ADVENTURE_Solid

HDDM solver for static elastic / elasto-plastic / large deformation stress analyses

Version: $\beta - 0.81$

User Manual

May 21, 2001

ADVENTURE Project

Contents

1. Outline

The ADVENTURE_Solid module is a finite element method analysis solver designed in the ADVENTURE Project [1] to perform the solid static analysis using Hierarchical Domain Decomposition Method (HDDM) [2, 3, 4] and parallel data processing. The ADVENTURE_Solid module has the following main features:

- Dynamic or static load distribution can be obtained using Hierarchical Domain Decomposition Method (HDDM) with execution on single CPU and parallel network-connected computers.

- It can perform the elastic, elasto-plastic, and geometric nonlinear stress analyses.

- The incremental load/displacement control method is used for the elasto-plastic/ nonlinear geometry analyses.

- Large increment width is possible by stress integration using the backward Euler method.

- Quick convergence at each increment step is reached by using the tangential stiffness, according to the Newton-Raphson method.

- Linear tetrahedral, linear hexahedral, quadratic tetrahedral, and quadratic hexahedral elements can be treated.

- The module is designed for *UNIX* and *LINUX* platforms.

- Using the parallel processing computer library MPI [5], the program can operate in various parallel computing environments using network-connected PC, workstations and MPPs (Massively Parallel Processors).

The operational flowchart of ADVENTURE System is shown in Fig. 1.

1. Preparation of the mesh data (ADVENTURE_TetMesh)

The ADVENTURE_TetMesh module performs mesh decomposition of the analyzed object. Other programs for mesh decomposition can be used changing the corresponding data format.

2. Setup of the boundary conditions (ADVENTURE_BCtool)
   Corresponded boundary conditions added to the mesh and the material properties are set up using the ADVENTURE_BCtool module.

3. Decomposition of domain (ADVENTURE_Metis).
   The ADVENTURE_Metis module decomposes an entire type analysis model. Parallel processing is possible.

4. FEM analysis (ADVENTURE_Solid).
   The ADVENTURE_Solid module performs finite element method analysis using the decomposed analysis model prepared at the earlier step. Parallel processing is possible.

5. Post system (ADVENTURE_Visual).
   The ADVENTURE_Visual module visualizes the analysis results. Parallel processing is possible.

MPI is required for parallel operation of ADVENTURE_Solid in *UNIX* or *Linux* computing environments. The freely available MPICH [6] (an implementation of the MPI standard) supports many computing environments, however another programs for the MPI standard can be used. For operation of ADVENTURE_Solid in a single mode, MPI is not necessary.

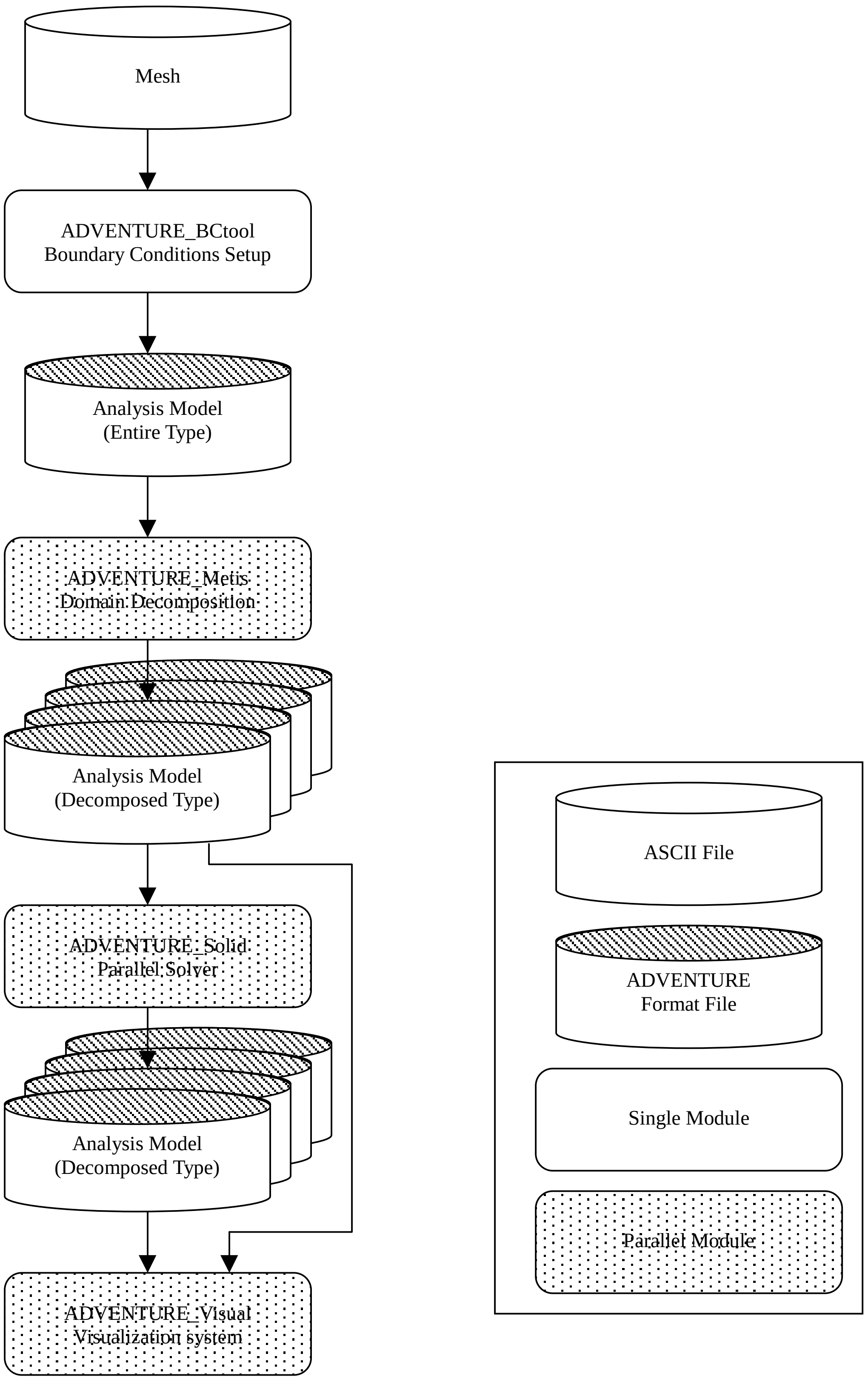

Fig. 1.   Operational flowchart

2. Parallel Processing and Domain Decomposition Methods

In the ADVENTURE_Solid module, parallel processing using the hierarchical domain decomposition method is possible. Fig. 2 shows the hierarchical domain decomposition of the entire model. The model is decomposed in two steps. Large decomposed unit of the first hierarchy level refers as “Part”, and smaller unit(s) of the decomposed “Part” (2nd hierarchy level) refer(s) as “Domain(s)”.

Domain decomposition is performed by ADVENTURE_Metis prior to operation of ADVENTURE_Solid. The details are given in the ADVENTURE_Metis User Manual. Several methods are prepared in ADVENTURE_Solid to treat the assignment of each decomposed domain to the network node (process) in the most effective way. In order to optimize the use of computer memory and computing time, the USER should consider optimum domain decomposition.

MPI is used as parallel libraries for ADVENTURE_Solid and at the starting time of execution, the number of processes started at once will be decided in according to the environment settings defined by the USER. Commonly, one process is assigned for one CPU, however a number of processes can be also assigned to one CPU. Here, the terms “node”, “process”, and “CPU” are used for convenience without any special distinctions.

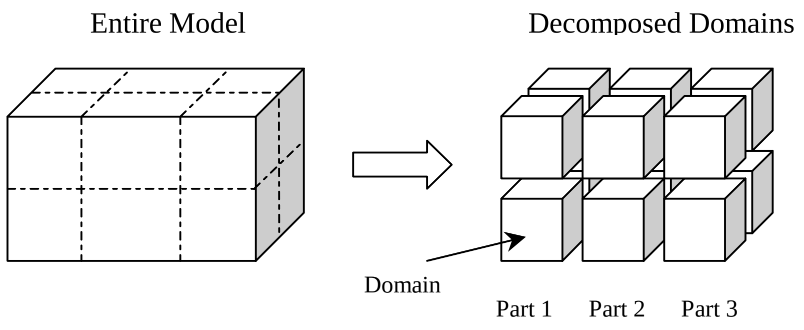

Fig. 2.   Hierarchical domain decomposition

2.1. Parallel Processing Methods and Number of "Parts"

Depending on parallel processing methods, three executable programs are prepared in the ADVENTURE_Solid module. Each executable program requires special domain decomposition that will be described below.

1. ***Single version*** (**`advsolid-s`**)

   The calculation is performed as a single process. MPI is not required for the program compilation and execution. There are no limitations on numbers of "Parts" or "Domains"; the model prepared for parallel computation can be used for the single processor without adjustments (Fig. 3). In the single processor, computational and data reprocessing procedures for each "Part" occur in the same order as it would be occurred in the parallel computing system. If parallel computation is not performed well, the single version of the program can be used as a "checker".

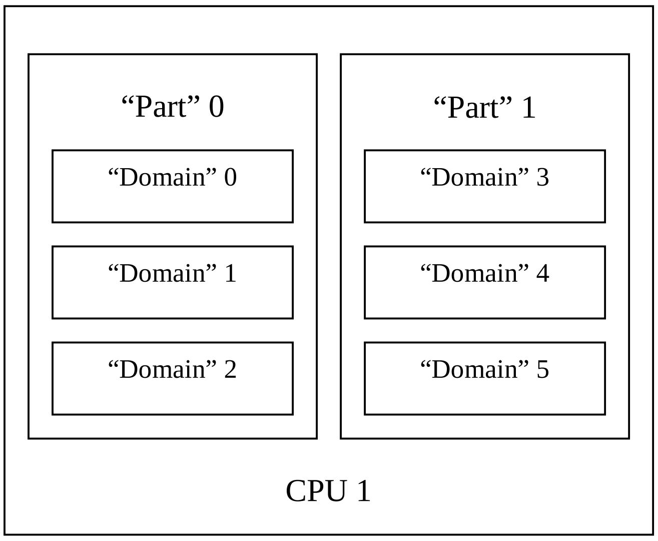

Fig. 3. Adjustment of domains to CPU (Single version)

2. ***Static load distribution version*** (**`advsolid-p`**)

   As it is shown in Fig. 4, the calculations are performed in parallel and the processes are assigned statically. One process is automatically assigned for one "Part" prior to program execution, and, in this case, to decrease network communications between nodes, the number of "Parts" should be equal to the number of network nodes. This version works efficiently if all nodes have the same performance (uniform system).

