

Week 1: 11 Nov - 13 Nov

Started this week learning about how to use MPI in Athena++. Followed the [3D blast wave tutorial](#) and managed to get it working on Thunderbird with HDF5. Jono also recommended learning a text editor, so I spent an hour learning the basics of Vim. **Note:** HDF5 is better to use than .vtk with parallel computing as it allows all the processors being used to write the data in one file compared to a file for each processor that need to be joined (less hassle).

Jono then gave me an Athena++ hydrodynamic turbulence input script to play around with, and some MATLAB scripts that analyse the energy spectrum of the fluid in question. At the moment, we model the fluid in a cube. There are two different modes that we're wanting to focus on: decaying turbulence (disturb the fluid initially then leave it to its own devices) and continuously driven turbulence. Ran the Athena++ code with 3 different grid sizes for both modes; see screenshots below.



Figure 1: Face-on view of the 3D forced turbulence simulations with different grid sizes; density plotted

These were just the simulations with none of the parameters changed in the input script. Using the MATLAB scripts I also obtained the energy spectrum of the fluid; was very rough due to the low resolution. Wanted to try start a 256^3 grid size simulation before I left on Wednesday but didn't have enough time to set up; will try again later. Thursday and Friday of this week were spent at the [Otago Software Carpentry Workshop](#).

Next Week: Play around with parameters. Want to run the larger grid size simulations to obtain a better energy spectrum that fits the $k^{-5/3}$ law, will add screenshots of spectrum then. Plot the total energy over time for both modes; should observe fluctuations in the energy for forced turbulence.

Week 2: 18 Nov - 22 Nov

This week I ran bigger simulations for both decay and forced turbulence from last week in order to be able to plot the energy spectrum and time evolution. The grid size ranged from 32 to 256, and runs over 30 seconds. All simulations left the parameters in `athinput.turb` unchanged.

Calculated the turnover time $\tau \sim L/u_l$ of eddies on the scale of the box to get an idea of the timescales involved in the energy cascade. This is important in the decaying case as all the input energy dissipates within a few turnover times, so this allowed me to get an approximate time range to average the energy spectrum over. For the decay simulations I used $L = 1$ and $u_L = \sqrt{u_x^2 + u_y^2 + u_z^2}$ taken at the start of the simulation; for the continuous simulations the average turnover time in the saturated state was used. The data was gotten from the `Turb.hst` file.

Continuous Forcing: For the continuously forced case, the 256 grid size gave the best result. This is expected as it is able to simulate smaller scale eddies compared to lower resolution simulations, allowing more of the energy cascade to be observed. We see that the energy spectrum does follow the $k^{-5/3}$ law (shown in Fig. 2a) for a given range of wavevectors.

The energy evolution (Fig. 2b) shows an increase in kinetic energy until it levels out after a few turnover times. This leveling out is due to the energy dissipation rate matching the energy input rate from the forcing. There is still some variation around the mean value, which arises from fluctuations due to turbulence.

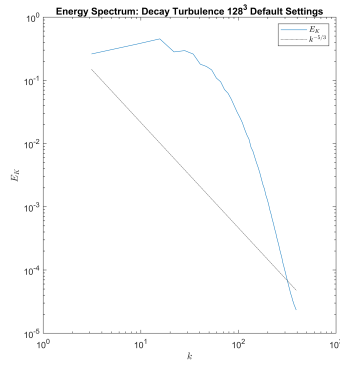


Figure 2: Plots showing the energy spectrum and time evolution of the 256 grid size continuous forcing simulation

Decay: The energy spectrum (Fig 3a), averaged over the first two turnover cycles, is not

as well defined as the continuous case. I think this is because the energy depends on time instead of being approximately constant. The energy evolution (Fig. 3b) shows that the total kinetic energy decreases over time due to dissipation from viscosity.

For the spectrum evolution (Figure 3c) I took snapshots of the spectrum at 3 second intervals, with a snapshot at 0.1 seconds to observe the energy input (the curve sharply peaked at $k \sim 10$). We see that the energy cascades down through the length scales directly after input (at 6 seconds), and then decreases as it is dissipated as heat at the micro scale (shown by the “sinking” of the spectrum). This is expected as we saw that the total energy in Fig. 3b is decreasing.



