HERCULES Documentation

Release v1.0

Ping He

Jun 14, 2019

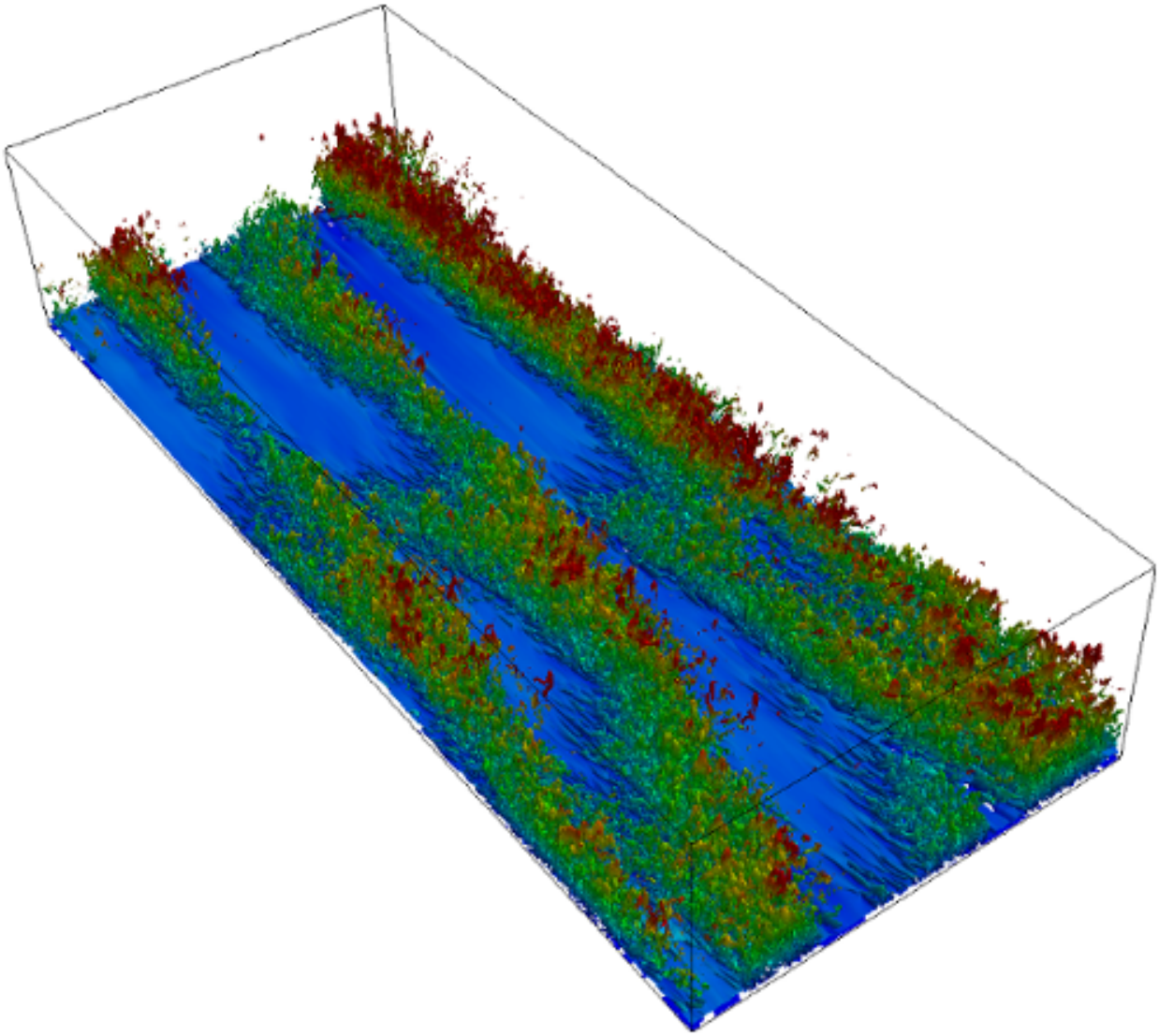
Contents

CHAPTER 1

HERCULES: A High-order Finite-difference Solver for Incompressible Boundary Layer Flows

HERCULES is an open-source computational fluid dynamics (CFD) code for simulating incompressible boundary layer flows. HERCULES is developed for high-performance turbulence simulations, and it can be used to conduct direct numerical simulation (DNS) of neutrally and stably stratified turbulent open-/closed-channel flows, as well as Ekman layer flows. HERCULES is written in Fortran 90. It has been tested on a number of HPC systems, e.g., ARL HPC Excalibur, AFRL HPC Lightening, and TACC Stampede, and is shown to have excellent parallel efficiency with up to 10,000 CPU cores.

