1 Introduction

1.1 Motivation

Hydrodynamics forms an important part of ecological dynamics. Because of their small size and need for water, most microbes are forced to live in water. These types of water can very depending on the microbe, from cheese, to the ocean, to the insides of larger organisms.

Water can also affect the fitness of microbes. The movement of water controls access to important parameters such as nutrients, temperature and salinity. Mixing of water can ensure fresh supplies of poorly soluble oxygen or it could lead to being washed away from important food sources.

My specific inspiration comes from this paper [1]. In the paper, they simulate the fluid flows around a compact marine waste pellet, which can be used to simulate the diffusion of food coming off that particle. My goal was to replicate the fluid flow simulated in this paper by implementing a parallelized algorithm for the navier stokes equation with CUDA. Ideally, I'd be able to implement boundary conditions more complicated than the spherical conditions given in the paper.

2 Methodology

2.1 Physics

The description of the physics and the algorithm used for my project mostly follows [2].

2.1.1 Navier Stokes Equation

One of the most common ways to simulate hydrodynamics is with the navier-stokes equation. Because I'm working with water, I'll use the assumptions for incompressible flow. I'll also work in a regime of low turbulence to simplify calculations. And also work in two dimensions.

The equation is made up of three parameters, a pressure field $p(x,t)$ and a velocity field with an x $\underline{u_x(x,t)}$ and y $\underline{u_y(x,t)}$ component. The equation comes from the assumption of momentum balance

$$\frac{du}{dt} + (u \cdot \nabla)\, u = -\frac{1}{\rho}\nabla p + \frac{\eta}{\rho}\nabla^2 u + f \tag{1}$$

and mass balance

$$\nabla \cdot u = 0 \tag{2}$$

where ρ is the density of the fluid, η is the viscosity of the fluid, and f is an external force vector which could be gravity for example.

Because these are pdes, boundary conditions must be specified. There are three types of boundary conditions - Dirichlet, Neumann, and a mix of the

two. Dirichlet boundary conditions set specific values of the pde. For example, keeping pressure or velocity fixed at an inlet or outlet. Or, setting the velocity to 0 in the case of a no slip condition at some solid surface. The Neumann boundary condition sets the normal of a variable to a fixed parameter. For example, at the surface of a solid boundary the velocity normal to the surface is set to 0 because the fluid cannot penetrate the boundary. The fluid can still go around the surface

2.1.2 Poiseuille Flow

This analytical formula for flow can be used to demonstrate the accuracy of my computed solutions.

Assume one dimensional flow for the navier stokes equation. Then the equation simplifies to

$$\frac{dp}{dx} = \rho\nu\frac{d^2u}{dx^2} \tag{3}$$

which with no slip conditions on each side, $u(0) = 0$ and $u(d) = 0$ can be solved so that the velocity profile is given by

$$u(x) = \frac{1}{2\rho\nu}x\,(d - x)$$

which is known as Poiseuille flow.

2.2 The Algorithm

To solve the navier-stokes equation computationally, I'll use a forward Euler scheme to discretize in the time domain and finite difference to discretize in the space domain.

The forward Euler discretization of equation (1) yields the following formula for stepping from iterations n to $n + 1$

$$u^{n+1} = u^n + \Delta t\,(u \cdot \nabla)\,u - \Delta t\frac{1}{\rho}\nabla p + \Delta t\frac{\nu}{\rho}\nabla^2 u + \Delta t f \tag{4}$$

which can't be reliably solved because it doesn't take into account the mass balance from equation (2). The pressure term can be removed and an intermediate velocity u^{inter}calculated as

$$u^{inter} = u^n + \Delta t\,(u \cdot \nabla)\,u + \Delta t\nu\nabla^2 u + \Delta t f \tag{5}$$

