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

Reference manual

License

This software is one of the results of the JST CREST ``ppOpen-HPC: Open Source Infrastructure for Development and Execution of Large-Scale Scientific Applications on Post-Peta-Scale Supercomputers with Automatic Tuning (AT)" project.

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 API reference

All APIs have prefix 'ppohFEM_'. Subroutine and Function are available.
`use ppohFEM` is required to use these APIs.

1.1 Initialize, Finalize

```
subroutine ppohFEM_init()
```

Initializes the ppOpen-APPL/FEM environment.

```
subroutine ppohFEM_finalize()
```

Finalizes the ppOpen-APPL/FEM environment.

1.2 Read mesh file

```
subroutine ppohFEM_get_mesh(idx_mesh)
```

Reads distributed mesh files as `idx_mesh`'th mesh.

1.3 Initialize stiffness matrix

```
subroutine ppohFEM_mat_con(idx_mesh, idx_mat)
```

Initialize `idx_mat`'th stiffness matrix according to `idx_mesh`'th mesh connectivity.

```
subroutine ppohFEM_mat_clear(idx_mat)
```

Clear all value of matrix element and right hand side vector of `idx_mat`'th stiffness matrix.

```
subroutine ppohFEM_mat_clear_b(idx_mat)
```

Clear all value of right hand side vector of `idx_mat`'th stiffness matrix.

```
subroutine ppohFEM_mat_copy(idx_mesh, idx_mat_src, idx_mat_dest)
```

Copy `idx_mat_src`'th stiffness matrix to `idx_mat_dest`.

1.4 Manage linear equations

```
integer function ppohFEM_get_ndof(idx_mat)
```

Get number of degree of freedom of `idx_mat`'th stiffness matrix.

`integer function ppohFEM_set_hecMAT_ndof(idx_mat, arg_ndof)`
Set number of degree of freedom of `idx_mat`'th stiffness matrix as `arg_ndof`.

`subroutine ppohFEM_set_hecMAT_matrix_element_value(idx_mat, i_row, j_col_node, k_row_dof, l_col_dof, val)`
Set matrix element value of idx_mat'th stiffness matrix. Location in matrix is given by row and column of node id and degree of freedom index.

`subroutine ppohFEM_mat_ass_elem(idx_mat, nn, nodLOCAL, stiffness)`
Add element stiffness matrix to the `idx_mat` 'th global stiffness matrix.

`real*8 function ppohFEM_get_hecMAT_matrix_element_value(idx_mat, inode, jnode, idof, jdof)`
Get the value of stiffness matrix element at `inode`'th row, `idof`'th degree of freedom, `jnode`'th column, `jdof`'th degree of freedom.

1.5 Mesh data access

`integer function ppohFEM_get_n_node(idx_mesh)`
Gets the number of local node (include external node) in `idx_mesh`'th mesh.

`integer function ppohFEM_get_nn_internal(idx_mesh)`
Gets the number of internal node in `idx_mesh`'th mesh.

`integer function ppohFEM_get_n_elem_type(idx_mesh)`
Gets the number of element type used in `idx_mesh`'th mesh.

`integer function ppohFEM_get_n_global_node(idx_mesh)`
Gets the number of global node in `idx_mesh`'th mesh.

`integer function ppohFEM_get_elem_type_index(idx_mesh, itype)`
Gets the array index of the first element of `itype`'th element type in `idx_mesh`'th mesh.

`integer function ppohFEM_get_elem_type_item(idx_mesh, itype)`

Gets the element type of the itype'th element type in idx_mesh'th mesh.

integer function ppohFEM_get_elem_type_item(idx_mesh, itype)
Gets the element type of the itype'th element type in idx_mesh'th mesh.

integer function ppohFEM_get_node_item_of_element(idx_mesh, icel, j)
Gets the node ID of the j'th node of icel'th element type in idx_mesh'th mesh.

subroutine ppohFEM_get_node_coord(idx_mesh, idx_node, XYZ)
Gets the node coordination of the j'th node of idx_node'th node in idx_mesh'th mesh.

integer function ppohFEM_get_section_ID(idx_mesh, icel)
Gets the sectionID belonging of the icel'th element in idx_mesh'th mesh.

integer function ppohFEM_get_num_node_group(idx_mesh)
Gets the number of node group in idx_mesh'th mesh.

integer function ppohFEM_node_grp_name_to_ID(idx_mesh, grp_id_name)
Gets the node group ID which has the name grp_id_name in idx_mesh'th mesh.

integer function ppohFEM_get_num_nodes_in_node_group(idx_mesh, j)
Gets the number of nodes belonging to the j'th node group.

integer function ppohFEM_get_num_nodes_in_node_group(idx_mesh, j)
Gets the number of nodes belonging to the j'th node group.

integer function ppohFEM_get_num_elements_in_element_group(idx_mesh, j)
Gets the number of elements belonging to the j'th element group.

integer function ppohFEM_get_element_item_in_element_group(idx_mesh, ig, ik)
Gets the element ID of ik'th element belonging to the jg'th element group in idx_mesh'th mesh.

1.6 Calculate local stiffness matrix

`integer function ppohFEM_get_max_node_of_elem_type (ictype)`
Gets the number of node of the `ictype` element type.

`integer function ppohFEM_get_num_quadpoints_of_element(ictype)`
Gets the number of quadpoints of the `ictype` element type.

`subroutine ppohFEM_get_quadpoints_of_element(ictype, iqpoint, naturalCoord(:))`
Return the coordinate of `iqpoint`'th quadpoint on `naturalCoord(3)` in real*8 value.

