V-TOUGH — an Enhanced Version of the TOUGH Code for the Thermal and Hydrologic Simulation of Large-Scale Problems in Nuclear Waste Isolation

J.J. Nitao
Earth Sciences Department
Lawrence Livermore National Laboratory

UCID--21954

DE90 010351

November 6, 1989
DRAFT 1.0
QA Level III WBS 1.2.2.2.2
Activity I-20-1

V-TOUGH — an Enhanced Version of the TOUGH Code for the Thermal and Hydrologic Simulation of Large-Scale Problems in Nuclear Waste Isolation

J.J. Nitao
Earth Sciences Department
Lawrence Livermore National Laboratory

Abstract

The TOUGH code developed at Lawrence Berkeley Laboratory (LBL) is being extensively used to numerically simulate the thermal and hydrologic environment around nuclear waste packages in the unsaturated zone for the Yucca Mountain Project. At the Lawrence Livermore National Laboratory (LLNL) we have rewritten approximately 80 percent of the TOUGH code to increase its speed and incorporate new options. The geometry of many problems requires large numbers of computational elements in order to realistically model detailed physical phenomena, and, as a result, large amounts of computer time are needed. In order to increase the speed of the code we have incorporated fast linear equation solvers, vectorization of substantial portions of code, improved automatic time stepping, and implementation of table look-up for the steam table properties. These enhancements have increased the speed of the code for typical problems by a factor of 20 on the Cray 2 computer. In addition to the increase in computational efficiency we have added several options: vapor pressure lowering; equivalent continuum treatment of fractures; energy and material volumetric, mass and flux accounting; and Stefan-Boltzmann radiative heat transfer.

Contents

1 Introduction

Yucca Mountain, Nevada, is currently being investigated by the Department of Energy as a proposed repository site for the storage of high-level nuclear waste. Investigations at Lawrence Livermore National Laboratory include the numerical modelling of the thermal and fluid environment around the waste package in conjunction with laboratory and field experiments. The TOUGH code, which was originally developed at Lawrence Berkeley Laboratory [Preuss,1987], is one of the codes being used to simulate the thermal and hydrologic environment around the nuclear waste packages. These simulations involve the transport of water, air, and heat in the porous fractured rock found at the site. The reader is referred to a previous report [Nitao, 1988] for references related to the work in this area. In this report we wish to describe the extensive modifications that have made to the code. Overall, about 80 percent of the code has been rewritten, enough, we feel, to justify calling it V-TOUGH, which stands for Vectorized TOUGH.

The original LBL TOUGH code is very modular and "cleanly" coded which is unusual for a large simulation code. We have attempted to continue this practice in our enhanced version. The new version runs about 20 times faster than the original version for the near-field nuclear waste simulation problem described previously in [Nitao,1988].

2 Vectorization

Most supercomputers, such as the Cray, Convex, and Alliant computers, have special hardware that enables the very fast execution of arithmetic operations between arrays, or vectors, of numbers. In order to take advantage of this capability the code must be vectorized: writ-

ten such that, as much as possible, the longest do loops are inner and no data dependencies occur from one pass of the loop to the other passes. Optimizing the code in such a manner may require the use of more physical memory since in vectorization there is an advantage to storing quantities in arrays. Thus, there is some trade-off between memory usage and speed, which may be one of the reasons why the original code was not highly vectorized. The vectorization that we have performed may not have been feasible several years ago when the added memory was not available

In this section we do not consider the vectorization involved in the linear equation solver; this aspect will be covered in a later subsection devoted to that topic. The primary areas where vectorization was performed is in the subroutine MULTI which generates the Jacobian matrix, EOS which calls the PVT subroutines, and in the various PVT subroutines themselves. These routines were all completely rewritten in order to vectorize as many loops as possible. Vectorization often requires, as alluded to before, more physical memory usage because of the need for temporary arrays. In order to reduce the amount of memory we took advantage of the fact that most modern compilers re-use the memory of local variables (unless the variable is declared to be static by the FORTRAN SAVE command) by placing it back onto to a "heap" after return from the subroutine. Thus, as much as possible, arrays were not placed in COMMON but kept as local variables. Although much of the PVT and constitutive type computations were highly vectorized, with regards to the characteristic curves, only the equivalent continuum characteristic curves are vectorized; but, with the new code structure it would not be difficult to vectorize the other characteristic curves as well.

