Fluid-structure interaction simulation software

by

Dr. Chennakesava Kadapa
(c.kadapa@swansea.ac.uk)

Swansea University, Swansea, UK.

Version : 0.1

Date : August 28, 2020

Contents

Chapter 1

Execution of the program

Once the source is code compiled and built, an executable called `mpap` is generated.

Serial version:

To run the program in the serial mode, use the following syntax.

```
./mpap  <path-to-directory-of-input-file>   <name-of-the-input-file>
```

Parallel version:

To run the program in the parallel mode using N processors, use the following syntax.

```
mpirun -n N  ./mpap  <path-to-directory-of-input-file>   <name-of-the-input-file>
```

Chapter 2

Generic input file

The input file always starts with a key word MPAP2 and consists of three main blocks:

1.) Domain type: All the data related to grid, fluid properties and immersed objected is specified in this block. For CutFEM approach, this block starts with "BEGIN HBSPLINECUTFEM 1" and ends with "END HBSPLINECUTFEM 1".

2.) Time functions: This block contains the time functions for time-dependent input velocity/tractions when specifying boundary conditions or to prescribe the motion of immersed solids. This block starts with "BEGIN TIME_FUNCTIONS" and ends with "END TIME_FUNCTIONS".

3.) Run control: This block specifies what type of solver to choose, the time step, max iterations for Newton-Raphson scheme, to print output and write output files etc.. This block starts with "BEGIN RUN_CONTROL" and ends with "END RUN_CONTROL".

A typical input file looks like:

```
MPAP2

BEGIN HBSPLINECUTFEM 1
<<<<data>>>>
END HBSPLINECUTFEM 1


BEGIN TIME_FUNCTIONS
<<<<data>>>>
END TIME_FUNCTIONS


BEGIN RUN_CONTROL
<<<<data>>>>
END RUN_CONTROL
```

Chapter 3

Cartesian fluid grid

In the present formulation fluid is modelled on a cartesian grid which is discretised with hierarchical B-Splines. A cartesian grid is a rectangular domain in two-dimensions and cuboid in three-dimensions. The complete description of a fluid grid requires the following details.

- Dimension of the domain
- Origin of the domain in each coordinate direction
- Length of the domain in each coordinate direction
- Degree of B-Splines in each coordinate direction
- Number of elements in each coordinate direction
- Local refinement details
- Properties of the fluid
- Dirichlet boundary conditions
- Neumann boundary conditions
- Control parameters

In the input file any line that starts with a ”!” is not effective. It helps the user to write useful comments for better understanding of the input file.

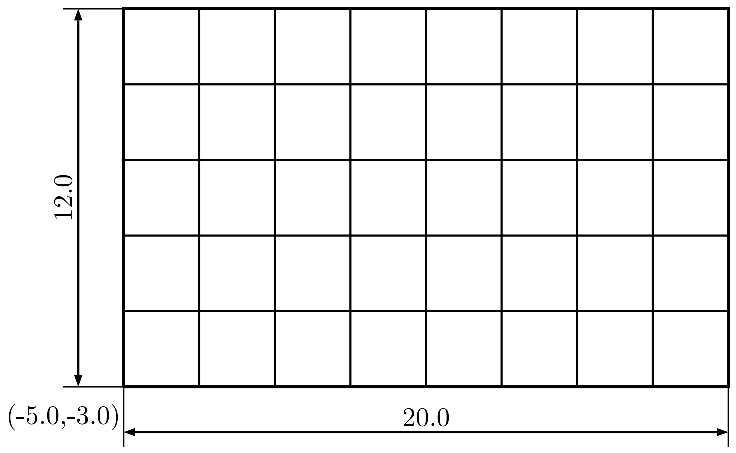

Figure 3.1: A typical cartesian grid.

3.1 Dimension of the domain

To specify the dimension of the fluid domain and number of degree of freedom (ndof) at each control point. For incompressible Navier-Stokes in 2D, ndof=3 and for a 3D problem ndof=4.

