

**HORSES3D**  
A **H**igh-**O**rders (DG) **S**pectral **E**lement **S**olver  
**User Manual**

Numath group

February 1, 2023

# Contents

<b>1</b>	<b>Compiling the code</b>	<b>3</b>
<b>2</b>	<b>Input and Output Files</b>	<b>4</b>
2.1	Input Files . . . . .	4
2.2	Output Files . . . . .	4
<b>3</b>	<b>Running a Simulation</b>	<b>5</b>
3.1	Control File (*.control) - Overview . . . . .	5
3.2	Boundary conditions . . . . .	6
<b>4</b>	<b>Restarting a Case</b>	<b>8</b>
<b>5</b>	<b>Physics related keyword</b>	<b>9</b>
5.1	Compressible flow . . . . .	9
5.1.1	Shock-capturing . . . . .	9
5.1.2	Acoustic . . . . .	12
5.2	Incompressible Navier-Stokes . . . . .	13
5.3	Cahn-Hilliard . . . . .	13
5.4	Complementary Modes . . . . .	13
5.4.1	Wall Function . . . . .	13
5.4.2	Tripping . . . . .	13
<b>6</b>	<b>Implicit Solvers with Newton linearisation</b>	<b>15</b>
6.1	General Keywords . . . . .	15
6.2	Keywords for the BDF Methods . . . . .	15
6.3	Keywords for the Rosenbrock-Type Implicit Runge-Kutta Methods . . . . .	16
6.4	Jacobian Specifications . . . . .	16
<b>7</b>	<b>Explicit Solvers</b>	<b>18</b>
<b>8</b>	<b>Nonlinear <math>p</math>-Multigrid solver (FAS)</b>	<b>19</b>
<b>9</b>	<b>p-Adaptation Methods</b>	<b>21</b>
9.1	Multiple truncation error estimations . . . . .	22
<b>10</b>	<b>Immersed boundary method</b>	<b>23</b>
10.0.1	STL file . . . . .	24
10.0.2	Computing forces . . . . .	24
10.0.3	Moving bodies . . . . .	24
<b>11</b>	<b>Monitors</b>	<b>26</b>
11.1	Residual Monitors . . . . .	26
11.2	Statistics Monitor . . . . .	26
11.3	Probes . . . . .	26
11.4	Surface Monitors . . . . .	27
11.5	Volume Monitors . . . . .	28

<b>12 Advanced User Setup</b>	<b>29</b>
12.1 Routines of the Problem File: <i>ProblemFile.f90</i> . . . . .	29
12.2 Compiling the Problem File . . . . .	29
<b>13 Postprocessing</b>	<b>31</b>
13.1 Visualization with Tecplot Format: <i>horses2plt</i> . . . . .	31
13.1.1 Solution Files (*.hsol) . . . . .	31
13.1.2 Statistics Files (*.stats.hsol) . . . . .	32
13.2 Extract geometry . . . . .	32
13.3 Merge statistics tool . . . . .	32

# Chapter 1

## Compiling the code

- Clone the git repository or copy the source code into a desired folder.
- Go to the folder Solver.
- Run configure script.

```
$ ./configure
```

- Install using the Makefile:

```
$ make all <<Options>>
```

with the desired options (bold are default):

- MODE=DEBUG/**RELEASE**
- COMPILER=ifort/**gfortran**
- COMM=PARALLEL/**SEQUENTIAL**
- PLATFORM=MACOSX/**LINUX**
- ENABLE\_THREADS=NO/**YES**
- WITH\_MKL=y/**n**

For example:

```
$ make all COMPILER=ifort COMM=PARALLEL
```

- The ENABLE\_THREADS=YES flag enables shared memory simulations using OpenMP.
- The COM=PARALLEL flag enables distributed memory simulations using MPI.
- To compile the code linking it with METIS (that is an option for creating the mesh partitions of MPI runs), it is needed that before compilation and running, an environment variable called METIS\_HOME is found. This variable must contain the path to the HDF5 installation folder (it must have been compiled with the same compiler as HORSES3D).
- To compile the code linking it with HDF5 (necessary for reading HOPR meshes), it is needed that before compilation and running, an environment variable called HDF5\_ROOT is found. This variable must contain the path to the METIS installation folder (it must have been compiled with the same compiler as HORSES3D). In addition, the lib folder must be added to the environment variable LD\_LIBRARY\_PATH.
- If you use *environment modules*, it is advised to use the HORSES3D module file:

```
$ export MODULEPATH=$HORSES_DIR/ utils / modulefile :$MODULEPATH
```

where \$HORSES\_DIR is the installation directory.

- It is advised to run the *make clean* or *make allclean* command if some options of the compilation routine needs to be changed and it has been compiled before.

## Chapter 2

# Input and Output Files

DONT USE TABS!

### 2.1 Input Files

- Control file (\*.control)
- Mesh file (\*.mesh / \*.h5 / \*.msh)
- Polynomial order file (\*.omesh)
- Problem File (ProblemFile.f90)

Notes on the GMSH format (\*.msh) and general workflow using GMSH.

- Curved geometry supported up to polynomial order 5.
- Curved geometry should be generated using following options: tools -i options -i mesh -i general -i element order.
- HORSES3D can read mesh format 4.1 and 2.2 (legacy format).
- The solution to most of the problems mesh reading is to load it in GMSH and export to format 2.2 to have a clean ASCII file.

### 2.2 Output Files

- Solution file (\*.hsol)
- Horses mesh file (\*.hmesh)
- Boundary information (\*.bmesh)
- Partition file (\*.pmesh)
- Polynomial order file (\*.omesh)
- Monitor files (\*.volume / \*.surface / \*.residuals)

## Chapter 3

# Running a Simulation

### 3.1 Control File (\*.control) - Overview

The control file is the main file for running a simulation. A list of all the mandatory keywords for running a simulation and some basic optional keywords is presented in Table 3.1. The specific keywords are listed in the other chapters.

Table 3.1: General keywords for running a case.

Keyword	Description	Default value
solution file name	<i>CHARACTER</i> : Path and name of the output file. The name of this file is used for naming other output files.	<b>Mandatory keyword</b>
simulation type	<i>CHARACTER</i> : Specifies if NSLITE3D must perform a 'steady-state' or a 'time-accurate' simulation.	'steady-state'
time integration	<i>CHARACTER</i> : Can be 'implicit', 'explicit', or 'FAS'. The latter uses the Full Algebraic Storage (FAS) multigrid scheme, which can have implicit or explicit smoothers.	'explicit'
polynomial order	<i>INTEGER</i> : Polynomial order to be assigned uniformly to all the elements of the mesh. If the keyword <i>polynomial order file</i> is specified, the value of this keyword is overridden.	—*
polynomial order i polynomial order j polynomial order k	<i>INTEGER</i> : Polynomial order in the i, j, or k component for all the elements in the domain. If used, the three directions must be declared explicitly, unless you are using a polynomial order file. If the keyword <i>polynomial order file</i> is specified, the value of this keyword is overridden.	—*
polynomial order file	<i>CHARACTER</i> : Path to a file containing the polynomial order of each element in the domain.	—*
restart	<i>LOGICAL</i> : If .TRUE., initial conditions of simulation will be read from restart file specified using the keyword <i>restart file name</i> .	<b>Mandatory keyword</b>
cfl	<i>REAL</i> : A constant related with the <b>convective</b> Courant-Friedrichs-Lewy (CFL) condition that the program will use to compute the time step size.	—**
dcfl	<i>REAL</i> : A constant related with the <b>diffusive</b> Courant-Friedrichs-Lewy (DCFL) condition that the program will use to compute the time step size.	—**
dt	<i>REAL</i> : Constant time step size.	—**
final time	<i>REAL</i> : This keyword is mandatory for time-accurate solvers	—
mesh file name	<i>CHARACTER</i> : Name of the mesh file. The currently supported formats are <i>.mesh</i> (SpecMesh file format) and <i>.h5</i> (HOPR hdf5 file format).	<b>Mandatory keyword</b>
mesh inner curves	<i>LOGICAL</i> : Specifies if the mesh reader must suppose that the inner surfaces (faces connecting the elements of the mesh) are curved. This input variable only affects the hdf5 mesh reader.	.TRUE.
number of time steps	<i>INTEGER</i> : <i>Maximum</i> number of time steps that the program will compute.	<b>Mandatory keyword</b>

Table 3.1: General keywords for running a case - continued.

