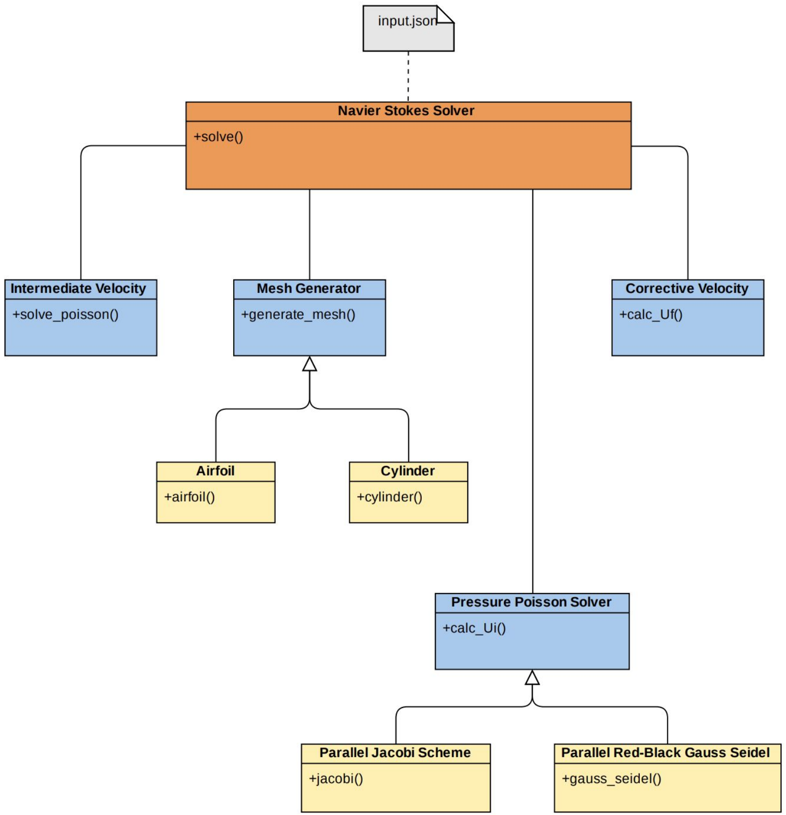
Developing an Efficient Navier-Stokes Solver Using Parallelized Iterative Schemes

Nathan Spilker & Kevin Andrade

Introduction

- Motivation: Computational Fluid Dynamics software can be computationally expensive and inefficient
- Approach: Implement parallel schemes to improve solver performance

Solver Architecture



Solver Architecture: Input Parameters

- input.json
- Number of processors

Parameter	Definition	Options
geometry	Shape	airfoil, cylinder, none
scale	Scaling of the geometry	
center	Center point of the geometry	

Parameter	Definition	Options
input_solver	Method for solving Pressure Poisson	jacobi, gauss_seidel
length_x	Length of the grid in the x-direction	
length_y	Length of the grid in the y-direction	
n_x	Number of grid points in the x-direction	
n_y	Number of grid points in the y-direction	
nu	Kinematic viscosity	
rho	Density	
iters	Number of iterations	
dt	Timestep	
eps	Minimum error allowed for solver convergence	
maxitr	Maximum number of iterations allowed for convergence per timestep	

Parameter	Definition	Options
u_init	Initial u-velocity for boundary conditions	
v_init	Initial v-velocity for the boundary	
u_flow	Initial flow velocity for the U grid	
v_flow	Initial flow velocity for the V grid	
cond_type	Type of flow condition	

Theory

Governing Equations →

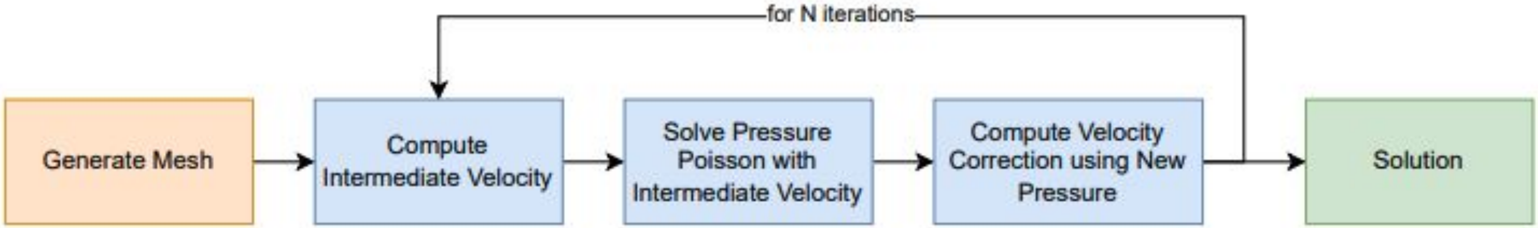
Navier Stokes: $$\frac{\partial U}{\partial t} + U \cdot \nabla U = -\frac{1}{\rho}\nabla p + \nu \nabla^2 U$$