`subroutine ppohFEM_get_global_deriv(ictype, nn, localCoord(:), elecoord(:,:), det, gderiv(:,:))`
Return the shape derivative as `gderiv` and determinant as `det` in global coordinate system. `ictype` is element type. `nn` is number of elemental nodes. `real*8 localCoord(3)` is current position with natural coordination. `real*8 elecoord(3,nn)` is nodal coordination of element.

`subroutine ppohFEM_get_shape_func(ictype, localCoord(:), spfunc(:))`
Return the shape function in `spfunc(20)`. `ictype` is element type. `real*8 localCoord(3)` is current position with natural coordination.

`real*8 function ppohFEM_get_weight(ictype, np)`
Return the weight value in `np`'th gauss point of `ictype` element type.

1.7 Boundary condition

`subroutine ppohFEM_set_stiffMAT_B(idx_mat, inode, jdof, val)`
Set the Neumann boundary condition. Set the value of right hand side vector as `val` at `inode`'th node's `jdof`'th degree of freedom in `idx_mat`'th matrix.

`subroutine ppohFEM_mat_ass_bc(idx_mat, inode, jdof, RHS)`
Set the Dirichlet boundary condition as `val` at `inode`'th node's `jdof`'th degree of freedom in `idx_mat`'th matrix.

`real*8 function ppohFEM_get_rhs_norm(idx_mat)`
Get the 2-norm of right hand side vector in `idx_mat`'th matrix.

1.8 Solver

`subroutine ppohFEM_solver_set_iter(idx_mat, niter)`
Set the maximum iteration count of iterative solver as `niter` in `idx_mat`'th matrix.

`subroutine ppohFEM_solver_set_method(idx_mat, nmethod)`
Set the solver as `nmethod` in `idx_mat`'th matrix, such as 1:CG, 2:BiCBSTAB, 3:GMRES, 4:GPBiCG.

`subroutine ppohFEM_solver_set_precond(idx_mat, nprecond)`
Set the preconditioner as `nprecond` in `idx_mat`'th matrix, such as such as 2:SSOR, 3:Jacobi.

`subroutine ppohFEM_solve_11(idx_mesh, idx_mat)`
Executes the linear solver for the matrix with degree of freedom is 1.

`subroutine ppohFEM_solve_22(idx_mesh, idx_mat)`
Executes the linear solver for the matrix with degree of freedom is 2.

`subroutine ppohFEM_solve_33(idx_mesh, idx_mat)`
Executes the linear solver for the matrix with degree of freedom is 3.

`subroutine ppohFEM_solve_66(idx_mesh, idx_mat)`
Executes the linear solver for the matrix with degree of freedom is 6.

1.9 Visualize

`subroutine ppohFEM_visualize_init()`
Initialize visualizer.

`subroutine ppohFEM_visualize_result_data_init(idx_mesh, idx_result_data, node_item_ndof, elem_item_ndof, node_item_label, elem_item_label)`

Initialize result data structure.

Argument `node_item_ndof` contains number of items on a node and number of degree of freedom for each item.

example of `node_item_ndof`

XYZ coordinate

```
allocate node_item_ndof(1)
node_item_ndof(1)=3 !XYZ
```

XYZ coordinate, pressure

```
allocate node_item_ndof(2)
node_item_ndof(1)=3 !XYZ
node_item_ndof(2)=1 !P
```

XYZ, pressure, UVW flow

```
allocate node_item_ndof(3)
node_item_ndof(1)=3 !XYZ
node_item_ndof(2)=1 !P
node_item_ndof(3)=3 !UVW
```

`elem_item_ndof` also have ndof information items on an element. Arguments for node or element are optional. Keyword argument can be used.

ex1: items on node only

```
call  ppohFEM_visualize_result_data_init(idx_mesh,  idx_result_data,
node_i     tem_ndof=ndof, node_item_label=label)
```

ex2: items on element only

```
call  ppohFEM_visualize_result_data_init(idx_mesh,  idx_result_data,
elem_i     tem_ndof=ndof, elem_item_label=label)
```

ex3: items both on node and element

```
call  ppohFEM_visualize_result_data_init(idx_mesh,  idx_result_data,
node_i     tem_ndof, node_item_label, elem_item_ndof, elem_item_label)
```

`subroutine ppohFEM_visualize_set_node_item_val(idx_result_data, inode, jcomp, kdof, val)`
Set result value as `val` at `idx_result_data`'th result data, `inode`'th node, `jcomp`'th component, `kdof`'th degree of freedom.

`subroutine ppohFEM_visualize_set_elem_item_val(idx_result_data, ielem, jcomp, kdof, val)`
Set result value as `val` at `idx_result_data`'th result data, `ielem`'th element, `jcomp`'th component, `kdof`'th degree of freedom.

`subroutine ppohFEM_visualize(idx_mesh, idx_result_data, step, max_step, interval)`
Output visualize result file. `result_data` is `hecmwST_result_data` type structure. Currently AVS UCD format is supported. Output file basename is given in a line after keyword
`!RESULT, NAME=vis_out, IO=out`
in a file `hecmw_ctrl.dat`. File name of *.inp file is given by argument `step, max_step, interval`. Output file format is given in a keyword !output_type in a file `hecmw_vis.ini`. This file must be placed in same directory with `hecmw_ctrl.dat`.

`subroutine ppohFEM_visualize_finalize()`
Finalize visualizer.