HORSES3D

A **H**igh-**Or**der (DG) **S**pectral **E**lement **S**olver

User Manual

Numath group

October 10, 2023

Contents

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/HPC/**RELEASE**
  - COMPILER=ifort/**gfortran**
  - COMM=PARALLEL/**SEQUENTIAL**
  - ENABLE_THREADS=NO/**YES**
  - WITH_MKL=YES/**NO**

  For example:

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

- MODE=DEBUG/HPC/**RELEASE** flag enables various compiler flags for different levels of code optimization. Furthermore, MODE=HPC disables residual file writing to improve performance.
- 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 rutine 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 -¿ options -¿ mesh -¿ general -¿ 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	*CHARACTER*: Path and name of the output file. The name of this file is used for naming other output files.	**Mandatory keyword**
simulation type	*CHARACTER*: Specifies if HORSES3D must perform a 'steady-state' or a 'time-accurate' simulation.	'steady-state'
time integration	*CHARACTER*: 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	*INTEGER*: Polynomial order to be assigned uniformly to all the elements of the mesh. If the keyword *polynomial order file* is specified, the value of this keyword is overridden.	–*
polynomial order i polynomial order j polynomial order k	*INTEGER*: 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 *polynomial order file* is specified, the value of this keyword is overridden.	–*
polynomial order file	*CHARACTER*: Path to a file containing the polynomial order of each element in the domain.	–*
restart	*LOGICAL*: If .TRUE., initial conditions of simulation will be read from restart file specified using the keyword *restart file name*.	**Mandatory keyword**
cfl	*REAL*: A constant related with the **convective** Courant-Friedrichs-Lewy (CFL) condition that the program will use to compute the time step size.	–**
dcfl	*REAL*: A constant related with the **diffusive** Courant-Friedrichs-Lewy (DCFL) condition that the program will use to compute the time step size.	–**
dt	*REAL*: Constant time step size.	–**
final time	*REAL*: This keyword is mandatory for time-accurate solvers	–
mesh file name	*CHARACTER*: Name of the mesh file. The currently supported formats are *.mesh* (SpecMesh file format) and *.h5* (HOPR hdf5 file format).	**Mandatory keyword**
mesh inner curves	*LOGICAL*: 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	*INTEGER*: *Maximum* number of time steps that the program will compute.	**Mandatory keyword**

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

Keyword	Description	Default value
output interval	*INTEGER*: 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.	**Mandatory keyword**
convergence tolerance	*REAL*: Residual convergence tolerance for steady-state cases	**Mandatory keyword**
partitioning	*CHARACTER*: 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	*CHARACTER*: 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 (ρ, 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 *ManufacturedSolutions.f90*. 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	*CHARACTER*: Type of boundary condition to be applied. Options are: Inflow, Outflow, NoSlipWall, FreeSlipWall, Periodic, User-defined.	N/A

By default, NoSlipWall is adiabatic. Isothermal wall can be activated with the following block:

```
#define boundary myBoundary1__myBoundary2__myBoundary3
        type = NoSlipWall
        wall type (adiabatic/isothermal) = isothermal
        wall temperature = 2000.0d0  !Wall temperature in K
# end
```

It is also possible to set a moving wall with the keyword wall velocity = *value*.

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 commercial 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'.

Chapter 4

Spatial Discretization

HORSES3D uses a high-order discontinuous Galerkin Spectral Element (DGSEM) method. As part of the numerical method, several options of volume and surface discretization options can be selected. Some options are only available in certain solvers.

Table 4.1: Keywords for spatial discretization.

Keyword	Description	Default value
Discretization nodes	*CHARACTER*: Node type to use in the solution. Options are: • Gauss • Gauss-Lobatto	'Gauss'
inviscid discretization	*CHARACTER*: DG discretization of inviscid fluxes. Options are: • Standard • Split-Form	'Standard'
averaging	*CHARACTER*: Type of averaging function to use in numerical fluxes and split-forms (if in use). Options are: • Standard • Morinishi • Ducros • Kennedy-Gruber • Pirozzoli • Entropy conserving • Chandrasekar • Skew-symmetric 1 (only for Incompressible Navier–Stokes) • Skew-symmetric 2 (only for Incompressible Navier–Stokes)	–
viscous discretization	*CHARACTER*: Method for viscous fluxes. Options are: • BR1 • BR2 • IP	'BR2'

Table 4.1: Keywords for spatial discretization - continued.

Keyword	Description	Default value
riemann solver	*CHARACTER*: Riemann solver for inviscid fluxes. Options are: • Central • Roe (Only for compressible Navier–Stokes) • Standard Roe (Only for compressible Navier–Stokes) • Roe-Pike (Only for compressible Navier–Stokes) • Low dissipation Roe (Only for compressible Navier–Stokes) • Lax-Friedrichs (Only for compressible and Incompressible Navier–Stokes) • ES Lax-Friedrichs (Only for compressible Navier–Stokes, not including RANS) • u-diss (Only for compressible Navier–Stokes, not including RANS) • Rusanov (Only for compressible Navier–Stokes) • Matrix dissipation (Only for compressible Navier–Stokes) • Viscous NS (Only for compressible Navier–Stokes, not including RANS) • Exact (Only for Incompressible Navier-Stokes, and Multiphase)	**Mandatory keyword**

Chapter 5

Restarting a Case

Table 5.1: Keywords for restarting a case.

Keyword	Description	Default value
restart	*LOGICAL*: If .TRUE., initial conditions of simulation will be read from restart file specified using the keyword *restart file name*.	**Mandatory keyword**
restart file name	*CHARACTER*: Name of the restart file to be written and, if keyword *restart* = .TRUE., also name of the restart file to be read for starting the simulation.	**Mandatory keyword**
restart polorder	*INTEGER*: 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 nodetype	*CHARACTER*: node type of the solution to restart from (*Gauss* or *Gauss-Lobatto*). This keyword is only needed when the restart node type is different than the current case.	same as case's
restart polorder file	*CHARACTER*: 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	*CHARACTER*: Path to solution file. This can be used to estimate the discretization error of a solution when restarting from a higher-order solution.	–
NS load from NSSA	*LOGICAL*: Used only for NS simulations that are restarted from RANS SA.	.FALSE.

Chapter 6

Physics related keyword

6.1 Compressible flow

Table 6.1: Keywords for compressible flow (Euler / Navier-Stokes).

