Pencil Code: Quick Start guide

Illa R. Losada, Michiel Lambrechts, Elizabeth Cole, Philippe Bourdin

February 4, 2015

Contents

1 Download the Pencil Code **2**

2 Configure the shell **2**

3 Fortran **2**
3.1 Fortran on a mac machine . . . 2

4 Try a sample **3**
4.1 Setting up... . . . 3
4.2 Makefile . . . 3
4.2.1 Single-processor . . . 3
4.2.2 Multi-processor . . . 3
4.3 Compiling... . . . 4
4.3.1 Using a different compiler (optional) . . . 4
4.3.2 Chaning compiler options (optional) . . . 4
4.4 Running... . . . 4
4.5 IDL visualization (optional) . . . 5
4.6 Setting up Python for data processing (optional) . . . 5
4.6.1 Python modules requirements . . . 5
4.6.2 Installation . . . 5
4.6.3 Using the Pencil module . . . 6

5 Another example: helically forced turbulence **6**
5.1 Solution. . . . 6
5.1.1 Critical value for the magnetic diffusivity. . . . 6
5.1.2 Magnetic Reynolds number . . . 7
5.1.3 Growth rate of the magnetic field . . . 7
5.1.4 Structure of the magnetic field. . . . 8
5.1.5 Fitting B_{zx} . . . 8

1 Download the Pencil Code

The Pencil Code is an open source code written mainly in `Fortran` and available under GPL. General information can be found at our official homepage:

`http://pencil-code.nordita.org/`.

The latest version of the code can be downloaded with `svn`. In the directory where you want to put the code, type:

```
svn checkout http://pencil-code.googlecode.com/svn/trunk/ pencil-code
```

The downloaded `pencil-code` directory contains several sub-directories

1. `doc`: a very important directory containing the pencil-code `manual.tex`. The pdf of the latest version of the manual is created simply by typing `make` in the `pencil-code/doc` directory. For example, the code structure can be further explored in Section 4 of the manual.
2. `samples`: contains many sample problems
3. `config`: has all the configuration files
4. ...

2 Configure the shell

For a quick start, you need to load some environment variable into your shell. First, you enter to the freshly downloaded directory:

```
cd pencil-code
```

Depending on which shell you use, you can do that by a simple command:

```
. sourceme.sh
```

that will work for `bash` and all `sh`-compatible shells, while this command:

```
source sourceme.csh
```

is for `tcsh` and any `csh`-compatible shell.

3 Fortran

A `Fortran` and a `C` compiler are needed to compile the code. Both compilers should belong to the same distribution package and version (e.g. GNU GCC 4.8.3, 64 bit Linux).

3.1 Fortran on a mac machine

For Mac, you first need to install `Xcode` from the `AppleDeveloper` site `http://developer.apple.com/`. This requires you to first register as a member. An easy to install `gfortran` can be found at

http://gcc.gnu.org/wiki/GFortranBinaries. Just download it and it comes with an installer. It installs in the directory /usr/local/gfortran with a symbolic link in /usr/local/bin/gfortran. It might be necessary to add the following line in the .cshrc-file in the home folder:

```
setenv PATH /usr/local/bin:\$PATH
```

4 Try a sample

Go to a folder that contains one of the many available samples, e.g.:

```
cd samples/1d-tests/jeans-x
```

You may also start with a fresh directory and copy over the files from one of the samples.

4.1 Setting up...

One command sets up all needed symbolic links to the original Pencil Code directory:

```
pc_setupsrc
```

4.2 Makefile

Two basic configuration files define a simulation setup: src/Makefile.local contains a list of modules that are being used, and src/cparam.local defines the grid size and the number of processors to be used. Take a quick look at these files...

4.2.1 Single-processor

An example using the module for only one processor would look like:

```
MPICOMM=nompicomm
```

For most modules there is also a no-variant which switches that functionality off.

In src/cparam.local the number of processors needs to be set to 1 accordingly:

```
integer, parameter :: ncpus=1,nprocx=1,nprocy=1,nprocz=ncpus/(nprocx*nprocy)
integer, parameter :: nxgrid=128,nygrid=1,nzgrid=128
```

4.2.2 Multi-processor

If you like to use MPI for multi-processors simulations, be sure that you have a MPI library installed and change src/Makefile.local to use MPI:

```
MPICOMM=mpicomm
```

Change the ncpus setting in src/cparam.local. Think about how you want to distribute the volume on the processors — usually, you should have 128 grid points in the x-direction to take advantage of the SIMD processor unit. For compilation, you have to use a configuration file that includes the _MPI suffix, see below.