For a two-dimensional problem:

```
! ndof   ndim
   3      2
```

For a three-dimensional problem:

```
dimensions
! ndof   ndim
   4      3
```

3.2 Origin of the domain

Origin of the grid is the point where the grid starts. The coordinates are the minimum values in each coordinate direction.

For a two-dimensional problem:

```
origin
! X0   Y0
-5.0  -3.0
```

For a three-dimensional problem:

```
origin
! X0   Y0   Z0
-1.0  -1.0  -1.0
```

3.3 Grid dimensions

Specify the length of the cartesian grid in each coordinate direction.

For a two-dimensional problem:

```
grid dimensions
! lx   ly
20.0  12.0
```

For a three-dimensional problem:

```
grid dimensions
```

```
! lx     ly     lz
20.0    12.0    1.0
```

3.4 Polynomial degrees

Specify the degree of the B-Splines to be used in each dimension. In this version the degree of B-Splines must be the same in each coordinate direction. However, the user is expected to enter the value in each coordinate direction.

For a two-dimensional problem:

```
polynomial degrees
!   p    q
    2    2
```

For a three-dimensional problem:

```
polynomial degrees
!   p    q    r
    3    3    3
```

3.5 Number of elements

Specify the number of elements to be used in each dimension.

For a two-dimensional problem:

```
number of elements
!   nx    ny
    8     5
```

For a three-dimensional problem:

```
number of elements
!   nx    ny    nz
    20    10    10
```

3.6 Fluid properties

To define the material properties and other input data for the fluid domain. The input requires at least 11 values.

1.) Number of Gauss points in each coordinate direction. Always choose $a + 1$ where a is the order of B-Splines.

2.) This value is unused at the moment. The user must enter some value.

3.) This is a flag to define whether the two-dimensional problem is plane flow (value 0) or axisymmetric flow (value 1). This value is ineffective for 3D problems.

4.) Fluid density.

5.) Fluid viscosity.

6.) Body force in x-direction.

7.) Body force in y-direction.

8.) Body force in z-direction.

9.) SUPG stabilisation. This is for development purpose only. This parameter is calculated internally. Always choose 1.0.

10.) PSPG stabilisation. This is for development purpose only. This parameter is calculated internally. Always choose 1.0.

11.) LSIC stabilisation. This is for development purpose only. This parameter is calculated internally. Always choose 1.0.

A typical example:

```
fluid properties
 ! nGP  flag  axsy  rho   mu    fx   fy   fz   SUPG  PSPG  LSIC
    3     0     1   10.0  0.02  0.0  0.0  0.0   1.0   1.0   1.0
```

3.7 Refinement type

To specify the type and related parameters for performing the local refinement. The input requires three values.

1.) Refinement algorithm type. In the current version, always choose 5.

2.) Number of refinement levels to be included.

3.) This value is ineffective in the current version. But always choose some value.

The following input performs level 3 refinement:

```
refinement type
! type   nref   depth
    5      3      2
```

The refinement level (nref) should always be less than or equal to the number of maximum refinement level for which limits are refined.

3.8 Mesh refinement limits

To specify the lower and upper bounds of the locations for local refinement. The input requires 5 values for each entry for 2D problem and 7 values for each entry for 3D problem. The first entry specifies the refinement level to which the limits are applicable. Second and third entries are the lower and upper bounds in X-direction; 4th and 5th entries are lower and upper bounds in Y-direction; and 6th and 7th values are corresponding values in Z-direction.

There can be any number of entries for the refinement on the same hierarchy level. This facilitates performing local refinement in several isolated locations on the same hierarchy level. For 2D problem values for Z-direction are ignored.

```
mesh refinement limits
 !level   xl    xu    yl    yu   zl   zu
    1     3.0   6.0   2.7   9.0
    1     9.4  12.7   3.0   8.8
    2     3.4   6.7   3.0   8.8
    3     4.1   4.5   1.5   9.6
```

The refinement process at level $k+1$ is performed only in the zone of intersection of limits of level k and level $k+1$. For example, with the above input, refinement at level 2 is performed in the area [3.4,6.0]x[2.7,8.8].