Mass Conservation: $$\nabla \cdot U = 0$$

Discretized NS → $$\frac{U^{n+1} - U^n}{\delta t} = -U^n \cdot \nabla U^n - \frac{1}{\rho}\nabla p^{n+1} + \nu \nabla^2 U^n$$

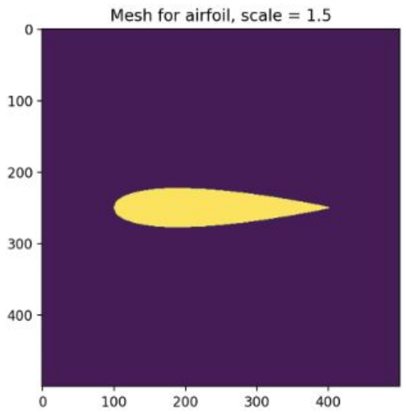
Pressure Poisson → $$\nabla^2 p^{n+1} = \frac{\rho}{\delta t}\nabla \cdot U^*$$

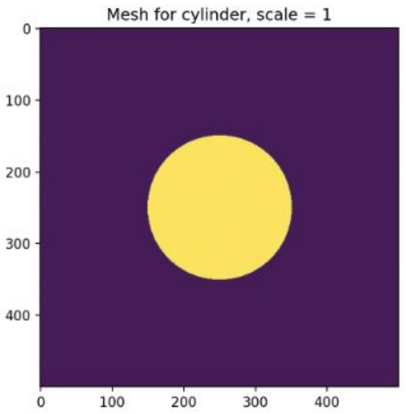
Solver Workflow



Solver Workflow: Generate Mesh

- Mesh is represented as an [n_x, n_y] grid
- Available Geometries:
  - Cylinder
  - Airfoil
- Geometry Inputs
  - Scale
  - Center





Solver Workflow: Compute Intermediate Velocity

- Each partial derivative is calculated using differencing schemes

$$U^* = U^n + \Delta t[\nu(\frac{\partial^2 U^n}{\partial x^2} + \frac{\partial^2 U^n}{\partial y^2}) - (U^n \frac{\partial U^n}{\partial x} + V_n \frac{\partial U^n}{\partial y})]$$

$$V_i = V^n + \Delta t[\nu(\frac{\partial^2 V_n}{\partial x^2} + \frac{\partial^2 V_n}{\partial y^2}) - (U^n \frac{\partial V_n}{\partial x} + V_n \frac{\partial U^n}{\partial y})]$$

Solver Workflow: Solve Pressure Poisson

- Algorithms Available:
  - Parallel Jacobi
  - Parallel Red-Black Gauss Seidel

$$\nabla^2 p^{n+1} = -\frac{\rho}{\Delta t}\nabla \cdot U^*$$

$$u_i^{n+1} = \frac{1}{4}(u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - h^2 f_{i,j})$$

