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

ppOpen-MATH/VIS

ver. 0.2.0

User's guide

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

Changes in release 0.2.0

Functionality added or changed:

- ✓ Number of vertices in UCD file is much fewer (less then 20%) thanthat of ver.0.1.0.

Table of Contents

1. Introduction

ppOpen-MATH/VIS is a parallel visualization library. ppOpen-MATH/VIS utilizes information on background voxel meshes with adaptive mesh refinement (AMR), and creates a single file with a reduced size from large-scale distributed files. Users can browse the results on a PC using commercial or open-source visualization tools, such as AVS or ParaView. ppOpen-MATH/VIS generates files in UCD-format (Unstructured Cell Data), which can be read by these tools.

ppOpen-MATH/VIS covers both structured and unstructured meshes from both the Flat-MPI and OpenMP/MPI Hybrid parallel programming models. ppOpen-MATH/VIS ver.0.2.0 just includes capability for structured finite-difference-type meshes with the Flat-MPI parallel programming model (ppOpen-MATH/VIS-FDM3D).

ppOpen-MATH/VIS is written in Fortran 90 and C, and can be called from programs written in both Fortran and C.

2. Installation and Quick Start

2.1 ppohVIS _0.2.0.tar

The “ppohVIS_0.2.0.tar” archive includes the following:

- Source code files of “ppOpen-MATH/VIS ver.0.2.0”
- Source code files of “fdm_test;”; fdm_test is a 3D solver for transient diffusion equations with explicit time-marching and calls ppOpen-MATH/VIS-FDM3D
- Sample Makefiles
- Sample data files for “fdm_test (1 variable version, 3 variable version)”

2.2 Structure of Directories

The “ppohVIS_0.2.0.tar” archive includes the following directories. `$(CUR)` denotes the directory where the “ppohVIS_0.2.0.tar” archive is unpacked.

Name of Directory	Contents
`$(CUR)/src`	source code files of ppohVIS_FDM3D_0.1.0.tar
`$(CUR)/examples/FDM/src`	source code files of “fdm_test” (1 variable version)
`$(CUR)/examples/FDM/run`	sample data sets of “fdm_test”
`$(CUR)/examples/FDM/run/ppohVIS`	directory for storing output files
`$(CUR)/examples/FDM/run/results`	directory for storing sample output files
`$(CUR)/examples/FDM3/src`	source code files of “fdm_test” (3 variable version)
`$(CUR)/examples/FDM3/run`	sample data sets of “fdm_test”
`$(CUR)/examples/FDM3/run/ppohVIS`	directory for storing output files
`$(CUR)/examples/FDM3/run/results`	directory for storing sample output files
`$(CUR)/include`	directory that holds the created module files
`$(CUR)/lib`	directory that holds the created libraries
`$(CUR)/bin`	directory that holds the created exec. files
`$(CUR)/doc`	Documents
`$(CUR)/etc`	examples of ‘`Makefile.in`’

2.3 Quick Start

(1) Preparation

- C and Fortran 90 compilers (Operations have been verified by Intel, PGI, Fujitsu compilers)
- MPI library

(2) Modify 'Makefile.in'

Samples of 'Makefile.in' are found in `$(CUR)/etc`.

Examples of 'Makefile.in'	Compiler, Parallel Programming Models
`$(CUR)/etc/Makefile.in.fx10`	Flat MPI for Fujitsu FX10
`$(CUR)/etc/Makefile.in.general`	PGI & Intel Compiler

Options in 'Makefile.in'	Descriptions
$(CC)	C compiler with MPI
$(COPTFLAGS)	Optimization options for the C compiler
$(CXX)	C++ compiler with MPI
$(CXXOPTFLAGS)	Optimization options for the C++ compiler
$(FC)	F77 compiler with MPI
$(FCOPTFLAGS)	Optimization options for the F77 compiler
$(F90)	F90 compiler with MPI
$(F90OPTFLAGS)	Optimization options for the F90 compiler
$(PREFIX)/include	directory that holds the installed module files
$(PREFIX)/lib	directory that holds the installed libraries
$(PREFIX)/bin	directory that holds the installed exec. files

***** NOTICE ***: $(PREFIX) directory must be specified as ABSOLUTE/FULL path.**

(3) Compile/install ppOpen-MATH/VIS-FDM3D

Operations	Created files (libraries, module files, exec. files)
$> cd $(CUR)/** **$> make clean	
$> make	`$(CUR)/lib/libppohvisfdm3d.a (for C)` `$(CUR)/lib/libfppohvisfdm3d.a (for Fortran)`

