ADVENTURE_Thermal

Steady / Non-steady Heat Conduction Analysis with HDDM

Version: 2.0

User's Manual

March, 2016

ADVENTURE Project

Contents

List of Figures

List of Tables

Preface to the new version

It is a blessing that we have added new functions in the AdvThermal-2.0.
Here are some highlights of the new features in this version.

- A sparse matrix library “libsparse” with various compressed sparse matrix formate.
- Compressed Sparsed Row (CSR) is default sparse matrix storage schemes.
- Balancing Domain Decomposition in the single mode (advthermal-s).
- The CG solver in the single mode (advthermal-s).
- Galerkin Method for unsteady heat analysis
- Options for convection and radiation boundary conditions in makefem_thermal
- Options for all element heat source in makefem_thermal
- New example problems
- Options for writing no result, surface result only and surface-interface result
- A new tool “pfemsolv” to solve ineterior node given the information of surface and interface nodes

1. Introduction

The current document contains information on the ADVENTURE_Thermal finite element analysis solver designed in ADVENTURE Project [1] for analysis of steady and non-steady heat conduction in solid using Hierarchical Domain Decomposition Method with parallel data processing techniques.

1.1. Program Features

ADVENTURE_Thermal has the following features.

- ADVENTURE_Thermal supports the dynamic load distribution of CPUs in parallel computing environments using the Hierarchical Domain Decomposition method (HDDM).
- ADVENTURE_Thermal supports the Balancing Domain Decomposition (BDD)[8] as CG preconditioner for HDDM solver.
- ADVENTUR_Thermal supports different sparse matrix storage schemes using libsparse library.
- ADVENTURE_Thermal supports the single version where all calculations are performed as a single process.
- ADVENTURE_Thermal supports steady and non-steady heat conduction analyses.
- ADVENTURE_Thermal supports linear tetrahedral elements and quadratic tetrahedral elements.
- ADVENTURE_Thermal operates in UNIX and Linux environments.
- ADVENTURE_Thermal uses the Message Passing Interface (MPI) library [6] for parallel data processing.

1.2. Operational Environments

The ADVENTURE_Thermal operates in the following operational environments.

Operating system	Unix, Linux
Data processing library	MPI

1.3. Program Compilation and Installation

To compile the ADVENTUR_Thermal module, you need properly installed MPI environment and ADVENTURE_IO libraries on your computer. The following procedure should be followed to compile the ADVENTURE_Thermal module.

1.3.1 File Extraction from Archive

The necessary data are contained in AdvThermal-2.0.tar.gz. The directories described in subsection 1.3.2 will be created after decompressing the archive file by using the following command.

```
gunzip -c AdvThermal-2.0.tar.gz | tar xvf -
```

1.3.2. Substructures of Directories

After decompressing the AdvThermal-2.0.tar.gz archive file, the directory AdvThermal-2.0 will be created. The contents of AdvThermal-2.0 are shown in the *Table 1*.

Table 1. Contents of Directories

Subdirectory Name	Contents
hddmsrc	Source file of ADVENTURE_Thermal
doc	Documents (Including User's Manual)
tools	Tools for setting up boundary conditions
libfem	Library for finite element method

Except the directories mentioned in *Table 1*, some files will be created in AdvThermal-2.0 directory for auto configuration.

1.3.3. Compilation Method

(1). Install the ADVENTURE_IO module according to its User manual.
(2). Go to the top directory and execute the following command:

```
%  ./configure
%  make
```

After execution of shell script configure, all necessary computing environment will be recorded into the Makefile.

The shell script configure uses the following options. The absolute path to the top directory should be mentioned.

--with-advio=directory
This option is used to define the top directory of ADVENTURE_IO. Default is "$HOME/ADVENTURE".

--with-mpicc=command
This option is used to define the C compiler for MPI. The default is mpicc.

Parallel versions of ADVENTURE_Thermal will not be compiled if the C compiler for MPI is not found.

--prefix=install_dir
This option is used to define the top directory specified by install_dir for program installation. Only the executable modules will be installed in the directory install_dir/bin. The default directory is /$HOME/ADVENTURE.

Other configure options will be described in Chapter 4.

1.3.4 Installation of Executable Module

Execute the command `make install`.

```
% make install
```

The default directory for installation is `$(HOME)/ADVENTURE/`. To change the directory for installation, execute the command

```
% make install prefix=<install_dir>
```

where the option <*install_dir*> should include a full path to the directory for installation.

The following files will be installed.

`bin/advthermal-s`	← Executable module
`bin/advthermal-p`	← Executable module
`bin/advthermal-h`	← Executable module
`bin/makefem_thermal`	← Tool for entire FEA model data
`bin/pfemsolv`	← Interior dof solution tool using surface and interface dof
`doc/AdvThermal/manual-jp.pdf`	← User's Manual in Japanese
`doc/AdvThermal/manual-en.pdf`	← User's Manual in English
`doc/AdvThermal/README.eucJP`	← Brief information in Japanese
`doc/AdvThermal/README`	← Brief information in English
`doc/AdvThermal/copyright`	← Copyright agreement

1.4. Program Execution

The ADVENTURE_Thermal module can be executed in 3 versions. You do not need mpirun to execute the single mode of ADVENTURE_Thermal. The command of execution of 3 versions is described below.

Single mode

`% advthermal-s` *[options] data_dir*

Parallel mode with static job distribution using MPI

`% mpirun [`*options for* `mpirun] advthermal-p [`*options*`]` *data_dir*

Parallel mode with dynamic job distribution using MPI

`% mpirun [`*options for* `mpirun] advthermal-h [`*options*`]` *data_dir*

The options `[`*options for* `mpirun]` are specified for the `mpirun`. The options [*options*] are specified for the ADVENTURE_Thermal executable (see *Section 5.2* of the current manual for details). The option *data_dir* should contain a name of the top directory with data files for analysis (input/output directory).

Necessary options (mpirun)

-np ***n*** : the number of machines (corresponding to the number of parts).
-machinefile ***filename*** **:** The files contain the name of network machines.

Necessary options (advthermal_s or advthermal_p or advthermal_h)

The options of 3 modes of ADVENTURE_Thermal will be described in Section 5.2.

2. Parallel Processing and Analysis Solver

ADVENTURE_Thermal can perform the steady and non-steady heat conduction analyses with dynamic load distribution between CPUs using parallel data processing techniques. These features will be described below.

2.1. Parallel Processing

ADVENTURE_Thermal uses the Hierarchical Domain Decomposition method to provide parallel processing of analysis data. An entire-type model is decomposed in two steps (*Figure 1*) by the ADVENTURE_Metis module prior to execution of ADVENTURE_Thermal. A large decomposed unit of the first hierarchy level refers as *Part*, and smaller units of the decomposed *Part* (2^{nd} hierarchy level) refer as *Subdomains*. The details are given in the User's Manual of the ADVENTURE_Metis module. ADVENTURE_Thermal supports several methods of job distribution to use the CPUs in the most efficient way. The Message Passing Interface (MPI) library is used for parallel data processing. The number of processes started at once depends on user-defined environment.

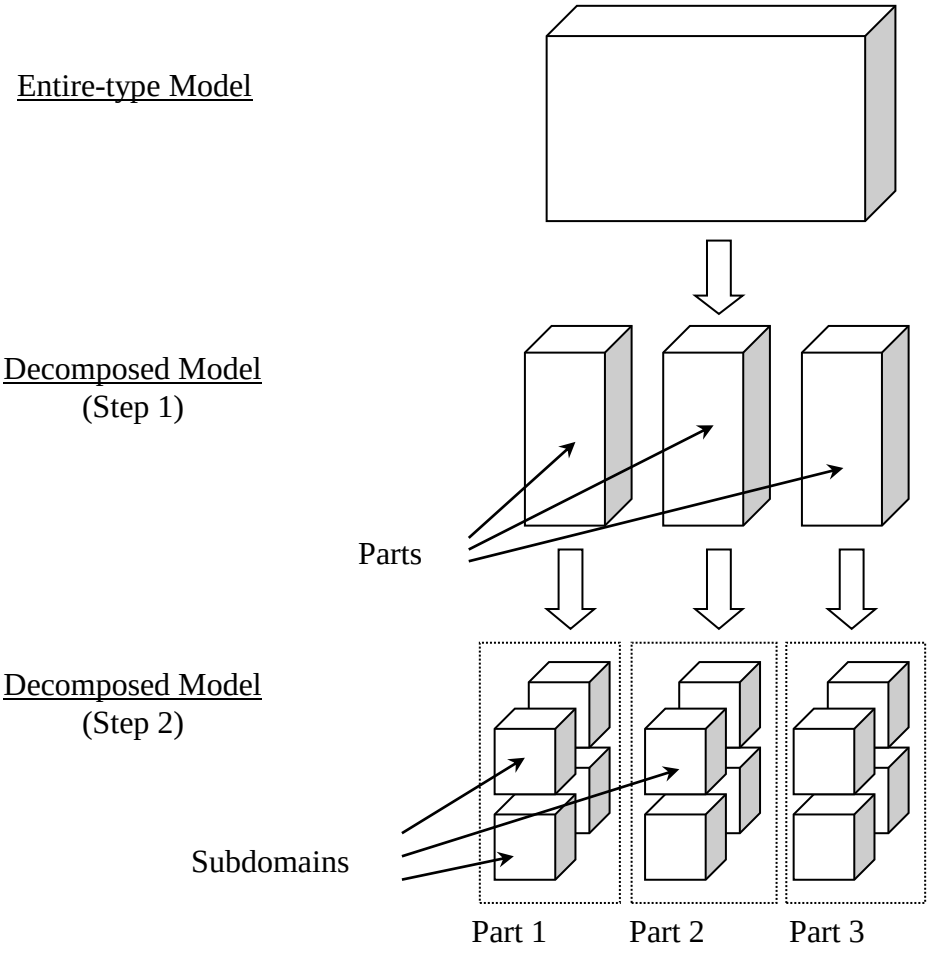

Figure 1. Hierarchical Domain Decomposition

The distributed package contains three versions of ADVENTURE_Thermal.

(1) **Single version (`advthermal-s`)**

A single CPU does all calculations without parallel data processing. The program can be compiled and executed without MPI. There are no limitations on number of "Domains" and "Parts". The model prepared for parallel computation can be used for the single processors without adjustment (*Figure 2*). In the single processor, the computational and data reprocessing procedure for each "Part" occur in the same order as it would be occurred in the parallel computing system. If the parallel computation is not performed well the single version of the program can be used as a checker.

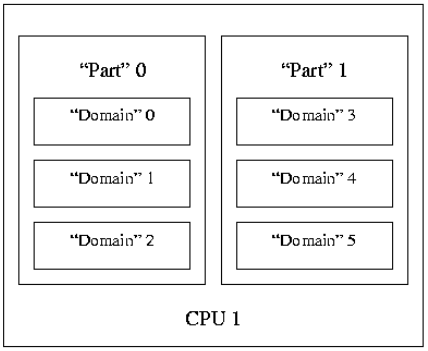

Figure 2. Adjustment of Domain to CPUs (Single version)

(2) **Static job distribution version (`advthermal-p`)**

One CPU treats one *Part* and the processes are statically distributed between CPUs as shown in *Figure 3*. The number of CPUs should correspond to the number of "Parts". This version works efficiently if all nodes have the same performance (uniform system).

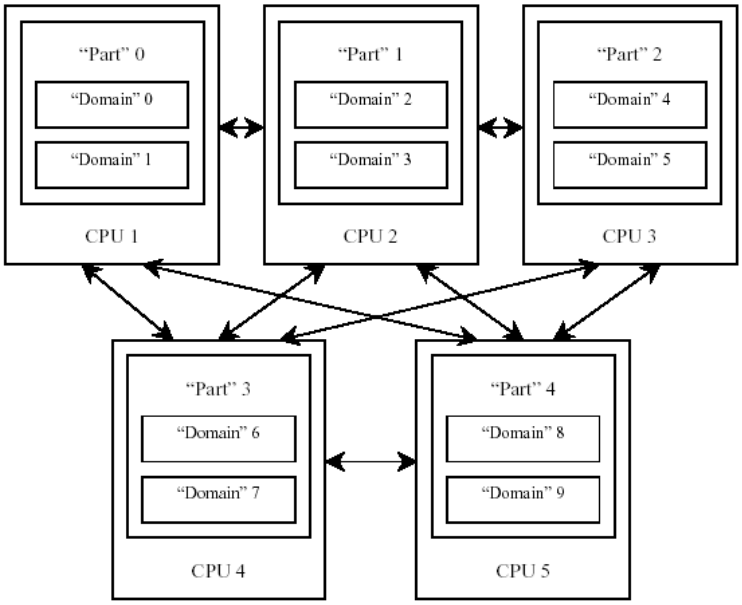

Figure 3. Adjustment of Domains to CPUs (Static load distribution version)

(3) **Dynamic job distribution version (`advthermal-h`)**

The processes are dynamically distributed between CPUs. All CPUs are subdivided into *Parent* CPUs and *Child* CPUs. The *Child* CPUs calculate "Domains" and the *Parent* CPUs collect the calculated information. The number of available CPUs should be more than the number of "Parts". Each "Part" will be assigned to one CPU, and the remained CPUs will be used for calculations of "Domains" (*Figure 4*).