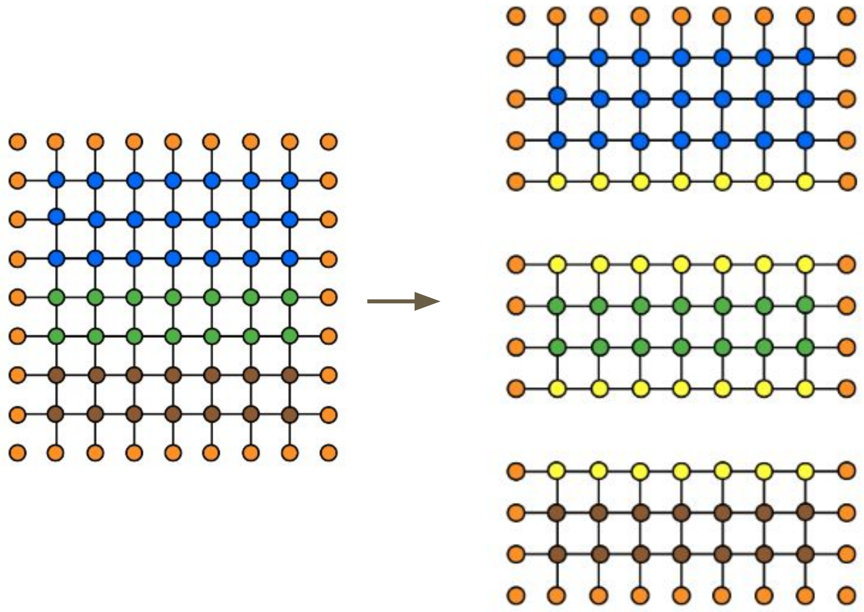
Jacobi Scheme

$$u_i^{n+1} = \frac{1}{4}(u_{i+1,j}^n + u_{i-1,j}^{n+1} + u_{i,j+1}^n + u_{i,j-1}^{n+1} - h^2)$$

Gauss Seidel Scheme

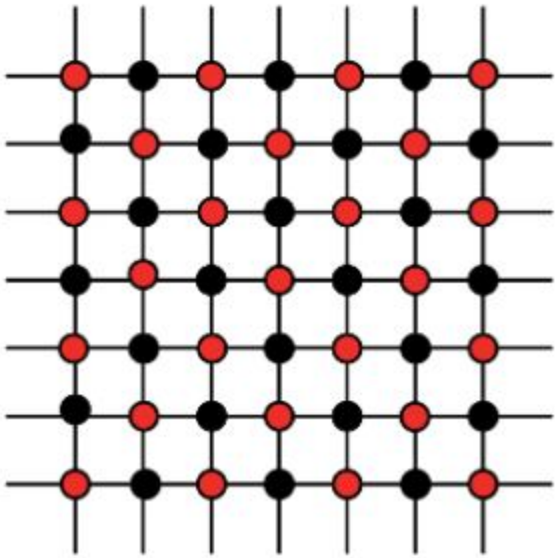
Parallel Jacobi Scheme

- Row calculations are distributed across various cores
- Ghost cells are used to communicate between cores



Parallel Red-Black Gauss Seidel Scheme

- Cells independent of each other are separated into red and black cells
- Red cells are calculated first
- Black cells are updated with new Red cell values
- Red and Black cell calculation is parallelized separately



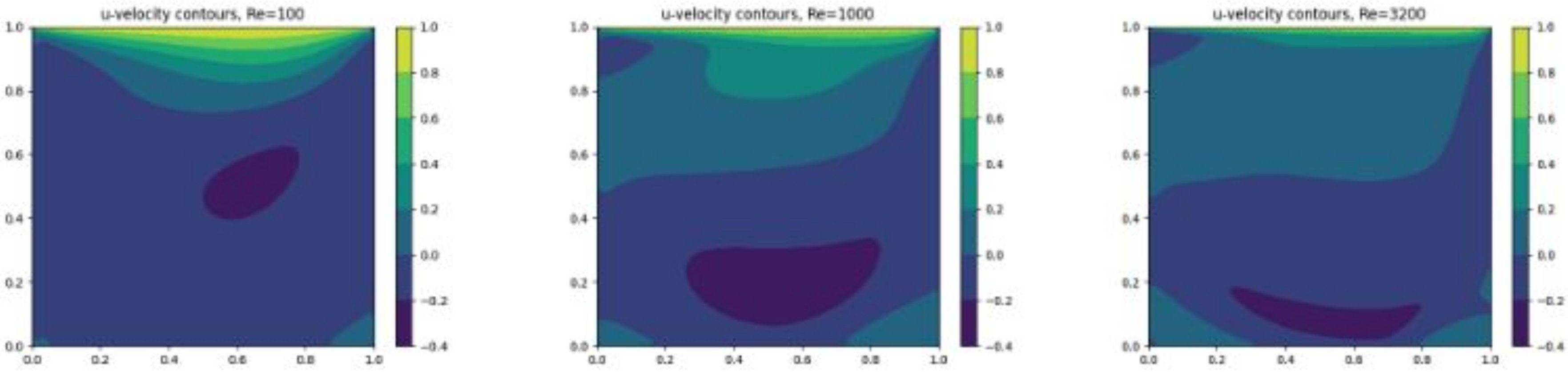
Solver Workflow: Compute Velocity Correction