4.3 Compiling...

In order to compile the code, you can use a pre-defined configuration file corresponding to your compiler package. E.g. the default compilers are gfortran together with gcc and the code is being built with default options by issuing the command:

```
pc_build
```

4.3.1 Using a different compiler (optional)

If you prefer to use a different compiler package (e.g. using ifort or MPI), you may try:

```
pc_build -f compilers/Intel
pc_build -f compilers/GNU-GCC_MPI
```

More pre-defined configurations are found in the directory pencil-code/config/compilers/*.conf.

4.3.2 Chaning compiler options (optional)

Of course you can also create a configuration file in any subdirectory of pencil-code/config/hosts/. By default, pc_build looks for a config file that is based on your host-ID, which you may see with the command:

```
pc_build -i
```

You may add your modified configuration with the filename host-ID.conf, where you can change compiler options according to the Pencil Code manual.

4.4 Running...

The initial conditions are set in start.in and the parameters for the main simulation run can be found in run.in. In print.in you can choose which physical quantities are written to the file data/time_series.dat.

It is time to run the code with — be sure you have an empty data directory:

```
mkdir data
pc_run
```

Welcome to the world of Pencil Code! Visualizing the output can be done with IDL or Python, see below.

4.5 IDL visualization (optional)

Start `idl` from the command line. Several `idl` have been written (you can find them in `pencil-code/idl`) to facilitate inspecting the data (which can be found in raw format in `jeans-x/data` directory). For example, let us inspect the time series data

```
IDL> pc_read_ts, obj=ts
```

The structure `ts` contains several variables that can be inspected by

```
IDL> print, tag_names(ts)
IT T UMAX RHOMAX
```

The diagnostic `UMAX`, the maximal velocity, is available since it was set in `jeans-x/print.in` (more on diagnostic output can be found in section 5.4 in the manual). We can now plot the evolution of the maximal velocity in time

```
IDL> plot, ts.t, alog(ts.umax)
```

after the initial perturbation we inserted in `start.in`.

The complete state of the simulation, a snapshot, is saved (as `jeans-x/data/proc0/VAR$\dots@`, every `dsnap` time units (see `jeans-x/run.in`). These states can be inspected, for example

```
IDL> pc_read_var, obj=ff, ivar=1, /trimall
```

Similarly `tag_names` will provide us with the available variables,

```
IDL> print, tag_names(ff)
T X Y Z DX DY DZ UU LNRHO POTSELF
```

so the logarithm of the density in the simulated domain can be inspected by

```
IDL>plot, ff.lnrho
```

4.6 Setting up Python for data processing (optional)

4.6.1 Python modules requirements

The basic needed modules are: numpy and matplotlib.

- numpy: all array definitions and operations.
- matplotlib: plotting.

Other really useful modules are: ipython and scipy.

- ipython: enhanced python interpreter.
- scipy: science functions and utilities.

4.6.2 Installation

Untar the `tar.gz` file or go to the directory and simple type as root or sudoed:

```
python setup.py install
```

For a user installation (no root permission):

```
python setup.py install --user
```

4.6.3 Using the Pencil module

Import the module:

```
import pencil as pc
```

Some useful functions:

pc.read_ts	Read "time_series.dat" file. Parameters are added as members of the class.
pc.read_slices	read 2D slice binary files and return two arrays: one of (nslices,vsize,hsize) and other of tim
pc.animate_interactive	Assemble a 2D animation from a 3D array.

5 Another example: helically forced turbulence

Question: Simulate the saturation behaviour of a dynamo from helically forced turbulence with forcing wave-number $k_f = 3$ in units of the box wave-number $k_1 = 1$. Use the sample `samples/helical-MHDturb`.

1. Determine the critical value of the magnetic diffusivity above which there is a growth of the rms magnetic filed, `brms`, in the file `data/time\_series.data`
2. Determine the corresponding value of the magnetic Reynolds number, $Re_M = u_{rms}/\eta\kappa_f$
3. Determine the growth rate of the magnetic fiedl for a choseen value of the magnetic diffusivity about half the critical vaule.
4. Determine the estructure of the magnetic field. Consider the evolution of 3 different magnetic field averages: the `xy` average `bmz`, the `yz` average `bmx` and the `xz` average `bmy`. Run until saturation and determine which of the three averages dominates in the end.
5. Fit the resulting ($\bar{B^2}$) to the expression:

$$F(t;B_0,t_s) = B_0^2[1-\exp^{-2\eta\kappa_1^2(t-t_s)}] \tag{1}$$

5.1 Solution.

5.1.1 Critical value for the magnetic diffusivity.

Calculation of the critical value for the magnetic diffusivity implies running the code for different values of the magnetic diffusivity and checking up at which value the field starts growing.

