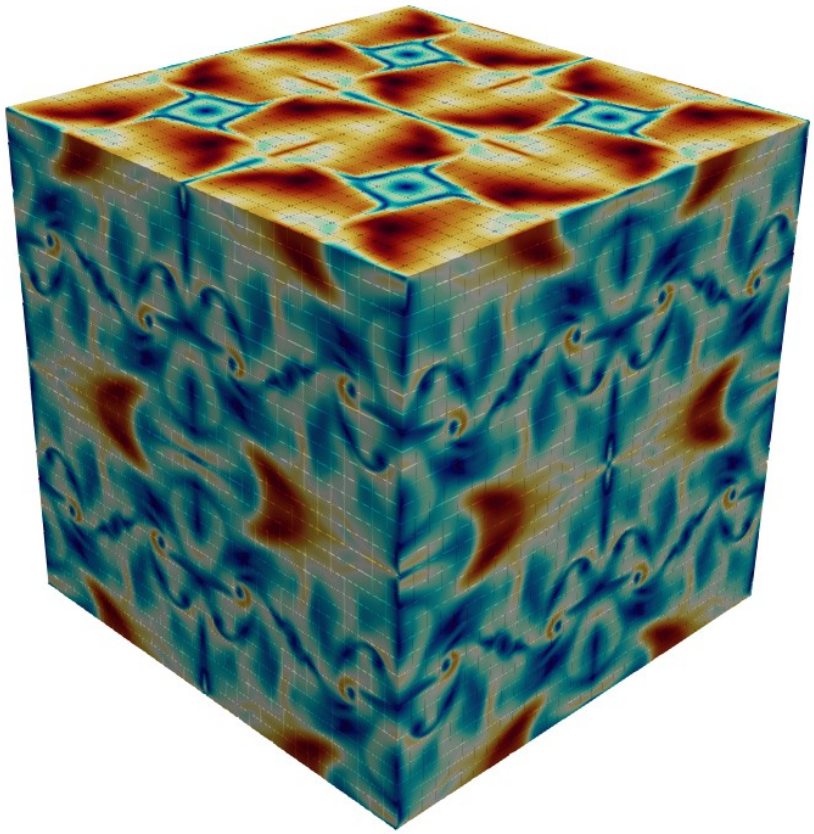
HORSES3D: Cylinder FlowS

A high-order discontinuous Galerkin solver for flow simulations and multi-physics applications



Universidad politécnica de madrid
Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio

1. Synopsis

HORSES3D is a multiphysics environment where the compressible Navier-Stokes equations, the incompressible Navier–Stokes equations, the Cahn–Hilliard equation and entropy–stable variants are solved. Arbitrary high–order, p–anisotropic discretisations are used, including static and dynamic p–adaptation methods (feature-based and truncation error-based). Explicit and implicit time-steppers for steady and time-marching solutions are available, including efficient multigrid and preconditioners. Numerical and analytical Jacobian computations with a coloring algorithm have been implemented. Multiphase flows are solved using a diffuse interface model: Navier–Stokes/Cahn–Hilliard. Turbulent models implemented include RANS: Spalart-Allmaras and LES: Smagorinsky, Wale, Vreman; including wall models. Immersed boundary methods can be used, to avoid creating body fitted meshes. Acoustic propagation can be computed using Ffowcs-Williams and Hawkings models.

HORSES3D supports curvilinear, hexahedral, conforming meshes in GMSH, HDF5 and SpecMesh/HOHQMesh format. A hybrid CPU-based parallelisation strategy (shared and distributed memory) with OpenMP and MPI is followed.

2. Installation

HORSES3D is an open-source code and it's available in GitHub: https://github.com/loganoz/horses3d

To download the code for a Unix-based OS, you can simply type in the terminal:

```
>> git clone https://github.com/loganoz/horses3d.git
```

As an alternative, you can download the zip file by clicking on the '<> Code' green button and then on 'Download ZIP'. To extract all directories, run:

```
>> unzip horses3d-master.zip
```

3. Compilation

HORSES3D is an object-oriented Fortran 2008 solver, that can be compiled using gcc and the Intel compiler, in Unix-based operating systems. We recommend using recent versions of such compilers (2019 or newer).