$> make install	`$(PREFIX)/lib/libppohvisfdm3d.a (for C)`
	`$(PREFIX)/lib/libfppohvisfdm3d.a`
	`(for Fortran)`

(4) Compile/install “fdm_test”

(one variable version)

Operations	Created files (libraries, module files, exec. files)
$> cd $(CUR)/	
$> make fdm1	$(CUR)/examples/FDM/run/fdm_test

(three variable version)

Operations	Created files (libraries, module files, exec. files)
$> cd $(CUR)/	
$> make fdm3	$(CUR)/examples/FDM3/run/fdm_test

***** NOTICE ***: Process (4) must be done after (3).**

(5) Running the code

(one variable version)

$> cd $(CUR)/examples/FDM/run

$> mpirun -np 8 fdm_test (or corresponding operations)

Output files are created in $(CUR)/examples/FDM/run/ppohVIS

(three variable version)

$> cd $(CUR)/examples/FDM3/run

$> mpirun -np 8 fdm_test (or corresponding operations)

Output files are created in $(CUR)/examples/FDM3/run/ppohVIS

(6) Clean/Uninstall

$> cd $(CUR)/

$> make clean Clean files

$> make uninstall Delete all installed files and directories

$> make fdm1_clean Clean file

$> make fdm3_clean Clean file

3. How to use ppOpen-MATH/VIS-FVM3D

3.1 Header File

This library's module file (Fortran) or header file (C) must be included if ppOpen-MATH/VIS/FVM3D is used.

FORTRAN

```
use ppohVIS_FDM3D_Util
```

C

```
#include "ppohVIS_FDM3D_Util.h"
```

4. API Reference

4.1 Parameters

FORTRAN

```
!... Integer
  integer, parameter :: IKind = 4
!... Double precision
  integer, parameter :: RKind = 8
!... Length of character variables for file name
  integer, parameter :: PPOHVIS_FDM3D_FILE_NAME_LEN = 256
!... Length of character variables for labels of data
  integer, parameter :: PPOHVIS_FDM3D_LABEL_LEN     = 256
```

C

```
/* Length of character variables for file name */
#define PPOHVIS_FDM3D_FILE_NAME_LEN 256
/* Length of character variables for labels of data */
#define PPOHVIS_FDM3D_LABEL_LEN 256
```

4.2 Derived Types

(1) Control Data

FORTRAN

```
  type, public :: ppohVIS_FDM3D_stRefineControl
    real   (kind=RKind) :: AvailableMemory  !... available memory size (not used)
    integer(kind=IKind) :: MaxRefineLevel   !... max refinement level
    integer(kind=IKind) :: MaxVoxelCount    !... max voxel number
  end type ppohVIS_FDM3D_stRefineControl

  type, public :: ppohVIS_FDM3D_stControl
    !... information for control on AMR
    type(ppohVIS_FDM3D_stRefineControl) :: Refine
  end type ppohVIS_FDM3D_stControl
```

C

```
struct ppohVIS_FDM3D_stRefineControl {
        double AvailableMemory;  /* available memory size (not used) */
        int MaxRefineLevel;      /* max refinement level             */
        int MaxVoxelCount;       /* max voxel number                 */
};

struct ppohVIS_FDM3D_stControl {
        /* information for control on AMR */
        struct ppohVIS_FDM3D_stRefineControl Refine;
};
```

(2) FDM Mesh Data

FORTRAN

```
  type, public :: ppohVIS_FDM3D_stStrGrid
    integer(kind=IKind) :: NumX     !... number of internal meshes in X-dir.(local)
    integer(kind=IKind) :: NumY     !... number of internal meshes in Y-dir.(local)
    integer(kind=IKind) :: NumZ     !... number of internal meshes in Z-dir.(local)
    real(kind=RKind)    :: DeltaX   !... mesh size in X-dir.
    real(kind=RKind)    :: DeltaY   !... mesh size in Y-dir.
    real(kind=RKind)    :: DeltaZ   !... mesh size in Z-dir.
    real(kind=RKind)    :: OriginX  !... location of origin for local mesh (X)
    real(kind=RKind)    :: OriginY  !... location of origin for local mesh (Y)
    real(kind=RKind)    :: OriginZ  !... location of origin for local mesh (Z)
  end type ppohVIS_FDM3D_stStrGrid
```