(a) Averaged Spectrum



(b) Energy Evolution



(c) Spectrum Evolution

Figure 3: Plots showing the energy spectrum and time evolution of the 128 grid size decay simulation

The magnetic and thermal energy had no relevance to these simulations; it's just part of the MATLAB script that I forgot to remove.

I learnt that it's good to run the simulations at different resolutions starting with the lowest as it allows you to get a rough idea of what is going to happen without the expense of computing time. It also helps as you can compare with the model to see whether the configuration used is worth investigating. The higher resolutions could differ as turbulence depends strongly on all length scales in the inertial subrange, which are included in the bigger simulations, but it still helps to see what could happen.

Week 3: 25 Nov - 29 Nov

Ran the linear wave problem in Athena to observe Alfvén waves. The pressure and density of the plasma should be constant as these waves travel, which was observed in the simulation (Fig. 4). Alfvén waves have the following dispersion relationship:

$$\omega = k_{\parallel} v_A$$

where k_{\parallel} is the component of the wavevector parallel to the magnetic field and $v_A = B_0 / \sqrt{4\pi\rho}$ is the Alfvén velocity at which the waves move, where B_0 and ρ are the magnetic field strength and fluid density.

The linear wave problem has the default settings (I'm not sure about units): $\rho = 1$, $\mathbf{B}_0 = (1, \sqrt{2}, 0.5) \Rightarrow B_0 = \sqrt{13}/2$. This gives a value for the Alfvén velocity of $v_A \approx 0.5086$. The next thing to do is to see whether the wave gives us this value through the dispersion relationship above.



Figure 4: Plots of different properties of the fluid as a linear wave passes through

I wrote a small Python script to calculate and compare the simulation and theoretical Alfvén velocities. The angular frequency was calculated by measuring the (simulated) time it took for a peak of the wave to travel through the box, as it has periodic boundary conditions, and dividing this by 2π . The magnitude of the wavevector was gotten by dividing 2π by the wavelength. All quantities were eyeballed as only a quick check was needed. The result is seen in Fig. 5, showing the dispersion relationship holds in the simulation.

```
print("Alfven speed:", v_A) Alfven speed: 0.5085536181410275
print("Calculated speed:", v_pred) Calculated speed: 0.5045045045045046
print("Ratio:", v_pred / v_A) Ratio: 0.9920379808695018
```

Figure 5: Output of the Python script I wrote

Questions (answered):

- Is \mathbf{k} automatically parallel to \mathbf{B} when the `ang_3_vert` variable is set to `True`?

No, it just rotates the axes such that \mathbf{k} is along the y-axis. Can make it parallel to \mathbf{B} through code.

- Why are the density and pressure constant for an Alfvén wave?

Because the Alfvén wave is a **shear wave** i.e. no compression at all. This means $\nabla \cdot \mathbf{u} = 0 \implies D\rho/Dt = 0$ i.e. the density is constant in time and thus so is the pressure.

- Why is a rectangle used instead of a square for the region of interest?

Just to make the numbers work out right such that the waves are continuous over the periodic boundaries. Can make it a square if needed.

The next step was simulating turbulence in the case of MHD. Essentially the same code was used as the hydrodynamic turbulence case, with a couple of updates to allow for the initialisation of a mean magnetic field in the region of interest.

In order to get critical balance, we want $u_{\perp} \sim v_{A0} = B_0/\sqrt{4\pi\rho}$. Athena sets $4\pi \rightarrow 1$, so we get $v_{A0} = B_0/\sqrt{\rho}$. We can set $B_0 = 1$ and $\rho = 1$ such that $u_{\perp} \sim v_{A0} = 1$. To do this in the simulation I've set $dE/dt = 1$ as this means $u_{\perp}^3/L \sim 1$ (the box size has $L = 1$; letting everything work out to 1 makes it simpler).

Forgot to add the magnetic field flag so have to redo larger simulations. **A good check is to look at the first few snapshots in VisIt to see if the magnetic fields (or other parameters) are set up the way I wanted.**

Trying out Athena's Python codes for plotting and reading data files to help write the `spectrum.m` code in Python. Haven't made much progress yet as I've been focusing on writing the `structure_function.py` code.

