UCRL-ID-122055

Summary of Pre and Post-Processors for V-TOUGH

Stephanie Daveler

Lawrence Livermore National Laboratory
Livermore, CA 94551

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Die

DISCLAIMER

**Portions of this document may be illegible
in electronic image products. Images are
produced from the best available original
document.**

Introduction	1
Preprocessors	1
Input File	1
Sample Input File	3
Sample Output File	3
Known bugs	4
Postprocessors	4
What Is Exttool?	4
Why Another Plotting Package?	4
Input	6
The Time-History File	6
Buffers	7
Ascii Data Files	7
Output	8
SAC Devices	8
Data File	8
Buffer	8
HDF output	8
HDF Image	8
HDF Scientific Data Set	8
Spyglass Ascii	9
Commands	9
Macros	9
3D Output	10

Introduction

This report summarizes the preprocessor utility, LMESH and the postprocessor utility, EXTOOL. These utilities support the input file generation and postprocessing analysis for V-TOUGH¹.

LMESH is a mesh generator for V-TOUGH and generates a rectangular or cylindrical mesh in two dimensions. The format for the LMESH summary is a discussion of input options, followed by a sample problem and output from the sample problem.

EXTOOL is an analysis tool that extracts, manipulates and plots V-TOUGH output. The format for the EXTOOL summary is a discussion of the development of EXTOOL and an overview of the programs capabilities.

Preprocessors

LMESH is used to generate grid and input file information. While this information can be originated by hand, LMESH makes the process easier especially for large models. The output information, which is used by V-TOUGH as input, consists of the ELEME data block, and the CONNE data block. These data blocks are discussed in detail in the *V-TOUGH User's Guide*². The grid information is not used by V-TOUGH. However, it is needed for some postprocessing options such as contour plots and images. LMESH needs an input file in order to run. The format of the input file is discussed below. The program prompts for both the input file name and an output file name. It also generates a file named lmesh.diag for input file debugging purposes.

Input File

LMESH generates a 2-d mesh which is represented using x and z coordinates. The origin of the xz mesh system is always (0,0), with the x and z axes increasing in a positive direction. In cylindrical coordinates the x axis plays the role of the radius axis. Currently, a theta axis is not available.

The input is free format. Carriage returns are ignored except as a terminator for comments. Data can be placed in any column. Any text after the character ! is considered a comment and ignored by the program.

The input file consists of data blocks. Each data block begins with a keyword. A data block can have sub data blocks which also begin with a keyword. Keywords always begin with the character \$. A keyword can be followed by arguments which are numeric data or options in the form of strings. The following is a list of current keywords and their arguments in brackets. The arguments must be given in the order specified.

\$anisotropic	If present will set the ISO parameter in CONNE to 1 for x direction and 2 for z direction; if not present, will set to 1 for both direction as in previous versions.
\$grav [on or off]	Turns gravity on or off, the negative z axis is always down.
\$order [xz or zx]	Ordering of the elements; xz orders by x direction first and zx by z direction first.

-
1. Modeling code. Vectorized TOUGH (transport of unsaturated groundwater and heat).
 2. This is a draft manual that can be requested by sending email to daveler2@llnl.gov

\$thick [float number] The thickness of the model (actually dy).

\$coord [cyl2 or rec2] The type of coordinate system; cyl2 for r-z cylindrical and rec2 for x-z rectangular meshes.

\$dx [integer number] [keyword]

Input information for gridding in x direction; [integer number] is the number of grid steps; [keyword] can be any of the following \$equal, \$subdiv; future options will be available later for exponential, logarithmic, etc. types of gridding; any number of \$dx data blocks can be specified; gridding starts at x = 0.0 and continues in increasing x for each subsequent \$dx data block

\$dy [integer number] [keyword]

Input information for gridding in y direction; same arguments as for \$dx (Future option).

\$dz [integer number] [keyword]

Input information for gridding in z direction; same arguments as for \$dx.

