, HOUT (Meters):

```
INLET AND EXIT PORTS PRESSURE HEADS (M)
0.148     0.09
```

IV.2. Example input chemistry data file: CHMLT2.DAT

This section shows an example of the chemical parameter data file. The file must be named CHMLT2.DAT; it is read in by subroutine CHMREAD in module RWCLT2.FOR. A complete listing of the file is shown followed by a detailed discussion of the data.

```
   EXAMPLE CHMLT2.DAT DATA FILE.  8/2/1989
   TEST RUN NUMBER 1 - SINGLE INJECTION WELL
NPRT   TO  TMAX   DTO  (DAYS)  (RUN CONTROL DATA)
  6   0.0  0.50  0.01
DLO (DIFFUSION) (M^2/DAY) KSAND   KCLAY   KORG (M^3/KG)
  0.000143                 0.0     0.0     0.0
INLET TANK   EXIT TANK   STREAM  (CHEM CONC.) (KG/M^3)
    0.0         0.0        0.0
INJECTION WELL CONC.S (KG CHEM/KG SOLUTION)
     0.0002
NO. OF SWITCHING TIMES
  1
SWITCHING TIMES FOR INJECTION WELLS (DAYS)
  0.0
NUMBER OF BURIED SOURCES
  1
I  J  CONCENTRATION OF BURIED SOURCE (KG CHEM/DAY)
2  2   0.0
XLAM1O  XLAMIR  XLAMRA  XSLM1O  XSLMRA (FIRST ORDER LOSSES) (1/DAY)
  0.0     0.0     0.0     0.0     0.0
INITIAL CHEM. CONC. DIST. (KG/M^3)
15*0.0   ( 1)
15*0.0   ( 2)
15*0.0   ( 3)
15*0.0   ( 4)
15*0.0   ( 5)
15*0.0   ( 6)
15*0.0   ( 7)
15*0.0   ( 8)
15*0.0   ( 9)
15*0.0   (10)
15*0.0   (11)
15*0.0   (12)
15*0.0   (13)
15*0.0   (14)
15*0.0   (15)
15*0.0   (16)
15*0.0   (17)
15*0.0   (18)
15*0.0   (19)
15*0.0   (20)
15*0.0   (21)
15*0.0   (22)
15*0.0   (23)
15*0.0   (24)
15*0.0   (25)
15*0.0   (26)
15*0.0   (27)
```

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 (WATLT2.DAT). The one additional feature of Fortran's list-directed input that is used in this example is the "repeat" factor. That feature is described at the end of this section.

Reading from data file CHMLT2.DAT.

First, read two-line heading, using variable STRING for both lines, which will be printed out on files CHMLT2.OUT and CHEMFOUT.OUT:

```
EXAMPLE CHMLT2.DAT DATA FILE.  8/2/1989
TEST RUN NUMBER 1 - SINGLE INJECTION WELL
```

Read run control data NPRT, T0, TMAX, DT0:

```
NPRT   T0  TMAX   DT0  (DAYS)  (RUN CONTROL DATA)
  6   0.0  0.50  0.01
```

NPRT is the number of times to print the distribution of chemical during the run; T0 is the starting time, usually 0; TMAX is the time to end the run; 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 1 (section V.6), delta-T must satisfy a certain stability criterion in order for the method to be stable. The program computes this criterion (DT1), and if the delta-T chosen by the user is greater than one-half of DT1, the program sets DT0 to one-half of DT1, thus ensuring that the process will be stable. The value of DT1 is printed out on the file CHMLT2.OUT, as "Computed DTMAX."

Compute the print times:

```
DELTPR = (TMAX-T0)/NPRT
For I=1 to NPRT
  PRTIME(I) = DELTPR*I + T0 - DT0*0.01
```

The print times are equally spaced during the run.

Display the print times PRTIME(I) for I=1 to NPRT.

Read the chemical parameters:

DLO (DIFFUSION) (M^2/DAY)	KSAND	KCLAY	KORG (M^3/KG)
0.000143	0.0	0.0	0.0

DLO is the diffusion coefficient; KSAND, KCLAY, and KORG are the linear equilibrium distribution constants.

Read inlet and exit port boundary chemical concentrations CIN, COUT, CO:

INLET TANK	EXIT TANK	STREAM (CHEM CONC.) (KG/M^3)
0.0	0.0	0.0

CIN and COUT are the initial chemical concentrations in the inlet and outlet tanks, respectively. CO is the free stream chemical concentration.

Read injection well chemical concentrations (kg chem/kg solution). The minimum number is one well, which may have strength zero.

Read NINJW (number of injection wells) concentrations CHMCON(I):

INJECTION WELL CONC.S (KG CHEM/KG SOLUTION)
0.0002

There must be NINJW concentrations on one or more data lines, following the ONE text line; they must be in the same order as the injection wells listed in the WATLT2.DAT file. Remember that NINJW and the coordinates I, J are read from the WATLT2.DAT file.

Read number of switching times NSWT (must be at least 1):

NO. OF SWITCHING TIMES
1

Read NSWT switching times SWTIME(I) for the injection wells:

```
SWITCHING TIMES FOR INJECTION WELLS (DAYS)
  0.0
```

The SWTIME's are the times (in days) at which the injection wells are to be switched on or off. The switching times must be in increasing order on one or more data lines, following the ONE text line. The first, third, fifth, etc., times are "switch on" times; the second, fourth, etc., times are "switch off" times. There must be at least one switching time.

The amount of fluid being injected must remain constant throughout the run, because a change in injection rate would change the hydraulic pressure distribution. "Switching" means turning the chemical on or off. When the wells are on, the fluid contains the specifed concentration of chemical (CHMCON and CSWELN); when they are off, only fluid is being injected. The SWTIME's apply to all injection wells. The first time specifies when the wells are switched on for the first time. The second time, if any, specifies when the wells are switched off. The third time switches them on, etc. The times must be in increasing order because the program does not sort them. If the first time is equal to or less than T0 (the starting time), the wells are switched on at the beginning of the run. If the first time is greater than TMAX (the ending time), the wells are never switched on. Any SWTIME's greater than TMAX have no effect on the run.

The actual times at which wells are switched will be at the TIME points (T0, T0+DT0, T0+2*DT0, ...) closest to the specified times. DT0 is "small" for the method used in this program, so switching the wells at the nearest time points is sufficiently close to the specified times.

Read number of buried chemical sources NBSOUR, where the minimum is one source, which may have strength zero:

```
NUMBER OF BURIED SOURCES
  1
```

Read buried source positions and strengths.

Read I, J, QCHM1S(I,J) for NBSOUR times:

```
I  J  CONCENTRATION OF BURIED SOURCE (KG CHEM/DAY)
2  2   0.0
```

There must be NBSOUR data lines, following the ONE text line. Note that the location is given by indices, not by coordinates, and the source must be at an interior point, not on the boundary. Here there are no buried sources, so a source of strength 0 is placed at position (2,2). Also note that the input data gives the concentration in units of (kg chem/day); the program divides the data by the appropriate volumes to get units of (kg chem/m^3 day).

Read first order loss parameters:

```
XLAM1O  XLAMIR  XLAMRA  XSLM1O  XSLMRA (FIRST ORDER LOSSES) (1/DAY)
  0.0     0.0     0.0     0.0     0.0
```

XLAM1O, XLAMIR, and XLAMRA are the rate constants for first order losses in the free phase due to microbial action, irreversible processes, and radioactive decay, respectively; XSLM1O and XSLMRA are the rate constants for first order losses in the sorbed phase due to microbial action and radioactive decay, respectively.

Read initial chemical distribution at time zero.
For J=1 to NSLYP1
  Read COLD(I,J) for I=1 to NSLXP1:

```
INITIAL CHEM. CONC. DIST. (KG/M^3)
15*0.0   ( 1)
15*0.0   ( 2)
15*0.0   ( 3)
15*0.0   ( 4)
15*0.0   ( 5)
15*0.0   ( 6)
```

```
15*0.0   ( 7)
15*0.0   ( 8)
15*0.0   ( 9)
15*0.0   (10)
15*0.0   (11)
15*0.0   (12)
15*0.0   (13)
15*0.0   (14)
15*0.0   (15)
15*0.0   (16)
15*0.0   (17)
15*0.0   (18)
15*0.0   (19)
15*0.0   (20)
15*0.0   (21)
15*0.0   (22)
15*0.0   (23)
15*0.0   (24)
15*0.0   (25)
15*0.0   (26)
15*0.0   (27)
```

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, 15*0.0 represents 15 values of 0.0. The integers in parentheses, (1), ... (27), show the value of J for each line; they are ignored by the READ statement in the program, because they are "excess" data. As an example, if one wanted to make the sixth number in line 19 have the value 1.23, line 19 above would be changed to:

```
5*0.0  1.23  9*0.0   (19)
```

This represents 5 zeroes, followed by 1.23, followed by 9 zeroes; a total of 15 numbers.

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 WATLT2.DAT file.

IV.3. Choosing the ADII method parameters

The program uses an iterative method, called the ADII method, to compute the steady-state water pressure at the nodal points in the aquifer. ADIPRM, ADIMAX, DELTA, and NMOD are heuristic parameters used in the ADII method, to help improve the rate of convergence. The user must choose appropriate values for these parameters; the choices depend on certain variables in the input data. If the choices are reasonably good, the computed water pressure will be the same, regardless of the values chosen. There may be a few very small differences for different choices; however, if ADIPRM is too large, the program may converge to a wrong answer! In addition to the variables mentioned above, the user must also choose the tolerance for convergence TLRNWA, and the iteration limit NLSOR. In the following discussion, it is assumed that NFLAG(4) = 0; toward the end, there is a discussion of the "interactive" mode, with NFLAG(4) = 1.

At each step in the iteration, ADIPRM is used in the computation of an improved approximation to the water pressure. At the end of each step, the program computes the root mean square error RMSE, which is a measure of the amount of change in the array representing the pressure. The value of RMSE should keep decreasing; when it becomes smaller than TLRNWA, the computed pressure values are accepted. DELTA, NMOD, and ADIMAX control the way in which ADIPRM is changed. Every NMODth iteration, ADIPRM is compared with ADIMAX. If ADIPRM is less than ADIMAX, ADIPRM is multiplied by (1 + DELTA); otherwise, ADIPRM is set back to the initial value that was read in.

As discussed under NFLAG's in section IV.1, the choice of NFLAG(3) = 1 is recommended, so that the program will display the ADII method convergence data. Experimentation, involving a number of runs, may be needed to find "good" values for the parameters involved in the ADII method. If NFLAG(3) = 1, the

program will display the values of ADIPRM, RMSE, RATIO, and ICADII every NMODth iteration. If NFLAG(3) = 0, only a dot is displayed. RATIO is the ratio of successively printed values of RMSE; larger values of RATIO indicate more rapid convergence. If RATIO is less than 1.0, the method is diverging; this is usually a temporary situation. ICADII is the number of iterations that has been performed. NLSOR is another limit chosen by the user. If the number of iterations ICADII exceeds NSLOR, the program stops and displays an error message.

The example WATLT2.DAT data file includes, on its fourth line, values for the ADII parameters that work reasonably well for the example scenario. The tolerance TLRNWA = 1.0E-8 assures good accuracy for the computed pressures. NLSOR = 3000 is a very generous limit; it should not take that many iterations to reach convergence. The values ADIPRM = 100.0, DELTA = 0.2, ADIMAX = 200.0, and NMOD = 20 have been found through numerous trials to be "good" choices for the scenario in the example file.

It is difficult to give good guidelines for choosing these variables. Choosing a smaller value for NMOD results in more frequent display of the ADII convergence data, as well as more frequent modification of ADIPRM. We have used values in the range from 2 to 20 for NMOD. For "well-behaved" scenarios like the one in the example, values of DELTA in the range 0.05 to 0.4 have been tried; even DELTA = 0.0 works. For more intractable scenarios, we have used values as large as 900.0 for DELTA.

The most critical variables are ADIPRM and ADIMAX. DELTA is also important, but less critical. The proper choices of ADIPRM and ADIMAX depend on the hydraulic conductivity KTHSTS and on the node spacing. These choices vary directly with KTHSTS; if KTHSTS were 4 times larger, then ADIPRM and ADIMAX should also be 4 times larger. The choices of ADIPRM and ADIMAX vary inversely as the square of the node spacing. If the nodes were spaced 5 times farther apart, then these variables should be divided by 25, the

square of 5.

If the choice of the variables is good, the displayed value of RATIO should be relatively "large", say 2 or larger; the larger the better. It will vary during the process, perhaps increasing or decreasing. If the RATIO is very small, somewhat larger than 1, and stays small but varying, then ADIPRM was probably too small. If the RATIO steadily decreases and approaches the value (1+DELTA), then ADIPRM is probably much too large; the program may converge to a wrong answer. Note that the discussion in this paragraph applies to the situation when NMOD = 20. If NMOD is smaller, then RATIO will also be smaller.

For difficult scenarios, one can use an "interactive" mode. If NFLAG(4) = 1, the program will display a "prompt" message every NMODth iteration and allow the user to enter a new value for ADIPRM. The value is read with Fortran's list-directed format; any reasonable number representation is acceptable. In this mode of usage, the values of DELTA and ADIMAX do not matter.

When NFLAG(4) = 0, ADIMAX, DELTA, NMOD, and the initial value of ADIPRM actually specify a sequence of values for ADIPRM (Varga, 1962, pages 209-217). It is, of course, desirable to try to find values for the variables that minimize the number of iterations needed for convergence. It is nearly impossible to analytically find a sequence of values for ADIPRM that is optimum for the general case with non-uniform spacing of nodes. An analytic solution is possible when the nodes are uniformly spaced and the aquifer is square.

In typical program runs, the time to compute the pressure by the ADII method is much smaller than the time to do the chemistry processing, so one shouldn't spend too much time trying to find the "best" values for the ADII parameters.

V. Running the program and output files.

Instructions for running the program

(1) The executable program must have been installed, as described in section II. It will be in a file named LTRSKAQ2.EXE. The Batch files must also have been copied from the installation diskette. They will be in files having the extension BAT.

(2) The two data files WATLT2.DAT AND CHMLT2.DAT must be prepared as described in section IV. They should be on the same diskette or in the same subdirectory as the program and Batch files.

(3) The system must be set so that its "current" directory and drive is the one where the program and data files reside.

If the files are on a diskette, type

```
A:
```

or,

```
B:
```

according to which drive the diskette is in.

If the files are in the same subdirectory of a hard disk type:

```
C:
CD \LTAQ2
```

The notation "\LTAQ2" should be replaced by the actual "path", if it is different from this one.

(4) The batch file RUNLTAQ2.BAT should be used to run the program. There are two choices here. Typing

```
RUNLTAQ2
```

with no arguments has the same effect as typing the name of the

program:

LTRSKAQ2

In either case, the program produces the five output files having the extension ".OUT". See the discussion later in this section.

The other way to run the program is to use a "root" name as an argument:

RUNLTAQ2 [rootname]

This will run the program and then call the SAVLTAQ2 batch file to copy the output files except DEBUGLT2.OUT into files with the common "rootname" and various extensions. This is useful when it is desirable to save the output files from several runs. If LTRSKAQ2 has been run without invoking SAVLTAQ2, then SAVLTAQ2 can be run directly to "save" the files:

SAVLTAQ2 [rootname]

Typing SAVLTAQ2 with no argument will cause it to display information describing what it does.

Program and data files not in same place

The system must be set to the drive and subdirectory where the data files reside. See step (3) above. The Batch files RUNLTAQ2.BAT and SAVLTAQ2.BAT must be in the same drive and subdirectory as the data files. However, the executable program file can be somewhere else. In this case, the batch file RUNLTAQ2.BAT must be modified so that it specifies the correct path when calling LTRSKAQ2. For example, on the computers used in developing this program, LTRSKAQ2.EXE is in the "path" C:\RSKERL\LTAQ2\. In this case, the two lines in RUNLTAQ2.BAT that read "ltrskaq2" must be changed to "c:\rskerl\ltaq2\ltrskaq2".

An alternative is to leave RUNLTAQ2.BAT as distributed, and copy the executable file LTRSKAQ2.EXE into the subdirectory containing the data files. However, this takes up much more

memory; the batch files are much smaller then the executable file.

Running the program

When the program is run, it will display information about itself as shown below and wait for the user to press the ENTER (or RETURN) key.

```
        TWO-DIMENSIONAL STEADY WATER FLOW
        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 is
isotropic and homogeneous.  Velocity components,
dispersion coefficients, and other variables
are calculated at each nodal point.  Chemical
concentration values are computed at each nodal
point at each selected print time.  The flow field
is confined to the interior of the rectangular
boundaries.  Finite Difference (space centered)
methods are used for both fields.

++++++++++++++++++++++++++++++++++++++++++++++++++++
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
++++++++++++++++++++++++++++++++++++++++++++++++++++
Press ENTER or RETURN to continue:
```

If it is decided at this point not to run this program, type Control C or Control Break. Otherwise, press ENTER (or RETURN) as directed.

The program will begin processing and displaying messages as it proceeds. On a fast computer, most of these messages disappear from the screen before they can be read. If the program should stop due to a data error, or some other unexpected reason, the last message(s) displayed will give an indication of the problem that caused the stoppage.

Using NFLAG(3)=1 in the WATLT2.DAT file is recommended, as discussed in section IV. This causes the program to display the

convergence parameters for the ADII method during the hydrology processing. This indicates whether or not the method is converging. The display when the program is used on the example data files is shown below:

```
Beginning master loop for the hydraulic field

     ADIPRM          RMSE          RATIO       ICADII
  1.00000E+02   1.40333E-02   7.12591E+03        20
  1.20000E+02   1.55163E-03   9.04425E+00        40
  1.44000E+02   2.55283E-04   6.07807E+00        60
  1.72800E+02   4.61051E-05   5.53697E+00        80
  2.07360E+02   1.02864E-05   4.48213E+00       100
  1.00000E+02   1.99373E-06   5.15939E+00       120
  1.20000E+02   2.84466E-07   7.00869E+00       140
  1.44000E+02   5.13745E-08   5.53711E+00       160
  1.72800E+02   9.40371E-09   5.46321E+00       179
  Took     179 iterations.
```

If the ADII method fails to converge, or if it converges too slowly, the program can be stopped by pressing Control C or Control Break. The parameters in the WATLT2.DAT file can be revised for a new try.

If the hydraulic pressure field is successfully computed, and NFLAG(1)=2, the program will go on to read the chemistry data in file CHMLT2.DAT and compute the chemical transport. During this phase, it will display the simulated time at intervals specified by DT0 (delta T). For the example data, this display is shown below with some lines omitted:

```
Beginning master loop for the dynamic chemical field

Time =        .0100
Time =        .0200
Time =        .0300
Time =        .0400
Time =        .0500
...    ...    ...    ...    ...    ...    ...    ...
Time =        .4700
Time =        .4800
Time =        .4900
Time =        .5000

Time t meets or exceeds TMAX! CEASE COMPUTING!

Writing chemical conc. distrib. to CHEMFOUT.OUT

End of this simulation run.
Normal Fortran Termination
```

Output files

Also shown above are the final messages from the program. At this point, the output files can be examined or printed out.

File WATLT2.OUT shows the information read from WATLT2.DAT, together with computed data. File CHMLT2.OUT plays a similar role with respect to CHMLT2.DAT. Both of these output files should be examined to be sure that the data were read in correctly. All of the numbers in these two files are labeled to indicate what they represent.