Define the next velocity timestep as

$$u^{n+1} = u^{inter} + \delta u \tag{6}$$

where $\delta u = -\Delta t\frac{1}{\rho}\nabla p$ is both a correction for the velocity and the gradient of pressure. Requiring mass balance for u^{n+1} implies

$$\nabla \cdot \left(u^{inter} + \delta u\right) = 0 \tag{7}$$

which results in an equation for pressure given by

$$\nabla^2 p = \frac{\rho}{\Delta t} \nabla \cdot u^{inter} \tag{8}$$

The final velocity is given by

$$u^{n+1} = u^{inter} - \frac{\Delta t}{\rho} \nabla p$$

Overall, the algorithm can be summarized as taking 3 steps. The first step is to calculate the intermediate velocity from equation (5). The next step is to calculate the pressure from equation (8). The final step is to update the velocity by adding the contribution of the pressure field to the intermediate velocity.

2.3 Implementation

The algorithm was partly implemented with cuda code. Because solving the Poisson equation from (8) requires multiple iterations of averaging the pressure function, it was chosen first to implement as a parallel algorithm. Bottlenecks from transfering data between the host and gpu were avoided by iterating the algorithm many times before sending back to the host.

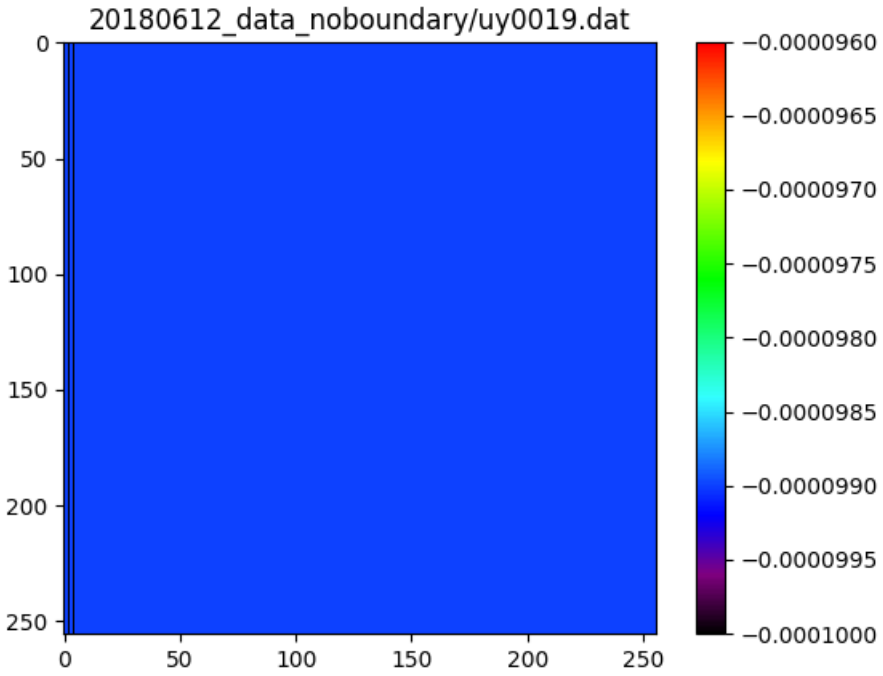
The CUDA code was written as a kernal with condition if statements to seperate the computation of the boundary conditions vs the internal averaging.

Simulations were run with "nvcc -o navier saxpy.cu && ./navier && ./moviescript.sh" which compiled the program, then called the program, and finally called a script to make plots to visulize the output.