To compile the code, first go into the folder called *Solver*:

```
>> cd horses3d-master/Solver
```

Then, configure the project:

```
>> ./configure
```

Once the configuration is finished, you can build and compile the solver:

```
>> make clean

>> make all [options]
```

with the desired *options* (defaults are bold):

- PLATFORM=MACOSX/**LINUX**
- MODE=DEBUG/**RELEASE**
- COMPILER=ifort/**gfortran**
- COMM=PARALLEL/**SEQUENTIAL**
- ENABLE_THREADS=NO/**YES**
- WITH_PETSC=**NO/**YES
- WITH_METIS=**NO/**YES
- WITH_HDF5=**NO/**YES
- WITH_MKL=**NO/**YES

For example, to compile the code with the Intel compiler and MPI (parallel):

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

All flags that are nor specified will be compiled with the default values.

4. How to run a simulation?

All details on how to run a simulation can be found in the **User Manual**, which is located in: **/horses3d-master/doc/UserManual.pdf**

Also, there are some tutorials inside the folder: **/horses3d-master/tutorials**

However, here we show a simple example of a flow field around a cylinder at $\Re = 40$.

First, go to the folder with the test case of the cylinder:

```
>> cd tutorials/Cylinder
```

Inside that folder you will find:

- **Control file (Cylinder_Re40.control)**: Inside that file you can specify all the parameters required to run the simulation.
- **RESULTS folder:** The output of the simulation will be stored inside that folder. If the folder doesn't exist, it will not be automatically created during the simulation and it will crash.
- **SETUP folder**: Inside this folder, there is a file called **ProblemFile.f90**, which allows you to select the initial condition and special boundary conditions, and to perform other operations during the simulation.
- **MESH folder**: Folder with the mesh of the domain.

First, you need to compile the setup file. For now, let's keep the ProblemFile.f90 as it is by default (the ProblemFile.f90 is already configured for every test case):

```
>> cd SETUP

>> make [options]
```

```
>> cd ..
```

Note: When you run the *make* command, use the same options that were used to compile the whole code in the Solver file.

Then, you can open the *control file* and modify the simulation parameters. Let's take a look at the main options:

```
Flow equations       = "NS" !The set of equations to be solved (NavierStokes)
mesh file name       =  "MESH/cyl_circ.msh" !Path to the mesh
Polynomial order     = 3 !Polynomial order inside each element of the mesh
Number of time steps  = 10000 !Maximum number of time steps
Output Interval      = 100 !The residuals will be printed each 100 iterations
autosave mode        = "iteration" !Save the solution based on the nº of iterations
autosave interval    = 2500 !Save solution each 2500 iterations
Convergence tolerance = 1.d-10 !The simulation will stop if residuals are below tol
cfl                  = 0.4 !CFL number. You can specify the time step dt instead
dcfl                 = 0.4 !Diffusive CFL number
mach number          = 0.2
Reynolds number      = 40.0
AOA theta            = 0.0 !Polar angle of attack
AOA phi              = 00.0 !Azimuthal angle of attack
solution file name   = "RESULTS/Cylinder_Re40.hsol" !Name of the results file
save gradients with solution = .true. !Whether gradients are saved or not
restart              = .false. !Restart from previous solution
restart file name    = "RESULTS/Cylinder_Re40.hsol" !Restart solution
riemann solver       = roe !Riemann solver for inviscid fluxes
simulation type      = steady-state !Type of the simulation
time integration     = explicit !Time integration approach
explicit method      = RK3 !Time integration scheme (Runge-Kutta 3 in this case)
```

This is a small list. The whole set of parameters can be found in the **User Manual**: **/horses3d-master/doc/UserManual.pdf**