\$equal [real number] [real number]

Keyword to \$dx, \$dy, or \$dz; specifies equal spaced gridding between the first argument and the second; e.g. \$dx 3 \$equal 2. 8 specifies 3 grid subdivisions in the x direction from x = 2. to x = 8. and will result in dx = 2.

\$subdiv [list of real numbers]

Keyword to \$dx, \$dy, \$dz; specifies subgrid subdivision in respective direction as given by the list of numbers.

\$lab [key block] [i1] [i2] [j1] [j2] [k1][k2] [element name] [rock type]
[single or multiple] [volume]

Input data for element information; this data block always appear after the \$dx, \$dy, \$dz data; the model is subdivided into key blocks for accommodating element information.

[key block] key block name

[i1][i2] [j1][j2] [k1][k2]

Range of subgrid indices in the x,y,z directions; specifies the portion of the mesh to set; different \$lab data blocks can have the same key block names in order to have arbitrary shaped key blocks.

[single]

Specifies that the key block will be a single element; subsequent \$lab data blocks with the same key block name will add to this element while still keeping it a single element.

[multiple]

Specifies that the key blocks will be individual elements as specified by the subgrid.

[volume]

if non-zero it will override any element volume computations done by the program.

Sample Input File

An example of an LMESH input file is shown below.

```
! Sample Input, you can put a title here
$grav on
$order xz
$thick 10.
$coord rec2

$dx 5 $equal 0. 5. ! Note that I can put a comment here.
$dz 2 $equal 0. 4.

! the $lab data blocks always appear after $dx, $dy, $dz data

$lab kb1 5 5 1 1 1 2 bnd bndry single 0.
$lab kb2 1 4 1 1 1 2 el rock1 multiple 0.
$lab kb1 3 5 1 1 1 1 bn bndry single 0.
$lab kb3 2 2 1 1 1 2 a test single 0.
$lab kb3 2 4 1 1 2 2 a test single 0.

! ----- This is the end of the sample input-----
```

Sample Output File

Output below was generated by LMESH using the sample input file. The data blocks identified by the strings ELEME and CONNE can be copied to a V-TOUGH input file. The grid information defines where elements are located and can be used for postprocessing.

```
ELEME
el 1 rock1 2.000e+01 x 5.000e-01 y 0.000e+00 z 1.000e+00
a 1 test 8.000e+01 x 1.500e+00 y 0.000e+00 z 1.000e+00
bn 1 bndry 8.000e+01 x 2.500e+00 y 0.000e+00 z 1.000e+00
el 2 rock1 2.000e+01 x 5.000e-01 y 0.000e+00 z 3.000e+00

CONNE
el 1a 1 1 5.000e-01 5.000e-01 2.000e+01 0.000e+00
el 2a 1 1 5.000e-01 5.000e-01 2.000e+01 0.000e+00
a 1bn 1 1 5.000e-01 5.000e-01 2.000e+01 0.000e+00
a 1bn 1 1 5.000e-01 5.000e-01 2.000e+01 0.000e+00
el 1el 2 1 1.000e+00 1.000e+00 1.000e+01 -1.000e+00
bn 1a 1 1 1.000e+00 1.000e+00 2.000e+01 -1.000e+00

GRID
el 1 1 1 1 15.0000e-010.0000e+001.0000e+00
1.000e+00 1.000e+01 2.000e+00
a 1 2 1 1 21.5000e+000.0000e+001.0000e+00
1.000e+00 1.000e+01 2.000e+00
bn 1 3 1 1 32.5000e+000.0000e+001.0000e+00
1.000e+00 1.000e+01 2.000e+00
bn 1 4 1 1 33.5000e+000.0000e+001.0000e+00
1.000e+00 1.000e+01 2.000e+00
bn 1 5 1 1 34.5000e+000.0000e+001.0000e+00
1.000e+00 1.000e+01 2.000e+00
el 2 1 1 2 45.0000e-010.0000e+003.0000e+00
```

```

1.000e+00 1.000e+01 2.000e+00
a 1 2 1 2 21.5000e+000.0000e+003.0000e+00
1.000e+00 1.000e+01 2.000e+00
a 1 3 1 2 22.5000e+000.0000e+003.0000e+00
1.000e+00 1.000e+01 2.000e+00
a 1 4 1 2 23.5000e+000.0000e+003.0000e+00
1.000e+00 1.000e+01 2.000e+00
bn 1 5 1 2 34.5000e+000.0000e+003.0000e+00
1.000e+00 1.000e+01 2.000e+00