Keyword	Description	Default value
output interval	<i>INTEGER</i> : In steady-state, this keyword indicates the interval of time steps to display the residuals on screen. In time-accurate simulations, this keyword indicates how often a 3D output file must be stored.	<b>Mandatory keyword</b>
convergence tolerance	<i>REAL</i> : Residual convergence tolerance for steady-state cases	<b>Mandatory keyword</b>
partitioning	<i>CHARACTER</i> : Specifies the method for partitioning the mesh in MPI simulations. Options are: 'metis' (the code must have been linked to METIS at compilation time, see Chapter 1), or 'SFC' (to use a space-filling curve method, no special compilation is needed for this option).	'metis'
manufactured solution	<i>CHARACTER</i> : Must have the value '2D' or '3D'. When this keyword is used, the program will add source terms for the conservative variables taken into account an exact analytic solution for each primitive variable $j$ ( $\rho, u, v, w, p$ ) of the form: $j = j_C(1) + j_C(2) \sin(\pi j_C(5)x) + j_C(3) \sin(\pi j_C(6)y) + j_C(4) \sin(\pi j_C(7)z)$ Where $j_C(i)$ are constants defined in the file <i>ManufacturedSolutions.f90</i> . Proper initial and boundary conditions must be imposed (see the test case). The mesh must be a unit cube.	–

\* One of these keywords must be specified

\*\* For Euler simulations, the user must specify either the CFL number or the time-step size. For Navier-Stokes simulations, the user must specify the CFL and DCFL numbers **or** the time-step size.

## 3.2 Boundary conditions

The boundary conditions are specified as blocks in the control file. The block start with a the keywords '#define' and ends with '#end'. Inside the block the options are specified as a pair of keywords and values, just as the normal body of the rest of the file.

Each boundary condition can be individually defined or if multiple boundaries are set with the same definition, it could be done on the same block (with the name separated by a double under score '\_' sign). The name of each boundary must match with the one specified at the mesh file.

The block in general can be seen below. Table 3.2 show the values for the type keyword, and the possible value for the parameters depends on the boundary condition.

```
#define boundary myBoundary1..myBoundary2..myBoundary3
      type          = typeValue
      parameter 1 = value_1
      parameter 2 = value_2
# end
```

Table 3.2: Keywords for Boundary Conditions.

Keyword	Description	Default value
type	<i>CHARACTER</i> : Type of boundary condition to be applied. Options are: Inflow, Outflow, NoSlipWall, FreeSlipWall, Periodic, User-defined.	N/A

For periodic boundary conditions, the second boundary that must be used as a complement must be specified by the keyword 'coupled boundary'. These two boundaries must have the same node position in all directions but one. For mesh files generated by comercial software where this strict rule is not imposed a comparison based on the minimum edge size of the face element can be used by a boolean parameter in the normal body of the control file (*not in the block body*), with the keyword 'periodic relative tolerance'.

*Juan's email (to be translated and adapted to the manual format as a complement):*

Hola Gente,

He tenido que hacer unas modificaciones bastante importantes en las BCs. Era la única parte del código que estaba “a la antigua” y no programada a objetos. Esto hacía que no fueran muy customizables, y por ejemplo las controlábamos con el número ese que siempre vale 0.0 jajaja. Ahora cada condición de contorno tiene los parámetros que necesitas y se pueden customizar. Lo malo es que ningún control file de los que tenéis van a seguir funcionando, pero os escribo los cambios para que sepáis adaptarlos, en cualquier caso, podéis pedirme ayuda y os cuento.

Los cambios del código son:

- Las condiciones de contorno se definen igual que los monitores, con los `#define` en la parte final del control file. Para definir una condición de contorno se hace:

```
#define boundary name
    type = Inflow/Outflow/NoSlipWall/FreeSlipWall/Periodic/User-defined
    parametro1 = #valor
    parametro2 = #valor
#end
```

- Los parámetros1, ... dependen de la condición de contorno que toque. Si no se especifica nada, pues está como estaba antes. Dos cambios importantes:

· He unificado las NoSlipWall (adiabática e isoterma) en una sola. Por defecto es adiabática. · En las periódicas es obligatorio ahora indicar a qué boundary se acopla (lo cual supone poco esfuerzo y reduce el tiempo de búsqueda al código)

```
#define boundary name
    type = Periodic
    coupled boundary = nombredelboundaryalqueseacopla
#end
```

- Se pueden definir más de una condición de contorno del tirón, por ejemplo si boundary1, boundary2 y boundary3 son inflows se puede hacer:

```
#define boundary boundary1__boundary2__boundary3
    type = Inflow
#end
```

es decir, separado con dos guiones bajos.

- Por pantalla, donde aparecía la info de las zonas y tal, también aparece qué BC tiene y cuáles son los parámetros.
- La BC outflowspecifyP la llamo simplemente Outflow. Más que nada por que antes había algunos ficheros de control con la BC Outflow y no existía, pero por defecto se mandaba a Inflow. Para evitar problemas, pues Outflow.
- Los archivos de condición de contorno están en /physics/common en lugar de cada uno su archivo. Esto es por que al final son todas iguales y si se añade una nueva es más facil agregar un nuevo archivo que hacerlo individualmente en cada ecuación.
- Los bcTypeDictionary bcValueDictionary desaparecen. Las BC están en el module physics/common/BoundaryConditions.f90 como variable global, se llama BCs y dentro aloja todas las condiciones de contorno (una por zona, y en el mismo orden de las zonas).

Creo que eso es todo, lamento si os supone mucho cambio en vuestros ficheros de control que estéis corriendo a día de hoy, y si rompo algo que no reflejen los test. Pero estos cambios eran necesarios para darle más versatilidad (por ejemplo en multifase el inflow necesita bastante customización, para definir caudales de cada fase y cosas así). Además, creo que el enfoque OOP va en la dirección del resto del código.



## Chapter 4

# Restarting a Case

Table 4.1: Keywords for restarting a case.

Keyword	Description	Default value
restart	<i>LOGICAL</i> : If <code>.TRUE.</code> , initial conditions of simulation will be read from restart file specified using the keyword <i>restart file name</i> .	<b>Mandatory keyword</b>
restart file name	<i>CHARACTER</i> : Name of the restart file to be written and, if keyword <i>restart</i> = <code>.TRUE.</code> , also name of the restart file to be read for starting the simulation.	<b>Mandatory keyword</b>
restart polorder	<i>INTEGER</i> : Uniform polynomial order of the solution to restart from. This keyword is only needed when the restart solution is of a different order than the current case.	same as case's
restart polorder file	<i>CHARACTER</i> : File containing the polynomial orders of the solution to restart from. This keyword is only needed when the restart solution is of a different order than the current case.	same as case's
get discretization error of	<i>CHARACTER</i> : Path to solution file. This can be used to estimate the discretization error of a solution when restarting from a higher-order solution.	—

## Chapter 5

# Physics related keyword

### 5.1 Compressible flow

Table 5.1: Keywords for compressible flow (Euler / Navier-Stokes).

