Dedalus: An Open-Source Spectral Magnetohydrodynamics Code

Keaton Burns

University of California Berkeley

Jeff Oishi

SLAC National Accelerator Laboratory

Received _____; accepted _____

ABSTRACT

We developed the magnetohydrodynamic (MHD) capabilities of Dedalus, an open-source Python-based hydrodynamics simulation, to explore and compare the properties of the standard and helical magnetorotational instabilities. Dedalus is a spectral code that uses external FFT libraries and parallelization with the goal of achieving speeds competitive with codes implemented in lower-level languages. This paper will outline the MHD equations as implemented in Dedalus, the methods taken to improve the performance of the code, and the initial results of our simulations.

Contents

1. Introduction

The magnetorotational instability (MRI) is a fluid instability that causes the onset of turbulence in magnetized disks and transports angular momentum outwards. It is believed to be an important mechanism in the physics of accretion disks. While not believed to exist in astrophysical disks, a second instability caused by a helical magnetic field has also been discovered. This helical magnetorotational instability (HRMI) may be easier to produce in ongoing laboratory experiments using liquid sodium. The applicability of such experiments to the processes in astrophysical disks is the subject of our investigation.

2. Dedalus Development

Dedalus is an open-source spectral hydrodynamics code, written in Python 2.7. It was designed to be flexible and easy to use, with the FFTs handled by external libraries and/or parallelization to abate the performance penalties of using a high-level language. The code makes extensive use of object-oriented programming, facilitating the modular implementation of different domain representations and physics. In addition to the magnetohydrodynamics (MHD) physics described below, Dedalus is also currently being used to study linear cosmology and baryon acoustic oscillations -NEED REFERENCE-. Dedalus is currently hosted on a public bitbucket repository.

2.1. MHD Equations

We begin with the equations governing incompressible MHD: the incompressible Navier-Stokes equation with the Lorentz force and viscosity, the induction equation with

magnetic diffusivity, mass continuity, and Gauss's law for magnetism (in Gaussian units, and with ∂_t indicating $\frac{\partial}{\partial t}$):

$$\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{\nabla p}{\rho_0} + \frac{\mathbf{F_L}}{\rho_0} + \nu \nabla^2 \mathbf{u}, \tag{1}$$

$$\partial_t \mathbf{B} = \nabla \times (\mathbf{u} \times \mathbf{B}) + \eta \nabla^2 \mathbf{B}, \tag{2}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{3}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{4}$$

Expanding the Lorentz force as

$$\mathbf{F_L} = \frac{(\nabla \times \mathbf{B}) \times \mathbf{B}}{4\pi} = \frac{\mathbf{B} \cdot \nabla \mathbf{B}}{4\pi} - \frac{\nabla B^2}{8\pi}, \tag{5}$$

Eq. (1) becomes

$$\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{\nabla P_{tot}}{\rho_0} + \frac{\mathbf{B} \cdot \nabla \mathbf{B}}{4\pi \rho_0} + \nu \nabla^2 \mathbf{u}, \tag{6}$$

$$P_{tot} = p + \frac{B^2}{8\pi}. \tag{7}$$

2.2. Spectral Implementation

In a periodic domain, any sufficiently smooth field variable can be represented by its discrete Fourier decomposition on the grid. That is, for some function f, we can write

$$f(\mathbf{x}, t) = \sum_{\mathbf{k}} \hat{f}(\mathbf{k}, t) e^{i x_i k_i}, \tag{8}$$

$$\hat{f}(\mathbf{k}, t) = \sum_{\mathbf{x}} f(\mathbf{x}, t) e^{-i x_i k_i}. \tag{9}$$

In Fourier space, the spatial derivatives in the governing equations transform to trivial multiplications: $\nabla \xrightarrow{\text{FT}} i\mathbf{k}$. Pseudospectral codes, such as Dedalus, use Fast Fourier Transforms (FFTs) to compute these spatial derivatives and transform the problem into a system of ordinary differential equations, which are then temporally integrated in Fourier space.