HERCULES is configured for turbulent channel flow simulations in a rectangular wall-bounded domain with periodic boundaries in the horizontal directions. It solves the Navier-Stokes equations and the temperature equation using a high-order finite-difference approach. Spectral discretization can also be used for horizontal derivatives.



The HERCULES repository comprises of five main directories:

- doc: documentation
- license: license files
- misc: some utilities for pre-processing
- src: HERCULES source code
- tutorials: sample DNS simulations

Contents:

1.1 Download

HERCULES-v1.0 is available at:

https://github.com/friedenhe/hercules/archive/v1.0.zip

The latest version is available at:

https://github.com/friedenhe/hercules

1.2 Installation

To install HERCULES, you need a Linux environment and these software packages:

1. A GNU or Intel C compiler, e.g., gcc or icc.
2. A GNU or Intel Fortran compiler, e.g., gfortran or ifort
3. A MPI software, e.g., openmpi, mvapich2, or mpich

NOTE: if you use the GNU C and Fortran compilers, your MPI software should be also compiled by GNU.

The installation of HERCULES is straightforward. If you use **GNU compilers**, run:

```
sh install_GNU.sh
```

The installation will be automatically done.

NOTE: HERCULES depends on two external libs: FFTW and 2DECOMP_FFT. So when you run install_GNU.sh, these two libs will be (automatically) compiled first.

If you use **Intel compilers**, run this instead:

```
sh install_Intel.sh
```

Similarly, you can install HERCULES on **Cray** by running:

```
sh install_Cray_Intel.sh
```

After the installation is done, an executive named **hercules.exe** should be generated. This is the main program you will use for DNS simulations. In addition, you should see a file named **parameters.input** which defines the input parameters for the simulations. HERCULES comes with a default **parameters.input** file for DNS of plane closed-channel flows at Re_tau=180.

1.3 Tutorials

To run HERCULES, you need the main program **hercules.exe** (which should be generated after the installation) and the input file **parameters.input**. See the detailed explanation of the input parameters at the end of this page. In addition, you need to put the initial turbulent flow fields in the **results** folder. Otherwise, it will take a very long time to generate turbulence from a laminar flow field.

When you run **hercules.exe**, the code will try to read initial fields named **init_u.dat, init_v.dat**, etc. from the **results** folder. The best way to get the initial flow fields is from your previous simulations. Note that HERCULES will save intermediate flow fields named **bak_u.dat, bak_v.dat**, etc. for re-start runs. You can rename them to **init_u.dat, init_v.dat**, etc.

The above trick will not work if you are running HERCULES for the first time. Therefore, I prepare some coarse-resolution initial turbulent flow fields (e.g., **coarse_u_32_cubic.dat**) for your first run. These coarse flow fields come with HERCULES, they are in the **misc/Turb_Gen** folder. To interpolate these coarse-resolution flow fields to any resolution you need, you can use the Matlab code **Interpolate_Initial_Fields.m** provided in the **misc/Turb_Gen** folder. After you run the Matlab code, some initial fields named **init_u.dat, init_v.dat**, etc. will be generated. You need to move these files to the **results** folder.

NOTE: if you don't have an access to Matlab, use Octave (https://www.gnu.org/software/octave) instead. Octave is a free alternative to Matlab and it can be installed on Ubuntu by running:

```
sudo apt-get install octave
```

To run the Matlab script using Octave, do:

```
octave Interpolate_Initial_Fields.m
```

The followings are the specific steps you need to follow to run HERCULES for a few DNS benchmarks.

1. **Plane Closed-channel Flows at Re_tau=180**

   This is a classical DNS benchmark case for turbulent channel flows, please refer to Moser, Kim, and Mansour (1999), "Direct numerical simulation of turbulent channel flow up to Re_tau=590" in Physics of Fluids. (hereinafter referred to as MKM99).

   Since the default setup of HERCULES is for MKM99, you don't need to change **parameters.input**. You only need to prepare some initial turbulent fields. Here are the steps you need to follow.

   - Go to the **misc/Turb_Gen** folder, and run the Matlab script **Interpolate_Initial_Fields.m**. This script will generate some files named **init_u.dat, init_v.dat**, etc.
   - Move the generated files **init_u.dat, init_v.dat**, etc. to the **results** folder.
   - In the main folder, start the simulation using 2 CPU cores by running:

     ```
     mpirun -np 2 hercules.exe
     ```

   - The simulation results (including the instantaneous, mean, and re-start files) will be stored in the **results** folder. These files are in binary format. To plot the results (e.g., the mean profiles), you can use the Matlab script **Plot_Mean_Profiles.m** in the **/misc/Post_Processing** folder.
   - By default, the simulation will be run for 20 non-dimensional times. You probably need a longer time for a fully developed turbulent field. To do this, after the above simulation is done, you can rename the re-start files **bak_u.dat** to **init_u.dat** (same for other variables). Then, you can re-run the simulation with the new initial fields:

     ```
     mpirun -np 2 hercules.exe
     ```

   - If you want to use more CPU cores, you need to change p_row and p_col in **parameters.input** accordingly. For example, you want to run HERCULES using 8 cores, you need to change p_row=4 and p_col=2 in **parameters.input**, and run:

     ```
     mpirun -np 8 hercules.exe
     ```

   So far, you should be able to run HERCULES for the MKM99 case. Good luck!