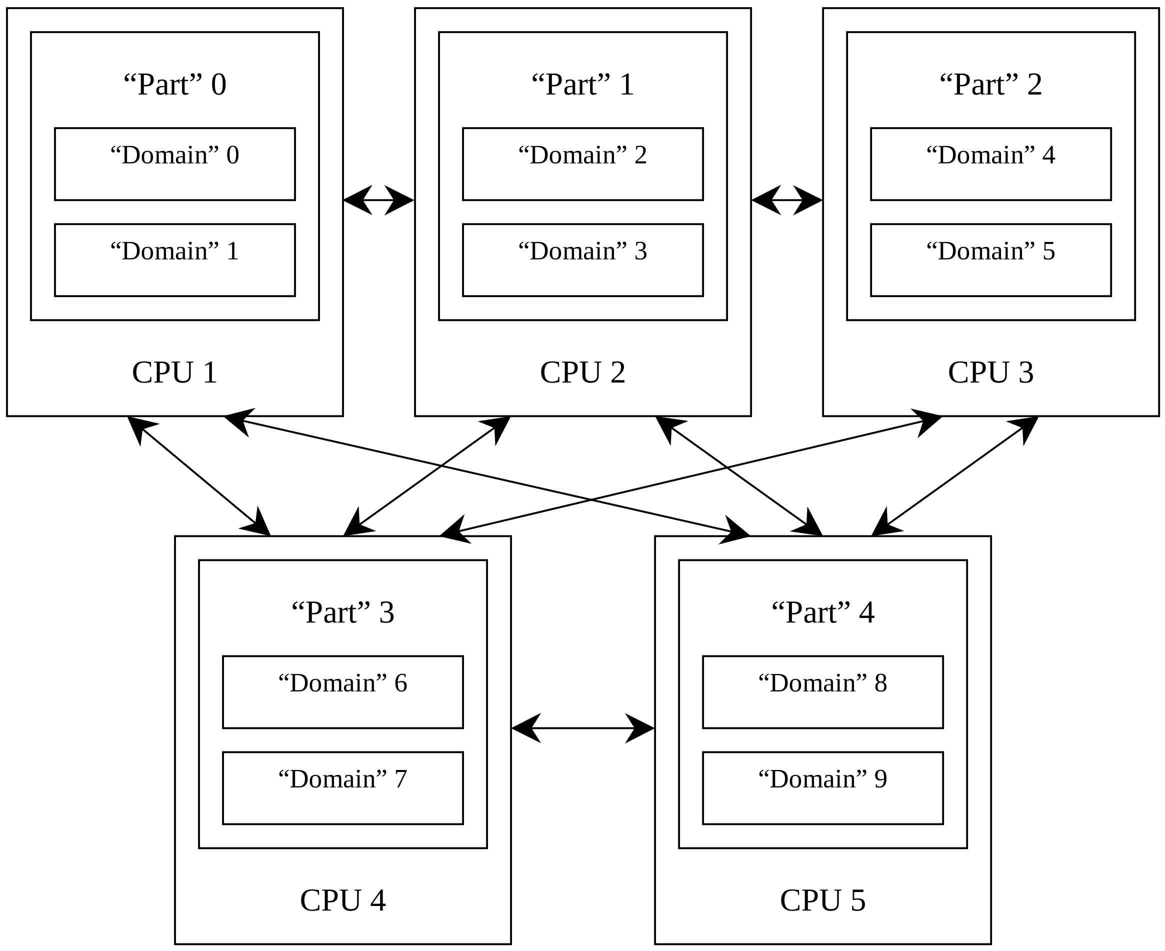

Fig. 4. Adjustment of domains to CPUs (Static load distribution version)

3. *Dynamic load distribution version* (`advsolid-h`)

The treatment of "Parts" and "Domains" is accomplished dynamically when the domain-decomposed data prepared by ADVENTURE_Metis are read by ADVENTURE_Solid and subdivided between CPUs in the way that one "Part" will be assigned for one CPU and the remained CPUs will be used for treatment of "Domains" (Fig. 5). Here, such sharing will be referred as the "Parent" process and the "Child" process. The "Child" processes do necessary calculations of "Domains" and, later, the "Parent" process handles the calculated results. The number of "Parts" should be less than the total number of CPUs considered for calculation. Since the communications will be concentrated on the "Parent" nodes, the efficiency will be decreased if the number of the "Child" nodes is much larger than the number of the "Parent" nodes. To overcome this problem, the "Parent" nodes are also considered to work in parallel and processing of "Parents" can also be distributed.

Since the "Child" nodes will perform many calculations, the jobs assignment direction should be selected in order to get maximum performance. For example, if 10 CPUs are used, it is better to assign 1 or 2 "Parent" nodes and large job should be assigned to the remained "Child" nodes.

The balance of calculation work can be dynamically adjusted, however the total performance of this job distribution method is worse than that of the static load distribution method for uniform computer environments due to large data transfer by network communications. The dynamic load distribution is well suited for non-uniform computer environments. The "Parent" and the "Child" processes are always shifted in time and can be done by the same CPU on one network node. In this case, the processes should be subdivided in the following manner: the "Parent" process should have the MPI rank from 0 to $N_{part} - 1$, and the "Child" process should have the MPI rank from N_{part} to $N_{proc} - 1$, where N_{part} is the number of "Parts" and N_{proc} is the number of processes. For example, if 8 nodes will be used to distribute 2 "Parent" and 8 "Child" processes, the total number of execution processes will be 10. The following figure shows a part of setup file (for **`mpich`** ***`ch_p4`*** *device*) **`machine_file`** for assigning host names, which should be set up by the USER (See Appendix C). 10 processes will be started at once: 2 "Parent" processes and 8 "Child" processes. First two lines are corresponded to the "Parent" processes. **`host0`** and **`host1`** here take the functions of "Parent" and "Child" simultaneously. For detailed information on the setup procedures, refer to the MPI-related documents. It should be mentioned that depending on computing environments and programs for the MPI standard implementation, the data transfer could be slowed down.

```
host0
host1
host0
host1
host2
host3
host4
host5
host6
host7
```

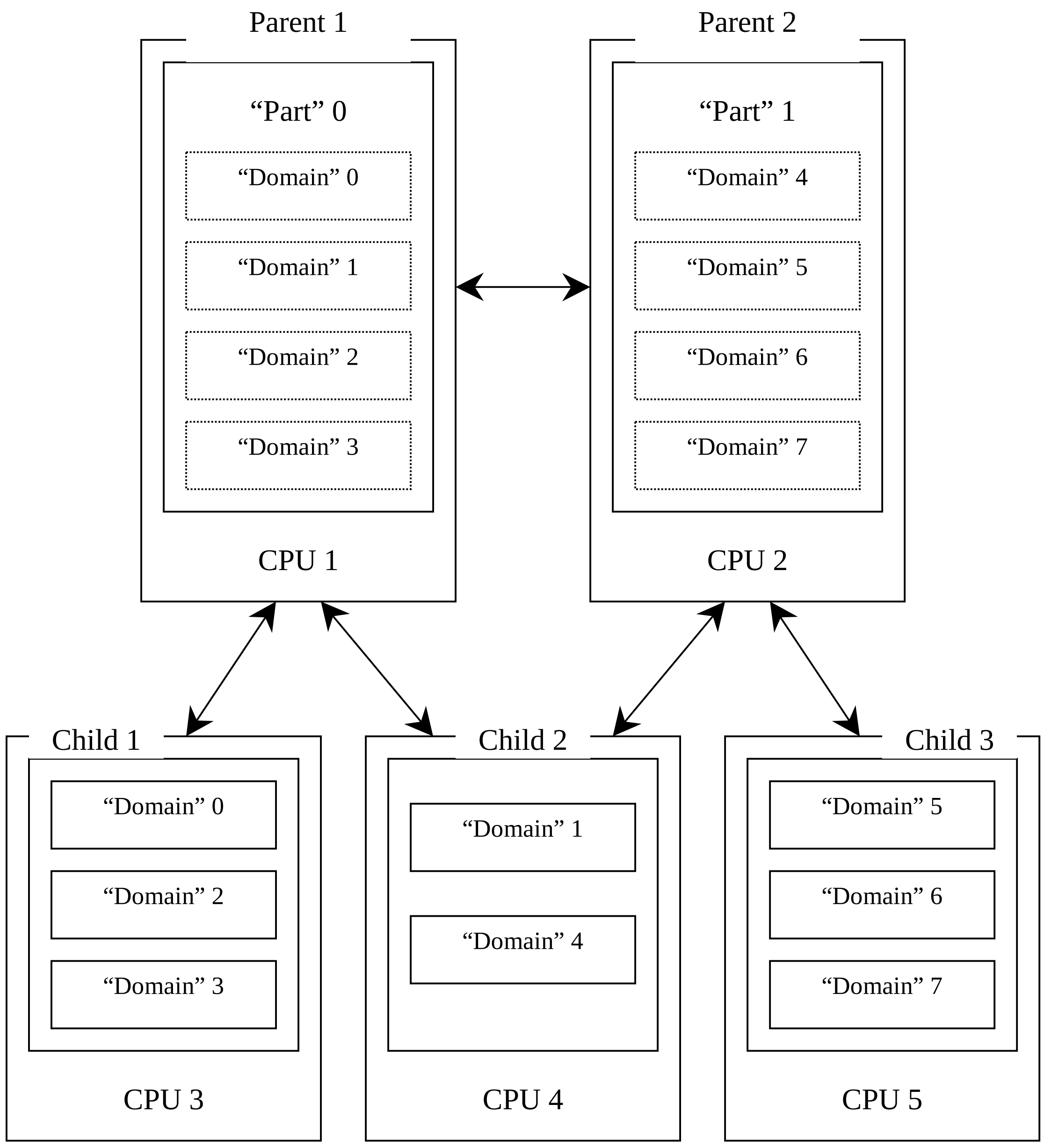

Fig. 5. Adjustment of "Domains" to CPUs (Dynamic load distribution version)

2.2. How to Decide the Number of Domain Decompositions

The previous chapter was devoted to the common features and basic principles of job assigning processes of the ADVENTURE_Solid module depending on the parallel processing environments. Basically, the number of “Parts” should be decided based on the method used for parallel processing, the number of nodes used in a network, and the computing environments. Since, computing time may not be optimal due to the difference in memory consumption by the nodes for the “Parent” and “Child” processes, even if their number is equal, the proper domain decomposition should be done to increase the computational performance.

The number of “Domains” should be decided taken into account the memory used by computation processes. The stiffness matrixes (and their inversed matrixes) are memorized for each “Domain” applying the Skyline method that occupies the largest memory volume. As more detailed domain decomposition is done, as less memory is required.

In the case of using the Static load distribution version of ADVENTURE_Solid, the number of “Parts” and network nodes decides the amount of data transfer; and the USER should not consider about the speed of data transfer.

Combinations of the following methods are used in ADVENTURE_Solid, and the computing time is strongly dependent on the method applied and the number of “Domains”. The Direct method is used to obtain the displacement inside “Domain” and the Iterative CG method is used to obtain the displacement on the boundaries between Domains”. The computing time for 1 CG step will be as shorter as more detailed decomposition of the “Domain” is done, because the ratio of the Direct method use in total combination of methods will be decreased. To reach the convergence using less iteration steps, rough domain decomposition is preferable; otherwise, the ratio of Direct method used will be high that will require more time for computation. The total computing time varies depending on many factors. The number of iteration steps to reach desired convergence depends on boundary conditions. It was found that the convergence occurs faster if many displacement boundary conditions (Dirichlet boundary conditions) are set. From the experience, it can be said that if many Dirichlet boundary conditions are set, detailed domain decomposition will be preferable to decrease the total computing time. However, there is an optimum for the number of decomposed “Domains” and Dirichlet boundary conditions that influence on the computing time.

If very detailed domain decomposition by ADVENTURE_Metis is done, some elements may not be included into “Domain” that will result in errors and termination of the ADVENTURE_Solid module. In order to avoid such situations, the number of elements in one “Domain” after decomposition should be not less than 20.

Good performance can be achieved if the number of elements in one “Domain” lies between 20 and 100. The number of elements in one “Domain” should be small if many displacement boundary conditions are set, and, the number of elements in one “Domain” can be large if few displacement boundary conditions are set. If the “Domain” consists of linear tetrahedral elements, which have the smallest number of nodes, the large number of these elements is preferable.

