ppOpen-HPC:
Open Source Infrastructure for Development and Execution of Large-Scale Scientific Applications on Post-Peta-Scale Supercomputers with Automatic Tuning (AT).

ppOpen-APPL/FEM software

ppohFEM

ver. 1.0.1

User's guide

License

Change History

The change history lists the changes from version to version in the ppohFEM source code. We update this section as we add new features. Note that we tend to update the user's guide at the same time we make changes to ppohFEM.

Changes in release 1.0.0
Functionality added or changed:

Fix APIs of Fortran version.

Contents

1. Introduction 5
2. How to install the application 5
  2.1. Software required for installation 5
  2.2. Unzipping the archive file 5
  2.3. Preparation for compilation. 6
  2.4. Build the library, sample program and utilities 6
  2.5. Running the sample program and utilities. 6
3. System overview 7
  3.1. Overview 7
  3.2. Structure of the ppOpen-APPL/FEM library 7
4. Functions of ppOpen-APPL/FEM 7
  4.1. Data structure 7
  4.2. Domain decomposition 8
  4.3. Parallel communication library 8
  4.4. Linear equation solver 8
  4.5. Element library 8
  4.6. Input/Output file 9
5. How to use ppOpen-APPL/FEM APIs to write FEM application code 9
  5.1. Use module 9
  5.2. Initialize 10
  5.3. Reading a mesh file, construction of model data 10
  5.4. Definition of a material constant 10
  5.5. Memory allocation 10
  5.6. The loop-processing portion for accessing element data 10
  5.7. Shape function processing 11
  5.8. Construction of the element stiffness matrix 11
  5.9. Add the element stiffness matrix into the whole matrix 11
  5.10. Set of boundary conditions 11
  5.11. Linear solver 12
  5.12. Output the calculated results 12
  5.13. Finalize 12
6. Mesh file format 13

6.1. Definition of file format 13
6.1.1. Nodes 13
6.1.2. Elements 13
6.1.3. Section 14
6.1.4. Material 14
6.1.5. Node group 14
6.1.6. Emd 15
6.2. Element entity numbering 16
6.2.1. Tetrahedral linear element (341) entity numbering 16
6.2.2. Tetrahedral quadratic element (342) entity numbering 16
6.2.3. Prism linear element (351) entity numbering 17
6.2.4. Prism quadratic element (352) entity numbering 17
6.2.5. Hexahedral linear element (361) entity numbering 18
6.2.6. Hexahedral quadratic element (362) entity numbering 18
6.2.7. Triangular linear element (731) entity numbering 19
6.2.8. Quadrilateral linear shell element (741) entity numbering 19

1. Introduction

This software is a middleware suite that allows a Finite Element Method (FEM) analysis code developer to devote himself to development of the application software by offering a set of functions commonly used in FEM. Installation of this software will create a library where the functions of this software are encapsulated. A sample FEM application program is also created. By linking this library to an FEM program that a user creates, the functions of this software suite can be utilized.

2. How to install the application

2.1. Software required for installation

This software requires the following developer tools and libraries to be available for use in the build procedure.

- A Fortran compiler
- A C compiler
- An MPI library
- The METIS5 library

Here, we have used the following compiler and libraries:
Intel Compiler 12.0.3
OpenMPI 1.4.1
METIS 5.0.2

Fujitsu C/C++ Compiler Driver Version 1.2.1
Fujitsu Fortran Driver Version 1.2.1
FUJITSU MPI Library 1.2.1
METIS 5.1.0

2.2. Unzipping the archive file

The entire set of programs, including the sample program, is in one file in the form of a tar + gzip file.
Unzip this file using the tar-command. The unpacked file group consists of five directories, named `app_flow`, `app_heat`, `app_struct`, `doc`, `etc` and `ppohFEM`. The `app_flow`, `app_heat` and `app_struct` directories are sample application programs which use ppohFEM library. Each directory contains application source codes of flow

dynamics, heat transfer analysis and structural analysis respectively. The `doc` directory contains user guide and reference manual. The `etc` directory contains example `Makefile.in` files, which describe architecture depend settings. The `ppohFEM` directory contains all sources for ppohFEM library.

The `ppohFEM` directory contains following directories, named `bin`, `include`, `lib`, `src` and `tool`. A mesh partitioning tool will be created in the `bin` directory, headers and library will be created in the `include` and `lib` directory respectively, if installation is performed. The `src` directory contains source code of ppohFEM library. The `tool` directory contains source code of mesh partitioner.