Keyword	Description	Default value
Mach number	*REAL*:	**Mandatory keyword**
Reynolds number	*REAL*:	**Mandatory keyword**
Prandtl number	*REAL*:	0.72
Turbulent Prandtl number	*REAL*:	Equal to Prandtl
AOA theta	*REAL*: Angle of attack (degrees), based on the spherical coordinates polar angle (θ) definition	0.0
AOA phi	*REAL*: Angle of attack (degrees), based on the spherical coordinates azimuthal angle (φ) definition	0.0
LES model	*CHARACTER(*)*: Options are: • Vreman • Wale • Smagorinsky • None	None
Wall model	*CHARACTER*:	linear

6.1.1 Shock-capturing

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 μ (similar to the viscosity of the Navier-Stokes equations) and α (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 6.2: Keywords for shock-capturing algorithms in the Navier-Stokes equations.

Keyword	Description	Default value
Enable shock-capturing	*LOGICAL*: Switch on/off the shock-capturing stabilization	.FALSE.
Shock sensor	*CHARACTER*: Type of sensor to be used to detect discontinuous regions Options are: • Zeros: always return 0 • Ones: always return 1 • Integral: integral of the sensor variable inside each element • Integral with sqrt: square root of the Integral sensor • Modal: based on the relative weight of the higher order modes • Truncation error: estimate the truncation error of the approximation • Aliasing error: estimate the aliasing error of the approximation • GMM: clustering sensor based on the divergence of the velocity and the gradient of the pressure	Integral
Shock first method	*CHARACTER*: Method to be used in the middle region of the sensor ($a \in [0, 1]$). Options are: • None: Do not apply any smoothing • Non-filtered: Apply the selected viscous flux without SVV filtering • SVV: Apply an entropy-stable, SVV-filtered viscous flux	None
Shock second method	*CHARACTER*: Method to be used in the top-most region of the sensor ($a = 1$). Options are: • None: Do not apply any smoothing • Non-filtered: Apply the selected viscous flux without SVV filtering • SVV: Apply an entropy-stable, SVV-filtered viscous flux	None
Shock viscous flux 1	*CHARACTER*: Viscous flux to be applied in the elements where $a \in [0, 1]$. Options are: • Physical • Guermond-Popov (only with entropy variables gradients)	–

Table 6.2: Keywords for shock-capturing algorithms in the Navier-Stokes equations – continuation.

Keyword	Description	Default value
Shock viscous flux 2	*CHARACTER*: Viscous flux to be applied in the elements where $a = 1$. Options are: • Physical • Guermond-Popov (only with entropy variables gradients)	–
Shock update strategy	*CHARACTER*: Method to compute the variable parameter of the specified shock-captruing approach in the middle region of the sensor. Options are: • Constant • Sensor • Smagorinsky: only for *non-filtered* and *SVV*	Constant
Shock mu 1	*REAL/CHARACTER(*)*: Viscosity parameter μ_1, or C_s in the case of LES coupling	0.0
Shock alpha 1	*REAL*: Viscosity parameter α_1	0.0
Shock mu 2	*REAL*: Viscosity parameter μ_2	μ_1
Shock alpha 2	*REAL*: Viscosity parameter α_2	α_1
Shock alpha/mu	*REAL*: Ratio between α and μ. It can be specified instead of α itself to make it dependent on the corresponding values of μ, and it is compulsory when using LES coupling	–
SVV filter cutoff	*REAL/CHARACTER(*)*: Cutoff of the filter kernel, P. If "automatic", its value is adjusted automatically	"automatic"
SVV filter shape	*CHARACTER(*)*: Options are: • Power • Sharp • Exponential	Power
SVV filter type	*CHARACTER(*)*: Options are: • Low-pass • High-pass	High-pass

Table 6.2: Keywords for shock-capturing algorithms in the Navier-Stokes equations – continuation.

Keyword	Description	Default value
Sensor variable	*CHARACTER(*)*: Variable used by the sensor to detect shocks. Options are: • rho • rhou • rhov • rhow • u • v • w • p • rhop • grad rho • div v	rhop
Sensor lower limit	*REAL*: Lower threshold of the central sensor region, s_1	**Mandatory keyword** (except GMM)
Sensor higher limit	*REAL*: Upper threshold of the central sensor region, s_2	**Mandatory keyword** (except GMM)
Sensor TE min N	*INTEGER*: Minimum polynomial order of the coarse mesh used for the truncation error estimation	1
Sensor TE delta N	Polynomial order difference between the solution mesh and its coarser representation	1
Sensor TE derivative	*CHARACTER**: Whether the face terms must be considered in the estimation of the truncation error or not. Options are: • Non-isolated • Isolated	Isolated
Sensor number of clusters	*INTEGER*: Maximum number of clusters to use with the GMM sensor	2
Sensor min. steps	*INTEGER*: Minimum number of time steps that an element will remain detected. The last positive value will be used if the sensor "undetects" an element too early	1

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

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

- high-pass: $\hat{F}^{\mathrm{H}}_{ijk} = \hat{F}^{\mathrm{1D}}_{i}\hat{F}^{\mathrm{1D}}_{j}\hat{F}^{\mathrm{1D}}_{k}$,
- low-pass: $\hat{F}^{\mathrm{L}}_{ijk} = 1 - \left(1 - \hat{F}^{\mathrm{1D}}_{i}\right)\left(1 - \hat{F}^{\mathrm{1D}}_{j}\right)\left(1 - \hat{F}^{\mathrm{1D}}_{k}\right)$,

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 μ_1 can be set to "Smagorinsky" to use the implemented SVV-LES approach. In this case, the $\mu = \mu_{\mathrm{LES}}$ viscosity is computed following a Smagorinsky formulation with $C_s = \mu_2$ and the viscosity parameters do not depend on the region anymore,