Keyword	Description	Default value
Mach number	<i>REAL</i> :	<b>Mandatory keyword</b>
Reynolds number	<i>REAL</i> :	<b>Mandatory keyword</b>
Prandtl number	<i>REAL</i> :	0.72
Turbulent Prandtl number	<i>REAL</i> :	Equal to Prandtl
AOA theta	<i>REAL</i> : Angle of attack (degrees), based on the spherical coordinates polar angle ( $\theta$ ) definition	0.0
AOA phi	<i>REAL</i> : Angle of attack (degrees), based on the spherical coordinates azimuthal angle ( $\varphi$ ) definition	0.0
LES model	<i>CHARACTER</i> (*): Options are: <ul style="list-style-type: none"> <li>• Smagorinsky</li> <li>• None</li> </ul>	None
Wall model	<i>CHARACTER</i> :	linear

#### 5.1.1 Shock-capturing

*WARNING: The functionality explained in this section is still experimental and may change in future iterations.*

The shock-capturing module helps stabilize cases with discontinuous solutions, and may also improve the results of under-resolved turbulent cases. It is built on top of a *Sensors* module that detects problematic flow regions, classifying them according to the value of the sensor,  $s$ , mapped into the interval  $a \in [0, 1]$ ,

$$a = \begin{cases} 0, & \text{if } s \leq s_0 - \Delta s/2, \\ \frac{1}{2} \left[ 1 + \sin \left( \frac{s-s_0}{\Delta s} \right) \right], & \text{if } s_0 - \Delta s/2 < s < s_0 + \Delta s/2, \\ 1, & \text{elsewhere.} \end{cases}$$

The values of  $s_0 = (s_1 + s_2)/2$  and  $\Delta s = s_2 - s_1$  depend on the sensor thresholds  $s_1$  and  $s_2$ .

At the moment, flow regions where  $a \leq 0$  are considered smooth and no stabilization algorithm can be imposed there. In the central region of the sensor, with  $0 < a < 1$ , the methods shown in the next table can be used and even scaled with the sensor value, so that their intensity increases in elements with more instabilities. Finally, the higher part of the sensor range can implement a different method from the table; however, the intensity is set to the maximum this time.

All the methods implemented introduce artificial dissipation into the equations, which can be filtered with an SVV kernel to reduce the negative impact on the accuracy of the solution. Its intensity is controlled with the

parameters  $\mu$  (similar to the viscosity of the Navier-Stokes equations) and  $\alpha$  (scaling of the density-regularization term of the Guermond-Popov flux), which can be set as constants or coupled to the value of the sensor or to a Smagorinsky formulation.

Table 5.2: Keywords for shock-capturing algorithms in the Navier-Stokes equations.

Keyword	Description	Default value
Enable shock-capturing	<i>LOGICAL</i> : Switch on/off the shock-capturing stabilization	.FALSE.
Shock sensor	<i>CHARACTER</i> : Type of sensor to be used to detect discontinuous regions Options are: <ul style="list-style-type: none"> <li>• Zeros: always return 0</li> <li>• Ones: always return 1</li> <li>• Grad rho: based on the norm of the density gradient</li> <li>• Modal: based on the relative weight of the higher order modes</li> <li>• Truncation error: estimate the truncation error of the approximation</li> <li>• Aliasing error: estimate the aliasing error of the approximation</li> </ul>	Grad rho
Shock first method	<i>CHARACTER</i> : Method to be used in the middle region of the sensor ( $a \in [0, 1]$ ). Options are: <ul style="list-style-type: none"> <li>• None: Do not apply any smoothing</li> <li>• Non-filtered: Apply the selected viscous flux without SVV filtering</li> <li>• SVV: Apply an entropy-stable, SVV-filtered viscous flux</li> </ul>	None
Shock second method	<i>CHARACTER</i> : Method to be used in the top-most region of the sensor ( $a = 1$ ). Options are: <ul style="list-style-type: none"> <li>• None: Do not apply any smoothing</li> <li>• Non-filtered: Apply the selected viscous flux without SVV filtering</li> <li>• SVV: Apply an entropy-stable, SVV-filtered viscous flux</li> </ul>	None
Shock viscous flux 1	<i>CHARACTER</i> : Viscous flux to be applied in the elements where $a \in [0, 1]$ . Options are: <ul style="list-style-type: none"> <li>• Physical</li> <li>• Guermond-Popov (only with entropy variables gradients)</li> </ul>	–
Shock viscous flux 2	<i>CHARACTER</i> : Viscous flux to be applied in the elements where $a = 1$ . Options are: <ul style="list-style-type: none"> <li>• Physical</li> <li>• Guermond-Popov (only with entropy variables gradients)</li> </ul>	–

Table 5.2: Keywords for shock-capturing algorithms in the Navier-Stokes equations – continuation.

Keyword	Description	Default value
Shock update strategy	<i>CHARACTER</i> : Method to compute the variable parameter of the specified shock-capturing approach in the middle region of the sensor. Options are: <ul style="list-style-type: none"> <li>• Constant</li> <li>• Sensor</li> <li>• Smagorinsky: only for <i>non-filtered</i> and <i>SVV</i></li> </ul>	Constant
Shock mu 1	<i>REAL/CHARACTER(*)</i> : Viscosity parameter $\mu_1$ , or $C_s$ in the case of LES coupling	0.0
Shock alpha 1	<i>REAL</i> : Viscosity parameter $\alpha_1$	0.0
Shock mu 2	<i>REAL</i> : Viscosity parameter $\mu_2$	$\mu_1$
Shock alpha 2	<i>REAL</i> : Viscosity parameter $\alpha_2$	$\alpha_1$
Shock alpha/mu	<i>REAL</i> : Ratio between $\alpha$ and $\mu$ . It can be specified instead of $\alpha$ itself to make it dependent on the corresponding values of $\mu$ , and it is compulsory when using LES coupling	–
SVV filter cutoff	<i>REAL/CHARACTER(*)</i> : Cutoff of the filter kernel, $P$ . If "automatic", its value is adjusted automatically	"automatic"
SVV filter shape	<i>CHARACTER(*)</i> : Options are: <ul style="list-style-type: none"> <li>• Power</li> <li>• Sharp</li> <li>• Exponential</li> </ul>	Power
SVV filter type	<i>CHARACTER(*)</i> : Options are: <ul style="list-style-type: none"> <li>• Low-pass</li> <li>• High-pass</li> </ul>	High-pass
Sensor variable	<i>CHARACTER(*)</i> : Variable used by the sensor to detect shocks. Options are: <ul style="list-style-type: none"> <li>• rho</li> <li>• rho_u</li> <li>• rho_v</li> <li>• rho_w</li> <li>• u</li> <li>• v</li> <li>• w</li> <li>• p</li> <li>• rho_p</li> </ul>	rho_p
Sensor lower limit	<i>REAL</i> : Lower threshold of the central sensor region, $s_1$	<b>Mandatory keyword</b>
Sensor higher limit	<i>REAL</i> : Upper threshold of the central sensor region, $s_2$	<b>Mandatory keyword</b>
Sensor TE min N	<i>INTEGER</i> : Minimum polynomial order of the coarse mesh used for the truncation error estimation	1

Table 5.2: Keywords for shock-capturing algorithms in the Navier-Stokes equations – continuation.

Keyword	Description	Default value
Sensor TE delta N	Polynomial order difference between the solution mesh and its coarser representation	1
Sensor TE derivative	<i>CHARACTER*</i> : Whether the face terms must be considered in the estimation of the truncation error or not. Options are: <ul style="list-style-type: none"> <li>• Non-isolated</li> <li>• Isolated</li> </ul>	Isolated

### Spectral Vanishing Viscosity

The introduction of an SVV-filtered artificial flux helps dissipate high-frequency oscillations. The baseline viscous flux can be chosen as the Navier-Stokes viscous flux or the flux developed by Guermond and Popov. In any case, this flux is expressed in a modal base where it is filtered by any of the following three filter kernels:

- power:  $\hat{F}_i^{1D} = (i/N)^P$ ,
- sharp:  $\hat{F}_i^{1D} = 0$  if  $i < P$ ,  $\hat{F}_i^{1D} = 1$  elsewhere,
- exponential:  $\hat{F}_i^{1D} = 0$  if  $i \leq P$ ,  $\hat{F}_i^{1D} = \exp\left(-\frac{(i-N)^2}{(i-P)^2}\right)$  elsewhere.