Inside the control file, the boundary conditions are also defined. The name of each surface must be defined in the mesh file.

```
#define boundary cylinder !Name of the boundary surface
  type = NoSlipWall !All velocities are 0: u=0, v=0, w=0
#end

#define boundary left__right
  type = FreeSlipWall !Perpendicular velocities are 0
#end

#define boundary inlet
  type = Inflow
#end

#define boundary outlet
  type = Outflow
#end
```

Finally, you can define monitors and probes to check the solution during the simulation.

```
#define surface monitor 1
   Name = cyl-drag !Identifier of the drag monitor (choose the name you want)
   Marker = cylinder !Name of the boundary surface
   Variable = drag !Variable to be monitored
   Direction = [1.0,0.0,0.0] !Direction of the force
   Reference surface = 1.0        !Size of the reference surface to compute the drag
#end
```

```
#define surface monitor 2
  Name = cyl-lift !Lift monitor
  Marker = cylinder
  Variable = lift
  Direction = [0.0,1.0,0.0]
  Reference surface = 1.0
#end


#define probe 1 !Definition of a probe (value of a variable in a specific point)
  Name = wake_u !Identifier of the wake_u probe
  Position = [2.0,0.0,0.5] !Where to compute the value of the variable
  Variable = u ¡Variable to be monitored
#end
```

Once you are satisfied with the parameters defined in the *control file*, you can run the simulation. The executable files of HORSES3D can be found inside the folder /horses3d-master/Solver/bin:

- **horses3d.ns**: To solve the Navier-Stokes equations.
- **horses3d.nssa**: To solve the RANS Navier-Stokes Spalart-Allmaras equations.
- **horses3d.ins**: To solve the Incompressible Navier-Stokes equations.
- **horses3d.ch**: To solve the Cahn-Hilliard equations.
- **horses3d.mu**: To solve multiphase flows with a Cahn–Hilliard model coupled with the incompressible Navier–Stokes equations.

For the cylinder test case, we need to solve the Navier-Stokes equations. Therefore, to start the simulation (from the Cylinder folder) type:

```
>> ../../Solver/bin/horses3d.ns Cylinder_Re40.control
```

To make things simple, a dynamic link of the executable file has been automatically created in the current folder. If it has been removed, you can create a new one with:

```
>> ln -s ../../Solver/bin/horses3d.ns
```

And then, start the simulation:

```
>> horses3d.ns Cylinder_Re40.control
```

Once you have simulated the cylinder at $\Re = 40$, you can use the control file for $\Re = 100$ included in the tutorial folder and repeat the process. Note that at $\Re = 40$ the flow is steady while at $\Re = 100$ it is unsteady. You can restart the unsteady simulation from the steady one.

5. Post-processing

The results of the simulation will be stored inside the RESULTS folder in *.hsol files, which are binary files generated by HORSES3D. To visualize those results, we need to convert the results to a readable format *.tec, which can be visualize with the free CFD visualization tool **Paraview** (https://www.paraview.org/download/) or **Tecplot**.

To convert the solution, run (from the Cylinder folder):

```
>> ../../Solver/bin/horses2plt solution_path.hsol mesh_path.hmesh [options]
```

The most important available options are:

- **--output-mode**: 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. **FE is faster** in Paraview and Tecplot.
- **--output-variables**: Output variables separated by commas (see full list in the **User Manual**).

A complete list of the options is in the section 14.1 of the **User Manual**.

For the cylinder test case, let's convert the solution to show the density and the three components of the velocity:

```
>> ../../Solver/bin/horses2plt RESULTS/final_solution.hsol
MESH/final_mesh.hmesh --output-mode=FE --output-variables=rho,V
```

Note: "final_solution" and "final_mesh" must be changed to the names of the files with the final solution and the final mesh (the name depends on the number of iterations).

Again, you can also create a dynamic link to the executable file horses2plt:

```
>> ln -s ../../Solver/bin/horses2plt

>> horses2plt RESULTS/final_solution.hsol MESH/final_mesh.hmesh --output-
mode=FE --output-variables=rho,V
```

The conversion will generate a *.tec file inside the RESULTS folder, that can be opened from Paraview.