2.3. Preparation for compilation.

Examine the compiler and MPI environment of the computer, and then edit the "Makefile.in" file in the installation directory.

Set the MPI-C compiler name in a variable called `CC`.

The linker options of C compiler can be set in a variable called `LDFLAGS`.

The optimize options of C compiler can be set in a variable called `OPTFLAGS`.

The other options of C compiler can be set in a variable called `CFLAGS`.

Set the MPI-Fortran compiler name in a variable called `F90`.

The linker options of Fortran compiler can be set in a variable called `F90LDFLAGS`.

The optimize options of Fortran compiler can be set in a variable called `F90OPTFLAGS`.

The other options of Fortran compiler can be set in a variable called `F90FLAGS`.

Set the full path of the METIS5 library in a variable called `METISDIR`.

2.4. Build the library, sample program and utilities.

When you run

```
make
```

in the installation directory, the compilation is performed in each directory. The library, `ppohFEM/lib/libppohFEM.a` and the mesh partitioner, `ppohFEM/bin/ppohFEM_part`, will be created.

The example of FEM applications are created in `app_struct` and `app_heat` directories.

To clean up a directory, run

```
make clean
```

2.5. Running the sample program and utilities.

Example of FEM application programs such as structural analysis program, heat transfer analysis program and fluid dynamics program are created as `app_struct/bin/app_struct` , `app_heat/bin/app_heat` and `app_flow/bin/app_flow` respectively. Source codes of these programs are stored in `src` directories in `app_struct`, `app_heat` and `app_flow` directories.

Calculation-examples (input and output files) are stored in `app_struct/test01`, `app_struct/test02`, `app_heat/test` and `app_flow/test` directories. Set up the MPI program-execution environment and run the program `../bin/app_struct` and `../bin/app_heat`. The input files are prepared for 4 MPI processes calculation. The calculation results obtained by linux are stored in the `result` directory. Results are stored in *.inp files with Micro AVS format.

The mesh file partitioner is compiled as `ppohFEM/bin/ppohFEM_part` Executing this serial program gives a partitioned mesh file. Number of partition is given in the variable `DOMAIN` in a `hecmw_part_ctrl.dat` file.

3. System overview

3.1. Overview

This system is a library used to develop a set of FEM-analysis code. The ppOpen-APPL/FEM library provides common functions of importance when developing and coding a large scale FEM-simulation. An analysis-code developer can thus concentrate on the development of FEM application software.

3.2. Structure of the ppOpen-APPL/FEM library

This library includes the following functions:

1. The ppOpen-APPL/FEM library
   Management of the stiffness matrix and vector
   An element library
   Management of boundary conditions
   Parallel linear solvers
   File input/output
2. A utility software application for parallel mesh domain decomposition
3. An example program for linear elastic structural analysis and heat transfer analysis

4. Functions of ppOpen-APPL/FEM

4.1. Data structure

ppOpen-APPL/FEM handles three dimension FEM mesh consist of node and element.

Mesh structure is read from distributed mesh file for each MPI process and user can access to the node and element via APIs of ppOpen-APPL/FEM.
The sparse matrix A, x and b vectors in the linear algebraic equation *Ax=b* is managed by the ppOpen-APPL/FEM library. In FEM algorithms, in order to determine A for a region, the element stiffness matrix must be added into the whole stiffness matrix. This function is provided by the ppOpen-APPL/FEM library.

4.2. Domain decomposition

ppOpen-APPL/FEM provides a tool for domain decomposition for parallel computing. Domain decomposition is the process where a mesh is divided according to the number of parallel processes, offering a mesh file for each process. A domain decomposition program, named `ppohFEM_part`, reads a mesh file in a ppOpen-APPL/FEM form, and outputs the divided mesh file.

4.3. Parallel communication library

The ppOpen-APPL/FEM library uses MPI as its parallel communication library. All MPI functions are wrapped by ppOpen-APPL/FEM functions. The communication pattern between MPI processes are managed by mesh partitioner and user need not to take care about source and destination of MPI communication.

4.4. Linear equation solver

This library provides parallel linear equation solvers. In advance of analysis, domain decomposition of the mesh data of a whole domain is carried out, and the mesh data for every partial region is created. In each processor, creation of a stiffness matrix is done independently. By performing communication between domains, using MPI in a linear algebra solver, the compatibility of the whole domain is ensured and parallel computing is made possible.
The following linear solvers are provided.
Preconditioned iterative method: CG, GMRES, BiCGSTAB, GPBiCG
Preconditioner: Jacobi, SSOR

4.5. Element library

ppOpen-APPL/FEM includes 10 types of shape functions, including Hexa (linear, quadratic), Tetra (linear, quadratic), Prism (linear, quadratic), Pyramid (linear, quadratic), and Shell (linear, quadratic).