```

Known bugs

LMESH does not check for disconnected elements with type 'single'.

Postprocessors

Several post-processors have been used in connection with V-TOUGH. Initially, a sequence of utilities were used to extract and plot V-TOUGH information. This changed in 1991 as a new postprocessor, EXTOOL, was developed. Currently, EXTOOL, is the main postprocessor for the modeling codes V-TOUGH and NUFT¹. In the following sections, a history of V-TOUGH post-processing is discussed along with an overview of EXTOOL. This overview describes some of Extool's capabilities and suggests reasons for using this code instead of another postprocessor. More detailed information on EXTOOL can be found in the *Extool User's Manual* and the *Extool Programmer's Guide*. Both these manuals are drafts, and can be requested by sending email to daveler2@llnl.gov.

What Is Extool?

Extool, which is short for Extractor Tool, is an application that gathers, manipulates, and displays data. It is written in C and Fortran. Extool has a graphical user interface, which is implemented using Xview, and must be run in a Window environment. Currently, it has been compiled and run on Sun workstations running OpenWindows.

Extool was designed to quickly obtain, manipulate, and output data from the modeling code V-TOUGH. It is written specifically for V-TOUGH and not as a general postprocessing code. This makes Extool less flexible than other postprocessors but allows for assumptions that increase the speed and ease of use. Some of Extool's capabilities include extracting data from a special output file written by V-TOUGH or NUFT, performing basic data manipulations, and graphically displaying data. Graphics output can also be generated in PostScript format or as a hierarchical data format (HDF) 8-bit image. Data values can be written to disk in an ascii data file or an HDF scientific data set. Information in HDF format can be transferred to more sophisticated data processors. For repetitive operations, there is a simple macro language that can set window items, press buttons, and issue system commands.

Why Another Plotting Package?

Historically, data from V-TOUGH was postprocessed using a number of programs. Because

1. Modeling code. (Nonisothermal Unsaturated-Saturated Flow and Transport Model).

V-TOUGH creates very large ascii data files, the first program used was CNVBIN. CNVBIN was written to find all data in V-TOUGH's output file and store it in an unformatted file. Next, EXTBIN was written to find specific data in the unformatted file and store it in an ascii file. Two programs were written to read the ascii file and produce a graphics file with either a history plot or a contour plot. Another code prints the graphics file, and another displays it. Figure 1 shows the initial postprocessing setup.

Figure 1. Initial setup.

One problem with this sequence is the number of transitions needed to convert V-TOUGH's output into a format acceptable to other postprocessors. Another problem is flexibility. The user gets history plots and contour plots. This process can also be slow. A commercial plotting package takes care of some of these problems, but not all of them. We decided to change V-TOUGH's output and to manage the postprocessing using Exttool. Data can now flow from the modeling codes V-TOUGH or NUFT as shown in Figure 2.

Exttool becomes a quick way of viewing data from the modeling codes. More sophisticated data transformations are passed off to commercial products.

Figure 2. Current V-TOUGH postprocessing.

Input

Extool was initially designed to extract data from time-history files. To help with data processing, it also accepts several other data formats and can operate on data left in memory. These input options are discussed below.