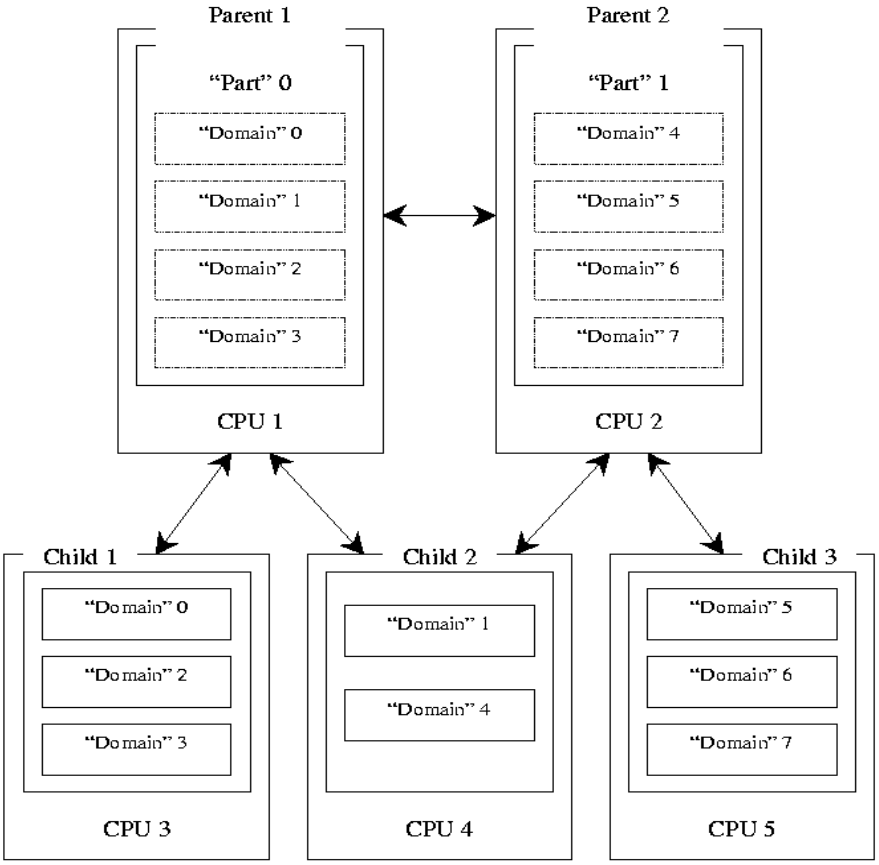

Figure 4. Adjustment of Domains to CPUs (Dynamic load distribution version)

2.2. Characteristics of solver

ADVENTURE_Thermal uses four types of solver to solve the linear equations those come after finite element analysis by using HDDM system. The types of linear equation solver are divided according to the preconditioning techniques used in iterative method.

- HDDM solver:
  This solver does not use any special preconditioner through the iterative method of HDDM system. This solver uses Diagonal Scaling through the iterative method of HDDM system using specific solver option.

- BDD solver:
  In ADVENTURE_Thermal module, direct method is used to obtain the temperature inside "Subdomain" and iterative method is used to obtain the temperature on the boundaries between subdomains.
  This module uses the HDDM system, which satisfies continuity among subdomains through iterative calculations such as Conjugate Gradient (CG) method. So it is absolutely necessary to reduce the number of iterations with a preconditioning technique especially for large problems. This solver uses a powerful CG preconditioner known as Balancing Domain Decomposition (BDD)[8] to meet this need. The BDD is a variation of Neumann-Neumann preconditioner. It solves a "coarse problem" with few degrees of freedom per subdomain in each CG iteration. For heat conductivity analysis [9], this solver uses one degree of freedom per subdomain to construct the coarse matrix. The coarse matrix is solved by parallel LU decomposition. By using this solver, the number of iteration as well as computational time is reduced comparing with HDDM solver.

  In this solver a preconditioning matrix is made in the first CG loop. So a portion of computational time is consumed to make the preconditioner. Some times it is about 15-25% of total computational time. The time per iteration for BDD solver is more than that of HDDM solver. Though time per iteration becomes larger, BDD is an efficient solver as it reduces the total number of iterations. BDD solver needs more memory then HDDM solver. Users have high memory computational environment are suggested to use BDD solver.

  BDD must solve a Neumann-Neumann problem in each iteration. In Neumann-Neumann problem the matrix that represents the subdomain or subproblem may be singular. To overcome this difficulties BDD solver uses a regularization parameter.

- BDD-DIAG solver:
  This solver is similar to BDD. But in this solver it does not need to solve the Neumann-Neumann problem in Balancing Domain Decomposition algorithm. This solver requires less memory than the BDD solver. Depending on the model, it may differ the computational time with that of BDD solver. The users who do not have computational environment with enough memory to use BDD solver are recommended to use BDD-DIAG solver. By testing some model it has been found that the ratio of memory required for BDD-DIAG solver to that of BDD solver is 7/10.

- CG solver:
  This solver uses the Conjugate Gradient method to solve the whole problem. Only advthermal-s supports this solver. This solver is suitable for a single processor. The number of subdomains per part should be one for this solver.

2.3 Sparse Matrix Storage Schemes

In order to take the advantage of the large number of zero elements, special formats are required to store sparse matrices. In this module, for symmetry sparse matrix, only triangular part of the matrix is stored. The main goal is to represent only the non-zero

elements (*nnz*) considering the memory requirements and computation time. Several sparse matrices storage formats are listed below.

- Compressed Sparse Row (CSR)
- Coordinate Storage (COO)
- Diagonal Coordinate Storage (DCOO)
- Compressed Sparse Column (CSC)
- Modified Compressed Sparse Row (MSR)
- Incremental Compressed Sparse Row (ICSR)
- Variable Block Compressed Sparse Row (VBCSR)
- Diagonal Block Compressed Sparse Row (DBCSR)

Details of these storage formats are discussed in the LIBSPARSE library document.

2.4 ADVENTURE_Metis

The computational performance of ADVENTURE_Thermal module depends on the proper domain decomposition using the ADVENTURE_Metis. To execute the ADVENTURE_Metis the number of parts and number of subdomains should be determined before. Basically, the number of "Parts" should be decided based on the method used for parallel processing, the number of nodes used in network, and the computing environments. The number of "Domains" should be decided based on the memory used of computational processes. It has been found that as more detailed domain decomposition is done less memory is required. In case of static job distribution (advthermal-p), good performance can be achieved by using BDD or BDD_DIAG if the number of elements in one domain lies 180 to 370 while in case of dynamic job distribution (advthermal-h) the number of element in one domain lies 350 to 450. This range has been found by investigating some test models. For other models this rang may be semi optimum. The total number of domains does not effect on the number of iterations for BDD and BDD-DIAG solver.

The number of elements in "Domain" that should be created by ADVENTURE_Metis module can be calculated using the following equation.

$$n = N_{element} / (N_{part} * N_{domain})$$

where: *n* is the number of elements in the considered "Domain",
$N_{element}$ is the total number of elements,
N_{part} is the total number of "Parts",
N_{domain} is the total number of "Domains" in the "Parts".

Compared with the static job distribution method, much data transfer accomplished between the "Parent" and the "Child" in case of dynamic job distribution method. The static job distribution method results in better performance for uniform computer environments.

3. Analysis Algorithm

The algorithm of analysis using the ADVENTURE_Thermal module is shown in *Figure 5*.

(1) Creation of mesh data.
Mesh of the entire-type model data are prepared by ADVENTURE_TetMesh.

(2) Setting of boundary conditions.
Boundary conditions are set to mesh using the pre-processor module ADVENTURE_BCtool.

(3) Conversion of analysis model data
The mesh data of the entire model and boundary conditions data are converted to entire FEA model data (adventure format) using the **makefem** tool of ADVENTURE_BCtool. The **makefem_thermal** tool of AdvThermal can also be used for this purpose.

(4) Domain decomposition.
Domain decomposition of the entire-type analysis model is done by ADVENTURE_Metis.

```
% mpirun [mpi_options] adventure_metis -difn 1 [options]
                                    model_filename directory_name div_num
```

The degree-of-freedom used for nodal displacements in static analyses of solids is 3. However, the degree-of-freedom used for temperature in heat conduction analyses should be 1. The necessary option `-difn 1` is used to set the degree-of-freedom for inner boundary nodes to 1.

(5) Heat conduction analysis.
The HDDM-type model data are analyzed by finite element analysis solver ADVENTURE_Thermal.

(6) Visualization of analysis results.
The analysis results can be visualized using ADVENTURE_PostTool or advauto_thermalview.

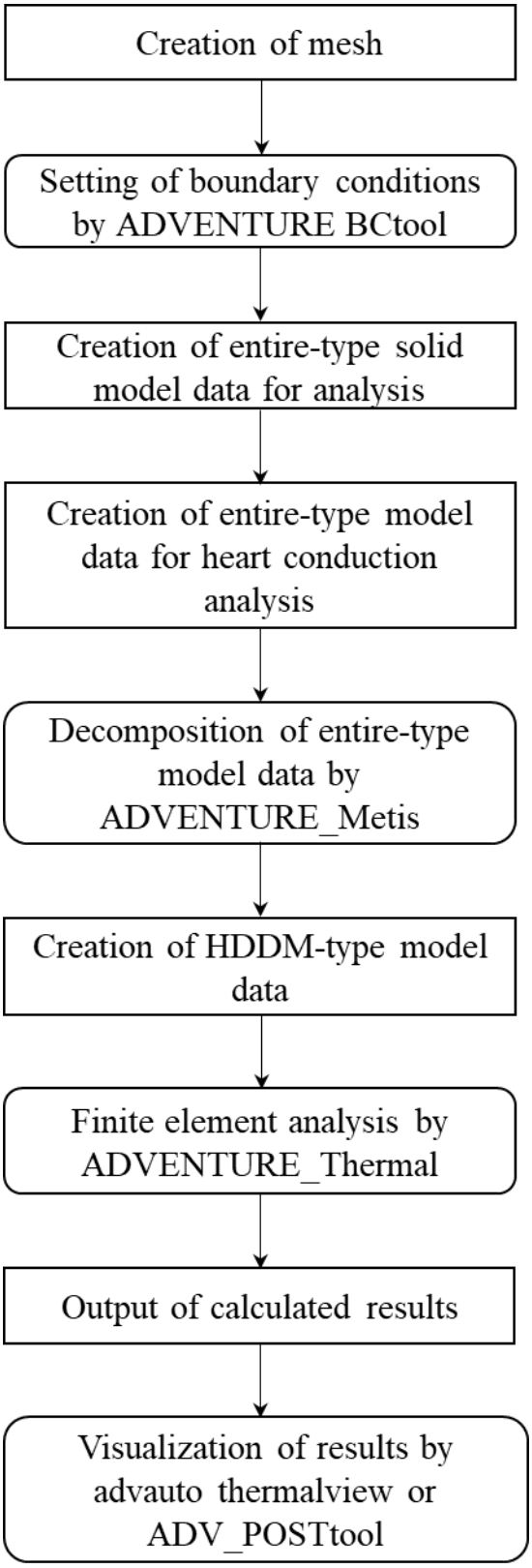

Figure 5. Algorithm of Analysis Using ADVENTURE_Thermal Module.

3.1. Transient Analysis

The backward finite difference approximation and the *Crank-Nicolson* method can be used in transient analyses. The algorithm is shown in *Figure 6*. It includes 2 loops. Time integration iterations are performed by the outer loop and iterative calculations by the CG method based on the hierarchical domain decomposition are performed by the inner loop.

