HORSES3D

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

User Manual

Andrés Rueda
Many others (future?)

February 24, 2022

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/**RELEASE**
  - COMPILER=ifort/**gfortran**
  - COMM=PARALLEL/**SEQUENTIAL**
  - PLATFORM=MACOSX/**LINUX**
  - ENABLE_THREADS=NO/**YES**

  For example:

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

- 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.

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 worflow using GMSH.

- Curved geometry supported up to polynomial order 5.
- 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 NSLITE3D 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**
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

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 minimun 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 condicion 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 (adiabatica 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 zones 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	*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 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.	–

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	*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: • Smagorinsky • None	None
Wall model	*CHARACTER(*)*:	linear

5.1.1 Spectral Vanishing Viscosity

WARNING: The functionality explained in this section is still experimental and may change in future iterations.
WARNING: This method does not support MPI parallelisation yet.

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

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 $\mu = \mu_{\text{LES}}$ viscosity is computed following a Smagorinsky formulation with $C_s = \mu_2$ and the viscosity parameters do not depend on the region anymore,

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

Table 5.2: Keywords for SVV stabilisation (Navier-Stokes solver only).

Keyword	Description	Default value
Enable SVV	*LOGICAL*:	.FALSE.
SVV viscosity 1	*REAL/CHARACTER(*)*: Viscosity parameter μ_1 or "Smagorinsky" for the SVV-LES approach	0.0
SVV alpha viscosity 1	*REAL*: Viscosity parameter α_1	0.0
SVV viscosity 2	*REAL*: Viscosity parameter μ_2, or C_s in the case of SVV-LES	$\mu_2 = 0.0$ $C_s = 0.2$
SVV alpha viscosity 2	*REAL*: Viscosity parameter α_2	0.0
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	**Mandatory keyword**
SVV filter type	*CHARACTER(*)*: Options are: • Low-pass • High-pass	**Mandatory keyword**
SVV dissipation type	*CHARACTER(*)*: Options are: • Physical • Guermond	Physical

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 5.3 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 5.3: 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 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

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 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 5.4.

Table 5.4: 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)*: Lenght 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 constains 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 initilialize 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 initilialize 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
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'

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	*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

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	*CHARACTER*: Rosenbrock scheme to be used. Currently, only the *RO6-6* 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	*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 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
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' and 'RKOpt'.	RK3
rk order	*INTEGER*: 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
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 8.1: 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 7.1. 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 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	*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.
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.

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

Keyword	Description	Default value
adjust nz	*LOGICAL*: If .TRUE., the order accross 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 9.1).	–
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.	–

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

Monitors

The monitors are specified individually as blocks in the control file. The only general keyword that can be specified is explained in Table 10.1.

Table 10.1: Keywords for monitors.

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

10.1 Residual Monitors

10.2 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.

10.3 Probes

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

Table 10.2: 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**

10.4 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 10.3: 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])	–

10.5 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 10.4: 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 11

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 11.1.

11.1 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.

11.2 Compiling the Problem File

The Problem Fie 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/**RELEASE**
- COMPILER=ifort/**gfortran**
- COMM=PARALLEL/**SEQUENTIAL**

- PLATFORM=MACOSX/**LINUX**
- ENABLE_THREADS=NO/**YES**

Chapter 12

Postprocessing

For postprocessing the Simulation Results

12.1 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 12.1: 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 12.1.1.	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

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

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

12.1.1 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)
- v
- T
- $Vabs$
- $rhou$
- rho
- w
- $Mach$
- V
- $rhov$
- u
- p
- s
- Ht
- $rhow$

- $rhoe$
- 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
- $omega$
- $omega\_x$
- $omega\_y$
- $omega\_z$
- $omega\_abs$
- $Qcrit$

12.1.2 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

12.2 Extract geometry

Under construction.

12.3 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.