6. Custom simulation

Based on the test cases available in HORSES3D, it's possible to create custom simulations. The first step is to create a new folder for your simulation and copy the same folder structure from one of the tutorials. Your resulting folder should have:

- My_custom_folder/
  - RESULTS/
  - MESH/
    - Custom_mesh.msh (other formats available)
  - SETUP/
    - ProblemFile.f90

        - Makefile
    - Custom_simulation.control

It is recommended to let the ProblemFile.f90 as it is. However, you will have to modify the subroutine **UserDefinedInitialCondition** in case your custom simulation and the reference test case require different initial conditions.

The control file Custom_simulation.control follows the same structure that the control file used for the test case. If you want to use additional options, take a look at the **User Manual.**

Finally, you will need a mesh to run your simulation. The available formats are *.mesh / *.h5 / *.msh. In case you don't have previous experience creating a mesh, a good starting point is the open-source application **Gmsh** (https://gmsh.info/#Download). It is a simple meshing tool with a lot of tutorials, and it is especially useful for 2D extruded meshes.

Note: HORSES3D requires a 3D mesh to run a simulation. In case you want to run a 2D problem, create a 2D mesh and extrude it one element in the z-direction.

6.1. Create a simple mesh with Gmsh

To create a mesh with Gmsh, you can use the GUI or you can create the mesh through a script. The latter is recommended as it gives you an accurate control and you don't have to perform every action manually each time you want to modify a parameter. The script files for Gmsh have a *.geo extension. Here is an example of the *.geo script for a 2D cylinder:

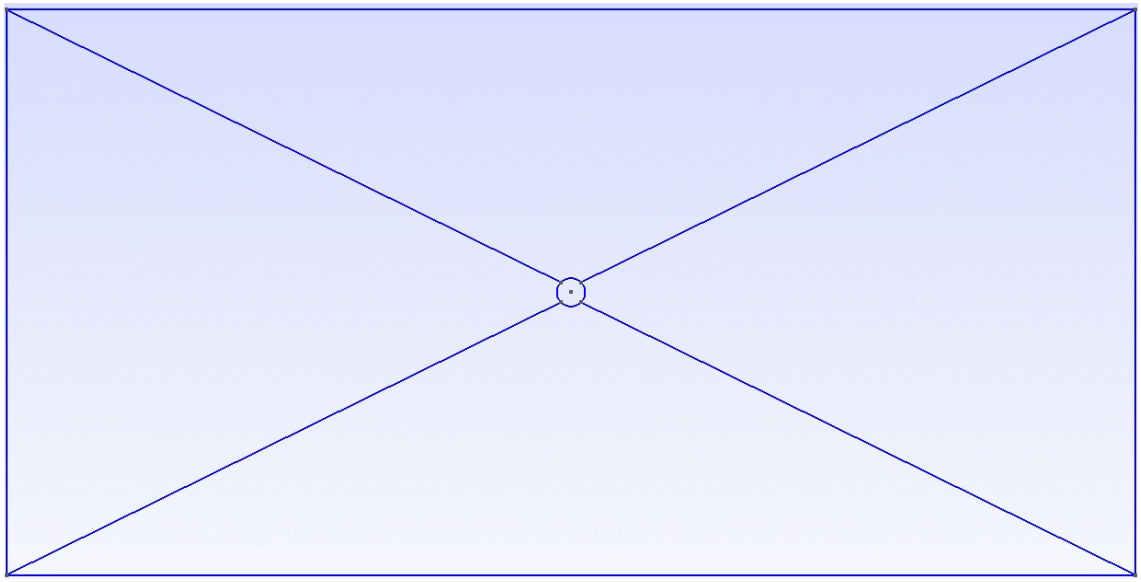
```
//Import the geometry module

SetFactory("OpenCASCADE");

//Define your domain

xmax =  20;

xmin = -20;

ymax =  10;

ymin = -10;

zmax =  1.0;

zmin =  0.0;

shift = 0.5;

R = 0.5;


//Define some points to divide the domain in regions with 4 sides

Point(1) = {xmin, ymin, zmin, 3.0}; //{x_pos, y_pos, z_pos, mesh_tolerance}

Point(2) = { xmax, ymin, zmin, 3.0};

Point(3) = { xmax,  ymax, zmin, 3.0};
```

```
Point(4) = {xmin,  ymax, zmin, 3.0};
Point(5) = {-Cos(45*Pi/180)*R+shift,  Sin(45*Pi/180)*R+shift, zmin, 0.01};
Point(6) = {-Cos(45*Pi/180)*R+shift, -Sin(45*Pi/180)*R+shift, zmin, 0.01};
Point(7) = { Cos(45*Pi/180)*R+shift, -Sin(45*Pi/180)*R+shift, zmin, 0.01};
Point(8) = { Cos(45*Pi/180)*R+shift,  Sin(45*Pi/180)*R+shift, zmin, 0.01};
Point(9) = {0+shift, 0+shift, zmin, 1.0};

//Define the lines that connect the points
Line(1) = {1, 2}; //Line from Point1 to Point2
Line(2) = {2, 3};
Line(3) = {3, 4};
Line(4) = {4, 1};
Line(5) = {4, 5};
Line(6) = {1, 6};
Line(7) = {7, 2};
Line(8) = {8, 3};
//Create the cylinder (a circle in 2D) with 4 arcs using Point9 as the middle point
Circle(9)  = {5, 9, 8}; //Circle arc from Point5 to Point8 and Point9 as the center
Circle(10) = {8, 9, 7};
Circle(11) = {7, 9, 6};
Circle(12) = {6, 9, 5};
//Create the Curve Loops (groups of 4 lines to create a closed region)
//In this step, it is very important to consider the orientation of each line.
//The 4 lines must create a circular loop.
Curve Loop(1) = {6, 12, -5, 4}; //Loop: Line6, Circle12, Line5(reversed), Line4
Curve Loop(2) = {1, -7, 11, -6};
Curve Loop(3) = {2, -8, 10, 7};
Curve Loop(4) = {3, 5, 9, 8};
//Create one surface for each Curve Loop
Plane Surface(1) = {1}; //Surface 1 is inside Curve Loop 1
```

```
Plane Surface(2) = {2};
Plane Surface(3) = {3};
Plane Surface(4) = {4};

//To create a high-quality mesh, it’s important to select a seed in each line.
//This way, you can control the accuracy in each region.
//1) Create a seed of 50 points in Line5(reversed), Line6(reversed), Line7 and Line8
//   Then, apply a progression of 1.1 instead of a uniform seed. The progression
//   allows you to gather the points near one end of each line.
Transfinite Curve{-5, -6, 7, 8} = 50 Using Progression 1.1;
//2) Create a uniform seed of 15 points in the external boundaries of your domain.
//Also, convert each surface to transfinite to create hexahedral elements.
//Bottom
Transfinite Curve{11, 1} = 15;
Transfinite Surface{2};

//Left
Transfinite Curve{4, 12} = 15;
Transfinite Surface{1};

//Up
Transfinite Curve{3, 9} = 15;
Transfinite Surface{4};

//Right
Transfinite Curve{2, 10} = 15;
Transfinite Surface{3};

//Recombine the surfaces to create the desired pattern
Recombine Surface {1, 2, 3, 4};
```

```
//Extrude the 2D mesh into a 3D mesh with 1 layer of height=zmax in the z-direction
Extrude {0, 0, zmax} {
 Surface{1,2,3,4};
 Recombine;
 Layers{1};
}
//Finally, assign labels to each surface and volume. These names will be used to define
// the boundary conditions in the control file used by HORSES3D.
//Before this step, load the *.geo file in the Gmsh GUI and check manually the number
// of each surface and volume.
//First, let’s create a new Physical Surface with the number 33 (use an empty number
// for the new surface) based on the original Surface8.
Physical Surface("inlet", 33) = {8};
//Repeat the operation for every boundary
Physical Surface("outlet", 34) = {14};
Physical Surface("up", 35) = {18};
Physical Surface("down", 36) = {10};
Physical Surface("wall1", 37) = {13, 17, 9, 20};
Physical Surface("wall2", 38) = {2, 3, 4, 1};
Physical Surface("cylinder", 39) = {19, 6, 16, 12};
//The whole volume is limited by Surfaces 1, 2, 3 and 4.
Physical Volume("fluid", 40) = {4, 3, 2, 1};
// uncoment next line to assing a order 3
// Mesh.ElementOrder = 3;
```