Taking the Fourier transforms of Eqs. (5), (3), and (4) yields

$$\partial_t \hat{\mathbf{u}} = -\widehat{\mathbf{u} \cdot \nabla \mathbf{u}} - \frac{i\mathbf{k}\hat{P_{tot}}}{\rho_0} + \frac{\widehat{\mathbf{B} \cdot \nabla \mathbf{B}}}{4\pi\rho_0} - \nu k^2 \hat{\mathbf{u}}, \tag{10}$$

$$i\mathbf{k} \cdot \hat{\mathbf{u}} = 0, \tag{11}$$

$$i\mathbf{k} \cdot \hat{\mathbf{B}} = 0. \tag{12}$$

We then take the scalar product of $i\mathbf{k}$ with Eq. (10) to arrive at an expression for the total pressure:

$$\frac{\hat{P_{tot}}}{\rho_0} = \frac{i\mathbf{k} \cdot \widehat{\mathbf{u} \cdot \nabla \mathbf{u}}}{k^2} - \frac{i\mathbf{k} \cdot \widehat{\mathbf{B} \cdot \nabla \mathbf{B}}}{4\pi\rho_0 k^2}. \tag{13}$$

Using the identity $\nabla \times (\mathbf{A} \times \mathbf{B}) = \mathbf{A}(\nabla \cdot \mathbf{B}) - \mathbf{B}(\nabla \cdot \mathbf{A}) + (\mathbf{B} \cdot \nabla)\mathbf{A} - (\mathbf{A} \cdot \nabla)\mathbf{B}$ along with Eqs. (2), (3), and (4), we see

$$\partial_t \mathbf{B} = \mathbf{B} \cdot \nabla \mathbf{u} - \mathbf{u} \cdot \nabla \mathbf{B} + \eta \nabla^2 \mathbf{B}. \tag{14}$$

Taking the Fourier transform of Eq. (14), we arrive at the time-evolution equation for $\hat{\mathbf{B}}$:

$$\partial_t \hat{\mathbf{B}} = \widehat{\mathbf{B} \cdot \nabla \mathbf{u}} - \widehat{\mathbf{u} \cdot \nabla \mathbf{B}} - \eta k^2 \mathbf{B}. \tag{15}$$

The nonlinear terms ($\widehat{\mathbf{u} \cdot \nabla \mathbf{u}}$, $\widehat{\mathbf{u} \cdot \nabla \mathbf{B}}$, $\widehat{\mathbf{B} \cdot \nabla \mathbf{u}}$, and $\widehat{\mathbf{B} \cdot \nabla \mathbf{B}}$) in Eqs. (10) and (15) are computed in real space. To eliminate aliasing effects, we employ the Orszag 2/3 rule, zeroing any mode with a k component greater than or equal to 2/3 of the Nyquist wavenumber in that direction. This zeroing is done before every reverse Fourier transform, after every forward Fourier transform, and at each temporal evolution.

2.3. Temporal Integration

The ODEs produced by the physics modules, along with the initial conditions specified at the start of the simulation, form an initial value problem that is integrated using explicit

Runge-Kutta methods, specifically the second-order midpoint method. For simulations with viscosity and/or magnetic diffusivity, an integrating factor is used to evaluate the linear steps used to construct the Runge-Kutta stages. Consider Eq. (10) for a specified mode $\mathbf{k}$, with the non-viscous terms considered to be constant during an integration step:

$$\partial_t \hat{\mathbf{u}}(t) + \nu k^2 \hat{\mathbf{u}}(t) = \mathbf{RHS}. \tag{16}$$

This is an equation of the form $y'(x) + P(x)y = Q(x)$, which has the exact solution

$$y(x) = \frac{\int Q(x)M(x)dx}{M(x)}, \tag{17}$$

where $M(x) = e^{\int P(x)dx}$ is called the integrating factor. Hence we find the solution of Eq. (16) at time $t + dt$ to be

$$\hat{\mathbf{u}}(t+dt) = \Big[\hat{\mathbf{u}}(t) + \frac{\mathbf{RHS}}{\nu k^2}(e^{\nu k^2 dt} - 1)\Big] e^{-\nu k^2 dt}. \tag{18}$$

2.4. Shearing Box

To study the effect of the MRI, we perform local simulations where our domain represents a small part of an astrophysical disk. The domain is taken to be a co-rotating box, whose left edge is a distance r_0 from the axis of rotation, and whose length in each dimension is much less than this fiducial radius. In the co-rotating frame, we take the unit vector $\mathbf{e_x}$ in the outward radial direction, and the unit vector $\mathbf{e_z}$ along the axis of rotation. The box is rotating with an angular velocity $\mathbf{\Omega_0} = \Omega_0 \mathbf{e_z} = \Omega(r_0)\mathbf{e_z}$.

The radial dependence of angular velocity in a Keplerian disk, $\Omega(r) = \sqrt{GM} r^{-3/2}$, gives rise to a background linear shear flow in this domain: with the domain moving at the angular velocity of the left (inner) edge, the box will shear in the x direction with a velocity of $-\frac{3}{2}\Omega_0 x \mathbf{e_y}$, as shown in Fig. 1. This shear motivates the construction of a domain representation and a corresponding physics implementation to handle MHD in a box with an arbitrary local linear shear.