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

The file CHEMFOUT.OUT shows the computed chemical concentrations at selected print times.

The file DEBUGLT2.OUT shows "debugging" information, if any of this information was selected by setting certain NFLAG's. See the discussion of NFLAG's at the end of section III. If none of these NFLAG's were set to 1, DEBUGLT2.OUT will be empty.

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.

If the program was run using the second method, with a "root" name argument, then copies of four of the output files will have been made. For example, if the program is run by the command

```
RUNLTAQ2  TEST3
```

then the following copies are made:

Output file	Copy file
CHEMFOUT.OUT	TEST3.CMF
CHMLT2.OUT	TEST3.CHM
DEBUGLT2.OUT	(not copied)
HYDROOUT.OUT	TEST3.HYD
WATLT2.OUT	TEST3.WAT

The listing of the WATLT2.OUT file produced when the program is run with the example data files as input is shown below.

```
 EXAMPLE WATLT2.DAT DATA FILE FOR MODEL LTRSKAQ2.  8/16/1989
 TEST RUN NUMBER 1 - SINGLE INJECTION WELL

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

  RUN CONTROL INFORMATION.
  ADIPRM=  1.00000E+02  NLSOR=  3000  TLRNWA=  1.00000E-08
  DELTA=   2.00000E-01  NMOD=     20  ADIMAX=  2.00000E+02
  ZTHRSH=  1.00000E-09

  NFLAG(I)=  2  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

       NODE COORDINATES (M)

  XNODE(I)
    1        2        3        4        5        6        7        8
0.       .04530   .13600   .27300   .41000   .54700   .57800   .61000

    9       10       11       12       13       14       15
.64650   .68200   .81900   .95600   1.0840   1.1747   1.2200


  YNODE(J)
    1        2        3        4        5        6        7        8
0.       .05000   .10000   .25000   .40000   .55000   .70000   .85000

    9       10       11       12       13       14       15       16
1.0000   1.0700   1.1400   1.2200   1.3200   1.4200   1.5200   1.6040

   17       18       19       20       21       22       23       24
1.6733   1.7500   1.8330   1.9170   2.0000   2.1250   2.2500   2.3750

   25       26       27
2.5000   2.7500   3.0000


  DX(I)
    1        2        3        4        5        6        7        8
.04530   .09070   .13700   .13700   .13700   .03100   .03200   .03650

    9       10       11       12       13       14
.03550   .13700   .13700   .12800   .09070   .04530
```

```
   DY(J)
     1        2        3        4        5        6        7        8
 .05000   .05000   .15000   .15000   .15000   .15000   .15000   .15000

     9       10       11       12       13       14       15       16
 .07000   .07000   .08000   .10000   .10000   .10000   .08400   .06930

    17       18       19       20       21       22       23       24
 .07670   .08300   .08400   .08300   .12500   .12500   .12500   .12500

    25       26
 .25000   .25000


   WIDTH     OF AQUIFER (M)= 1.220E+00
   LENGTH    OF AQUIFER (M)= 3.000E+00
   THICKNESS OF AQUIFER (M)= 3.000E-01

   INLET PORT TANK LENGTH (M)=  1.000E-01
   EXIT  PORT TANK LENGTH (M)=  1.000E-01

   INLET/OUTLET HYDRAULIC PRESSURES (M WATER)
   HIN=  1.48000E-01  HOUT=  9.00000E-02

        BASIC SOIL CHARACTERIZING PARAMETERS (KG/M^3)

   RHOSND=  2.660E+03  RHOCLA=  2.650E+03  RHOORG= 1.300E+03

   RHOWAT=  1.000E+03

        TABLE OF SOIL PROPERTIES

   KTHSTS=  5.61600E+00 (M/DAY)

   TORT=    6.70000E-01

   EPS=     3.65000E-01 (M^3/M^3)

   PCTSAN=  9.96000E-01

   PCTCLA=  4.00000E-03

   PCTORG=  0.00000E+00

   DISPLX=  3.50000E-03 (M)

   DISPLY=  3.50000E-03 (M)


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

     I   J  QWELIN(I,J)
   --------------------
     8  10   1.25242E+04


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

     I   J  QWELOT(I,J)
   --------------------
   NO NON-ZERO EXTRACTION WELLS.
```

The listing of the CHMLT2.OUT file produced when the program

is run with the example data files as input is now shown. The format of the print-out for the "initial chemical distribution" is the narrow format, produced when NFLAG(8) = 0. If NFLAG(8) =1 in the WATLT2.DAT data file, a wider format is printed, with 12 columns of the matrix printed on each "page", instead of 7.

```
      EXAMPLE CHMLT2.DAT DATA FILE.  8/2/1989
      TEST RUN NUMBER 1 - SINGLE INJECTION 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=  0.000E+00  TMAX=  5.000E-01  DTO=  1.000E-02
   COMPUTED DTMAX=  2.150E-02

   NPRT=     6;  PRINT TIMES (DAYS) ARE:

.08323    .16657    .24990    .33323    .41657    .49990

        CHEMICAL PARAMETERS

   DLO=  1.430E-04 (M^2/DAY)

   KSAND=  0.000E+00 (M^3/KG SAND)
   KCLAY=  0.000E+00 (M^3/KG SILT)
   KORG=   0.000E+00 (M^3/KG ORGANICS)

   CIN=    0.000E+00 (KG/M^3)
   COUT=   0.000E+00 (KG/M^3)
   CO=     0.000E+00 (KG/M^3)


        TABLE OF FIRST ORDER LOSS PROCESS COEFFICIENTS (1/DAY)

   XLAM1O=  0.00000E+00

   XLAMIR=  0.00000E+00

   XLAMRA=  0.00000E+00

   XSLM1O=  0.00000E+00

   XSLMRA=  0.00000E+00


   BURIED CHEMICAL SOURCE CONCENTRATION (KG CHEM/M^3 DAY)

     I   J  QCHM1S(I,J)
   --------------------
   NO NON-ZERO BURIED SOURCES.


   INJECTION WELLS CHEMICAL CONCENTRATION (KG CHEM/KG SOLUTION)

     I   J  CSWELN(I,J)
   --------------------
```

8 10 2.000E-04

SWITCHING TIMES FOR INJECTION WELLS (DAYS)

0.

INITIAL CHEMICAL DISTRIBUTION (KG/M^3)

PAGE 1

Y		X 0.	.0453	.1360	.2730	.4100	.5470	.5780
		1	2	3	4	5	6	7
0.	1	0.	0.	0.	0.	0.	0.	0.
.0500	2	0.	0.	0.	0.	0.	0.	0.
.1000	3	0.	0.	0.	0.	0.	0.	0.
.2500	4	0.	0.	0.	0.	0.	0.	0.
.4000	5	0.	0.	0.	0.	0.	0.	0.
.5500	6	0.	0.	0.	0.	0.	0.	0.
.7000	7	0.	0.	0.	0.	0.	0.	0.
.8500	8	0.	0.	0.	0.	0.	0.	0.
1.000	9	0.	0.	0.	0.	0.	0.	0.
1.070	10	0.	0.	0.	0.	0.	0.	0.
1.140	11	0.	0.	0.	0.	0.	0.	0.
1.220	12	0.	0.	0.	0.	0.	0.	0.
1.320	13	0.	0.	0.	0.	0.	0.	0.
1.420	14	0.	0.	0.	0.	0.	0.	0.
1.520	15	0.	0.	0.	0.	0.	0.	0.
1.604	16	0.	0.	0.	0.	0.	0.	0.
1.673	17	0.	0.	0.	0.	0.	0.	0.
1.750	18	0.	0.	0.	0.	0.	0.	0.
1.833	19	0.	0.	0.	0.	0.	0.	0.
1.917	20	0.	0.	0.	0.	0.	0.	0.
2.000	21	0.	0.	0.	0.	0.	0.	0.
2.125	22	0.	0.	0.	0.	0.	0.	0.
2.250	23	0.	0.	0.	0.	0.	0.	0.
2.375	24	0.	0.	0.	0.	0.	0.	0.
2.500	25	0.	0.	0.	0.	0.	0.	0.
2.750	26	0.	0.	0.	0.	0.	0.	0.
3.000	27	0.	0.	0.	0.	0.	0.	0.

PAGE 2

Y		X .6100	.6465	.6820	.8190	.9560	1.084	1.175
		8	9	10	11	12	13	14
0.	1	0.	0.	0.	0.	0.	0.	0.
.0500	2	0.	0.	0.	0.	0.	0.	0.
.1000	3	0.	0.	0.	0.	0.	0.	0.
.2500	4	0.	0.	0.	0.	0.	0.	0.
.4000	5	0.	0.	0.	0.	0.	0.	0.
.5500	6	0.	0.	0.	0.	0.	0.	0.
.7000	7	0.	0.	0.	0.	0.	0.	0.
.8500	8	0.	0.	0.	0.	0.	0.	0.
1.000	9	0.	0.	0.	0.	0.	0.	0.
1.070	10	0.	0.	0.	0.	0.	0.	0.
1.140	11	0.	0.	0.	0.	0.	0.	0.
1.220	12	0.	0.	0.	0.	0.	0.	0.
1.320	13	0.	0.	0.	0.	0.	0.	0.
1.420	14	0.	0.	0.	0.	0.	0.	0.
1.520	15	0.	0.	0.	0.	0.	0.	0.
1.604	16	0.	0.	0.	0.	0.	0.	0.
1.673	17	0.	0.	0.	0.	0.	0.	0.
1.750	18	0.	0.	0.	0.	0.	0.	0.
1.833	19	0.	0.	0.	0.	0.	0.	0.
1.917	20	0.	0.	0.	0.	0.	0.	0.
2.000	21	0.	0.	0.	0.	0.	0.	0.
2.125	22	0.	0.	0.	0.	0.	0.	0.
2.250	23	0.	0.	0.	0.	0.	0.	0.

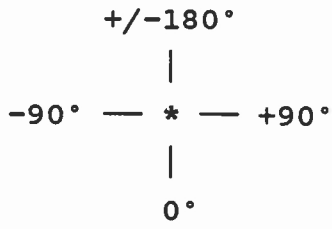
```
2.375 24  0.        0.        0.        0.        0.        0.        0.
2.500 25  0.        0.        0.        0.        0.        0.        0.
2.750 26  0.        0.        0.        0.        0.        0.        0.
3.000 27  0.        0.        0.        0.        0.        0.        0.

    PAGE  3
       X   1.220
  Y           15
0.      1  0.
.0500   2  0.
.1000   3  0.
.2500   4  0.
.4000   5  0.
.5500   6  0.
.7000   7  0.
.8500   8  0.
1.000   9  0.
1.070  10  0.
1.140  11  0.
1.220  12  0.
1.320  13  0.
1.420  14  0.
1.520  15  0.
1.604  16  0.
1.673  17  0.
1.750  18  0.
1.833  19  0.
1.917  20  0.
2.000  21  0.
2.125  22  0.
2.250  23  0.
2.375  24  0.
2.500  25  0.
2.750  26  0.
3.000  27  0.

        RETARDATION AND OVER ALL FIRST ORDER LOSS PROCESS COEFFICIENT

    RETARD=  0.00000E+00

    LAMDA=   0.00000E+00 (1/DAY)
```

A listing of part of the HYDROOUT.OUT file produced when the program is run with the example data files as input now follows. The first part of the output is the pressure field. The second part shows the Darcy velocity field, in polar 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,-16.6° .09058, .000° .05954, 19.5°

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.



EXAMPLE WATLT2.DAT DATA FILE FOR MODEL LTRSKAQ2. 8/16/1989
TEST RUN NUMBER 1 - SINGLE INJECTION 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

Y		X 0.	.0453	.1360	.2730	.4100	.5470	.5780
		1	2	3	4	5	6	7
0.	1	.14800	.14800	.14800	.14800	.14800	.14800	.14800
.0500	2	.14717	.14717	.14717	.14717	.14718	.14718	.14718
.1000	3	.14634	.14634	.14635	.14635	.14635	.14635	.14635
.2500	4	.14386	.14386	.14386	.14387	.14388	.14388	.14388
.4000	5	.14137	.14137	.14137	.14139	.14141	.14142	.14142
.5500	6	.13886	.13886	.13887	.13890	.13895	.13898	.13898
.7000	7	.13632	.13632	.13634	.13641	.13650	.13658	.13658
.8500	8	.13373	.13373	.13377	.13388	.13406	.13427	.13429
1.000	9	.13104	.13105	.13110	.13127	.13160	.13218	.13231
1.070	10	.12975	.12975	.12980	.12998	.13035	.13122	.13178
1.140	11	.12842	.12843	.12848	.12864	.12897	.12957	.12972
1.220	12	.12688	.12688	.12692	.12706	.12731	.12765	.12769
1.320	13	.12490	.12490	.12493	.12504	.12520	.12536	.12538
1.420	14	.12289	.12289	.12291	.12298	.12308	.12317	.12317
1.520	15	.12084	.12085	.12086	.12091	.12097	.12102	.12102
1.604	16	.11912	.11912	.11913	.11916	.11920	.11923	.11923
1.673	17	.11768	.11768	.11769	.11772	.11774	.11776	.11777
1.750	18	.11609	.11609	.11610	.11612	.11613	.11615	.11615
1.833	19	.11437	.11437	.11437	.11438	.11440	.11440	.11440
1.917	20	.11262	.11262	.11262	.11263	.11264	.11264	.11264

2.000	21	.11089	.11089	.11089	.11090	.11090	.11090	.11090
2.125	22	.10828	.10828	.10828	.10828	.10829	.10829	.10829
2.250	23	.10567	.10567	.10567	.10567	.10567	.10567	.10567
2.375	24	.10306	.10306	.10306	.10306	.10306	.10306	.10306
2.500	25	.10045	.10045	.10045	.10045	.10045	.10045	.10045
2.750	26	.09522	.09522	.09522	.09522	.09522	.09522	.09522
3.000	27	.09000	.09000	.09000	.09000	.09000	.09000	.09000

PAGE 2

		X	.6100	.6465	.6820	.8190	.9560	1.084	1.175
Y			8	9	10	11	12	13	14
0.	1		.14800	.14800	.14800	.14800	.14800	.14800	.14800
.0500	2		.14718	.14718	.14718	.14718	.14717	.14717	.14717
.1000	3		.14635	.14635	.14635	.14635	.14635	.14635	.14634
.2500	4		.14389	.14388	.14388	.14388	.14387	.14386	.14386
...	...		...	...	...	...	...	...	...
2.500	25		.10045	.10045	.10045	.10045	.10045	.10045	.10045
2.750	26		.09522	.09522	.09522	.09522	.09522	.09522	.09522
3.000	27		.09000	.09000	.09000	.09000	.09000	.09000	.09000

PAGE 3

		X	1.220
Y			15
0.	1		.14800
.0500	2		.14717
.1000	3		.14634
.2500	4		.14386
...	...		...
2.500	25		.10045
2.750	26		.09522
3.000	27		.09000

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

PAGE 1

		X	0.	.0453	.1360	.2730
Y			1	2	3	4
0.	1		.0, .000°	.09298, .000°	.09293, .000°	.09279, .000°
.0500	2		.0, .000°	.09299,-.008°	.09294,-.023°	.09279,-.036°
.1000	3		.0, .000°	.09301,-.017°	.09296,-.047°	.09279,-.074°
.2500	4		.0, .000°	.09318,-.048°	.09310,-.136°	.09284,-.214°
...	...		...	...	...	...
2.500	25		.0, .000°	.11733,-.002°	.11734,-.005°	.11735,-.008°
2.750	26		.0, .000°	.11735,-.001°	.11735,-.001°	.11735,-.002°
3.000	27		.0, .000°	.11735, .000°	.11735, .000°	.11735, .000°

PAGE 2

		X	.4100	.5470	.5780	.6100
Y			5	6	7	8
0.	1		.09262, .000°	.09250, .000°	.09249, .000°	.09249, .000°
.0500	2		.09261,-.032°	.09249,-.013°	.09248,-.006°	.09248, .000°
.1000	3		.09260,-.066°	.09247,-.026°	.09246,-.013°	.09246, .000°
.2500	4		.09251,-.196°	.09228,-.077°	.09226,-.040°	.09225,-.000°
.4000	5		.09231,-.444°	.09181,-.181°	.09176,-.094°	.09175,-.000°
.5500	6		.09195,-.977°	.09073,-.420°	.09062,-.218°	.09058, .000°
.7000	7		.09150,-2.18°	.08821,-1.05°	.08785,-.551°	.08772, .008°
.8500	8		.09202,-4.98°	.08245,-3.27°	.08011,-1.90°	.07862, .064°
1.000	9		.09945,-10.8°	.08098,-16.6°	.05585,-19.4°	.00738,-2.17°
1.070	10		.10841,-13.5°	.13765,-40.4°	.16527,-51.1°	.10392,-7.52°
1.140	11		.11493,-9.48°	.13611,-11.3°	.15619,-8.29°	.19115,-.166°
1.220	12		.11814,-5.81°	.13217,-3.75°	.13688,-2.22°	.14033, .069°
1.320	13		.11907,-3.22°	.12586,-1.61°	.12678,-.849°	.12717, .029°
1.420	14		.11884,-1.84°	.12211,-.831°	.12242,-.430°	.12253, .006°
1.520	15		.11841,-1.08°	.12006,-.461°	.12021,-.238°	.12026, .001°
1.604	16		.11811,-.694°	.11907,-.287°	.11915,-.148°	.11918,-.000°

1.673	17	.11791,-.482°	.11855,-.195°	.11860,-.101°	.11862,-.001°
1.750	18	.11774,-.324°	.11815,-.129°	.11819,-.067°	.11820,-.000°
1.833	19	.11762,-.212°	.11788,-.083°	.11790,-.043°	.11791,-.000°
1.917	20	.11753,-.138°	.11770,-.054°	.11771,-.028°	.11772,-.000°
2.000	21	.11747,-.090°	.11758,-.035°	.11759,-.018°	.11759,-.000°
2.125	22	.11742,-.048°	.11748,-.019°	.11748,-.010°	.11748, .000°
2.250	23	.11739,-.026°	.11742,-.010°	.11742,-.005°	.11742, .000°
2.375	24	.11737,-.014°	.11739,-.005°	.11739,-.003°	.11739, .000°
2.500	25	.11737,-.007°	.11737,-.003°	.11738,-.001°	.11738, .000°
2.750	26	.11736,-.002°	.11736,-.001°	.11736,-.000°	.11736, .000°
3.000	27	.11736, .000°	.11736, .000°	.11736, .000°	.11736, .000°