The number of elements in a “Domain” that should be created by ADVENTURE_Metis module can be calculated using the following equation:

$$n = \frac{N_{element}}{N_{part} \times N_{domain}} \qquad (1),$$

where: n is the number of elements in the considered “Domain” (should be within the interval from 20 to 100),

$N_{element}$ is the total number of elements,

N_{part} is the number of “Parts”,

N_{domain} is the number of “Domains” in the “Part”.

Compared with the static load distribution method, much data transfer between the “Parent” and the “Child” is accomplished in the case of dynamic load distribution method. The volume of data transfer depends on the way of domain decomposition. I.e., as more detailed domain decomposition is done, as much data transfer is required between the processes to adjust the boundaries between and within “Domains”. The speed of data transfer depends on the network environment and generally, the static load distribution method results in better performance for uniform computer environments.

3. Analysis Functions

The ADVENTURE_Solid module can perform the elastic, elasto-plastic, large-displacement and large-strain analyses in nonlinear geometry taking into account material characteristics. The possible combinations of these analyses are:

- Linear elastic analysis;
- Large-displacement small-strain elastic analysis (Total Lagrange method);
- Large-displacement large-strain elastic analysis (Updated Lagrange method);
- Elasto-plastic analysis;
- Large-displacement small-strain elasto-plastic analysis (Total Lagrange method);
- Large-displacement large-strain elasto-plastic analysis (Updated Lagrange method).

These analyses can be performed with the functions described below.

3.1. Common Functions in Analyses

3.1.1. Function of Parallel Processing

The mentioned analyses can be performed by 3 methods depending on computer environments:

- Program execution using a single CPU;
- Program execution using static load distribution method for parallel data processing;
- Program execution using dynamic load distribution method for parallel data processing.

3.1.2. Elements

Four types of solid elements listed below are supported (Appendix A). The model for analysis should contain the same type of elements because their co-existence is not supported in the current version.

- Linear tetrahedral element (1 integral point)
- Quadratic tetrahedral element (4 or 5 integral points)
- Linear hexahedral element (8 integral points, reduced integration related to volumetric strain)
- Quadratic hexahedral element (27 integral points)

3.1.3. Boundary Conditions

The following boundary conditions can be applied:

- Node forced displacement;
- Node concentrated load (the surface load can be converted to the node-concentrated load by the ADVENTURE_BCtool module).

3.1.4. Volumetric Force

It is possible to add the deadweight depending on gravity.

3.2. Functions in Linear Elastic Analysis

3.2.1. Modeling of Material

The following uniform and isotropic material properties can be applied:

- Young’s Modulus;
- Poisson’s Ratio;
- Material Density (if gravity setup option is used).

3.2.2. Output Results

Depending on selected program options for analysis, the following data can be obtained:

- Displacement (node);
- Reaction force (node);
- Stress tensor (element / integration point / node);
- Equivalent stress (element / integration point / node);
- Strain tensor (element / integration point / node).

3.3. Functions in Nonlinear Analysis

The loads and displacements are assumed as increments according to the Strain increment theory to perform the increment nonlinear analyses. Information on formulations of the method can be found in [4] and [7]. The data processing is accomplished by 3 large loops as it is shown in Fig. 6. The outside loop is the loop for the load increment. A repetition by the Newton-Raphson method using the Consistent tangential stiffness is performed in the inside increment loop. It made possible to take comparatively large increment step. CG iterations used in the hierarchical domain decomposition method are performed inside the Newton-Raphson loop.

3.3.1. Elasto-Plastic Analysis Modeling

The bi-linear type stiffening functions for the von Mises yield conditions are applied for modeling in the elasto-plastic analysis. The following material properties can be used in addition to the properties for the elastic analysis:

- Work hardening coefficient;
- Initial yield stress.

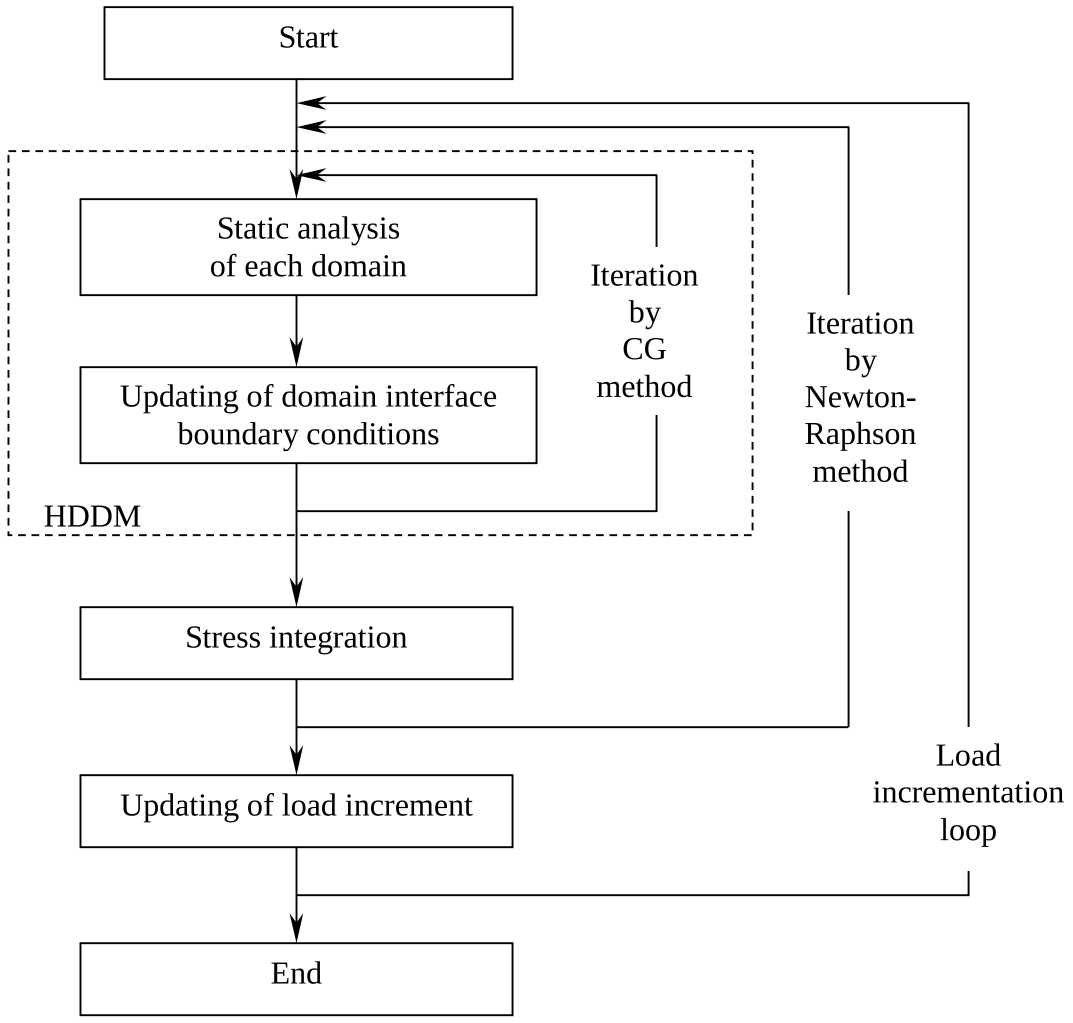
Start
Static analysis of each domain
Updating of domain interface boundary conditions
Iteration by CG method
HDDM
Iteration by Newton-Raphson method
Stress integration
Updating of load increment
Load incrementation loop
End

Fig. 6.   Flowchart of processes in nonlinear analysis

3.3.2. Control of Increment Step

The USER should appropriately set the increment step and the interval width, which can be controlled. Large increment can be set for the initial steps and smaller increment should be set for the further steps. Necessary information can be found in Chapter 5.2.4.

3.3.3. Output Results

In addition to the output of the linear elastic analysis, the following output results can be obtained for each increment step or for the specified increment step:

- Tensor of plasticity strain (element / integration point / node);
- Equivalent plasticity strain (element / integration point / node);
- Yield stress (element / integration point / node);
- Yield domain (element / integration point).

4. Input and Output Data

4.1. Flow of Input/Output Data Processing

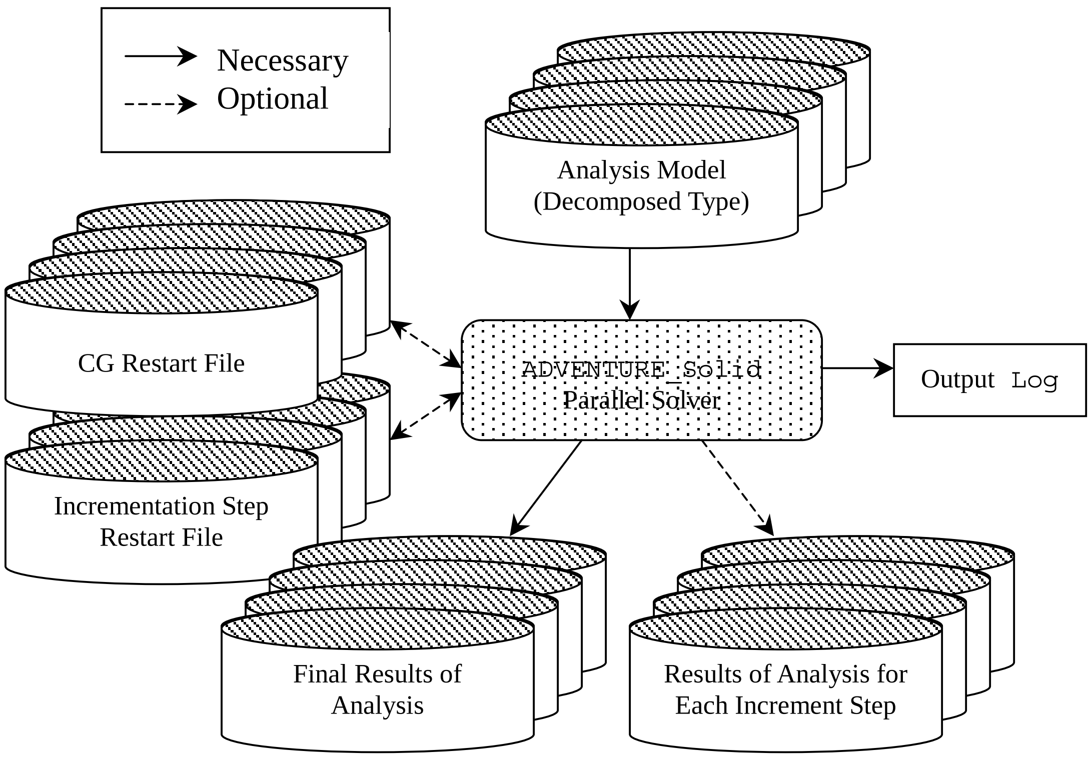

Fig. 7. Flowchart of input/output data processing

The input and output data processing done by the ADVENTURE_Solid module is shown in Fig. 7. Except the output log that can be displayed on the view screen, all data files have the ADVENTURE File format (binary). One data file contains the information about one “Part” of the decomposed analysis model. The files containing the decomposed analysis model, prepared in advance by the ADVENTURE_Metis domain decomposer module, are used as input for the ADVENTURE_Solid module. The ADVENTURE_Solid output files “Analysis Results” contain the information on physical quantities of node displacement and stress. The output of physical quantities can be set by options of the ADVENTURE_Solid module. In the case of nonlinear analysis, the output can be done for each increment step. The output information is stored in the binary ADVENTURE File format form for each of the decomposed “Part”.

The restart function is provided for the cases of limited continuation of execution by saving the data into files and restarting the program from the moment of interruption of the calculations. Two types of restart files can be used: CG Restart File and Increment Step Restart File. CG Restart File should be used to restart the linear elastic analysis, and Increment Step Restart File should be used to restart the nonlinear analysis.