4.6. Input/Output file

The name of input/output file-names for ppOpen-APPL/FEM are specified via master control file named `hecmw_ctrl.dat`.

One must prepare distributed mesh files for ppOpen-APPL/FEM by partitioning from an entire mesh file. The name of an entire mesh file "`entmesh.msh`" is given in `hecmw_ctrl.dat` as

```
!MESH, NAME=part_in, TYPE=HECMW-ENTIRE
entmesh.msh
```

The file format of entire mesh file is shown on chapter 6.

To partition the entire mesh file to the distributed mesh files, one must set the number of partition and name of distributed mesh file.

The basename of distributed mesh file is given in `hecmw_ctrl.dat` as

```
!MESH, NAME=part_out, TYPE=HECMW-DIST
distmesh
```

The number of process is attached as extension to the basename.

The number of partition is given in `hecmw_part_ctrl.dat` with `DOMAIN` keyword as

```
!PARTITION,TYPE=NODE-BASED,METHOD=PMETIS,DOMAIN=4,UCD=part.inp
```

The kind of output file is defined by application. ppOpen-APPL/FEM provides APIs to write output files in MicroAVS UCD format.

5. How to use ppOpen-APPL/FEM APIs to write FEM application code

A sample program is given in the `app_struct` and `app_heat` directories. These example shows how to write a program for three-dimensional static linear elasticity-analysis and three-dimensional heat transfer-analysis using the ppOpen-APPL/FEM APIs. The API call portions are shown in the source code. The code sample of elasticity-analysis is shown here.

5.1. Use module

Use module "m_ppohFEM" in order to use the ppOpen-APPL/FEM library.

```
use ppohFEM
```

APIs of ppOpen-APPL/FEM library have "`ppohFEM_`" prefix. These APIs can be used by this module.

5.2. Initialize

Initialize ppOpen-APPL/FEM. MPI communication environment is also initialized.

```
call ppohFEM_init
```

5.3. Reading a mesh file, construction of model data

Read a mesh file with the `ppohFEM_get_mesh` subroutine. `idx_mesh` is index of mesh data stored in ppOpen-APPL/FEM. One can access to the mesh by this index.

```
call ppohFEM_get_mesh(idx_mesh)
```

After this, perform the analysis control-file load operation. Since this process is a task that each application developer mounts uniquely, regardless of the API, an explanation is omitted.

5.4. Definition of a material constant

The example program reads the material constant from mesh structure stored in the middleware. It also read material constants from a control file. The constitutive law matrix D is created using this material constant.

5.5. Memory allocation

Allocate memory to ensure space for the linear equations.

Example program generate one equation with three degrees of freedom.

`ppohFEM_mat_con` generate stiffness matrix non-zero matrix element pattern from mesh connectivity. `ppohFEM_mat_clear` zero-clear each matrix element.

```
call ppohFEM_mat_con(idx_mesh, idx_mat)
call ppohFEM_mat_clear(idx_mat)
```

Next, one need to calculate the value of each matrix element and create stiffness matrix.

5.6. The loop-processing portion for accessing element data

To access each element, following loop structure is used. In ppOpen-APPL/FEM, local index of element is sorted for the element type. For the first loop, `itype` specify the index of element type used in the mesh. `iS` and `iE` gives local element index for each `itype` element. `iS` gives element type defined by ppOpen-APPL/FEM. For the second loop, `icel` gives local element index of current element. One can do some process for this element to make element stiffness matrix.

```
do itype=1, ppohFEM_get_n_elem_type(idx_mesh)
```

```
        iS=ppohFEM_get_elem_type_index(idx_mesh, itype-1)+1
        iE=ppohFEM_get_elem_type_index(idx_mesh, itype)
        ic_type=ppohFEM_get_elem_type_item(idx_mesh, itype)
        do icel=iS, iE
! process for each element
        end do
end do
```

5.7. Shape function processing

This part specifies the shape function corresponding to the type of shape elements that are described in the mesh file. One gets the element type with the `ppohFEM_get_elem_type_item` function. One then specifies the shape function according to the element type. The shape function specified here is acquired from the element library which is a part of the ppOpen-APPL/FEM library. For example, `ShapeFunc_hex8n` is the shape function of a hexahedral quadratic element, which has 8 nodes and 9 integration points.