The extension to three dimensions allows the introduction of two types of kernels based on the one-dimensional ones:

- high-pass:  $\hat{F}_{ijk}^H = \hat{F}_i^{1D} \hat{F}_j^{1D} \hat{F}_k^{1D}$ ,
- low-pass:  $\hat{F}_{ijk}^L = 1 - \left(1 - \hat{F}_i^{1D}\right) \left(1 - \hat{F}_j^{1D}\right) \left(1 - \hat{F}_k^{1D}\right)$ ,

being the low-pass one more dissipative and, thus, more suited to supersonic cases. The high-pass filter, on the other hand, works better as part of the SVV-LES framework for turbulent cases.

The cutoff parameter  $P$  can be set as "automatic", which uses a sensor to differentiate troubled elements from smooth regions. The stabilisation strategy then depends on the region:

- smooth regions:  $P = 4$ ,  $\mu = \mu_2$ ,  $\alpha = \alpha_2$ ,
- shocks:  $P = 4$ ,  $\mu = \mu_1$ ,  $\alpha = \alpha_1$ .

In addition to this, the viscosity  $\mu_1$  can be set to "Smagorinsky" to use the implemented SVV-LES approach. In this case, the  $\mu = \mu_{LES}$  viscosity is computed following a Smagorinsky formulation with  $C_s = \mu_2$  and the viscosity parameters do not depend on the region anymore,

$$\mu = C_s^2 \Delta^2 |S|^2, \quad \alpha = \alpha_1.$$

#### 5.1.2 Acoustic

The Ffowcs Williams and Hawkings (FWH) acoustic analogy is implemented as a complement to the compressible NS solver. It can run both during the execution of the NS (in-time) or as at a post-process step. The version implemented includes both the solid and permeable surface variations, but both of them for a static body subjected to a constant external flow, i.e. a wind tunnel case scenario. The specifications for the FWH are divided in two parts: the general definitions (including the surfaces) and the observers definitions. The former is detailed in Table 5.3, while the latter are defined in a block section, similar to the monitors (see Chapter 11):

```
#define acoustic observer 1
  name      = SomeName
  position  = [0.d0, 0.d0, 0.d0]
#end
```

To run the in-time computation, the observers must be defined in the control file. Beware that adding an additional observer will require to run the simulation again. To use the post-process computation, the solution on the surface must be saved at a regular time. Beware that it will need more storage. To run the post-process calculation the horses.tools binary is used, with a control file similar to the one use for the NS simulation (without monitors), and adding the keywords "tool type" and "acoustic files pattern", as explained in Table 5.3.

Table 5.3: Keywords for acoustic analogy

Keyword	Description	Default value
acoustic analogy	<i>CHARACTER(*)</i> : This is the main keyword for activating the acoustic analogy. The only options is: "FWH".	–
acoustic analogy permable	<i>LOGICAL</i> : Defines if uses a permable or solid approach.	.FALSE.
acoustic solid surface	<i>CHARACTER(*)</i> : Array containing the name of each boundary to be used as a surface for integration. In the form: '[bc1,bc2,bc3]'. Mandatory for using the solid surface variant.	–
acoustic surface file	<i>CHARACTER(*)</i> : Path to a fictitious surface that will be used for integration. It must be tailored made for the mesh. Mandatory for using the permable surface variant.	–
observers flush interval	<i>INTEGER</i> : Iteration interval to flush the observers information to the files.	100
acoustic solution save	<i>LOGICAL</i> : Defines wheter it saves the NS solution on the surface. Mandatory for post-process computation.	.FALSE.
acoustic save timestep	<i>REAL</i> : Controls the time or iteration at which the FWH will be calculated (and saved if specified). If the key is missing it will be done at each timestep.	–
acoustic files pattern	<i>CHARACTER(*)</i> : Pattern to the path of all the saved solutions on the surface (To be used in horses.tools for the post-process calculation).	–
tool type	<i>CHARACTER(*)</i> : Necessary for post-process calculation. Defines the type of post-process of horses.tools. For the FWH analogy the value must be "fwh post"	–

## 5.2 Incompressible Navier-Stokes

## 5.3 Cahn-Hilliard

## 5.4 Complementary Modes

### 5.4.1 Wall Function

The wall function overwrites the viscous flux on the specified boundaries based on an specific law using a Newman condition. It must be used as a complement of no slip boundary condition. Table 5.4 shows the parameters that can be set in the control file. The frictional velocity is calculated using the instantaneous values of the first node (either Gauss or Gauss-Lobatto) of the element neighbour of the face element (at the opposite side of the boundary face). Currently is only supported for the compressible Navier-Stokes solver.

The standard wall function uses the Reichardt law, solving the algebraic non-linear equation using the newton method to obtain the frictional velocity. The ABL function uses the logarithmic atmospheric boundary layer law, using the aerodynamic roughness; the frictional velocity is without using any numerical method.

### 5.4.2 Tripping

A numerical source term is added to the momentum equations to replicate the effect of a tripping mechanism used commonly in experimental tests. The forcing is described via the product of two independent functions: one that depends streamwise and vertical directions (space only) and the other one describing the spanwise direction and time (space and time). It can be used for the compressible NS, both LES and RANS. The keywords for the trip options are listed in table 5.5.

Table 5.4: Keywords for Wall Function

Keyword	Description	Default value
Wall Function	<i>CHARACTER(*)</i> : This is the main keyword for activating the wall function. Identifies the wall law to be used. Options are: <ul style="list-style-type: none"> <li>• Standard: uses the Reichardt law.</li> <li>• ABL: uses the atmospheric boundary layer law.</li> </ul>	–
Wall Function Boundaries	<i>CHARACTER(*)</i> : Array containing the name of each boundary to be used. In the form: '[bc1,bc2,bc3]'. Mandatory for using the wall function.	–
Wall Function kappa	<i>REAL</i> : von Karman constant	0.38
Wall Function C	<i>REAL</i> : Log law 'C' constant	4.1
Wall Function Seed	<i>REAL</i> : Initial value for the newton method	1.0
Wall Function Damp	<i>REAL</i> : Initial value damp for the newton method	1.0
Wall Function Tolerance	<i>REAL</i> : Tolerance for the newton method	$10^{-10}$
Wall Function max iter	<i>INTEGER</i> : Maximum number of iterations for the newton method	100
Wall Roughness	<i>REAL</i> : Aerodynamic roughness for the ABL wall function. Mandatory value for the ABL law.	–
Wall Plane Displacement	<i>REAL</i> : Plane displacement due to roughness for the ABL wall function	0.0

Table 5.5: Keywords for Tripping model

Keyword	Description	Default value
use trip	<i>LOGICAL</i> : This is the main keyword for activating the trip	.FALSE.
trip time scale	<i>REAL</i> : Time interval between the change of the time dependent part of the trip.	<b>Mandatory</b>
trip number of modes	<i>INTEGER</i> : Number of Fourier modes in the spanwise direction of the trip.	<b>Mandatory</b>
trip z points	<i>INTEGER</i> : Number of points to create the Fourier Transformation of the spanwise direction, it must be greater than the number of modes and should be ideally equal to the number of discretization points of the mesh in the same direction.	<b>Mandatory</b>
trip attenuation	<i>REAL ARRAY(2)</i> : Length scale of the gaussian attenuation of the trip, the first position is the streamwise direction and the second is the wall-normal direction.	<b>Mandatory</b>
trip zone	<i>CHARACTER(*) ARRAY(:)</i> : Boundary condition name that contains at least one surface where the trip center is located. It can be either one or two boundary conditions, the latter used to generate a trip in two different positions (i.e. pressure and suction sides of an airfoil).	<b>Mandatory</b>
trip center	<i>REAL</i> : Position of the origin of the trip in the streamwise direction.	<b>Mandatory</b>
trip center 2	<i>REAL</i> : Position of the origin of the second trip, if used, in the streamwise direction.	–
trip amplitude	<i>REAL</i> : Maximum time varying amplitude of the trip.	1.0
trip amplitude steady	<i>REAL</i> : Maximum steady amplitude of the trip.	0.0
random seed 1	<i>INTEGER</i> : Number used to initialize the random number generator of the trip. It can vary in different simulations but must remain constant for a restart.	930187532
random seed 2	<i>INTEGER</i> : Number used to initialize the random number generator of the trip. It can vary in different simulations but must remain constant for a restart.	597734650

## Chapter 6

# Implicit Solvers with Newton linearisation

### 6.1 General Keywords

The keywords for the implicit solvers are listed in table 6.1

Table 6.1: Keywords for implicit solvers.