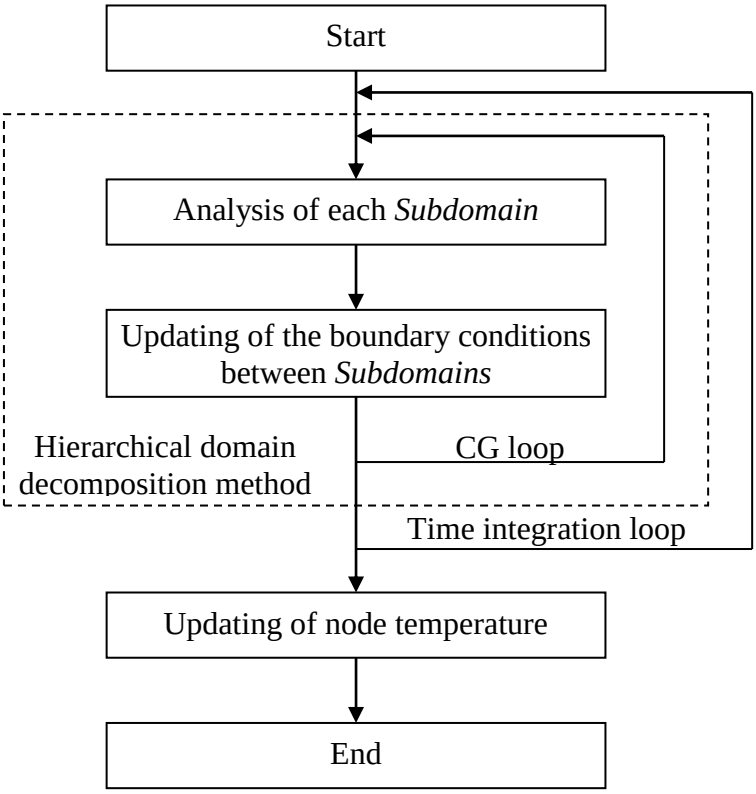

Figure 6. Algorithm of Transient Analysis

3.2. Input / Output Data

The files used by ADVENTURE_Thermal are shown in *Figure 7*. All files, except the job log file, have the binary ADVENTURE format. The data for one *Part* are stored in one file.

The ADVENTURE_Thermal module uses the input HDDM-type model data (hierarchically domain-decomposed data) files prepared by ADVENTURE_Metis. The calculated temperature is stored for all nodes in HDDM-type output data files. The output can be done for each step of time integration. Calculations can be terminated with saving of the data into temporary restart files and restarted using the restart data files. Two kinds of restart files can be used.

1). Restart file for CG loop. Used for steady analyses.
2). Restart file for time integration loop. Used for transient analyses.

If the user uses BDD solver then another restart file for coarse matrix (LU decomposition) can be used.

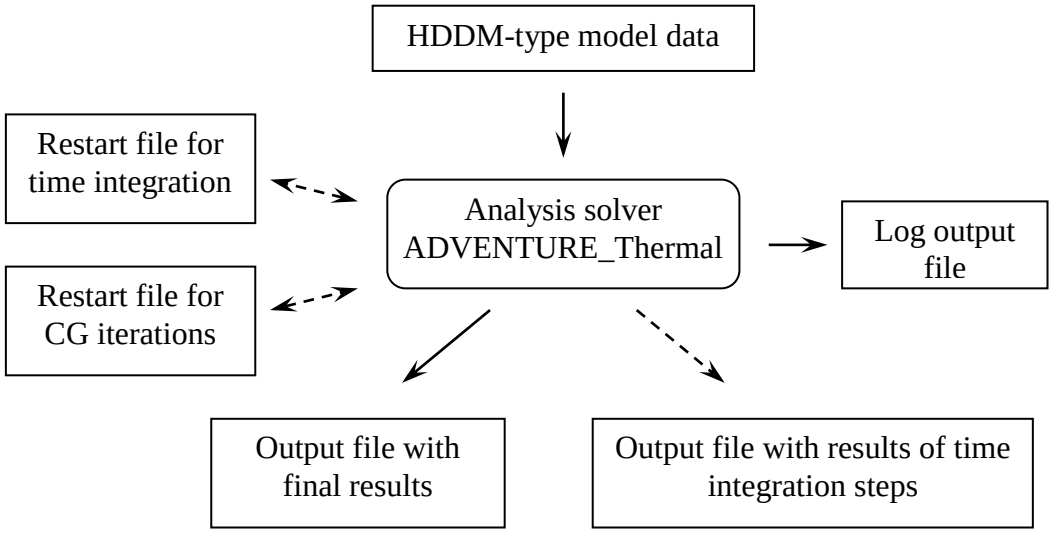

Figure 7. Input and Output Files

3.3. Standard of Temperature Units

The temperature data for ADVENTURE_Thermal must have the unit of degree *Celsius*. Other temperature units are not supported.

3.4. Boundary Conditions

The following boundary conditions can be set.

- Specified temperature (set for nodes)
- Specified heat flux (set for nodes)
- Specified heat convection (set for surface)
- Specified heat radiation (set for surface; only for transient analysis)

The boundary conditions for specified heat radiation can be used only for transient analyses.

3.5. Material Properties

The following isotropic material properties can be specified.

- Thermal conductivity
- Specific heat (data necessary for transient analysis)
- Density (data necessary for transient analysis)
- Stefan-Boltzmann constant (data needed for the specified heat radiation boundary conditions)
- Calorific value

3.6. Output Results

The following values can be saved to files.

- All nodal temperature values
- All nodal flux values
- Surface and inteface nodal tempearture values
- Surface and interface nodal flux values.
- Surface nodal temperature values
- Suface nodal flux values

One file of ADVENTURE binary format contains information on one *Part*.

4. Program Compilation and Installation

4.1. Compile

To compile the ADVENUTRE_Thermal module, you need properly installed MPI environment and ADVENTURE_IO libraries on your computer.
The following procedure should be followed to compile the ADVENTURE_Thermal module:

1. ./configure
2. make

Both of the commands should be executed from the top directory of ADVENTURE_Thermal module. After execution of shell script **configure**, all necessary computing environment will be recorded into the **Makefile.**

The shell script **configure** uses the following options. The absolute path to the top directory should be mentioned.

--with-advio=*directory*
This option is used to define the top directory of ADVENTURE_IO. Default is "$HOME/ADVENTURE".

--with-mpicc=*command*
This option is used to define the C compiler for MPI. The default is **mpicc**.
Parallel versions of ADVENTURE_Thermal will not be compiled if the C compiler for MPI is not found.

--with-mpi-cflags=CFLAGS
The options for C compiler are specified by CFLAGS if the program is compiled for MPI environment. For example, the following statement can be used if it is necessary to specify the include files for MPI.
--with-mpi-cflags="-I/usr/local/include/mpi"
The options specified here by CFLAGS for MPI compiler can be used together with the options for the single version of the program (options for CC compiler)**.**

--with-mpi-libs =LIBS
This option is used to define the MPI links. For example, the following statement can be used to define the MPI libraries.
--with-mpi-libs="-L/usr/local/lib/mpi –lmpi"
The necessary options specified here for MPI link, can be used together with the necessary options for the single version of the program (options for CC compiler).

--enable-optimize
The optimization for compilation is performed. If any other options are required for optimization, the following option should be used.

--enable-optimize=CFLAGS
The optimization for compilation is performed using the options specified by CFLAGS.

--prefix=install_dir
This option is used to define the top directory specified by *install_dir* for program installation. Only the executable modules will be installed in the directory *install_dir*/**bin.** The default directory is /$HOME/ADVENTURE**.**

If the compilation using the supplied configure shell script is failed, the samples of **Makefile** prepared in each subdirectory should be used for compilation. **Makefile.sample** should be copied to **Makefile** in each directory contained **Makefile.sample**. The **Makefile.in.sample** should also be copied to **Makefile.in** in the top directory of ADVENTURE_Thermal module.

The following macros should be changed in the **Makefile.in** in accordance with the concrete computational environment.

ADVSYS_DIR	← Top directory of ADVENTURE system
ADVIO_CONFIG	← Full path to ADVENTURE_IO script *advsys-config*
MPI_CC	← *C* compiler for MPI
MPI_LINKER	← *C* linker for MPI
CC	← *C* compiler
LINKER	← *C* linker
CFLAGS	← Options for optimization

- After changing the Makefile.in, execute the command `make` in the top directory.

```
% make
```

The files in different directory can also be compiled separately by executing make command every time in each directory. In that case, the files located in the **libfem** should be compiled before the files located in the directory **hddmsrc**.

4.2. Installation of Executable Module

- Execute the command `make install`.

```
% make install
```

The default directory for installation is `$(HOME)/ADVENTURE/`. To change the directory for installation, execute the command

```
% make install prefix=<install_dir>
```

where the option <*install_dir*> should include a full path to the directory for installation.

The following files will be installed.

`bin/advthermal-s`	← Executable module
`bin/advthermal-p`	← Executable module
`bin/advthermal-h`	← Executable module
`bin/makefem_thermal`	← Tool for entire FEA model data
`bin/pfemsolv`	← Interior dof solution tool using boundary and interface dof
`doc/AdvThermal/manual-jp.pdf`	← User's Manual in Japanese
`doc/AdvThermal/manual-en.pdf`	← User's Manual in English
`doc/AdvThermal/README.eucJP`	← Brief information in Japanese
`doc/AdvThermal/README`	← Brief information in English
`doc/AdvThermal/copyright`	← Copyright agreement

5. Program Execution

The ADVENTURE_Thermal module can be executed in 3 modes. To execute ADVENTURE_Thermal with `mpirun`, use the following commands.

- Single mode

  `% advthermal-s [`*options*`]` *data_dir*

- Parallel mode with static job distribution using MPI

  `% mpirun [`*options for* `mpirun] advthermal-p [`*options*`]` *data_dir*

- Parallel mode with dynamic job distribution using MPI

  `% mpirun [`*options for* `mpirun] advthermal-h [`*options*`]` *data_dir*

The options `[`*options for* `mpirun]` are specified for the `mpirun`. The options [*options*] are specified for the ADVENTURE_Thermal executable (see *Section 5.2* of the current manual for details). The option *data_dir* should contain a name of the top directory with data files for analysis (input/output directory).

5.1. Names of Input / Output Files

The default names of input and output files are presented below. The files are located under the top directory defined by *data_dir*. Here, *P* indicates the *Part* number and *S* indicates the step number of the time integration loop.

- HDDM-type analysis model file:
  data_dir`/model/advhddm_in_P.adv`

- Analysis results (steady analysis):
  data_dir`/result/advhddm_out_P.adv`

- Analysis results (transient analysis):
  data_dir`/result/advhddm_out_S_P.adv`

- Restart file for CG loop:
  data_dir`/cg-res/advhddm_in_P.adv`

- Restart file for time integration loop:
  data_dir`/result/advhddm_out_S_P.adv`

5.2. Command Options

The following command options can be used.

5.2.1. Options for Transient Analysis

- `-ns` The option is used to execute the transient analysis. It can be used with the following options.
- `--cn` The option specifies that the time integration will be done by the *Crank-Nicolson* method. The default is backward finite difference scheme.
- `--gn` The option specifies that the time integration will be done by *Galerkin* method.
- `--step` *n* The option specifies the maximum number of iterations for the time integration loop. The default number is 10.
- `--out-interval` *n* The option specifies that the output results of each *n* step will be printed. There is no default value; only the results of the last step are printed.
- `--dt` *x* The option specifies the range *x* of time interval. The default value is 1.0.
- `--init` *x* The option specifies the initial temperature *x* for all nodes. The default value is 0 ^{o}C.
- `--use-resin` *n* The option specifies the time integration step *n* from which the analysis will be restarted.

5.2.2. Options Related to Elements

- `-tet10-integ5` The option is used to set 5 integral points for quadratic tetrahedral elements. The default integration is done with 4 integral points.

5.2.3. Options for Iteration Control

ADVENTURE_Thermal uses the CG method to solve the linear equations of stiffness matrixes. The following options can be used to control the iterations by CG method.

- `-cg-tol` *x* The option specifies the tolerance for convergence of iterations. The iterations stop when the relative error (ratio of the current CG residual to the initial CG residual) becomes smaller than the tolerance *x*. The

default value is 1.0×10^{-6}.

- `-cgloop-max` *n* — The option specifies the maximum number of CG iterations. The default value is 1000.
- `-use-cg-resin` — The option specifies from which step the CG restart file will be read to restart the analysis. This option can be used only for steady analyses. No restart file will be read by default.
- `-resout-cglast` — The option specifies that the CG restart file will be created at the last step of CG loop. The file will be created whether the iterations have been converged or the limit number of iterations has been exceeded without convergence. No restart file is created by default.

5.2.4. Options for Sparse Matrix Storage Formats

These options use functions from LIBSPARSE library (details are Appendix D)

- `-keep-kmat-csr` — This option specifies that the subdomain matrix will be stored as CSR format. This is default option.
- `-keep-kmat-coo` — This option specifies that the subdomain matrix will be stored as COO format.
- `-keep-kmat-msr` — This option specifies that the subdomain matrix will be stored as MSR format.
- `-keep-kmat-csc` — This option specifies that the subdomain matrix will be stored as CSC format.
- `-keep-kmat-icsr` — This option specifies that the subdomain matrix will be stored as ICSR format.
- `-keep-kmat-vbcsr` — This option specifies that the subdomain matrix will be stored as VBCSR format.
- `-keep-kmat-dcoo` — This option specifies that the subdomain matrix will be stored as DCOO format.
- `-keep-kmat-dbcsr` — This option specifies that the subdomain matrix will be stored as DBCSR format. This option is used for only ADVENTURE_Solid.
- `-keep-kmat-sky` — This option specifies that the subdomain matrix will be stored as skyline format.