3 Steam Table Look-up

The steam table properties include the mass density and specific internal energy of subcooled and superheated water as well as the saturation pressure. In the formulation used by the code, the density and internal energy are expressed as functions of pressure and temperature while the saturation pressure is computed as a function of temperature. The original code used complex functional fits to compute the steam table values. The number of arithmetic operations required by these formulae are very large and can, therefore, use a significant amount of computer time. We have, therefore, replaced them by an efficient table look-up scheme. Table look-up applied to steam tables is somewhat complicated by the fact that range of interest for the pressure varies with temperature and the fact that the properties of water vary at different rates above and below the saturation line so that it is difficult to maintain a constant pressure interval without generating an excessively large table. (The advantage of using tables for the dependent variable having constant intervals will be seen later.) In the method that we used, we first generate the table for the saturation pressure function at every 2^o C intervals from 5^o to 501^o C (these limits as well as others can easily be changed). The tables for the density and internal energy of subcooled water are then generated for different pressures at these same temperature points. The pressure intervals are constant and start from the saturation pressure at the temperature point and increase to a maximum pressure which we took to be 24.7 MPa. It was found that only three pressure intervals (four pressure points) were sufficient to give good accuracy due to the fact that the properties of subcooled water are almost linear in temperature and pressure. A similar procedure is used to generate the superheated steam properties with the pressure points

beginning at the saturation pressure and decreasing at constant intervals to a minimum pressure of 100 Pa. (In the look-up stage, the ideal gas law is used for pressures below this minimum.) Thirty pressure intervals are used for the superheated water properties.

The table look-up algorithm is now described. Suppose we are interested in the determining the internal energy u of superheated water at some temperature and pressure (T_0, P_0). We first show how to find the saturation pressure $P_{sat}(T_0)$ at T_0. If ΔT is the temperature interval the table index i is defined by

$$i = T_0/\Delta T \tag{1}$$

and is used to linearly interpolated between the table values $Ptab_{sat}(i)$ and $Ptab_{sat}(i+1)$. We now must find the table position j for the pressure given by

$$j = (P_0 - P_{sat}(T_0))/\Delta P_i \tag{2}$$

where ΔP_i is the constant pressure interval for the i-th temperature position. The value of u is now determined by a two dimensional interpolation between the table values of u at (i,j), $(i+1,j)$, $(i,j+1)$, and $(i+1,j+1)$. Linear interpolation is performed first with respect to the j index, first at i and then at $i+1$. The two values that result are then linearly interpolated. This procedure can be performed very efficiently since the use of constant intervals eliminates the need to search through the table and is very easily vectorized.

The arrays holding the tables must be initialized at the beginning of each run either by computing the values from subroutines that were in the original code or, if it exists, the table is read in from an unformatted file called TABLE that was created from a previous run.

4 Linear Equation Solvers

At each time step the model needs to solve a set of non-linear equations for the primary variables. The Newton-Raphson algorithm which is used to solve the equations is an iterative procedure requiring the solution of a system of linear equations at each iteration. There can be several thousand computational elements and each element has associated with it three equations. Therefore, the linear equation solution can consume a significant amount of time.

The enhanced model has three separate user options for the linear equation solver to be used: Option (0), block-banded gaussian elimination with no pivoting; Option (1), standard banded gaussian elimination with partial pivoting from the Linpack library; and Option (2), a Cray 2 highly optimized banded gaussian elimination with partial pivoting [White,1988]. Note that the MA28 solver used in the original code is not available. Option (0) was the solver used to run the problem described in [Nitao,1988]. The other two options were added to the code later. On the Cray 2, option (1) with the Linpack routines optimized for the Cray 2 is about twice as fast as option (0) while option (2) is about 5 times faster. Thus, option (2) is the preferred method on the Cray 2. Option (1) is included because optimized versions of the Linpack routines are available on many machines including the Alliant computer. Option (0) seems to be faster than option (1) when non-optimized Linpack routines are used on such machines as the Sun Workstation. Note that portable FORTRAN source code is available for option (0) and for the non-machine optimized version of option (1), but portions of option (2) is not portable since they are in Cray 2 assembly language.