In figure 1 the growth of the rms magnetic field versus time can be analyzed for different magnetic diffusivity values.

Using these results, I set the critical value for the magnetic diffusivity in $\eta = 22e-3$.

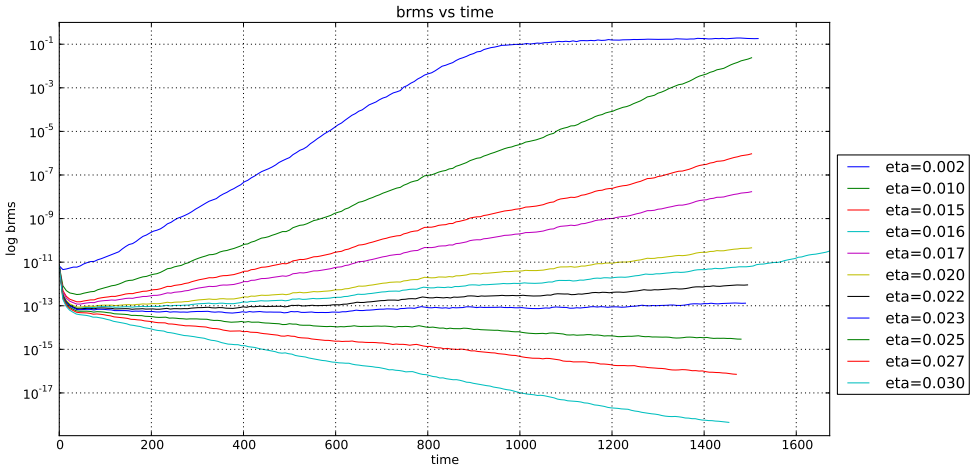

Figure 1: log(brms) vs time for different values of magnetic diffusivity.

5.1.2 Magnetic Reynolds number

The magnetic Reynolds number is defined as:

$$Re_M = \frac{u_{rms}}{\eta \kappa_f} \tag{2}$$

We have set $\kappa_f = 3$, the critical value for the magnetic diffusivity found is $\eta = 22e-3$ and the mean value for u_{rms} is $\bar{u_{rms}} = 0.14$, so the corresponding magnetic Reynolds number is $Re_M = 2.15$.
The magnetic Reynolds number is the ratio between convection and diffusion. When $Re_M \gg 1$, convection dominates, whereas for $Re_M \approx 1$, or less, diffusion becomes important. So, in this exercise, diffusion is about to become important. In fact, the table 5.1.2 shows that Re_M decreases as η increases

η	Re_M
2e-3	22.39
10e-3	4.73
15e-3	3.16
16e-3	2.97
17e-3	2.78
20e-3	2.37
22e-3	2.15
23e-3	2.06
25e-3	1.89
27e-3	1.75
30e-3	1.58

5.1.3 Growth rate of the magnetic field

In order to determine the growth rate of the magnetic field for a value of the magnetic diffusivity, one must fit the sloped part of the logarithm of *brms* vs time curve, as shown in the figures 2 and 3.

The parameter a in a linear regression fit ($y = ax + b$) is a measure of the growth rate of the magnetic field. The growth depends on the value of the magnetic diffusivity, and it decreases as the magnetic diffusivity increases.

η	a	b
2e-3	0.0121	-12.12
10e-3	0.0075	-13.11

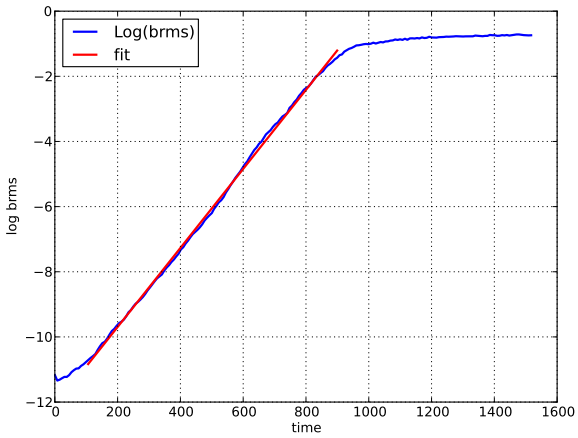

Figure 2: Growth rate of the magnetic field for a value of the magnetic diffusivity $\eta = 2e-3$.

