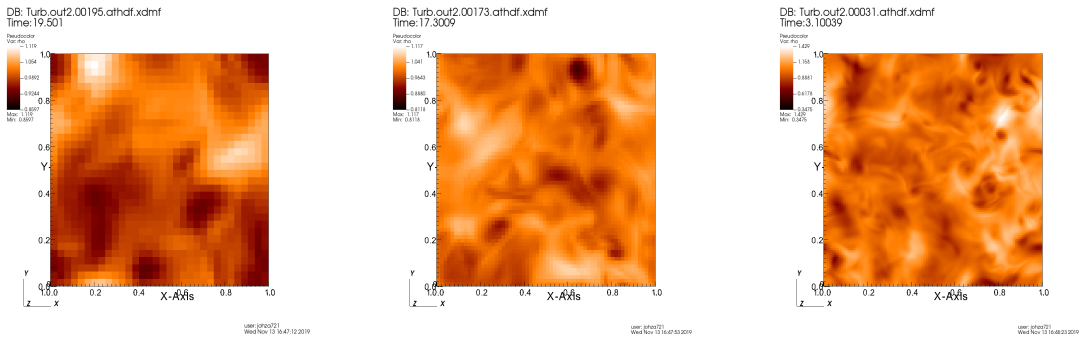
Turbulence in Plasma

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.



(a) Grid Size: 32 (b) Grid Size: 64 (c) Grid Size: 128

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.

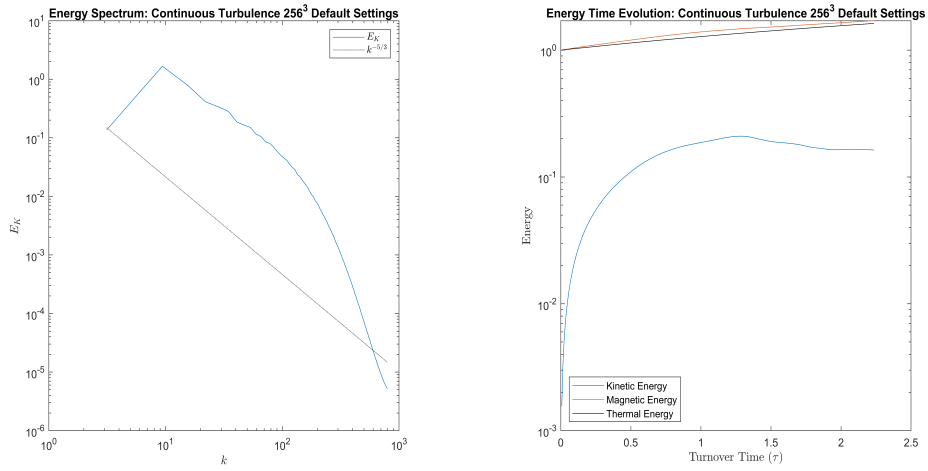
Week 2: 18 Nov - 22 Nov

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 continously 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.



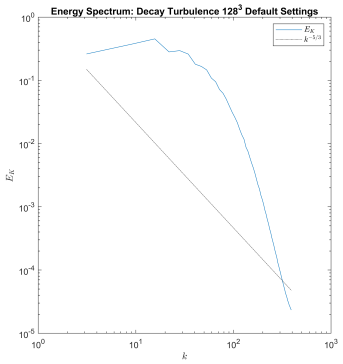
(a) Averaged Spectrum (b) Energy Evolution

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

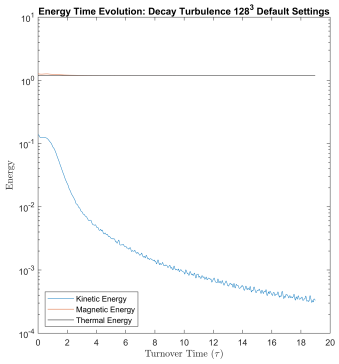
Decay: The energy spectrum (Fig 3a), averaged over the first two turnover cycles, is not

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.