C

```
struct ppohVIS_FDM3D_stStrGrid {
        int NumX;        /* number of internal meshes in X-dir.(local) */
        int NumY;        /* number of internal meshes in Y-dir.(local) */
        int NumZ;        /* number of internal meshes in Z-dir.(local) */
        double DeltaX;   /* mesh size in X-dir. */
        double DeltaY;   /* mesh size in Y-dir. */
        double DeltaZ;   /* mesh size in Z-dir. */
        double OriginX;  /* location of origin for local mesh (X) */
        double OriginY;  /* location of origin for local mesh (Y) */
        double OriginZ;  /* location of origin for local mesh (Z) */
};
```

(3) Results

If multiple variables are defined at each mesh, please specify “number of DOF (components)”. Components are stored in “Value” array so that inner-loops correspond to loops of DOF.

FORTRAN

```
!... EntityType : node data
  integer(kind=IKind), parameter, public :: ppohVIS_FDM3D_ResultNode = 0
!... EntityType : element data
  integer(kind=IKind), parameter, public :: ppohVIS_FDM3D_ResultElement = 1

  type, public :: ppohVIS_FDM3D_stResult
    !... number of entity's (node or element)
    integer(kind=IKind)                          :: ItemCount
    !... EntityType
    !...   ppohVIS_FDM3D_ResultNode or ppohVIS_FDM3D_ResultElement
    integer(kind=IKind)                          :: EntityType
    !... DOF of data
    integer(kind=IKind)                          :: FreedomCount
    !... labels of data
    character(len=PPOHVIS_FDM3D_LABEL_LEN)       :: Label

    !... data to be visualized is stored in 1D array as follows
    !...   (Value(FreedomCount*(J-1)+I), J=1,ItemCount), I=1,FreedomCount)
    real(kind=RKind), dimension(:), allocatable :: Value
  end type ppohVIS_FDM3D_stResult

  type, public :: ppohVIS_FDM3D_stResultCollection
    !... number of data sets
    integer(kind=IKind)                                    :: ListCount
    !... list of data sets
    type(ppohVIS_FDM3D_stResult), dimension(:), allocatable :: Results
  end type ppohVIS_FDM3D_stResultCollection
```

C

```
enum ppohVIS_FDM3D_eResultEntityType {
        ppohVIS_FDM3D_ResultNode,      /* EntityType : node data */
        ppohVIS_FDM3D_ResultElement,   /* EntityType : element data */
};

struct ppohVIS_FDM3D_stResult {
        /* number of entity' s (node or element) */
        int ItemCount;
        /* EntityType                                                    */
        /*   ppohVIS_FDM3D_ResultNode or ppohVIS_FDM3D_ResultElement     */
        enum ppohVIS_FDM3D_eResultEntityType EntityType;
        /* DOF of data */
        int FreedomCount;
        /* labels of data */
        char Label[PPOHVIS_FDM3D_LABEL_LEN];

        /* data to be visualized is stored in 1D array as follows        */
        /*   for(j=0; j<ItemCount; j++) {                                */
        /*    for(i=0; i<FreecomCount; i++) {                            */
        /*     Value[FreedomCount*j+i] = A; }; };                        */
        double *Value;
};

struct ppohVIS_FDM3D_stResultCollection {
        /* number of data sets */
        int ListCount;
        /* list of data sets */
        struct ppohVIS_FDM3D_stResult **Results;
};
```

4.3 Subroutines and Functions

ppohVIS_FDM3D_Init

This function initializes the library. This function MUST be called before calling any other functions of this library. An active MPI communicator must be provided as a parameter. The current version only accepts "MPI_COMM_WORLD".

FORTRAN

```
subroutine PPOHVIS_FDM3D_INIT(comm, iErr)
  integer(kind=IKind), intent(in)    :: comm
  integer(kind=IKind), intent(inout) :: iErr
```

C

```
extern int
ppohVIS_FDM3D_Init(
        ppohVIS_FDM3D_Comm comm);
```

Parameters

comm	MPI communicator for visualization. The current version only accepts "MPI_COMM_WORLD".
iErr	(Only for Fortran) Error code, =0: successful termination, other than 0: failed

ppohVIS_FDM3D_GetControl

This function reads information for control of this library. The format of the control file is explained in Chapter 5. Corresponding information can be created without calling this function.