4.2. About the Unit System

Conversion functions of data unit systems are not implemented in the current version of the program; the unit system of the input data should be consistent.

4.3. Processes of Input and Output

The input and output processes can be accomplished in the static work distribution version for all of the processes, or in the dynamic work distribution version for the parent processes. The input and output are consisted of a number of files where each “Part” is treated as one file, and, depending on the process that takes control of the “Part”, the input and output are carried out independently for each “Part”. In order to be used by all processes, the path name excluding the “Part” number in the filename should be the same. In the case of using the parallel computer environment presented by workstation clusters connected by network, to share the files by NFS from each network node the same paths should be set up (for convenience). If it cannot be done, the input files should be copied in advance by *ftp* to each network node where the directory, which can be referred by a common path name, should be prepared.

By default, file input and file output are carried out in the exclusive directory. The input and output processes are not accomplished simultaneously; the access to the hard disk is made one by one in order of a “Part” number. It is designed to eliminate the decreasing of performance by concentrated access to the disk if file sharing is carried out by NFS. However, to make the treated file independent, it is possible to use the local hard disk of each node and parallel access can be done without problems. The parallel file access using local disk can be done by setting up the option “**`-file-para`**” when starting ADVENTURE_Solid module (See Chapter 5.2.7). To use the local hard disk, the analysis model files should be located in the directory with the same path prepared in advance.

4.4. Input Data

As it was mentioned in Chapter 4.1, the domain-decomposed FEM analysis model should be prepared prior to execution of the ADVENTURE_Solid module. These processes are done in the following sequence:

1. Creation of mesh file;
2. Creation of analysis model file with setting the boundary conditions and the material properties;
3. Domain decomposition and creation of the domain-decomposed type analysis model file.

Each procedure will be explained below.

4.4.1. Mesh File

At first, the mesh file of the analyzed object is created in the ASCII format. The linear tetrahedron, quadratic tetrahedron, linear hexahedron, and quadratic hexahedron can be used. However, the mixture of different elements is improper, and the mesh should be altogether created using the same type of elements.

The tetrahedral mesh can be created by using the ADVENTURE_TetMesh module. Refer to the user manual of ADVENTURE_TetMesh for detailed information. Moreover, if the mesh system is prepared by other mesh creation tool, the tetrahedron and hexahedron can be input into the ADVENTURE System. The following example represents the file of linear hexahedral element. In the cube with the edge length of 2, 27 nodes and 2 x 2 x 2 = 8 elements are created.

```
1      8
2          0      1      4      3      9      10     13     12
3          1      2      5      4      10     11     14     13
4          3      4      7      6      12     13     16     15
5          4      5      8      7      13     14     17     16
6          9      10     13     12     18     19     22     21
7          10     11     14     13     19     20     23     22
8          12     13     16     15     21     22     25     24
9          13     14     17     16     22     23     26     25
10
11     27
12        -1.00000000e+00      -1.00000000e+00      -1.00000000e+00
13         0.00000000e+00      -1.00000000e+00      -1.00000000e+00
14         1.00000000e+00      -1.00000000e+00      -1.00000000e+00
15        -1.00000000e+00       0.00000000e+00      -1.00000000e+00
16         0.00000000e+00       0.00000000e+00      -1.00000000e+00
17         1.00000000e+00       0.00000000e+00      -1.00000000e+00
18        -1.00000000e+00       1.00000000e+00      -1.00000000e+00
19         0.00000000e+00       1.00000000e+00      -1.00000000e+00
20         1.00000000e+00       1.00000000e+00      -1.00000000e+00
21        -1.00000000e+00      -1.00000000e+00       0.00000000e+00
22         0.00000000e+00      -1.00000000e+00       0.00000000e+00
23         1.00000000e+00      -1.00000000e+00       0.00000000e+00
24        -1.00000000e+00       0.00000000e+00       0.00000000e+00
25         0.00000000e+00       0.00000000e+00       0.00000000e+00
26         1.00000000e+00       0.00000000e+00       0.00000000e+00
27        -1.00000000e+00       1.00000000e+00       0.00000000e+00
28         0.00000000e+00       1.00000000e+00       0.00000000e+00
29         1.00000000e+00       1.00000000e+00       0.00000000e+00
30        -1.00000000e+00      -1.00000000e+00       1.00000000e+00
31         0.00000000e+00      -1.00000000e+00       1.00000000e+00
32         1.00000000e+00      -1.00000000e+00       1.00000000e+00
33        -1.00000000e+00       0.00000000e+00       1.00000000e+00
34         0.00000000e+00       0.00000000e+00       1.00000000e+00
35         1.00000000e+00       0.00000000e+00       1.00000000e+00
36        -1.00000000e+00       1.00000000e+00       1.00000000e+00
37         0.00000000e+00       1.00000000e+00       1.00000000e+00
38         1.00000000e+00       1.00000000e+00       1.00000000e+00
```

The 1st line represents the total number of elements. From the 2nd to the 9th lines, the element connectivity is shown line by line. The connectivity is expressed by a row of the node numbers, which constitutes the element. In the case of linear hexahedron, eight node numbers are queuing in a line, which sequences are decided for every element. Refer to Appendix A for details. The 11th line represents the total number of nodes. The lines from 12th to 38th represent the *x, y,* and *z* coordinates of each node. Each line is corresponded to the data of one node in order of node number that starts from 0 and counts until the "total node number –1".

4.4.2. FEM Analysis Model (Entire Type)

After creation of the mesh file, the boundary conditions and the material properties can be added to mesh using the ADVENTURE_BCtool module.

The following boundary conditions can be applied for all of the analyses:

- Displacement boundary conditions
  The forced displacement boundary conditions can be applied. It is necessary to set up 6 degrees of freedom (even for the load boundary conditions) to eliminate the rigid body mode.

- Load boundary conditions
  The load boundary conditions can be added to the object. Depending on the face load conditions, the load will be transformed to the node equivalent force.

The following material properties are required for all of the analyses:

- Young’s Modulus (Real number of scalar);
- Poisson’s Ratio (Real number of scalar).

In addition to the values mentioned above, the following material properties are required for elasto-plastic analysis:

- Initial yield stress (Real number of scalar);
- Work hardening coefficient (Real number of scalar).

Moreover, the following parameters should be set up if the gravity data are added:

- Mass density (Real number of scalar);
- Gravity acceleration (Real number of 3-dimensional vector).

The analysis model file is created in the ADVENTURE File format (binary) accepted to present the data used for FEM analysis model in the ADVENTURE Project. The data are handled by units called “ADV Document”. One file may contain several ADV Documents with information on the node coordinates, element connectivity, etc.

Refer to [8] and the USER Manual for detailed information. An example of the data contents supplied with the current package is located in the subdirectory **`sample_data/`**. The USER can use the ADVENTURE_IO module to convert binary data into the text format.

To read and write the data in the ADVENTURE File format, the ADVENTURE_IO module is used as a library in the ADVENTURE Project, as well as by the ADVENTURE_Solid module. To convert the binary data into a text format (*ASCII*), **`advshow`** is supplied with current package. Detailed information can be found in Appendix B.2.

4.4.3. Domain Decomposition of FEM Analysis Model

As it was mentioned previously, the FEM analysis model for ADVENTURE_Solid is prepared using the hierarchical domain decomposition module ADVENTURE_Metis. The ADVENTURE_Metis module is designed for execution in parallel computer environments using MPI. Refer to the ADVENTURE_Metis USER Manual for the details. The created file of decomposed analysis model has also the ADVENTURE File format, but it serves as the format to which some was extended. The example of the data contents supplied in with the current package is located in the subdirectory **`sample_data/`**.

4.5. Analysis Results File

4.5.1. Output Physical Quantities

The analysis model and its appearance are recorded in the ADVENTURE File format for each “Part”. The USER can select physical quantities that will be printed out. In the nonlinear analysis, the data output for each of the increment step is also possible. The physical quantities that can be printed out are summarized in Table 1.

Table 1. Possible output physical quantities

Physical quantity	*Output point*	*Label name*
Displacement	Node	Displacement
Reaction force	Node	ReactionForce
Stress tensor	Element	Stress
Stress tensor	Integral point	Stress@IntegrationPoint
Stress tensor	Node	NodalStress
Equivalent stress	Element	EquivalentStress
Equivalent stress	Integration point	EquivalentStress@IntegrationPoint
Equivalent stress	Node	NodalEquivalentStress
Strain tensor	Element	Strain
Strain tensor	Integration point	Strain@IntegrationPoint
Strain tensor	Node	NodalStrain
Only in the case of elasto-plastic analysis, the following output is possible		
Plastic strain tensor	Element	PlasticStrain
Plastic strain tensor	Integral point	PlasticStrain@IntegrationPoint
Plastic strain tensor	Node	NodalPlasticStrain
Equivalent plastic strain	Element	EquivalentPlasticStrain
Equivalent plastic strain	Integral point	EquivalentPlasticStrain@IntegrationPoint
Equivalent plastic strain	Node	NodalEquivalentPlasticStrain
Yield stress	Element	YieldStress
Yield stress	Integral point	YieldStress@IntegrationPoint
Yield stress	Node	NodalYieldStress
Yield region	Element	PlasticState
Yield region	Integral point	PlasticState@IntegrationPoint

For the yield region, if the stress is on the yield surface, the value is 1, if not, the value is 0.

Since the displacement and reaction force quantities are obtained on a node, they are recorded for the node. The other quantities are internally evaluated by finding an integral point, and the average values for the element are evaluated by arithmetic averaging taken onto account the integral points. Compared to version β-0.8, in the current version, to convert the data into the nodal data values, an extrapolation from the integral point's data to the nodal data of the element is done independently for each element with consequent arithmetical averaging taken into account the values for all connected elements. The element equivalent stress is obtained in the same way.

In the case of conversion of the integral point's data to the element's data in the plasticity region, if all integral points are belonged to the yield surface, the value is 1; if any integral point is found be non-belonged to the yield surface, the value is 0.

The labels listed in Table 1 specify the physical quantities for output. As it was already mentioned, the analysis results have the ADVENTURE File format, where FEGenericAttribute (FEGA) is used as a Document to describe the element's data. HDDM_FEGenericAttribute (HDDM_FEGA) Document is used for the output of the hierarchical domain decomposition. Necessary output physical quantities stored in the HDDM_FEGA Document can be accessed by selecting the labels listed in Table 1. To merge the necessary data, special program **`hddmmrg`** is supplied in the current package; the data can be found by setting the label name in the program's option.

The nodal displacement and nodal equivalent stress are set for the default output. To obtain another quantities, the necessary options should be set. In the case of nonlinear analysis, the output can be done for each increment step and the output interval can be also specified. The output for each step is not specified by the default settings, and, if necessary, the options should be added before the program execution.

Different filenames are used for the step-by-step output result file and the final result file. The output data should be specified separately.

4.5.2. Post Processing of Analysis Results

The obtained results can be visualized by using the ADVENTURE_Visual module. To perform the procedures that are not supported in ADVENTURE_Visual, the program called **`hddmmrg`** is supplied with the current package. By using **`hddmmrg`**, the domain-decomposed-type file with analysis results of the ADVENTURE File format can be merged and converted into entire-type ASCII format with adding the numbers to the nodes and elements. The labels listed in Table 1 should be used to obtain the necessary physical quantities. Refer to Appendix B.1 for the detailed information on **`hddmmrg`** usage.