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

6.1.2 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 6.3, while the latter are defined in a block section, similar to the monitors (see Chapter 12):

```
#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 6.3.

Table 6.3: Keywords for acoustic analogy

Keyword	Description	Default value
acoustic analogy	*CHARACTER(*)*: This is the main keyword for activating the acoustic analogy. The only options is: "FWH".	–
acoustic analogy permeable	*LOGICAL*: Defines if uses a permeable or solid approach.	.FALSE.
acoustic solid surface	*CHARACTER(*)*: 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	*CHARACTER(*)*: Path to a fictitious surface that will be used for integration. It must be tailor-made for the mesh. Mandatory for using the permeable surface variant.	–
observers flush interval	*INTEGER*: Iteration interval to flush the observers information to the files.	100
acoustic solution save	*LOGICAL*: Defines whether it saves the NS solution on the surface. Mandatory for post-process computation.	.FALSE.
acoustic save timestep	*REAL*: 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	*CHARACTER(*)*: 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	*CHARACTER(*)*: Necessary for post-process calculation. Defines the type of post-process of horses.tools. For the FWH analogy the value must be "fwh post"	–

6.2 Incompressible Navier-Stokes

Among the various incompressible Navier-Stokes models, HORSES3D uses an artificial compressibility formulation, which converts the elliptic problem into a hyperbolic system of equations, at the expense of a non divergence–free velocity field. However, it allows one to avoid constructing an approximation that satisfies the inf–sup condition. This methodology is well suited for use as a fluid flow engine for interface–tracking multiphase flow models, as it allows the density to vary spatially.

The artificial compressibility system of equations is

$$\rho_t + \vec{\nabla} \cdot (\rho \vec{u}) = 0, \tag{6.1}$$

$$(\rho \vec{u})_t + \vec{\nabla} \cdot (\rho \vec{u}\vec{u}) = -\vec{\nabla} p + \vec{\nabla} \cdot \left(\frac{1}{\mathrm{Re}} \left(\vec{\nabla}\vec{u} + \vec{\nabla}\vec{u}^T \right) \right) + \frac{1}{\mathrm{Fr}^2} \rho \vec{e}_g, \tag{6.2}$$

$$p_t + \rho_0 c_0^2 \vec{\nabla} \cdot \vec{u} = 0, \tag{6.3}$$

The factor ρ_0 is computed as $max(\rho_1/\rho_1, \rho_2/\rho_1)$.

This solver is run with the binary horses3d.ins.

Table 6.4: Keywords for incompressible Navier-Stokes solver

Keyword	Description	Default value
reference velocity (m/s)	*REAL*: Reference value for velocity	–
number of fluids (1/2)	*INTEGER*: Number of fluids present in the simulation	1
maximum density (kg/mˆ3)	*REAL*: Maximum value used in the limiter of the density	Huge(1.0)
minimum density (kg/mˆ3)	*REAL*: Minimum value used in the limiter of the density	-Huge(1.0)
artificial compressibility factor	*REAL*: Artificial compressibility factor ${c_0}^2$.	–
gravity acceleration (m/sˆ2)	*REAL*: Value of gravity acceleration	–
gravity direction	*REAL*: Array containing direction of gravity. Eg. $[0.0, -1.0, 0.0]$.	–

The incompressible Navier Stokes solver has two modes: with 1 fluid and with 2 fluids. The required keywords are listed below.

Table 6.5: Keywords for incompressible Navier-Stokes solver. Mode with 1 fluid

Keyword	Description	Default value
density (kg/m^3)	*REAL*: Density of the fluid	–
viscosity (pa.s)	*REAL*: Viscosity of the fluid	–

Table 6.6: Keywords for incompressible Navier-Stokes solver. Mode with 2 fluids

Keyword	Description	Default value
fluid 1 density (kg/m^3)	*REAL*: Density of the fluid 1	–
fluid 1 viscosity (pa.s)	*REAL*: Viscosity of the fluid 1	–
fluid 2 density (kg/m^3)	*REAL*: Density of the fluid 2	–
fluid 2 viscosity (pa.s)	*REAL*: Viscosity of the fluid 2	–

6.3 Multiphase

The multiphase flow solver implemented in HORSES3D is constructed by a combination of the diffuse interface model of Cahn–Hilliard with the incompressible Navier–Stokes equations with variable density and artificial compressibility. This model is entropy stable and guarantees phase conservation with an accurate representation of surface tension effects. The modified entropy-stable version approximates:

$$c_t + \vec{\nabla} \cdot (c\vec{u}) = M_0 \vec{\nabla}^2 \mu, \tag{6.4}$$

$$\sqrt{\rho}\left(\sqrt{\rho}\vec{u}\right)_t + \vec{\nabla} \cdot \left(\frac{1}{2}\rho\vec{u}\vec{u}\right) + \frac{1}{2}\rho\vec{u} \cdot \vec{\nabla}\vec{u} + c\vec{\nabla}\mu = -\vec{\nabla}p + \vec{\nabla} \cdot \left(\eta\left(\vec{\nabla}\vec{u} + \vec{\nabla}\vec{u}^T\right)\right) + \rho\vec{g}, \tag{6.5}$$

$$p_t + \rho_0 c_0^2 \vec{\nabla} \cdot \vec{u} = 0, \tag{6.6}$$

where c is the phase field parameter, M_0 is the mobility, μ is the chemical potential, η is the viscosity and c_0 is the artificial speed of sound. The factor ρ_0 is computed as $max(\rho_1/\rho_1, \rho_2/\rho_1)$. Mobility M_0 is computed from the control file parameters chemical characteristic time t_{CH}, interface width ϵ and interface tension σ with the formula $M_0 = L_{ref}^2 \epsilon/(t_{CH}\sigma)$.

The term $M_0\vec{\nabla}^2\mu$ can be implicity integrated to reduce the stiffnes of the problem with the keyword time integration = IMEX. This is only recomended if the value of M_0 is very high so that the time step of the explicit scheme is very small.

This solver is run with the binary horses3d.mu.

Table 6.7: Keywords for multiphase Navier-Stokes solver

Keyword	Description	Default value
fluid 1 density (kg/m^3)	*REAL*: Density of the fluid 1	–
fluid 1 viscosity (pa.s)	*REAL*: Viscosity of the fluid 1	–
fluid 2 density (kg/m^3)	*REAL*: Density of the fluid 2	–
fluid 2 viscosity (pa.s)	*REAL*: Viscosity of the fluid 2	–
reference velocity (m/s)	*REAL*: Reference value for velocity	–
maximum density (kg/m^3)	*REAL*: Maximum value used in the limiter of the density	Huge(1.0)
minimum density (kg/m^3)	*REAL*: Minimum value used in the limiter of the density	-Huge(1.0)
artificial compressibility factor	*REAL*: Artificial compressibility factor ${c_0}^2$.	–
gravity acceleration (m/s^2)	*REAL*: Value of gravity acceleration	–
gravity direction	*REAL*: Array containing direction of gravity. Eg. $[0.0, -1.0, 0.0]$.	–
velocity direction	*REAL*: Array containing direction of velocity used for the outflow BC. Eg. $[1.0, 0.0, 0.0]$.	–
chemical characteristic time (s)	*REAL*: t_{CH} controls the speed of the phase separation	–
interface width (m)	*REAL*: ϵ controls the interface width between the phases	–
interface tension (N/m)	*REAL*: σ controls the interface tension between the phases	–

6.4 Particles

WARNING: Lagrangian particles are only implemented for the compressible Navier Stokes (horses.ns binary)

WARNING: Lagrangian particles do not support MPI.

HORSES3D includes a two-way coupled Lagrangian solver. Particles are tracked along their trajectories, according to the simplified particle equation of motion, where only contributions from Stokes drag and gravity are retained,

$$\frac{dy_i}{dt} = u_i, \quad \frac{du_i}{dt} = \frac{v_i - u_i}{\tau_p} + g_i, \tag{6.7}$$

where u_i and y_i are the *ith* components of velocity and position of the particle, respectively. Furthermore, v_i accounts for the continuous velocity of the fluid at the position of the particle. We consider spherical Stokesian particles, so their mass and aerodynamic response time are $m_p = \rho_p \pi D_p^3/6$ and $\tau_p = \rho_p D_p^2/18\mu$, respectively, ρ_p being the particle density and D_p the particle diameter.

Each particle is considered to be subject to a radiative heat flux I_o. The carrier phase is transparent to radiation, whereas the incident radiative flux on each particle is completely absorbed. Because we focus on relatively small volume fractions, the fluid-particle medium is considered to be optically thin. Under these hypotheses, the direction of the radiation is inconsequential, and each particle receives the same radiative heat flux, and its temperature T_p is governed by