2.4 Results and Conclusions

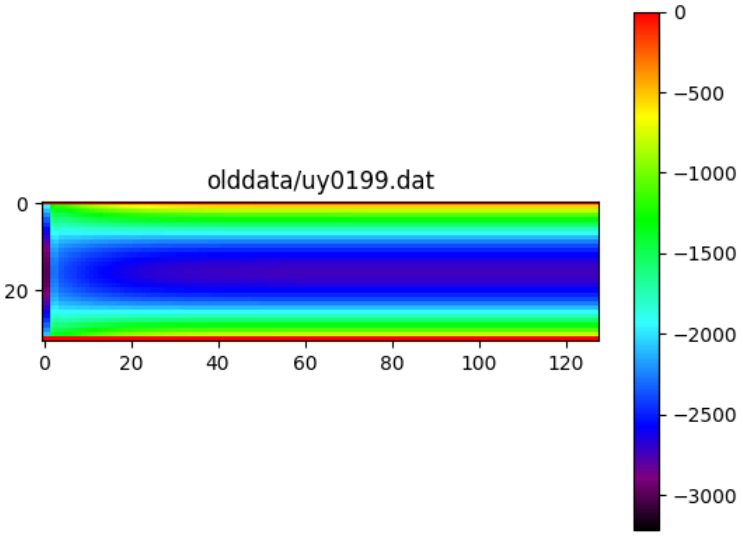
2.4.1 "1d-like" flow without any confinement

The results here are meant to show the implementation can reproduce results that can be found in analytical form. Code used here is found in noconfinement.cu. The velocity profile starting at zero settles to a uniform level set by the inlet. The outlet as a fixed pressure boundary condition while the other 2 boundaries have Newmann boundary conditions that set the normal components of the velocity to zero. Because there is no impediment to the flow, the velocity is the same everywhere.



2.4.2 Fluid flow in a channel with no slip

The code used here is given as noslip.cu. At the boundaries perpendicular to the inlet and outlet, fluid flow velocities were set to zero. The physical situation corresponding to this is fluid moving between two infinite, parallel walls. The figure below shows the results of fluid flow in the y direction when this no slip condition was used. The primary feature is that in the middle of the channel, there is a velocity maximum which corresponds well with Poiseuille flow.



3 Parallelizing Visualization

To visualize the progress of my simulations, I made videos. Making video frames is an "embarrasingly parallel" problem because frames for each timestep don't need to communicate with each other. Python has a library called the multiprocessing that enables a function to be executed simultaneously on different threads with different sets of data. I'll use the module to test how adding threads can speed up plotting of my data.

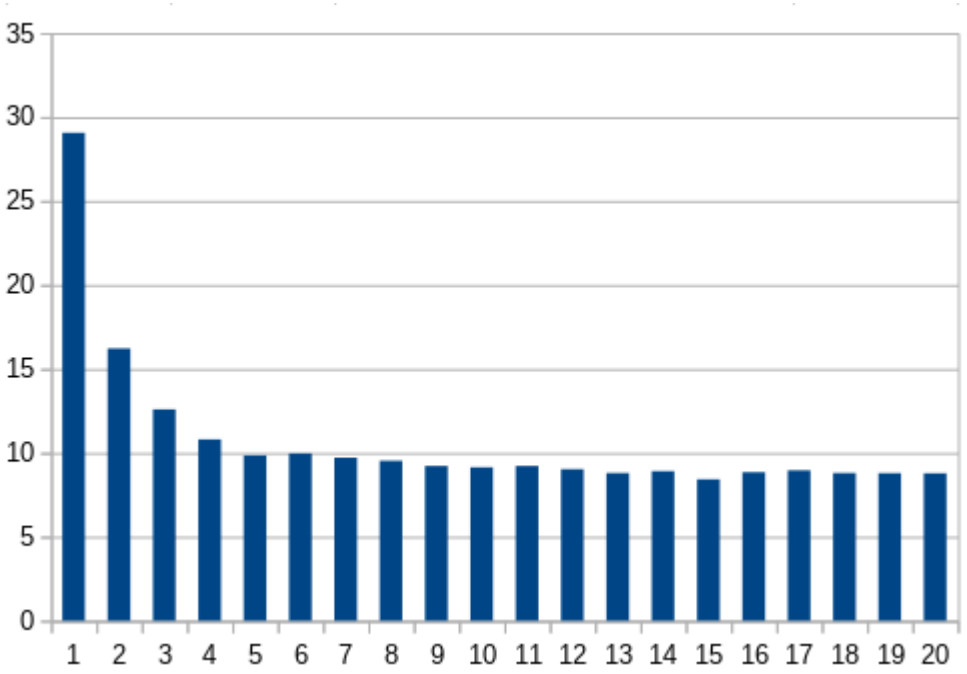
The script is included as pplotallparallel.py. The arguments for the script are the folder for the data plus the number of cores. For example the velocity field "pplotallparallel.py data/ux 8" This was also ran after the simulation with moviescript.sh