PAGE 3

	X	.6465	.6820	.8190	.9560
Y		9	10	11	12
0.	1	.09249, .000°	.09251, .000°	.09263, .000°	.09280, .000°
.0500	2	.09249, .007°	.09250, .014°	.09262, .033°	.09280, .035°
.1000	3	.09246, .015°	.09248, .029°	.09261, .068°	.09281, .073°
.2500	4	.09226, .045°	.09229, .088°	.09253, .201°	.09286, .212°
.4000	5	.09177, .106°	.09183, .204°	.09235, .455°	.09299, .466°
.5500	6	.09063, .247°	.09078, .474°	.09204, .996°	.09334, .960°
.7000	7	.08792, .639°	.08839, 1.19°	.09171, 2.21°	.09425, 1.90°
.8500	8	.08049, 2.25°	.08331, 3.66°	.09245, 4.95°	.09660, 3.51°
1.000	9	.05954, 19.5°	.08425, 16.6°	.09981, 10.4°	.10208, 5.54°
1.070	10	.15874, 49.1°	.13150, 37.1°	.10803, 12.7°	.10576, 5.78°
1.140	11	.15347, 8.87°	.13334, 11.8°	.11427, 9.17°	.10925, 5.10°
1.220	12	.13621, 2.63°	.13075, 4.24°	.11757, 5.75°	.11226, 3.94°
1.320	13	.12660, 1.02°	.12539, 1.83°	.11870, 3.24°	.11461, 2.63°
1.420	14	.12236, .500°	.12195, .942°	.11863, 1.87°	.11589, 1.69°
1.520	15	.12019, .272°	.12000, .521°	.11830, 1.10°	.11657, 1.06°
1.604	16	.11914, .168°	.11904, .324°	.11803, .709°	.11688, .713°
1.673	17	.11859, .114°	.11852, .221°	.11786, .494°	.11704, .509°
1.750	18	.11819, .075°	.11814, .146°	.11771, .332°	.11715, .349°
1.833	19	.11790, .049°	.11787, .094°	.11759, .217°	.11722, .232°
1.917	20	.11771, .031°	.11769, .061°	.11752, .142°	.11727, .153°
2.000	21	.11759, .020°	.11758, .039°	.11746, .093°	.11730, .101°
2.125	22	.11748, .011°	.11747, .021°	.11741, .050°	.11733, .054°
2.250	23	.11742, .006°	.11742, .011°	.11739, .027°	.11734, .029°
2.375	24	.11739, .003°	.11739, .006°	.11737, .014°	.11735, .016°
2.500	25	.11738, .002°	.11737, .003°	.11736, .007°	.11735, .008°
2.750	26	.11736, .000°	.11736, .001°	.11736, .002°	.11735, .002°
3.000	27	.11736, .000°	.11736, .000°	.11736, .000°	.11735, .000°

PAGE 4

	X	1.084	1.175	1.220		
Y		13	14	15		
0.	1	.09293, .000°	.09298, .000°	.0, .000°		
.0500	2	.09294, .023°	.09299, .008°	.0, .000°		
.1000	3	.09296, .047°	.09301, .017°	.0, .000°		
.2500	4	.09310, .136°	.09318, .048°	.0, .000°		
...	...	...	...	...	...	...
2.500	25	.11734, .005°	.11733, .002°	.0, .000°		
2.750	26	.11735, .001°	.11735, .001°	.0, .000°		
3.000	27	.11735, .000°	.11735, .000°	.0, .000°		

A listing of part of the CHEMFOUT.OUT file produced when the program is run with the example data files as input now follows. Most of the output is devoted to showing the chemical concentration at the nodal points of the aquifer, at each of the

print times listed near the beginning of the CHMLT2.OUT file. In two-dimensional arrays, as printed in this file and the files mentioned previously, the numbers are printed in a variable format. Here are some examples:

0. .00 5.78E-07 .00229 .16690 1.2160

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 WATLT2.DAT data file, are printed as ".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 zero's show where the value is very small.

Other output in the file includes messages telling when the wells are switched on or off, and messages to indicate whether they are currently on or off. There are also cumulative masses.

```
      EXAMPLE CHMLT2.DAT DATA FILE.  8/2/1989
      TEST RUN NUMBER 1 - SINGLE INJECTION WELL

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

        OUTPUT DATA FOR CHEMICAL SYSTEM

   INJECTION WELLS SWITCHED ON AT TIME T=     .0000 (DAYS)

   TIME T=     .0900 (DAYS)
   CUMULATIVE CHEMICAL MASSES (KG)

   XMASS=   1.35937E-04  XMFONW=  0.00000E+00  XMSOUR=  1.62144E-04
   XMASIN= -3.77404E-34  XMASOT=  0.00000E+00

   INJECTION WELLS ARE CURRENTLY ON

        CHEMICAL CONCENTRATION DISTRIBUTION (KG/M^3)

   PAGE  1
      X   0.        .0453     .1360     .2730     .4100     .5470     .5780
 Y            1         2         3         4         5         6         7
0.     1  0.        0.        0.        0.        0.        0.        0.
.0500  2  0.        0.        0.        0.        0.        0.        0.
```

.1000	3	0.	0.	0.	0.	0.	0.	.00
.2500	4	0.	0.	0.	0.	0.	.00	.00
...	...	...	...	...	...	...	...	...
2.500	25	0.	0.	0.	0.	0.	0.	0.
2.750	26	0.	0.	0.	0.	0.	0.	0.
3.000	27	0.	0.	0.	0.	0.	0.	0.

PAGE 2

		X	.6100	.6465	.6820	.8190	.9560	1.084	1.175
Y			8	9	10	11	12	13	14
0.	1	0.	0.	0.	0.	0.	0.	0.	0.
.0500	2	.00	0.	0.	0.	0.	0.	0.	0.
.1000	3	.00	.00	0.	0.	0.	0.	0.	0.
.2500	4	.00	.00	.00	0.	0.	0.	0.	0.
.4000	5	.00	.00	.00	.00	0.	0.	0.	0.
.5500	6	.00	.00	.00	.00	.00	0.	0.	0.
.7000	7	.00	.00	.00	.00	.00	.00	0.	0.
.8500	8	5.78E-07	2.39E-07	2.55E-08	.00	.00	.00	.00	.00
1.000	9	8.77E-04	3.57E-04	4.79E-05	4.49E-07	1.32E-09	.00	.00	.00
1.070	10	.16690	.06303	.01166	1.19E-04	3.88E-07	.00	.00	.00
1.140	11	.06108	.01783	.00229	2.32E-05	7.85E-08	.00	.00	.00
1.220	12	.00932	.00200	1.83E-04	1.66E-06	5.48E-09	.00	.00	.00
1.320	13	6.28E-04	1.02E-04	6.79E-06	5.18E-08	.00	.00	.00	.00
1.420	14	2.60E-05	3.21E-06	1.56E-07	.00	.00	.00	.00	.00
1.520	15	6.86E-07	6.35E-08	2.23E-09	.00	.00	.00	.00	.00
1.604	16	1.36E-08	.00	.00	.00	.00	.00	.00	.00
1.673	17	.00	.00	.00	.00	.00	.00	.00	.00
1.750	18	.00	.00	.00	.00	.00	.00	.00	.00
1.833	19	0.	0.	0.	0.	0.	0.	0.	0.
1.917	20	0.	0.	0.	0.	0.	0.	0.	0.
2.000	21	0.	0.	0.	0.	0.	0.	0.	0.
2.125	22	0.	0.	0.	0.	0.	0.	0.	0.
2.250	23	0.	0.	0.	0.	0.	0.	0.	0.
2.375	24	0.	0.	0.	0.	0.	0.	0.	0.
2.500	25	0.	0.	0.	0.	0.	0.	0.	0.
2.750	26	0.	0.	0.	0.	0.	0.	0.	0.
3.000	27	0.	0.	0.	0.	0.	0.	0.	0.

PAGE 3

		X	1.220
Y			15
0.	1	0.	
.0500	2	0.	
.1000	3	0.	
.2500	4	0.	
.4000	5	0.	
...	...	...	...
2.500	25	0.	
2.750	26	0.	
3.000	27	0.	

TIME T= .1700 (DAYS)
CUMULATIVE CHEMICAL MASSES (KG)

XMASS= 2.66106E-04 XMFONW= 0.00000E+00 XMSOUR= 3.06272E-04
XMASIN= -2.28395E-29 XMASOT= 4.14765E-32

INJECTION WELLS ARE CURRENTLY ON

CHEMICAL CONCENTRATION DISTRIBUTION (KG/M^3)

PAGE 1

		X	0.	.0453	.1360	.2730	.4100	.5470	.5780
Y			1	2	3	4	5	6	7
0.	1	.00	.00	.00	.00	.00	.00	.00	
.0500	2	.00	.00	.00	.00	.00	.00	.00	

TIME T= .5000 (DAYS)
CUMULATIVE CHEMICAL MASSES (KG)

XMASS= 8.04108E-04 XMFONW= 0.00000E+00 XMSOUR= 9.00800E-04
XMASIN= -5.42525E-24 XMASOT= 5.13649E-19

INJECTION WELLS ARE CURRENTLY ON

CHEMICAL CONCENTRATION DISTRIBUTION (KG/M^3)

PAGE 1

	X	0.	.0453	.1360	.2730	.4100	.5470	.5780
Y		1	2	3	4	5	6	7
0.	1	.00	.00	.00	.00	.00	.00	.00
.0500	2	.00	.00	.00	.00	.00	.00	.00
.1000	3	.00	.00	.00	.00	.00	.00	.00
.2500	4	.00	.00	.00	.00	.00	.00	.00
.4000	5	.00	.00	.00	.00	.00	.00	.00
.5500	6	.00	.00	.00	.00	.00	.00	.00
.7000	7	.00	.00	.00	.00	1.03E-08	1.10E-07	1.58E-07
.8500	8	.00	.00	1.70E-09	9.62E-08	2.44E-06	2.40E-05	3.24E-05
1.000	9	-2.67E-08	4.14E-09	2.51E-07	1.47E-05	3.78E-04	.00383	.00572
1.070	10	-4.18E-07	6.28E-08	3.91E-06	2.43E-04	.00695	.07886	.12163
1.140	11	-6.08E-07	9.64E-08	5.74E-06	3.32E-04	.00851	.08282	.12160
1.220	12	-4.16E-07	6.92E-08	3.96E-06	2.21E-04	.00557	.05696	.08851
1.320	13	-1.53E-07	2.66E-08	1.46E-06	8.14E-05	.00215	.02498	.04271
1.420	14	-4.01E-08	7.24E-09	3.86E-07	2.18E-05	6.18E-04	.00822	.01553
1.520	15	-8.26E-09	1.53E-09	8.00E-08	4.63E-06	1.42E-04	.00215	.00447
1.604	16	-1.65E-09	.00	1.60E-08	9.51E-07	3.14E-05	5.35E-04	.00122
1.673	17	.00	.00	3.23E-09	1.98E-07	7.03E-06	1.33E-04	3.29E-04
1.750	18	.00	.00	.00	3.35E-08	1.27E-06	2.66E-05	7.15E-05
1.833	19	.00	.00	.00	4.67E-09	1.90E-07	4.38E-06	1.27E-05
1.917	20	.00	.00	.00	.00	2.50E-08	6.33E-07	1.99E-06
2.000	21	.00	.00	.00	.00	2.97E-09	8.20E-08	2.77E-07
2.125	22	.00	.00	.00	.00	.00	6.54E-09	2.38E-08
2.250	23	.00	.00	.00	.00	.00	.00	1.83E-09
2.375	24	.00	.00	.00	.00	.00	.00	.00
2.500	25	.00	.00	.00	.00	.00	.00	.00
2.750	26	.00	.00	.00	.00	.00	.00	.00
3.000	27	.00	.00	.00	.00	.00	.00	.00

PAGE 2

	X	.6100	.6465	.6820	.8190	.9560	1.084	1.175
Y		8	9	10	11	12	13	14
0.	1	.00	.00	.00	.00	.00	.00	.00
.0500	2	.00	.00	.00	.00	.00	.00	.00
.1000	3	.00	.00	.00	.00	.00	.00	.00
.2500	4	.00	.00	.00	.00	.00	.00	.00
.4000	5	.00	.00	.00	.00	.00	.00	.00
.5500	6	.00	.00	.00	.00	.00	.00	.00
.7000	7	1.43E-07	1.50E-07	8.98E-08	7.70E-09	.00	.00	.00
.8500	8	2.71E-05	3.07E-05	1.99E-05	1.85E-06	6.85E-08	1.28E-09	.00
1.000	9	.00616	.00531	.00318	2.88E-04	1.05E-05	1.90E-07	3.06E-09
1.070	10	.17461	.11386	.06628	.00534	1.74E-04	2.97E-06	4.66E-08
1.140	11	.15255	.11505	.07038	.00657	2.39E-04	4.36E-06	7.16E-08
1.220	12	.11081	.08358	.04771	.00428	1.59E-04	3.01E-06	5.14E-08
1.320	13	.05506	.04010	.02051	.00163	5.84E-05	1.11E-06	1.97E-08
1.420	14	.02078	.01450	.00663	4.65E-04	1.56E-05	2.92E-07	5.36E-09
1.520	15	.00621	.00415	.00171	1.06E-04	3.29E-06	6.04E-08	1.13E-09
1.604	16	.00175	.00112	4.20E-04	2.32E-05	6.74E-07	1.20E-08	.00
1.673	17	4.89E-04	3.03E-04	1.04E-04	5.15E-06	1.40E-07	2.43E-09	.00
1.750	18	1.10E-04	6.54E-05	2.05E-05	9.26E-07	2.36E-08	.00	.00
1.833	19	2.02E-05	1.16E-05	3.36E-06	1.38E-07	3.28E-09	.00	.00
1.917	20	3.25E-06	1.80E-06	4.82E-07	1.80E-08	.00	.00	.00

2.000	21	4.68E-07	2.51E-07	6.21E-08	2.13E-09	.00	.00	.00
2.125	22	4.15E-08	2.14E-08	4.92E-09	.00	.00	.00	.00
2.250	23	3.29E-09	1.64E-09	.00	.00	.00	.00	.00
2.375	24	.00	.00	.00	.00	.00	.00	.00
2.500	25	.00	.00	.00	.00	.00	.00	.00
2.750	26	.00	.00	.00	.00	.00	.00	.00
3.000	27	.00	.00	.00	.00	.00	.00	.00

PAGE 3

X		1.220
Y		15
0.	1	.00
.0500	2	.00
.1000	3	.00
.2500	4	.00
.4000	5	.00
.5500	6	.00
.7000	7	.00
.8500	8	.00
1.000	9	-2.02E-08
1.070	10	-3.18E-07
1.140	11	-4.64E-07
1.220	12	-3.17E-07
1.320	13	-1.16E-07
1.420	14	-3.05E-08
1.520	15	-6.27E-09
1.604	16	-1.25E-09
1.673	17	.00
1.750	18	.00
1.833	19	.00
1.917	20	.00
2.000	21	.00
2.125	22	.00
2.250	23	.00
2.375	24	.00
2.500	25	.00
2.750	26	.00
3.000	27	.00

VI. Description of the program

VI.1. Organization

The program is written in Fortran 77 with extensions (see section VI.2); the "include" feature of Microsoft Fortran (version 4.01) is used for Common Blocks and for global parameters. There are 7 "include" files, all with the extension ".LT2". There are 12 Fortran source files, all with the extension ".FOR". Each of the Fortran source files is called a "module". Some of them contain only one subroutine; others contain 2 or more subroutines. Below is a list of the "include" files and modules, showing what is contained in them.

Include File	Contents
CSIZE.LT2	Global Parameters (Symbolic Constants)
CADII.LT2	Common Blocks ADII1, ADII2, ADII3, ADII4, ADII5
CCHEM.LT2	Common Blocks CHEM1, CHEM2, CHEM3, CHEM4, CHEM5, CHEM6, CHEM10, CHEM11, CHEM12
CINDX.LT2	Common Blocks INDEX1, INDEX2, INDEX3
CMISC.LT2	Common Blocks CHARS, DENSI2, GEOM1, GEOM2, GEOM3, MASSES, PRESS1, RUNCT1, RUNCT2, RUNCT4, STATV1, STATV2, SOIL10, WELLS1, WELLS2
CNRK.LT2	Common Block NRK
CTHOS.LT2	Common Blocks THOS1, THOS2, VELCT1

Module	Contents
LTRSKAQ2.FOR	Main Program LTRSK2
ADILT2.FOR	Subroutine ADILT2
CHMLT2.FOR	Subroutine CHEM
ITGRLT2.FOR	Subroutines INTGRL, COMSRC
NEWTWO.FOR	Subroutine NEWTWO

OUTLT2.FOR	Subroutines PRINT1, PRINT2, PRINT3, PRINT4, ARYSTR, ARYPOL, NUMSTR, NUMFIX, POLAR
PSTARLT2.FOR	Subroutines PSTAR, FEVAL; Function CLAG
RWCLT2.FOR	Subroutines CHMREAD, CHMOUT, CHMWRT
RWWLT2.FOR	Subroutines FLOREAD, FLOOUT, FLOWRT, TESTIJ
TCMLT2.FOR	Subroutine TCMLT2
THOMLT2.FOR	Subroutine THOMLT2
WATLT2.FOR	Subroutine FLUID

VI.2. Non-standard features

The program is written in Standard Fortran 77, except for the following:

* It uses Microsoft Fortran's "include" feature. The "include" directives can be changed to the form used by another version of Fortran, or the "include" files can be substituted in place of the directives.

* Lower case letters are used in comments and in character strings. They can be changed to upper case.

* Some subroutine names are more than 6 characters long. They can be truncated.

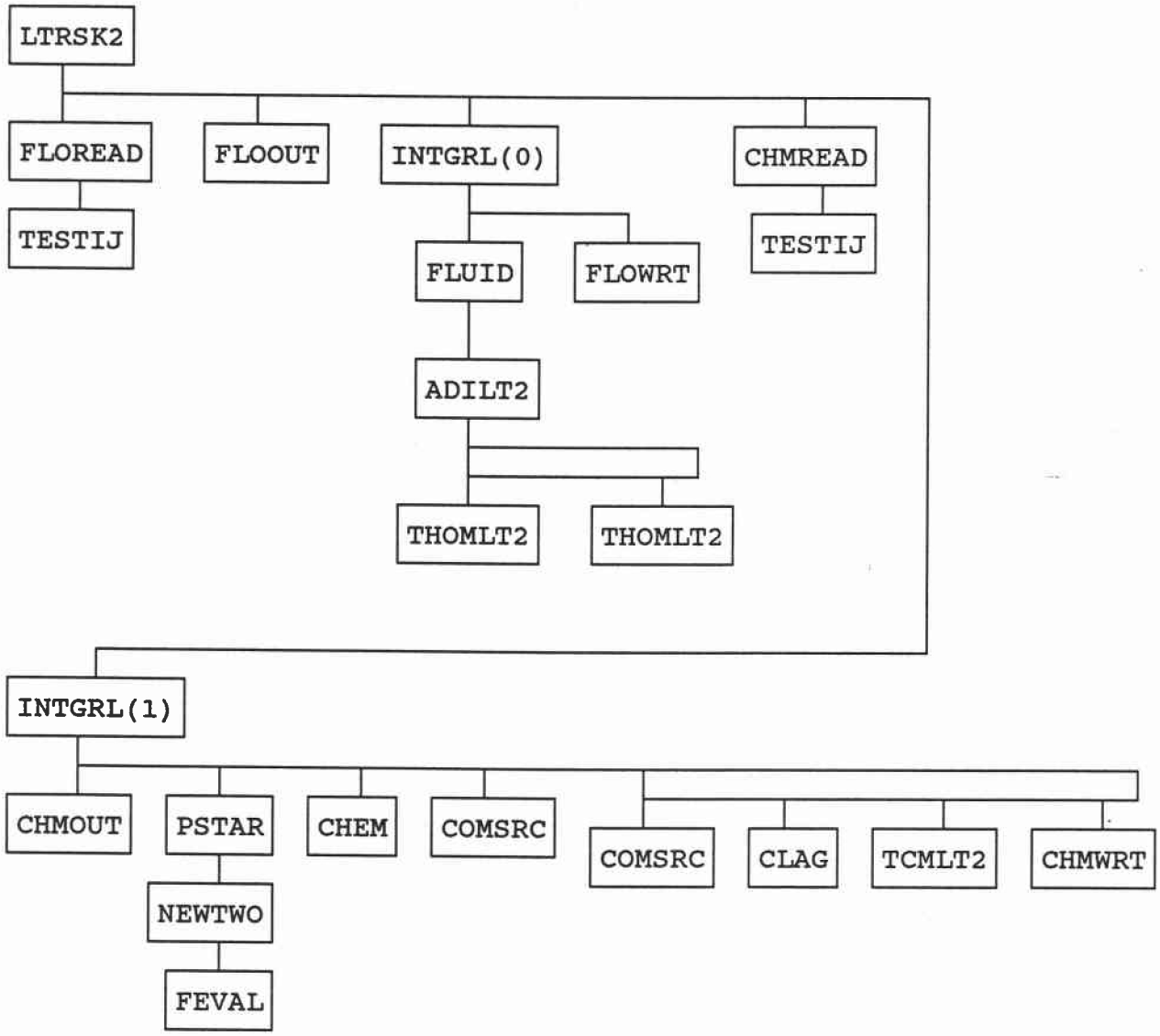
* Some Format statements use the "$" edit descriptor which means: no carriage-return-line-feed at end of line. These can be removed if necessary.