5. Execution Method

As it was mentioned in Chapter 2, there are three versions of the ADVENTURE_Solid module (the names of executable modules are listed):

- **`advsolid-s`** single version;
- **`advsolid-p`** version for static work distribution in parallel computing environments;
- **`advsolid-h`** version for dynamic work distribution in parallel computing environments.

Depending on the computer environments, one of the listed modules should be used. The single version of the program does not require MPI for compilation and execution. The other two parallel versions require MPI for compilation and execution. The compilation and execution procedures may vary in accordance with the MPI environment. The execution procedures will be described for the most commonly used **`mpich`** [6]. If another MPI package is used, the mismatching part should be adjusted to execute the programs.

The single version of the program can be executed by:

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

In the case of **`mpich`** [6], the parallel version of the program can be executed using **`mpirun`** command:

% **`mpirun`** [*options_for_mpirun*] **`advsolid-p`** [*options*] *data_dir*
or
% **`mpirun`** [*options_for_mpirun*] **`advsolid-h`** [*options*] *data_dir*

Here, [*options_for_mpirun*] are the command options for **`mpirun`** (see Appendix C). If it is considered to use the environment other than **`mpich`**, the part **`mpirun`** [*options_for_mpirun*] should be substituted by commands for the environment used.
[*options*] are the options used by ADVENTURE_Solid. They are basically the same for all three versions of the program and are used for selection of analyses types and other output parameters (will be described afterwards). The last parameter *data_dir* is the path to the top directory where the input/output data files are located.

5.1. Filenames for Input and Output

The top directory *data_dir* where the input-output data files are located, should be specified when the ADVENTURE_Solid module is executed. The following input-output filenames are set up by default:

- Analysis model file:
  data_dir`/model/advhddm_in_`***`P`***`.adv`

- File containing final analysis results:
  data_dir`/result/advhddm_out_`***`P`***`.adv`

- File containing the analysis results by increment step:
  data_dir`/result/advhddm_incrout_`***`S`***`_`***`P`***`.adv`

- CG restart file:
  data_dir`/cg-res/advhddm_cgres_`***`P`***`.adv`

- Increment step restart file:
  data_dir`/incr-res/advhddm_incrres_`***`S`***`_`***`P`***`.adv`

Here ***P*** is the “Part” number and ***S*** is the increment step number.

5.2. Program Options

The following options can be used for program execution.

5.2.1. Specification of Type of Analysis

The following options are used to specify the type of analysis. If no options are added for the program execution, the linear elastic analysis will be performed.

- **`-ep`**
  The option is used to perform the elasto-plastic analysis. It is necessary to specify

the work hardening coefficient and the initial yield stress at the time of model creation.

- **`-tl`**

  The option is used to perform the geometric nonlinear analysis by the Total Lagrange method. It is effective for the elastic and elasto-plastic analyses with large displacement and small strain problems. It cannot be specified together with the **`–ul`** option.

- **`-ul`**

  The option is used to perform the geometric nonlinear analysis by the Updated Lagrange method. It is effective for the elastic and elasto-plastic analyses with large displacement and large strain problems. It cannot be specified together with the **`–tl`** option.

The following combination of options can be applied.

Table 2. Type of analyses and setup options

Type of analysis	*Program options*
Linear elastic analysis	
Large-displacement small-strain elastic analysis	**`-tl`**
Large-displacement large-strain elastic analysis	**`-ul`**
Elasto-plastic analysis	**`-ep`**
Elasto-plastic large-displacement small-strain analysis	**`-ep -tl`**
Elasto-plastic large-displacement large-strain analysis	**`-ep -ul`**

Since the nonlinear analyses use the increment method, it is necessary to specify the increment step except for the linear elastic analysis.

Moreover, it is possible to add the self-load using the following options. In this case, it is necessary to specify the gravity and material density at the time of model creation. The load increment can be controlled in nonlinear analyses by the sub-option -- **`bf-with`** of the option **`–incr-step`** at each of the increment step.

- **`-gravity`**

  The option is used to take into account the gravity forces.

5.2.2. Options Related to Elements

- **`-selective-intg`**
  The option is used to perform the reduction of degree of integration for the volumetric strain in the element integration. It is effective only for linear hexahedral element.

- **`-tet10-integ5`**
  Five-point integration is performed in element integration if the quadratic tetrahedral element is used. Four-point integration is used if no option is specified.

5.2.3. Options for Data Input and Output

The output data on nodal displacement and nodal equivalent stress are recorded by default, but the output for each increment step is not carried out. It is possible to control the necessary output information by the following options:

- **`-result`** [*sub-options*]
  The option is used to specify the data, which will be recorded into the final analysis results file.

- **`-no-result`** [*sub-options*]
  The option is used to specify the data, which will not be recorded into the final analysis results file.

- **`-incr-result`** [*sub-options*]
  The option is used to specify the analysis results data output by increment step.

- **`-no-incr-result`** [*sub-options*]
  The option is used to specify the analysis results data, which will not be recorded by increment step.

The concrete data for output should be specified by the sub-options, which should follow the main option. Two or more sub-options can be specified at once. The following sub-options can be used:

- **`--disp`**

Node displacement

- **`--reac`**
  Node reaction force

- **`--estr`**
  Element equivalent stress

- **`--estr-n`**
  Node equivalent stress

- **`--estr-i`**
  Integral point equivalent stress

- **`--str`**
  Element stress tensor

- **`--str-n`**
  Node stress tensor

- **`--str-i`**
  Integral point stress tensor

- **`--stra`**
  Element strain tensor

- **`--stra-n`**
  Node strain tensor

- **`--stra-i`**
  Integral point strain tensor

- **`--plstra`**
  Element plasticity strain tensor (effective only for elasto-plastic analysis)

- **`--plstra-n`**

Node plasticity strain tensor (effective only for elasto-plastic analysis)

- **`--plstra-i`**
  Integral point plasticity strain tensor (effective only for elasto-plastic analysis)

- **`--eqplstra`**
  Element equivalent plasticity strain (effective only for elasto-plastic analysis)

- **`--eqplstra-n`**
  Node equivalent plasticity strain (effective only for elasto-plastic analysis)

- **`--eqplstra-i`**
  Integral point equivalent plasticity strain (effective only for elasto-plastic analysis)

- **`--ystr`**
  Element yield stress (effective only for elasto-plastic analysis)

- **`--ystr-n`**
  Node yield stress (effective only for elasto-plastic analysis)

- **`--ystr-i`**
  Integral point yield stress (effective only for elasto-plastic analysis)

- **`--eipl`**
  Element yield region (effective only for elasto-plastic analysis)

- **`--eipl-i`**
  Integral point yield region (effective only for elasto-plastic analysis)

For example, in the case when the options "**`-result --disp --str –estr-n --stra-n`**" are specified, the node displacement, the element stress tensor, the node equivalent stress, and the node strain tensor will be recorded into the final analysis results file.

5.2.4. Increment Step Control Options

For the nonlinear analyses, it is necessary to specify an appropriate increment step by the following options:

- **`-incr-step`** *n* [*sub_options*]
  The analysis will be performed for *n* increment steps. By default, for the displacement and load boundary conditions, and for the deadweight (which should be set up by option) the increment of one step will be 1/*n*. It is possible to set this option repeatedly and the increment of the steps will be minced one by one as specified.

This option has the following sub-options that should be placed after **`-incr-step`** *n*.

- **`--bc-width`** *x*
  For the enforced displacement and load boundary conditions, set up in the analysis model file, the factor *x* can be applied to the displacement and the load which will be added at one increment step. If no *x* value is specified, 1/*n* (*n* is the number of steps) value will be accepted.

- **`--bf-width`** *x*
  For the volumetric force (dead mass), the factor *x* can be applied to add it with one increment step. If no *x* value is specified, 1/*n* (*n* is the number of steps) value will be accepted.

- **`--output-interval`** *x*
  In the increment step specified by the option **`-incr-step`**, the output of the analysis results will be done for the specified number of steps. By default, no output will be done.

- **`--output-last`**
  The output of the analysis results will be done for the last step specified by the option **`-incr-step`**. By default, no output will be done.

- **`--resout-interval`** *n*
  In the increment step specified by the option **`-incr-step`**, the output of the restart

file will be done for the specified number of steps. By default, no output will be done.

- **`--resout-last`**
  The output of the restart file will be done for the last increment step specified by the option **`-incr-step`**. By default, no output will be done.

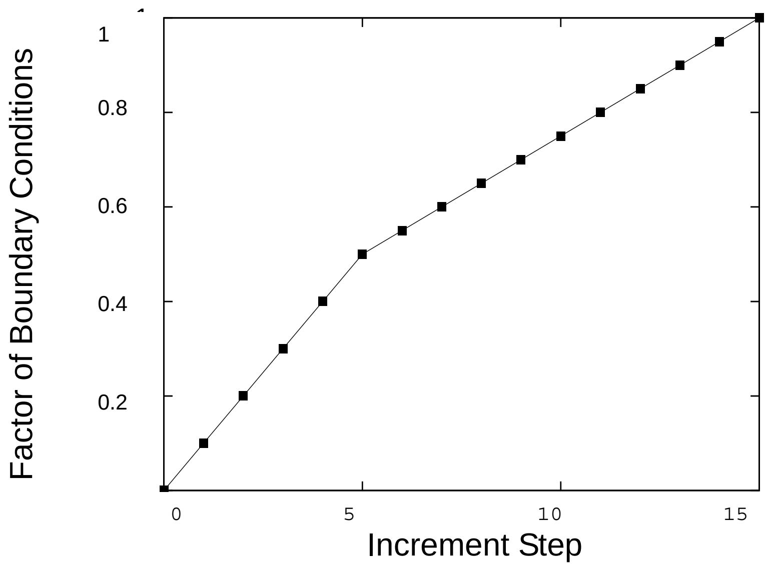

Fig. 8. Example of the increment step setup

The example of setting the options is presented in Fig. 8. If in the elasto-plastic analysis, the displacement and load are added 5 times by a factor of 0.1 from the beginning, and, later, 10 times by a factor of 0.05, the following options should be used: “**`-ep -incr-step 5 --bc-width 0.1 -incr-step 10 --bc-width 0.05`**”. In this case, the vertical axis is corresponded to the factor of load or displacement that will be applied at the increment steps.

It is possible to set up the moments when the results will be recorded into the file. For example, if the displacement and the element stress are recorded once after 5th increment step from the beginning, and for 10 consequent steps by 2-step interval (5 times), the following options should be used:
“ **`-ep -incr-step 5 --bc-width 0.1 --output-last -incr-step 10 --bc-width 0.05 --output-interval 2 -incr-result --disp --str`**“.

The following options should be used to restart the analysis using the Increment Step Restart file:

- **`-use-incr-resin`** *n*
  The analysis can be restarted from the *n* increment step if the output data was recorded into the Increment Step Restart file in advance.

5.2.5. Control Options for Iteration Methods

In order to solve linear equations by global stiffness matrix, the iteration calculations are performed by CG method; and Newton-Raphson method is used for each increment step.

The following options are used in ADVENTURE_Solid to control CG iterations:

- **`-cg-tol`** *x*
  The tolerance *x* to judge about the convergence is specified. The CG residual error obtained at the first step is set for the base value, and the residual errors of CG method, increment step, and Newton-Raphson iterations are compared with the base value. The calculations are supposed to be converged if the relative residual error is smaller than the tolerant value. In the case of nonlinear analysis, it is necessary to set the tolerance of the Newton-Raphson method smaller. The default tolerance is 1.0×10^{-6}.

- **`-cgloop-max`** *n*
  The number of CG iteration steps can be set by *n*. The default value is 10000. There are cases when the default number of iterations is not enough to reach the convergence. Thus, for large-scale analysis, larger number of iterations is recommended.

