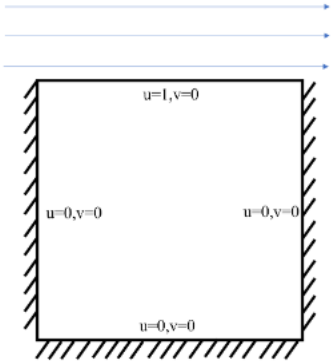
Fall 2022: ME759 Final Project Proposal

Project Title: Comparisons of Different Implementations in Chorin's Fractional Step Methods for solving Incompressible 2D Navier Stokes Equations
Link to git repo for project: https://git.doit.wisc.edu/BLU38/repo759
Problem statement: This project will compare two implementations on Chorin's FSM Navier Stokes Solver between GPU computing (cuda) and CPU computing (sequential and OpenMP). To simplify the project, the project will be conducted in the lid-driven cavity scenario, which has been widely used to test the validation and performance of a solver of Navier Stokes Equations. The standard case of lid-driven cavity problem is fluid contained in a square domain with Dirichlet boundary conditions on all sides, with three stationary sides and one moving side. Here is the setup of the fluid field of this project. A structured mesh grid will be constructed in various sizes to discover the scales of parallelization.



Motivation/Rationale: The Navier-Stokes equations are groups of partial differential equations describing the movement of Newtonian fluid in a given space and time. Numerically solving the Navier Stokes equation can be complex and time-consuming as it requires to solve hyperbolic, parabolic and elliptic problems for each component in equations. Witnessed such difficulty, Fractional Step Method (FSM) has been come up to solve NS equations numerically. The idea of FSM is to split up the equations into its constituent pieces and alternate between advance equation in time. Generally, there are two steps in FSM: the predictor step, and the corrector step. In the predictor step, the velocity fields will be evaluated by decoupling the pressure terms, and this process can be discretized into combinations of multiple summation and multiplications which are naturally suitable for parallelization. In the corrector step, the pressure Poisson equation will be solved based on the pressure field in previous step and this also can be parallelized because solving pressure Poisson equation is nothing more than solving Ax=B.

Explain how you contemplate going about it: Since the objective of this project is to compare the performance of sequential vs GPU vs OpenMP, a benchmark will be proposed that count the elapsed time different computations. Such benchmark can be either written by myself or from open source, and I haven't decided yet. CudaBLAS or other BLAS library might be utilized as well in my project when I am solving the Poisson pressure equation.

ME759 aspects the proposed work draws on:

- Discretization of the velocity field is a natural task that it can be parallelized by applying the Jacobian matrix. Cuda and OpenMP/MPI will be applied to parallelize the computation and I will implement this part from the scratch.
- Solving pressure field is another task that can be parallelized because we can solve pressure gradient at each node simultaneously. However, if we apply Jacobian matrix or Gauss-Sedel, to avoid the race condition, we might need to create a buffer matrix, which double the usage of memory. Solving pressure Poisson equation is really trick, I have the plan to do this part on my own however I might go for some libraries if I encounter difficulties.
- Temporal discretization can also be parallelized if we use implicit Euler discretization. The task is nothing but more than solving the linear equation Ax=B. For this part, I might utilize some library to save some effort. **I am wondering if there is open source parallelization-based linear solver I can applied here.**

Deliverables:
Three programs of implementations: sequential code, cuda code, openMP code.
A paper for reporting the output and benchmark of three implementations.

How you will demonstrate what you accomplished: the project will be documented, summarized and report in final project paper. In the report, we will list all the details of implementations as well as the algorithms for the parallelization. The performances of each implementation will be reported and discussed in the last part of the paper. Furthermore, I will compare the effects of the parallelization in each sub-step of FSM, such as predictor step, and corrector step.

Other remarks:
Hi Professor Negrut,
I am thinking if I can use some libraries for solving the linear equation (Ax=B). This can be part of FSM solver but I think if I implemented this linear solver will cost too much time. Is it ok that I directly use solver here and what are some choices I have for the solvers?