* Some debugging code in NEWTWO has "D" in column 1; these can be changed to "C" or the lines can be deleted.

VI.3. General flow diagram of LTRSKAQ2 program

LTRSK2 is the main program. The other boxes represent subroutines and functions. The flow is: first downward, then to the right. Thus, LTRSK2 calls FLOREAD, then FLOOUT, then INTGRL

with argument 0. INTGRL calls FLUID, which calls ADILT2, which calls THOMLT2 in a loop. This call is made twice during each execution of the loop. When the loop terminates, INTGRL calls FLOWRT. Then LTRSK2 calls CHMREAD, etc. There is one other major loop, in the second call of INTGRL. The first loop computes the steady-state hydraulic pressure field; the second loop computes the time-varying chemical concentrations. Calls to subroutines in the module OUTLT2.FOR are not shown.



REFERENCES

Lindstrom, F., L. Boersma, M. A. Barlaz, and F. Beck. 1989. Transport and fate of water and chemicals in laboratory scale, single layer, aquifers. Volume 1. Mathematical Model. Special Report 845, Oregon Agricultural Experiment Station, Corvallis, Oregon. 52pp.

Varga, R. S. 1962. Matrix Iterative Analysis. Prentice-Hall, Englewood CLiffs, NJ. 322 pp.

Appendix. Listing of Fortran program

The source code for LTRSKAQ2 is on the installation diskette, in the subdirectory SOURCE. There is a READ.ME file in that subdirectory that tells how to install the source code.

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

```
* Include file: CSIZE.LT2              Last revision: August 9, 1989
*                                      For LTRSKAQ2, etc.
      INTEGER IX,IY,MAXPRT,MAXSWT,NDIM
      PARAMETER(IX=31,IY=35,MAXPRT=100,MAXSWT=100,NDIM=2)
```

```
* Include file: CADII.LT2              Last revision: December 6, 1988
*                                      For LTRSKAQ2, etc.
C
      DOUBLE PRECISION ALT1,ALT2,ALT3
      DOUBLE PRECISION ADT1,ADT2X,ADT2Y
      DOUBLE PRECISION ADT3,ADT2
      DOUBLE PRECISION AUT1,AUT2,AUT3
      DOUBLE PRECISION XDUM,YDUM
C
      COMMON/ADII1 / ALT1(IX,IY),ALT2(IX,IY),ALT3(IX,IY)
      COMMON/ADII2 / ADT1(IX,IY),ADT2X(IX,IY),ADT2Y(IX,IY)
      COMMON/ADII3 / ADT3(IX,IY),ADT2(IX,IY)
      COMMON/ADII4 / AUT1(IX,IY),AUT2(IX,IY),AUT3(IX,IY)
      COMMON/ADII5 / XDUM(IX,IY),YDUM(IX,IY)
```

```
* Include file: CCHEM.LT2              Last revision: January 31, 1989
*                                      For LTRSKAQ2, etc.
C
      DOUBLE PRECISION DL0
      DOUBLE PRECISION DISPLX,DISPLY
      DOUBLE PRECISION KSAND,KCLAY,KORG
      DOUBLE PRECISION QCHM1S,CSWELN,CHMCON
      DOUBLE PRECISION DCHLX,DCHLY,DCHLXY
      DOUBLE PRECISION XLAM10,XLAMIR,XLAMRA,XSLM10,XSLMRA
      DOUBLE PRECISION DTRET,RETARD,LAMDA
      DOUBLE PRECISION CIN,COUT,C0
      DOUBLE PRECISION CDENOM,SOURC
C
```

```
      COMMON/CHEM1 / DL0
      COMMON/CHEM2 / DISPLX,DISPLY
      COMMON/CHEM3 / KSAND,KCLAY,KORG
      COMMON/CHEM4 / QCHM1S(IX,IY),CSWELN(IX,IY),CHMCON(IX*IY)
      COMMON/CHEM5 / DCHLX(IX,IY),DCHLY(IX,IY),DCHLXY(IX,IY)
      COMMON/CHEM6 / XLAM10,XLAMIR,XLAMRA,XSLM10,XSLMRA
      COMMON/CHEM10/ DTRET,RETARD,LAMDA
      COMMON/CHEM11/ CIN,COUT,C0
      COMMON/CHEM12/ CDENOM(IX,IY),SOURC(IX,IY)
```

```
* Include file: CINDX.LT2          Last revision: January 31, 1989
*                                   For LTRSKAQ2, etc.
C
      INTEGER NSLXM1,NSLYM1,NSLXXX,NSLYYY,NSLXP1,NSLYP1
      INTEGER NFLAG, NFUNC
      INTEGER INJI, INJJ
C
      COMMON/INDEX1/ NSLXM1,NSLYM1,NSLXXX,NSLYYY,NSLXP1,NSLYP1
      COMMON/INDEX2/ NFLAG(20), NFUNC(IX,IY)
      COMMON/INDEX3/ INJI(IX*IY), INJJ(IX*IY)
```

```
* Include file: CMISC.LT2          Last revision: July 19, 1989
*                                   For LTRSKAQ2, etc.
C
      CHARACTER*80      HEAD1, HEAD2
      DOUBLE PRECISION RHOSND,RHOORG,RHOCLA,RHOWAT
      DOUBLE PRECISION DX,DY,XLYIN,XLYOUT,XLW
      DOUBLE PRECISION XNODE,YNODE
      DOUBLE PRECISION PSTARX,PSTARY
      DOUBLE PRECISION XMASS,XMFONW,XMSOUR,XMASIN,XMASOT
      DOUBLE PRECISION HIN,HOUT,PHNEWX,PHNEWY
      INTEGER           NPRT,NSWT
      DOUBLE PRECISION T0,TMAX,DT0,DT1,PRTIME,SWTIME
      INTEGER           NLSOR
      DOUBLE PRECISION ADIPRM,TLRNWA
      INTEGER           NMOD
      DOUBLE PRECISION DELTA,ADIMAX,ZTHRSH
      DOUBLE PRECISION HOLD,COLD
      DOUBLE PRECISION HNEW,CNEW
      DOUBLE PRECISION KTHSTS,TORT,EPS,PCTSAN,PCTCLA,PCTORG
      DOUBLE PRECISION QWELIN,QWELOT
      INTEGER           NINJW, NEXTW
      LOGICAL           WELLON
```

```
C
      COMMON/CHARS/  HEAD1, HEAD2
      COMMON/DENSI2/ RHOSND,RHOORG,RHOCLA,RHOWAT
      COMMON/GEOM1 / DX(IX),DY(IY),XLYIN,XLYOUT,XLW
      COMMON/GEOM2 / XNODE(IX),YNODE(IY)
      COMMON/GEOM3 / PSTARX(IX,IY),PSTARY(IX,IY)
      COMMON/MASSES/ XMASS,XMFONW,XMSOUR,XMASIN,XMASOT
      COMMON/PRESS1/ HIN,HOUT,PHNEWX(IX,IY),PHNEWY(IX,IY)
      COMMON/RUNCT1/ T0,TMAX,DT0,DT1,NPRT,NSWT,PRTIME(MAXPRT),
     &               SWTIME(MAXSWT+1)
      COMMON/RUNCT2/ ADIPRM,NLSOR,TLRNWA
      COMMON/RUNCT4/ DELTA,NMOD,ADIMAX,ZTHRSH
      COMMON/STATV1/ HOLD(IX,IY),COLD(IX,IY)
      COMMON/STATV2/ HNEW(IX,IY),CNEW(IX,IY)
      COMMON/SOIL10/ KTHSTS,TORT,EPS,PCTSAN,PCTCLA,PCTORG
      COMMON/WELLS1/ QWELIN(IX,IY),QWELOT(IX,IY)
      COMMON/WELLS2/ NINJW, NEXTW, WELLON
```

```
* Include file: CNRK.LT2              Last revision: December 13, 1988
*                                     For LTRSKAQ2, etc.
C
      DOUBLE PRECISION        DT, XI, YJ, COEF
      COMMON /NRK/            DT, XI, YJ, COEF(8)
```

```
* Include file: CTHOS.LT2             Last revision: July 19, 1989
*                                     For LTRSKAQ2, etc.
C
      DOUBLE PRECISION ATDT,BTDT,CTDT
      DOUBLE PRECISION XTDUM,YTDUM
      DOUBLE PRECISION VLXX,VLYY
      LOGICAL PSERR
C
      COMMON/THOS1 / ATDT(IX*IY),BTDT(IX*IY),CTDT(IX*IY)
      COMMON/THOS2 / XTDUM(IX*IY),YTDUM(IX*IY)
      COMMON/VELCT1/ VLXX(IX,IY),VLYY(IX,IY),PSERR
```