5.2.5. Options for different solvers

ADVENTURE_Thermal uses the BDD solver to solve the linear equations of stiffness matrix. The following options can be used to control the BDD solver

- `-solver cg` — This options specifies that the whole problem will be solved by CG method. (only for advthermal-s).
- `-solver hddm` — This options specifies that a diagonal preconditioner will be used in PCG method in the HDDM system.
- `-solver bdd` — This option specifies that the BDD solver will be used.
- `-solver bdd-diag` — This option specifies that the BDD solver will be used with diagonal scaling in the Neumann-Neumann problem inside the BDD algorithm.
- `-solver bdd -iLU` — This option specifies that the IBDD will be used.
- `-solver bdd-diag -iLU` — This option specifies that IBDD-DIAG [11] will be used.
- -resout-bdd-cmat — This option specifies that the coarse matrix after LU decomposition will be saved in a file for restart. No file is created by default.
- `-use-bdd-cmat` — This option specifies that coarse-matrix will be read from the file to reuse.
- `-bdd-dir` *dir* — This option specifies name *dir* of directory for coarse-matrix input/output data. The default name is **bdd.** This option is used after using the –resout-bdd-cmat option.
- `-bdd-cmat-file` *file* — This option specifies the name of coarse-matrix input/output files to restart. The default is **advhbdd_cmat_*** where '* 'is the processor number. This option is used after using the -resout-bdd-cmat option.
- `-ginv-alpha` *x* — This option specifies the value of factor for alpha-regularization. The default value is 10^{-3}. You can use this option only if you use BDD solver.

5.2.6. Options for Output Filename Specification

Usually, the user should set only the name of the top directory for analysis data. However, the filenames, other than the default filenames, can be specified adding the following options to the command line. Here, *S* is used for the step number of time integration and *P* is the *Part*'s number.

- `-model-file` *file* — The option specifies the name of input data files with analysis model. The characters `_`*P*`.adv` will be added to the filename set by the option *file*. The default filename is `advhddm_in`.
- `-model-dir` *dir* — The option specifies the name *dir* of directory with input data. The default name is `model`.
- `-result-file` *file* — The option specifies the name of output results files. The characters `_`*P*`.adv` (for steady analysis) or `_`*S*`_`*P*`.adv` (for transient analysis) will be added to the filename set by *file*. The default filename is `advhddm_out`.
- `-result-dir` *dir* — The option specifies the name *dir* of directory with output results. The default name is `result`.
- `-ns-resin-file` *file* — The option specifies the filename of input restart files for time integration steps. The characters `_`*S*`_`*P*`.adv` will be added to the filename set by *file*. The default filename is `advhddm_out`.
- `-ns-resin-dir` *dir* — The option specifies the name *dir* of directory with restart files for time integration steps. The default name is `result`.
- `-cg-resin-file` *file* — The option specifies the filename of input restart files for CG steps. The characters `_`*P*`.adv` will be added to the filename set by *file*. The default filename is `advhddm_cgres`.
- `-cg-resin-dir` *dir* — The option specifies the name *dir* of directory with restart files for CG steps. The default name is `cg-res`.
- `-cg-resout-file` *file* — The option specifies the filename of output restart files for CG steps. The characters `_`*P*`.adv` will be added to the filename set by *file*. The default filename is `advhddm_cgres`.

- `-cg-resout-dir` *dir* — The option specifies the name *dir* of directory with restart files for CG steps. The default name is `cg-res`.
- `-result-surface-interface` — The option outputs the external surface and interface results for the hddm system. For this option the advFile (before metis) should be created by makefem_thermal using –with-surface option.
- `-result-surface` — The option outputs the external surface results for the hddm system. For this option the advFile (before metis) should be created by makefem_thermal using –with-surface option.
- `-no-result` — With this options no result will be printed in the result file.

5.2.7. Other Options

- `-file-para` — The option sets the parallel data processing mode. An exclusive data control is used for default mode.
- `-memlimit` *n* — The option specifies the upper limit of memory *n* [in *Mbytes*], which can be used for one process. If this limit is exceeded, the process will be terminated. The default value is 256 [*Mbytes*].
- `-help` or `-h` — These options are used to display the help information.
- `-version` or `-v` — These options are used to display the version of the code.
- `-help-ns` — This option is used to display the help information on possible control options for transient analysis.
- `-help-cg` — This option is used to display the help information on possible control options for CG iterations.
- `-help-bdd` — This option is used to display the help information on possible control options for BDD solver.

Appendix

A. Supported Elements

ADVENTURE_Thermal supports only linear and quadratic tetrahedral elements. However, to set boundary conditions for heat convection and heat radiation, the stiffness matrixes should be created for the boundary surfaces of model. In the case of linear tetrahedral elements, the integration is performed for the linear triangular elements formed from the surface nodes, and in the case of quadratic tetrahedral elements, the integration is performed for the quadratic triangular elements formed from the surface nodes.

A.1. Linear Tetrahedral Element

(1). Nodes. The element contains 4 nodes with connectivity and numbering shown in *Figure 8*.

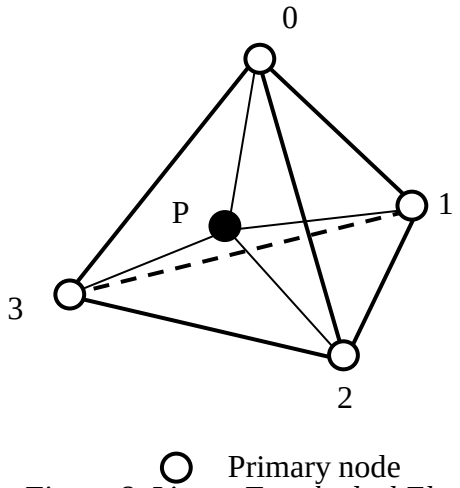

Figure 8. Linear Tetrahedral Element

(2). Integral points. The element has 1 integral point. The integral point *P* has the following volumetric coordinates (L_0, L_1, L_2, L_3).

L_0 = volume of tetrahedron *P123* / volume of tetrahedron *0123* (1)
L_1 = volume of tetrahedron *P023* / volume of tetrahedron *0123* (2)
L_2 = volume of tetrahedron *P013* / volume of tetrahedron *0123* (3)
L_3 = volume of tetrahedron *P012* / volume of tetrahedron *0123* (4)

Table 2. Integral Points of Linear Tetrahedral Element

Integral point number	L_0	L_1	L_2	L_3
0	1/4	1/4	1/4	1/4

A.2. Quadratic Tetrahedral Element

(1). Nodes. The element contains 10 nodes with connectivity and numbering shown in *Figure 9.*

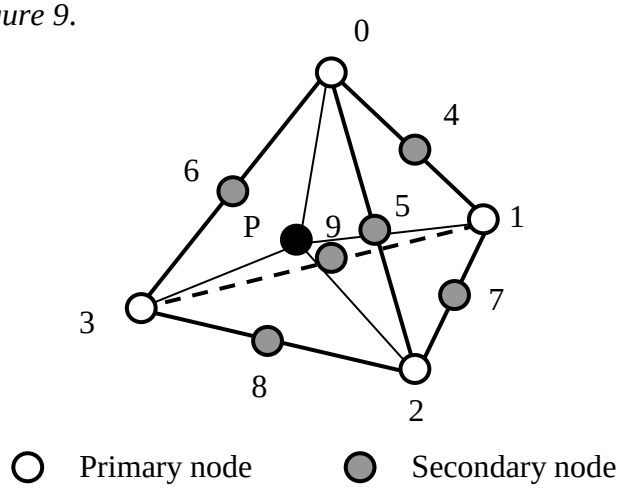

Figure 9. Quadratic Tetrahedral Element

(2). Integral points. The element has 4 integral points (default). It can be changed to 5 by command options. The integral point *P* has the following volumetric coordinates (L_0, L_1, L_2, L_3).

L_0 = volume of tetrahedron *P123* / volume of tetrahedron *0123* (5)
L_1 = volume of tetrahedron *P023* / volume of tetrahedron *0123* (6)
L_2 = volume of tetrahedron *P013* / volume of tetrahedron *0123* (7)
L_3 = volume of tetrahedron *P012* / volume of tetrahedron *0123* (8)

Table 3. Integral Points of Linear Tetrahedral Element (4 integral points)

Integral point number	L_0	L_1	L_2	L_3
0	β	α	β	β
1	β	β	α	β
2	β	β	β	α
3	α	β	β	β

α = 0.58541019662496845446
β = 0.13819660112501051518

Table 4. Integral Points of Linear Tetrahedral Element (5 integral points)

Integral point number	L_0	L_1	L_2	L_3
0	1/4	1/4	1/4	1/4
1	1/6	1/2	1/6	1/6
2	1/6	1/6	1/2	1/6
3	1/6	1/6	1/6	1/2
4	1/2	1/2	1/6	1/6

A.3. Linear Triangular Element

The linear triangular elements are used for integration when it is necessary to set the convection and radiation boundary conditions for linear tetrahedral elements.

(1). Nodes. The element contains 3 nodes with connectivity and numbering shown in *Figure10.*

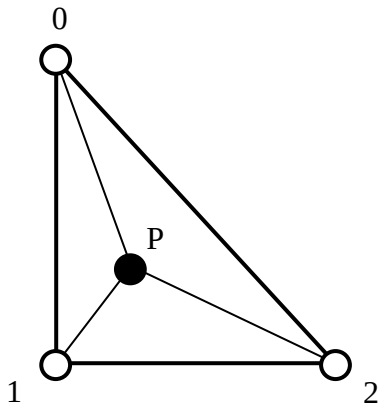

Figure 10. Linear Triangular Element

(2). Integral points. The element has 1 integral point. The integral point *P* has the following volumetric coordinates (L_0, L_1, L_2).

L_0 = area of triangular *P12* / area of triangular *012* (9)
L_1 = area of triangular *P02* / area of triangular *012* (10)
L_2 = area of triangular *P01* / area of triangular *012* (11)

Table 5. Integral Points of Linear Triangular Element

Integral point number	L_0	L_1	L_2
0	1/3	1/3	1/3

A.4. Quadratic Triangular Element

The quadratic triangular elements are used for integration when it is necessary to set the convection and radiation boundary conditions for quadratic tetrahedral elements.

(1). Nodes. The element contains 6 nodes with connectivity and numbering shown in *Figure 11*.

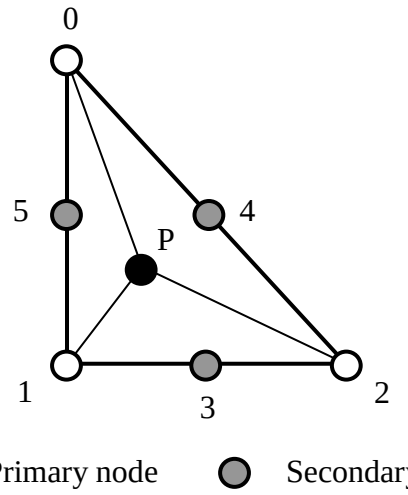

Figure 11. Quadratic Triangular Element

(2). Integral points. The element has 3 integral points. The integral point *P* has the following volumetric coordinates (L_0, L_1, L_2).

L_0 = area of triangular *P12* / area of triangular *012* (12)
L_1 = area of triangular *P02* / area of triangular *012* (13)
L_2 = area of triangular *P01* / area of triangular *012* (14)

Table 6. Integral Points of Quadratic Triangular Element

Integral point number	L_0	L_1	L_2
0	1/2	1/2	0
1	0	1/2	1/2
2	1/2	0	1/2

B. Setup of Boundary Conditions

The format of boundary condition data, which is used by ADVENTURE_Thermal will be presented below on examples.

B.1. Boundary Conditions for Temperature

Example

```
-----------------------------------------
[Properties]
1: content_type=FEGenericAttribute
2: num_items=81
3: fega_type=NodeVariable
4: label=Temperature
5: format=i4f8
6: index_byte=4
[Data]
0 0 1.000000e+02
1 0 1.000000e+02
3 0 1.000000e+02
58 0 1.000000e+02
59 0 1.000000e+02
60 0 1.000000e+02
.  .  .  .
.  .  .  .
.  .  .  .
```

The format of `[Data]` is (from left): the node number, the directional component, and the temperature. Since, the degree-of-freedom of nodes for heat conduction analysis is 1, setting of directional components, as it would be done for structure mechanics analysis, is unnecessary. All directional components are set to 0.