- **`-nokeep-kmat`**
  By default, the stiffness matrix prepared at the first step of CG iteration is memorized and later used for calculations. Setting this option, the matrix can be prepared at each CG iteration step without consequent memorization. Using this option will permit to decrease the necessary memory size, however the computing time will be increased.

- **`-use-cg-resin`**
  The calculations can be restarted from the step recorded in the CG Restart Input file. This function can be used only in the linear elastic analysis. This function is not effective by default.

- **`-resout-cgstep`** *n*
  The output to the CG Restart file is done for every *n* CG step. No output is done if 0 value in set. By default, no output is performed.

- **`-resout-cglast`**
  The last CG step is recorded into the CG Restart file. The output is done if the convergence is reached or not reached within determined number of iteration steps. By default, no output is performed.

The following options can be used to control the iterations by the Newton-Raphson method:

- **`-newton-tol`** *x*
  The tolerance *x* to judge about the convergence is specified. The CG residual error obtained at the first step is set for the base value, and the residual errors of CG method, increment step, and Newton-Raphson iterations are compared with the base value. The calculations are supposed to be converged if the relative residual error of Newton-Raphson iteration is smaller than the tolerant value. It is necessary to set the tolerance of the Newton-Raphson method larger than that for the CG method. The default tolerance is 1.5×10^{-6}.

- **`-newton-max`** *n*
  The maximum number of iterations can be set. The default value is 10. If the convergence is not obtained within the specified number of steps, larger limit should be specified to overcome the problem. Thought, more detailed breakdown of the increment step is better than increase of the number of iterations.

5.2.6. Options for Changing of Input and Output Filenames

As it was mentioned in Chapter 5.1 for the basic method of setting the environment for the input and output data files, it is specified by default that they should be located in the top directory. If it is necessary to change the location of the input and output data files, the following options should be used. Here, ***`P`*** corresponds to the number of "Part" and ***`S`*** corresponds to the number of increment step.

- **`-model-file`** *file*
  The input analysis model filename is assigned by *file*. The default filename is **`advhddm_in`**. The extension **`_`*`P`*`.adv`** will be added to the filename automatically.

- **`-model-dir`** *dir*
  The name of the subdirectory containing the input analysis model files is assigned by *dir*. The default subdirectory name is **`model`**.

- **`-result-file`** *file*
  The final output analysis results filename is assigned by *file*. The default filename is **`advhddm_out`**. The extension **`_`*`P`*`.adv`** will be added to the filename automatically.

- **`-result-dir`** *dir*
  The name of the subdirectory containing the output analysis results files is assigned by *dir*. The default subdirectory name is **`result`**.

- **`-incr-result-file`** *file*
  The filename of the output analysis results for increment step is assigned by *file*. The default filename is **`advhddm_incrout`**. The extension ***`S`*`_`*`P`*`.adv`** will be added to the filename automatically.

- **`-incr-result-dir`** *dir*
  The name of the subdirectory containing the output analysis results files for increment step is assigned by *dir*. The default subdirectory name is **`result`**.

- **`-incr-resin-file`** *file*
  The filename of the input increment step restart file is assigned by *file*. The default

filename is **`advhddm_incrres`**. The extension ***`S`*`_`*`P`*`.adv`** will be added to the filename automatically.

- **`-incr-resin-dir`** *dir*
  The name of the subdirectory containing the increment step restart files is assigned by *dir*. The default subdirectory name is **`incr-res`**.

- **`-incr-resout-file`** *file*
  The filename of the output increment step restart file is assigned by *file*. The default filename is **`advhddm_incrres`**. The extension ***`S`*`_`*`P`*`.adv`** will be added to the filename automatically.

- **`-incr-resout-dir`** *dir*
  The name of the subdirectory containing the output increment step restart files is assigned by *dir*. The default subdirectory name is **`incr-res`**.

- **`-cg-resin-file`** *file*
  The filename of the input CG restart file is assigned by *file*. The default filename is **`advhddm_cgres`**. The extension **`_`*`P`*`.adv`** will be added to the filename automatically.

- **`-cg-resin-dir`** *dir*
  The name of the subdirectory containing the input CG restart files is assigned by *dir*. The default subdirectory name is **`cg-res`**.

- **`-cg-resout-file`** *file*
  The filename of the output CG restart file is assigned by *file*. The default filename is **`advhddm_cgres`**. The extension **`_`*`P`*`.adv`** will be added to the filename automatically.

- **`-cg-resout-dir`** *dir*
  The name of the subdirectory containing the output CG restart files is assigned by *dir*. The default subdirectory name is **`cg-res`**.

5.2.7. Other Options

In addition, there are other options:

- **`-file-para`**
  The option is used to set all processes to be done in parallel (see Chapter 4.3).

- **`-memlimit`** *n*
  The option is used to set up the memory limit *n* (in Mbytes). The program will be terminated if the limit will be overcome. Although, the actual memory necessary for the large-scale analysis in unknown, it is possible to set up the memory limit for one process to prevent disk swapping. In fact, the amount of memory is secured dynamically and the actual memory used by processes will be not equal.

- **`-help`** or **`-h`**
  The main help messages can be displayed.

- **`-help-output`**
  The help messages for output data addresses can be displayed.

- **`-help-incr`**
  The help messages for the increment step specification can be displayed.

- **`-help-iter`**
  The help messages for CG method control options and Newton-Raphson method control options can be displayed.

5.3. Script `advsolid` for ADVENTURE_Solid Execution

In addition to three executable modules: **`advsolid-s`**, **`advsolid-p`**, and **`advsolid-h`**, the script file **`advsolid`** is supplied for convenience.

- The options can be specified in the setup file.

- It is unnecessary to specify the full path to the executable module in the command line to execute the parallel versions **advsolid-p** and **advsolid-h** using **mpirun** of **mpich**.

- All programs: the single version, the static load distribution version, and the parallel load distribution version can be started using the supplied script file **advsolid**.

The actual way of program execution is:

% advsolid [-show] [-log *logfile***] [-single|-para|-parahddm]**
[*options_for_mpirun***] [-conf** *conffile*| **--] [***solver_options***]**
[*data_dir***]**

The meanings of each option are:

- **-show**
  The option is used to show what will be executed without actual execution.

- **-log** *logfile*
  The output information that will be displayed on the monitor will be recorded also into the output file with filename defined by *logfile*.

- **-single |-para |-parahddm**
  The options are used to execute the single version, the static load distribution version, and the dynamic load distribution version of the program, correspondingly.

- *options_for_mpirun*
  The options used by **mpirun** should be specified.

- **-conf** *conffile*|**--**
  The settings are read from the file defined by *conffile*. The options for ADVENTURE_Solid should be specified after "--" if no setup file is used.

- *solver_options*
  The options for ADVENTURE_Solid discussed in Chapter 5.2 can be set here.

- *data_dir*
  The top directory where the analysis model data are located should be specified here (described in Chapter 5.2) if no setup file is used.

All of the previously listed options except **`-conf`** *conffile* can be set in the setup file *conffile*. The setup file is read by **`advsolid`** as a *Bourne shell* script, where the options are defined as variables:

- **`MODE`**
  The option **`single`** should be used to execute the single version of the program. To execute the static load distribution version of the program, the option **`para`** should be used. To execute the dynamic load distribution version of the program, the option **`parahddm`** should be used. If any other characters will be typed in this line, the priority will be given to the option, typed in the command line. If no option is specified, the single version of the program will be executed.

- **`MPIRUN`**
  In the case of using parallel computer environments, the MPI execution command can be specified. For **`mpich`** the default is **`mpirun`**. For that MPI which has no such execution command a blank space should be left after "**`MPIRUN=`**". No option in the command line for this case is required.

- **`MPIOPTS`**
  The MPI execution options for parallel computer environments can be specified. The default is blank. No settings are necessary for that MPI which has no such execution command. If the variables are simultaneously set by **`MPIOPTS`** and by *options_for_mpirun* in the command line, it will result in the following setting: "$**`MPIOPTS`** *options_for_mpirun*" (both options written in the command line and in the setup file will be sequentially applied).

- **`LOGFILE`**
  The log data appeared during program execution on the monitor can be saved to the file, which name can be defined here. No settings are necessary if no log is going to be saved. If the filename of the output log is defined in the command line by the option **`-log`** *logfile*, the priority will be given to the filename defined in the command line. By default, the log file will not be created.

- **PROGOPTS**
  The options used by the ADVENTURE_Solid module discussed in Chapter 5.2 can be defined here. If the variables are simultaneously set by **PROGOPTS** and by *solver_options* in the command line, it will result in the following setting: “$**PROGOPTS** *options_for_mpirun*“ (both options written in the command line and in the setup file will be sequentially applied). By default, no options are specified.

- **DATADIR**
  The name of the top directory containing analysis data discussed in Chapter 5.2 can be specified here. If the variables are set by **DATADIR** and by *data_dir* in the command line simultaneously, the priority will be given to the options specified in the command line. This option must be specified in either way.

An example of the setup file is given below. In this case, the elastic analysis is performed by the static load distribution version in two network nodes. The node displacement and equivalent stress are recorded by default. The outputs of the element stress tensor and the element strain tensor are specified by **PROGOPTS**. The data will be stored in the directory **cube_p2d2**.

```
##### set parallel mode #######################
MODE=para
##### program name of mpirun ##################
MPIRUN=/usr/bin/mpirun
##### options for mpirun #####################
MPIOPTS="-np 2"
##### set if you want save log to file ########
LOGFILE="run.log"
##### Options for AdvSolid ###################
PROGOPTS="-result --str --stra"
##### Data directory to be analyzed ###########
DATADIR=cube_p2d2
```

the program execution command will be as follows if the setup file is named **`advsolid.conf`**:

```
% advsolid -conf advsolid.conf
```

If it is necessary to change only the directory name from **`cube_p2d2`** to **`another_model`** without changing the setup file, the execution command can be:

```
% advsolid -conf advsolid.conf another_model
```

6. Program Compilation and Installation

6.1. Program Compilation

The *C* compiler, the MPI environment, and the ADVENTURE_IO module are necessary to compile the ADVENTURE_Solid module. In the environments without MPI, MPI should be installed prior to the ADVENTURE_Solid module compilation (MPICH [6] can be used). If only single version of the ADVENTURE_Solid module is supposed to be used, the MPI environment is not required. The ADVENTURE_IO module should be installed and compiled before the compilation of the ADVENTURE_Solid module. Since the tool programs for reprocessing of log files are written by *PERL*, it is preferable that *PERL* being installed, thought it not essential for the compilation of the ADVENTURE_Solid module.

The following procedure is accepted for the compilation of the ADVENTURE_Solid module:

1. **`./configure [options]`**

2. **`make`**

Both of commands should be executed from the top directory of ADVENTURE_Solid. By executing the shell script **`configure`**, the necessary computing environment will be created and recorded into the file **`Makefile`**.

The following options can be used with **`configure`** command. The absolute path to the directories should be specified.

- **`--with-adv=`***directory*
  The option is used to define the top directory of ADVENTURE_IO

- **`--with-mpicc=`***command*
  The option is used to define the name of *C* compiler for MPI. The default is **`mpicc`**. Parallel versions of the ADVENTURE_Solid module 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*
  The 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 option described below should be used.

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

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

default directory is **`/usr/local`**.

The following environmental variables can be applied to change the *C* compiler used for the single program version and the common part of the single and parallel program versions. These options should be set prior to execution of the shell script **`./configure`**.

- **`CC`**
  The option is used to define the name of *C* compiler.