M O D U L E S

```
*         File: LTRSKAQ2.FOR          Last revision: July 19, 1989
C
      PROGRAM LTRSK2
C
C  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  LTRSK2 = Two Dimensional Water and Chemical Transport in the
C         long thin RSKERL physical aquifer. (VERSION 2.1)
C
C  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C   Homogeneous and isotropic confined aquifer - constant
C   saturated water conductivity function - no flow boundaries
C   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 (HORIZONTAL)
C  TRANSPORT AND FATE OF LOW WATER SOLUBILITY CHEMICALS IN AN
C  AQUIFER.
C  THE PROGRAM IS MODULAR IN DESIGN.  COMMON BLOCKS ARE DEFINED IN
```

```
C  INCLUDE FILES WHOSE NAMES HAVE THE EXTENSION "LT2".  THE MAJOR
C  SECTIONS OF THE PROGRAM ARE:
C     1)  PROGRAM LTRSK2.  THE CALLING PROGRAM.
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 HEAD AND VELOCITY COMPONENTS
C  IF SO CALLED FOR.
C     5)  INTGRL.  ORGANIZES AND SOLVES STEADY STATE FLOW SYSTEM;
C  TIME INTEGRATION OF THE DYNAMIC CHEMICAL FIELD DISTRIBUTION
C  IF SO CALLED FOR.
C     6)  FLUID.  WATER PRESSURE HEAD SYSTEM SOLVED IN THE STEADY
C  STATE; VELOCITY COMPONENTS COMPUTED IF SO CALLED FOR.
C  THE CLASSICAL PEACEMAN-RACHFORD ADII METHOD IS USED.
C     7)  ADILT2.  CLASSICAL PEACEMAN-RACHFORD ALTERNATING
C  DIRECTIONS ITERATIVE IMPLICIT ALGORITHM.
C     8)  THOMLT2.  CLASSICAL THOMAS ALGORITHM SET UP FOR THE WATER
C  PRESSURE HEAD FIELD.
C     9)  CHMREAD.  INPUT OF ALL CHEMICAL SYSTEM PARAMETERS IF SO
C  CALLED FOR.
C    10)  CHMOUT.  OUTPUT OF ALL CHEMICAL SYSTEM PARAMETERS.
C    11)  CHMWRT.  OUTPUT OF THE DYNAMIC CHEMICAL DISTRIBUTION AT
C  SELECTED VALUES OF TIME.
C    12)  PSTAR.  COMPUTES COORDINATES OF P* POINTS FOR CHEMICAL
C  CONCENTRATION.
C    13)  NEWTWO.  SOLVES TWO-DIMENSIONAL NON-LINEAR SYSTEM BY
C  NEWTON-RAPHSON METHOD.
C    14)  FEVAL.  COMPUTES VALUES OF FUNCTIONS AND DERIVATIVES FOR
C  NEWTWO.
C    15)  CHEM.  COMPUTES MATRICES USED IN SOLVING THE CHEMICAL SYSTEM.
C    16)  COMSRC.  COMPUTES ELEMENTS OF SOURC ARRAY.
C    17)  CLAG.  A FUNCTION USED BY INTGRL TO COMPUTE CHEMICAL
C  CONCENTRATION AT THE P* POINTS.
C    18)  TCMLT2.  COMPUTES TOTAL CHEMICAL MASS IN AQUIFER, ETC.
C    19)  PRINT1, PRINT2, PRINT3, PRINT4, ARYSTR, ARYPOL, NUMSTR,
C  NUMFIX, POLAR.  SUBROUTINES TO PRODUCE VARIABLE-FORMAT OUTPUT.

C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
$include:'CSIZE.LT2'
$include:'CCHEM.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
$include:'CADII.LT2'
$include:'CTHOS.LT2'
C
      CHARACTER ENTER
      INTEGER   ICHEM
C
```

```
C     WELCOME MESSAGE
C
      WRITE(*,*)'          TWO-DIMENSIONAL STEADY WATER FLOW     '
      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 is'
      WRITE(*,*)' isotropic and homogeneous.  Velocity components,'
      WRITE(*,*)' dispersion coefficients, and other variables'
      WRITE(*,*)' are calculated at each nodal point.  Chemical '
      WRITE(*,*)' concentration values are computed at each nodal '
      WRITE(*,*)' point at each selected print time.  The flow field '
      WRITE(*,*)' is confined to the interior of the rectangular '
      WRITE(*,*)' boundaries.  Finite Difference (space centered) '
      WRITE(*,*)' methods are used for both fields. '
      WRITE(*,*)
      WRITE(*,*)' ++++++++++++++++++++++++++++++++++++++++++++++++++'
      WRITE(*,*)' Copyright 1989 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) 754-2441'
      WRITE(*,*)' ++++++++++++++++++++++++++++++++++++++++++++++++++'
      WRITE(*,1)'  Press ENTER or RETURN to continue:'
    1 FORMAT(A,$)
      READ(*,2) ENTER
    2 FORMAT(A)
      WRITE(*,*)
      WRITE(*,*)
C
C   UNITS 2, 4, 5, 6, AND 9 USED FOR OUTPUT, UNITS 1 AND 3 FOR INPUT
C
      WRITE(*,*) ' Input file1  is: WATLT2.DAT'
      WRITE(*,*) ' Output file2 is: WATLT2.OUT'
      WRITE(*,*) ' Output file5 is: HYDROOUT.OUT'
      WRITE(*,*)
      WRITE(*,*) ' Input file3  is: CHMLT2.DAT'
      WRITE(*,*) ' Output file4 is: CHMLT2.OUT'
      WRITE(*,*) ' Output file6 is: CHEMFOUT.OUT'
      WRITE(*,*) ' Output file9 is: DEBUGLT2.OUT'
      WRITE(*,*)
C
      OPEN(UNIT=1,FILE='WATLT2.DAT',STATUS='OLD')
      OPEN(UNIT=2,FILE='WATLT2.OUT')
      OPEN(UNIT=3,FILE='CHMLT2.DAT',STATUS='OLD')
      OPEN(UNIT=4,FILE='CHMLT2.OUT')
      OPEN(UNIT=5,FILE='HYDROOUT.OUT')
```

```
      OPEN(UNIT=6,FILE='CHEMFOUT.OUT')
      OPEN(UNIT=9,FILE='DEBUGLT2.OUT')
C
      WRITE(*,*)' Calling subroutine FLOREAD'
      WRITE(*,*)
      CALL FLOREAD
      WRITE(*,*)' All flow field data successfully read in.'
      WRITE(*,*)
C
C     WRITE INPUT AND CALCULATED PARAMS. TO WATLT2.OUT
C
      WRITE(*,*)' Writing input and calculated params. to WATLT2.OUT'
      CALL FLOOUT
      WRITE(*,*)
C
C  SIMULATE SYSTEM OPERATION
C
      WRITE(*,*)' Beginning system simulation.'
      WRITE(*,*)
C
      ICHEM=0
C
      CALL INTGRL(ICHEM)
      WRITE(*,*)' Writing hydraulic field to HYDROOUT.OUT'
      WRITE(*,*)
C
      IF(NFLAG(1).NE.2) GO TO 10
C
      WRITE(*,*)' You have chosen to simulate the chemical field too!'
      WRITE(*,*)
      ICHEM=1
      WRITE(*,*)' Calling subroutine CHMREAD'
      WRITE(*,*)
      CALL CHMREAD
      WRITE(*,*)' All chemical field data successfully read in'
      WRITE(*,*)
C
C     WRITE INPUT AND CALCULATED DATA TO CHMLT2.OUT
C
      WRITE(*,*)' Writing input and calculated params. to CHMLT2.OUT'
      WRITE(*,*)
      CALL INTGRL(ICHEM)
      WRITE(*,*)' Writing chemical conc. distrib. to CHEMFOUT.OUT'
      WRITE(*,*)
      GO TO 20
C
   10 WRITE(*,*)' You have chosen to compute the water flow field only'
      WRITE(*,*)
C
   20 WRITE(*,*)' End of this simulation run.'
```

```
      STOP' Normal Fortran Termination'
      END
```

```
*         File: ADILT2.FOR          Last revision: August 15, 1989
C
      SUBROUTINE ADILT2
C
C  5-POINT COMPUTATIONAL MOLECULE FROM THE FINITE DIFFERENCE METHOD.
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  !!!!! THIS VERSION IS FOR LTRSKAQ2,ETC. ONLY !!!!!
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
$include:'CSIZE.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
$include:'CADII.LT2'
$include:'CTHOS.LT2'
C
      INTEGER I, ICADII, J, KKK, NINTNO, NNTEST
      DOUBLE PRECISION ADI000, ATEST, ATEST1, ATEST2, DELTA0
      DOUBLE PRECISION RATIO, RMSE, SUMERR, XTESTN, XTESTO
      DOUBLE PRECISION XDUMN(IX,IY),XTEMP(IX,IY)
C
      WRITE(*,1000)
 1000 FORMAT(7X,'ADIPRM',9X,'RMSE',9X,'RATIO',6X,'ICADII')
C
C  BEGIN LOOP FOR THE ADIIM METHOD.
C
      NINTNO=NSLXM1*NSLYM1
      ICADII=0
      ATEST1=1.0D+2
      ADI000=ADIPRM
      DELTA0=DELTA
  300 CONTINUE
C
C  FIRST SWEEP OF THE ADIIM PROCEDURE(VERTICAL ORDERING OF NODAL POINTS)
C
C  TOP-LEFT HAND NODE.
      ATDT(2)=0.0D0
      BTDT(2)=ADIPRM+ADT2Y(2,2)
      CTDT(2)=ADT3(2,2)
      YTDUM(2)=YDUM(2,2)+(ADIPRM-ADT2X(2,2))*XDUM(2,2)
     &       -AUT2(2,2)*XDUM(3,2)
```

```
C
C  LEFT HAND CENTRAL NODES.
C
      DO 169, J=3,NSLYM1
      ATDT(J)=ADT1(2,J)
      BTDT(J)=ADIPRM+ADT2Y(2,J)
      CTDT(J)=ADT3(2,J)
      YTDUM(J)=YDUM(2,J)+(ADIPRM-ADT2X(2,J))*XDUM(2,J)
     &       -AUT2(2,J)*XDUM(3,J)
  169 CONTINUE
C
C  BOTTOM-LEFT HAND NODE.
C
      ATDT(NSLYYY)=ADT1(2,NSLYYY)
      BTDT(NSLYYY)=ADIPRM+ADT2Y(2,NSLYYY)
      CTDT(NSLYYY)=0.0D0
      YTDUM(NSLYYY)=YDUM(2,NSLYYY)+(ADIPRM-ADT2X(2,NSLYYY))*
     &      XDUM(2,NSLYYY)-AUT2(2,NSLYYY)*XDUM(3,NSLYYY)
C
C  CENTRAL BLOCK OF NODES.
C
      KKK=NSLYYY
      DO 170, I=3,NSLXM1
C
C  TOP NODES.
C
      KKK=KKK+1
      ATDT(KKK)=0.0D0
      BTDT(KKK)=ADIPRM+ADT2Y(I,2)
      CTDT(KKK)=ADT3(I,2)
      YTDUM(KKK)=YDUM(I,2)-ALT2(I,2)*XDUM(I-1,2)
     &      +(ADIPRM-ADT2X(I,2))*XDUM(I,2)-AUT2(I,2)*XDUM(I+1,2)
C
C  INTERIOR BLOCK OF NODES
C
      DO 171, J=3,NSLYM1
      KKK=KKK+1
      ATDT(KKK)=ADT1(I,J)
      BTDT(KKK)=ADIPRM+ADT2Y(I,J)
      CTDT(KKK)=ADT3(I,J)
      YTDUM(KKK)=YDUM(I,J)-ALT2(I,J)*XDUM(I-1,J)
     &      +(ADIPRM-ADT2X(I,J))*XDUM(I,J)-AUT2(I,J)*XDUM(I+1,J)
  171 CONTINUE
C
C  BOTTOM NODES
C
      KKK=KKK+1
      ATDT(KKK)=ADT1(I,NSLYYY)
      BTDT(KKK)=ADIPRM+ADT2Y(I,NSLYYY)
      CTDT(KKK)=0.0D0
```

```
      YTDUM(KKK)=YDUM(I,NSLYYY)-ALT2(I,NSLYYY)*XDUM(I-1,NSLYYY)
     &     +(ADIPRM-ADT2X(I,NSLYYY))*XDUM(I,NSLYYY)
     &     -AUT2(I,NSLYYY)*XDUM(I+1,NSLYYY)
C
  170 CONTINUE
C
C  TOP-RIGHT HAND NODE.
C
      KKK=KKK+1
      ATDT(KKK)=0.0D0
      BTDT(KKK)=ADIPRM+ADT2Y(NSLXXX,2)
      CTDT(KKK)=ADT3(NSLXXX,2)
      YTDUM(KKK)=YDUM(NSLXXX,2)-ALT2(NSLXXX,2)*XDUM(NSLXM1,2)
     &     +(ADIPRM-ADT2X(NSLXXX,2))*XDUM(NSLXXX,2)
C
C  RIGHT HAND CENTRAL NODES.
C
      DO 177, J=3,NSLYM1
      KKK=KKK+1
      ATDT(KKK)=ADT1(NSLXXX,J)
      BTDT(KKK)=ADIPRM+ADT2Y(NSLXXX,J)
      CTDT(KKK)=ADT3(NSLXXX,J)
      YTDUM(KKK)=YDUM(NSLXXX,J)-ALT2(NSLXXX,J)*XDUM(NSLXM1,J)
     &     +(ADIPRM-ADT2X(NSLXXX,J))*XDUM(NSLXXX,J)
  177 CONTINUE
C
C  BOTTOM RIGHT HAND NODE.
C
      KKK=KKK+1
      ATDT(KKK)=ADT1(NSLXXX,NSLYYY)
      BTDT(KKK)=ADIPRM+ADT2Y(NSLXXX,NSLYYY)
      CTDT(KKK)=0.0D0
      YTDUM(KKK)=YDUM(NSLXXX,NSLYYY)-ALT2(NSLXXX,NSLYYY)*
     &     XDUM(NSLXM1,NSLYYY)+(ADIPRM-ADT2X(NSLXXX,NSLYYY))*
     &     XDUM(NSLXXX,NSLYYY)
C
C     SOLVE THE TRIDIAGONAL SYSTEM VIA THE CLASSICAL THOMAS ALGORITHM.
C
      CALL THOMLT2
C
C     RESTRUCTURE THE INTERMEDIATE VERTICAL LINE DERIVED ESTIMATES.
C
      KKK=2
      DO 180, I=2,NSLXXX
      DO 181, J=2,NSLYYY
      XTEMP(I,J)=XTDUM(KKK)
      KKK=KKK+1
  181 CONTINUE
  180 CONTINUE
C
```

```
C+++++++++++++++++++++++++++++++++++++++++++++++
C      WRITE(*,*)'Step one of the ADII METHOD'
C      DO 700 J=2,NSLYYY
C  700 WRITE(*,701)(XTEMP(I,J),I=2,NSLXXX)
C  701 FORMAT(1X,10E12.6)
C      WRITE(*,*)
C+++++++++++++++++++++++++++++++++++++++++++++++
C
C     THE FIRST STEP OF THE ADII METHOD HAS BEEN COMPLETED.  NOW BEGIN
C     THE SECOND STEP, THE HORIZONTAL LINE DERIVED ESTIMATION.
C
C  SECOND SWEEP OF THE ADIIM PROCEDURE(HORIZONTAL ORDERING
C  OF NODAL POINTS).
C
C  TOP-LEFT HAND NODE.
      ATDT(2)=0.0D0
      BTDT(2)=ADIPRM+ADT2X(2,2)
      CTDT(2)=AUT2(2,2)
      YTDUM(2)=YDUM(2,2)+(ADIPRM-ADT2Y(2,2))*XTEMP(2,2)
     &       -ADT3(2,2)*XTEMP(2,3)
C
C  TOP-CENTRAL NODES.
C
      DO 185, I=3,NSLXM1
      ATDT(I)=ALT2(I,2)
      BTDT(I)=ADIPRM+ADT2X(I,2)
      CTDT(I)=AUT2(I,2)
      YTDUM(I)=YDUM(I,2)+(ADIPRM-ADT2Y(I,2))*XTEMP(I,2)
     &       -ADT3(I,2)*XTEMP(I,3)
  185 CONTINUE
C
C  TOP-RIGHT HAND NODE.
C
      ATDT(NSLXXX)=ALT2(NSLXXX,2)
      BTDT(NSLXXX)=ADIPRM+ADT2X(NSLXXX,2)
      CTDT(NSLXXX)=0.0D0
      YTDUM(NSLXXX)=YDUM(NSLXXX,2)+(ADIPRM-ADT2Y(NSLXXX,2))*
     &      XTEMP(NSLXXX,2)-ADT3(NSLXXX,2)*XTEMP(NSLXXX,3)
C
      KKK=NSLXXX
C
C  CENTRAL BLOCK OF NODES.
C
      DO 186, J=3,NSLYM1
C
C  LEFT HAND NODES.
      KKK=KKK+1
      ATDT(KKK)=0.0D0
      BTDT(KKK)=ADIPRM+ADT2X(2,J)
      CTDT(KKK)=AUT2(2,J)
```

```
      YTDUM(KKK)=YDUM(2,J)-ADT1(2,J)*XTEMP(2,J-1)
     &     +(ADIPRM-ADT2Y(2,J))*XTEMP(2,J)-ADT3(2,J)*XTEMP(2,J+1)
C
C  INTERIOR BLOCK OF NODES
C
      DO 187, I=3,NSLXM1
      KKK=KKK+1
      ATDT(KKK)=ALT2(I,J)
      BTDT(KKK)=ADIPRM+ADT2X(I,J)
      CTDT(KKK)=AUT2(I,J)
      YTDUM(KKK)=YDUM(I,J)-ADT1(I,J)*XTEMP(I,J-1)
     &     +(ADIPRM-ADT2Y(I,J))*XTEMP(I,J)-ADT3(I,J)*XTEMP(I,J+1)
  187 CONTINUE
C
C  RIGHT HAND NODES
C
      KKK=KKK+1
      ATDT(KKK)=ALT2(NSLXXX,J)
      BTDT(KKK)=ADIPRM+ADT2X(NSLXXX,J)
      CTDT(KKK)=0.0D0
      YTDUM(KKK)=YDUM(NSLXXX,J)-ADT1(NSLXXX,J)*XTEMP(NSLXXX,J-1)
     &     +(ADIPRM-ADT2Y(NSLXXX,J))*XTEMP(NSLXXX,J)
     &     -ADT3(NSLXXX,J)*XTEMP(NSLXXX,J+1)
C
  186 CONTINUE
C
C  BOTTOM SET OF NODES.
C
C  LEFT HAND NODE
C
      KKK=KKK+1
      ATDT(KKK)=0.0D0
      BTDT(KKK)=ADIPRM+ADT2X(2,NSLYYY)
      CTDT(KKK)=AUT2(2,NSLYYY)
      YTDUM(KKK)=YDUM(2,NSLYYY)-ADT1(2,NSLYYY)*XTEMP(2,NSLYM1)
     &     +(ADIPRM-ADT2Y(2,NSLYYY))*XTEMP(2,NSLYYY)
C
C  BOTTOM CENTRAL NODES.
C
      DO 188, I=3,NSLXM1
      KKK=KKK+1
      ATDT(KKK)=ALT2(I,NSLYYY)
      BTDT(KKK)=ADIPRM+ADT2X(I,NSLYYY)
      CTDT(KKK)=AUT2(I,NSLYYY)
      YTDUM(KKK)=YDUM(I,NSLYYY)-ADT1(I,NSLYYY)*XTEMP(I,NSLYM1)
     &     +(ADIPRM-ADT2Y(I,NSLYYY))*XTEMP(I,NSLYYY)
  188 CONTINUE
C
C  BOTTOM RIGHT HAND NODE.
C
```

```
      KKK=KKK+1
      ATDT(KKK)=ALT2(NSLXXX,NSLYYY)
      BTDT(KKK)=ADIPRM+ADT2X(NSLXXX,NSLYYY)
      CTDT(KKK)=0.0D0
      YTDUM(KKK)=YDUM(NSLXXX,NSLYYY)-ADT1(NSLXXX,NSLYYY)*
     &     XTEMP(NSLXXX,NSLYM1)+(ADIPRM-ADT2Y(NSLXXX,NSLYYY))*
     &     XTEMP(NSLXXX,NSLYYY)
C
C     SOLVE THE TRIDIAGONAL SYSTEM VIA THE CLASSICAL THOMAS ALGORITHM.
C
      CALL THOMLT2
C
C     RESTRUCTURE THE INTERMEDIATE HORIZONTAL LINE DERIVED ESTIMATES.
C
      KKK=2
      DO 190, J=2,NSLYYY
      DO 191, I=2,NSLXXX
      XDUMN(I,J)=XTDUM(KKK)
      KKK=KKK+1
  191 CONTINUE
  190 CONTINUE
C
C+++++++++++++++++++++++++++++++++++++++++++++++
C      WRITE(*,*)'Step two of the ADII METHOD'
C      DO 702 J=2,NSLYYY
C  702 WRITE(*,701)(XDUMN(I,J),I=2,NSLXXX)
C      WRITE(*,*)
C+++++++++++++++++++++++++++++++++++++++++++++++
C
C     THE SECOND STEP OF THE ADII METHOD HAS BEEN COMPLETED.  NOW BEGIN
C     THE CONVERGENCE TEST STEP.
C
C  CONVERGENCE TEST
C
      ICADII=ICADII+1
      SUMERR=0.0D0
      DO 150, I=2,NSLXXX
        DO 151, J=2,NSLYYY
          XTESTN=XDUMN(I,J)
          XTESTO=XDUM(I,J)
          IF(XTESTN.EQ.0.0D0) GO TO 704
          ATEST=(XTESTN-XTESTO)/XTESTN
          SUMERR=SUMERR+ATEST*ATEST
  704     CONTINUE
  151   CONTINUE
  150 CONTINUE
      SUMERR=SUMERR/FLOAT(NINTNO)
      RMSE=DSQRT(SUMERR)
      ATEST2=RMSE
      IF(RMSE.EQ.0.0D0) ATEST2=1.0D0
```

```
      RATIO=ATEST1/ATEST2
      IF(RMSE.GE.TLRNWA) GO TO 160
C
C  REDEFINE VARIABLES
C
      DO 155, I=2,NSLXXX
         DO 156, J=2,NSLYYY
            XDUM(I,J)=XDUMN(I,J)
  156    CONTINUE
  155 CONTINUE
      GO TO 200
C
C  EVERY NMOD'TH ITERATION, DISPLAY ADII PARAMETERS (OR DISPLAY A
C  DOT), AND CHANGE ADIPRM.
C
  160 IF(ICADII.GT.NLSOR) GO TO 500
      NNTEST=MOD(ICADII,NMOD)
      IF(NNTEST.NE.0) GO TO 710
      IF(NFLAG(3).EQ.1) THEN
         WRITE(*,1162) ADIPRM,RMSE,RATIO,ICADII
 1162    FORMAT(1P,5X,E12.5,2X,E12.5,2X,E12.5,2X,I5)
      ELSE
         WRITE(*,'('' .'',$)')
      ENDIF
      ATEST1=ATEST2
C