5.8. Construction of the element stiffness matrix

For each integration point of the shape function, the gradient in the spatial coordinates of each node of the element is obtained from the API. An element strain matrix B is obtained by means of this value. The element stiffness matrix is generated from the weight of the shape function and the Jacobian determinant at each integration point.

```
call getQuadPoint(etype, LX, naturalCoord(:))
call getGlobalDeriv(itype, nn, nauralcoord, elem, det, gderiv)
```

5.9. Add the element stiffness matrix into the whole matrix

An element stiffness matrix is added into the whole matrix of the selected linear equation.

```
call ppohFEM_mat_ass_elem(idx_mat, nn, nodLOCAL, stiffness)
```

5.10. Set of boundary conditions

ppOpen-APPL/FEM handles node group defined in mesh file. Boundary conditions are given to each node in the node group. Neumann boundary condition can be applied by set the right hand side value of equation by `ppohFEM_set_stiffMAT_B`. Dirichlet boundary condition can be applied by `ppohFEM_mat_ass_bc`.

Following statement set value as `val` to `idof`'th degree of freedom on `inode`'th node of `imat`'th matrix.

```
call ppohFEM_set_stiffMAT_B(imat, inode, idof, val)
```

Following statement set Dirichlet boundary condition as `val` to `idof`'th degree of freedom on `inode`'th node of `imat`'th matrix.

```
call ppohFEM_mat_ass_bc(imat, inode, idof, val)
```

5.11. Linear solver

Execute the linear solver after identifier of the linear equation *Ax=b*. Solver parameters are set in matrix. `ppohFEM_solve_33` read these parameters and do iterative solver.

Following statemene set number of maximum iteration as `niter` in `imat`'th matrix.

```
ppohFEM_solver_set_iter(imat, niter)
```

Following statemene set method of iterative solver as `nmethod` in `imat`'th matrix, such as 1:CG, 2:BiCBSTAB, 3:GMRES, 4:GPBiCG.

```
ppohFEM_solver_set_method(imat, nmethod)
```

Following statemene set method of preconditioner as `nprecond` in `imat`'th matrix, such as 2:SSOR, 3:Jacobi.

```
ppohFEM_solver_set_precond(imat, nprecond)
```

Set these parameters and execute the linear solver by following statement gives solution vector stored in `imat`'th matrix.

```
call ppohFEM_solver_set_precond(imesh, imat)
```

5.12. Output the calculated results

Calculated results can be output in the MicroAVS UCD format. An output file is created for each mesh part. Set output data by `ppohFEM_visualize_set_node_item_val` or `ppohFEM_visualize_set_elem_item_val` and call `ppohFEM_visualize()`

5.13. Finalize

Finalize ppOpen-APPL/FEM :

```
call ppohFEM_finalize
```

6. Mesh file format

6.1. Definition of file format

The entire mesh file takes a Header, Parameter and Data structure. The Header line is start with "!" and has name of the header and Parameters. Parameter has name and value. The value of the parameter is given by "=". The structure of Data line is different for each header and described at each section. A delimiter is comma. White spaces are ignored. When the line is begin with "#", the line is comment.

6.1.1. Nodes

Describe the location of each node.

Header name: NODE

Parameter: nothing

Data: `#NODE_ID, #Xcoord, #Ycoord, #Zcoord`

`#NODE_ID`: Global node ID. It must be unique for whole mesh.

`#Xcoord,#Ycoord, #Zcoord`: x, y, z coordination of `#NODE_ID`. Unit is free.

6.1.2. Elements

Describe the information of the elements.

Header name: ELEMENT

Parameter: TYPE

TYPE define the element type. Following type can be used.

341: Tetrahedral element (Linear)

342: Tetrahedral element (Quadratic)

351: Triangular prism element (Linear)

352: Triangular prism element (Quadratic)

361: Hexahedral element (Linear)

362: Hexahedral element (Quadratic)

731: Triangular shell element (Linear)

741: Quadrilateral shell element (Linear)

Data: `#ELEM_ID, #node1, #node2, #node3...`

`#ELEM_ID`: Global Element ID. It must be unique for whole mesh.

`#node1, #node2, #node3...` Connectivity of nodes in element.

6.1.3. Section

Describe the information of section of elements. Elements in the same section have same physical property.