Keyword	Description	Default value
<b>time integration</b>	<i>CHARACTER</i> : This is the main keyword for activating the implicit solvers. The value of it should be set to 'implicit' for the BDF solvers and to 'rosenbrock' for Rosenbrock schemes.	'explicit'
linear solver	<i>CHARACTER</i> : Specifies the linear solver that has to be used. Options are: <ul style="list-style-type: none"><li>• 'petsc': PETSc library Krylov-Subspace methods. Available in serial, but use with care (PETSc is not thread-safe, so OpenMP is not recommended). Only available in parallel (MPI) for preallocated Jacobians (see next section).</li><li>• 'pardiso': Intel MKL PARDISO. Only available in serial or with OpenMP.</li><li>• 'matrix-free gmres': A matrix-free version of the GMRES algorithm. Can be used without preconditioner or with a recursive GMRES preconditioner using 'preconditioner=GMRES'. Available in serial and parallel (OpenMP+MPI)</li><li>• 'smooth': Traditional iterative methods. One can select either 'smoother=WeightedJacobi' or 'smoother=BlockJacobi'.</li><li>• 'matrix-free smooth': A matrix-free version of the previous solver. Only available with 'smoother=BlockJacobi'.</li></ul>	'petsc'

### 6.2 Keywords for the BDF Methods

The BDF methods implemented in HORSES3D use a Newton's method



Table 6.2: Keywords for the BDF solvers.

Keyword	Description	Default value
bdf order	<i>INTEGER</i> : If present, the solver uses a BDF solver of the specified order. BDF1 - BDF5 are available, and BDF2 - BDF5 require constant time steps.	1
jacobian by convergence	<i>LOGICAL</i> : When <i>.TRUE.</i> , the Jacobian is only computed when the convergence falls beneath a threshold (hard-coded). This improves performance.	<i>.FALSE.</i>
compute jacobian every	<i>INTEGER</i> : Forces the Jacobian to be computed in an interval of iterations that is specified.	Inf
print newton info	<i>LOGICAL</i> : If <i>.TRUE.</i> , the information of the Newton iterations will be displayed.	<i>'FALSE.'</i>
implicit adaptive dt	<i>LOGICAL</i> : Specifies if the time-step should be computed according to the convergence behavior of the Newton iterative method and the linear solver.	<i>.FALSE.</i>
newton tolerance	<i>REAL</i> : Specifies the tolerance for the Newton's method.	$10^{-6}$ for time-accurate simulations, or $MaxResidual \times a$ for steady-state simulations, where $a$ is the keyword <i>newton factor</i>
newton max iter	<i>INTEGER</i> : Maximum number of Newton iterations for BDF solver.	30
linsolver max iter	<i>INTEGER</i> : Maximum number of iterations to be taken by the linear solver. This keyword only affects iterative linear solvers.	500
newton factor	<i>REAL</i> : In simulations that are not time-accurate, the tolerance of the Newton's method is a function of the residual: $MaxResidual \times a$ , where $a$ is the specified value.	$10^{-3}$
linsolver tol factor	<i>REAL</i> : The linear solver tolerance is a function of the absolute error of the Newton's method: $tol = \ e\ _{\infty} * a^i$ , where $e$ is the absolute error of the Newton's method, $i$ is the Newton iteration number, and $a$ is the specified value.	0.5
newton first norm	<i>REAL</i> : Specifies an assumed infinity norm of the absolute error of the Newton's method at the iteration 0 of the time step 1. This can change the behavior of the first Newton iterative method because of the dependency of the linear system tolerance on the absolute error of the Newton's method (see keyword <i>linsolver tol factor</i> ).	0.2

### 6.3 Keywords for the Rosenbrock-Type Implicit Runge-Kutta Methods

Table 6.3: Keywords for the Rosenbrock schemes.

Keyword	Description	Default value
rosenbrock scheme	<i>CHARACTER</i> : Rosenbrock scheme to be used. Currently, only the <i>RO6-6</i> is implemented.	–

### 6.4 Jacobian Specifications

The Jacobian must be defined using a block of the form:

```
#define Jacobian
    type = 2
```

```

    print info = .TRUE.
    preallocate = .TRUE.
#end

```

Table 6.4: Keywords for Jacobian definition block.

Keyword	Description	Default value
type	<p><i>INTEGER</i>: Specifies the type of Jacobian matrix to be computed. Options are:</p> <ol style="list-style-type: none"> <li>1. Numerical Jacobian: Uses a coloring algorithm and a finite difference method to compute the DG Jacobian matrix (only available with shared memory parallelization).</li> <li>2. Analytical Jacobian: Available with shared (OpenMP) or distributed (MPI) memory parallelization for advective and/or diffusive nonlinear conservation laws, <b>BUT</b> only for the standard DGSEM (no split-form).</li> </ol>	<b>Mandatory Keyword</b>
print info	<i>LOGICAL</i> : Specifies the verbosity of the Jacobian subroutines	.TRUE.
preallocate	<i>LOGICAL</i> : Specifies if the Jacobian must be allocated in preprocessing (.TRUE. - only available for advective/diffusive nonlinear conservation laws) or every time it is computed (.FALSE.)	.FALSE.

## Chapter 7

# Explicit Solvers

Explicit time integration schemes available in HORSES3D. The main keywords to use it are shown in Table 7.1.

Table 7.1: Keywords for the multigrid solver.

Keyword	Description	Default value
<b>time integration</b>	<i>CHARACTER</i> : This is the main keyword to activate the multigrid solvers. The value of it should be set to 'FAS' for the Full Approximation Scheme (FAS) nonlinear multigrid solvers and to 'AnisFAS' for anisotropic FAS schemes.	'explicit'
<b>simulation type</b>	<i>CHARACTER</i> : Specifies if HORSES3D must perform a 'steady-state' or a 'time-accurate'. If 'time-accurate' the solver switches to BDF integration and uses FAS as a pseudo problem solver. Compatible only with 'FAS'.	'steady-state'
explicit method	<i>CHARACTER</i> : Select desired Runge-Kutta solver. Options are: 'Euler', 'RK3', 'RK5' and 'RKOpt'.	RK3
rk order	<i>INTEGER</i> : Order of Runge-Kutta method optimized for steady-state solver ('RKOpt'). Possible orders are from 2 to 7.	2

## Chapter 8

# Nonlinear $p$ -Multigrid solver (FAS)

The code has an implementation of the Full Approximation Scheme (FAS) nonlinear  $p$ -multigrid method. The main keywords to use it are shown in Table 8.1.

Table 8.1: Keywords for the multigrid solver.