C  MODIFY ADIPRM, OR READ IN A NEW VALUE FOR IT.
C
      IF (NFLAG(4).EQ.0) THEN
         IF(ADIPRM.LT.ADIMAX) THEN
            ADIPRM=ADIPRM*(1.0D0+DELTA)
         ELSE
            ADIPRM=ADI000
         ENDIF
      ELSE
  410    WRITE(*,1500)
 1500    FORMAT(' Enter new value for ADIPRM: ',$)
         READ(*,*,ERR=420) ADIPRM
         GOTO 430
  420    WRITE(*,*) ' *** ERROR in data entry! ***'
         GOTO 410
  430 ENDIF
C
C  UPDATE VARIABLES
C
  710 DO 157, I=2,NSLXXX
         DO 158, J=2,NSLYYY
            XDUM(I,J)=XDUMN(I,J)
  158    CONTINUE
  157 CONTINUE
```

```
      GO TO 300
C
  500 WRITE(*,1400)
 1400 FORMAT(/,10X,'*** WARNING: ICADII EXCEEDS NLSOR***',/)
      STOP 1
C
  200 WRITE(*,1162) ADIPRM,RMSE,RATIO,ICADII
      WRITE(*,1201) ICADII
 1201 FORMAT(5X,' Took ',I6,' iterations.')
      ADIPRM=ADI000
      DELTA=DELTA0
      RETURN
      END
```

```
*          File: CHMLT2.FOR          Last revision: July 19, 1989
C
      SUBROUTINE CHEM
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
$include:'CSIZE.LT2'
$include:'CCHEM.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
$include:'CADII.LT2'
C
      INTEGER I, J
      DOUBLE PRECISION CONSTX, CONSTY
      DOUBLE PRECISION DENOMI
      DOUBLE PRECISION DXYIM, DXYIP, DXYJM, DXYJP
      DOUBLE PRECISION DXXIM, DXXIP, DYYJM, DYYJP
C
C     DEFINE MATRIX ELEMENTS FOR CHEMICAL FIELD TIME MARCHING MATRIX.
C
      DTRET=DT0/(1.0D0+RETARD)
C
      DO 30, I=2,NSLXXX
         CONSTX=(DX(I-1)+DX(I))/2.0D0
         DO 31, J=2,NSLYYY
            CONSTY=(DY(J-1)+DY(J))/2.0D0
C
            DXYIM=(DCHLXY(I-1,J)+DCHLXY(I,J))/(8.0D0)
            DXYIP=(DCHLXY(I,J)+DCHLXY(I+1,J))/(8.0D0)
            DXYJM=(DCHLXY(I,J-1)+DCHLXY(I,J))/(8.0D0)
            DXYJP=(DCHLXY(I,J)+DCHLXY(I,J+1))/(8.0D0)
C
            DXXIM=(DCHLX(I-1,J)+DCHLX(I,J))/(2.0D0*DX(I-1))
            DXXIP=(DCHLX(I,J)+DCHLX(I+1,J))/(2.0D0*DX(I))
```

```
C
            DYYJM=(DCHLY(I,J-1)+DCHLY(I,J))/(2.0D0*DY(J-1))
            DYYJP=(DCHLY(I,J)+DCHLY(I,J+1))/(2.0D0*DY(J))
C
            DENOMI=CONSTX*CONSTY
C
            ALT1(I,J)=DT0*(DXYIM+DXYJM)/DENOMI
            ALT2(I,J)=DT0*(CONSTY*DXXIM+(DXYJM-DXYJP))/DENOMI
            ALT3(I,J)=-DT0*(DXYJP+DXYIM)/DENOMI
C
            ADT1(I,J)=DT0*(CONSTX*DYYJM+(DXYIM-DXYIP))/DENOMI
            ADT2(I,J)=-DT0*(CONSTY*(DXXIP+DXXIM)+
     &          CONSTX*(DYYJP+DYYJM))/DENOMI
            ADT3(I,J)=DT0*(CONSTX*DYYJP+(DXYIP-DXYIM))/DENOMI
C
            AUT1(I,J)=-DT0*(DXYIP+DXYJM)/DENOMI
            AUT2(I,J)=DT0*(CONSTY*DXXIP+(DXYJP-DXYJM))/DENOMI
            AUT3(I,J)=DT0*(DXYIP+DXYJP)/DENOMI
C
   31    CONTINUE
   30 CONTINUE
C
      IF(NFLAG(14).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,ALT1(I,J),ALT2(I,J),ALT3(I,J),
     &         ADT1(I,J),ADT2(I,J),ADT3(I,J),
     &         AUT1(I,J),AUT2(I,J),AUT3(I,J)
  801    CONTINUE
  800 CONTINUE
  802 FORMAT(1X,2I3,9E12.6)
      WRITE(9,*)
C+++++++++++++++++++++++++++++++++++++++++
C
  950 CONTINUE
C
      RETURN
      END
```

```
*       File: ITGRLT2.FOR       Last revision:  July 19, 1989
C
      SUBROUTINE INTGRL(ICHEM)
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      INTEGER ICHEM
$include:'CSIZE.LT2'
$include:'CCHEM.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
$include:'CADII.LT2'
$include:'CTHOS.LT2'
C
      INTEGER I, J, KPRT, KSWT
      DOUBLE PRECISION ARG1, ARG2
      DOUBLE PRECISION CON1, CON2, CON3, CON4, CON5, CON6
      DOUBLE PRECISION CONST1, CONST2, CONST3, CONST4
      DOUBLE PRECISION ECIN, ECOUT, EEIN, EEOUT
      DOUBLE PRECISION EPSRET, ERTEMP, PRET
      DOUBLE PRECISION SUIN1, SUIN2, SUIN3, SUOUT1, SUOUT2
      DOUBLE PRECISION TEMP, TIME
      DOUBLE PRECISION UIN1, UIN2, UIN3, UOUT1, UOUT2
      DOUBLE PRECISION XLTEMP, XMAG, XSTEMP
C
      DOUBLE PRECISION CLAG
      EXTERNAL         CLAG
C
      IF(ICHEM.EQ.1) GO TO 2000
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)=0.0D0
       HOLD(I,NSLYP1)=0.0D0
  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
      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
      CONST1=(2.0D0+(DX(2)/DX(1))+(DX(1)/DX(2)))/(2.0D0+DX(2)/DX(1))
      CONST2=(DX(1)/DX(2))/(2.0D0+DX(2)/DX(1))
      CONST3=(2.0D0+(DX(NSLXXX)/DX(NSLXM1))+(DX(NSLXM1)/DX(NSLXXX)))/
     &             (2.0D0+(DX(NSLXM1)/DX(NSLXXX)))
      CONST4=(DX(NSLXXX)/DX(NSLXM1))/
     &             (2.0D0+(DX(NSLXM1)/DX(NSLXXX)))
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
      DO 221, I=1,NSLXP1
       DO 220, J=1,NSLYP1
          HNEW(I,J)=HIN+(HOUT-HIN)*(YNODE(J)/YNODE(NSLYP1))+
     &             HNEW(I,J)
  220  CONTINUE
  221 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)=-KTHSTS*PHNEWX(I,J)
           VLYY(I,J)=-KTHSTS*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
      GO TO 5000
C
C  SECTION 3: DYNAMIC CHEMICAL FIELD DISTRIBUTION.
C
 2000 CONTINUE
      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
C  RETARDATION AND FIRST ORDER LOSS PROCESS COEFFICIENTS.
C
      PRET=(1.0D0-EPS)/EPS
      RETARD=PRET*(PCTSAN*RHOSND*KSAND+PCTCLA*
     &            RHOCLA*KCLAY+PCTORG*RHOORG*KORG)
      EPSRET=EPS*(1.0D0+RETARD)
      XLTEMP=XLAM10+XLAMIR+XLAMRA
      XSTEMP=XSLM10+XSLMRA
      LAMDA=RETARD*XSTEMP+XLTEMP
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)/EPSRET
          VLYY(I,J)=VLYY(I,J)/EPSRET
  310  CONTINUE
  311 CONTINUE
C
C  DISPERSION COEFFICIENTS.
C
```

```
      ERTEMP=TORT*DL0/(1.0D0+RETARD)
      DO 301, I=2,NSLXXX
       DO 300, J=1,NSLYP1
          XMAG=DSQRT(VLXX(I,J)*VLXX(I,J)+VLYY(I,J)*VLYY(I,J))
          DCHLX(I,J)=ERTEMP+(DISPLY*VLXX(I,J)*VLXX(I,J)+
     &             DISPLX*VLYY(I,J)*VLYY(I,J))/XMAG
          DCHLY(I,J)=ERTEMP+(DISPLX*VLXX(I,J)*VLXX(I,J)+
     &             DISPLY*VLYY(I,J)*VLYY(I,J))/XMAG
          DCHLXY(I,J)=(DISPLY-DISPLX)*VLXX(I,J)*VLYY(I,J)/XMAG
  300  CONTINUE
  301 CONTINUE
C
      DO 305, J=1,NSLYP1
        DCHLX(1,J)=ERTEMP
        DCHLY(1,J)=ERTEMP
        DCHLXY(1,J)=0.0D0
        DCHLX(NSLXP1,J)=ERTEMP
        DCHLY(NSLXP1,J)=ERTEMP
        DCHLXY(NSLXP1,J)=0.0D0
  305 CONTINUE
C
C  COMPUTE DT1 (MAXIMUM DELTA-T); STABILITY CRITERION.
C
      DT1 = 0.D0
      DO 341, I=1,NSLXXX
       DO 340, J=1,NSLYYY
          CDENOM(I,J)=(LAMDA+QWELIN(I,J)/
     &                        (EPS*RHOWAT) )/(1.0D0+RETARD)
          TEMP = CDENOM(I,J) +
     &           2.D0*( ABS(VLXX(I,J))/DX(I) +
     &                ABS(VLYY(I,J))/DY(J) +
     &            DCHLX(I,J)/DX(I)**2 + DCHLY(I,J)/DY(J)**2 )
          IF (TEMP .GT. DT1) DT1 = TEMP
  340  CONTINUE
  341 CONTINUE
      IF (DT1 .NE. 0.D0) THEN
         DT1 = 1.0D0 / DT1
      ELSE
         STOP 'DT1 0.0???'
      ENDIF
C
C  PRINT CHEMICAL PROPERTIES ON FILE CHMLT2.OUT.
C
      CALL CHMOUT
C
      WRITE(4,9100)
 9100 FORMAT(//,10X, 'RETARDATION AND OVER ALL FIRST ORDER LOSS',
     &           ' PROCESS COEFFICIENT ',/)
      WRITE(4,9200) RETARD
 9200 FORMAT(1P,5X, 'RETARD= ',E12.5,/)
```

```
      WRITE(4,9300) LAMDA
 9300 FORMAT(1P,5X, 'LAMDA=  ',E12.5,' (1/DAY)',/)
C
C  IF DT1/2<DT0, USE DT1/2 AS DELTA-T.
C
      IF (DT1*0.5D0.LT.DT0) DT0=DT1*0.5D0
C
C  COMPUTE P*(i,j) FOR INTERIOR NODES.
C
      PSERR=.FALSE.
      DO 321, I=2,NSLXXX
       DO 320, J=2,NSLYYY
          CALL PSTAR(I,J)
  320  CONTINUE
  321 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 'DEBUGTL2.OUT'
C
      IF(NFLAG(7).NE.0) THEN
        WRITE(9,9500)
 9500   FORMAT(/,6X,'COORDINATES OF THE P* POINTS',//,
     &         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 331, I=2,NSLXXX
          DO 330, J=2,NSLYYY
             WRITE(9,9600)I,J,XNODE(I),PSTARX(I,J),
     &                YNODE(J),PSTARY(I,J),NFUNC(I,J)
 9600        FORMAT(2(1X,I4),2X,2F10.4,3X,2F10.4,2X,I3)
  330     CONTINUE
  331    CONTINUE
      ENDIF
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
      SUIN2 =0.0D0
      SUOUT1=0.0D0
      DO 350 I=1,NSLXXX
        SUIN1=SUIN1+DX(I)*(VLYY(I,1)+VLYY(I+1,1))/2.0D0
        SUIN2=SUIN2+DX(I)*(DCHLY(I,1)/DY(1)+VLYY(I,1)+
     &         DCHLY(I+1,1)/DY(1)+VLYY(I+1,1))/2.0D0
```

```
        SUOUT1=SUOUT1+DX(I)*(DCHLY(I,NSLYP1)/DY(NSLYYY)+
     &         DCHLY(I+1,NSLYP1)/DY(NSLYYY))/2.0D0
  350 CONTINUE
C
      UIN1 =SUIN1 *EPSRET/XNODE(NSLXP1)
      UIN2 =SUIN2 *EPSRET/XNODE(NSLXP1)
      UOUT1=SUOUT1*EPSRET/XNODE(NSLXP1)
      ARG1=DT0*UIN2 /XLYIN
      ARG2=DT0*UOUT1/XLYOUT
      EEIN =DEXP(-ARG1)
      EEOUT=DEXP(-ARG2)
      ECIN =1.0D0-EEIN
      ECOUT=1.0D0-EEOUT
C
      CONST1=(2.0D0+(DX(2)/DX(1))+(DX(1)/DX(2)))/(2.0D0+DX(2)/DX(1))
      CONST2=(DX(1)/DX(2))/(2.0D0+DX(2)/DX(1))
      CONST3=(2.0D0+(DX(NSLXXX)/DX(NSLXM1))+(DX(NSLXM1)/DX(NSLXXX)))/
     &            (2.0D0+(DX(NSLXM1)/DX(NSLXXX)))
      CONST4=(DX(NSLXXX)/DX(NSLXM1))/
     &            (2.0D0+(DX(NSLXM1)/DX(NSLXXX)))
C
      DO 355, J=2,NSLYYY
        COLD(1,J)=CONST1*COLD(2,J)-CONST2*COLD(3,J)
        COLD(NSLXP1,J)=-CONST4*COLD(NSLXM1,J)+CONST3*COLD(NSLXXX,J)
  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
      TMAX = TMAX - 0.01D0*DT0
      KPRT=1
      KSWT=1
      XMASS =0.0D0
      XMFONW=0.0D0
      XMSOUR=0.0D0
      XMASIN=0.0D0
      XMASOT=0.0D0
C
      CALL CHEM
      CALL COMSRC
C
C  BEGIN LOOP
C
 3000 CONTINUE
C
C  IF NEXT INJECTION WELL SWITCHING TIME IS LESS THAN HALF OF
```

```
C  DELTA-T (DT0/2) AHEAD, (OR WITHIN HALF OF DELTA-T IN THE PAST),
C  SWITCH THE WELLS.
C
      IF (NINJW.GT.0 .AND. SWTIME(KSWT)-TIME.LE.DT0*0.5D0) THEN
         WELLON=.NOT. WELLON
         DO 360, I=1,NINJW
            IF (WELLON) THEN
               CSWELN(INJI(I),INJJ(I))=CHMCON(I)
            ELSE
               CSWELN(INJI(I),INJJ(I))=0.D0
            ENDIF
360      CONTINUE
C
         CALL COMSRC
C
         KSWT=KSWT+1
         IF (WELLON) THEN
            WRITE(6,9700) 'ON', TIME
         ELSE
            WRITE(6,9700) 'OFF', TIME
         ENDIF
      ENDIF
9700  FORMAT(/,5X,'INJECTION WELLS SWITCHED ',A,' AT TIME T=',F10.4,
     &   ' (DAYS)')
C
C  SUB SECTION 3.D: COMPUTE CHEMICAL FIELD VARIABLES.
C
      DO 371, I=2,NSLXXX
       DO 370, J=2,NSLYYY
          CNEW(I,J)=ALT1(I,J)*COLD(I-1,J-1)+ALT2(I,J)*COLD(I-1,J)+
     &            ALT3(I,J)*COLD(I-1,J+1)+ADT1(I,J)*COLD(I,J-1)+
     &            ADT2(I,J)*COLD(I,J)+ADT3(I,J)*COLD(I,J+1)+
     &            AUT1(I,J)*COLD(I+1,J-1)+AUT2(I,J)*COLD(I+1,J)+
     &            AUT3(I,J)*COLD(I+1,J+1)+SOURC(I,J)+
     &            CLAG(I,J)-DT0*CDENOM(I,J)*COLD(I,J)
  370  CONTINUE
  371 CONTINUE
C
      SUIN3 =0.0D0
      SUOUT2=0.0D0
      DO 380 I=1,NSLXXX
        SUIN3=SUIN3+DX(I)*((DCHLY(I,1)/DY(1))*COLD(I,2)+
     &     (DCHLY(I+1,1)/DY(1))*COLD(I+1,2))/2.0D0
        SUOUT2=SUOUT2+DX(I)*((DCHLY(I,NSLYP1)/DY(NSLYYY))*
     &     COLD(I,NSLYYY)+
     &     (DCHLY(I+1,NSLYP1)/DY(NSLYYY))*COLD(I+1,NSLYYY))/2.0D0
  380 CONTINUE
C
      UIN3 =SUIN3 *EPSRET/XNODE(NSLXP1)
      UOUT2=SUOUT2*EPSRET/XNODE(NSLXP1)
```

```
      CIN=CIN*EEIN+((C0*UIN1+UIN3)/UIN2)*ECIN
      COUT=COUT*EEOUT+(UOUT2/UOUT1)*ECOUT
      DO 381 I=1,NSLXP1
        CNEW(I,1)=CIN
        CNEW(I,NSLYP1)=COUT
  381 CONTINUE
C
      DO 356, J=1,NSLYP1
       CNEW(1,J)=CONST1*CNEW(2,J)-CONST2*CNEW(3,J)
       CNEW(NSLXP1,J)=-CONST4*CNEW(NSLXM1,J)+CONST3*CNEW(NSLXXX,J)
  356 CONTINUE
C
      TIME=TIME+DT0
C
      CALL TCMLT2
C
      WRITE(*,9900)' Time = ',TIME
 9900 FORMAT(1X,A,2X,F10.4)
      IF(TIME.LT.PRTIME(KPRT)) GO TO 4000
      KPRT=KPRT+1
C
      CALL CHMWRT(TIME)
C
 4000 DO 391, I=1,NSLXP1
       DO 390, J=1,NSLYP1
          COLD(I,J)=CNEW(I,J)
  390  CONTINUE
  391 CONTINUE
      IF(TIME.LT.TMAX) GO TO 3000
      WRITE(*,*)
      WRITE(*,*)' Time t meets or exceeds TMAX! CEASE COMPUTING! '
      WRITE(*,*)
 5000 RETURN
      END
C
C *********************************************
C
      SUBROUTINE COMSRC
C
C  COMPUTES ELEMENTS OF SOURC ARRAY.  CALL COMSRC AFTER CALLING CHEM,
C  AND WHENEVER THE WELLS ARE SWITCHED.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
$include:'CSIZE.LT2'
$include:'CCHEM.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
C
      INTEGER I, J
C
```

```
      DO 21, I=2,NSLXXX
         DO 20, J=2,NSLYYY
            SOURC(I,J)=DTRET*
     &          (QCHM1S(I,J)+QWELIN(I,J)*CSWELN(I,J))/
     &          (EPS)
   20    CONTINUE
   21 CONTINUE
C
      RETURN
      END
```

```
*NEWTWO                                 Last Revision:  March 29, 1989
C                                       Source File:  NEWTWO.FOR
      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, I, J, 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
D     WRITE(*,2050)
D     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} + δ(1), and                          *
*     Y{n+1} = G(X{n},Y{n}) = Y{n} + δ(2).                              *
*  δ(1) and δ(2) are found by solving the system:                       *
*          Fx * δ(1) +  Fy * δ(2) = -F                                  *
*          Gx * δ(1) +  Gy * δ(2) = -G                                  *
*  which arises from the Taylor series expansion of F(X+δ(1),Y+δ(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
D               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.')
 2030 FORMAT(1X,'*NEWTWO* - Singular Jacobian Matrix.  Bye.')
 2040 FORMAT(1X,'*NEWTWO* - The impossible has happened (case).')
D2050 FORMAT(/,1X,'ITER ', '          TEST', '          X(N)')
D2060 FORMAT(1X,I3, 2X, 1P, 4(1X,G13.6), /, (T20,3(1X,G13.6)) )
      END
```

```
*         File: OUTLT2.FOR          Last revision: July 19, 1989
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/March/May/July 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.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
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 PRINT4(LUN,NX,NY,ARX,ARY)
C
$include:'CSIZE.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
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.LT2'
$include:'CMISC.LT2'
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)='°'
      RETURN
      END
```

```
*         File: PSTARLT2.FOR        Last revision: March 29, 1989
C
C  PSTAR computes the coordinates of the point P*(i,j), with some
C  help from subroutines  NEWTWO  and FEVAL.
C  By Gilbert A. Bachelor, December 1988.
C
      SUBROUTINE PSTAR(I,J)