$$\frac{d}{dt}(m_p c_{V,p} T_p) = \frac{\pi D_p^2}{4} I_o - \pi D_p^2 h (T_p - T), \tag{6.8}$$

where $c_{V,p}$ is the specific heat of the particle, which is assumed to be constant with respect to temperature. T_p is the particle temperature and h is the convective heat transfer coefficient, which for a Stokesian particle can be calculated from the Nusselt number $Nu = hD_p/k = 2$.

In practical simulations, integrating the trajectory of every particle is too expensive. Therefore, particles are agglomerated into parcels, each of them accounting for many particles with the same physical properties, position, velocity, and temperature. The evolution of the parcels is tracked with the same set of equations presented for the particles.

The two-way coupling means that fluid flow is modified because of the presence of particles. Therefore, the Navier-Stokes equations are enriched with the following source terms:

$$\boldsymbol{S} = \beta \begin{bmatrix} 0 \\ \sum_{n=1}^{N_p} \frac{m_p}{\tau_p}(u_{1,n} - v_1)\delta(\mathbf{x} - \mathbf{y}_n) \\ \sum_{n=1}^{N_p} \frac{m_p}{\tau_p}(u_{2,n} - v_2)\delta(\mathbf{x} - \mathbf{y}_n) \\ \sum_{n=1}^{N_p} \frac{m_p}{\tau_p}(u_{3,n} - v_3)\delta(\mathbf{x} - \mathbf{y}_n) \\ \sum_{n=1}^{N_p} \pi D_p^2 h (T_{p,n} - T)\delta(\mathbf{x} - \mathbf{y}_n) \end{bmatrix}, \tag{6.9}$$

where δ is the Dirac delta function, N_p is the number of parcels, β is the number of particles per parcel and $u_{i,n}$, $\mathbf{y}_{i,n}$, $T_{p,n}$ are the velocity, spatial coordinates, and temperature of the parcel *nth*. The dimensionless form of the Navier Stokes equations can be seen in the appendix at the end of this document.

Particles are solved in a box domain inside the flow domain. The box is defined with the keywords "minimum box" and "maximum box". The boundary conditions for the particles are defined with the keyword "bc box". Possible options are inflow/outflow, periodic and wall (with perfect rebound).

Table 6.8: Keywords for particles Navier-Stokes solver

Keyword	Description	Default value
lagrangian particles	*LOGICAL*: If .true. activates particles	.false.
stokes number	*REAL*: Stokes number which for Stokesian particles is $St = \frac{\rho_p D_p^2 u_o}{18 L_o \mu_o}$	–
Gamma	*REAL*: Ratio between specific heat of particles and fluid $\Gamma = c_{v,p}/c_v$	–
phi_m	*REAL*: Ratio between total mass of particles and fluid $\phi_m = \frac{m_p N_p}{\rho_o L_o^3}$	–
Radiation source	*REAL*: Non dimensional radiation source intensity $I_o^* = \frac{I_o D_p}{4 k_o T_o}$	–
Froude number	*REAL*: Froude number $Fr = \frac{u_o}{\sqrt{g_o L_o}}$	–
high order particles source term	*LOGICAL*: source term with high order dirac delta or averaged in the whole element	.false.
number of particles	*INTEGER*: Total number of parcels in the simulation	–
particles per parcel	*REAL*: β particles per parcel	–

Table 6.9: Keywords for particles Navier-Stokes solver - continued

Keyword	Description	Default value
Gravity direction	*INTEGER*: Array with direction of gravity. Only required if Fr number is specified.[0,0,-1]	–
particles file	*CHARACTER(*)*: Path to file with initial position of the particles. If not provided, initialization without particles inside the domain. Not compatible with injecting particles in the domain (see keyword below).	–
vel and temp from file	*LOGICAL*: If .true. Initial velocity and temperature of particles read from file. If .false., initial velocity and temperature of particles are the same as flow at the position of the particle.	–
injection	*LOGICAL*: If .true. injection of particles through a face of the box.	–
particles injection	*INTEGER*: Array with a vector indicating the direction of the injection. Eg., [0,1,0]	–
particles per step	*INTEGER*: Number of particles injected per time step.	–
particles iter period	*INTEGER*: Iteration period for particles injection. Set to 1 to inject particles every time step.	–
particles injection velocity	*REAL*: Array with particles injection non dimensional velocity. Eg., [0.d0,1.d0,0.d0]	–
particles injection temperature	*REAL*: Particles injection non-dimensional temperature	–
minimum box	*REAL*: Array with minimum x,y,z coordinates of box with particles. Eg., [0.d0,0.d0,0.d0]	–
maximum box	*REAL*: Array with maximum x,y,z coordinates of box with particles. Eg., [4.d-2,1.6d-1,4.d-2]	–
bc box	*INTEGER*: Array with boundary conditions for particles box in the form [yz,xz,xy]. 0 is inflow/outflow, 1 is wall, 2 is periodic. Eg., [2,0,2]. In this example planes yz and xy are periodic, while plane xz is inflow/outflow for particles.	–

6.5 Complementary Modes

6.5.1 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 6.10 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.

Table 6.10: Keywords for Wall Function

Keyword	Description	Default value
Wall Function	*CHARACTER(*)*: This is the main keyword for activating the wall function. Identifies the wall law to be used. Options are: • Standard: uses the Reichardt law. • ABL: uses the atmospheric boundary layer law.	–
Wall Function Boundaries	*CHARACTER(*)*: 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	*REAL*: von Karman constant	0.38
Wall Function C	*REAL*: Log law 'C' constant	4.1
Wall Function Seed	*REAL*: Initial value for the newton method	1.0
Wall Function Damp	*REAL*: Initial value damp for the newton method	1.0
Wall Function Tolerance	*REAL*: Tolerance for the newton method	10^{-10}
Wall Function max iter	*INTEGER*: Maximum number of iterations for the newton method	100
Wall Roughness	*REAL*: Aerodynamic roughness for the ABL wall function. Mandatory value for the ABL law.	–
Wall Plane Displacement	*REAL*: Plane displacement due to roughness for the ABL wall function	0.0
Wall Function Use Average	*LOGICAL*: Use the time average of the velocity in the wall function, each time step the time average is recalculated.	.FALSE.

6.5.2 Tripping

A numerical source term is added to the momentum equations to replicate the effect of a tripping mechanism used commonly in explerimental 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 6.11.

Table 6.11: Keywords for Tripping model

Keyword	Description	Default value
use trip	*LOGICAL*: This is the main keyword for activating the trip	.FALSE.
trip time scale	*REAL*: Time interval between the change of the time dependent part of the trip.	**Mandatory**
trip number of modes	*INTEGER*: Number of Fourier modes in the spanwise direction of the trip.	**Mandatory**
trip z points	*INTEGER*: 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.	**Mandatory**
trip attenuation	*REAL ARRAY(2)*: Length scale of the gaussian attenuation of the trip, the first position is the streamwise direction and the second is the wall-normal direction.	**Mandatory**
trip zone	*CHARACTER(*) ARRAY(:)*: Boundary condition name that constrains at least one surface where the trip center is located. It can be ither 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).	**Mandatory**
trip center	*REAL*: Position of the origin of the trip in the streamwise direction.	**Mandatory**
trip center 2	*REAL*: Position of the origin of the second trip, if used, in the streamwise direction.	–
trip amplitude	*REAL*: Maximum time varying amplitude of the trip.	1.0
trip amplitude steady	*REAL*: Maximum steady amplitude of the trip.	0.0
random seed 1	*INTEGER*: 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	*INTEGER*: 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 7

Implicit Solvers with Newton linearisation

7.1 General Keywords

The keywords for the implicit solvers are listed in table 7.1

Table 7.1: Keywords for implicit solvers.

Keyword	Description	Default value
time integration	*CHARACTER*: 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	*CHARACTER*: Specifies the linear solver that has to be used. Options are: • '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). • 'pardiso': Intel MKL PARDISO. Only available in serial or with OpenMP. • '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) • 'smooth': Traditional iterative methods. One can select either 'smoother=WeightedJacobi' or 'smoother=BlockJacobi'. • 'matrix-free smooth': A matrix-free version of the previous solver. Only available with 'smoother=BlockJacobi'.	'petsc'