The Time-History File

In the context of this document, a time-history file has a special meaning. It is an ascii file created by a modeling code. Both V-TOUGH and NUFT have options to output time-history files. The time-history format was designed to store data using as little space as possible. It consists of header information, followed by data values. Based on the header information, it is possible to identify each data value in the file. The time-history option also gives a user control over which data are saved and when. This is necessary because disk space is limited. Computer models generally have some parameters that change quickly and must be saved frequently to observe their behavior. Other parameters change slowly. To save them frequently wastes disk space.

The time-history option gives a user control over what data to save and when to save them. This is done by specifying extraction sets for a modeling run. An extraction set consists of a set of variables, a set of locations, and a triggering mechanism. Figure 3 shows a sample model domain. At each location, the model calculates parameters such as temperature, pressure, and liquid saturation. It also calculates the velocity in which parameters, such as gas, move from one location to another. These calculations are performed to some specified time limit.

Location 1	Location 5	Location 9	Location 13
Location 2	Location 6	Location 10	Location 14
Location 3	Location 7	Location 11	Location 15
Location 4	Location 8	Location 12	Location 16

Figure 3. Model domain.

To capture data for the model domain in Figure 3, a user can define a number of extraction sets. Some examples of extraction sets are shown below. The first example saves everything. This may be too much for a model with a large domain and a long time limit. The second example is more selective.

EXTRACTION SET 1

Variables: Save all parameters.

Locations: All locations.

Trigger: At each time step.

EXTRACTION SET 2

Variables: Save temperature.

Locations: At location 11, location 12, location 15, and location 16.

Trigger: When the liquid saturation at location 1 or location 2 exceeds.5.

Buffers

When Exttool gets data, they are stored internally in a buffer. Normally, this information is removed after Exttool completes the specified operation. A user, however, has the option to keep these data in memory and use them as input to another command.

Ascii Data Files

Exttool also accepts ascii data files in PLTSAC and CONTSAC formats, which are the formats used by the postprocessors shown in Figure 1. The PLTSAC format defines paired data sets, and the CONTSAC format defines gridded data sets. Both data types can be specified using a free format, which is flexible and easy to use, and a fixed format, which expects certain information on certain lines. The details of these formats are described in the *Exttool User's Manual*.

An auxiliary program also converts evenly spaced gridded data from an ascii file to a format accepted by Exttool. The ascii file must have the format x-axis location, y-axis location, data value.

Output

The user has the option to direct Extool command output to different destinations. For history output (paired data), one can select Seismic Analysis Code¹ (SAC) devices, a data file, or buffer. For snapshot data (gridded data), the options are SAC devices, data file, buffer, HDF image, HDF scientific data set, or Spyglass ascii. These options are discussed below.

SAC Devices

The option SAC devices generates graphics output. Graphical output consists of history plots, contour plots, vector plots, and color images. This output can be displayed on a window and written to a disk file. The disk files are unformatted and referred to as SGF files. Extool converts SGF files to PostScript, which is a standard graphics language. An auxiliary program will convert SGF files to encapsulated PostScript. Output from this program has been successfully transferred to FrameMaker, a word processing software.

Data File

Historically, data were stored in PLTSAC and CONTSAC formats. These formats are still used by our group and are output by Extool. (See the section on Ascii Data Files for more information.)

Buffer

(See the section on Buffers for more information.)

HDF output

The hierarchical data format (HDF) was developed at the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign. HDF output is an Extool option and was implemented using a software library developed at NCSA. HDF files are accepted by different software applications and can be converted to other standard formats.

HDF Image

The HDF image output option generates a color image and is a good option for creating movies or videos. This information is written to a file as 8-bit values. The color table is also stored in the file. HDF image output can be created for all time steps in the model. The files can then be played back, using the auxiliary program HDFVIEW. This allows a user to view a specific parameter through time or to visually travel through a three-dimensional (3D) grid. Output from this option has been successfully used to generate a video.²

Another auxiliary program, Make_Movie, allows multiple HDF image files, created by Extool, to be displayed on a single canvas. Using this program, a user can show multiple parameters on the same canvas. Optionally, annotations can be added and a new series of HDF image files created.