Originally wrote `structure_function.py` using just lists. For 1 million random pairs of points on a 128^3 continuously forced turbulence grid (non-MHD), the average run time was about 100 seconds. After learning about `numpy` and utilizing this, the same problem had its run time reduced to about 20 seconds; this is a massive improvement. Not sure whether I need to test for a larger number of points. Below in Fig. 6 I've added the initial output of the function; showing the distribution of distances between the points and the velocity structure function at $t = 0.1$ s.

I've just realised that it doesn't look right in Fig. 6 as I've used the snapshot in the first 0.1 second after the simulation has begun, so the turbulence will not have fully developed. Will look into more detail next week.

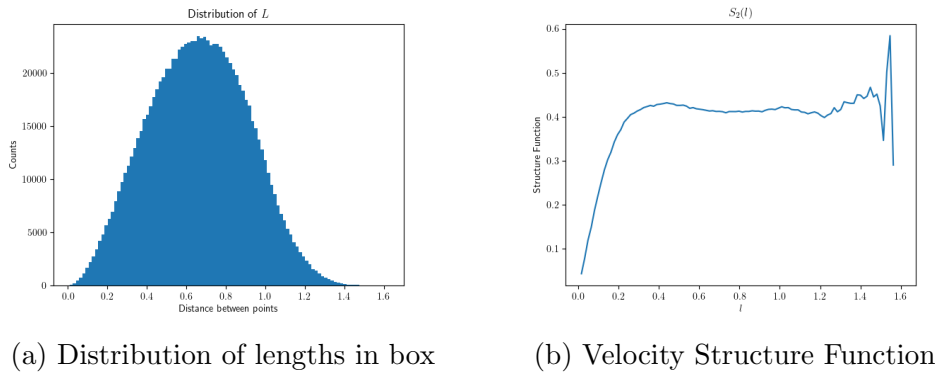


Figure 6: Output of the structure function code for the hydrodynamic 128^3 forced turbulence simulation

Week 4: 2 Dec - 6 Dec

After talking with Jono about the structure function code I wrote last week, I updated it such that it generates a log spaced l -grid and generates the random points closer together in order to be able to see the finer details in the small l range (inertial subrange). After updating the code the l distribution and structure functions are shown in Fig. 7. The lengths between the points are now skewed to be smaller which is what was wanted, and the structure function is very close to following Kolmogorov's $l^{2/3}$ rule.

I've created a [GitHub repo](#) to hold the Python code that calculates these here.

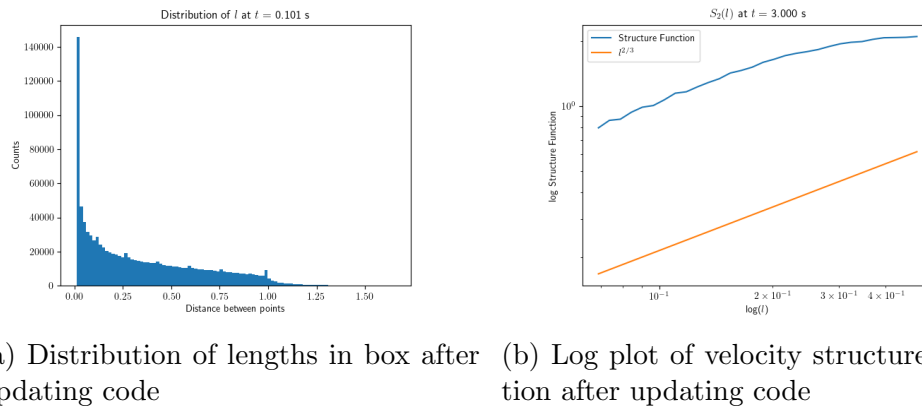
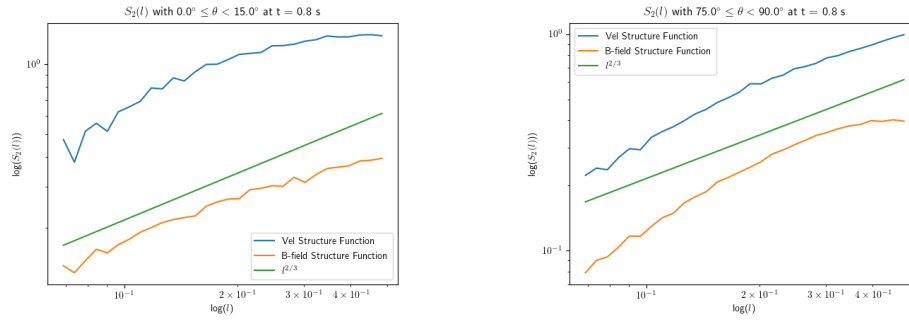


