1 Theoretical background

1.1 System of equations

We are considering forced MHD turbulence, which is described by the following set of equations:

$$\rho \frac{D\boldsymbol{U}}{Dt} = (\nabla \times \boldsymbol{B}) \times \boldsymbol{B} - \nabla p + \nabla \cdot (2\nu\rho\mathsf{S}) + \rho \boldsymbol{f}, \quad (1)$$

$$\frac{\partial \boldsymbol{B}}{\partial t} = \nabla \times [\boldsymbol{U} \times \boldsymbol{B} - \eta\,(\nabla \times \boldsymbol{B} - \mu \boldsymbol{B})], \quad (2)$$

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \boldsymbol{U}, \quad (3)$$

where ρ is the density, $\boldsymbol{U}$ is the velocity field, $\boldsymbol{B}$ is normalized such that the magnetic energy density is $\boldsymbol{B}^2/2$ (without the 4π factor), and $D/Dt = \partial/\partial t + \boldsymbol{U} \cdot \nabla$ is the advective derivative. Further, η is the microscopic magnetic diffusivity, p is the fluid pressure, $\mathsf{S}_{ij} = \frac{1}{2}(U_{i,j} + U_{j,i}) - \frac{1}{3}\delta_{ij}\nabla \cdot \boldsymbol{U}$ are the components of the trace-free strain tensor S (commas denote partial spatial derivatives), ν is the kinematic viscosity, and $\boldsymbol{f}$ is the turbulent forcing function.

1.2 Forcing of turbulence

We use a solenoidal random forcing function $\boldsymbol{f}$ of the form

$$\boldsymbol{f}(\boldsymbol{x}, t) = Re\{N \boldsymbol{f}_{\boldsymbol{k}(t)} \exp[i\boldsymbol{k}(t) \cdot \boldsymbol{x} + i\phi(t)]\}, \quad (4)$$

where $\boldsymbol{k}(t) = (k_x, k_y, k_z)$ is a random time-dependent wave vector, $\boldsymbol{x} = (x, y, z)$ is position, and $\phi(t)$ with $|\phi| < \pi$ is a random phase. On dimensional grounds the normalization factor is chosen to be $N = f_0 c_\mathrm{s}(k c_\mathrm{s}/\delta t)^{1/2}$, where f_0 is a nondimensional factor, $k = |\boldsymbol{k}|$, and δt is the length of the timestep. The $\delta t^{-1/2}$ dependence ensures that the forcing, which is delta-correlated in time, is properly normalized such that the correlator of the forcing function is independent of the length of the time step, δt. We force the system with eigenfunctions of the curl operator,

$$\boldsymbol{f}_{\boldsymbol{k}} = \frac{i\boldsymbol{k} \times (\boldsymbol{k} \times \boldsymbol{e}) - \sigma|\boldsymbol{k}|(\boldsymbol{k} \times \boldsymbol{e})}{\sqrt{1+\sigma^2}\ \boldsymbol{k}^2 \sqrt{1 - (\boldsymbol{k} \cdot \boldsymbol{e})^2/\boldsymbol{k}^2}}, \quad (5)$$

where $\boldsymbol{e}$ is an arbitrary unit vector needed in order to generate a vector $\boldsymbol{k} \times \boldsymbol{e}$ that is perpendicular to $\boldsymbol{k}$. Note that $|\boldsymbol{f}_{\boldsymbol{k}}|^2 = 1$ and, for $\sigma = 1$, $i\boldsymbol{k} \times \boldsymbol{f}_{\boldsymbol{k}} = |\boldsymbol{k}|\boldsymbol{f}_{\boldsymbol{k}}$, so the helicity density of this forcing function satisfies

$$\boldsymbol{f} \cdot \nabla \times \boldsymbol{f} = |\boldsymbol{k}|\boldsymbol{f}^2 > 0 \quad (\text{for } \sigma = 1) \quad (6)$$

at each point in space. We note that since the forcing function is like a delta-function in $\boldsymbol{k}$-space, this means that all points of $\boldsymbol{f}$ are correlated at any instant in time, but are