FORTRAN

```
subroutine PPOHVIS_FDM3D_GETCONTROL(cFileName, pControl, iErr)
  character(len=PPOHVIS_FDM3D_FILE_NAME_LEN), intent(in)    :: cFileName
  type(ppohVIS_FDM3D_stControl),              intent(out)   :: pControl
  integer(kind=IKind),                        intent(inout) :: iErr
```

C

```
extern struct ppohVIS_FDM3D_stControl *
ppohVIS_FDM3D_GetControl(char *cFileName);
```

Parameters

cFileName	Name of control file
pControl	(Only for Fortran) Information for control obtained from control file. Returned value in C. If “NULL” is returned, the function aborts.
iErr	(Only for Fortran) Error code, =0: successful termination, other than 0: failed

ppohVIS_FDM3D_SetStrGrid

This function registers mesh data in the library, and creates local mesh IDs. Local numbering of result data (ppohVIS_FDM3D_stResult%Value) must be identical with that of this information. The method used for local mesh numbering is as follows:

```
nNode = 0;
for(iZ=0; iZ<pGrid->NumZ+1; iZ++) {
  for(iY=0; iY<pGrid->NumY+1; iY++) {
    for(iX=0; iX<pGrid->NumX+1; iX++) {
      pNode->ID[nNode]         = nNode+1;                                  //Node ID
      pNode->Coords[3*nNode  ] = pGrid->OX + pGrid->DX * (double)(iX); //X-coord.
      pNode->Coords[3*nNode+1] = pGrid->OY + pGrid->DY * (double)(iY); //Y-coord.
      pNode->Coords[3*nNode+2] = pGrid->OZ + pGrid->DZ * (double)(iZ); //Z-coord.
      nNode++;
};};};
```

```
nElem = 0;
for(iZ=0; iZ<pGrid->NumZ; iZ++) {
  for(iY=0; iY<pGrid->NumY; iY++) {
    for(iX=0; iX<pGrid->NumX; iX++) {
      pElem->ID[nElem] = nElem+1;      //Mesh ID
      iBase = (pGrid->NumZ+1)*(pGrid->NumY+1)*iZ+(pGrid->NumY+1)*iY+iX+1;
      pElem->Node[nElem*8  ] = iBase;  //Connectivity
      pElem->Node[nElem*8+1] = iBase                                      +1;
      pElem->Node[nElem*8+2] = iBase                         +pGrid->NumX+1;
      pElem->Node[nElem*8+3] = iBase                         +pGrid->NumX;
      pElem->Node[nElem*8+4] = iBase+pGrid->NumX*pGrid->NumY;
      pElem->Node[nElem*8+5] = iBase+pGrid->NumX*pGrid->NumY              +1;
      pElem->Node[nElem*8+6] = iBase+pGrid->NumX*pGrid->NumY+pGrid->NumX+1;
      pElem->Node[nElem*8+7] = iBase+pGrid->NumX*pGrid->NumY+pGrid->NumX;
      nElem++;
};};};
```

FORTRAN

```
subroutine ppohVIS_FDM3D_SETSTRGRID(pGrid, iErr)
  type(ppohVIS_FDM3D_stStrGrid), intent(in)    :: pGrid
  integer(kind=IKind),           intent(inout) :: iErr
```

C

```
extern int
ppohVIS_FDM3D_SetStrGrid(
        struct ppohVIS_FDM3D_stStrGrid *pGrid);
```

Parameters

pGrid	FDM mesh data. Location of the origin and mesh number and size in each direction are stored in the derived type ppohVIS_FDM3D_stStrGrid
iErr	(Only for Fortran) Error code, =0: successful termination, other than 0: failed

ppohVIS_FDM3D_Visualize

This function creates UCD files. Variables on nodes (pResultNode) and those on elements (pResultElem) are defined as different arrays. If ListCount is set to 0, the corresponding data is not visualized. Multiple UCD files can be created simultaneously. The name of a file are determined in the following manner:

cFileHeader.[label of result data].iStep.inp

FORTRAN

```
subroutine PPOHVIS_FDM3D_VISUALIZE(pResultNode, pResultElem, pControl,        &
                                   cFileHeader, iStep, iErr)
  type(ppohVIS_FDM3D_stResultCollection), intent(in)    :: pResultNode
  type(ppohVIS_FDM3D_stResultCollection), intent(in)    :: pResultElem
  type(ppohVIS_FDM3D_stControl),          intent(in)    :: pControl
  character(len=ppohVIS_FDM3D_LABEL_LEN), intent(in)    :: cFileHeader
  integer(kind=IKind),                    intent(in)    :: iStep
  integer(kind=IKind),                    intent(inout) :: iErr
```