Keyword	Description	Default value
<b>time integration</b>	<i>CHARACTER</i> : This is the main keyword to activate the multigrid solvers. The value of it should be set to 'FAS' for the Full Approximation Scheme (FAS) nonlinear multigrid solvers and to 'AnisFAS' for anisotropic FAS schemes.	'explicit'
<b>simulation type</b>	<i>CHARACTER</i> : Specifies if HORSES3D must perform a 'steady-state' or a 'time-accurate'. If 'time-accurate' the solver switches to BDF integration (the exact method can be set using 'bdf order' option) and uses FAS as a local steady-state problem solver. Compatible only with 'FAS'.	'steady-state'
multigrid levels	<i>INTEGER</i> : Number of multigrid levels for the computations.	<b>Mandatory keyword</b>
delta n	<i>INTEGER</i> : Interval of reduction of polynomial order for creating coarser multigrid levels.	1
multigrid output	<i>LOGICAL</i> : If .TRUE., the residuals at the different multigrid levels will be displayed.	.FALSE.
mg sweeps	<i>INTEGER</i> : Number of smoothing sweeps to be taken.	1*
mg sweeps pre	<i>INTEGER</i> : Number of pre-smoothing sweeps to be taken.	1*
mg sweeps post	<i>INTEGER</i> : Number of post-smoothing sweeps to be taken.	1*
mg sweeps coarsest	<i>INTEGER</i> : Number of pre- and post-smoothing sweeps to be taken on the coarsest multigrid level.	Average between pre-sweeps and post-sweeps
mg sweeps exact	<i>INTEGER(:)</i> : Alternative to 'mg sweeps'. Defines exact number of pre- and post-smoothing sweeps to be taken on each level. Index of the array indicates the MG level for the sweeps to be performed, e.g. [1,4] performs 1 pre-sweep and 1 post-sweep on level 1 and 4 pre-post-sweeps on level 2.	1*
mg sweeps pre exact	<i>INTEGER(:)</i> : Alternative to 'mg sweeps pre'. Defines exact number of pre-smoothing sweeps to be taken on each level. Index of the array indicates the MG level for the sweeps to be performed, e.g. [1,4] performs 1 pre-sweep on level 1 and 4 pre-sweeps on level 2.	1*
mg sweeps post exact	<i>INTEGER(:)</i> : Alternative to 'mg sweeps post'. Defines exact number of post-smoothing sweeps to be taken on each level. Index of the array indicates the MG level for the sweeps to be performed, e.g. [1,4] performs 1 post-sweep on level 1 and 4 post-sweeps on level 2.	1*

Table 8.1: Keywords for the multigrid solver - continued.

Keyword	Description	Default value
mg smoother	<i>CHARACTER</i> : The smoothing technique to be used. The keywords and possible explicit smoothers are the same as the 'explicit method' in 7.1. For the semi-implicit residual relaxation use 'BIRK5'.	RK3
fasfmg residual	<i>REAL</i> : When this keyword is used, the code uses a full multigrid (FMG) method to obtain an initial condition for the simulation. The initial condition has the specified residual.	–
fasfmg save solutions	<i>LOGICAL</i> : Save the solutions that are obtained at the different FMG levels. Only usable when <i>fasfmg residual</i> is used.	.FALSE.
postsmooth option	<i>CHARACTER</i> : When this keyword is used, the code performs extra post-smoothing sweeps, so that the final residual after completing the post-smoothing is lower than the residual achieved by the pre-smoothing. The options are: <ul style="list-style-type: none"> <li>• <i>f-cycle</i>: Do the extra post-smoothing with an FMG cycle.</li> <li>• <i>smooth</i>: Do normal smoothing.</li> </ul>	–
smooth fine	<i>REAL</i> : Extra pre-smoothing is performed on a multigrid level of order $P$ , until a residual is obtained $\ \tilde{\mathfrak{R}}^P\ _\infty < \eta \ \tilde{\mathfrak{R}}^N\ _\infty$ , where $N$ is the polynomial order of the next (coarsest) grid, and $\eta$ is the specified value.	–
max mg sweeps	<i>INTEGER</i> : Maximum number of smoothing sweeps to be performed. This only makes sense if one uses the keywords <i>postsmooth option</i> and/or <i>smooth fine</i> .	10000
mg initialization	<i>LOGICAL</i> : Sets the initial explicit residual smoothing with RK3 and local time stepping.	.FALSE.
initial residual	<i>REAL</i> : Threshold for the $\ \tilde{\mathfrak{R}}^P\ _\infty$ after which solver switches from the 'mg initialization' settings to user specified.	1.0
initial cfl	<i>REAL</i> : CFL and DCFL number for initial residual smoothing.	0.1

\* The user must specify *mg sweeps pre* **and** *mg sweeps post*, or *mg sweeps*.

## Chapter 9

# p-Adaptation Methods

The p-adaptation methods are used when the p-adaptation region is specified in the control file:

```
#define p-adaptation
  Truncation error type = isolated
  truncation error      = 1.d-2
  Nmax                  = [10,10,10]
  Nmin                  = [2,2,2]
  Conforming boundaries = [InnerCylinder,sphere]
  order across faces    = N*2/3
  increasing            = .FALSE.
  write error files     = .FALSE.
  adjust nz            = .FALSE.
  mode                  = time
  interval              = 1.d0
  restart files         = .TRUE.
  max N decrease        = 1
  padapted mg sweeps pre      = 10
  padapted mg sweeps post    = 12
  padapted mg sweeps coarsest = 20
#end
```

Table 9.1: Keywords for the p-adaptation algorithms.

Keyword	Description	Default value
truncation error type	<i>CHARACTER</i> : Can be either "isolated" or "non-isolated".	isolated
truncation error	<i>REAL</i> : Target truncation error for the p-adaptation algorithm.	<b>Mandatory keyword</b>
coarse truncation error	<i>REAL</i> : Truncation error used for coarsening.	same as truncation error
Nmax	<i>INTEGER</i> (3): Maximum polynomial order in each direction for the p-adaptation algorithm.	<b>Mandatory keyword</b>
Nmin	<i>INTEGER</i> (3): Minimum polynomial order in each direction for the p-adaptation algorithm.	[1,1,1]
conforming boundaries	<i>CHARACTER</i> (*): Specifies the boundaries of the geometry that must be forced to be conforming after the p-adaptation process.	–
order across faces	<i>CHARACTER</i> : Mathematical expression to specify the maximum polynomial order jump across faces. Currently, only $N * 2/3$ and $N - 1$ are supported.	$N - 1$
increasing	<i>LOGICAL</i> : If .TRUE. the multi-stage FMG adaptation algorithm is used.	.FALSE.

Table 9.1: Keywords for the p-adaptation algorithms - continued.

Keyword	Description	Default value
write error files	<i>LOGICAL</i> : If <i>.TRUE.</i> , the program writes a file per element containing the directional tau-estimations. The files are stored in the folder <i>./TauEstimation/</i> . When the simulation has several adaptation stages, the new information is just appended.	<i>.FALSE.</i>
adjust nz	<i>LOGICAL</i> : If <i>.TRUE.</i> , the order accross faces is adjusted in the directions xi, eta, and zeta of the face (being zeta the normal direction). If <i>.FALSE.</i> , the order is only adjusted in the xi and eta directions. The adjustment currently consists (hard-coded) in allowing jumps in the polynomial order of at most 1.	<i>.FALSE.</i>
mode	<i>CHARACTER</i> : p-Adaptation mode. Can be <i>static</i> , <i>time</i> or <i>iteration</i> . Static p-adaptation is performed once at the beginning of a simulation for steady or unsteady simulations. Unsteady adaptation can be by <i>time</i> or by <i>iteration</i> .	<i>static</i>
interval	<i>INTEGER/REAL</i> : In dynamic p-adaptation cases, this keyword specifies the iteration (integer) or time (real) interval for p-adaptation.	<i>huge number</i>
restart files	<i>LOGICAL</i> : If <i>.TRUE.</i> , the program writes restart files before and after the p-adaptation.	<i>.FALSE.</i>
max N decrease	<i>INTEGER</i> : Maximum decrease in the polynomial order in every p-adaptation procedure.	$N - N_{min}$
post smoothing residual	<i>REAL</i> : Specifies the maximum allowable deviation of $\partial_t q$ after the p-adaptation procedure.	–
post smoothing method	<i>CHARACTER</i> : Either RK3 or FAS.	RK3, if the last keyword is activated
estimation files	<i>CHARACTER</i> : Name of the folder that contains the error estimations obtained with the multi tau-estimation (section 9.1).	–
estimation files number	<i>INTEGER(2)</i> : First and last estimation stages to be used for p-adaptation.	Mandatory if last keyword is used.
padapted $\ll keyword \gg$	<i>MULTIPLE</i> : Specifies control file keywords that should be replaced after the adaptation procedure. Currently, only 'mg sweeps', 'mg sweeps pre', 'mg sweeps post', and 'mg sweeps coarsest' are supported.	–