3.9 Dirichlet boundary conditions

The boundaries of the background grid are numbered as shown in Fig. 3.2. For the purpose of clarity the numbering scheme is explained below.

X: 1 for the boundary with outward normal in negative X-direction and 2 for the boundary with outward normal in positive X-direction.

Y: 3 for the boundary with outward normal in negative Y-direction and 4 for the boundary with outward normal in positive Y-direction.

Z: 5 for the boundary with outward normal in negative Z-direction and 6 for the boundary with outward normal in positive Z-direction.

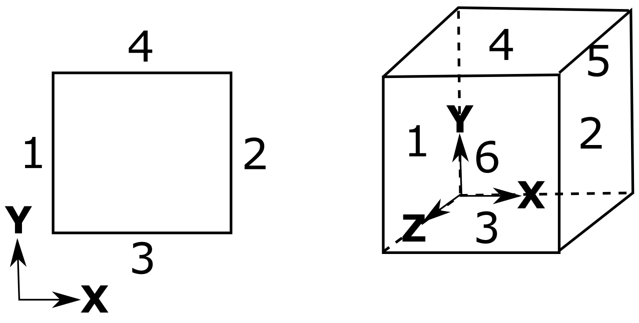

Figure 3.2: Numbering of boundaries of the grid.

The data for specifying Dirichlet boundary conditions (BC) on the boundaries of the cartesian grid requires 6 values for each entry.

1.) Side of the grid on which Dirichlet BC is applied.

2.) Degree of freedom number to which BC is applied.

3.) Value of the Dirichlet BC.

4.) Penalty parameter.

5.) A flag to specify whether to use Nitsche method (value=1) or not (value=0). When this flag is set to 0 then the method becomes standard penalty method of applying BCs.

6.) To choose between symmetric (value=1.0) and unsymmetric (value=-1.0) variants of Nitsche's method.

Though penalty-free unsymmetric Nitsche method works fine for steady problems, it is not recommended for unsteady flows. So, it is always recommended to use some positive value for penalty parameter. However, it should not be too large, as such large values result in spurious oscillations. Always choose in the range of 10-1000.

The following input specifies a Dirichlet BC of value 0.25 on Y-velocity for side 3 of the grid and uses symmetric Nitsche method with a penalty parameter of 100.

```
dirichlet boundary conditions
 ! side   dof   value   penalty   isNistche   NitscheFact
   3       2    0.25    100.0        1           1.0
```

The above input specifies a Dirichlet boundary condition value of 0.25 on the 2nd degree of freedom on the bottom side with a penalty parameter of 1000.0.

3.10 Neumann boundary conditions

To specify the Neumann boundary conditions on the boundaries of the Cartesian grid. The input format is almost the same as that of Dirichlet boundary conditions without the parameter related to Nitsche method.

```
neumann boundary conditions
! side   dof   value
    1     1    0.39
    6     3    2.0
```

The above input specifies a Neumann boundary condition value of 0.39 in X-direction on the left side (side=1) and a value of 2.0 in the Z-direction on the front face (side=6) of the three-dimensional grid.

3.11 Neumann boundary conditions

To specify the Derivative boundary conditions on the boundaries of the Cartesian grid. The input format is almost the same as that of Dirichlet and Neumann boundary conditions. This option is useful for imposing zero-gradient boundary condition on the outflow for high Reynolds number flows. The condition is imposed using the penalty method.

```
derivative boundary conditions
! side   dof   value   PENALTY
    2     1    0.0     10.0
    2     2    0.0     10.0
```

3.12 Control parameters

To specify the convergence tolerance, type of time integration scheme and associated parameters. This requires three input values.

1.) Convergence tolerance for the Newton-Raphson scheme.