C

```
ppohVIS_FDM3D_Visualize(
        struct ppohVIS_FDM3D_stResultCollection *pResultNode,
        struct ppohVIS_FDM3D_stResultCollection *pResultElement,
        struct ppohVIS_FDM3D_stControl *pControl,
        char *cFileHeader,
        int iStep);
```

Parameters

pResultNode	Results on nodes. Set ListCount=0, if results on nodes are not included.
pResultElem	Results on elements. Set ListCount=0, if results on nodes are not included.
pControl	Control data for this library
cFileHeader	Header of UCD files
iStep	Step ID
iErr	(Only for Fortran) Error code, =0: successful termination, other than 0: failed

ppohVIS_FDM3D_Finalize

This function terminates operations by the library. This function must be called after all the needed functions of the library have been used.

FORTRAN

```
subroutine PPOHVIS_FDM3D_FINALIZE(iErr)
  !... Error Code
  integer(kind=IKind), intent(inout) :: iErr
```

C

```
extern int
ppohVIS_FDM3D_Finalize(void);
```

5. Format of Control File

An example of a control file is as follows:

```
[Refine]
AvailableMemory = 2.0     Available memory size (GB), not available in this version.
MaxVoxelCount   = 500     Maximum number of voxels
MaxRefineLevel  = 20      Maximum number of refinement levels
```

6. Example-I (one variable version)

Figure 1 shows the procedures for the installation of example code using ppOpen-MATH/VIS-FDM3D. The source code can be found in `$(CUR)/examples/FDM/src/test.f`, which is a very simple 3D FDM code sample for transient diffusion equations with explicit time-marching. A directory for storing UCD files (`./ppohVIS`) must be prepared in the directory where the code is running. In this case, the UCD files are stored in `$(CUR)/examples/FDM/run/ppohVIS`.

Control files (`input.dat` for simulations by `fdm_test`, and `control.dat` for visualization by ppOpen-MATH/VIS-FDM3D) must be in the directory where the code is running. The names of these control files are fixed.

```
$> cd ppohVIS_0.2.0

$> make
$> make install

$> ls examples/FDM/src/test.f
   test.f

$> make fdm1

$> cd examples/FDM/run

$> ls input.dat
   input.dat

$> ls control.dat
   control.dat

$> mpirun -np 8 fdm_test (or corresponding operation)

$> ls ppohVIS/

ppohVIS_FDM3D_PHI.10000.inp   ppohVIS_FDM3D_PHI.40000.inp  ppohVIS_FDM3D_PHI.80000.inp
ppohVIS_FDM3D_PHI.100000.inp  ppohVIS_FDM3D_PHI.50000.inp  ppohVIS_FDM3D_PHI.90000.inp
ppohVIS_FDM3D_PHI.20000.inp   ppohVIS_FDM3D_PHI.60000.inp
ppohVIS_FDM3D_PHI.30000.inp   ppohVIS_FDM3D_PHI.70000.inp
```

Fig.1 Procedures for installation of example code using ppOpen-MATH/VIS-FDM3D

Figure 2 shows geometry and boundary conditions of the problem solved in `fdm_test`. Each finite-difference mesh is a geometric cube the edge length of which is equal to 1.0. The number of meshes in each direction is NX, NY and NZ. Uniform volume flux generation is applied in the cell at the origin (the red circle in Fig.2). Figure 3 shows a control file for `fdm_test` (input.dat). The entire model has `NX*NY*NZ` meshes, and divides into `(IP,JP,KP)` parts in each direction, therefore the

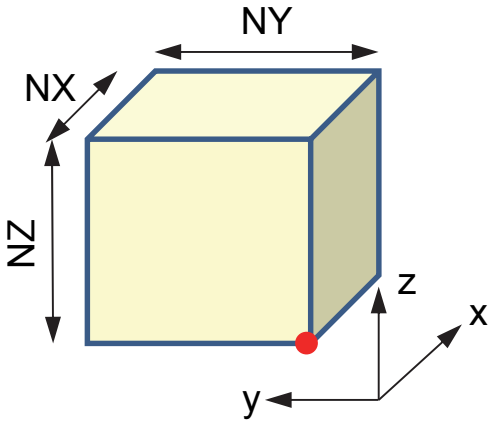

Fig.2 Geometry and Boundary Conditions, Uniform Volume Flux Generation at the Origin (Red Circle)