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      INTEGER I, J, MITER
      DOUBLE PRECISION TOLER
      PARAMETER(MITER=10,TOLER=1.0D-4)
$include:'CSIZE.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
$include:'CNRK.LT2'
$include:'CTHOS.LT2'
      INTEGER IERR
      DOUBLE PRECISION AJAC(NDIM,NDIM), RHS(NDIM)
      DOUBLE PRECISION X(NDIM), PAR
      EXTERNAL FEVAL
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)=VLXX(I,J)
      COEF(5)=VLYY(I,J)
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
      COEF(2)=(VLXX(I,J)-VLXX(I-1,J))/DX(I-1)
      COEF(6)=(VLYY(I,J)-VLYY(I-1,J))/DX(I-1)
      COEF(3)=(VLXX(I,J)-VLXX(I,J-1))/DY(J-1)
      COEF(7)=(VLYY(I,J)-VLYY(I,J-1))/DY(J-1)
      COEF(4)=(VLXX(I,J)+VLXX(I-1,J-1)-VLXX(I-1,J)-VLXX(I,J-1))/
     &   (DX(I-1)*DY(J-1))
      COEF(8)=(VLYY(I,J)+VLYY(I-1,J-1)-VLYY(I-1,J)-VLYY(I,J-1))/
     &   (DX(I-1)*DY(J-1))
      GOTO 500
```

```
C
C  Quadrant 2.
C
200    CONTINUE
       COEF(2)=(VLXX(I+1,J)-VLXX(I,J))/DX(I)
       COEF(6)=(VLYY(I+1,J)-VLYY(I,J))/DX(I)
       COEF(3)=(VLXX(I,J)-VLXX(I,J-1))/DY(J-1)
       COEF(7)=(VLYY(I,J)-VLYY(I,J-1))/DY(J-1)
       COEF(4)=(VLXX(I,J-1)+VLXX(I+1,J)-VLXX(I,J)-VLXX(I+1,J-1))/
     &    (DX(I)*DY(J-1))
       COEF(8)=(VLYY(I,J-1)+VLYY(I+1,J)-VLYY(I,J)-VLYY(I+1,J-1))/
     &    (DX(I)*DY(J-1))
       GOTO 500
C
C  Quadrant 3.
C
300    CONTINUE
       COEF(2)=(VLXX(I,J)-VLXX(I-1,J))/DX(I-1)
       COEF(6)=(VLYY(I,J)-VLYY(I-1,J))/DX(I-1)
       COEF(3)=(VLXX(I,J+1)-VLXX(I,J))/DY(J)
       COEF(7)=(VLYY(I,J+1)-VLYY(I,J))/DY(J)
       COEF(4)=(VLXX(I-1,J)+VLXX(I,J+1)-VLXX(I,J)-VLXX(I-1,J+1))/
     &    (DX(I-1)*DY(J))
       COEF(8)=(VLYY(I-1,J)+VLYY(I,J+1)-VLYY(I,J)-VLYY(I-1,J+1))/
     &    (DX(I-1)*DY(J))
       GOTO 500
C
C  Quadrant 4.
C
400    CONTINUE
       COEF(2)=(VLXX(I+1,J)-VLXX(I,J))/DX(I)
       COEF(6)=(VLYY(I+1,J)-VLYY(I,J))/DX(I)
       COEF(3)=(VLXX(I,J+1)-VLXX(I,J))/DY(J)
       COEF(7)=(VLYY(I,J+1)-VLYY(I,J))/DY(J)
       COEF(4)=(VLXX(I+1,J+1)+VLXX(I,J)-VLXX(I+1,J)-VLXX(I,J+1))/
     &    (DX(I)*DY(J))
       COEF(8)=(VLYY(I+1,J+1)+VLYY(I,J)-VLYY(I+1,J)-VLYY(I,J+1))/
     &    (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)=X(1)
      PSTARY(I,J)=X(2)
C
C  Print error messages if anything is wrong.
C
      IF (IERR.LT.1) THEN
         PSERR=.TRUE.
         PRINT 1000,IERR,I,J
1000     FORMAT(1X,'Error #',I3,' in computing PSTAR for I=',
     &      I3,', J=',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
1100     FORMAT(1X,'Error in computing PSTAR for I=',I3,', J=',I3,/,
     &      1X,'Point not in correct region.')
       ENDIF
       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.lt2'
*****************************************************************
*  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
C ****************************************
*CLAG
C  CLAG function computes COLD(P*(I,J))
C  Coded by Gilbert A. Bachelor, December 1988.
C
      DOUBLE PRECISION FUNCTION CLAG(I,J)
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      INTEGER I, J
$include:'CSIZE.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
      DOUBLE PRECISION ALF,BET,GAM,DEL
      DOUBLE PRECISION DDX,DDY
C
C  Compute the coefficients, according to which quadrant P*(i,j)
C  is in.
C
      ALF=COLD(I,J)
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)-COLD(I-1,J))/DX(I-1)
      GAM=(COLD(I,J)-COLD(I,J-1))/DY(J-1)
```

```
      DEL=(COLD(I,J)+COLD(I-1,J-1)-COLD(I-1,J)-COLD(I,J-1))/
     &    (DX(I-1)*DY(J-1))
      GOTO 500
C
C  Quadrant 2.
C
200   CONTINUE
      BET =(COLD(I+1,J)-COLD(I,J))/DX(I)
      GAM=(COLD(I,J)-COLD(I,J-1))/DY(J-1)
      DEL=(COLD(I,J-1)+COLD(I+1,J)-COLD(I,J)-COLD(I+1,J-1))/
     &    (DX(I)*DY(J-1))
      GOTO 500
C
C  Quadrant 3.
C
300   CONTINUE
      BET =(COLD(I,J)-COLD(I-1,J))/DX(I-1)
      GAM=(COLD(I,J+1)-COLD(I,J))/DY(J)
      DEL=(COLD(I-1,J)+COLD(I,J+1)-COLD(I,J)-COLD(I-1,J+1))/
     &    (DX(I-1)*DY(J))
      GOTO 500
C
C  Quadrant 4.
C
400   CONTINUE
      BET =(COLD(I+1,J)-COLD(I,J))/DX(I)
      GAM=(COLD(I,J+1)-COLD(I,J))/DY(J)
      DEL=(COLD(I+1,J+1)+COLD(I,J)-COLD(I+1,J)-COLD(I,J+1))/
     &    (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) - XNODE(I)
      DDY = PSTARY(I,J) - YNODE(J)
C
      CLAG = ALF + BET*DDX + GAM*DDY + DEL*DDX*DDY
C
      RETURN
      END
```

```
*         File: RWCLT2.FOR          Last revision:  August 16, 1989
C
      SUBROUTINE CHMREAD
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
$include:'CSIZE.LT2'
$include:'CCHEM.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
C
      CHARACTER STRING*80
      DOUBLE PRECISION DELTPR, QTEMP
      INTEGER I, J, KKK, NBSOUR
      LOGICAL OKAY
C
C  SET UP FORMATS
C
   10 FORMAT(1X,7F10.4)
 2000 FORMAT(A)
 2300 FORMAT(5X,A)
C
C  R/W TWO HEADER LINES, THEN ENTER RUN CONTROL INFORMATION
C
      WRITE(*,*)' Reading from data file CHMLT2.DAT'
      WRITE(*,*)
C
      WRITE(*,*)' Reading run control data. '
      WRITE(*,*)
C
      READ (3,2000) STRING
      WRITE(4,2300) STRING
      WRITE(6,2300) STRING
C
      READ (3,2000) STRING
      WRITE(4,2300) STRING
      WRITE(6,2300) STRING
C
      READ (3,2000) STRING
      READ(3,*) NPRT,T0,TMAX,DT0
      IF (NPRT .GT. MAXPRT) THEN
         WRITE(*,*) ' *** NPRT exceeds MAXPRT! ***'
         NPRT=MAXPRT
      ENDIF
      DELTPR=(TMAX-T0)/FLOAT(NPRT)
      DO 17, I=1,NPRT
         PRTIME(I)=DELTPR*FLOAT(I) + T0 - DT0*1.0D-2
   17 CONTINUE
```

```
C
      WRITE(*,*)' The print times are as follows:  '
      WRITE(*,*)
      WRITE(*,10)(PRTIME(I),I=1,NPRT)
      WRITE(*,*)
C
C  REMEMBER THE CONTROL FLAGS
C
C*********************************************************************
C
C  NFLAG(1)=0 MEANS: COMPUTE THE HYDRAULIC PRESSURE FIELD ONLY.
C  NFLAG(1)=1 MEANS: COMPUTE THE HYDRAULIC PRESSURE FIELD AND THE
C         VELOCITY COMPONENTS.
C  NFLAG(1)=2 MEANS: COMPUTE THE HYDRAULIC PRESSURE FIELD, THE
C         VELOCITY COMPONENTS, AND THE DISPERSION COMPONENTS,
C         AND THE DYNAMIC CHEMICAL FIELD DISTRIBUTION.
C
C  NFLAG(3)=1 MEANS: DISPLAY THE HYDRAULIC FIELD CONVERGENCE DATA.
C  NFLAG(3)=0 MEANS: SKIP THE ABOVE DISPLAY.
C
C  NFLAG(4)=0 MEANS: MODIFY ADIPRM EVERY NMOD'TH ITERATION, USING
C                    DELTA AND ADIMAX.
C  NFLAG(4)=1 MEANS: READ IN A NEW VALUE FOR ADIPRM EVERY NMOD'TH
C                    ITERATION.
C
C  NFLAG(7)=1 MEANS: WRITE THE COORDINATES OF THE P* POINTS.
C  NFLAG(7)=0 MEANS: DO NOT WRITE THE 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(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 THOMAS ALGORITHM MATRIX ELEMENTS AND
C          THE COMPLETE "KNOWN VECTOR" COMPONENTS, EVERY TIME
C          THROUGH THE ALGORITHM.
C  NFLAG(13)=0 MEANS: DO NOT WRITE ANY THOMAS ALGORITHM MATRIX AND/OR
C          "KNOWN VECTOR" COMPONENTS.
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  NOTE: THE OUTPUTS ENABLED BY NFLAGS 7, 10, 11, 12, 13, AND 14 ARE
C  WRITTEN ON THE FILE 'DEBUGLT2.OUT'.
C
C*********************************************************************
C
C  READ INPUT CHEMICAL PARAMETERS
C
      WRITE(*,*)' Reading chemical parameters.'
      WRITE(*,*)
      READ(3,2000) STRING
      READ(3,*) DL0,KSAND,KCLAY,KORG
C
C  READ IN INLET & EXIT PORT BOUNDARY CHEMICAL CONCENTRATIONS.
C
      WRITE(*,*)' Reading inlet and exit port boundary data.'
      WRITE(*,*)
      READ(3,2000) STRING
      READ(3,*) CIN,COUT,C0
C
C  READ IN INJECTION WELL CHEMICAL CONCENTRATIONS(KG/KG).
C
      WRITE(*,*)' Reading injection wells chem. conc. '
      WRITE(*,*)' Minimum number is one well of strength zero.'
      READ(3,2000) STRING
      READ(3,*) (CHMCON(I),I = 1, ABS(NINJW))
C
C  READ SWITCHING TIMES FOR INJECTION WELLS.  THE 1ST, 3RD, ETC., TIMES
C  ARE "SWITCH ON" TIMES; THE 2ND, 4TH, ETC., TIMES ARE "SWITCH OFF"
C  TIMES.  THERE MUST BE AT LEAST ONE SWITCHING TIME; ITS VALUE MAY
C  EXCEED TMAX (IN WHICH CASE THE WELLS ARE NEVER SWITCHED ON).
C
      WRITE(*,*) ' Reading number of switching times (at least 1) '
      WRITE(*,*)
      READ(3,2000) STRING
      READ(3,*) NSWT
      IF (NSWT .GT. MAXSWT) THEN
         WRITE(*,*) ' *** NSWT exceeds MAXSWT! ***'
      ENDIF
C
      WRITE(*,*) ' Reading the switching times for injection wells.'
      WRITE(*,*) ' Must be in increasing order.'
      WRITE(*,*)
      READ(3,2000) STRING
      READ(3,*) (SWTIME(MIN(I,MAXSWT)),I=1,NSWT)
C  ADD A 'SIGNAL' TIME TO THE ARRAY WHICH IS GREATER THAN TMAX.
      SWTIME(MIN(NSWT,MAXSWT)+1)=TMAX+1.0
```

```
C  THE WELLS ARE INITIALLY OFF:
      WELLON=.FALSE.
C
C  READ IN NUMBER OF BURIED CHEMICAL SOURCES.
C
      READ(3,2000) STRING
      READ(3,*)  NBSOUR
C
      WRITE(*,*)' Reading buried source positions and strengths- '
      WRITE(*,*)' Minimum is one source of strength zero.'
      WRITE(*,*)
      READ (3,2000) STRING
      DO 305, KKK = 1, NBSOUR
         READ(3,*) I, J, QTEMP
         CALL TESTIJ(I,J,QTEMP,OKAY)
         IF (.NOT. OKAY) THEN
            WRITE(*,*) ' Error was found in file CHMLT2.DAT.'
            STOP 1
         ENDIF
         QCHM1S(I,J) = QTEMP
  305 CONTINUE
C
C  READ IN FIRST ORDER LOSS PARAMETERS.
C
      WRITE(*,*)' Reading first order loss parameters.'
      WRITE(*,*)
      READ(3,2000) STRING
      READ(3,*) XLAM1O,XLAMIR,XLAMRA,XSLM1O,XSLMRA
C
C  READ INITIAL CHEMICAL DISTRIBUTION AT TIME ZERO.
C
      WRITE(*,*)' Reading initial chem. distribution. '
      WRITE(*,*)
      READ(3,2000) STRING
      DO 14, J=1,NSLYP1
         READ(3,*) (COLD(I,J),I=1,NSLXP1)
   14 CONTINUE
C
      RETURN
      END
C
C ***************************************
C
      SUBROUTINE CHMOUT
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
$include:'CSIZE.LT2'
$include:'CCHEM.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
```

```
C
      INTEGER I, J
      LOGICAL NONE
C
C  PRINT OUT RUN CONTROL INFORMATION.
C
      WRITE(4,98765)
98765 FORMAT(/,
     &  5X,'** NOTE!  SI units are indicated, but any units  ** ',
     &  /,5X,'** can be used, as long as they are CONSISTENT.  **')
      WRITE(4,3)
    3 FORMAT(/,10X,'RUN CONTROL INFORMATION.',/)
      WRITE(4,4) T0,TMAX,DT0,DT1
    4 FORMAT(10X,'TIMES IN DAYS:',/,
     &   1P,5X,'T0= ',E10.3,2X,'TMAX= ',E10.3,2X,'DT0= ',E10.3,
     &   2X,/,5X,'COMPUTED DTMAX= ',E10.3,/)
      WRITE(4,5) NPRT
    5 FORMAT(5X,'NPRT= ',I5,';  PRINT TIMES (DAYS) ARE:',/)
      CALL PRINT1(4,NPRT,PRTIME)
C
C  PRINT OUT CHEMICAL PARAMETERS.
C
      WRITE(4,300)
  300 FORMAT(/,10X,'CHEMICAL PARAMETERS',/)
      WRITE(4,301) DL0
  301 FORMAT(1P,5X,'DL0= ',E10.3,' (M^2/DAY)',/)
      WRITE(4,304) KSAND,KCLAY,KORG
  304 FORMAT(1P,5X,'KSAND= ',E10.3,' (M^3/KG SAND)',/,
     &          5X,'KCLAY= ',E10.3,' (M^3/KG SILT)',/,
     &          5X,'KORG=  ',E10.3,' (M^3/KG ORGANICS)',/)
C
C  PRINT OUT THE INLET AND EXIT BOUNDARY CHEMICAL CONCENTRATIONS.
C
      WRITE(4,507)CIN,COUT,C0
  507 FORMAT(1P,5X,'CIN=   ',E10.3,' (KG/M^3)',/,
     &          5X,'COUT=  ',E10.3,' (KG/M^3)',/,
     &          5X,'C0=    ',E10.3,' (KG/M^3)',/)
      WRITE(4,*)
      WRITE(4,320)
  320 FORMAT(10X,'TABLE OF FIRST ORDER LOSS PROCESS',
     & ' COEFFICIENTS (1/DAY)',/)
      WRITE(4,321) XLAM1O
  321 FORMAT(5X,'XLAM1O= ',1P,E12.5,/)
      WRITE(4,324) XLAMIR
  324 FORMAT(5X,'XLAMIR= ',1P,E12.5,/)
      WRITE(4,327) XLAMRA
  327 FORMAT(5X,'XLAMRA= ',1P,E12.5,/)
      WRITE(4,329) XSLM1O
  329 FORMAT(5X,'XSLM1O= ',1P,E12.5,/)
      WRITE(4,331) XSLMRA
```

```
  331 FORMAT(5X,'XSLMRA= ',1P,E12.5,/)
C
C  WRITE OUT THE TABLE OF SOURCE CHARACTERIZING PARAMETERS.
C
      WRITE(4,400)
  400 FORMAT(/,5X,'BURIED CHEMICAL SOURCE CONCENTRATION ',
     &   '(KG CHEM/M^3 DAY)',/)
      WRITE(4,401)
  401 FORMAT(5X,'  I   J  QCHM1S(I,J)',/,5X,20('-'))
      NONE=.TRUE.
      DO 402, J=1,NSLYP1
         DO 402, I=1,NSLXP1
            IF (QCHM1S(I,J) .NE. 0.0) THEN
               NONE=.FALSE.
               WRITE(4,340) I,J,QCHM1S(I,J)
            ENDIF
  402 CONTINUE
      IF (NONE) WRITE(4,510)
  510 FORMAT(5X,'NO NON-ZERO BURIED SOURCES.')
C
      WRITE(4,4000)
 4000 FORMAT(//,5X,'INJECTION WELLS CHEMICAL CONCENTRATION ',
     &   '(KG CHEM/KG SOLUTION)',/)
      WRITE(4,4010)
 4010 FORMAT(5X,'  I   J  CSWELN(I,J)',/,5X,20('-'))
      NONE=.TRUE.
      DO 4020, I=1,NINJW
         IF (CHMCON(I) .NE. 0.0D0) THEN
            NONE=.FALSE.
            WRITE(4,340) INJI(I),INJJ(I),CHMCON(I)
         ENDIF
 4020 CONTINUE
      IF (NONE) THEN
         WRITE(4,511)
         NINJW=-ABS(NINJW)
      ENDIF
  511 FORMAT(5X,'NO NON-ZERO INJECTION WELLS.')
C
      WRITE(4,4030)
 4030 FORMAT(//,5X,'SWITCHING TIMES FOR INJECTION WELLS (DAYS)',/)
  340 FORMAT(5X,I3,1X,I3,2X,1P,E10.3)
      CALL PRINT1(4,MIN(NSWT,MAXSWT),SWTIME)
C
C  PRINT OUT INITIAL DISTRIBUTION OF CHEMICAL.
C
      WRITE(4,345)
  345 FORMAT(/,10X,'INITIAL CHEMICAL DISTRIBUTION (KG/M^3)',/)
      CALL PRINT3(4,NSLXP1,NSLYP1,COLD)
C
C  PRINT HEADINGS ON FILE 'CHEMFOUT.OUT'
```

```
C
      WRITE(6,98765)
      WRITE(6,9400)
 9400 FORMAT(/,10X,'OUTPUT DATA FOR CHEMICAL SYSTEM')
C
      RETURN
      END
C
C ***************************************
C
      SUBROUTINE CHMWRT(TIME)
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DOUBLE PRECISION TIME
$include:'CSIZE.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
C
      INTEGER I, J
C
      WRITE(6,6) TIME
    6 FORMAT(/,5X,'TIME T=',F10.4,' (DAYS)',/,
     &   5X,'CUMULATIVE CHEMICAL MASSES (KG)')
      WRITE(6,2500) XMASS,XMFONW,XMSOUR,XMASIN,XMASOT
 2500 FORMAT(1P,/,5X,'XMASS=  ',E12.5,2X,'XMFONW= ',E12.5,2X,
     &               'XMSOUR= ',E12.5,
     &          /,5X,'XMASIN= ',E12.5,2X,'XMASOT= ',E12.5,/)
      IF (NINJW .GT. 0) THEN
         IF (WELLON) THEN
            WRITE(6,2510) 'ON'
         ELSE
            WRITE(6,2510) 'OFF'
         ENDIF
      ENDIF
 2510 FORMAT(5X,'INJECTION WELLS ARE CURRENTLY ',A,/)
      WRITE(6,7)
    7 FORMAT(10X,'CHEMICAL CONCENTRATION DISTRIBUTION (KG/M^3)',/)
C
      CALL PRINT3(6,NSLXP1,NSLYP1,CNEW)
C
      RETURN
      END
```

```
*         File: RWWLT2.FOR          Last revision: August 16, 1989
C
      SUBROUTINE FLOREAD
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
$include:'CSIZE.LT2'
$include:'CCHEM.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