The whole file Custom_mesh.geo is located in **/tutorials/Cylinder/mesh_gmsh/**. Load it in the Gmsh GUI. You should see the following geometry:

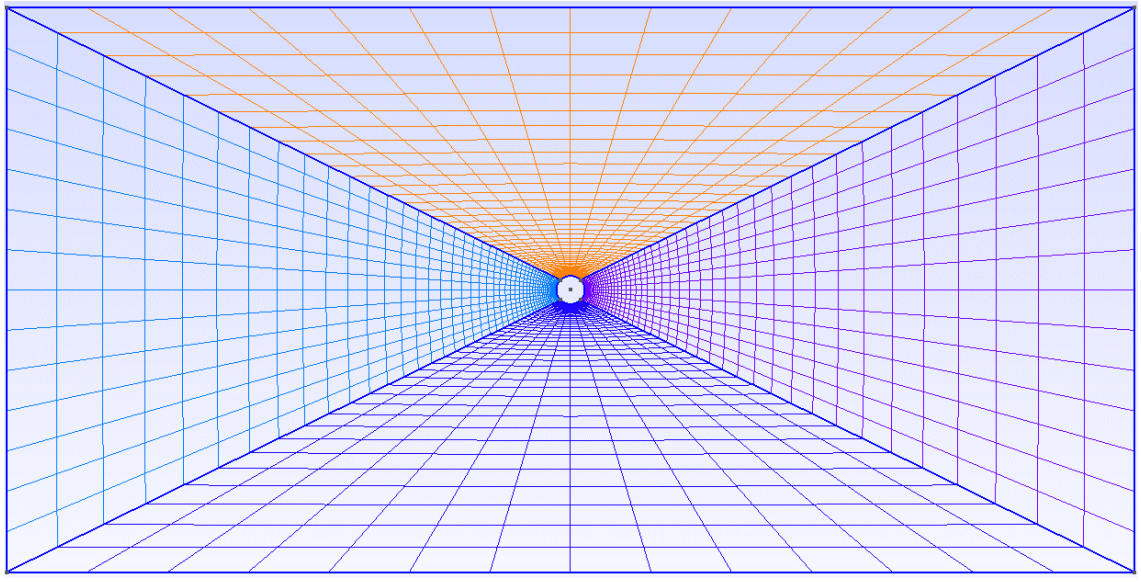


If you want to see the number of each point and surface, double-click on the screen and select "*All geometry options…*". Inside that menu, select visibility and choose what you want to see on the screen.

To create the mesh, click over *mesh* from the options of the tree on the left of the screen.

1) Click over 2D to create a face mesh based on your geometry.
2) Click over 3D to create the volumetric mesh and obtain the final mesh.
3) Click on File/Save Mesh to obtain your Custom_mesh.msh file.

The final mesh should look like this one:



Note: HORSES3D requires a mesh formed by hexahedral elements only. Check this requirement is fulfilled from the GUI before you export the final mesh:

1) Click on Tools/Options/Mesh
2) Go to the menu "Color" and select Coloring mode by Element Type.
3) Check if the whole mesh has the color assigned to hexahedra.

Copy the mesh file inside the MESH folder of your project and write the path to your new mesh in the control file.

You can control the curvature of the mesh with the last line, removing the comment. Current versions can handle up to order 5.

NOW YOU CAN RUN YOUR OWN SIMULATION!!

Note: Don't forget to compile the ProblemFile.f90 by running *make* inside the SETUP folder before you run the simulation.

Note2: Don't forget to adjust the boundary conditions (name and type) in your control file.