B.2. Boundary Conditions for Heat Flux

Example

```
----------------------------------------
[Properties]
1: content_type=FEGenericAttribute
2: num_items=81
3: fega_type=NodeVariable
4: label=HeatFlux
5: format=i4f8
6: index_byte=4
[Data]
0 0 0.000000e+00
1 0 3.333333e+01
3 0 0.000000e+00
58 0 6.666666e+01
59 0 6.666666e+01
60 0 6.666666e+01
.  .  .  .
.  .  .  .
.  .  .  .
```

The format of `[Data]` is (from left): the node number, the directional component, and the heat flux. The heat flux shown here was converted from the surface heat flux to the node-concentrated heat. If q is the heat flux per unit area S, the node-concentrated heat for quadratic tetrahedral element can be presented as

- P0

  $q_0 = 0$
- P1

  $q_1 = 0$
- P2

  $q_2 = 0$
- P3

  $q_3 = q \times S/3$
- P4

  $q_4 = q \times S/3$
- P5

  $q_5 = q \times S/3$

B.3. Boundary Conditions for Heat Convection

Example

```
-----------------------------------------
[Properties]
1: content_type=FEGenericAttribute
2: num_items=8
3: fega_type=ElementVariable
4: label=HeatConvection
5: format=i4f8f8
6: index_byte=4
[Data]
0 1 1.000000e+02 1.231002e+02
5 3 1.000000e+02 1.231002e+02
.  .  .  .
.  .  .  .
.  .  .  .
```

The format of `[Data]` is (from left): the element number, the surface number, the outer contact temperature, and the heat convection coefficient. The numbering of surfaces is done in a way that the surface numbers of each element been equal to the number of the node opposite to the surface. For example, the surface number 0 is opposite to the node number 0.

B.4. Boundary Conditions for Heat Radiation

Example

```
-----------------------------------------
[Properties]
1: content_type=FEGenericAttribute
2: num_items=8
3: fega_type=ElementVariable
4: label=HeatRadiation
5: format=i4f8f8f8
6: index_byte=4
[Data]
0 1 1.000000e+02 1.000000e+00 1.000000e+00
5 3 1.000000e+02 1.000000e+00 1.000000e+00
.  .  .  .
.  .  .  .
.  .  .  .
```

The format of `[Data]` is (from left): the element number, the surface number, the temperature of emitter, the emissivity, and the geometrical viewfactor.

B.5 Example of Material Properties Data:

(1). An example of a one-material model

```
---------------------------------
HeatConductivity 200
Density 10.0
SpecificHeat 100.0
StefanBoltzmanConstant 5.67e-6
InternalHeatGeneration 0.0
```

From the top: the heat conductivity coefficient, the material density, the specific heat, the *Stefan-Boltzmann* constant, and the internal heat generation.

(2). An example of a multi-material model

```
---------------------------------
#materialInfo
materialN 2
propertyN 5
HeatConductivity 100
Density 5000
SpecificHeat 41.78
StefanBoltzmanConstant 5.67e-6
InternalHeatGeneration 0.0
HeatConductivity 50
Density 2500
SpecificHeat 20.0
StefanBoltzmanConstant 5.67e-6
InternalHeatGeneration 0.0
#volumeInfo
volumeN 2
1
0
```

C. Tool Program

C.1 makefem_thermal

The boundary conditions and material properties attached to mesh can be saved in an entire-type FEA model file of ADVENTURE binary format by using **makefem_thermal** tool. This tool supports the following boundary conditions:

1. Temperature
2. Flux
3. Convection
4. Radiation

Input:

Mesh data file (extension is msh)
Mesh surface data file (extension is fgr)
File with boundary conditions (extension is cnd)
Material properties data file (extension is dat)

Output:

Entire-type FEA model file (extension is adv)

The following argument should be specified with **makefem_thermal** in the command line.

% makefem_thermal mshFile fgrFile cndFile matFile advFile [options]

mshFile :	the name of the mesh data file
fgrFile :	the name of the mesh surface data file
cndFile :	the name of the boundary conditions data file
matFile:	the name of the material properties data file
advFile:	the name of entire-type FEA model file

File format of boundary conditions data file(cndFile)

```
gravity 0 0 0                ← Dummy for thermal problem
boundary 2                   ← Number of boundary conditions
tempOnFaceGroup 0 1 10       ← Temperature in the surface group 0 is 10[ºC]
fluxOnFaceGroup 5 1 100      ← Flux on the surface group 5 is 100[W/m2]
```

tempOnFaceGroup is the temperature of face group
fluxOnFaceGroup is the flux on face group

The makefem_thermal tool in the present version of ADVENTURE Thermal supports the convection and radiation boundary conditions.
cnd file for convection boundary conditions

```
gravity 0 0 0                ← Dummy for thermal problem
boundary 2                   ← Number of boundary conditions
transOnFaceGroup 0 10.0 100.0 ← Surface 0 has ambient temperature10[ºC]
                                and heat transfer co-effiecient 100 (W/m2.0C)
```

Option:

-with-surface	This option will save the information of surface in the advFile.
-elm-src-file file name	This option will read internal heat generation of all elements from the text file

Heat source file will be like below

#HeatSourceInfo
SourceN 34
6.95e+04
--
--
--

C.2 pfemsolv

The pfemsolv is used to solve the interior unknowns of subdomains. The external surface and interface coniditions are considered as boundary conditions for each subdomain. This tool reads the mesh data from model directory and boundary conditions from result_surface directory.

Command:

% mpirun [options for mpirun] pfemsolv data_dir

Input files:
HDDM-type analysis model file:
data_dir/model/advhddm_in_P.adv

Boundary Conditions
data_dir/result_surface/advhddm_out_P.adv

Analysis results (steady analysis):
data_dir/`result/advhddm_out_P.adv`

D. libsparse

LIBSPARSE-0.2b

libsparse is a C library which carries out a number of operations on sparse matrices, particularly matrix vector multiplication using various sparse formats. It can convert from skyline format to other sparse formats and vice versa. Currently it only supports the symmetric matrices used in some modules of ADVENTURE Systems.

Include "advlas.h" where functions of libsparse is to be used.

Features of libsparse

1. This library can convert skyline matrix to other compressed storage formats and vice versa.
2. It can multiply sparse matrix with a vector (y += Ax).
3. It is based on ADVENTURE system and depends on some of files in source file of some of ADVENTURE modules. So before use of this library it should be properly linked with source file of ADVENTURE solve modules.
4. It supports only Symmetric matrices used in ADVENTURE_Thermal and ADVENTURE_Solid.
5. It can create different sparse matrix indexes using the nodal connectivity information.

Matrix formats [10] that are recognized include:
CSR: Compressed Sparse Row
CSC: Compressed Sparse Column
COO: Coordinate format
DCOO: Diagonal Coordinate format
MSR: Modified Compressed Sparse Row
SKY: Skyline format
ICSR: Incremental Compressed Sparse Row
VBCSR: Variable Block Compressed Sparse Row
DBCSR: Diagonal Block Compressed Sparse Row (for ADVENTURE_Solid)

Main Functions:
The following table shows the description of functions used in the LIBSPARSE library.

NewDMatrix()	It defines the indexes of sparse formats.
advlas_mkindex()	It reads subdomain mesh data and prepares the indexes of sparse formats. Different functions are used for different sparse formats.

advlas_cpmat_sky2nz()	It converts the skyline format to other sparse formats. Different functions are used for different sparse formats.
advlas_ldl_decomposite()	It decomposes the skyline matrix in ldl format
advlas_matmult_vec_add()	It multiplies a matrix with a vector and stores the value in another vector.
advlas_ldl_solve()	It solves the linear system using forward and backward substitutions.

Descriptions of Functions:

function to make Dmatrix
NewDMatrix(matdim, node_dim, matrix type)
output: return dmat

function to make matrix index
advlas_mkindex(nnd, nel, nd_elm, nop, node_dim, dmat)
Output: dmat

function to copy skyline matrix to non-zero only matrix
advlas_cpmat_sky2nz(skymat, nzmat, precon_sw)
Output: nzmat

function to multiply matrix with a vector
advlas_matmult_vec_add (dmat, temp, reac)
Output: reac

function to decompose skyline matrix as ldl format
advlas_ldl_decomposite (dmat, work)
Output: dmat (decomposed form)

function to solve ldl skyline matrix
advlas_ldl_solve(dmat, solution)
Output: solution

function to delete Dmatrix
DeleteDMatrix(&dmat)

Parameters:

int **nnd** - number of nodes
int **nel** – number of elements

int **nd_elm** – number of nodes per element
int **nop**- node connectivity in elements
int **node_dim** – unknown per nodes (1 for thermal and 3 for solid)
int **precon_sw** – preconditioning switch
int ***temp** – input vector
int ***reac** – output vector
structure DMatrix **dmat** - structure of matrix index and value
structure DMatrix **skymat** - structure of skyline matrix index and value
structure DMatrix **dmat** - structure of non-zero only matrix index and value

Command line options:

`-keep-kmat-csr`	This option specifies that the subdomain matrix will be stored as CSR format. This is default option.
`-keep-kmat-coo`	This option specifies that the subdomain matrix will be stored as COO format.
`-keep-kmat-msr`	This option specifies that the subdomain matrix will be stored as MSR format.
`-keep-kmat-csc`	This option specifies that the subdomain matrix will be stored as CSC format.
`-keep-kmat-icsr`	This option specifies that the subdomain matrix will be stored as ICSR format.
`-keep-kmat-vbcsr`	This option specifies that the subdomain matrix will be stored as VBCSR format.
`-keep-kmat-dcoo`	This option specifies that the subdomain matrix will be stored as DCOO format.
`-keep-kmat-dbcsr`	This option specifies that the subdomain matrix will be stored as DBCSR format. This option is only for ADVENTURE_Solid.
`-keep-kmat-sky`	This option specifies that the subdomain matrix will be stored as skyline format.

Sparse Matrix Storage Schemes

In order to take the advantage of the large number of zero elements, special formats are required to store sparse matrices. In this module, for symmetry sparse matrix, only triangular part of the matrix is stored. The main goal is to represent only the non-zero elements (*nnz*) considering the memory requirements and computation time. Several sparse matrices storage formats are discussed below.

Compressed Sparse Row (CSR)

Compressed sparse row format is popular and the most general purpose storage format for the sparse matrix. The elements are stored using three arrays: **data, row_ptr and col_ind.**

$$A = \begin{bmatrix} 1.0 & 2.0 & 0.0 & 6.0 & 0.0 \\ 2.0 & 3.0 & 4.0 & 7.0 & 9.0 \\ 0.0 & 4.0 & 5.0 & 0.0 & 0.0 \\ 6.0 & 7.0 & 0.0 & 8.0 & 10.0 \\ 0.0 & 9.0 & 0.0 & 10.0 & 11.0 \end{bmatrix}$$

data: The float array data of length *nnz* stores the element of *A* row by row.
col_ind: The integer vector col_ind of length *nnz* contains the column indices which correspond to the non-zero elements in the vector *data.*
row_ptr: The integer vector *row_ptr* of length *nrow* contains the pointers to the beginning of each row in the array *data* and *col_ind*.
With the **row_ptr** array we can easily compute the number of non-zero elements in the i^{th} row as *row_ptr*[i+1] - *row_ptr*[i].

The CSR representation of an example symmetric matrix *A*:

row_ptr	1	2	4	6	9	11

data	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0

col_ind	1	1	2	2	3	1	2	4	2	4	5

Coordinate Storage (COO)

The simplest sparse matrix storage structure is COO. It uses three arrays of length *nnz* to store the sparse matrix: **data, row_ind and col_ind.**
data: The float array *data* stores the non-zero elements of the sparse matrix row by row.
row_ind: The row_ind stores the row indices of the corresponding element.
col_ind: The col_ind stores the column indices of the corresponding element.
The COO storage format of the example matrix A is shown below.

row_ind	1	2	2	3	3	4	4	4	5	5	5

col_ind	1	1	2	2	3	1	2	4	2	4	5

data	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0

If the diagonal elements are stored first then the storage format is called Diagonal Coordinate Storage (DCOO).

Compressed Sparse Column (CSC)

In this sparse matrix storing format the elements are stored using three arrays: **data, col_ptr and row_ind.**
data: The float array data of length *nnz* contains the non-zero elements of *A* column by column.
row_ind: The integer array row_ind of length nnz contains the row indices which correspond to the non-zero elements in the array data.
col_ptr: The integer array col_ptr of length nrow contains the pointers to the beginning of each column in the array data and row_ind. With the col_ptr array we can easily compute the number of non-zero elements in the i^{th} column as col_ptr[i+1] - col_ptr[i].

col_ptr	1	4	8	9	11	11

data	1.0	2.0	6.0	3.0	4.0	7.0	9.0	5.0	8.0	10.0	11.0

row_ind	1	2	4	2	3	4	5	3	4	5	5

Modified Sparse Row (MSR)

The modified sparse row format has only two parallel arrays of equal length (nnz +1): A float array **data** and an integer array **col_ind**.
data: The first *n* position in data contains the diagonal elements of the matrix in order. The position *n+1* of the array data is not used, or may sometimes be used to carry other information concerning the matrix. Starting at position *n+1*, the non-zero elements of data excluding the diagonal elements, are stored by row.
col_ind: The *n+1* first position of col_ind contains the pointer to the beginning of each row in data. The rest position of col_ind represents its column indices which corresponds to the non-zero elements (off-diagonal) in the array data.
Thus for matrix A, the two arrays are shown in below.

data	1.0	3.0	5.0	8.0	11.0		2.0	4.0	6.0	7.0	9.0	10.0

col_ind	1	2	4	6	9	11	1	2	1	2	2	4

The restriction of MSR method is that principal diagonal element of coefficient matrix must be non-zero.