## 9.1 Multiple truncation error estimations

A static p-adaptation procedure can be driven by a set of error estimations, which have to be performed beforehand in a simulation with the following block:

```
#define multi tau-estimation
    truncation error type = isolated
    interval               = 10
    folder                 = MultiTau
#end
```

## Chapter 10

# Immersed boundary method

The immersed boundary is activated during the simulation if the following lines are specified in the control file:

```
#define IBM
  name                      = myIBM
  active                    = .true.
  penalization              = 1.0d-6
  semi implicit             = .false.
  number of objects         = 5
  number of interpolation points = 15
  band region               = .true.
  band region coefficient    = 1.3
  compute distance          = .true.
  clip axis                 = 1
  aab                       = .false.
  describe                  = .true.
  plot obb                  = .false.
  plot kdtree               = .false.
  plot mask                 = .true.
  plot band points          = .false.
#end
```

A folder called 'IBM' must be created.

Table 10.1: Keywords for the immersed boundary method.

Keyword	Description	Default value
name	<i>CHARACTER</i> : name assigned to immersed boundary method job	
active	<i>LOGICAL</i> : When .TRUE. the immersed boundary method is active.	.FALSE.
penalization	<i>REAL</i> : Specifies the value of the penalization term, <i>i.e.</i> $\eta$ .	$\Delta t$
semi implicit	<i>LOGICAL</i> : The source term is treated in a semi-implicit manner.	.FALSE.
number of objects	<i>INTEGER</i> : Specifies the maximum number of objects inside a leaf of the KD-tree.	5
number of interpolation points	<i>INTEGER</i> : Number of points used for the interpolation of the variables' values on the surface. It's needed for the computation of the forces.	15
band region	<i>LOGICAL</i> : If it's true, the band region is computed, otherwise it is not.	.FALSE.
band region coefficient	<i>INTEGER</i> : A region $n$ -times the oriented bounding box is created (where $n$ is the band region coefficient): all the points inside this region belong to the band region. The band region's points are used for the interpolation. It must be greater then 1 and 'band region' must be .true. .	1.5



Table 10.1: Keywords for the immersed boundary method - continued.

Keyword	Description	Default value
compute distance	<i>LOGICAL</i> : If it's true, the distance between the points in the band region and the stl file is computed, otherwise it is not. If the distance is not required, turn it off since it is an expansive operation.	.FALSE.
clip axis	<i>INTEGER</i> : It's the axis along which the stl is cut. It is only needed if the forces are computed so that the integration of the variables is performed only on the portion of the stl surface lying inside the mesh. 1 corresponds to <i>x-axis</i> , 2 with <i>y-axis</i> and 3 with <i>z-axis</i>	0
aab	<i>LOGICAL</i> : The Axis Aligned Box is computed instead of the Oriented Bounding box. It is recommended when ' <i>clip axis</i> ' $\neq$ 0.	.FALSE.
describe	<i>LOGICAL</i> : The immersed boundary parameters are printed on the screen.	.FALSE.
plot obb	<i>LOGICAL</i> : The oriented-bounding box is plotted.	.FALSE.
plot kdtree	<i>LOGICAL</i> : The kd-tree is plotted.	.FALSE.
plot mask	<i>LOGICAL</i> : The degrees of freedom belonging to the mask are plotted.	.FALSE.
plot band points	<i>LOGICAL</i> : The band region's points are plotted.	.FALSE.

### 10.0.1 STL file

Immersed boundary requires, along with the mesh, a STL file. It must be put in the MESH folder with the mesh. The STL file name must be in lowercase character. In some programs, like AutoCAD, a STL file has always positive coordinates: the mesh should be built according to this consideration.

In the case of 2D simulations, the STL can be automatically cut by horses3D through the addition of the line *clip axis* (described in the previous section) so that only the STL portions inside the mesh are considered.

Table 10.2: Keywords for stl files.

Keyword	Description	Default value
number of stl =	<i>INTEGER</i> : Number of stl files.	1
stl file name =	<i>CHARACTER</i> : The name of the STL file, without extension; it has to be inside the folder "MESH" .	<b>Mandatory keyword</b>
stl file nameN =	<i>CHARACTER</i> : The name of the N <sup>th</sup> -STL file (where N starts from 2), without extension; for the first STL just use "stl file name". It has to be inside the folder "MESH" .	none

### 10.0.2 Computing forces

In order to compute the forces on a body, the monitor should be defined as usual but the "Marker=" has to be equal to the name of the stl file on which the user wants to compute the forces. Given a STL file called "stlname", the monitor should be:

```
#define surface monitor 1
    marker = stlname
    .
    .
    .
#end
```

The result of this operation is a .tec file inside the RESULTS folder. This file contains a scalar or a vector data projected on the body surface.

### 10.0.3 Moving bodies

If one or more of the stl files move, then the following lines must be added:

```

#define stl motion 1
    stl name      = mySTL
    type          = rotation
    angular velocity = 134.5d0
    motion axis   = 2
#end

```

Table 10.3: Keywords for a moving stl.

Keyword	Description	Default value
stl name	<i>CHARACTER</i> : name of the moving stl; it has to be equal to the name of one of the stl files.	<b>mandatory keyword</b>
type	<i>CHARACTER</i> : Type of motion, it can be ROTATION or LINEAR.	<b>Mandatory keyword</b>
angular velocity	<i>REAL</i> : Specifies the angular velocity. It must be in [Rad]/[s]	<b>Mandatory keyword for rotation type</b>
velocity	<i>REAL</i> : Specifies the translation velocity. It must be in [m]/[s]	<b>Mandatory keyword for linear type</b>
motion axis	<i>REAL</i> : Specifies the axis along which the rotation/translation occurs.	<b>Mandatory keyword</b>

# Chapter 11

## Monitors

The monitors are specified individually as blocks in the control file. The only general keyword that can be specified is explained in Table 11.1.

Table 11.1: Keywords for monitors.

Keyword	Description	Default value
monitors flush interval	<i>INTEGER</i> : Iteration interval to flush the monitor information to the monitor files.	100

### 11.1 Residual Monitors

### 11.2 Statistics Monitor

```
#define statistics
  initial time      = 1.d0
  initial iteration = 10
  sampling interval = 10
  dump interval     = 20
  @start
#end
```

By default, the statistic monitor will average following variables:

- u
- v
- w
- uu
- vv
- ww
- uv
- uw
- vw

A keyword preceded by @ is used in real-time to signalize the solver what it must do with the statistics computation:

- @start
- @pause
- @stop
- @reset
- @dump

After reading the keyword, the solver performs the desired action and marks it with a star, e.g. @start\*.

**ATTENTION:** Real-time keywords may not work in parallel MPI computations. I depends on how the system is configured.

### 11.3 Probes

```

#define probe 1
    name      = SomeName
    variable  = SomeVariable
    position  = [0.d0, 0.d0, 0.d0]
#end

```

Table 11.2: Keywords for probes.

Keyword	Description	Default value
name	<i>CHARACTER</i> : Name of the monitor.	<b>Mandatory Keyword</b>
variable	<i>CHARACTER</i> : Variable to be monitored. Implemented options are: <ul style="list-style-type: none"> <li>• pressure</li> <li>• velocity</li> <li>• u</li> <li>• v</li> <li>• w</li> <li>• mach</li> <li>• k</li> </ul>	<b>Mandatory Keyword</b>
position	<i>REAL(3)</i> : Coordinates of the point to be monitored.	<b>Mandatory Keyword</b>

## 11.4 Surface Monitors

```

#define surface monitor 1
    name          = SomeName
    marker        = NameOfBoundary
    variable      = SomeVariable
    reference surface = 1.d0
    direction     = [1.d0, 0.d0, 0.d0]
#end

```

Table 11.3: Keywords for probes.