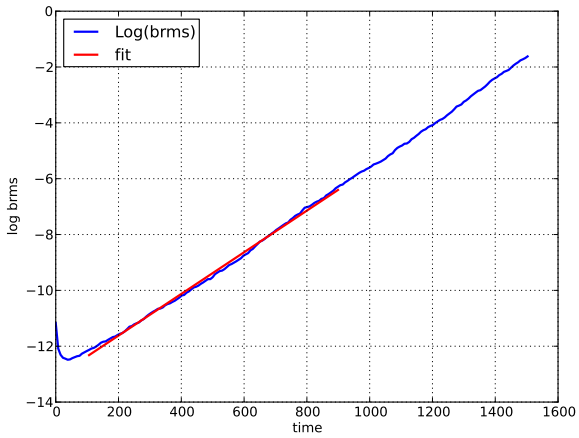

Figure 3: Growth rate of the magnetic field for a value of the magnetic diffusivity $\eta = 10e-3$.

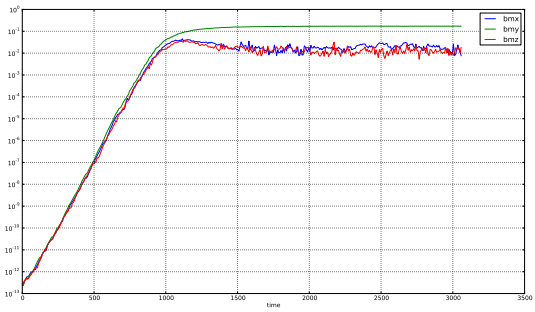

Figure 4: Averages of three different magnetic fields.

Figure 4 shows the evolution of three different magnetic field averages: the xy average bmz, the yz average bmx and the zx average bmy. From the figure one can see that the zx average dominates in the end.

5.1.4 Structure of the magnetic field.

5.1.5 Fitting B_{zx}

The strongest of the three field averages is the zx average represented in the figure 5 with different values of the expression 1. A rough fit for the field average takes the values $B_0 = 0.1712$ and $t_s = 930$. B_0 is the saturation field value and t_s is the best fit curve just before saturation.

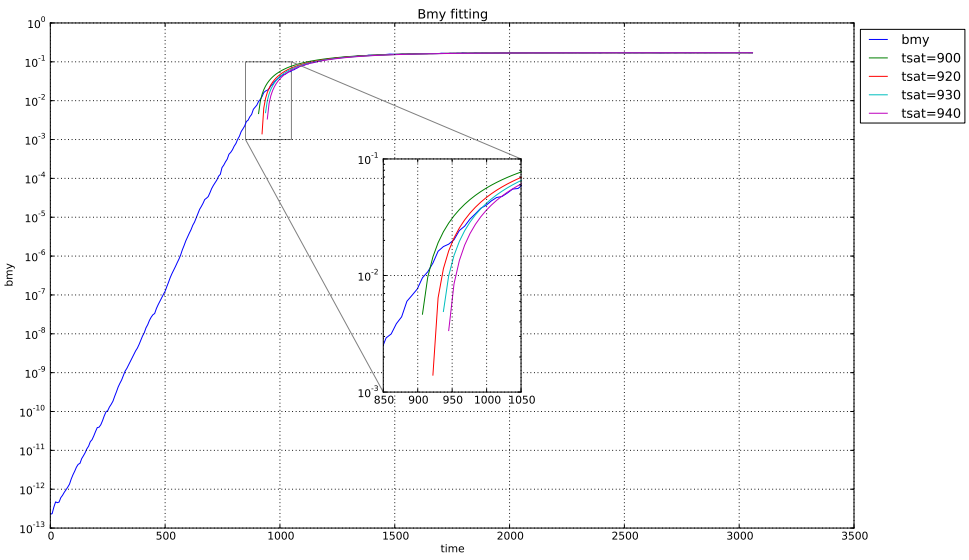

Figure 5: Bmy Fitting: $B_0 = 0.1712$ and $t_s = 930$.