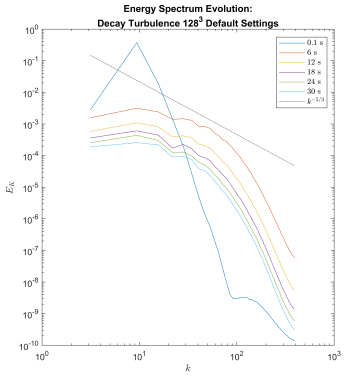
For the spectrum evolution (Figure 3c) I took snapshots of the spectrum at 3 second intervals, with a snapshot at 0.1 seconds to observe the energy input (the curve sharply peaked at $k \sim 10$). We see that the energy cascades down through the length scales directly after input (at 6 seconds), and 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.

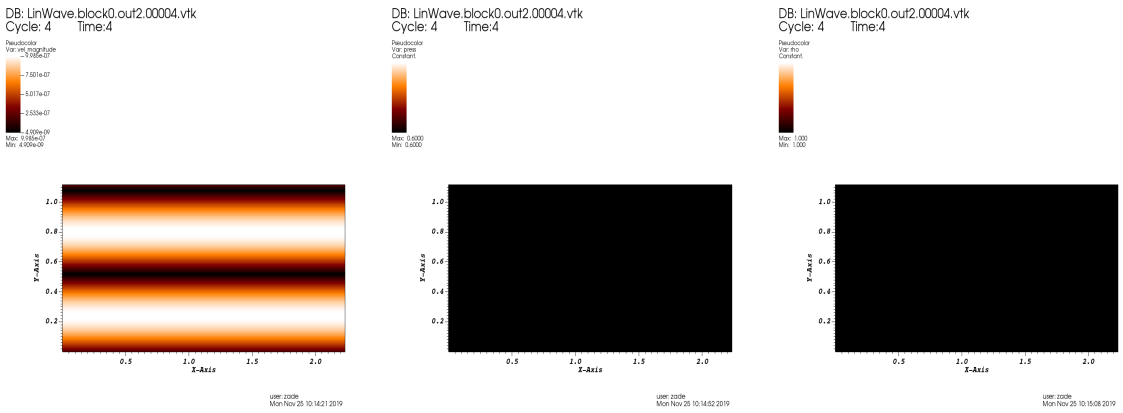
Week 3: 25 Nov - 29 Nov

Ran the linear wave problem in Athena to observe Alfv́en waves. The pressure and density of the plasma should be constant as these waves travel, which was observed in the simulation (Fig. 4). Alfv́en 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 Alfv́en 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): $\rho = 1$, $\mathbf{B}_0 = (1, \sqrt{2}, 0.5) \implies B_0 = \sqrt{13}/2$. This gives a value for the Alfv́en 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.



(a) Velocity Magnitude (b) Pressure (c) Density

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 Alfv́en 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 **k** automatically parallel to **B** when the `ang_3_vert` variable is set to `True`?

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

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

  Because the Alfv́en 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 insted 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. Essentialy 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.

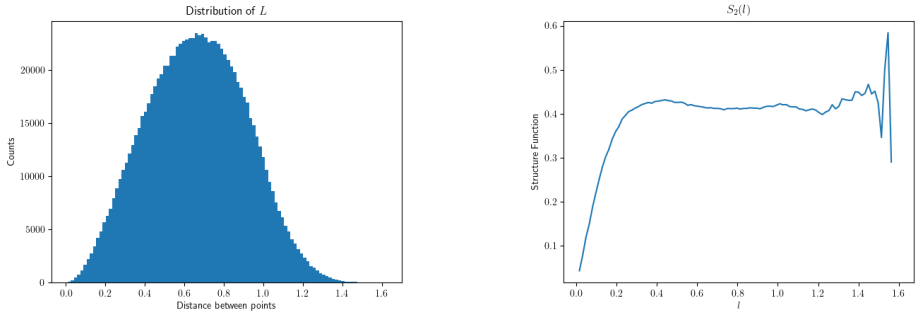
In order to get critical balance, we want $u_\perp \sim v_{A0} = B_0/\sqrt{4\pi\rho}$. Athena sets $4\pi \to 1$, so we get $v_{A0} = B_0/\sqrt{\rho}$. We can set $B_0 = 1$ and $\rho = 1$ such that $u_\perp \sim v_{A0} = 1$. To do this in the simulation I've set $dE/dt = 1$ as this means $u_\perp^3/L \sim 1$ (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 inital 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.