Variable Block Sparse Row (VBR)

The idea of this format is to exploit the non-zeros in contiguous locations by packing them. Unlike fixed-size blocking, the blocks will have variable lengths. As in fixed size blocking, if we know the column index of the first non-zero in a block, then we will also know the column indices of all its other non-zeros.This storage format requires an array *nz_ptr* (of length the number of blocks) in addition to the other three arrays used in CSR: data, col_ind and row_ptr.
data: The float array data stores the non-zero elements of *A* row by row.
col_ind: The integer array col_ind (of length the number of blocks) stores the column number of the first non-zero for each block.
row_ptr: The integer array row_ptr (of length the number of rows) to point to the position where the blocks of each row starts. The last element of the row_ptr is the number of blocks. **nz_ptr:** The integer array nz_ptr stores the location of the first non-zero of each block in the array data. An example of VBR is

nz_ptr	1	2	4	6	8	9	10

data	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0

col_ind	1	1	2	1	4	2	4

row_ptr	1	2	3	4	6	7

Incremental Compressed Sparse Row (ICSR)

The incremental compressed sparse row is a variant of MSR. Instead of 1D index itself, the difference with the 1D index of the previous nonzero is stored, as an increment. This storage format has two arrays with the following function:
data: The float array data of length nnz +1, stores non-zero elements of matrix *A*.
incr: The integer array incr of length nnz + 1 stores the increment with the previous non-zero.

ICSR representation of matrix *A* is shown as:

data	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0

incr	0	1	1	2	1	1	1	2	2	2	1

Skyline or Variable Band (SKY)

The Skyline representation becomes popular for direct solvers especially when pivoting is not necessary. The matrix elements are stored using three arrays: **data, row_ptrn, col_ind.**
data: The float array data stores the element of matrix *A* row by row.
col_ind: It contains column number of first element of each row.
row_ptr: This array points to the start of every row.

row_ptr	1	2	4	6	9

data	1.0	2.0	3.0	4.0	5.0	6.0	7.0	0.0	8.0	9.0	0.0	10.0	11.0

col_ind	1	1	2	1	2

This storage format stores some zero elements while other methods explained above does not.

Diagonal Block Compressed Sparse Row (DBCSR)

The idea of this format is to exploit fixed block shape of a matrix. Structural problem has such type of matrix. Supporting block shape is 3x3. In this format, if we know the column index of the first non-zero in a block, then we will also know the column indices of all its other non-zeros.

diag: This float array diag stores the nonzero of diagonal block (row wise).

data: The float array data stores the 3x3 block (block and row wise).

index_brow: This integer array points to the first element of first block of a row to the data array.

index_bcol: This integer array stores the column number of first element of each 3x3 block. The length is number of blocks.

1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21
22 23 24 25
26 27 28 29 30
31 32 33 34 35 36
37 38 39 40
41 42 43 44 45
46 47 48 49 50 51

diag	1	2	3	4	5	6	10	14	15	19	20	21	25	29	30	34	35	36	40

44	45	49	50	51

data	7	8	9	11	12	13	16	17	18	22	23	24	26	27	28	31	32	33

37	38	39	41	42	43	46	47	48

index_brow	0	0	9	18	27

index_bcol	0	3	0

E. Numerical Examples

Here simple examples of applications of the ADVENTURE_Thermal are described. In the interest of simplicity, the examples are limited to heat transfer problems on some test models and use of quadrilateral tetrahedral elements. These models are located in the directory sample_data/manual_example.

E.1 Numerical examples with temperature boundary conditions

Problem statement

Find the temperature distribution along the wall of a furnace shown in Fig. 12 The external width is W = 2 m, while the wall thickness is L = 0.5 m. The furnace is made of refractory brick with thermal conductivity k = 40 W/(m·^{0}K). The inner and the outer surfaces are kept at uniform constant temperatures t_i = 600 ^{0}C and t_o = 60 ^{0}C, respectively.

Analysis flow

1. Create geometry file and node density file

   temp.gm3d

   temp.gm3d

```
Box   0     0     0     2     2     2
Box   0.5   0.5   2     1     1     2
subtract
```

2. Create patch file using ADVENTURE_CAD

   % **advcad** temp.gm3d temp.pch 0.1

3. Automatic mesh generation using ADVENTURE_Tetmesh

   3a. Surface patch corrction

   % **advtmesh9p** temp

   3b. Linear tetrahedral mesh generation

   % **advtmesh9m** tempc

   3c. Linear tetrahedral to quadratic tetrahedral

   % **advtmesh9s** tempc

4. Extraction of mesh surface using ADVENTURE_BCtool-2.0

   % **msh2pch** tempcs.msh 3

5. Boundary conditions setup using ADVENTURE_BCtool-2.0

% **ADVENTURE_BcGUI_Ver_2_0** tempcs_3.pch tempcs_3.pcg

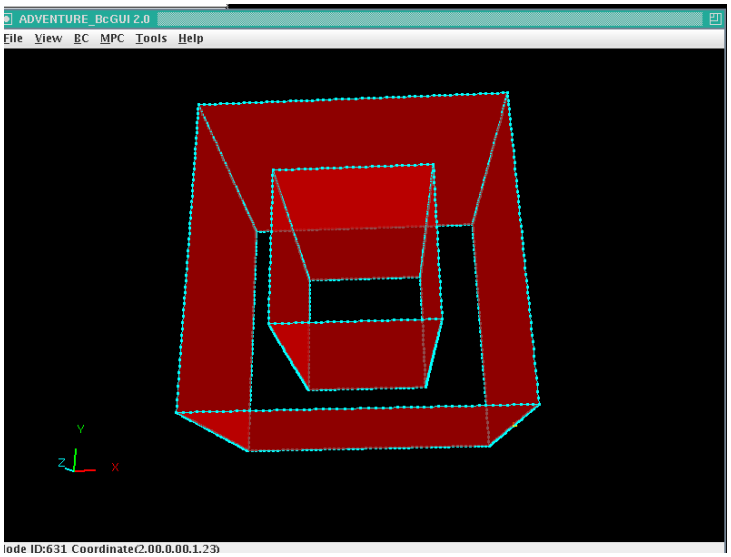

Figure 12. Example of temperature boundary condition set up

Output of ADVENTURE_BcGUI_2_0
temp.cnd

```
gravity 0.0 0.0 0.0
boundary 8
tempOnFaceGroup 0     0   0   60.0
tempOnFaceGroup 1     0   0   60.0
tempOnFaceGroup 4     0   0   60.0
tempOnFaceGroup 5     0   0   600.0
tempOnFaceGroup 6     0   0   600.0
tempOnFaceGroup 7     0   0   600.0
tempOnFaceGroup 8     0   0   600.0
tempOnFaceGroup 9     0   0   60.0
```

6. Create material data file

s_mat.dat

```
HeatConductivity 40
```

7. Create entire FEA model using ADVENTURE_BCtool-2.0

% **makefem3** tempcs.msh tempcs_3.fgr temp.cnd s_mat.dat temp.adv
(Output file will be temp.adv)

8. Create HDDM type model data file using ADVENTURE_Metis

% **mpirun –np 4 adventure_metis** –difn 1 temp.adv temp 700
(HDDM file will be saved in a directory “temp”)

9. Analyze heat conduction model using ADVENTURE_Thermal

% **mpirun –np 4 advthermal-p** temp
(A directory named “result” will be created in the directory of “temp”)

10. Create temperature data file

% **hddmmrg** Temperature temp
(Output of this command is Temperature.dat)

11. Visualize temperature distribution using ADVENTURE_Auto (It reads the Temperature.dat file)

% **advauto_thermalview** tempcs.msh tempcs_3.fgr

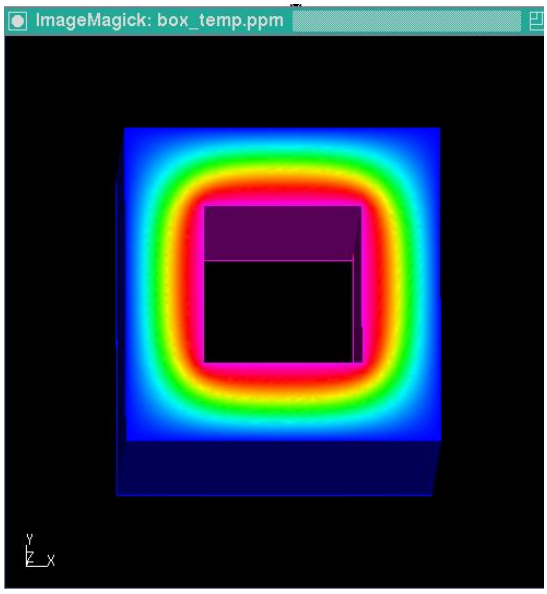

Figure 13. Temperature distribution using ADVENTURE_Auto

E.2 Numerical examples with flux boundary conditions

Problems Statement

Consider the conduction heat transfer in a cross section of a cylinder shown in *Figure 14,* thermal conductivity $k = 50$ *[W/mm·K]*, inner radius in 125 mm and outer radius 250 mm. The outer surface is maintained at a temperature of $T_1 = 10$ [°C] and heat flows through the inner surface at a rate of 100 [W/mm^2]. The rest of the surfaces are maintained with natural boundary conditions. We wish to determine the steady temperature distribution through the model using ADVENTURE_Thermal.

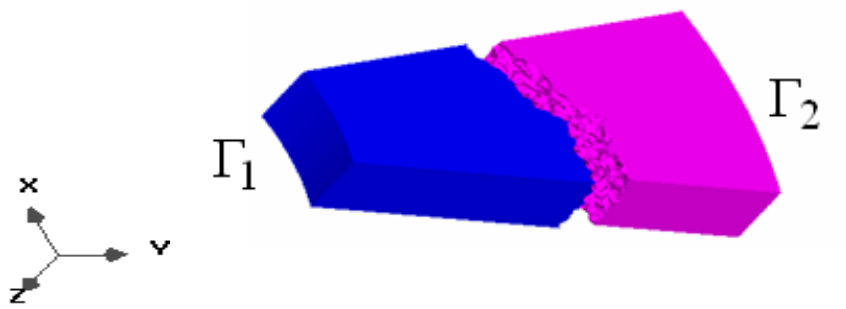

Figure 14. Analysis Model (Cross section of a cylinder)

Analysis flow

1. Create geometry file and node density file

   flux.igs

   flux.ptn

```
BaseDistance
4.0
```

2. Create patch file using ADVENTURE_TriPatch

   % **ADVENTURE_TriPatch** flux flux

3. Automatic mesh generation using ADVENTURE_Tetmesh

   3a. Surface patch corrction

   % **advtmesh9p** flux

   3b. Linear tetrahedral mesh generation

   % **advtmesh9m** fluxc

3c. Linear tetrahedral to quadratic tetrahedral
% **advtmesh9s** fluxc

4. Extraction of mesh surface using ADVENTURE_BCtool-2.0

% **msh2pch** fluxcs.msh 3

5. Boundary conditions setup using ADVENTURE_BCtool-2.0

% **ADVENTURE_BcGUI_Ver_2_0** fluxcs_3.pch fluxcs_3.pcg

Output of ADVENTURE_BcGUI_2_0

flux.cnd

```
gravity 0.0 0.0 0.0
boundary 2
tempOnFaceGroup 1      0    0    10
fluxOnFaceGroup 4      0    0    100
```

6. Create material data file

s_mat.dat

```
HeatConductivity 40
```

7. Create entire FEA model using ADVENTURE_BCtool-2.0

% **makefem3** fluxcs.msh fluxcs_3.fgr flux.cnd s_mat.dat flux.adv
(Output file will be flux.adv)

8. Create HDDM type model data file using ADVENTURE_Metis

% **mpirun –np 2 adventure_metis** –difn 1 flux.adv flux 700
(HDDM file will be saved in a directory “flux”)

9. Analyze heat conduction model using ADVENTURE_Thermal

% **mpirun –np 2 advthermal-p** flux
(A directory named “result” will be created in the directory of “flux”)

10. Create temperature data file

% **hddmmrg** Temperature flux
(Output of this command is Temperature.dat)

11. Visualize temperature distribution using ADVENTURE_Auto (It reads the Temperature.dat file)

% **advauto_thermalview** fluxcs.msh fluxcs_3.fgr

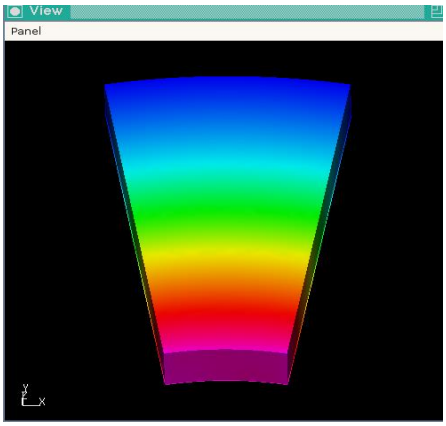

Figure 15. Temperature distribution using ADVENTURE_Auto

E.3 Numerical examples with convection boundary conditions

Problem statement

Consider the conduction heat transfer in a cross section of a hollow cylinder with heat conductivity, 8.64×10^{-2} W/(mm·K). The convection coefficient and temperature of the surrounding medium in the inner surface are 2.8372×10^{-3} W/(mm^2·K) and 38 ^{0}C . The convection coefficient and temperature of the surrounding medium in the outer surface are 1.4186×10^{-3} W/(mm^2·K) and -18 ^{0}C. We wish to determine the steady temperature distribution through the model using ADVENTURE_Thermal.