- **`CFLAGS`**
  The option is used to define the flags for *C* compiler.

- **`LIBS`**
  The option is used to specify the necessary libraries to be linked.

For example, the following statements can be used in the case of *C shell*:

```
% setenv CC /usr/local/bin/cc
% setenv CFLAGS "-O2 -g -Wall"
% ./configure
```

The following statements can be used in the case of *Bourne shell*:

```
$ CC=/usr/local/bin/cc
$ export CC
$ CFLAGS="-O2 -g -Wall"
$ export CFLAGS
$ ./configure
```

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`** and the necessary changes in each **`Makefile`** should be done in accordance with the concrete computational environment. The **`make`** command should be executed in each directory contained **`Makefile`**. The files located in the directory **`libfem/`** should be compiled before the files located in the directory **`solver/`**.

6.2. Program Installation

Only executable files will be installed. If the compilation is done using the configure script, the following command should be executed from the top directory:

```
% make install
```

The created executable modules will be installed in the directory defined by *install_directory*. If **`configure`** is not used and the compilation is done by adjusting **`Makefile`**, the following command should be executed from the directories contained **`Makefile`** except the directory **`libfem`**:

```
% make install
```

The destination directory for installation can be defined by **`INSTALL_BINDIR`** in **`Makefile`**. The default installation directory is **`bin`** located under the top directory.

Appendix

A. Allowable Types of Elements

ADVENTURE_Solid can use four types of elements shown in Table 3. The existence of different types of elements is not supported and only one type of element can be used in one analysis model.

Table 3. Allowable types of elements

Element type	*Number of nodes*	*Number of integration points*
Linear tetrahedron	4	1
Quadratic tetrahedron	10	4 (5)
Linear hexahedron	8	8
Quadratic hexahedron	20	27

A.1. Linear Tetrahedral Element

1. Node

The number of nodes is 4. The arrangement of the node numbers in the elements connectivity is shown in Fig. 9.

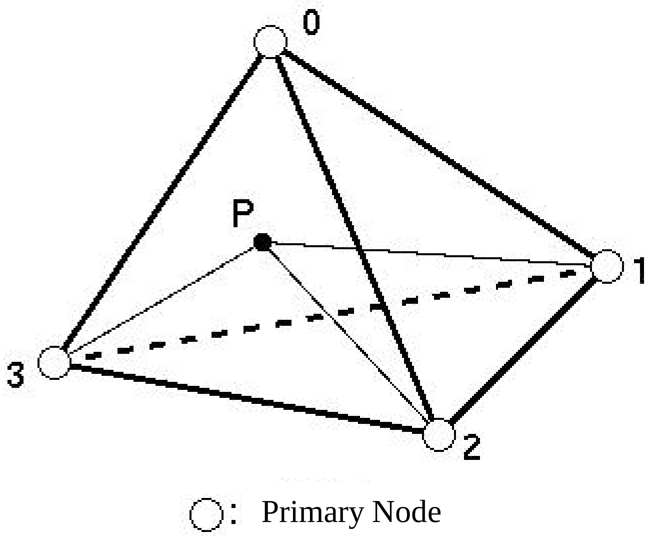

Fig. 9. Linear tetrahedral element

2. Integration Points

The number of integration points is 1. The integration point uses the volumetric coordinates (L_0, L_1, L_2, L_3) presented in Table 4. The point **P** shown in Fig. 9 has the following coordinates.

L_0 = Volume of tetrahedron P123 / Volume of tetrahedron 0123 (2)
L_1 = Volume of tetrahedron P023 / Volume of tetrahedron 0123 (3)
L_2 = Volume of tetrahedron P013 / Volume of tetrahedron 0123 (4)
L_3 = Volume of tetrahedron P012 / Volume of tetrahedron 0123 (5)

Table 4. Integration points of linear tetrahedral element

Integration 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. Node

The number of nodes is 10. The arrangement of the node number in the elements connectivity is shown in Fig. 10.

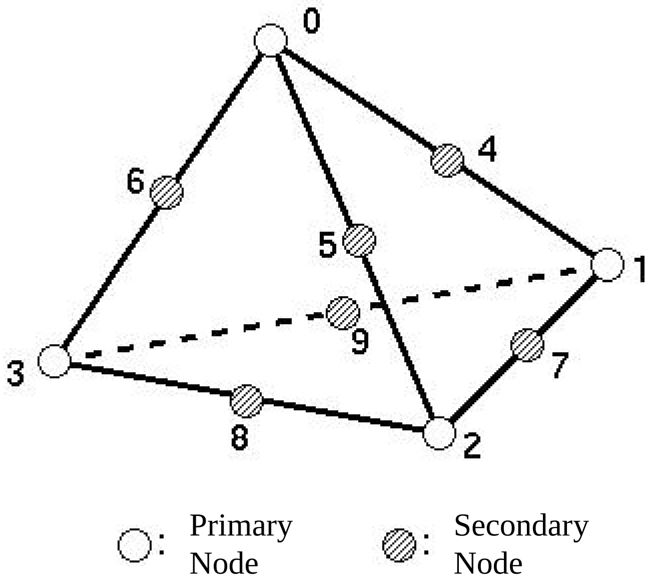

Fig. 10. Quadratic tetrahedral element

2. Integration Points

The default number of integration points is 4. The integration points use the volumetric coordinates (L_0, L_1, L_2, L_3) presented by equations (2) ~ (5) and Fig. 9. The integration points of quadratic tetrahedral element are shown in Table 5. The a and b values in Table 5 are:

$$\alpha = 0.58541019662496845446$$
$$\beta = 0.13819660112501051518$$

Table 5. Integration points of quadratic tetrahedral element (4 points)

Integration 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

The number of integration points can be set to 5 by using the option at the time of program execution. The volumetric coordinates for this case are presented in Table 6.

Table 6. Integration points of quadratic tetrahedral element (5 points)

Integration 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	1/6	1/6	1/6

A.3. Linear Hexahedral Element

1. Node

The number of nodes is 8. The arrangement of the node number in the elements connectivity is shown in Fig. 11.

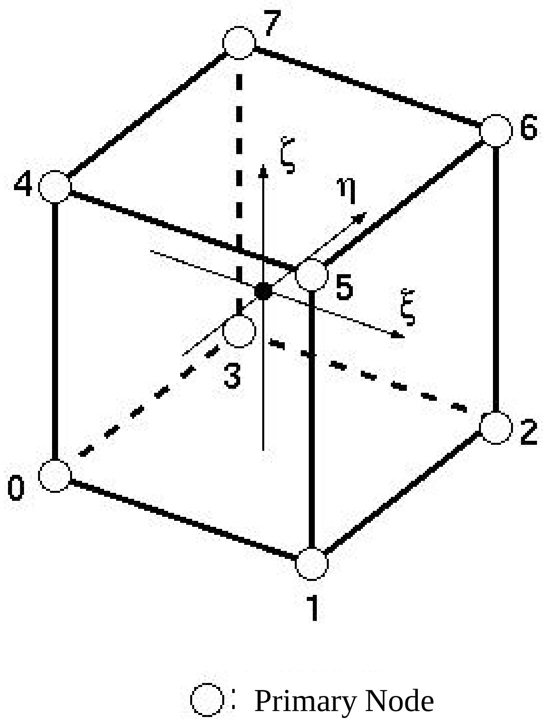

Fig. 11. Linear hexahedral element

2. Integration Points

The number of integration points is 8. The integration points use the normalized coordinates (ξ, η, ζ) ($-1 < \xi, \eta, \zeta < 1$) presented in Table 7.

Table 7. Integration points of linear hexahedral element

Integration point number	ξ	η	ζ
0	$-1/\sqrt{3}$	$-1/\sqrt{3}$	$-1/\sqrt{3}$
1	$1/\sqrt{3}$	$-1/\sqrt{3}$	$-1/\sqrt{3}$
2	$-1/\sqrt{3}$	$1/\sqrt{3}$	$-1/\sqrt{3}$
3	$1/\sqrt{3}$	$1/\sqrt{3}$	$-1/\sqrt{3}$
4	$-1/\sqrt{3}$	$-1/\sqrt{3}$	$1/\sqrt{3}$
5	$1/\sqrt{3}$	$-1/\sqrt{3}$	$1/\sqrt{3}$
6	$-1/\sqrt{3}$	$1/\sqrt{3}$	$1/\sqrt{3}$
7	$1/\sqrt{3}$	$1/\sqrt{3}$	$1/\sqrt{3}$

A.4. Quadratic Hexahedral Element

1. Node

The number of nodes is 20. The arrangement of the node number in the elements connectivity is shown in Fig. 12.

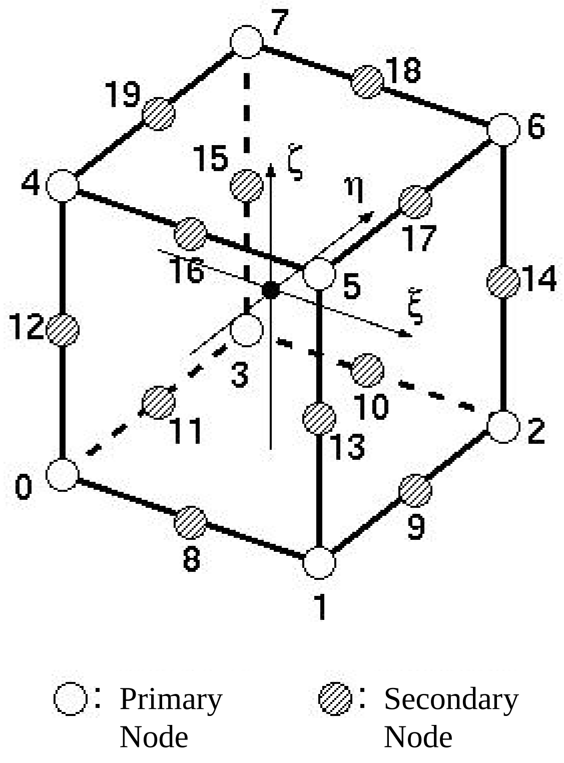

Fig. 12. Quadratic hexahedral element

2. Integration Points

The number of integration points is 27. The integration points use the normalized coordinates (ξ, η, ζ) ($-1 < \xi, \eta, \zeta < 1$) presented in Table 8.

Table 8. Integration points of quadratic hexahedral element

Integration point number	ξ	η	ζ
0	$-\sqrt{3/5}$	$-\sqrt{3/5}$	$-\sqrt{3/5}$
1	0	$-\sqrt{3/5}$	$-\sqrt{3/5}$
2	$\sqrt{3/5}$	$-\sqrt{3/5}$	$-\sqrt{3/5}$
3	$-\sqrt{3/5}$	0	$-\sqrt{3/5}$
4	0	0	$-\sqrt{3/5}$
5	$\sqrt{3/5}$	0	$-\sqrt{3/5}$
6	$-\sqrt{3/5}$	$\sqrt{3/5}$	$-\sqrt{3/5}$
7	0	$\sqrt{3/5}$	$-\sqrt{3/5}$
8	$\sqrt{3/5}$	$\sqrt{3/5}$	$-\sqrt{3/5}$
9	$-\sqrt{3/5}$	$-\sqrt{3/5}$	0
10	0	$-\sqrt{3/5}$	0
11	$\sqrt{3/5}$	$-\sqrt{3/5}$	0
12	$-\sqrt{3/5}$	0	0
13	0	0	0
14	$\sqrt{3/5}$	0	0
15	$-\sqrt{3/5}$	$\sqrt{3/5}$	0
16	0	$\sqrt{3/5}$	0
17	$\sqrt{3/5}$	$\sqrt{3/5}$	0
18	$-\sqrt{3/5}$	$-\sqrt{3/5}$	$\sqrt{3/5}$
19	0	$-\sqrt{3/5}$	$\sqrt{3/5}$
20	$\sqrt{3/5}$	$-\sqrt{3/5}$	$\sqrt{3/5}$
21	$-\sqrt{3/5}$	0	$\sqrt{3/5}$
22	0	0	$\sqrt{3/5}$
23	$\sqrt{3/5}$	0	$\sqrt{3/5}$
24	$-\sqrt{3/5}$	$\sqrt{3/5}$	$\sqrt{3/5}$
25	0	$\sqrt{3/5}$	$\sqrt{3/5}$
26	$\sqrt{3/5}$	$\sqrt{3/5}$	$\sqrt{3/5}$

B. Tools

In addition to the ADVENTURE_Solid module, the following tools are included in the current package.