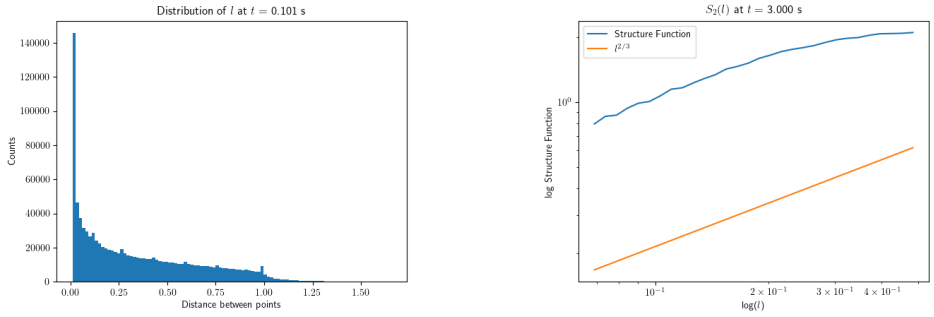
(a) Distribution of lengths in box (b) Velocity Structure Function

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 we 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.

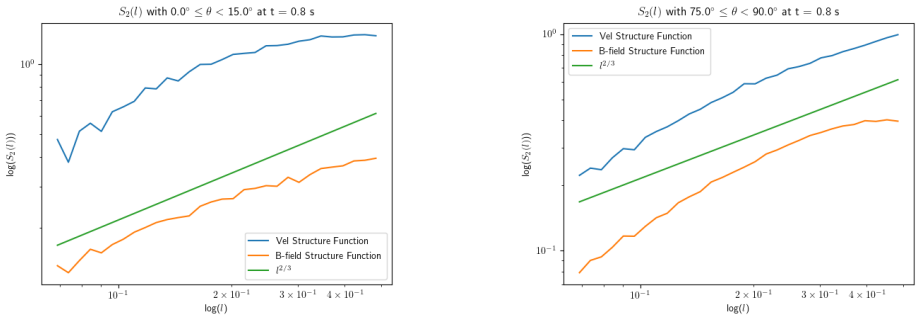


(a) Distribution of lengths in box after updating code (b) Log plot of velocity structure function after updating code

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 (not 100% sure at the moment):

Project Aim: To look at magneto-immutable turbulence in weakly collisonal plasmas when $|\Delta p| \sim B^2$ using the Braginskii approximation and in different regimes, and to potentially use the results of this to try gain insight into properties of the solar wind?

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

The input file had a $64 \times 32 \times 32$ grid box 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.

The next step is to try and obsereve 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.

Theory

Kolmogorov energy cascade law: Energy that is put into a turbulent fluid at the largest relevant scale doesn't immediately dissipate as heat. Instead it is cascaded down the length scales of the fluid (from macro to micro) in the formation of increasingly smaller eddies, and is dissipated as heat at the microscopic scale due to viscous effects. This produces a kind of order out of the (seemingly) chaos of turbulence.

For a given eddy length scale l, we can find a relation between the wavenumber ($k = 2\pi/l$) and the energy contained in the fluid $E(k)$. This theory was developed by Kolmogorov and says $E(k) \sim k^{-5/3}$ for a given range of wavenumbers, called the **inertial subrange**. This law fails at small scales (large k) for the reason explained above, and at large scales (small k) as this is where the energy is being injected into the system.

