CUDA-MHD: A GPU-accelerated Numerical Scheme for Magnetohydrodynamics

Brian Kubisiak and Yubo Su

May 12, 2016

1 Summary

Magnetohydrodynamics (MHD) is the study of fluid flow subject to changing magnetic fields. Its diverse range of applications range from laboratory experiments to solar phenomena to supernovae, the spectacular deaths of stars hundreds of times the size of our Sun. MHD is described by a set of highly complex nonlinear partial differential equations that require numerical simulations of high accuracy to capture all interesting phenomena, a computationally expensive task. Nevertheless, the numerical solution is highly parallelizable and is well-suited to acceleration via GPU computation.

2 Background

The ideal MHD equations with the assumption of magnetic permeability $\mu = 1$ consist of the following five conservation laws:

$$\vec{\nabla} \cdot \vec{B} = 0 \tag{1}$$

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot (\rho \vec{v}) = 0 \tag{2}$$

$$\frac{\partial (\rho \vec{v})}{\partial t} + \vec{\nabla} \cdot \left(\rho \vec{v} \vec{v} - \vec{B} \vec{B} \right) + \vec{\nabla} P^* = 0 \tag{3}$$

$$\frac{\partial \vec{B}}{\partial t} - \vec{\nabla} \times \left(\vec{v} \times \vec{B} \right) = 0 \tag{4}$$

$$\frac{\partial E}{\partial t} + \vec{\nabla} \cdot \left((E + P^*) \vec{v} - \vec{B} \left(\vec{B} \cdot \vec{v} \right) \right) = 0 \tag{5}$$

where ρ is the mass density, $\vec{v}$ is the velocity field (and $\rho\vec{v}$ carries interpretation momentum density), $\vec{B}$ is the magnetic field, E the total energy density and $P^* = P + B^2/2$ the total pressure where $P = (\gamma - 1)\left(E - \rho v^2/2 - B^2/2\right)$ is the gas pressure that satisfies the equation of state. The objective is then to propagate all of these variables forward in time from some initial state $\rho_0, (\rho\vec{v})_0, \vec{B}_0, \vec{E}_0, E_0, P_0^*$ while obeying the conservation laws.

To solve these equations, we refer to the watershed paper *Efficient magnetohydrodynamic simulations on graphics processing units with CUDA* (Wong et al. 2011) in which the ideal MHD equation were first implemented on the CUDA architecture. While CUDA has evolved greatly since then, the general techniques of the paper remain relevant. In the intended numerical scheme, the magnetic field is first held fixed while the fiuid variables are updated. An analogous procedure holding the fluid variables fixed and freeing the field is then performed to complete one time step. The detailed procedure whereby this evolution is performed to second-order accuracy can be found in the Wong et al. paper and will be the strategy we pursue.

3 Anticipated Challenges

The code will be immensely challenging simply due to the complexity of the full problem. Moreover, the highly nonlinear nature of the dynamics that give rise to the actually interesting MHD phenomena mandates numerical schemes of greater complexity and lower error than most simpler methods (e.g. Runge-Kutta is a lower bound on the required accuracy). However, the fact that it is a problem that has great interest in the scientific community means that there is ample literature on the preferred way to surmount these difficulties, at least computational obstacles.

There are also implementational difficulties that will arise when trying to use CUDA to speed up the computation. Since numerical MHD is linear in the number of timesteps, no speedup can be incurred unless all data are stored in GPU memory (otherwise, the simple cost of copying data will significantly decrease the performance gains of a GPU implementation). The logistics of this storage will be non-trivial to navigate. Additionally, the large number of variables and data that will be stored at each timestep present a considerable challenge to organize in a GPU-friendly way, and since the accuracy of a simulation scales with its resolution, a space-friendly algorithm will also need to be found (to maximize the resolution). The compromise between space-friendly resolution and time-friendly optimization will certainly be a hurdle during the implementation as well, one that will differ for our implementation on personal computers from those in literature run on supercomputers.

4 Goals

We will deliver a full codebase for numerical MHD simulation alongside a sample simulation in which a supernova of a star

with realistic initial conditions is simulated. The code will be capable of modelling any phenomena with appropriate choice of initial conditions.

5 Timeline

Week 1 (5/13–5/20) — We will complete literature reading and implement a CPU version of the desired MHD code. This implementation will be maximally modularized such that any GPU-accelerated functions can simply be substituted in "drag and drop" fashion.

Week 2 (5/20–5/27) — We will design the memory layout for the various variables in a GPU-friendly way. We will also begin considering how functions can be parallelized, a necessary part of finding an efficient memory layout scheme.

Week 3 (5/27–6/3) — We will implement the functions and run the simulation and obtain **FREAKING AWESOME RESULTS**.