2.) Type of time integration scheme.

   0 - Steady Navier-Stokes.

   2 - Generalised-α scheme.

   3 - Backward-Euler scheme.

3.) Spectral radius value for Generalised-α. This value is effective only when the time integration scheme is chosen to be generalise-α scheme.

```
control parameters
  !tol    tis  rho
 1.0e-6    2   0.2
```

Chapter 4

Fluid rigid-body interaction

For problems involving interaction of fluid and solid, the user must define the boundary of the solid and associated parameters.

4.1 CutFEM parameters

These parameters affect how integration is performed in cut cells; on the boundary edges and specify values for ghost-penalty parameters. The input requires at least 8 values.

1.) This value defines the integration procedure for cut cells. The input should be either 1 or 2. For integration using subtriangulation, choose 1. For integration using adaptive integration, choose 2. For 2D problem it is always recommended to choose subtriangulation.

2.) Number of Gauss points per immersed boundary edge. Should always be $a + 1$ where a is order of B-Splines used for the background fluid grid. Recommended value is 5 for B-Splines of order upto 3.

3.) Number of Gauss points per triangle in the subtriangulation. This value is effective only for gp_type=1 (subtriangulation). Recommended value is 7 for B-Splines of order upto 3.

4.) Adaptive integration level upto which the Gauss points are to be included for integration of cut cells. This value is effective only for gp_type=2 (adaptive integration).

5.) Number of adaptive integration levels below refLev1 that are to be used for merging weights of Gauss points. This value is effective only for gp_type=2 (adaptive integration).

6.) This value is not used at the moment. Always choose some value.

7.) Ghost penalty parameter for velocity. Always choose a small value but never smaller than 10^{-4}.

8.) Ghost penalty parameter for pressure. Always choose a small value but never smaller than 10^{-4}.

A typical input:

```
cutfem parameters
 ! gp_type   nGP_edge   nGP_tria   refLev1   refLev2   tmp2   gammU   gammP
      1          5          7          5         0        1    0.001   0.001
```

4.1.1 Immersed bodies

To define an immersed body (IB) and some data related to them. The input requires at least 5 values.

1.) The first entry specifies whether the IB is rigid (value=0) or flexible (value=1).

2.) For the current version of CutFEM, this value is inactive. Always choose 1.

3.) Penalty parameter for the Nitsche's method.

4.) A flag to specify whether to use Nitsche method (value=1) or not (value=0). When this flag is set to 0 then the method becomes standard penalty method for applying BCs. Always choose Nitsche's method, i.e., value=1.

5.) To choose between symmetric (value=1.0) and unsymmetric (value=-1.0) variants of Nitsche's method. Always choose unsymmetric Nitsche's method, i.e., value=-1.0.

```
immersed body data


 ! R/F   P/L   penalty   isNitsche   NitscheFact
    0      1    1.0e2         1           -1.0
```

The above input defines an immersed rigid body. The Dirichlet boundary conditions on the rigid bodies are imposed using symmetric Nitsche method with a penalty parameter of 100.

4.1.2 Immersed boundary points

For the fluid-rigid-body interaction only the boundary of the immersed solid is required. So, each immersed solid is described in terms of its boundary. In the current version, the boundary of the solid in 2D is input as a set of straight edges and in 3D as a set of linear triangles (3-noded triangle). To specify the points describing the boundary of the immersed boundary.

```
immersed points
!   id     xcoor     ycoor     zcoor
     1     5.1158    6.146
     2     5.2090    2.952
     3     6.1980    5.172
     4     3.1870    3.738
     5     7.0760    4.072
```

The above input specifies five points on the boundary of an immersed body for a 2D problem. There is no need to specify Z-coordinate for the 2D problem.

4.1.3 Immersed integration elements

Once the points are specified, the connectivity information for the edges$->$points and solids$->$ edges has to be provided. The immersed integration elements are the straight edges in 2D (or 3-noded triangles in 3D) on which boundary conditions are applied. Each entry requires 4 values for 2D problems and 5 values for 3D problems.