2. **Stably Stratified Closed-channel Flows at Re_tau=180 and Ri_tau=18**

   This is a benchmark case for stratified turbulent channel flows, please refer to Garcia-Villalba and del Alamo (2011), "Turbulence modification by stable stratification in channel flow" in Physics of Fluids (hereinafter referred to as GD11).

   To run HERCULES for GD11, you need to change some parameters in **parameters.input**. In addition, you need to prepare some initial turbulent fields. Here are the steps you need to follow.

   - Copy **tutorials/parameters.input.GD11** to the main folder and rename it to **parameters.input**; replace the old **parameters.input**.

- Go to the **misc/Turb_Gen** folder. In the Matlab script **Interpolate_Initial_Fields.m**, change nx2=288, ny2=288, nz2=128. These are the grid numbers we need for the GD11 case. Run the script, and it will generate some files named **init_u.dat, init_v.dat**, etc.
- Move the generated files **init_u.dat, init_v.dat**, etc. to the **results** folder.
- In the main folder, start the simulation using 2 CPU cores by running:

```
mpirun -np 2 hercules.exe
```

You can follow the steps 4 and 5 in the MKM99 case for plotting the output and doing the re-start runs.

3. **Neutrally Stratified Ekman Layer Flows at Re_g=400**

This is a benchmark case for Ekman layer flows, please refer to Coleman, Ferziger, and Spalart (1990), "A numerical study of the turbulent Ekman layer" in Journal of Fluid Mechanics. (hereinafter referred to as CFS90). Here are the steps you need to follow.

- Copy **tutorials/parameters.input.CFS90** to the main folder and rename it to **parameters.input**; replace the old **parameters.input**.
- Go to the **misc/Turb_Gen** folder. In the Matlab script **nterpolate_Initial_Fields.m**, change nx2=288, ny2=288, nz2=128. In addition, change Uscaling=0.06. Note that we need to scale the initial field for the CFS90 case. Run the script, and it will generate some files named **init_u.dat, init_v.dat**, etc.
- Move the generated files **init_u.dat, init_v.dat**, etc. to the **results** folder.
- In the main folder, start the simulation using 2 CPU cores by running:

```
mpirun -np 2 hercules.exe
```

Again, the CFS90 case needs a longer time to converge, you can follow the steps 4 and 5 in the MKM99 case for plotting the output and doing the re-start runs.

Here is a sample **parameters.input** file for the MKM99 case. The meaning of each parameter is explained as follows:

```
&domain
  p_row      = 2
  p_col      = 1
  lx         = 12.56637061
  ly         = 4.188790205
  lz         = 2.0
  nx         = 256
  ny         = 192
  nz         = 128
  zstretch   = 1.1
  ochannel   = 0
/

&modeling
  dp_opt     = 1
  cds        = 2
  issk       = 1
  dts        = 2
  isnoise    = 0
  noise_mag  = 0.1
  isdamp     = 0
  nzdamp     = 15
  cdamp      = 0.01
```

(continues on next page)

(continued from previous page)

```
/

&constants
  nu          = 0.005555556
  alpha       = 0.007936508
  got         = 0.0
  fc          = 0.0
  ug          = 0.0
  vg          = 0.0
  dt          = 0.001
  imax        = 20000
  u_mrf       = 0.0
  isscalar    = 0
  t_ref       = 0
  tbot        = -0.5
  ttop        = 0.5
  is_ri_var   = 0
  ri_str      = 0.0
  ri_end      = 0.0
/

&io
  restart     = 1
  istmsr      = 1
  ibackup     = 2000
  iinstfl     = 2000
  imeanfl     = 2000
  isxy2d      = 0
  xy2d_id     = 10,32,64
  isxz2d      = 0
  xz2d_id     = 32,64
  isyz2d      = 0
  yz2d_id     = 32,64
  intv_2d     = 1000
/
```

p_row, p_col These two parameters control the domain decomposition in the x and y directions for parallel simulation. The grid numbers nx and ny should be divisible by p_row, and the grid numbers ny and nz should be divisible by p_col. Make sure to use p_row*p_col cores for parallel simulation. To ensure high parallel efficiency, p_row shouldn't be too close to nx and ny, and p_col shouldn't be too close to ny and nz. In addition, p_row and p_col should be as close as possible.