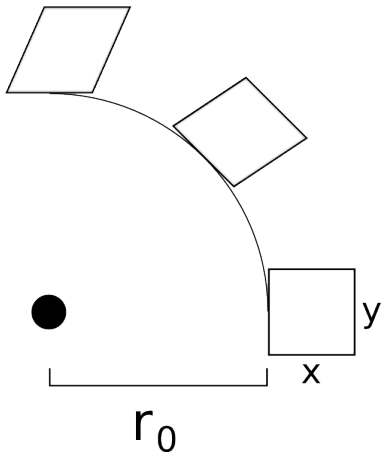

Fig. 1.— Co-rotating local domain as viewed from an inertial frame. Lower angular velocity at larger radii results in a local linear shear across the x direction of the local domain. The size of the box relative to r_0 is exaggerated for clarity.

Consider an arbitrary power-law shearing profile, $\Omega(r) = Cr^S$ (which arises from an attractive force of magnitude $\rho_0 C^2 r^{2S+1}$ and gives rise to a linear background shear in the local frame with velocity $S\Omega_0 x\mathbf{e_y}$. Hence $C = \sqrt{GM}$ and $S = -3/2$ for Keplerian rotation). In this case, the centrifugal ($-\mathbf{\Omega_0} \times (\mathbf{\Omega_0} \times \mathbf{r}) = {\Omega_0}^2 r\mathbf{e_x}$) and attractive ($-C^2 r^{2S+1}\mathbf{e_x}$) accelerations partially cancel (via approximations utilizing $x, y, z \ll r_0$):

$$\mathbf{a_{rad}} = ({\Omega_0}^2 - C^2 r^{2S}) r\mathbf{e_x} \approx -2S{\Omega_0}^2 x\mathbf{e_x}. \tag{19}$$

With this radial acceleration and the Coriolis acceleration ($-2\mathbf{\Omega_0} \times \mathbf{v}$), Eq. (1) for the velocity field in the rotating frame, $\mathbf{v}$, becomes

$$\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v} = -\frac{\nabla p}{\rho_0} + \frac{\mathbf{F_L}}{\rho_0} + \nu \nabla^2 \mathbf{v} - 2S{\Omega_0}^2 x\mathbf{e_x} - 2\mathbf{\Omega_0} \times \mathbf{v}. \tag{20}$$

Decomposing $\mathbf{v}$ into the background shear flow and velocity perturbations ($\mathbf{v} = S\Omega_0 x\mathbf{e_y} + \mathbf{u}$), Eq. (20) becomes

$$\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{\nabla p}{\rho_0} + \frac{\mathbf{F_L}}{\rho_0} + \nu \nabla^2 \mathbf{u} + 2\Omega_0 u_y \mathbf{e_x} - (2+S)\Omega_0 u_x \mathbf{e_y} - S\Omega_0 x \partial_y \mathbf{u}. \tag{21}$$

Due to the shear of our frame, the forward and reverse Fourier transforms become:

$$\hat{f}(\mathbf{k},t) = \sum_{\mathbf{x}} f(\mathbf{x},t) e^{-i(x_i k_i - S\Omega_0 x t k_y)}, \tag{22}$$

$$f(\mathbf{x},t) = \sum_{\mathbf{k}} \hat{f}(\mathbf{k},t) e^{i(x_i k_i - S\Omega_0 x t k_y)}, \tag{23}$$

to maintain periodicity along the shearing direction.

Hence, we define the fixed-grid wavevector $\mathbf{K}$ as a function of the Lagrangian (shearing) wavevector $\mathbf{k}$ and time, $\mathbf{K}(\mathbf{k},t) = (k_x - S\Omega_0 t k_y, k_y, k_z)$, and when transforming to Fourier space, the derivative operators become

$$\partial_x \rightarrow i(k_x - S\Omega_0 t k_y) = iK_x, \tag{24}$$

$$\partial_y \rightarrow ik_y = iK_y, \tag{25}$$

$$\partial_z \rightarrow ik_z = iK_z, \tag{26}$$

$$\partial_t \rightarrow \partial_t - iS\Omega_0 x k_y. \tag{27}$$

The analogs of Eqs. (10) and (13) are then

$$\partial_t \hat{\mathbf{u}} = -\widehat{\mathbf{u}\cdot\nabla\mathbf{u}} - \frac{i\mathbf{K}\hat{P_{tot}}}{\rho_0} + \frac{\widehat{\mathbf{B}\cdot\nabla\mathbf{B}}}{4\pi\rho_0} - \nu K^2\hat{\mathbf{u}} + 2\Omega_0\hat{u_y}\mathbf{e}_{\hat{\mathbf{x}}} - (2+S)\Omega_0\hat{u_x}\mathbf{e}_{\hat{\mathbf{y}}}, \tag{28}$$

$$\frac{\hat{P_{tot}}}{\rho_0} = \frac{i\mathbf{K}\cdot\hat{\mathbf{M}}}{K^2} - \frac{i\mathbf{K}\cdot\hat{\mathbf{L}}}{4\pi\rho_0 K^2} - \frac{2i\Omega_0\hat{u_y}K_x}{K^2} + \frac{(1+S)2i\Omega_0\hat{u_x}K_y}{K^2}. \tag{29}$$