3.1 Results and Conclusions

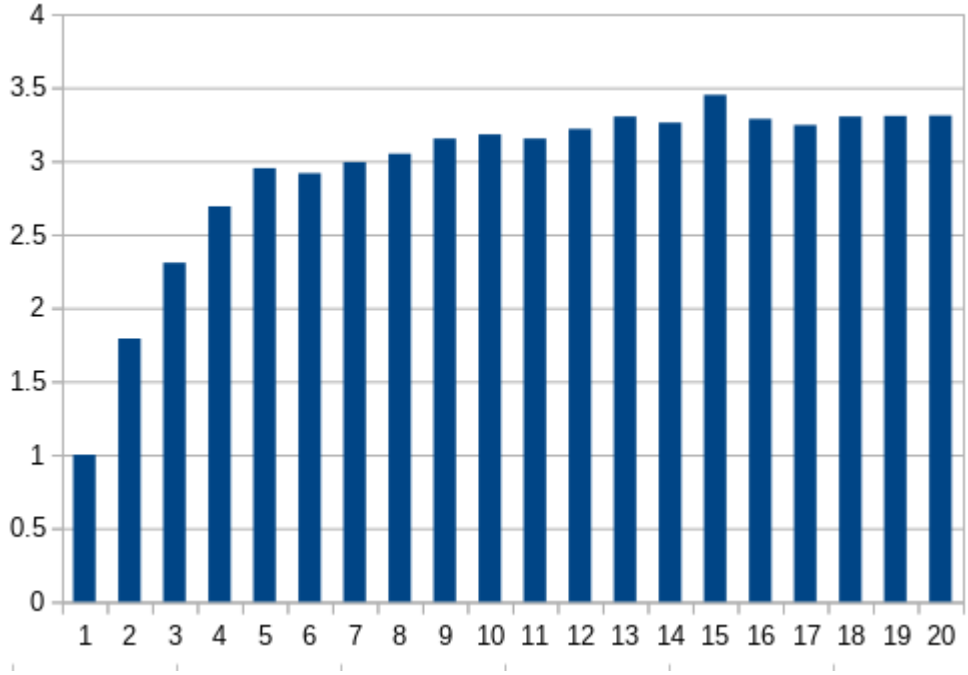
Plotted the pressure field for 100 time points. Running Python 3.6 with different levels of threads.

The workflow is as follows. First the data for the plots are loaded into memory with each timepoint getting its own thread. Next, minimum and maximum values of the dataset are collected to set the colorbar scale. Finally, threads are made with

Time Elapsed (seconds) vs number of threads



Speedup vs number of threads



Maximum speedup is about 3.3. Seem to be not much improvement beyond

5 threads. However, there no apparent cost to over requesting threads as long as they exist.

4 References

[1] "Fluid motion and solute distribution around sinking aggregates. I. Small-scale fluxes and heterogeneity of nutrients in the pelagic environment" Visser, André W. Jackson, George A. Mar Ecol Prog Ser 283: 55–71, 2004

[2] "Finite Difference Methods for the Stokes and Navier-Stokes Equations" Strikwerda, John SIAM J. ScI. STAT. COMPUT. Vol. 5, No. 1, March 1984

Source for microbial paper