5 Automatic Time Stepping

The automatic time step option in the original version of TOUGH was based on cutting back the time step when the Newton-Raphson iterations failed to converge. We found that a better algorithm is to control the maximum change in the solution vector from one time step to the next. This control can be accomplished by estimating the time step for the solution to change by a specified amount based on the solution change that ocurred in the previous two time steps. During an iteration, if the change in solution from the previous time step is too large, a re-estimation is performed and the time step started over. This method also controls to some extent the time discretization error.

We now describe the method in more detail. Let u_i be the i-th component of the solution vector at the current iteration of the Newton-Raphson method, and δu_i be equal to u_i minus the i-th component of the solution vector at the end of the previous time step. Now, let $(\delta u_i)_{max}$ be the maximum allowed change in the solution. We define the new reference time step δt by

$$\delta t = \frac{(1+w)(\delta u_i)_{max}}{w(\delta u_i)_{max} + \delta u_i} \tag{3}$$

Here, w is a "damping" factor [Grabowski et al., 1979] that is chosen between 0 to 1 in order to prevent the time step from changing too rapidly. We used a value of 0.8. The value of δt is adjusted to stay within 0.5 to 4.0 times the time step size taken in the previous time step. If the reference time step is less than the current time step because the solution changed too much, then the current time step is replaced by 0.8 times the reference, and the Newton-Raphson is restarted. The factor 0.8 is there to allow for some margin to prevent too many restarts.

6 Other Enhancements

6.1 Equivalent Continuum

Description of and references to the equivalent continuum method for modelling fracture-matrix flow are given in [Nitao,1988], and we assume that the reader is familiar with the theory. Suppose that the the suction pressures for the matrix and fracture are known functions of the liquid saturation, $p_{cm}(s_m)$ and $p_{cm}(s_f)$, respectively. Since the model computes the bulk saturation s_b of the equivalent medium instead of the matrix and fracture saturation, s_m and s_f, these quantities must be solved in terms of the the bulk saturation if we are to compute the characteristic curves of the matrix and fracture. The bulk saturation is defined in terms of s_m and s_f through the relation

$$s_b = \frac{s_f\phi_f + s_m(1-\phi_f)\phi_m}{\phi_f + (1-\phi_f)\phi_m} \tag{4}$$

where ϕ_m is the matrix porosity and ϕ_f is the fracture porosity defined in [Nitao,1988]. The equivalent continuum assumes that the suction pressures are equal

$$p_{cm}(s_m) = p_{cf}(s_f) \tag{5}$$

In our implementation we generate a table of suction head versus bulk saturation. The entry for the suction head is tabulated at exponentially spaced intervals. For each value of suction head we equate both the suction pressure of the matrix and fracture to the given value as assumed by (5). By inverting the saturation vs. suction head relation we may find s_m and s_f in terms of the current head value. Using (4) we compute the bulk saturation s_b. The resulting table is then used during the model run to compute suction pressure as a function of bulk saturation. The composite relative permeability of the equivalent continuum is a linear

combination of the relative permeabilities for the matrix and fractures. This quantity is also tabulated for each entry of suction head since the relative permeabilities can be computed using matrix and fracture saturations expressed as known function of suction head. Since the bulk saturation is known at the same point, the result is a table for equivalent continuum relative permeability versus bulk saturation.

6.2 Vapor Pressure Lowering

Vapor pressure lowering is implemented by lowering the saturated vapor pressure through the Kelvin type law [Marshall and Holmes, 1979]

$$p_v = p_{sat}(T) \exp \frac{-|p_c|}{\rho_l T R} \tag{6}$$

where

$p_{sat}(T)$	saturation pressure of bulk water as function of temperature T
ρ_l	liquid molar density
T	temperature in Kelvin
R	the gas constant

6.3 Radiative Heat Transfer

It is useful in some situations to be able to model radiative heat transfer between computational elements. An example is the heat transfer across the air gap between the waste package and the wall of the borehole. The Stefan-Boltzmann law is used to compute the flux q_R between elements 1 and 2 by the equation

$$q_R = C(T_2^4 - T_1^4) \tag{7}$$

where T_1 and T_2 are absolute temperatures and C is some user supplied and geometry dependent constant.