Derivation using dimensional analysis: The Navier-Stokes equation is

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u}$$

Turbulence arises from the nonlinear term $(\mathbf{u} \cdot \nabla)\mathbf{u}$ (represents the energy cascade), and energy is dissipated via the viscosity term $\nu\nabla^2\mathbf{u}$. Using dimensional analysis, at a given scale l with bulk velocity u_l we have $|(\mathbf{u} \cdot \nabla)\mathbf{u}| \sim u_l^2/l$ and $|\nu\nabla^2\mathbf{u}| \sim \nu u_l/l^2$ (using $\nabla \sim 1/l$). Reynolds number Re is the ratio of the turbulence term to the viscosity term:

$$\mathrm{Re} \sim \frac{|(\mathbf{u} \cdot \nabla)\mathbf{u}|}{|\nu\nabla^2\mathbf{u}|} = \frac{u_l l}{\nu}$$

The viscous term is much smaller than the nonlinear term for large l (due to the factor of $1/l^2$). This means that at large scales energy can't dissipate as heat, so the nonlinear term effectively cascades energy through eddies as it has nowhere else to go. Only when the two terms are comparable ($\mathrm{Re} \sim 1$) can the energy be dissipated as heat. This also explains the self-similarity in the energy spectrum across the inertial subrange.

For a given length scale l and velocity u_l (this is the **structure function**), the energy is $E \sim u_l^2$. The turn-over time for an eddy is $\tau \sim l/u_l$. The rate of energy dissipation is then $E/\tau \sim u_l^3/l = \text{const.}$ across the inertial subrange due to self-similarity.

We then have $u_l \sim l^{1/3} \sim k^{-1/3} \equiv u_k$. The energy in all eddies with wavenumber $\geq k$ is given by

$$E = u_k^2 \sim k^{-2/3} \sim \int_k^\infty E(\kappa)\, d\kappa \implies E(k) \sim k^{-5/3}$$

giving us Kolmogorov's result.

The structure function $\langle [u_l]^2 \rangle = \langle [u(x+l) - u(x)]^2 \rangle$ measures how smooth the velocity distribution is for a given length scale. It is small at large l as velocity is continous at that scale, and large at small l as velocity can change appreciatively over that scale.

MHD Equations: Using $D/Dt = \partial/\partial t + \mathbf{u} \cdot \nabla$ as the Lagrangian derivative.

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{u} \tag{1}$$

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla \left(c_s^2 \rho + \frac{B^2}{8\pi} \right) + \frac{(\mathbf{B} \cdot \nabla)\mathbf{B}}{4\pi} + \nu \nabla^2 \mathbf{u} \tag{2}$$

$$\frac{D\mathbf{B}}{Dt} = (\mathbf{B} \cdot \nabla)\mathbf{u} - (\nabla \cdot \mathbf{u})\mathbf{B} + \eta \nabla^2 \mathbf{B} \tag{3}$$

Equation 1 is the mass continuity equation (what comes in must go out).

Equation 2 gives the time evolution of the fluid due to forces, and is obtained from the Navier-Stokes equation:

- $-\nabla \left(c_s^2 \rho + \frac{B^2}{8\pi} \right)$: Says that the higher the mass and magnetic field density, the greater the force is on the fluid in a direction away from that density.
- $\frac{\mathbf{B} \cdot \nabla \mathbf{B}}{4\pi}$: acts as a "tension" on the magnetic field lines; i.e. if the magnetic field lines are perturbed it acts to restore to its inital condition.
- $\nu \nabla^2 \mathbf{u}$: usual viscosity term in the fluid.

Equation 3 determines the evolution of the magnetic field. Looks similar to the mass continuity equation, and is obtained from manipulating the Lorentz force equation and Maxwell's equations. Often called the "frozen in" condition as field lines move with the fluid:

- $(\mathbf{B} \cdot \nabla)\mathbf{u}$ and $(\nabla \cdot \mathbf{u})\mathbf{B}$: acts to stretch the field lines with the fluid. The first is stretching due to shear flows while the second is stretching due to compressible flows (not as important as we usually have $\nabla \cdot \mathbf{u} = 0$).
- $\eta \nabla^2 \mathbf{B}$: The "viscosity" term of the magnetic field; η is the resistance of the charged fluid. Gives a similar spectrum for the energy in the magnetic field as the one for the kinetic energy.