7.2 Keywords for the BDF Methods

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

Table 7.2: Keywords for the BDF solvers.

Keyword	Description	Default value
bdf order	*INTEGER*: 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	*LOGICAL*: When .TRUE., the Jacobian is only computed when the convergence falls beneath a threshold (hard-coded). This improves performance.	.FALSE.
compute jacobian every	*INTEGER*: Forces the Jacobian to be computed in an interval of iterations that is specified.	Inf
print newton info	*LOGICAL*: If .TRUE., the information of the Newton iterations will be displayed.	'.FALSE.'
implicit adaptive dt	*LOGICAL*: Specifies if the time-step should be computed according to the convergence behavior of the Newton iterative method and the linear solver.	.FALSE.
newton tolerance	*REAL*: 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 *newton factor*
newton max iter	*INTEGER*: Maximum number of Newton iterations for BDF solver.	30
linsolver max iter	*INTEGER*: Maximum number of iterations to be taken by the linear solver. This keyword only affects iterative linear solvers.	500
newton factor	*REAL*: 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	*REAL*: 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	*REAL*: 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 *linsolver tol factor*).	0.2

7.3 Keywords for the Rosenbrock-Type Implicit Runge-Kutta Methods

Table 7.3: Keywords for the Rosenbrock schemes.

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

7.4 Jacobian Specifications

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

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

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

Table 7.4: Keywords for Jacobian definition block.

Keyword	Description	Default value
type	*INTEGER*: Specifies the type of Jacobian matrix to be computed. Options are: 1. Numerical Jacobian: Uses a coloring algorithm and a finite difference method to compute the DG Jacobian matrix (only available with shared memory parallelization). 2. Analytical Jacobian: Available with shared (OpenMP) or distributed (MPI) memory parallelization for advective and/or diffusive nonlinear conservation laws, **BUT** only for the standard DGSEM (no split-form).	**Mandatory Keyword**
print info	*LOGICAL*: Specifies the verbosity of the Jacobian subroutines	.TRUE.
preallocate	*LOGICAL*: 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 8

Explicit Solvers

Explicit time integration schemes available in HORSES3D. The main keywords to use it are shown in Table 8.1.

Table 8.1: Keywords for the multigrid solver.

Keyword	Description	Default value
time integration	*CHARACTER*: 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'
simulation type	*CHARACTER*: 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	*CHARACTER*: Select desired Runge-Kutta solver. Options are: 'Euler', 'RK3', 'RK5', 'RKOpt', 'SSPRK33' and 'SSPRK43'.	RK3
rk order	*INTEGER*: Order of Runge-Kutta method optimized for steady-state solver ('RKOpt'). Possible orders are from 2 to 7.	2
limit timestep	*LOGICAL*: Activate the possitivity limiter of Zhang and Shu (only for SSPRK methods).	.false.
limiter minimum	*REAL*: Minimum value of density and pressure allowed by the limiter.	1e-13

Chapter 9

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 9.1.

Table 9.1: Keywords for the multigrid solver.

Keyword	Description	Default value
time integration	*CHARACTER*: 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'
simulation type	*CHARACTER*: 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	*INTEGER*: Number of multigrid levels for the computations.	**Mandatory keyword**
delta n	*INTEGER*: Interval of reduction of polynomial order for creating coarser multigrid levels.	1
multigrid output	*LOGICAL*: If .TRUE., the residuals at the different multigrid levels will be displayed.	.FALSE.
mg sweeps	*INTEGER*: Number of smoothing sweeps to be taken.	1*
mg sweeps pre	*INTEGER*: Number of pre-smoothing sweeps to be taken.	1*
mg sweeps post	*INTEGER*: Number of post-smoothing sweeps to be taken.	1*
mg sweeps coarsest	*INTEGER*: 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	*INTEGER(:)*: 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	*INTEGER(:)*: 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	*INTEGER(:)*: 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 9.1: Keywords for the multigrid solver - continued.

Keyword	Description	Default value
mg smoother	*CHARACTER*: The smoothing technique to be used. The keywords and possible explicit smoothers are the same as the 'explicit method' in 8.1. For the semi-implicit residual relaxation use 'BIRK5'.	RK3
fasfmg residual	*REAL*: 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	*LOGICAL*: Save the solutions that are obtained at the different FMG levels. Only usable when *fasfmg residual* is used.	.FALSE.
postsmooth option	*CHARACTER*: 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: • *f-cycle*: Do the extra post-smoothing with an FMG cycle. • *smooth*: Do normal smoothing.	–
smooth fine	*REAL*: Extra pre-smoothing is performed on a multigrid level of order P, until a residual is obtained $\left\|\tilde{\mathfrak{R}}^P\right\|_\infty < \eta \left\|\tilde{\mathfrak{R}}^N\right\|_\infty$, where N is the polynomial order of the next (coarsest) grid, and η is the specified value.	–
max mg sweeps	*INTEGER*: Maximum number of smoothing sweeps to be performed. This only makes sense if one uses the keywords *postsmooth option* and/or *smooth fine*.	10000
mg initialization	*LOGICAL*: Sets the initial explicit residual smoothing with RK3 and local time stepping.	.FALSE.
initial residual	*REAL*: Threshold for the $\left\|\tilde{\mathfrak{R}}^P\right\|_\infty$ after which solver switches from the 'mg initialization' settings to user specified.	1.0
initial cfl	*REAL*: 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 10

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 10.1: Keywords for the p-adaptation algorithms.

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

Table 10.1: Keywords for the p-adaptation algorithms - continued.

Keyword	Description	Default value
write error files	*LOGICAL*: If .TRUE., the program writes a file per element containing the directional tau-estimations. The files are stored in the folder *./TauEstimation/*. When the simulation has several adaptation stages, the new information is just appended.	.FALSE.
adjust nz	*LOGICAL*: If .TRUE., the order across faces is adjusted i n the directions xi, eta, and zeta of the face (being zeta the normal direction). If .FALSE., the order is only adjusted in the xi and eta directions. The adjustment currently consists (hard-cod ed) in allowing jumps in the polynomial order of at most 1.	.FALSE.
mode	*CHARACTER*: p-Adaptation mode. Can be *static*, *time* or *iteration*. Static p-adaptation is performed once at the beginning of a simulation for steady or unsteady simulations. Unsteady adaptation can be by *time* or by *iteration*.	*static*
interval	*INTEGER/REAL*: In dynamic p-adaptation cases, this keyword specifies the iteration (integer) or time (real) interval for p-adaptation.	*huge number*
restart files	*LOGICAL*: If .TRUE., the program writes restart files before and after the p-adaptation.	.FALSE.
max N decrease	*INTEGER*: Maximum decrease in the polynomial order in every p-adaptation procedure.	$N - N_{min}$
post smoothing residual	*REAL*: Specifies the maximum allowable deviation of $\partial_t q$ after the p-adaptation procedure.	–
post smoothing method	*CHARACTER*: Either RK3 or FAS.	RK3, if the last keyword is activated
estimation files	*CHARACTER*: Name of the folder that contains the error estimations obtained with the multi tau-estimation (section 10.1).	–