Contact: jennifer.schober@epfl.ch

different at the next timestep. Thus, the forcing function is delta-correlated in time (but the velocity is not). This is the forcing function used in Brandenburg (2001), Brandenburg & Dobler (2001), and other papers in that series.
For $\sigma = 0$, the forcing function is completely nonhelical and reduces to the simpler form

$$\boldsymbol{f}_{\boldsymbol{k}} = (\boldsymbol{k} \times \boldsymbol{e}) / \sqrt{\boldsymbol{k}^2 - (\boldsymbol{k} \cdot \boldsymbol{e})^2}. \tag{7}$$

For $0 < |\sigma| < 1$, the forcing function has fractional helicity, where $\sigma \approx \langle \boldsymbol{\omega} \cdot \boldsymbol{u} \rangle / (k_{\rm f} \langle \boldsymbol{u}^2 \rangle)$; see Sect. 4.5 of Ref. Brandenburg et al. (Astron. Nachr. 323, 99–122, 2002). In the code and the forcing_run_pars namelist, σ is called relhel.

2 Code setup

This is a 3D run with 32^3 cells, as can be seen in src/cparam.local:

```
integer, parameter :: ncpus=1,nprocy=1,nprocz=ncpus/nprocy,nprocx=1
integer, parameter :: nxgrid=32,nygrid=nxgrid,nzgrid=nxgrid
```

The modules included in this run are, see src/Makefile.local:

```
MPICOMM  = nompicomm
EOS      =   eos_idealgas
HYDRO    =   hydro
DENSITY  =   density
ENTROPY  = noentropy
MAGNETIC =   magnetic
GRAVITY  = nogravity
FORCING  =   forcing

FOURIER  =   fourier_fftpack
POWER    =   power_spectrum
```

The last two lines indicate the Fourier spectra are calculated on the way.

As can be seen above and in start.in, instead of solving the entropy equation, here an ideal equation is chosen. Furthermore, a weak magnetic field is set up in form of Gaussian noise:

```
&eos_init_pars
  gamma=1.
/
&magnetic_init_pars
  initaa='gaussian-noise', amplaa=1e-4
/
```

The forcing of turbulence is set in run.in:

```
&forcing_run_pars
  iforce='helical', force=0.07, relhel=1.
/
```


Contact: jennifer.schober@epfl.ch

In the code, the possible wavevectors are pre-calculated and stored in k.dat, which is being read in the beginning the code runs. To change the wavevectors (e.g., the typical value of k_f, you need to change the file. In the directory $PENCIL_HOME/samples/helical-MHDturb/K_VECTORS/ there are several such files prepared:

```
k10.dat  k1.dat   k2.dat   k3.dat  k5.dat
k15.dat  k27.dat  k30.dat  k4.dat  k8.dat
```

and more can be prepared in Python with the procedure python/generate_kvectors.py. Helical forcing is being used with maximum relative helicity (relhel=1). For non-helical forcing, simply set relhel=0.

3 Exercises

(a) Run the simulation and analyse the kinetic and magnetic energy spectra. Do they agree with what you expect for the forcing vector as given in k.dat? What is the slope of the kinetic energy spectrum?

(b) Measure the Reynolds number of the run. Set up new runs in which you change the Reynolds number by (i) changing ν, (ii) varying the strength of the forcing, and (iii) by using different forcing vectors. You can change the forcing vectors by coping pre-made vectors into your run directory and naming them k.dat, or you can generate your own forcing vectors with the python script.

(c) Analyze the evolution of B_rms. How does it evolve at late times? Can you guess what causes this late time evolution? How does changing relhel from 1 to 0 change the evolution of B_rms?

Contact: jennifer.schober@epfl.ch