6.4 Energy and Material Accounting

A new postprocessing capability that has been added to the model is the ability to keep track of the amount in and flow between elements of the energy and mass of various phases and components. In the TOUGH code, computational elements are classified as to their rock type. The phase and component fluxes between the various rocktypes are computed at the frequency of the time step output and saved onto a file. The total phasic and component energy and mass of each rock type is also written to this file. A post processing program is then used after the run to extract the desired data.

7 Discussion

With the new enhancements to the TOUGH code we are able to extend the size and complexity of models describing the near-field thermal and hydrological impact of nuclear waste packages. Large 2D problems with 2000 nodes, such as the one described in [Nitao,1988] can be run in less than one hour on the Cray 2; whereas, it would have required 20 hours before. Small 3D problems are also possible. Further work, will, however, be required to increase the efficiency of the code to handle larger 3D problems. Such problems spend a significant portion of their time solving the system of linear equations in the Newton-Raphson scheme. Therefore, the focus of future work will be on more efficient linear equation solvers.

References

Grabowski, J.W., P.K. Vinsome, R.C. Lin, A. Behie, and B. Rubin, A Fully Implicit General Purpose Finite-Difference Thermal Model for In Situ Combustion and Steam, Society of Petroleum Engineers paper, SPE 8396 (1979). NNA.900312.0192

Marshall, T.J. and J.W. Holmes, *Soil Physics*, Cambridge University Press (1979). NNA.891212.0026

Nitao, J.J., Numerical Modeling of the Thermal and Hydrological Environment around a Nuclear Waste Package using the Equivalent Continuum Approximation: Horizontal Emplacement, Lawrence Livermore National Laboratory Report, UCID-21444 (1988). NN1.880630.0013

Preuss, K., TOUGH User's Guide, Lawrence Berkeley Laboratory Report, LBL-20700, (1987). NNA.900312.0190

White, S., A More Robust Gaussian Elimination/Backsolve Package for the Cray 2, National Magnetic Fusion Energy Computer Center *Buffer*, vol. 12, no. 11 (Nov. 1988). NNA.900312.0191

A Appendix: Technical Summary of Changes to the TOUGH Code

A.1 List of New Features

1. The MA28 matrix solver is replaced by a banded solvers which have higher levels of vectorization. The new solvers are significantly better for rectangular or near rectangular meshes, but may not be optimal for irregular meshes. (Elements need to be ordered by the user to minimize matrix band-width. See the the later subsection which shows how to do this.)

2. The functions that evaluate the steam and water properties have been replaced by a fast table look-up algorithm. A file called TABLE containing table information will be created the first time. It will be used in subsequent runs instead of recomputing the tables.

3. The automatic time stepping has been significantly improved and is based on controlling the maximum changes in the solution vector at each time step.

4. The output files that the program generates have been changed. The names are now derived from a run name which the program requests from the user. This modification enables more than one run to be made at one time.

5. The save file information is written out each time that the output file is.

6. A new output file is created which contains mass and energy balances and fluxes between rock types.

7. The MESH, INCON, and LINEQ files are no longer created.

8. Coefficient generation and equation of state routines have been vectorized.

9. Vapor Pressure Lowering has been implemented.

10. Incorporated Stefan-Boltzmann radiative heat transfer.

11. Incorporated an equivalent continuum treatment of flow in fractured porous media. Multiple rock types can have different equivalent continuum parameters

12. The model was ported to the Unix environment. Through the use of the unix m4 macropreprocessor the model can be configured to run on either the Crays using the LTSS OS or on a Unix machine, e.g. the Sun or Alliant, making code development easier in the future.

13. The model has variable dimensioning using FORTRAN77 parameter statements. Dimensioning is done by modifying a file separate from the source code and using the m4 macro preprocessor.