C
      CHARACTER*80 STRING
      DOUBLE PRECISION QTEMP
      INTEGER I, J, KKK
      LOGICAL OKAY
C
C  FORMATS
C
 2000 FORMAT(A)
C
C  ENTER RUN CONTROL INFORMATION
C
      WRITE(*,*)' Reading from data file WATLT2.DAT'
      WRITE(*,*)' Reading run control information.'
      WRITE(*,*)
      READ(1,2000) HEAD1,HEAD2
      READ(1,2000) STRING
      READ(1,*) ADIPRM,NLSOR,TLRNWA,DELTA,ADIMAX,NMOD,ZTHRSH
      ZTHRSH=MIN(ZTHRSH,0.99D0)
C
C  READ IN CONTROL FLAGS
C
C*****************************************************************
C
C  NFLAG(1)=0 MEANS: COMPUTE THE HYDRAULIC PRESSURE FIELD ONLY.
C  NFLAG(1)=1 MEANS: COMPUTE THE HYDRAULIC PRESSURE FIELD AND THE
C         VELOCITY COMPONENTS.
C  NFLAG(1)=2 MEANS: COMPUTE THE HYDRAULIC PRESSURE FIELD, THE
C         VELOCITY COMPONENTS, AND THE DISPERSION COMPONENTS,
C         AND THE DYNAMIC CHEMICAL FIELD DISTRIBUTION.
C
C  NFLAG(3)=1 MEANS: DISPLAY THE HYDRAULIC FIELD CONVERGENCE DATA.
C  NFLAG(3)=0 MEANS: SKIP THE ABOVE DISPLAY.
C
C  NFLAG(4)=0 MEANS: MODIFY ADIPRM EVERY NMOD'TH ITERATION, USING
C                    DELTA AND ADIMAX.
C  NFLAG(4)=1 MEANS: READ IN A NEW VALUE FOR ADIPRM EVERY NMOD'TH
C                    ITERATION.
```

```
C
C  NFLAG(7)=1 MEANS: WRITE THE COORDINATES OF THE P* POINTS.
C  NFLAG(7)=0 MEANS: DO NOT WRITE THE 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(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 THOMAS ALGORITHM MATRIX ELEMENTS AND
C          THE COMPLETE "KNOWN VECTOR" COMPONENTS, EVERY TIME
C          THROUGH THE ALGORITHM.
C  NFLAG(13)=0 MEANS: DO NOT WRITE ANY THOMAS ALGORITHM MATRIX AND/OR
C          "KNOWN VECTOR" COMPONENTS.
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  NOTE: THE OUTPUTS ENABLED BY NFLAGS 7, 10, 11, 12, 13, AND 14 ARE
C  WRITTEN ON THE FILE 'DEBUGLT2.OUT'.
C
C*********************************************************************
C
      READ(1,2000) STRING
      READ(1,*)(NFLAG(I),I=1,20)
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,*) 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
            QCHM1S(I,J) = 0.D0
            HOLD(I,J)   = 0.D0
            COLD(I,J)   = 0.D0
            CSWELN(I,J) = 0.D0
  119    CONTINUE
  118 CONTINUE
C
C  NEXT, READ IN NODAL POSITIONS
C
      READ(1,2000) STRING
      READ(1,*) (XNODE(I),I=1,NSLXP1)
      READ(1,2000) STRING
      READ(1,*) (YNODE(I),I=1,NSLYP1)
C
C  NEXT, READ IN THE AQUIFER VERTICAL THICKNESS (M)
C
      READ(1,2000) STRING
      READ(1,*) XLW
C
C  NEXT, READ IN THE INLET AND EXIT PORT TANK LENGTHS (M)
C
      READ(1,2000) STRING
      READ(1,*) XLYIN,XLYOUT
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
```

```
      WRITE(*,*)' Reading soil particle density and water density.'
      WRITE(*,*)
C
C  READ SOIL PARTICLE DENSITY
C
      READ(1,2000) STRING
      READ(1,*) RHOSND,RHOCLA,RHOORG
C
C  READ WATER DENSITY
C
      READ(1,2000) STRING
      READ(1,*) RHOWAT
C
C  READ IN POROUS MEDIUM CHARACTERIZING PARAMETERS.
C
      WRITE(*,*)' Reading porous medium characterizing parameters such '
      WRITE(*,*)' as hydraulic conductivity, porosity, tortuosity, etc.'
      WRITE(*,*)
      READ(1,2000) STRING
      READ(1,*) KTHSTS,TORT,EPS,PCTSAN,PCTCLA,PCTORG,DISPLX,DISPLY
C
C  READ IN NUMBER OF NON-ZERO INJECTION AND EXTRACTION WELL POSITIONS.
C  A MINIMUM OF ONE IS REQUIRED, BUT ITS STRENGTH MAY BE ZERO.
C
      WRITE(*,*) ' Reading number of non-zero injection and extraction'
      WRITE(*,*) ' well positions, NINJW and NEXTW ...'
      WRITE(*,*)
      READ (1,2000) STRING
      READ (1,*) 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,*) I, J, QTEMP
         CALL TESTIJ(I,J,QTEMP,OKAY)
         IF (.NOT. OKAY) THEN
            WRITE(*,*) ' Error was found in file WATLT2.DAT.'
            STOP 1
         ENDIF
         INJI(KKK) = I
         INJJ(KKK) = J
         QWELIN(I,J) = QTEMP
  121 CONTINUE
C
      READ (1,2000) STRING
      DO 122, KKK = 1, NEXTW
```

```
         READ (1,*) I, J, QTEMP
         CALL TESTIJ(I,J,QTEMP,OKAY)
         IF (.NOT. OKAY) THEN
            WRITE(*,*) ' Error was found in file WATLT2.DAT.'
            STOP 1
         ENDIF
         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,*) HIN,HOUT
      RETURN
      END
C
C ****************************************
C
      SUBROUTINE TESTIJ(I,J,QTEMP,OKAY)
C
C  THIS SUBROUTINE TESTS I AND J TO MAKE SURE THEY ARE THE INDICES OF
C  INTERIOR POINTS.  IF THEY ARE, IT DIVIDES QTEMP BY THE APPROPRIATE
C  VOLUME AND SETS OKAY=TRUE.  IF NOT, IT PRINTS AN ERROR MESSAGE AND
C  SETS OKAY=FALSE.
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.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
      DOUBLE PRECISION QTEMP
      INTEGER I, J
      LOGICAL OKAY
C
      IF (2.LE.I .AND. I.LE.NSLXXX .AND.
     &    2.LE.J .AND. J.LE.NSLYYY) THEN
         OKAY=.TRUE.
         QTEMP=QTEMP/(XLW*(DX(I-1)+DX(I))*(DY(J)+DY(J-1))/4.0D0)
      ELSE
         OKAY=.FALSE.
         WRITE(*,*)' Data error: I=',I,'; J=',J,
     &             ' is not an interior point!'
      ENDIF
      RETURN
      END
C
C ****************************************
```

```
C
      SUBROUTINE FLOOUT
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
$include:'CSIZE.LT2'
$include:'CCHEM.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
C
      INTEGER I, J
      LOGICAL NONE
C
C  WRITE OUT RUN CONTROL INFORMATION.
C
      WRITE(2,2) HEAD1,HEAD2
      WRITE(5,2) 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.  **',/)
      WRITE(2,3) ADIPRM,NLSOR,TLRNWA,DELTA,NMOD,ADIMAX,ZTHRSH
    3 FORMAT(1P,5X,'RUN CONTROL INFORMATION.',/,
     & 5X,'ADIPRM= ',E12.5,2X,'NLSOR= ',I5,2X,'TLRNWA= ',E12.5,/,
     & 5X,'DELTA=  ',E12.5,2X,'NMOD=  ',I5,2X,'ADIMAX= ',E12.5,/,
     & 5X,'ZTHRSH= ',E12.5,/)
      WRITE(2,7)(NFLAG(I),I=1,20)
    7 FORMAT(5X,'NFLAG(I)=',20I3,/)
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)=',E10.3,/,
     &            5X,'LENGTH    OF AQUIFER (M)=',E10.3,/,
     &            5X,'THICKNESS OF AQUIFER (M)=',E10.3,/)
      WRITE(2,20) XLYIN,XLYOUT
   20 FORMAT(1P,5X,'INLET PORT TANK LENGTH (M)= ',E10.3,/,
     &          5X,'EXIT  PORT TANK LENGTH (M)= ',E10.3,/)
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= ',E12.5,2X,'HOUT= ',E12.5)
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= ',E10.3,2X,'RHOCLA= ',E10.3,2X,
     & 'RHOORG=',E10.3,/)
      WRITE(2,53) RHOWAT
   53 FORMAT(1P,5X,'RHOWAT= ',E10.3,/)
      WRITE(2,3095)
 3095 FORMAT(10X,'TABLE OF SOIL PROPERTIES',/)
      WRITE(2,4000) KTHSTS
 4000 FORMAT(1P,5X,'KTHSTS= ',E12.5,' (M/DAY)',/)
      WRITE(2,4010) TORT
 4010 FORMAT(1P,5X,'TORT=   ',E12.5,/)
      WRITE(2,4020) EPS
 4020 FORMAT(1P,5X,'EPS=    ',E12.5,' (M^3/M^3)',/)
      WRITE(2,4030) PCTSAN
 4030 FORMAT(1P,5X,'PCTSAN= ',E12.5,/)
      WRITE(2,4040) PCTCLA
 4040 FORMAT(1P,5X,'PCTCLA= ',E12.5,/)
      WRITE(2,4050) PCTORG
 4050 FORMAT(1P,5X,'PCTORG= ',E12.5,/)
      WRITE(2,4060) DISPLX
 4060 FORMAT(1P,5X,'DISPLX= ',E12.5,' (M)',/)
      WRITE(2,4070) DISPLY
 4070 FORMAT(1P,5X,'DISPLY= ',E12.5,' (M)',/)
C
      WRITE(2,210)
  210 FORMAT(/,5X,'TABLE OF INJECTION WELLS (KG WATER/M^3 DAY)',/)
      WRITE(2,211)
  211 FORMAT(5X,'  I   J  QWELIN(I,J)',/,5X,20('-'))
      NONE=.TRUE.
      DO 212, I=1,NINJW
         IF (QWELIN(INJI(I),INJJ(I)) .NE. 0.0D0) THEN
            NONE=.FALSE.
            WRITE(2,509) INJI(I),INJJ(I),QWELIN(INJI(I),INJJ(I))
```

```
         ENDIF
  212 CONTINUE
      IF (NONE) THEN
         WRITE(2,510) 'INJECTION'
         NINJW=-NINJW
      ENDIF
C
      WRITE(2,215)
  215 FORMAT(//,5X,'TABLE OF EXTRACTION WELLS (KG WATER/M^3 DAY)',/)
      WRITE(2,213)
  213 FORMAT(5X,'  I   J  QWELOT(I,J)',/,5X,20('-'))
      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) I,J,QWELOT(I,J)
            ENDIF
  214 CONTINUE
      IF (NONE) WRITE(2,510) 'EXTRACTION'
C
      WRITE(5,98765)
      WRITE(5,120)
  120 FORMAT(5X,' OUTPUT DATA FOR FLOW SYSTEM ',/)
C
      RETURN
  509 FORMAT(5X,I3,1X,I3,2X,1P,E12.5)
  510 FORMAT(5X,'NO NON-ZERO ',A,' WELLS.')
      END
C
C ****************************************
C
      SUBROUTINE FLOWRT
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
$include:'CSIZE.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
$include:'CTHOS.LT2'
C
      INTEGER I, J
C
      WRITE(5,6)
    6 FORMAT(10X,' HYDRAULIC PRESSURE FIELD (M WATER)',/)
      CALL PRINT3(5,NSLXP1,NSLYP1,HNEW)
C
      IF(NFLAG(1).EQ.0) GO TO 501
C
      WRITE(5,7)
    7 FORMAT(/,10X,'DARCY VELOCITY FIELD',/,
```

```
     &   5X,'(MAGNITUDE IN M/D,ANGLE IN DEGREES)',/)
      CALL PRINT4(5,NSLXP1,NSLYP1,VLXX,VLYY)
C
  501 RETURN
      END
```

```
*         File: TCMLT2.FOR          Last revision: June 7, 1989
C
      SUBROUTINE TCMLT2
C
C  CALCULATE THE TOTAL CHEMICAL MASS IN AQUIFER, CUMULATIVE FIRST ORDER
C  LOSSES, AND THE CUMULATIVE ZERO ORDER BURIED SOURCES CONTRIBUTION.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
$include:'CSIZE.LT2'
$include:'CCHEM.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
$include:'CTHOS.LT2'
C
      INTEGER I, J
      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
      ERTEMP=EPS*(1.0D0+RETARD)
      SUMM1=0.0D0
      SUMM2=0.0D0
      SUMM3=0.0D0
C
      DO 1000, I=1,NSLXXX
      DO 1001, J=1,NSLYYY
      F1=EPS*(1.0D0+RETARD)*CNEW(I,J)
      F2=EPS*(1.0D0+RETARD)*CNEW(I+1,J)
      F3=EPS*(1.0D0+RETARD)*CNEW(I,J+1)
      F4=EPS*(1.0D0+RETARD)*CNEW(I+1,J+1)
      F5N=EPS*LAMDA*CNEW(I,J)
      F6N=EPS*LAMDA*CNEW(I+1,J)
      F7N=EPS*LAMDA*CNEW(I,J+1)
      F8N=EPS*LAMDA*CNEW(I+1,J+1)
      F5O=EPS*LAMDA*COLD(I,J)
```

```
      F6O=EPS*LAMDA*COLD(I+1,J)
      F7O=EPS*LAMDA*COLD(I,J+1)
      F8O=EPS*LAMDA*COLD(I+1,J+1)
      FS1=QCHM1S(I,J)
      FS2=QCHM1S(I+1,J)
      FS3=QCHM1S(I,J+1)
      FS4=QCHM1S(I+1,J+1)
      FS5=QWELIN(I,J)*CSWELN(I,J)
      FS6=QWELIN(I+1,J)*CSWELN(I+1,J)
      FS7=QWELIN(I,J+1)*CSWELN(I,J+1)
      FS8=QWELIN(I+1,J+1)*CSWELN(I+1,J+1)
      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
 1001 CONTINUE
 1000 CONTINUE
C
      XMASS=XLW*SUMM1
      XMFONW=XMFONW+XLW*DT0*SUMM2/2.0D0
      XMSOUR=XMSOUR+XLW*DT0*SUMM3
C
      SUMM4=0.0D0
      SUMM5=0.0D0
      SUMM6=0.0D0
      SUMM7=0.0D0
      DO 1010, I=1,NSLXP1
      CFLXIO(I)=-ERTEMP*DCHLY(I,1)*(COLD(I,2)-
     &   COLD(I,1))/DY(1)+
     &   VLYY(I,1)*ERTEMP*COLD(I,1)
      CFLXIN(I)=-ERTEMP*DCHLY(I,1)*(CNEW(I,2)-
     &   CNEW(I,1))/DY(1)+
     &   VLYY(I,1)*ERTEMP*CNEW(I,1)
      CFLXOO(I)=-ERTEMP*DCHLY(I,NSLYP1)*(COLD(I,NSLYP1)-
     &   COLD(I,NSLYYY))/DY(NSLYYY)+
     &   VLYY(I,NSLYP1)*ERTEMP*COLD(I,NSLYP1)
      CFLXON(I)=-ERTEMP*DCHLY(I,NSLYP1)*(CNEW(I,NSLYP1)-
     &   CNEW(I,NSLYYY))/DY(NSLYYY)+
     &   VLYY(I,NSLYP1)*ERTEMP*CNEW(I,NSLYP1)
 1010 CONTINUE
      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
      XMASIN=XMASIN+DT0*(SUMM4+SUMM5)*XLW/2.0D0
      XMASOT=XMASOT+DT0*(SUMM6+SUMM7)*XLW/2.0D0
C
      RETURN
```

```
      END
```

```
*         File: THOMLT2.FOR          Last revision: July 19, 1989
C
      SUBROUTINE THOMLT2
C
C  3-POINT COMPUTATIONAL MOLECULE FOR THE TRIDIAGONAL SYSTEM.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
$include:'CSIZE.LT2'
$include:'CINDX.LT2'
$include:'CTHOS.LT2'
C
      INTEGER J, KKK
      INTEGER NTHOM1, NTHOP1, NTHOP2, NTHOS
      DOUBLE PRECISION CTEMP(IX*IY),DTEMP(IX*IY)
      DOUBLE PRECISION DENOM
C
      NTHOS=NSLXM1*NSLYM1
      NTHOP1=NTHOS+1
      NTHOP2=NTHOS+2
      NTHOM1=NTHOS-1
      IF(NFLAG(13).EQ.0) GO TO 8
C+++++++++++++++++++++++++++++++++++
      WRITE(9,*)
      WRITE(9,*) 'Thomas algorithm matrix & known vector'
      WRITE(9,*)
      DO 5, KKK=2,NTHOP1
    5 WRITE(9,6)KKK,ATDT(KKK),BTDT(KKK),CTDT(KKK),YTDUM(KKK)
    6 FORMAT(1X,I5,4E12.6)
C+++++++++++++++++++++++++++++++++++
C
    8 CTEMP(2)=CTDT(2)/BTDT(2)
      DTEMP(2)=YTDUM(2)/BTDT(2)
      DO 10, J=3,NTHOP1
      DENOM=BTDT(J)-ATDT(J)*CTEMP(J-1)
      CTEMP(J)=CTDT(J)/DENOM
      DTEMP(J)=(YTDUM(J)-ATDT(J)*DTEMP(J-1))/DENOM
   10 CONTINUE
C
      XTDUM(NTHOP1)=DTEMP(NTHOP1)
C
      DO 20, J=1,NTHOM1
      XTDUM(NTHOP1-J)=DTEMP(NTHOP1-J)-CTEMP(NTHOP1-J)*XTDUM(NTHOP2-J)
   20 CONTINUE
C
      RETURN
```

```
      END
```

```
*         File: WATLT2.FOR           Last revision:  July 19, 1989
C
      SUBROUTINE FLUID
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
$include:'CSIZE.LT2'
$include:'CINDX.LT2'
$include:'CMISC.LT2'
$include:'CADII.LT2'
$include:'CTHOS.LT2'
C
      INTEGER I, J
      DOUBLE PRECISION CONST1, CONST2, CONST3
      DOUBLE PRECISION CONST4, CONST5, CONST6
C
C  INITIALIZE THE THOMAS ALGORITHM VARIABLES
C
      DO 10, I=2,IX*IY
      ATDT(I)=0.0D0
      BTDT(I)=0.0D0
      CTDT(I)=0.0D0
      YTDUM(I)=0.0D0
   10 CONTINUE
C
      DO 15, I = 1, NSLXP1
      DO 15, J = 1, NSLYP1
           HOLD(I,J) = 0.D0
           HNEW(I,J) = 0.D0
   15 CONTINUE
C
C     DEFINE MATRIX ELEMENTS FOR WATER PRESSURE FIELD MATRIX.
C
      DO 30, I=2,NSLXXX
      CONST2=2.0D0/(DX(I-1)+DX(I))
      DO 31, J=2,NSLYYY
      CONST1=2.0D0/(DY(J-1)+DY(J))
C
      ADT1(I,J)=-KTHSTS*CONST1/DY(J-1)
      ADT2Y(I,J)=CONST1*(KTHSTS/DY(J-1)+KTHSTS/DY(J))
      ADT2X(I,J)=CONST2*(KTHSTS/DX(I-1)+KTHSTS/DX(I))
      ADT3(I,J)=-KTHSTS*CONST1/DY(J)
      ALT1(I,J)=0.0
      ALT2(I,J)=-KTHSTS*CONST2/DX(I-1)
      ALT3(I,J)=0.0
      AUT1(I,J)=0.0
```

```
      AUT2(I,J)=-KTHSTS*CONST2/DX(I)
      AUT3(I,J)=0.0
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 CONST3=(2.0D0+(DX(2)/DX(1))+(DX(1)/DX(2)))/(2.0D0+DX(2)/DX(1))
      CONST4=(DX(1)/DX(2))/(2.0D0+DX(2)/DX(1))
      CONST5=(2.0D0+(DX(NSLXXX)/DX(NSLXM1))+(DX(NSLXM1)/DX(NSLXXX)))/
     &      (2.0D0+(DX(NSLXM1)/DX(NSLXXX)))
      CONST6=(DX(NSLXXX)/DX(NSLXM1))/
     &      (2.0D0+(DX(NSLXM1)/DX(NSLXXX)))
C
      DO 20, J=2,NSLYYY
      ADT2X(2,J)=ADT2X(2,J)+ALT2(2,J)*CONST3
      AUT2(2,J)=AUT2(2,J)-ALT2(2,J)*CONST4
      ADT2X(NSLXXX,J)=ADT2X(NSLXXX,J)+AUT2(NSLXXX,J)*CONST5
      ALT2(NSLXXX,J)=ALT2(NSLXXX,J)-AUT2(NSLXXX,J)*CONST6
   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
C
      YDUM(I,NSLYYY)=(QWELIN(I,NSLYYY)-QWELOT(I,NSLYYY))/RHOWAT
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
      CALL ADILT2
C
      RETURN
      END
```