Keyword	Description	Default value
name	<i>CHARACTER</i> : Name of the monitor.	<b>Mandatory Keyword</b>
marker	<i>CHARACTER</i> : Name of the boundary where a variable will be monitored.	<b>Mandatory Keyword</b>
variable	<i>CHARACTER</i> : Variable to be monitored. Implemented options are: <ul style="list-style-type: none"> <li>• mass-flow</li> <li>• flow</li> <li>• pressure-force</li> <li>• viscous-force</li> <li>• force</li> <li>• lift</li> <li>• drag</li> <li>• pressure-average</li> </ul>	<b>Mandatory Keyword</b>
reference surface	<i>REAL</i> : Reference surface [area] for the monitor. Needed for "lift" and "drag" computations.	–
direction	<i>REAL(3)</i> : Direction in which the force is going to be measured. Needed for "pressure-force", "viscous-force" and "force". Can be specified for "lift" (default [0.d0,1.d0,0.d0]) and "drag" (default [1.d0,0.d0,0.d0])	–

## 11.5 Volume Monitors

Volume monitors compute the average of a quantity in the whole domain. They can be scalars(s) or vectors(v).

```
#define volume_monitor 1
    name      = SomeName
    variable = SomeVariable
#end
```

Table 11.4: Keywords for volume monitors.

Keyword	Description	Default value
name	<i>CHARACTER</i> : Name of the monitor.	<b>Mandatory Keyword</b>
variable	<i>CHARACTER</i> : Variable to be monitored. The variable can be scalar (s) or vectorial (v). Implemented options are:  <div> <div>(s) kinetic energy</div> <div>(s) kinetic energy rate</div> <div>(s) enstrophy</div> <div>(s) entropy</div> <div>(s) entropy rate</div> </div> <div> <div>(s) mean velocity</div> <div>(v) velocity</div> <div>(v) momentum</div> <div>(v) source</div> </div>	<b>Mandatory Keyword</b>

# Chapter 12

## Advanced User Setup

Advanced users can have additional control over a simulation without having to modify the source code and recompile the code. To do that, the user can provide a set of routines that are called in different stages of the simulation via the Problem file (*ProblemFile.f90*). A description of the routines of the Problem File can be found in section 12.1.

### 12.1 Routines of the Problem File: *ProblemFile.f90*

- UserDefinedStartup: Called before any other routines
- UserDefinedFinalSetup: Called after the mesh is read in to allow mesh related initializations or memory allocations.
- UserDefinedInitialCondition: called to set the initial condition for the flow. By default it sets an uniform initial condition, but the user can change it.
- UserDefinedState1, UserDefinedNeumann: Used to define an user-defined boundary condition.
- UserDefinedPeriodicOperation: Called before every time-step to allow periodic operations to be performed.
- UserDefinedSourceTermNS: Called to apply source terms to the equation.
- UserDefinedFinalize: Called after the solution computed to allow, for example error tests to be performed.
- UserDefinedTermination: Called at the the end of the main driver after everything else is done.

### 12.2 Compiling the Problem File

The Problem File must be compiled using a specific Makefile that links it with the libraries of the code. If you are using the *horses/dev* environment module, you can get templates of the *Problemfile.f90* and *Makefile* with the following commands:

```
$ horses-get-makefile
$ horses-get-problemfile
```

Otherwise, search the test cases for examples.

To run a simulation using user-defined operations, create a folder called **SETUP** on the path where the simulation is going to be run. Then, store the modified *ProblemFile.f90* and the *Makefile* in **SETUP**, and compile using:

```
$ make <<Options>>
```

where again the options are (bold are default):

- **MODE=DEBUG/RELEASE**
- **COMPILER=ifort/gfortran**
- **COMM=PARALLEL/SEQUENTIAL**

- PLATFORM=MACOSX/**LINUX**
- ENABLE\_THREADS=NO/**YES**

# Chapter 13

## Postprocessing

For postprocessing the Simulation Results

### 13.1 Visualization with Tecplot Format: *horses2plt*

HORSES3D provides a script for converting the native binary solution files (\*.hsol) into tecplot ASCII format (\*.tec), which can be visualized in Pareview or Tecplot. Usage:

```
$ horses2plt SolutionFile.hsol MeshFile.hmesh <<Options>>
```

The options comprise following flags:

Table 13.1: Flags for *horses2plt*.

Flag	Description	Default value
--output-order=	<i>INTEGER</i> : Output order nodes. The solution is interpolated into the desired number of points.	Not Present
--output-basis=	<i>CHARACTER</i> : Either <i>Homogeneous</i> (for equispaced nodes, or <i>Gauss</i> .	<i>Gauss</i> *
--output-mode=	<i>CHARACTER</i> : Either <i>multizone</i> or <i>FE</i> . The option <i>multizone</i> generates a Tecplot zone for each element. The option <i>FE</i> generates only one Tecplot zone for the fluid and one for each boundary (if --boundary-file is defined). Each subcell is mapped as a linear finite element. This format is faster to read by Paraview and Tecplot.	<i>multizone</i>
--output-variables=	<i>CHARACTER</i> : Output variables separated by commas. A complete description can be found in Section 13.1.1.	Q
--dimensionless	Specifies that the output quantities must be dimensionless	Not Present
--partition-file=	<i>CHARACTER</i> : Specifies the path to the partition file (*.pmesh) to export the MPI ranks of the simulation.	Not Present
--boundary-file=	<i>CHARACTER</i> : Specifies the path to the boundary mesh file (*.bmesh) to export the surfaces as additional zones of the Tecplot file.	Not Present

\* *Homogeneous* when --output-order is specified

Additionally, depending on the type of solution file, the user can specify additional options.

#### 13.1.1 Solution Files (\*.hsol)

For standard solution files, the user can specify which variables they want to be exported to the Tecplot file with the flag --output-variables=. The options are:

- *Q* (default)
- *rho*
- *u*
- *v*
- *w*
- *p*
- *T*
- *Mach*
- *s*
- *Vabs*
- *V*
- *Ht*
- *rho*
- *rho*
- *rho*



- *rhoe*
- *Ax\_Xi*
- *gradV*
- *u\_z*
- *omega\_x*
- *c*
- *Ax\_Eta*
- *u\_x*
- *v\_z*
- *omega\_y*
- *Nxi*
- *Ax\_Zeta*
- *v\_x*
- *w\_z*
- *omega\_z*
- *Neta*
- *ThreeAxes*
- *w\_x*
- *c\_x*
- *omega\_z*
- *Nzeta*
- *Axes*
- *u\_y*
- *c\_y*
- *omega\_abs*
- *Nav*
- *mpi\_rank*
- *v\_y*
- *c\_z*
- *Qcrit*
- *N*
- *eID*
- *w\_y*
- *omega*

### 13.1.2 Statistics Files (\*.stats.hsol)

Statistics files generate following variables by default (being  $S_{ij}$  the components of the Reynolds Stress tensor):

- Umean
- Sxx
- Sxy
- Vmean
- Syx
- Sxz
- Wmean
- Szz
- Syz

## 13.2 Extract geometry

Under construction.

## 13.3 Merge statistics tool

Tool to merge several statistics files. The usage is the following:

```
$ horses.mergeStats *.hsol --initial-iteration=INTEGER --file-name=CHARACTER
```

Some remarks:

- Only usable with statistics files that are obtained with the "reset interval" keyword and/or with individual consecutive simulations.
- Only constant time-stepping is supported.
- Dynamic p-adaptation is currently not supported.