A.2 Dimensioning and Compiling the Model

Steps in dimensioning and compiling model:

Step 1. Edit or create the file 'dim.h' which should contain the following three lines:

```
define(NELM,68)
define(NCONM,127)
define(NBHMX,6)
```

where the numbers after the resp. symbols represent

NELM must be $\geq$ no. of elements (68 in this example)

NCONM must be $\geq$ no. of connections (127)

NBHMX must be $\geq$ half bandwidth (6)

with option ILOPT=3 described later,

NBHMX must be exactly equal to the half

bandwidth of the system

Step 2. Execute the following command on the cray or sun:

m4 < raw-src-file > final-src-file

(dim.h is read in during this process and must be set with the desired model dimensions)

Step 3. To compile and load on the cray 2 for ILOPT option 1:

civic i=final-src-file,x

For linear equation option (ILOPT) 1

civic i=final-src-file,lib=(omnilib,baselib,fortlib),x

For ILOPT option 2

civic i=final-src-file,lib=(bfactsol,baselib,fortlib),x

Here, bfactsol is the library containing the Cray 2 optimized solver. This option is valid only for the Cray 2.

The above instructions for compiling the program refer to the computer system at the National Magnetic Fusion Energy Center at Lawrence Livermore National Laboratory. On other systems Step 3 should be replaced with system-specific FORTRAN compilation and linking with a library containing Linpack routines for linear equation solvers, or if only option ILOPT=0 is used, it is sufficient to provide dummy subroutine slots to the Linpack routines DGBFA and DGBSL which are not used by this option. In order to compile on the unix system use the "makefile" that is provided in the distribution.

A.3 Ordering the Elements to Minimize Bandwidth

The elements should be ordered in the ELEME data block so that the bandwidth will be minimized. For a rectangular grid the elements should be ordered so that they run in the smallest direction first. The one-half block bandwidth to set for NBHMX will then be the number of elements on the shortest side of the grid. The model prints the half band width value to the main output file immediately after the initialization data is outputted. A larger value of NBHMX can be used and then readjusted to the value in the print out.

A.4 Instructions for Running the Program

The program will prompt for the run name. It then creates the following files by adding the appropriate suffixes to the run name:

'i'	input file
'o'	output file

'v'	save file
'b'	energy and mass file

The run name must be seven characters or less in length.

A.5 New Input Options

The new input options are now described. We follow the same format style as the TOUGH manual [Preuss,1987].

New data block called DTSTP

The original time stepping algorithm in TOUGH is no longer present. The new time stepping algorithm is always active. The Newton-Raphson iteration limit NOITE is still used; time step is cut in half if the limit is reached. The switch MOP(16) is ignored. The algorithm obeys the maximum time step constraints DELTMX as well as the specified output times in the TIMES data block.

DTSTP	automatic time step control parameters; if data block is not present the default values described below are used
	Format (4E10.3)

Card DTSTP.1

DPGMAX max. change in gas phase pressure (default = 8.e5);

setting to zero gives the default

DSGMAX max. change in gas saturation (default = 0.25);

setting to zero gives the default

DTEMPMAX max. change in temperature (default = 20.0);

setting to zero gives the default

DXMAX max. change in mass fraction (default = 0.25);

setting to zero gives the default

Modified PARAM data block

PARAM

Card PARAM.1 Add or change to following:

MOP (1) if ≥ 9 print out automatic time step diagnostic information

if ≥ 8 print out maximum solution changes per time step

if ≥ 4 print out Newton Raphson convergence information

if ≥ 2 print out cpu times of various parts of the

program each time step for the first few steps;

print out where Newton-Raphson failed to convergence

if ≥ 1 print out when time step is reduced

MOP(18) if non-zero, prints out table of values of the characteristic curves at different temperatures for each rock types

New Characteristic Curves Options

The data fields for the equivalent continuum characteristic curve option described in [Nitao, 1988] are

IRP = 9 IRP(1) = absolute permeability of matrix

IRP(2) = van Genuchten alpha parameter for matrix

IRP(3) = van Genuchten beta parameter for matrix

IRP(4) = matrix porosity

IPR(5) = residual liquid saturation for matrix

ICP = 9 ICP(1) = absolute permeability of fracture

ICP(2) = van Genuchten alpha parameter for fracture

ICP(3) = van Genuchten beta parameter for fracture

ICP(4) = fracture porosity

ICP(5) = residual liquid saturation for fracture

Note that both IRP and ICP must be equal to 9. Different parameters can be set for different rock types.