B.1. Converter of Analysis Results to Entire Type Data (`hddmmrg`)

The program **`hddmmrg`** is designed to merge the domain-decomposed ADVENTURE Fomat files containing the analysis results data to the text data file. Since the output format is simple, the USER can process analysis results. Such data reprocessing is not necessary for visualization of the analysis results obtained by the ADVENTURE_Visual module.

The following command is used to execute **`hddmmrg`**:

`%` **`hddmmrg`** [*options*] *label* *data_dir*

Here, *data_dir* is the top directory where the analysis results of the domain-decomposed model are located. It is the same parameter as used for **`advsolid`** execution. *label* is the identification name of the file which will be created after reprocessing the data by **`hddmmrg`**. *label* should be selected from the names listed in Table 1 depending on the output done by ADVENTURE_Solid. Selective data can be recorded from the analysis results file by setting the necessary name of label. **`hddmmrg`** treats only one type (one label) of the output results data at one execution. If several results are going to be merged, **`hddmmrg`** should be executed for each type of the data.

By default, the analysis model file is *data_dir*/**`model`**/**`advhddm_in_`*`P`*`.adv`** and the analysis results file is *data_dir*/**`model`**/**`advhddm_out_`*`P`*`.adv`**, where ***`P`*** is the “Part” number.

The following options can be used with **`hddmmrg`**:

- **`-modelfile`** *file*
  The domain-decomposed type analysis model file created by ADVENTURE_Metis is set to *file* (the filename should be specified up to **`_`*`P`*`.adv`**).

- **-resultfile** *file*
  The filename of domain-decomposed analysis results file created by ADVENTURE_Solid is specified (the filename should be specified up to **_*P*.adv**). By default, it is **result/advhddm_out**.

- **-itemlist** *file*
  The option is used to specify the file from which the specific node(s) or element(s) data (set by a node number or an element number) will be read. By default, all information will be recorded into the file. The format of file is *ASCII*. The fist line in the file represents the number of nodes (elements) for output, and from the second line, the node (element) numbers are listed.

- **-h**
  The option is used to display help messages.

It is possible to create a list of data labels contained in the analysis results file by the following command:

% hddmmrg [*options***] -showlabel** *data_dir*

The data file can be set up using the option **–resultfile** *file*.

The format of the analysis results data merged by **hddmmrg** is presented below.

```
 1   label=Displacement
 2   num_items=125
 3
 4     0:    0.00000000e+00    0.00000000e+00     0.00000000e+00
 5     1:   -1.96064988e-06   -1.96064988e-06    -2.77081012e-06
 6     2:   -7.69281443e-07   -1.93681695e-06    -2.30353339e-06
 7     3:   -4.05759629e-21   -1.97623614e-06    -2.21997832e-06
 8     4:    7.69281443e-07   -1.93681695e-06    -2.30353339e-06
 9     5:    1.96064988e-06   -1.96064988e-06    -2.77081012e-06
10     6:   -1.93681695e-06   -7.69281443e-07    -2.30353339e-06
11
```

Here, the label name is displayed in the 1st line; the number of selected nodes (elements) is displayed in the 2nd line. The merged data are displayed from the 4th line where one line contains the data (displacement in this case) for one node (element) started with its number. One value will be displayed for the scalar type data, and three values in order of *x*, *y*, and *z* will be displayed for the vector type data. Thus, for the stress or strain tensors, six values are displayed in order of *xx*, *yy*, *zz*, *xy*, *yz*, *zx*.

For the data at the integral point, one line of the file contains the information on one element. The label name is displayed in the 1st line; the number of selected integral points is displayed in the 2nd line. The merged data are displayed from the 4th line where one line contains the data (in order presented in Appendix A) for one integral point started with element's number. The stress components $\sigma_{i,xx}$, etc. of the element are shown for each *i* integral point.

```
 1  label=Stress@IntegrationPoint
 2  num_items=64
 3
 4   0:        $\sigma_{0,xx}$ $\sigma_{0,yy}$ ... $\sigma_{0,zx}$    ... $\sigma_{1,xx}$ $\sigma_{1,yy}$    ... $\sigma_{1,zx}$
 5   1:        $\sigma_{0,xx}$ $\sigma_{0,yy}$ ... $\sigma_{0,zx}$    ... $\sigma_{1,xx}$ $\sigma_{1,yy}$    ... $\sigma_{1,zx}$
 6   2:        $\sigma_{0,xx}$ $\sigma_{0,yy}$ ... $\sigma_{0,zx}$    ... $\sigma_{1,xx}$ $\sigma_{1,yy}$    ... $\sigma_{1,zx}$
 7   3:        $\sigma_{0,xx}$ $\sigma_{0,yy}$ ... $\sigma_{0,zx}$    ... $\sigma_{1,xx}$ $\sigma_{1,yy}$    ... $\sigma_{1,zx}$
 8                    .                          .                    .
 9                    .                          .                    .
10                    .                          .                    .
11                    .                          .                    .
```

B.2. ADVENTURE Format File Viewer (`advshow`)

The program **`advshow`** supplied in the current package can be used to convert the files from the ADVENTURE File format (binary) to text format (*ASCII*). The following command should be used:

`% advshow` [*options*] *file1...*

The file for conversion should be specified by *file1*. Two or more filenames can be specified. A standard output is defined by default.

The following options can be used with **`advshow`**:

- **`-o`** *file*
  The output will be done to the file specified by *file* instead of standard output file.

- **`-p`**
  Only the Property part will be printed out (parts containing the data will not be printed out).

- **`-h`**
  The help messages will be displayed. The statement *file1*… is not necessary with this option.

B.3. Analyzer (`log2*`) of Log File Created by `advsolid`

The script files **`log2cnv-cg`**, **`log2cnv-nr`**, and **`log2info`** for operating with output logs created by the ADVENTURE_Solid module are supplied in the current package. Since these programs are written by *PERL*, *PERL* should be installed in advance.

- **`log2cnv-cg`**
  The script is used to convert the log data into the format that can be easily used to plot graphs. After conversion, each line of the file contains the data at the current step for:
  1). The number of CG iteration step;
  2). The relative residual error for the current step;
  3). The absolute residual for the current step;
  4). The computing time (dimension: seconds).

- **`log2cnv-nr`**
  The script is used to convert the log data into the format that can be easily used to plot graphs for Newton-Raphson iterations. After conversion, each line of the file contains the data at the current step for:
  1). The number of CG iteration step;
  2). The relative residual error for the current step;
  3). The absolute residual for the current step;
  4). The computing time (dimension: seconds);
  5). The number of Newton-Raphson iteration step.

- **`log2info`**
  The script is used to convert the log data into a brief summary. The data on memory used for calculations and total computing time will be printed out.

Two methods of log file input are common for all 3 scripts: standard input from a keyboard and input from a file. When input is done from a file, the filename should be specified as an argument for each script. In order to save the reprocessed log to the file, redirection functions or the option **`-log`** of the **`advsolid`** script should be used.

For example, to plot the CG residual by **`gnuplot`** using the log, which was saved as **`run.log`** after execution of ADVENTURE_Solid, the following commands can be used:

```
% log2cnv-cg run.log > cgconv.dat
% gnuplot
gnuplot> set logscale y
gnuplot> plot "cgconv.dat" using 1:2
```

Y-axis will be displayed in logarithmic scale. The graph for relative residual error versus CG steps can be obtained. In the last line, the statement “1:2” is corresponded to the relative residual error vs. CG step. It can be changed to “1:3” to plot the absolute residual error vs. CG step, or to “4:2” to plot the relative residual error vs. computing time.

C. Method of Using MPICH

The versions of ADVENTURE_Solid for parallel computing **`advsolid-p`** and **`advsolid-h`** use parallel libraries MPI [5]. Thought, there are a variety of parallel mounting systems in MPI, the method of using **`mpich`** [6], which was employed at the time of module creation, will be discussed. **`mpich`** is a free software for network connection of computers which supports a lot of platforms. It can be installed into the environments, where no MPI is prepared. The parallel versions of program can be executed if at least 2 PCs are connected by network.

Here, the execution method will be described on workstations connected in parallel by network (it is called **`ch_p4`** device in MPICH). Refer to the manual for **`mpich`** to obtain detailed information.

C.1. Preparation

`mpich` uses the UNIX command **`rsh`** to execute the programs on a remote host. The user should properly prepare a file **`.rhosts`** in the home directory to execute the programs in parallel mode. If NFS is used and the home directory can be accessed by all hosts, the file **`.rhosts`** can be prepared only in the shared directory.

The host names and the user names should be written in the **`.rhosts`** line-by-line. For example if the user name is **`user`** and the hosts names are **`host0`**, **`host1`**, **`host2`**, and **`host4`**, the content of the file will be look like:

```
host0  user
host1  user
host2  user
host3  user
host4  user
```

It is possible to define the hosts and user for total system by the file **`/etc/hosts.equiv`**. It will also permit to use **`.rhosts`**, but this file is not necessary to be prepared.

C.2. Execution

The following command should be used to execute the program *program* in parallel mode using MPICH from the command prompt:

% **`mpirun`** [*options_for_MPICH*] *program* [*options_for_program*]

Among a number of options for MPICH (refer to the **`mpich`** manual for details), the following options are frequently used:

- **`-np`** *number_of_hosts*
  The option is used to specify the number of hosts to be involved.

- **`-machnefile`** *machine_file*
  The option is used to change the default host, which performs parallel computing by the host written in the file *machine_file*.
  For example:

```
host0
host1
host2
host3
host4
```

Since the parallel execution is assigned sequentially from the host which was able to start (from the first host of this file), the actual number of hosts and the number of hosts written in this file in not necessary to be equal.

References

[1]. ADVENTURE Project Home Page: *http://adventure.q.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), 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. Yoshimura and G. Yagawa: Massively Parallel Elasto-Plastic Finite Element Analysis Using the Hierarchical Domain Decomposition Method, Transactions of Japan Society of Mechanical Engineers (JSME), 65-A, No. 634 (1999), 1201-1208 (in Japanese).

[5]. MPI Home Page: *http://www-unix.mcs.anl.gov/mpi/*

[6]. MPICH Home Page: *http://www-unix.mcs.anl.gov/mpi/mpich*

[7]. T. Hisada and H. Noguchi: Nonlinear Finite Element Method: Fundamentals and Applications, Maruzen, (1995) (in Japanese).

[8]. T. Miyamura, S. Tanaka, H. Takubo, S. Yoshimura and G. Yagawa: Standardization of Input/Output Data in Large Scale Parallel Computational Mechanics System, Internet Transactions of Japan Society for Computational Engineering and Science (JSCES), No. 20000028 (2000) (in Japanese). *http://homer.shinshu-u.ac.jp/jsces/trans/trans2000/No20000028.pdf*