Figure 7: Output of the structure function code for the hydrodynamic 128^3 forced turbulence simulation

Updated the code to calculate the l_{\parallel} and l_{\perp} components of the separation vector relative to the magnetic field to observe the critical balance law after a struggle getting it to work. The issue was that selecting the l values based on their angle with the magnetic field was

messing up their order, as they were moved to a new array but the code assumed an index correspondence between the velocity data and the l values. Fixed this by using logical masks on both arrays at once to keep the order when performing the MHD calculations. Fig. 8 shows the output of this code on the same 128^3 simulation as above but with magnetic fields enabled. They are potentially not correct as the parallel components in Fig. 8a don't seem to fall off quicker than the perpendicular components. Will check next week.



(a) Velocity and magnetic structure functions of parallel components (b) Velocity and magnetic structure functions of perpendicular components

Figure 8: Output of the structure function code for the MHD 128^3 forced turbulence simulation

Animations of the full structure functions can be found [here](#).

Week 5: 9 Dec - 13 Dec

The previous four weeks I learnt the basics of turbulence and critical balance in fluids and plasmas, and ran and explored some basic hydrodynamic and MHD simulations using Athena to get an idea of what the body of my project will be like. The aim of my project over the summer is the following:

Project Aim: To look at magneto-immutable turbulence in weakly collisional plasmas when $|\Delta p| \sim B^2$ in the Braginskii and collisionless regimes, and to potentially use the results of this to try gain insight into properties of the solar wind. Want to at least be able to reproduce the data in Figures 3-5 in Jono's paper on magneto-immutable turbulence.

The first step is to run the CGL sim at at high collision frequency and compare to the same sim using an adiabatic EOS, as this regime of collision frequency gives back MHD with an adiabatic EOS (allows particle velocities and thus pressure to reach an equilibrium).

The input file had a $2 \times 1 \times 1$ box (at a $64 \times 32 \times 32$ resolution) with a magnetic field of unit strength along the x -axis. Ran these simulations using the HLLE solver and got out a [CGL structure function](#) and [adiabatic MHD structure function](#). Note that $\beta = \frac{8\pi p}{B^2} = 1$ in this case. These animations look so similar that I thought I had ran the same simulation twice, but there are subtle differences. This is a good sign as it does output what the theory predicts. perfect

The next step is to try and observe critical balance in the CGL case, and then change the β value to 100. This will run about 10 times as slow but should give similar results. After that, will lower the collision frequency and do similar analysis.

The majority of this week was spent working on refining the structure function code and checking it on the results of the simulation above. I also added in some code to show the RMS fluctuations of the velocity and magnetic field over time.

Week 6: 16 Dec - 20 Dec

This week I ran a CGL simulation as in Week 5 but with a $128 \times 64 \times 64$ resolution. I played around with the energy input (dE/dt) at first as it needed to be set such that critical balance could be observed. The box has $L_{\perp}/L_{\parallel} = 0.5$ so in order to have critical balance we need the amplitude of the perpendicular fluctuations of velocity and magnetic field to be about 0.5. Setting $dE/dt = 0.5$ allowed this to happen, as seen in Fig. 9.