Analysis flow

1. Create geometry file and node density file

   conv.igs

   conv.ptn

```
BaseDistance
4.0
```

2. Create patch file (use without file extension)

   % **ADVENTURE_TriPatch** conv conv

3. Automatic mesh generation using ADVENTURE_Tetmesh

   3a. Surface patch correction
   % **advtmesh9p** conv

   3b. Linear tetrahedral mesh generation
   % **advtmesh9m** convc

   3c. Linear tetrahedral to quadratic tetrahedral
   % **advtmesh9s** convc

4. Extraction of mesh surface using ADVENTURE_BCtool-2.0

   % **msh2pch** convcs.msh 3

5. Boundary conditions setup using ADVENTURE_BCtool-2.0

   % **ADVENTURE_BcGUI_Ver_2_0** convcs_4.pch convcs_3.pcg

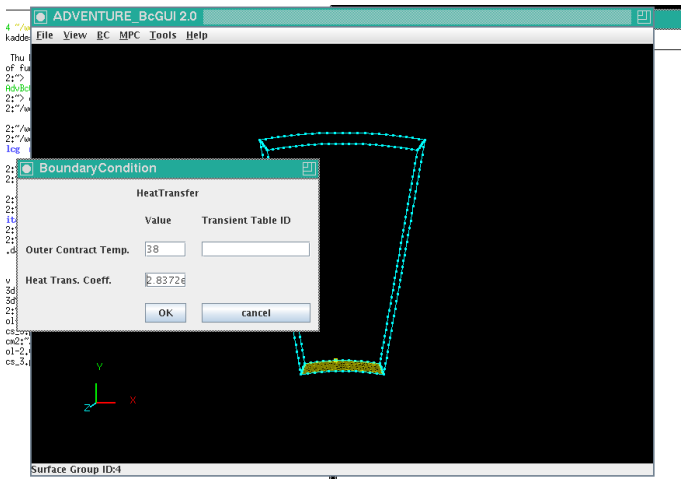

Figure 16. Example of convection boundary condition set up

Output of ADVENTURE_BcGUI_2_0
conv.cnd

```
gravity 0.0 0.0 0.0
boundary 2
transOnFaceGroup 1      -18      .0014186
transOnFaceGroup 4       38      .0028372
```

6. Create material data file

s_mat.dat

```
HeatConductivity 0.086475
```

7. Create entire FEA model using ADVENTURE_BCtool-2.0

% **makefem3** convcs.msh convcs_3.fgr conv.cnd s_mat.dat conv.adv

8. Create HDDM type model data file using ADVENTURE_Metis

% **mpirun –np 2 adventure_metis** –difn 1 conv.adv conv 71

9. Analyze heat conduction model using ADVENTURE_Thermal

% **mpirun –np 2 advthermal-p** conv
(A directory named “result” will be created in the directory of conv)

10. Create temperature data file

% **hddmmrg** Temperature conv
(Output of this command is Temperature.dat)

11. Temperature distribution using ADVENTURE_Auto

% **advauto_thermalview** convcs.msh convcs_3.fgr

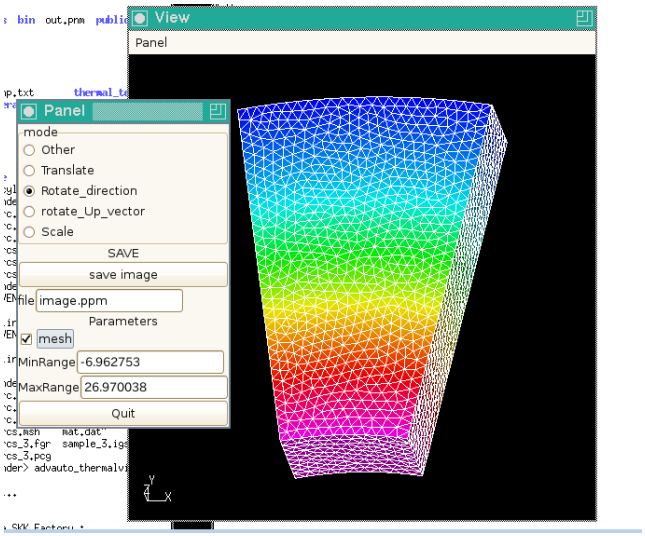

Figure 17. Temperature distribution using ADVENTURE_Auto

E.4 Numerical examples with internal heat generation

Problem statement

Consider the conduction heat transfer in a solid cylinder bar of radius 0.1m and thermal conductivity, 40 W/(m·^{0}K) that is heated by the passage of an electric current, which generates heat energy 4x10^6 (W/m^3). Heat is dissipated from the surface of the bar by convection into the surrounding medium at an ambient temperature of 20^{0}C. The heat transfer coefficient of the medium is 400 W/(m^2·^{0}K). We wish to determine the steady temperature distribution through the model using ADVENTURE_Thermal.

Analysis flow

1. Create geometry file using ADVENTURE_CAD

igen.gm3d

igen.gm3d

```
circle 0 0 0 0.1 0 0 0 0 1 16
extrude 0 0 0.5
```

2. Create patch file using ADVENTURE_CAD

% **advcad** igen.gm3d igen.pch 0.01

3. Automatic mesh generation using ADVENTURE_Tetmesh

3a. Surface patch correction
% **advtmesh9p** igen

3b. Linear tetrahedral mesh generation
% **advtmesh9m** igenc

3c. Linear tetrahedral to quadratic tetrahedral
% **advtmesh9s** igenc

4. Extraction of mesh surface using ADVENTURE_BCtool-2.0
% **msh2pch** igencs.msh 3

5. Boundary conditions setup using ADVENTURE_BCtool-2.0

% **ADVENTURE_BcGUI_Ver_2_0** igencs_3.pch igencs_3.pcg

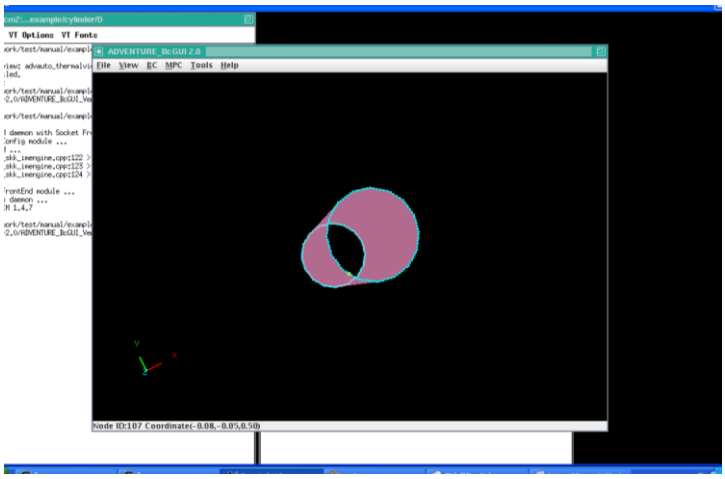

Figure 18. Example of convection boundary condition set up

Output of ADVENTURE_BcGUI_2_0
igen.cnd

```
gravity 0.0 0.0 0.0
boundary 1
transOnFaceGroup 0      20      400
```

6. Create material data file

s_mat.dat

```
HeatConductivity 40
InternalHeatGeneration 4.0E6
```

7. Create entire FEA model using ADVENTURE_BCtool-2.0

% **makefem3** igencs.msh igencs_3.fgr igen.cnd s_mat.dat igen.adv

8. Create HDDM type model data file using ADVENTURE_Metis

% **mpirun –np 2 adventure_metis** –difn 1 igen.adv igen 400

9. Analyze heat conduction model using ADVENTURE_Thermal

% **mpirun –np 2 advthermal-p** igen
(A directory named “result” will be created in the directory of conv)

10. Create temperature data file

% **hddmmrg** Temperature igen
(Output of this command is Temperature.dat)

11. Temperature distribution using ADVENTURE_Auto

% **advauto_thermalview** igencs.msh igencs_3.fgr

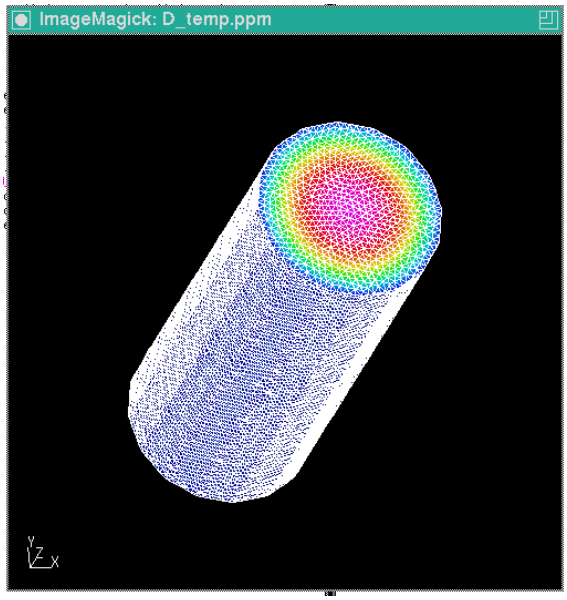

Figure 19. Temperature distribution using ADVENTURE_Auto

E.5 Numerical examples with multi material model

Calculate the stationary temperature distribution in a nuclear fuel element, whose shape can be approximated by a long circular cylinder of radius R = 15 mm. The element is covered with a protective steel layer of thickness L = 3 mm. Thermal conductivities of the nuclear fuel and of steel are k_f = 30 W/(m·^{o}K) and k_s = 36 W/(m·^{o}K), respectively. The rate of internal heat generation per unit volume of nuclear fuel is f = $4x10^6$ W/m^3, while the convection coefficient and the temperature of the surrounding medium are h_c = 100 W / (m^2·^{o}K) and t_c = 300 ^{o}C.

Analysis flow

1. Create geometry file (using Meshman) and node density file

Cylinder

cylinder.igs

cylinder.ptn

```
BaseDistance
3.0
```

Tube

tube.igs

tube.ptn

```
BaseDistance
3.0
```

2. Create patch file (use without file extension) using ADVENTURE_TriPatch

2a. Create patch file seperately

% **ADVENTURE_TriPatch** cylinder cylinder
% **ADVENTURE_TriPatch** tube tube

2b. Merge `cylinder` and `tube` to `multi`

% mrpach cylinder.pcm cylinder.pcg tube.pcm tube.pcg -o multi.pcm –g multi.pcg

3. Automatic mesh generation using ADVENTURE_Tetmesh

3a. Surface patch correction

% **advtmesh9p** multi

3b. Linear tetrahedral mesh generation

% **advtmesh9m** multi

3c. Linear tetrahedral to quadratic tetrahedral

% **advtmesh9s** multi

4. Extraction of mesh surface using ADVENTURE_BCtool-2.0

% **msh2pch** multics.msh 4

5. Boundary conditions setup using ADVENTURE_BCtool-2.0

5a. Displaying multi material model

% msh2pcm multics.msh

% **ADVENTURE_BcGUI_Ver_2_0** multics_V.pcm multics_V.pcg

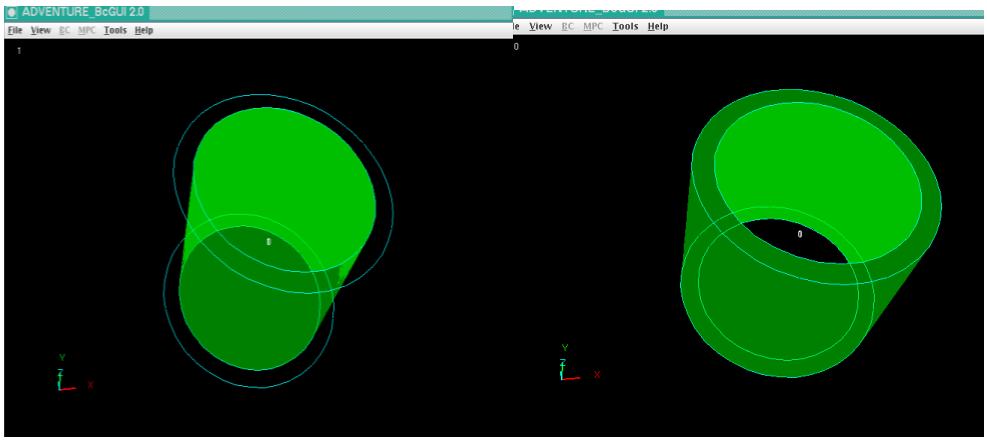

Figure 20. Volume 1 *Volume 0*

5b. Boundary condition setup

% **ADVENTURE_BcGUI_Ver_2_0** multics_4.pch multics_4.pcg

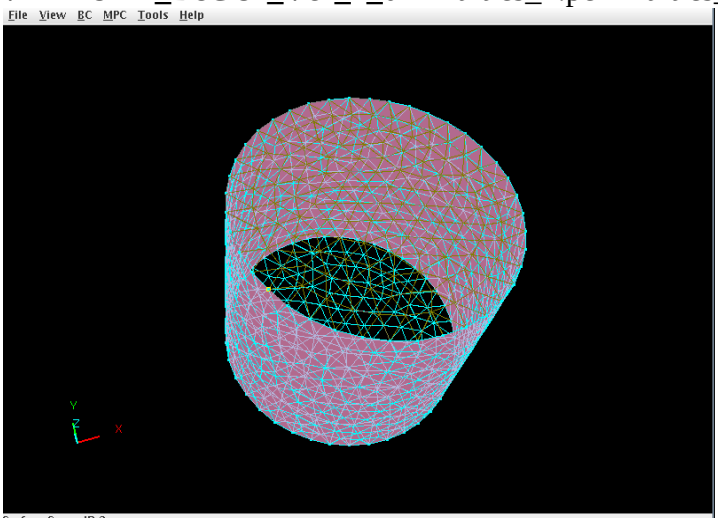

Figure 21. Example of convection boundary condition set up at the outer surface

Output of ADVENTURE_BcGUI_2_0
multi.cnd

```
gravity 0.0 0.0 0.0
boundary 1
transOnFaceGroup 0      300      100e-6
```