Eq. (14) becomes, in velocity perturbations,

$$\partial_t\mathbf{B} = \mathbf{B}\cdot\nabla\mathbf{u} - \mathbf{u}\cdot\nabla\mathbf{B} + \eta\nabla^2\mathbf{B} + S\Omega_0 B_x\mathbf{e}_{\mathbf{y}} - S\Omega_0 x\partial_y\mathbf{B}, \tag{30}$$

and the analog of Eq. (15) becomes

$$\partial_t\hat{\mathbf{B}} = \widehat{\mathbf{B}\cdot\nabla\mathbf{u}} - \widehat{\mathbf{u}\cdot\nabla\mathbf{B}} - \eta K^2\mathbf{B} + S\Omega_0\hat{B_x}\mathbf{e}_{\hat{\mathbf{y}}}. \tag{31}$$

2.5. FFTs and Parallelization

Although Dedalus is written in Python, the FFTs dominate the computational cost of a simulation, meaning that optimizing the FFTs largely negates the performance penalties of using a high-level language, while maintaining ease of use and speed of development.

Outsourcing the FFTs from Python to FFTW, a C-based library that optimizes FFT routines based on local hardware, results in a substantial speed improvement over Python's (i.e. numpy's) built-in FFT algorithms. Much greater gains can be made on a GPU using Nvidia's CUDA architecture to compute the FFTs and other calculations.

Finally, MPI-based parallelization was implemented, allowing a single simulation to simultaneously run as many separate tasks. To achieve this, the computational domain is evenly divided among the N tasks along the k_z direction in Fourier space. The reverse Fourier transform is then accomplished in a series of steps, as depicted in Fig. 2. First, each task performs a 2D IFFT in the k_x and k_y directions on its dataset, and the shearing phase shift as applied as in Eq. (23), if necessary. Second, an MPI All-To-All call is issued, in which each task evenly divides its data N times in the y direction, and sends the Nth slab to the Nth task. Each task then stacks the N slabs it has received, and performs a 1D IFFT in the k_z direction. The resulting datasets have gone through a full 3D IFFT, and the data is evenly divided among the tasks along the y direction in real space. The parallelized forward transform is the reverse of this process.

3. Magnetorotational Instability Results

Note: simulations in progress. I hope to have results by the end of the week.

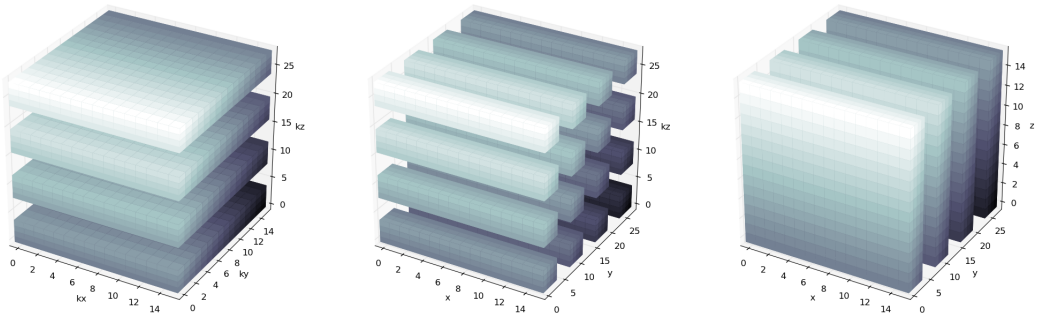

Fig. 2.— Task cuts along k_z in Fourier space, MPI All-To-All call, task cuts along y in real space.

4. Conclusion

Dedalus is among the first entirely open-source spectral MHD codes. The Python development environment facilitate ease of use and code development, and our emphasis on object-oriented techniques have helped make Dedalus a very modular code which can be easily adapted to study a variety of problems. The parallelized FFT algorithms, the use of external libraries, and the incorporation of GPU-based calculations using CUDA all contribute substantially to the performance of the code. With shearing-box MHD simulations underway, we hope to use Dedalus to help identify points of comparison and departure between current lab-based HMRI experiments and the standard MRI operating in astrophysical accretion disks.

5. References