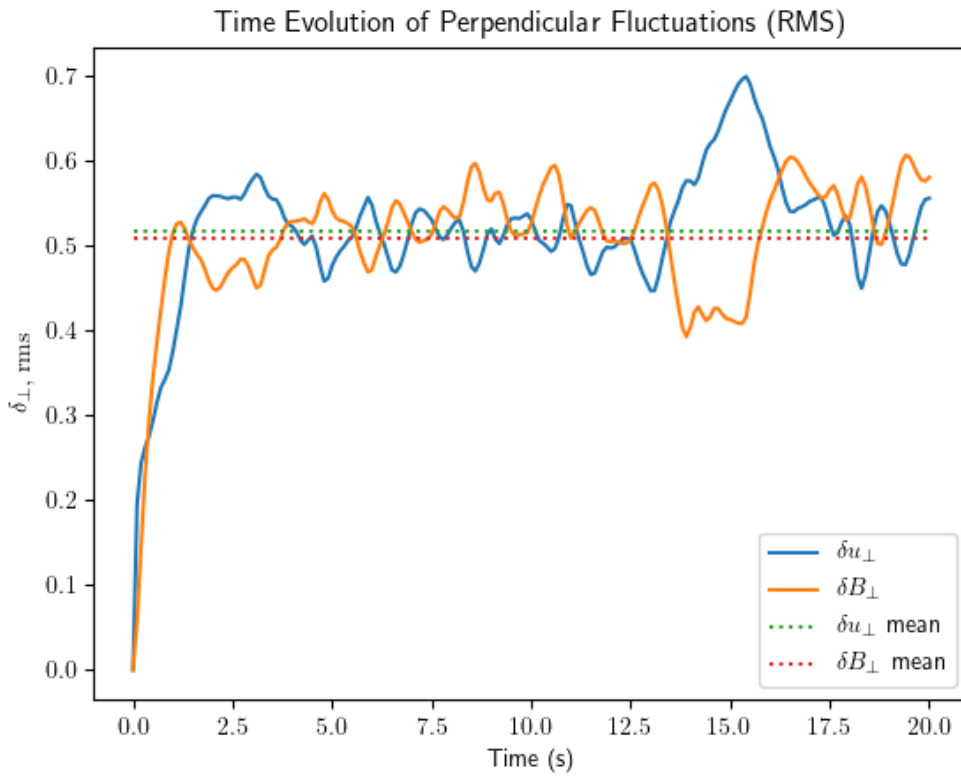


Figure 9: Perpendicular fluctuations of the simulation described above.

There is a correlation between the velocity and magnetic field fluctuations. This is because Alfvén waves are playing an important role in the structure of turbulence, and this causes a strong correlation when it is randomly forced.

Next steps to do when I get back are to run similar simulations at $\beta = 100$ and then simulations with the collisional frequency lowered and to try get back the results in Jono's magneto-immutable turbulence paper. I will try to work on the spectrum code over Christmas. Will also need to add some new diagnostics such as the PDF on page 11 of the magneto-immutable turbulence paper, as well as organising output files better. Will also have a chat with Jono about what he knows about the Sun as I've found that really interesting.

Week 7: 20 Jan - 24 Jan

Started off the week reading over the theory needed to bring myself back up to speed after the Christmas break, mainly the amplitude limits and magneto-immutable papers. Information slowly making sense as getting used to reading how papers are written.

Will continue running simulations with decreasing collision frequency. Have increased pressure by a factor of 100. Sound speed depends on pressure so increased pressure means increased sound speed. Athena compensates by lowering the time step meaning simulation takes longer to run. Jono will show me how to use NeSI on Friday to hopefully speed things up.

Am still working on spectrum code, need to buckle down and get it done. Two ways to calculate the PDF graph:

- Calculate $\hat{\mathbf{b}}\hat{\mathbf{b}} : \nabla \mathbf{u} = b_i b_j \partial_i u_j$ explicitly. Gradient of \mathbf{u} a little hard to do numerically so can Fourier transform $(u_x, u_y, u_z) \rightarrow (K_x, K_y, K_z)$ and calculate $\partial_i u_j = \text{ifft}(K_i \text{fft}(u_j))$ and then plugging into sum. Computationally expensive but accurate.
- Rearrange Equation 2.4 in paper to get $\hat{\mathbf{b}}\hat{\mathbf{b}} : \nabla \mathbf{u} = \Delta p / \mu_{\text{Brag}}$ where $\mu_{\text{Brag}} = p_0 / \nu_c$. Can use in Athena as ν_c, p_0 are parameters and Δp is output. Couldn't be used in SNOOPY as Δp was given by Equation 2.4 there.