The normal to the immersed edge/face should always point away from the active fluid domain. In 2D, this means that the boundary of the immersed solid is travelled in clock-wise direction.

1.) The id of the immersed edge/triangle.

2.) Whether this edge/triangle is active (value=1) or inactive (value=0). When value=0 i.e., when the edge/triangle is inactive boundary conditions are not applied on that edge/triangle. This is useful when the boundary of the immersed solid coincides with the boundary of background grid, or to impose non-zero traction boundary conditions. Non-zero traction BCs is yet to be implemented.

3.) The 1st point of the edge/triangle.

4.) The 2nd point of the edge/triangle.

5.) The 3rd point of the triangle. (Only for 3D).

```
immersed integration elements
 ! id   active   n1   n2
   1       1      1    2
   2       1      2    3
   3       1      3    4
   4       1      4    5
   5       1      5    1
```

4.2 Rigid body properties

In order to simulate the rigid fluid-structure interaction one has to specify the properties of the rigid-body, for example, its mass, damping, stiffness and the active/inactive DOFs.

- A rigid-body in 2D has three DOF and hence mass, damping and stiffness matrices are of size 3×3. So, each entry consists 9 values.
- A rigid-body in 3D has six DOF and hence mass, damping and stiffness matrices are of size 6×6. So, each entry consists 36 values.

By default, the immersed rigid solid is assumed to be fixed. Therefore, for the fixed rigid bodies, there is no need to specify any of the properties. However, the user may still specify these values. The properties of an immersed rigid-body are input as follows.

```
rigid body mass
 ! Mxx   Mxy   Mxt   Myx   Myy    Myt   Mtx   Mty   Mtt
   0.0   0.0   0.0   0.0   20.0   0.0   0.0   0.0   0.0

rigid body damping
 ! Cxx   Cxy   Cxt   Cyx      Cyy       Cyt   Ctx   Cty   Ctt
   0.0   0.0   0.0   0.0   0.0581195   0.0   0.0   0.0   0.0

rigid body stiffness
 ! Kxx   Kxy   Kxt   Kyx     Kyy     Kyt   Ktx   Kty   Ktt
   0.0   0.0   0.0   0.0   3.08425   0.0   0.0   0.0   0.0
```

Apart from the values of mass, damping and stiffness, the user also needs to input the data related to motion of the immersed rigid body. A value of 1 or -1 should be specified for each degree of freedom. A value of 1 indicates that the solid is free to move in the corresponding DOF and a value of -1 indicates that the solid is fixed in the corresponding DOF.

The input to specify that a rigid body in 2D is fixed in X- and θ- directions but can move in Y-direction:
rigid body degree of freedom

```
! d1  d2  d3
 -1    1  -1
```

The input to specify that a rigid body in 3D is free to move in X-, Y- and Z- directions but is

constrained with respect to rotations:
rigid body degree of freedom

```
! d1  d2  d3  d4  d5  d6
  1    1   1  -1  -1  -1
```

Similary, the prescribed motion of the rigid body in any DOF can be specified using the following syntax. The following block specifies the prescribed motion on DOF 3 which is rotational DOF is 2D and the translational DOF in Z-direction in 3D.

```
rigid body prescribed motion
! f(t) = p1 + p2*t + p3*sin(p4*t+p5) + p6*cos(p7*t+p8)
! dof  t0    t1     p1   p2   p3            p4          p5   p6   p7   p8
  3    0.0  1000.0  0.0  0.0  2.0943951023  0.31415926  0.0  0.0  0.0  0.0
```

Chapter 5

Time functions

1.) The id of the time function.

2.) t0 is the start time.

3.) t1 is the end time.

4.) p1, p2, p3, ... are the coefficients.

