Project Problem 4

Garrett King and Katie Schram

April 2018

This week, the main focus was working on the code and trying to get it to work. The work done on the simulation so far in included in the .py file in the GitHub repository and includes comments in line trying to explain what the code is looking to accomplish in each step. The point where things seem to break down is in the Euler Step method. At the moment, the code is running smoothly for times less than about 1 s and then sees the kinetic energy of the plasma take off toward infinity at any time after that. Something in the time updating must be going wrong, as print statements used as bug checks for all of the earlier steps showed that these are working as expected. To troubleshoot this problem in the code, the vx and vy arrays were printed at times after the bug occurs. The values rise very quickly and go to nan almost immediately after the 0.1 s mark. Looking at the Euler step, there does not appear to be anything wrong with the physics as we understand it now. This could indicate one of two things- a typographical mistake made in the code that is messing up the physics seen in the output or a lack of understanding of the problem preventing us from finding the mistake. We'll need to keep digging into this step of the code if we want to move forward.
Figure 1 shows the code working properly prior to it jumping off to 10e288 after running it for more than 1 s in simulation time. At the moment, it does not make sense why this wants to jump all of a sudden, as the kinetic energy is on its way back down. The output array for the velocity returns a whole array containing only nan in every entry, except at the boundaries. The Runtime warnings claim that the code is encountering invalid values in double scalars. The same grid spacing and time spacing is being used for the derivatives in every instance, so it should not be dividing by anything small. Having b or v take off to huge values just from the derivatives could be the problem, but looking at the code right now, it is not clear what is making this happen. We do have plots showing how v and b are evolving in time, which might provide insight into what the problem is when comparing with what others observe.

To figure out what could be going wrong in coding the problem, the following resource was consulted: `https://arxiv.org/pdf/1306.6883.pdf`. In this paper, the use of a python teaching code for undergraduate and graduate students is detailed. From what was read in this paper, it seems that our approach to the numerics is the same. They also use central finite differences to

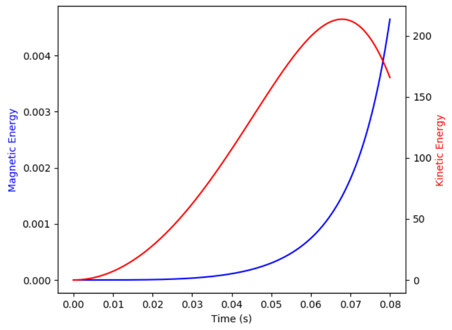

Figure 1: Evidence that the code works up to a certain time point before the bug.

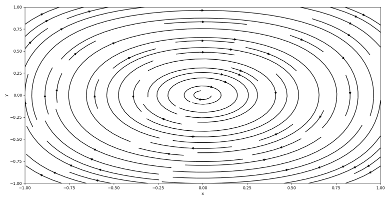

Figure 2: Stream plot of the initial velocity, which has the appearance of a TG vortex, indicating that it was set up properly.

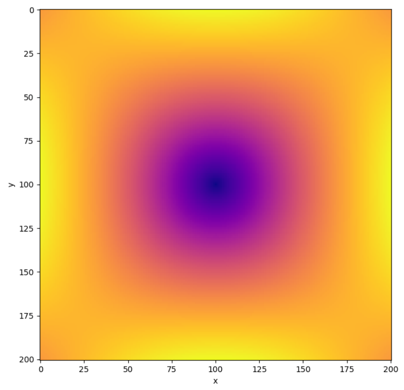

Figure 3: Heat map of the initial magnitude of the magnetic field at different points in the plane.

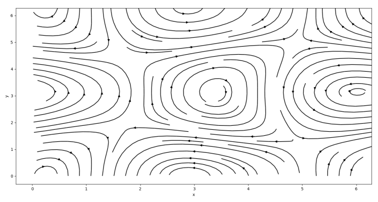
Figure 4: The velocity at a later time in a working run (t=0.35s).

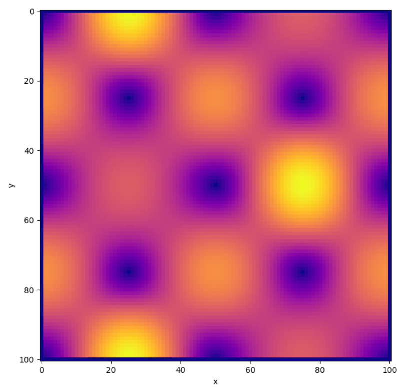

Figure 5: The magnetic field at a later time in a working run (t=0.35s).

calculate the derivatives with respect to position and update time with a Euler step. The actual physics was checked against `https://doi.org/10.1017/S002237780001638X`, which talks about the problem of reduced MHD in 2D sheets. Hopefully further study of these and the previous resources might give insight into the problem.

This week, Garrett did a majority of the coding and consulting references to make sure the parameters of the plasma were reasonable and that the simulation was not being hampered by unreasonable physics. After some more in depth reading, it is clear that the initial conditions are very important to this problem, as the equations are highly non-linear. Making sure the starting point isn't unreasonable is one task connecting this work and the code. For instance, in Nore et al., they found the flow was highly non-linear when the square amplitude of the magnetic field in Alfven velocity units was 10^{-4} times the velocity. Understanding the physics seems like it will be an important task going into the next week, as a good understanding of the physics is crucial to a functional code. Katie worked primarily on finding resources of where this simulation has been used before and providing resources for the code, since she is already familiar with the problem. She directed Garrett to the pyro paper to help in the coding process.