Then histogram. Can use these two methods as a check when in the Braginskii regime:

$$\mu_{\text{Brag}} \gg 1 \text{ and } \omega_A \ll \nu_c \implies p_0 \gg 1$$

Panicked when I saw that I was getting $\Delta p = 0$ for all time in my simulations, then realised I wasn't actually in the Braginskii regime at all (was still in the MHD regime). Ran another simulation with properties that should be in the right regime.

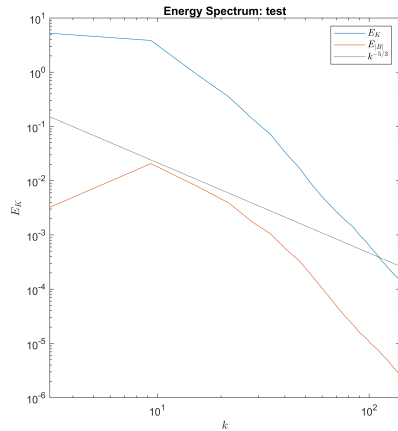
Want to be able to output ν_c from Athena code just to make it easier to calculate stuff.

Week 8: 27 Jan - 31 Jan

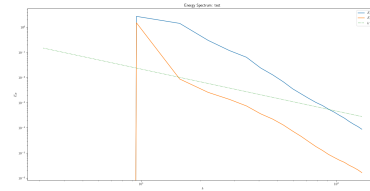
Started off with running some Athena simulations on NeSI to get a hang of how long I should be expecting to wait. Need to write PDF and spectrum code.

Rewrote the PDF code, and ended up getting the Fourier transform method giving a closer result to the paper than straight calculating it from Δp . It does look a little too symmetrical however so will investigate and put screenshots up later.

Got the spectrum code mostly working, except for something to do with normalisation at small k .



(a) Spectrum calculated using MATLAB



(b) Spectrum calculated using Python

Ran more simulations on NeSI, didn't know you could do more than one at once. Changed the following for the $2 \times 1 \times 1$ box using 40 CPUs on one node:

- **Resolution:** $320 \times 160 \times 160$
- **Meshblock:** $64 \times 80 \times 40$
- **Simulated Time:** 20 seconds down to 12 seconds. This is because we only need to run for 5-10 Alfvén travel times τ . In this case, $\tau = 2\pi/\omega_A = 2$ (as $\omega_A = 2$) so 12 simulated seconds is a good length.
- $p_0 = 10$
- ν_c : using range of 1, 4, 10.

Was confused about which parameters to use; Jono simplified it down considerably. Want to only vary **dimensionless parameters** as this reduces the risk of running a simulation done before (e.g. halving a dimensional parameter could double another causing a null result).

Will vary using the dimensionless parameter:

$$\text{It}_{\text{Brag}} \sim \frac{\rho v_A l_{\parallel}}{\mu_{\text{Brag}}} \left(\frac{\delta B_{\perp}}{B_0} \right)^{-2}$$

The term $\left(\frac{\delta B_{\perp}}{B_0} \right)^{-2}$ comes out to 4 as the fraction is 1/2; ρ and v_A are equal to 1 and l_{\parallel} is equal to 2. Using $\mu_{\text{Brag}} = p_0/\nu_c$ we get:

$$\text{It}_{\text{Brag}} \sim \frac{8\nu_c}{p_0}$$

Will keep $p_0 = 10$ and vary ν_c as above.

Talked with Jono about honours project, think I will keep going in this general direction of turbulence in plasmas but could change focus if needed further on. Let Jono know that I would like to do some more analytical stuff; he said he'll try make that a factor of whatever project I'll choose. Alo says it'd be good to write a report (why this physics is interesting, methods of running/calculating simulations, results etc) after I've had a play around with data from NeSI.

Gotten spectrum code pretty much working except for a few little bugs. PDF requires a little more work. Will iron out these bugs and add a It_{Brag} parameter/calculator in the code.