Header name: SECTION
Parameter: TYPE, EGRP, MATERIAL
TYPE define the type of elements in this section. Use SOLID for tetrahedral, prism, hexahedral elements. Use SHELL for shell elements.
EGRP define the element group name.
MATERIAL define the material group name defined by user in MATERIAL Header used in this section.
Data: `#Thickness, #Integpoints`
In the case of TYPE=SOLID, Data line can be omitted.
In the case of TYPE=SHELL, `#Thickness` define the shell cross section thickness. `#Integpoints` define the integral point in shell cross sectional direction.

6.1.4. Material

Definition of material physical properties.

Header name: MATERIAL
Parameter: NAME, ITEM
NAME define the material name. This name is used in SECTION header.
ITEM define the number of items in the Data lines.
Data:
(1st Line) !ITEM=1, SUBITEM=`#k`
(2nd Line) `#VAL1, #VAL2, #VAL3 .. #VALk`
(3rd Line) !ITEM=2, SUBITEM=`#j`
(4th Line) `#VAL1, #VAL2, #VAL3 .. #VALj`
Repeat these lines ITEM time.

6.1.5. Node group

Definition of node group.

Header name: NGROUP

Parameter: NGRP
NGRP define the node group name.
Data: `#node`
`#node` global node ID belonging to the node group.
Repeat above line until number of nodes in the node group.

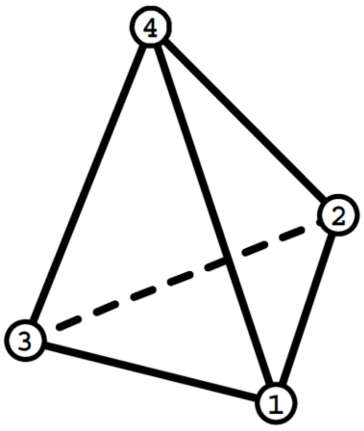
6.1.6. Emd

End of mesh data. When this header is displayed, the reading of the mesh data is completed.

Header name: END
Parameter: N/A
Data: N/A

6.2. Element entity numbering

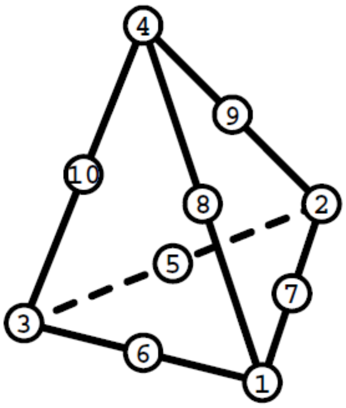
6.2.1. Tetrahedral linear element (341) entity numbering



Surface ID

ID	connectivity
1	2 - 3 - 4
2	1 - 4 - 3
3	1 - 2 - 4
4	1 - 3 - 2

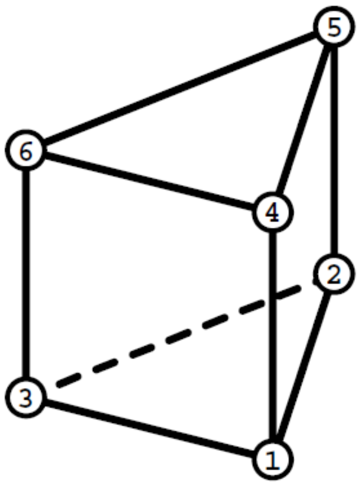
6.2.2. Tetrahedral quadratic element (342) entity numbering



Surface ID

ID	connectivity
1	2 - (5) - 3 - (10) - 4 - (9)
2	1 - (8) - 4 - (10) - 3 - (6)
3	1 - (7) - 2 - (9) - 4 - (8)
4	1 - (6) - 3 - (5) - 2 - (7)

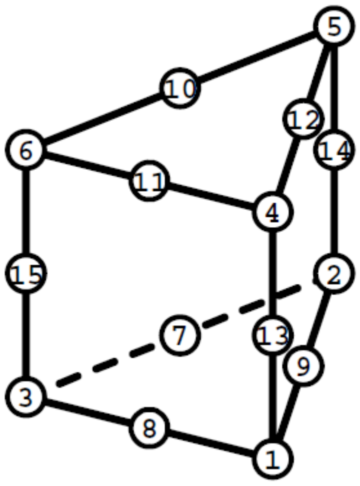
6.2.3. Prism linear element (351) entity numbering