HDF Scientific Data Set

The HDF scientific data set is an ascii file. It contains label information, grid dimensions, scales

-
1. Most of the graphics routines in Extool were created by the developers of LLNL's Seismic Analysis Code.
 2. LLNL Open Computing Center Visualization Laboratory.

for both axes, and a data value for each location in the grid. This output has been successfully transferred to the Macintosh application Spyglass.

Spyglass Ascii

Another output option for gridded data is Spyglass ascii. This ascii file contains the location of each grid block and the associated data value. Each line of information describes one grid block. It contains the following values: row distance, column distance, depth, and data value. This data type has not been successfully transferred to another application.

Commands

Exttool performs basic commands on data. It can graphically display history plots, contour plots, grid plots, vector plots, and color images. There are many options that can be set to affect the output of these commands. For more complicated data manipulations, several macro commands are available, as listed below. A special language can also be used to write a script for performing repetitive data processing.

Macros

All of the available macros are listed below followed by a brief description. A more detailed description can be found in the *Exttool User's Guide*.

Crossplot	Creates a scatter plot of two paired data sets based on common first-column data values. If first-column data values do not match, command values are calculated using linear interpolation.
Delta Time	Searches a series of snapshot buffers looking for the amount of time each location (grid block) stays within a specified range.
Dimensionless Liquid Saturation	Uses two snapshot buffers. The first must be the initial liquid saturation of a run. The second is a buffer from a later time. The calculation is defined as: $x = (\text{liquid saturation at time}_n) - (\text{liquid saturation at time}_{\text{init}})$ if $(x > 0.0)$ $\text{results} = x / (1 - \text{liquid saturation at time}_{\text{init}})$ else if $(x < 0.0)$ $\text{results} = x / (\text{liquid saturation at time}_{\text{init}})$ else $\text{results} = 0.0$
Interp Time	Searches through the second column of a paired data set for some occurrence of a value. The corresponding first column value is returned. Linear interpolation is used when necessary.
Line Extremes	Finds the value that is the minimum, maximum, absolute minimum, or absolute maximum in a history buffer (paired data).
Line Tracking	Calculates the average variable value for a series of lines segments, as they

move through time, assuming that water or air is moving through the matrix.

Location Average	Returns a weighted average using data from a set of grid blocks specified by a user.
Macro Processing	Accepts a file with system commands and commands to set and activate window items. Allows for automation of repetitive data manipulations.
Math	Subtracts, multiplies, divides, or adds two data buffers.
Merge	Concatenates paired data sets.
Null Grid	Defines null grid blocks. These blocks are not considered when drawing contour, vector, or image plots. Null areas are left blank and outlined with a dashed line.
Plane Extremes	Finds the value that is the minimum, maximum, absolute minimum, or absolute maximum in a snapshot buffer (gridded data).
Scale	Subtracts an offset from input data, then multiplies the resulting values by a scaling factor.
Slice	Defines a plane of gridded data, then draws a line through the plane. The values along the line are output as paired data (value, distance). The lines and planes must be parallel to some axis.
Sumflux	Defines a plane of gridded data, then draws a line through the plane. The command sums the perpendicular vectors along the line.
3D HDF Output	Calculates an evenly spaced 3D grid.

3D Output

Exttool does not project or transform 3D data. Three-dimensional grids are accepted by Exttool, but the user must define a plane of interest for most Exttool commands. The exception to this is the macro Three-Dimensional HDF Output. This macro creates an evenly spaced 3D grid by interpolating data values. The information is output as an HDF image data or an HDF scientific data set. A scientific data set is the best way to transfer this information to a product that is meant to operate on 3D data.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.