estimation files number	*INTEGER(2)*: First and last estimation stages to be used for p-adaptation.	Mandatory if last keyword is used.
padapted $\ll$ *keyword* $\gg$	*MULTIPLE*: 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.	–

10.1 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 11

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 11.1: Keywords for the immersed boundary method.

Keyword	Description	Default value
name	*CHARACTER*: name assigned to immersed boundary method job	
active	*LOGICAL*: When .TRUE. the immersed boundary method is active.	.FALSE.
penalization	*REAL*: Specifies the value of the penalization term, *i.e.* η.	Δt
semi implicit	*LOGICAL*: The source term is treated in a semi-implicit manner.	.FALSE.
number of objects	*INTEGER*: Specifies the maximum number of objects inside a leaf of the KD-tree.	5
number of interpolation points	*INTEGER*: 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	*LOGICAL*: If it's true, the band region is computed, otherwise it is not.	.FALSE.
band region coefficient	*INTEGER*: 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 11.1: Keywords for the immersed boundary method - continued.

Keyword	Description	Default value
compute distance	*LOGICAL*: 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	*INTEGER*: 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 *x-axis*, 2 with *y-axis* and 3 with *z-axis*	0
aab	*LOGICAL*: The Axis Aligned Box is computed instead of the Oriented Bounding box. It is recommended when '*clip axis*' $\neq 0$.	.FALSE.
describe	*LOGICAL*: The immersed boundary parameters are printed on the screen.	.FALSE.
plot obb	*LOGICAL*: The oriented-bounding box is plotted.	.FALSE.
plot kdtree	*LOGICAL*: The kd-tree is plotted.	.FALSE.
plot mask	*LOGICAL*: The degrees of freedom belonging to the mask are plotted.	.FALSE.
plot band points	*LOGICAL*: The band region's points are plotted.	.FALSE.

11.0.1 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 11.2: Keywords for stl files.

Keyword	Description	Default value
number of stl =	*INTEGER*: Number of stl files.	1
stl file name =	*CHARACTER*: The name of the STL file, without extension; it has to be inside the folder "MESH" .	**Mandatory keyword**
stl file nameN =	*CHARACTER*: The name of the N^{th}-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

11.0.2 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.

11.0.3 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 11.3: Keywords for a moving stl.

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

Chapter 12

Monitors

The monitors are specified individually as blocks in the control file. The only general keyword that can be specified is explained in Table 12.1.

Table 12.1: Keywords for monitors.

Keyword	Description	Default value
monitors flush interval	*INTEGER*: Iteration interval to flush the monitor information to the monitor files.	100

12.1 Residual Monitors

12.2 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
- uu
- uv
- v
- vv
- uw
- w
- ww
- vw

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

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

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.

12.3 Probes

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

Table 12.2: Keywords for probes.

Keyword	Description	Default value
name	*CHARACTER*: Name of the monitor.	**Mandatory Keyword**
variable	*CHARACTER*: Variable to be monitored. Implemented options are: • pressure • v • k • velocity • w • u • mach	**Mandatory Keyword**
position	*REAL(3)*: Coordinates of the point to be monitored.	**Mandatory Keyword**

12.4 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 12.3: Keywords for probes.

Keyword	Description	Default value
name	*CHARACTER*: Name of the monitor.	**Mandatory Keyword**
marker	*CHARACTER*: Name of the boundary where a variable will be monitored.	**Mandatory Keyword**
variable	*CHARACTER*: Variable to be monitored. Implemented options are: • mass-flow • force • flow • lift • pressure-force • drag • viscous-force • pressure-average	**Mandatory Keyword**
reference surface	*REAL*: Reference surface [area] for the monitor. Needed for "lift" and "drag" computations.	–
direction	*REAL(3)*: 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])	–

12.5 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 12.4: Keywords for volume monitors.

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

Chapter 13

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 13.1.

13.1 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.

13.2 Compiling the Problem File

The Problem File 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 were 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/HPC/**RELEASE**
- COMPILER=ifort/**gfortran**
- COMM=PARALLEL/**SEQUENTIAL**
- ENABLE_THREADS=NO/**YES**

Chapter 14

Postprocessing

For postprocessing the Simulation Results

14.1 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. It can also export the solution to the more recent VTKHDF format; however, note that this feature does **not** export boundary information or mesh files. Usage:

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

The options comprise following flags:

Table 14.1: Flags for *horses2plt*.

Flag	Description	Default value
--output-order=	*INTEGER*: Output order nodes. The solution is interpolated into the desired number of points.	Not Present
--output-basis=	*CHARACTER*: Either *Homogeneous* (for equispaced nodes, or *Gauss*.	*Gauss**
--output-mode=	*CHARACTER*: Either *multizone* or *FE*. The option *multizone* generates a Tecplot zone for each element. The option *FE* 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.	*multizone*
--output-variables=	*CHARACTER*: Output variables separated by commas.A complete description can be found in Section 14.1.1.	Q
--dimensionless	Specifies that the output quantities must be dimensionless	Not Present
--partition-file=	*CHARACTER*: Specifies the path to the partition file (*.pmesh) to export the MPI ranks of the simulation.	Not Present
--boundary-file=	*CHARACTER*: Specifies the path to the boundary mesh file (*.bmesh) to export the surfaces as additional zones of the Tecplot file.	Not Present
--output-type=	*CHARACTER*: Specifies the type of output file: *tecplot* or *vtkhdf*.	*tecplot*

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

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

14.1.1 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
- $rhou$
- $rhov$
- $rhow$
- $rhoe$
- c
- Nxi
- $Neta$
- $Nzeta$
- Nav
- N
- Ax_Xi
- Ax_Eta
- Ax_Zeta
- $ThreeAxes$
- $Axes$
- mpi_rank
- eID
- $gradV$
- u_x
- v_x
- w_x
- u_y
- v_y
- w_y
- u_z
- v_z
- w_z
- c_x
- c_y
- c_z
- $omega$
- $omega\_x$
- $omega\_y$
- $omega\_z$
- $omega\_abs$
- $Qcrit$

14.1.2 Statistics Files (*.stats.hsol)

Statistics files generate following variables by default (being Sij the components of the Reynolds Stress tensor):

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

14.2 Extract geometry

Under construction.

14.3 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.

14.4 Mapping result to different mesh

HORSES3D addons, *horsesConverter*, has a capability to map result into different mesh file, with both have a consistent geometry. This is done by performing interpolation with the polynomial inside each element for each node point of the new mesh. The type of node quadrature will follow the quadrature defined in the .hmesh file with selected polynomial order in the control file. A control input file is required and must has name *horsesConverter.convert*. The template of control input file will be generated by default when executing *./horsesConverter* in a directory without the control file. Error message is given when at least one node point of the new mesh is not within any element of the old mesh. After completion, a new result file is generated and named *Result_interpolation.hsol*. The required keywords in the control file are described in table 14.2. Command to execute:

```
$ ./horsesConverter
```

Table 14.2: Keywords for *meshInterpolation*.

Keyword	Description	Default value
Task	*meshInterpolation*	
Mesh Filename 1	Location of the origin mesh (*.hmesh)	
Boundary Filename 1	Location of the origin boundary mesh (*.bmesh)	
Result 1	Location of the solution file with origin mesh (*.hsol)	
Mesh Filename 2	Location of the target mesh (*.hmesh)	
Boundary Filename 2	Location of the target boundary mesh (*.bmesh)	
Polynomial Order	Polynomial order of the target mesh	(1, 1, 1)

14.5 Generate OpenFOAM mesh

Another functionality of *horsesConverter* addons is to convert the mesh files, (*.hmesh) and (*.bmesh), into OpenFOAM format, the *polyMesh* folder. Each element is discretised into $n_x \times n_y \times n_z$ cells distributed as Gauss-Lobatto nodes. The number of division of each element, (n_x, n_y, and n_z), is required in the control file, see section 14.4. After completion, a folder named *foamFiles* is generated. OpenFOAM mesh files, i.e. *points*, *faces*, *owner*, *neighbour*, and *boundary*, are located within *foamFiles/constant/polyMesh*. The required keywords in the control file are described in table 14.3. Command to execute:

```
$ ./horsesConverter
```

Table 14.3: Keywords for *horsesMesh2OF*.

Keyword	Description	Default value
Task	*horsesMesh2OF*	
Mesh Filename 1	Location of the origin mesh (*.hmesh)	
Boundary Filename 1	Location of the origin boundary mesh (*.bmesh)	
Polynomial Order	Number of division of each element (n_x, n_y, and n_z)	(1, 1, 1)

NOTE: Before running the mesh in the OpenFOAM environment, the type of boundaries inside the boundary file needs to be adjusted according to the actual type (*patch*, *wall*, and *symmetry*).

14.6 Generate HORSES3D solution file from OpenFOAM result

HORSES3D provides a capability to convert OpenFOAM result into HORSES3D solution file (*.hsol). The mesh of the OpenFOAM result must be generated by converting HORSES3D mesh files, see section 14.5. Beforehand, the OpenFOAM result must be converted into VTK format(*.vtk). This not only allows the result to be in the single file but also converts cell data into point data. In the OpenFOAM environment, the command for this conversion:

```
$ foamToVTK -fields "(U␣p␣T␣rho)" -ascii -latestTime
```

The necessary file (.vtk) required for the control file input is inside VTK folder, see section 14.4 for the control file template. The HORSES3D solution file is named *Result_OF.hsol*. The required keywords in the control file are described in table 14.4. Command to execute:

```
$ ./horsesConverter
```

Table 14.4: Keywords for *OF2Horses*.

Keyword	Description	Default value
Task	*OF2Horses*	
Mesh Filename 1	Location of the origin mesh (*.hmesh)	
Boundary Filename 1	Location of the origin boundary mesh (*.bmesh)	
Polynomial Order	Polynomial order of the solution file (.hsol)	(1, 1, 1)
VTK file	Location of VTK file (.vtk)	
Reynolds Number	Reynolds Number/m of the solution – L_{ref}=1.0m	

Table 14.4: Keywords for *OF2Horses* - continued.

Keyword	**Description**	**Default value**
Mach Number	Mach Number of the solution	
Reference pressure (Pa)	Reference Pressure	101325
Reference temperature (K)	Reference Temperature	288.889

Appendices

Non-dimensional Navier Stokes equations

To illustrate the roles of various terms in the governing equations, we present here the non-dimensionalized governing compressible Navier Stokes equations with particles. We define the non-dimensional variables as

$$x_i^* = x_i/L_o, \quad y_i^* = y_i/L_o, \quad t^* = tu_o/L_o, \quad \rho^* = \rho/\rho_o, \quad u_i^* = u_i/u_o \tag{14.1}$$

$$v_i^* = v_i/u_o, \quad p^* = p/\rho_o u_o^2, \quad e^* = e/u_o^2 \quad T^* = T/T_o, \quad T_p^* = T_p/T_o, \tag{14.2}$$

$$\mu^* = \mu/\mu_o, \quad \kappa^* = \kappa/\kappa_o \quad g_i^* = g_i/g_o \tag{14.3}$$

where the subscript o denotes a reference value. Under these scalings, the Navier Stokes equations become

$$\frac{\partial \rho^*}{\partial t^*} + \frac{\partial \rho^* u_j^*}{\partial x_j^*} = 0, \tag{14.4}$$

$$\frac{\partial \rho^* u_i^*}{\partial t^*} + \frac{\partial \rho^* u_i^* u_j^*}{\partial x_j^*} = -\frac{\partial p^*}{\partial x_i^*} + \frac{1}{Re}\frac{\partial \tau_{ij}^*}{\partial x_j^*} + \frac{1}{Fr^2} g_i^* - \beta \frac{\mu^* \phi_m}{N_p St} \sum_{n=1}^{N_p} (u_i^* - v_{i,n}^*)\delta(\mathbf{x}^* - \mathbf{y}_n^*), \tag{14.5}$$

$$\begin{aligned} \frac{\partial \rho^* e^*}{\partial t^*} + \frac{\partial u_j^*(\rho^* e^* + p^*)}{\partial x_j^*} =& \frac{1}{Re}\left[\frac{\partial \tau_{ij}^* u_j^*}{\partial x_i^*} + \frac{1}{(\gamma-1)PrM_o^2}\frac{\partial}{\partial x_j^*}\left(k^*\frac{\partial T^*}{\partial x_j^*}\right)\right] + \\ &+ \beta \frac{\phi_m}{3N_p}\frac{Nu}{(\gamma-1)PrM_o^2 St}\sum_{n=1}^{N_p}(T_{p,n}^* - T^*)\delta(\mathbf{x}^* - \mathbf{y}_n^*). \end{aligned} \tag{14.6}$$

Where $Re = \frac{\rho_o L_o u_o}{\mu_o}$ is the Reynolds number, $Fr = \frac{u_o}{\sqrt{g_o L_o}}$ is the Froude number, $\phi_m = \frac{m_p N_p}{\rho_o L_o^3}$, $St = \frac{\tau_p}{\tau_f}$ is the Stokes number which for Stokesian particles is $St = \frac{\rho_p D_p^2 u_o}{18 L_o \mu_o}$, the Prandtl number (which is assumed constant and equal to 0.72) $Pr = \frac{\mu_o c_p}{k_o}$, the Nusselt number $Nu = \frac{hD_p}{k_o} = 2$ (for Stokesian particles) and the Mach number $M_o = \frac{u_o}{\sqrt{\gamma R T_o}}$.

Non-dimensional particle equations

The non dimensional set of equations for the particles reads:

$$\frac{dy_i^*}{dt^*} = v_i^*, \quad \frac{dv_i^*}{dt^*} = \frac{\mu^*}{St}(u_i^* - v_i^*) + \frac{1}{Fr^2} g_i^*. \tag{14.7}$$

$$\frac{dT_p^*}{dt^*} = \frac{\gamma}{3\Gamma StPr}\left(I_o^* - Nu(T_p^* - T^*)\right), \tag{14.8}$$

where $I_o^* = \frac{I_o D_p}{4 k_o T_o}$ and $\Gamma = c_{v,p}/c_v$ is the ratio of the particle specific heat capacity to the fluid isochoric specific heat capacity.