lx, ly, lz Domain sizes in the x, y, and z directions.

nx, ny, nz Number of cells in the x, y, and z directions.

zstretch This parameter controls the grid refinement near the wall. The larger zstretch the finner grids near the wall. The default value for zstretch is 1.1.

ochannel This parameter controls the channel configuration. 0: closed channel flows (non-slip boundary condition for the top and bottom walls). 1: open channel flows (non-slip and slip boundary conditions for the bottom and top walls, respectively).

dp_opt This parameter controls the options to drive the flow. 1: constant friction. Re and Ri are friction Reynolds number and Richardson number in this case. 2: constant bulk velocity. Re and Ri are bulk Reynolds number and Richardson number in this case. 3: constant geostrophic wind for Ekman layer. Re and Ri are geostrophic Reynolds number and and bulk Richardson number in this case.

cds Differential scheme for spatial derivatives in the horizontal directions. 1: 2nd order central difference scheme. 2:

4th order central difference scheme. 3: Spectral method.

issk If the nonlinear terms are casted in skew-symmetric form (only for cds=3). For the spectral method, the skew-symmetric form has a better numerical stability (compared with the divergence form) and is strongly recommended. 0: no. 1: yes.

dts Time advancement scheme for the convective terms (for the diffusion term, the Crank-Nicolson scheme is used.) 1: 2nd order Adam-Bashforth. 2: 3rd order Runge-Kutta.

isnoise, noise_mag Add random noise for the initial fields (to generate turbulence). isnoise=1: add random noise to the initial fields. noise_mag: the magnitude (%) of the random noise if isnoise=1.

isdamp, nzdamp, cdamp Rayleigh damping parameters. isdamp=1: apply Rayleigh damping to the domain top. nzdamp: how many points for damping if isdamp=1. cdamp: damping coefficient if isdamp=1.

nu Kinematic viscosity. For the constant friction simulations (dp_opt=1), set nu=1/Re_tau. Similarly, for the constant mass flow rate simulation, set nu=1/Re_b. For the Ekman layer case, set nu=1/Re_g.

alpha Thermal diffusivity. Usually, we set alpha=nu/Pr, where Pr=0.7.

got g/T. Here g is the gravitational acceleration and T is temperature. Set this parameter to non-zero for stratified flow simulation. For the constant friction simulations (dp_opt=1), set got=Ri_tau. Similarly, for the constant mass flow rate simulation, set got=Ri_b. For the Ekman layer case, set got=Ri_g.

fc Coriolis parameter. This parameter is for Ekman layer simulation. Set it to nu*2.0.

ug, vg Geostrophic wind speeds. Set ug to 1.0 for Ekman layer simulation.

dt Time step. dt is restricted by the maximal CFL number.

imax How many steps to run.

u_mrf Moving speed of the reference frame. The default value is 0. Set it to be about half of the bulk velocity can reduce the maximal CFL number.

isscalar Whether to include scalar (temperature) for the simulation.

tbot and ttop Bottom wall and top wall temperature. For stratified closed channel flow case, set tbot=-0.5 and ttop=0.5. For stratified open-channel case, set tbot=-1.0 and ttop=0.0.

is_ri_var and ri_str and ri_end These parameters define the time-varying Richardson number value during the simulation. is_ri_var=1: Richardson number will vary during the simulation. ri_str: initial ri value if is_ri_var=1. ri_end: final ri value if is_ri_var=1.

restart 1: read initial field from "results/init_u.dat", "results/init_v.dat", etc. 0: use a default initial field (u=0,v=0,w=0,t=0,p=0).

istmsr istmsr=1: output time-series of u, v, w, p, t collected from a vertical line in the center of the domain.

ibackup, iinstfl, imeanfl These parameters control the output frequency of the backup, instantaneous, and mean fields. ibackup: output frequency of the backup fields. iinstfl: output frequency of the instantaneous fields. imeanfl: output frequency of the mean fields.

isxy2d, xy2d_id isxy2d=1: output 2D slices of u, v, w, t at specific x-y planes. xy2d_id: the k index for the 2D slice, you can give values for multiple levels.

isxz2d,xz2d_id isxz2d=1: output 2D slices of u, v, w, t at specific x-z planes. xz2d_id: the j index for the 2D slice, you can give values for multiple levels.

isyz2d, yz2d_id isyz2d=1: output 2D slices of u, v, w, t at specific y-z planes. yz2d_id: the i index for the 2D slice, you can give values for multiple levels.

intv_2d The output frequency of the 2D slices output (steps).

1.4 Development

HERCULES is written in Fortran 90. The Doxygen documentation of it source codes is at:

HERCULES Doxygen

1.5 Contact

If you have questions, please contact: Ping He (friedenhe@gmail.com)