Critical Balance: For turbulence in MHD, critical balance is the assumption that

$$k_\perp u_{\perp k} \sim k_\parallel v_A$$

across all scales in the inertial range, where $k_\perp$ and $k_\parallel$ are the components of $\mathbf{k}$ perpendicular and parallel to the magnetic field. Unlike in hydrodynamic turbulence, eddies in MHD spread out along the magnetic field lines as energy is cascaded down.

Alfv́en waves are shear waves (i.e. doesn't compress fluid $\iff \nabla \cdot \mathbf{u} = 0$) that travel through a charged fluid along magnetic field lines. Alfv́en 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 Alfv́en velocity at which the waves move, where B_0 and ρ are the magnetic field strength and fluid density. Because they travel along the magnetic field lines, viscosity doesn't affect them as much as it does sonic waves because the particles can't "talk" to each other. This means that smaller scale structures that would otherwise have been lost due to viscosity survive; this is what allowed finer details in the solar wind to be observed near Earth.

When simulating turbulence in MHD we choose a box whose size is small compared to the length scales of the forcing. We can consider the "forcing" on this box to be the energy coming from the energy cascade at larger scales. We align the box such that the longest edge is parallel to the magnetic field lines. Call this length L_x, and L_y the length of the side perpendicular to the magnetic fields. Then:

$$k_\parallel = \frac{2\pi}{L_x},\ k_\perp = \frac{2\pi}{L_y}$$

In order to respect the assumption of critical balance, these lengths must satisfy $L_y/L_x \sim u_{\perp k}/v_A$.

Structure Function Code: A quick overview of how the structure function code works:

- Read in data from Athena output file, get grid, position, velocity, **B**-field etc. information.
- Generate a given number of pairs of random points in the grid. Modified such that the pairs of points are closer together to observe finer detail.
- Translate to position vectors and find the distance l between each pair of points.
- Get the velocity information at each pair of points and calculate $\Delta u^2 \equiv |\mathbf{u}_1 - \mathbf{u}_2|^2$. Can do the same with the magnetic field if using MHD.
- Create a grid of l values to bin and for a given range of ls average the corresponding Δu^2. We now have a relationship between l and $\langle \Delta u^2 \rangle$; this is the structure function.

The direction doesn't matter as the turbulence is isotropic, so we always see the structure function following $l^{2/3}$. When doing an MHD simulation, the following additional steps are done:

- For each pair of points, find the angle between $\mathbf{l}$ and $\mathbf{B}_{\text{mean}}$, where $\mathbf{B}_{\text{mean}} = \frac{1}{2}[\mathbf{B}_1 + \mathbf{B}_2]$, using
$$\cos\theta = \hat{\mathbf{l}} \cdot \hat{\mathbf{B}}_{\text{mean}}$$
- Bin the corresponding l values against a range of θ values. The $l_\parallel$ values are in the range $0^\circ \leq \theta < 15^\circ$ and the $l_\perp$ values are in the range $45^\circ \leq \theta < 90^\circ$. Plot the velocity and magnetic field structure function for each angle bin. The structure function for the parallel distances should fall off faster.

The parallel and perpendicular structure functions differ now as the magnetic field introduces an anisotropy in the velocity.

Braginskii (not sure of proper name of theory): In a normal fluid, the temperature T is proportional to $\langle (v - \langle v \rangle)^2 \rangle$, and the pressure ($p = \rho T$) is proportional to temperature. The temperature is isotropic (?) in this case as collisions in the fluid average out the velocity components.

For a plasma with magnetic field lines threaded through, we can't define a single temperature/pressure now as particles have velocity components parallel and perpendicular to the field lines. As the collision frequency ν_c is generally low, these components can't equalise. We need to define a parallel and perpendicular pressure, $p_\parallel$ and $p_\perp$, to close the fluid's equations of motion.

In the high collision regime ($\nu_c \gg 1$) we should get back the MHD equations as the velocities can now equalise.