A modified Van Genuchten capillary pressure function is available which does not go to infinity as the liquid saturation approaches the residual value but is smoothly linearly extrapolated based on the slope of the Van Genuchten function at a user-specified saturation point S^*. This option is recommended over option ICP=7 which has a discontinuous slope at the cutoff point where the maximum pressure is attained. The equations defining the new modified version is:

$$\begin{aligned} P_{cap} &= f(S_e^*) + f'(S_e^*)(S_e - S_e^*), \quad S_e <= S_e^* \\ &= f(S_e), \qquad\qquad\quad S_e > S_e^* \end{aligned}$$

where

$$\begin{aligned} S_e &= (S_l - S_{lr})/(S_{ls} - S_{lr}) \\ S_e^* &= (S_l^* - S_{lr})/(S_{ls} - S_{lr}) \\ f(S_e) &= -P_0[|S_e|^{-1/\lambda} - 1]^{1-\lambda} \end{aligned}$$

ICP = 11 ICP(1) = λ

ICP(2) = S_{lr}

ICP(3) = $1/P_0$

ICP(4) = unused

ICP(5) = S_{ls}

ICP(6) = S^*

New data block OPTN

Card OPTN.1

Format (5I5)

ILIMSL, IDSOLC, KNUDSN, IPCTEM, IVPLOW, ILOPT

ILIMSL if non-zero, limits solution changes during each Newton-Raphson inner iteration (recommended value: 0)

IDSOLC if *not zero*, Newton Raphson convergence criteria is based on changes in the primary variables between iterations being less than the tolerance (recommended value: 1)

KNUDSN if non-zero, turns Knudsen diffusion on; this option is not complete and therefore not recommended

IPCTEM if non-zero, turns temperature dependent capillary pressure on, multiplies a factor based on the dependence of surface tension of water (the Leverett capillary curve option ICP=6 which already has the temperature dependence is unaffected by this option)

IVPLOW if non-zero, turns vapor pressure lowering on

ILOPT	option for linear equation solver,		
	if equal to	0,	uses block banded elimination as in previous versions
		1,	calls Linpack banded elimination routines sgbfa and sgbsl or their double precision versions; user needs to supply a local version of these routines
		2,	calls the solver optimized for the Cray 2 by S. White of the National Magnetic Fusion Energy Center

Modification of connection data format

Each connection data line in the CONNE data block has a new field, and each line is followed by an optional second line. For each connection data, columns 71-75 holds a flag ICONT. If ICONT is non-zero then another line of data is read in. (The user should set ICONT to 1 if a non-zero value is desired in case. this flag is later used to denote the number of extra data lines following the current data line.) This added line has the following format:

2e10.3

and contains the values for the following parameters:

RADCOEF coefficient for radiative transfer defined by defined by

$$Q = RADCOEF * AREA * (T2**4 - T1**4)$$

where Q is the radiative heat flux (W.) T1,T2 are absolute temperatures (K.) and AREA is flux area (M**2) set in the first line of the connection data

CONDCOEF coefficient multiplying the computed thermal conductivity for this connection; set to zero or blank if no thermal conduction is desired

If ICONT is set to zero (or blank) this added line is not read in and the value of RADCOEF is set to 0.0 and CONCOEF is set to 1.0.

A.6 Format of Mass and Energy Balance File

First line is header string indicating the number of rock types and the time in seconds.

For each rock type:

(a3,a2) rock type name and no.

(e13.5) total volume of gas phase

(e13.5) total volume of liquid phase

(e13.5) total mass of gas phase

(e13.5) total mass of liquid phase

(e13.5) total mass of water vapor

(e13.5) total mass of air in gas and liquid phase

(e13.5) total mass of water in gas and liquid phase

(e13.5) total energy of gas phase

(e13.5) total energy of liquid phase

(e13.5) total energy of water vapor

(e13.5) total energy of air in gas and liquid phase

(e13.5) total energy of water in gas and liquid phase

(e13.5) total energy of solid rock

Between every pair of rock types*:

(a3,a2,1x,a3,a2) names of the two rock types

(e12.4) mass flux of air in gas phase

(e12.4) mass flux of steam in gas phase

(e12.4) liquid mass flux

(e12.4) fluid convective heat flux

(e12.4) conductive heat flux

(e12.4) cum. mass flux of air in gas phase

(e12.4) cum. mass flux of steam in gas phase

(e12.4) cum. liquid mass flux

(e12.4) cum. fluid convective heat flux

(e12.4) cum. conductive heat flux

* positive flow is from the second rock type to the first rock type