6. Create material data file

m_material.dat

```
#materialInfo
materialN 2
propertyN 2
HeatConductivity    36.0e-3
InternalHeatGeneration 0.0
HeatConductivity    30.0e-3
InternalHeatGeneration 4.0e-3
#volumeInfo
volumeN    2
0
1
```

7. Create entire FEA model using ADVENTURE_BCtool-2.0

% **makefem3** multics.msh multics_4.fgr multi.cnd m_material.dat multi.adv

8. Create HDDM type model data file using ADVENTURE_Metis

% **mpirun –np 2 adventure_metis** –difn 1 multi.adv multi 40

9. Analyze heat conduction model using ADVENTURE_Thermal

% **mpirun –np 2 advthermal-p** multi
(A directory named “result” will be created in the directory of multi)

10. Create temperature data file

% **hddmmrg** Temperature multi
(Output of this command is Temperature.dat)

11. Temperature distribution using ADVENTURE_Auto

% **advauto_thermalview** multics.msh multics_4.fgr

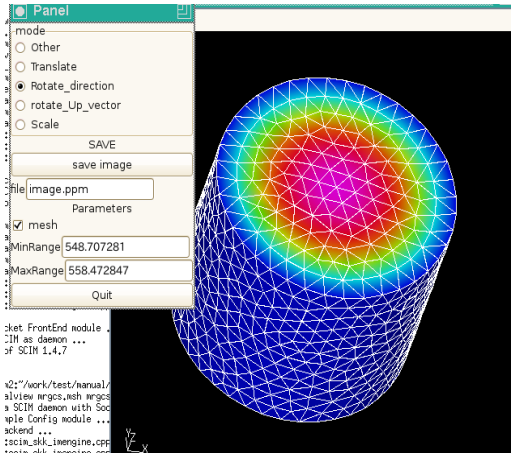

Figure 22. Temperature distribution using ADVENTURE_Auto

E.6 Numerical examples of with surface information

Calculate the steady temperature distribution of the part of a cooling stave shown in Figure-23. Consider the following conditions.

Air temperature is 50ºC, water temperature is 30ºC, hot gases temperature 1600ºC.

Heat convection coefficients: between furnace shell and atmosphere – 12 W/(m^2 K)
between water and inneside of the cooling stave- 8000 W/(m^2 K)
between hot gases and furnace shell -260 W/(m^2 K)

Heat conductivity: Furnace shell and steve body- 52.2 W/(m K)
Filling material - 0.35 W/(m K)
Invalid bricks and lining material- 21 W/(m K)
Formation of slag is not considered.

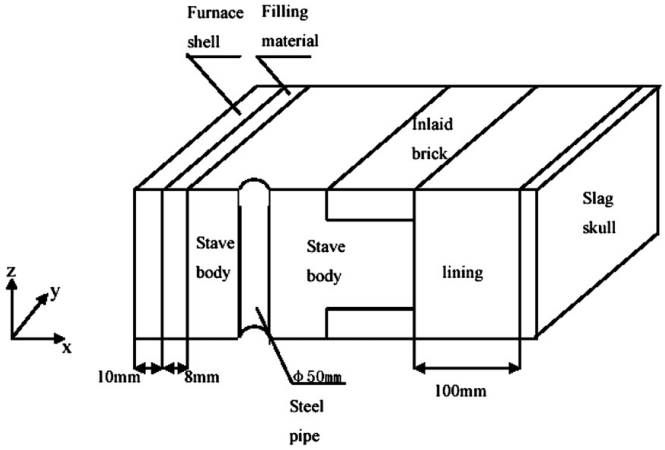

Figure 23 Part of cooling stave

Analysis Flow:

Following the steps 1-3 of the previous example for multi volume model we have the final mesh file partstavecs.msh.

1. Extraction of mesh surface using ADVENTURE_BCtool-2.0
   % msh2pch multics.msh 3

2. Boundary condition setup
   % ADVENTURE_BcGUI_Ver_2_0 partstavecs_3.pch partstavecs_3.pcg

Output of ADVENTURE_BcGUI_2_0
thermal.cnd

```
gravity 0.0 0.0 0.0
boundary 3
transOnFaceGroup 4     30.0     0.0080
transOnFaceGroup 6     1600      2.6E-4
transOnFaceGroup 7     50.0     0.12E-4
```

3. Create material data file
m_material.dat

```
materialN 3
propertyN 1
HeatConductivity    52.2e-3
HeatConductivity    0.35e-3
HeatConductivity    21e-3
volumeN    7
0
2
2
2
2
1
0
```

4. Create entire FEA model using makfem_thermal
% makefem_thermal partstavecs.msh partstavecs_4.fgr thermal.cnd m_material.dat partstave.adv –with-surface

5. Create HDDM type model data file using ADVENTURE_Metis
% mpirun –np 4 adventure_metis –difn 1 partstave.adv partstave 200

6. Analyze heat conduction model using ADVENTURE_Thermal
% mpirun –np 4 advthermal-p -result-surface-interface partstave
(A directory named “result_surface” will be created in the directory of partstave)

7. Calculate interior temperature
% mpirun –np 4 pfemsolv partstave
(A directory named “result” will be created in the directory of partstave)

8. Create temperature data file
% hddmmrg Temperature partstave
(Output of this command is Temperature.dat)

9. Temperature distribution using ADVENTURE_Auto
% advauto_thermalview partstavecs.msh partstavecs_4.fgr

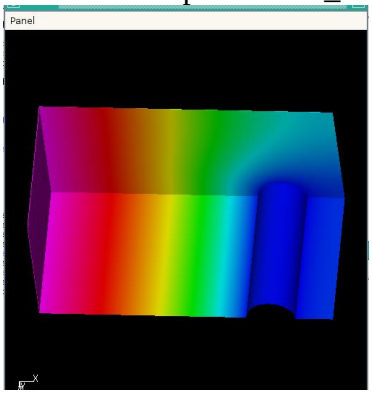

Figure 24 Temperature distribution using Advauto_thermalview

E.7 Numerical examples of large scale analysis

A large scale HTTR (High Temperature Test Reactor) model (*Figure 25*) with about 2 millions degrees of freedom (dof) is analyzed by ADVENTURE_Thermal module. As the boundary conditions for this model, some high temperature is set on the lower plan and some low temperature is set on the upper plan. *Figure 26*. shows the temperature distribution after solution by ADVENTURE_Thermal.

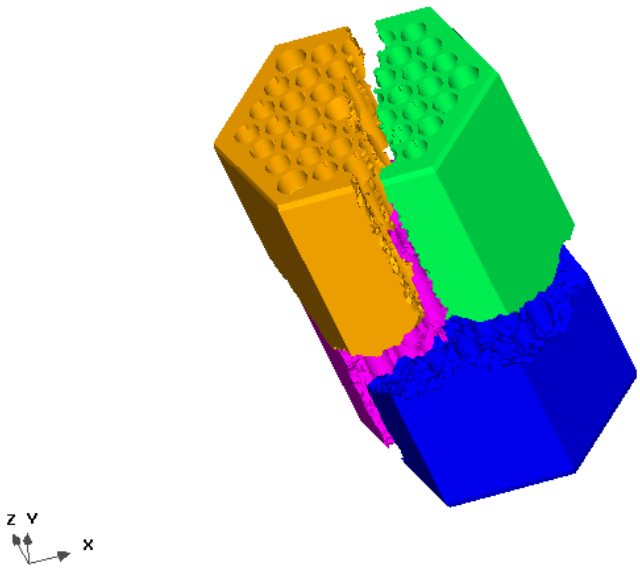

Figure 25. Domain Decomposition of HTTR Model

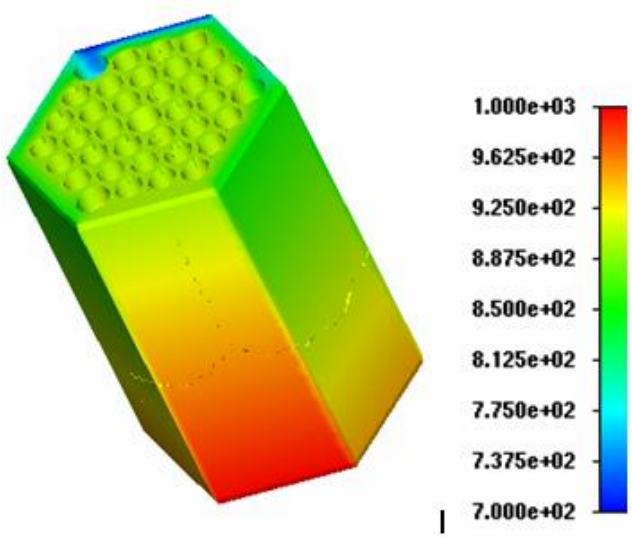

Figure 26. Temperature Distribution Visualized by ADVENTURE_Visual (Old)

References

[1]. ADVENTURE Project: http://adventure.sys.t.u-tokyo.ac.jp

[2]. G.Yagawa and R.Shioya: Parallel Finite Elements on a Massively Parallel Computer with Domain Decomposition, *Computing Systems in Engineering*, 4, Nos. 4-6 (1993), pp. 495-503.

[3]. G.Yagawa and R.Shioya: Massively Parallel Finite Element Analysis, Asakura-Shoten, (1998) (in Japanese).

[4]. T.Miyamura, H.Noguchi, R.Shioya, S.Yoshimaura and G.Yagawa: Massively Parallel Elastic-Plastic Finite Element Analysis Using the Hierarchical Domain Decomposition Method, *Transactions of Japan Society of Mechanical Engineers (JSME)*, 65-A, No.634(1999), pp. 1201-1208 (in Japanese).

[5]. R.Shioya, H.Kanayama, D.Tagami and E.Imamura: A Domain Decomposition Approach for Non-steady Heat Conductive Analysis, *Advances in Computational Engineering & Science*, 189.pdf, pp. 1-6, 2001.

[6]. MPI: http://www-unix.mcs.anl.gov/mpi/

[7]. MPICH: http://www-unix.mcs.anl.gov/mpi/mpich/

[8] Jan Mandel: Balancing Domain Decomposition, *Communications on Numerical Methods in Engineering*, 9(1993), 233-241

[9] R.Shioya, H. Kanayama, A.M.M.Mukaddes and M. Ogino: Heat Conductive Analysis with Balancing Domain Decomposition Method, *Journal of Theoretical and Applied Mechanics*, 52(2003), 43-53.

[10] Abul Mukid Mohammad Mukaddes, Masao Ogino and Ryuji Shioya, Performance Evaluation of Domain Decompositon Method with Sparse Matrix Storage Schemes in Moder Supercomputer, International Journal of Computational Methods, Volume 11, Issue Supp 01, Nov. 2014 DOI: 10.1142/S0219876213440076.

[11] A M M Mukaddes, M. Ogino, H. Kanayama, and R. Shioya, A Scalable Balancing Domain Decomposition Based Preconditioner for Large Scale Heat Transfer Problems, JSME International Journal, B-Fluid T. 49-2(2006), 533-540.