total number of MPI processes is equal to `IP*JP*KP` (=8 in this case). Figure 4 shows a control file for ppOpen-MATH/VIS-FDM3D (control.dat). Figure 5 shows the result after 100,000 steps.

```
64 64 64              NX, NY, NZ              Number of meshes in each direction
 2  2  2              IP, JP, KP              Number of processors in each direction
 1.0                  QO                      Volume flux at the origin
 1.0   10000.0        DT, TIMEmax             Fixed time step, Total time
100000                ITERmax                 Total number of time steps
 10000 1000.0         ITERfreq, TIMEfreq      Frequency of writing UCD files (time steps, time)
 0.10d-0              OMEGA                   Coef. for time-step, actual time step= OMEGA*DT
```

Fig.3 Control file (input.dat) for `fdm_test`

```
[Refine]
AvailableMemory = 2.0    Available memory size (GB), not available in this version.
MaxVoxelCount   = 500    Maximum number of voxels
MaxRefineLevel  = 20     Maximum number of refinement levels
```

Fig.4 Control file (control.dat) for `ppOpen-MATH/VIS-FDM3D`

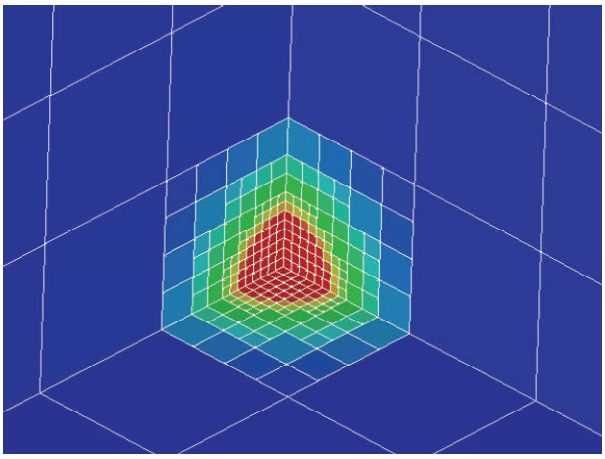

Fig.5 Results after 100,000 steps

7. Example-II (three variable version)

Figure 6 shows the procedures for the installation of example code using ppOpen-MATH/VIS-FDM3D. The source code can be found in `$(CUR)/examples/FDM3/src/test.f`, which is a very simple 3D FDM code sample for transient diffusion equations with explicit time-marching. A directory for storing UCD files (`./ppohVIS`) must be prepared in the directory where the code is running. In this case, the UCD files are stored in `$(CUR)/examples/FDM3/run/ppohVIS`.

Control files (`input.dat` for simulations by `fdm_test`, and `control.dat` for visualization by ppOpen-MATH/VIS-FDM3D) must be in the directory where the code is running. The names of these control files are fixed.

```
$> cd ppohVIS_0.2.0

$> make
$> make install

$> ls examples/FDM3/src/test.f
   test.f

$> make fdm3

$> cd examples/FDM3/run

$> ls input.dat
   input.dat

$> ls control.dat
   control.dat

$> mpirun -np 8 fdm_test (or corresponding operation)

$> ls ppohVIS/

ppohVIS_FDM3D_PHI.10000.inp   ppohVIS_FDM3D_PHI.40000.inp  ppohVIS_FDM3D_PHI.80000.inp
ppohVIS_FDM3D_PHI.100000.inp  ppohVIS_FDM3D_PHI.50000.inp  ppohVIS_FDM3D_PHI.90000.inp
ppohVIS_FDM3D_PHI.20000.inp   ppohVIS_FDM3D_PHI.60000.inp
ppohVIS_FDM3D_PHI.30000.inp   ppohVIS_FDM3D_PHI.70000.inp
```

Fig.6 Procedures for installation of example code using ppOpen-MATH/VIS-FDM3D

Figure 2 shows geometry and boundary conditions of the problem solved in `fdm_test`. Each finite-difference mesh is a geometric cube the edge length of which is equal to 1.0. The number of meshes in each direction is NX, NY and NZ. Uniform volume flux generation is applied in the cell at the origin (the red circle in Fig.3). Figure 3 shows a control file for `fdm_test` (input.dat). The entire model has `NX*NY*NZ` meshes, and divides into `(IP,JP,KP)` parts in each direction, therefore the total number of MPI processes is equal to `IP*JP*KP` (=8 in this case). Figure 4 shows a control file for ppOpen-MATH/VIS-FDM3D (control.dat).