Surface ID

ID	connectivity
1	2 - 3 - 6 - 5
2	3 - 1 - 4 - 6
3	1 - 2 - 5 - 4
4	3 - 2 - 1
5	4 - 5 - 6

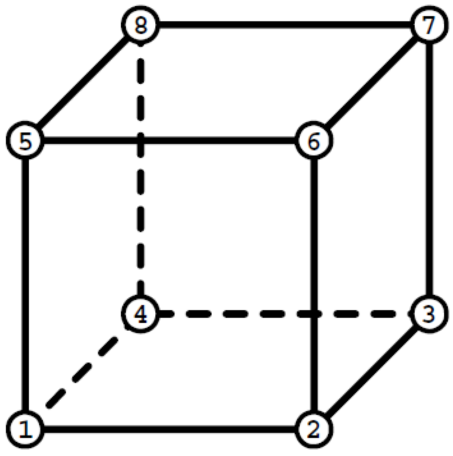
6.2.4. Prism quadratic element (352) entity numbering



Surface ID

ID	connectivity
1	2 - (7) - 3 - (15) - 6 - (10) - 5 - (14)
2	3 - (8) - 1 - (13) - 4 - (11) - 6 - (15)
3	1 - (9) - 2 - (14) - 5 - (12) - 4 - (13)
4	3 - (7) - 2 - (9) - 1 - (8)
5	4 - (12) - 5 - (10) - 6 - (11)

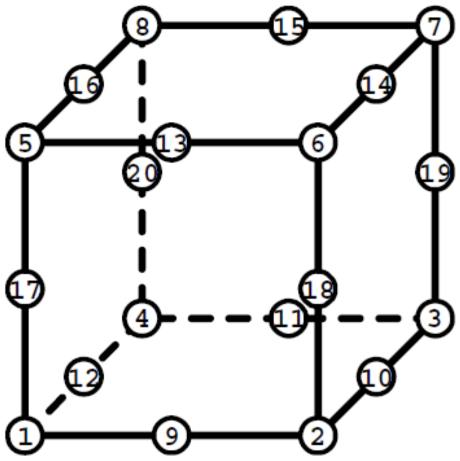
6.2.5. Hexahedral linear element (361) entity numbering



Surface ID

ID	connectivity
1	4 - 1 - 5 - 8
2	2 - 3 - 7 - 6
3	1 - 2 - 6 - 5
4	3 - 4 - 8 - 7
5	4 - 3 - 2 - 1
6	5 - 6 - 7 - 8

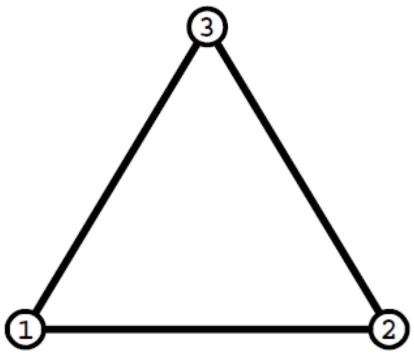
6.2.6. Hexahedral quadratic element (362) entity numbering



Surface ID

ID	connectivity
1	4 - (12) - 1 - (17) - 5 - (16) - 8 - (20)
2	2 - (10) - 3 - (19) - 7 - (14) - 6 - (18)
3	1 - (9) - 2 - (18) - 6 - (13) - 5 - (17)
4	3 - (11) - 4 - (20) - 8 - (15) - 7 - (19)
5	4 - (11) - 3 - (10) - 2 - (9) - 1 - (12)
6	5 - (13) - 6 - (14) - 7 - (15) - 8 - (16)

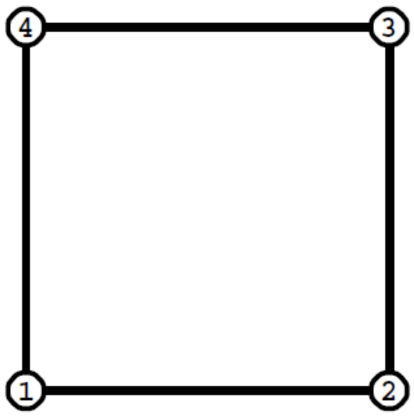
6.2.7. Triangular linear element (731) entity numbering



Surface ID

ID	connectivity
1	1 - 2 - 3 [表]
2	3 - 2 - 1 [裏]

6.2.8. Quadrilateral linear shell element (741) entity numbering



Surface ID

ID	connectivity
1	1 - 2 - 3 - 4 [表]
2	4 - 3 - 2 - 1 [裏]