Function with a constant value of one with start 0 and end time 10000.

```
BEGIN TIME_FUNCTIONS
!
! f(t) = p1 + p2*t + p3*sin(p4*t+p5) + p6*cos(p7*t+p8)
!
! id  t0    t1       p1    p2    p3    p4    p5     p6   p7    p8

  1  0.0  10000.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0

END TIME_FUNCTIONS
```

Function that varies linearly from until 1 unit and then remains constant at 1 until 10000 units.

```
BEGIN TIME_FUNCTIONS
!
! f(t) = p1 + p2*t + p3*sin(p4*t+p5) + p6*cos(p7*t+p8)
!
! id  t0    t1       p1    p2    p3    p4    p5     p6   p7    p8

  1  0.0      1.0   0.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0
  1  1.0  10000.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0

END TIME_FUNCTIONS
```

A sinusoidally varying function with an amplitude of 5 and frequency π .

```
BEGIN TIME_FUNCTIONS
!
! f(t) = p1 + p2*t + p3*sin(p4*t+p5) + p6*cos(p7*t+p8)
!
! id  t0    t1      p1    p2    p3    p4    p5     p6   p7    p8

  1  0.0  10000.0   0.0   0.0   1.0   3.14159265358979   0.0   0.0   0.0   0.0

END TIME_FUNCTIONS
```

Chapter 6

Examples

```
MPAP2

%=====================================================================

BEGIN HBSPLINECUTFEM 1

dimensions
! ndf   ndm
3   2

origin
! X0   Y0   Z0
0.0   0.0   0.0

grid dimensions
! lx   ly   lz
20.0   12.0   1.0

polynomial degrees
! p   q   r
2   2   0

number of elements
! nx   ny   nz
52   31   0

fluid properties
! nGP   flag   axsy   rho   mu   fx fy fz   SUPG   PSPG   LSIC   eps   gamm
2   0   1   8.e-4   5.0e-5   0.0 0.0 0.0   1.0   1.0   1.0   0.0   0.0

refinement type
```

```
! type   nref   ind
5   2   1


mesh refinement limits
!level   xl   xu   yl   yu   zl   zu
1   3.0   11.0   2.7   9.0
2   3.4   10.7   3.0   8.8
3   4.15   10.55   3.35   8.55


dirichlet boundary conditions
! side   dof   value   PEN   isNitsche   NitscheFact
1   1   5.0   100   1   1.0
1   2   0.0   100   1   1.0
3   2   0.0   100   1   1.0
4   2   0.0   100   1   1.0


neumann boundary conditions
! side   dof   value
3   1   0.0
3   2   0.0
4   1   0.0
4   2   0.0


immersed body data
! R/F   P/L   PEN   isNitsche   NitscheFact
0   1   100   1   -1.0
```

Chapter 7

Run control

```
BEGIN RUN_CONTROL

! begin batch mode
BATCH
! this sets the solver
anly,solv,,28,1,8,1,1,1,1
! this generates the grid file, without any solution field.
chen,data,,28,2,2,2,1
! time step size
dt,,0.10
! this generates the grid file, with the solution field.
vtk,flow,,28,1,1,1,1,1,1,1,0,0,1
! write nodal data to the output file starting with T
wrnd
! useful to stop the program before running the time loop if we want to
! check if the input domain is correct.
wait,mous
! Time loop; this particular one runs up to a final time of 500 seconds. 499.999999999 to av
!loop,,399.9999999999,2
! Time loop; to run 100 time steps
loop,,100
  ! update time functions
  time
  ! update the solver. Solves the solid problem and updates the fluid mesh.
  updt
  ! this step solves the fluid problem
  anly,tang,,28,1,1
  ! generates output in the VTK format. One for each time step.
  vtk,flow,,28,1,1,1,1,1,1,1,0,0,1
  wrnd
! go to next
next
! end of loop
end


END RUN_CONTROL
```

Additional details

```
! To generate one file for every 50th step use
vtk,flow,,28,1,1,1,50,1,1,1,0,0,1
```