- Each partial derivative is calculated using differencing schemes

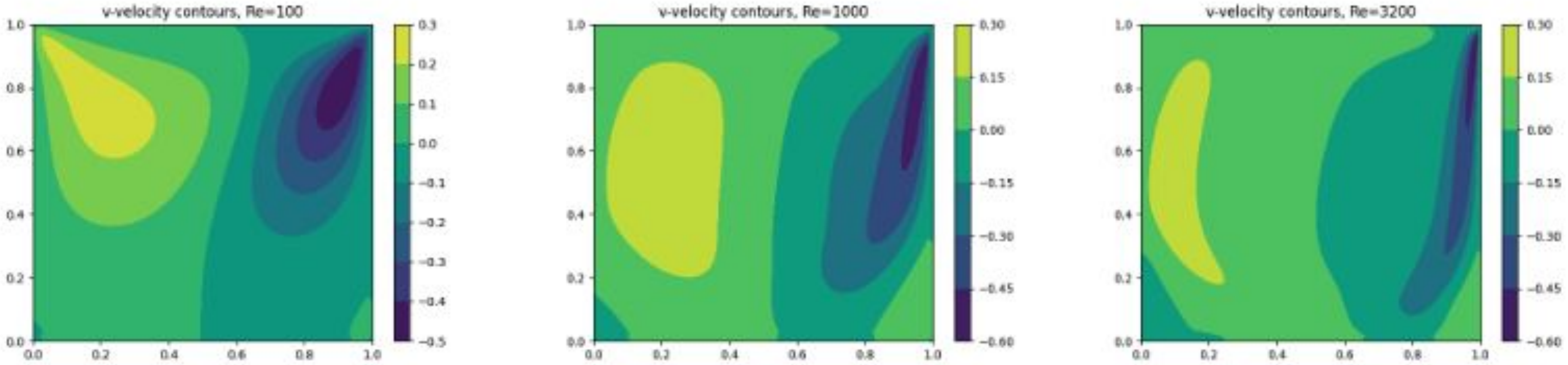
$$U^{n+1} = -\frac{\Delta t}{\rho}\nabla p^{n+1} + U^* \quad \Longrightarrow \quad \begin{aligned} \frac{\partial p}{\partial x} &= \frac{p(i,j) - p(i-1,j}{\Delta x} \\ \frac{\partial p}{\partial y} &= \frac{p(i,j) - p(i,j-1}{\Delta y} \end{aligned}$$

Lid Driven Cavity Flow

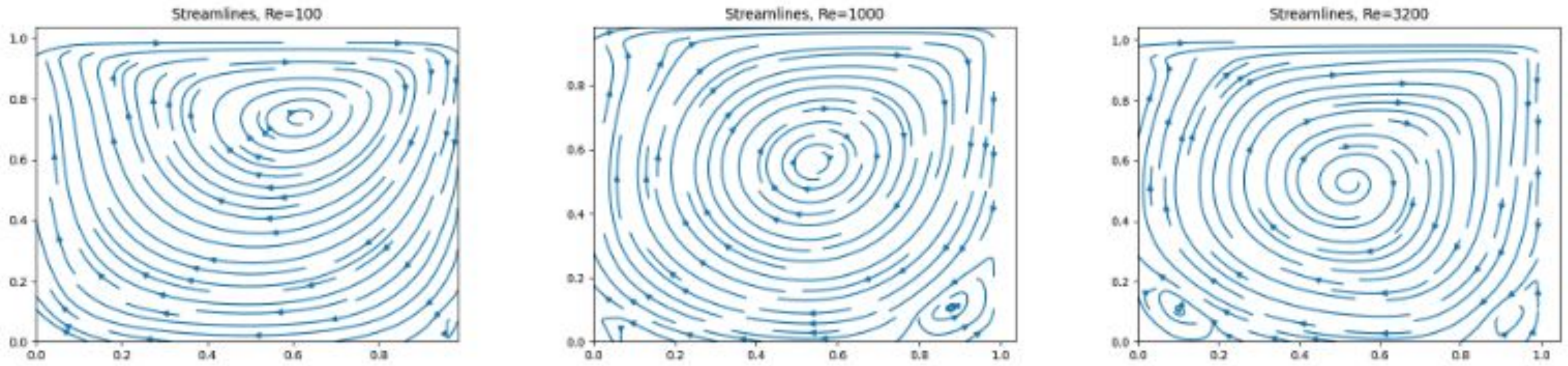
Results: Lid Driven Cavity Flow



Results: Lid Driven Cavity Flow

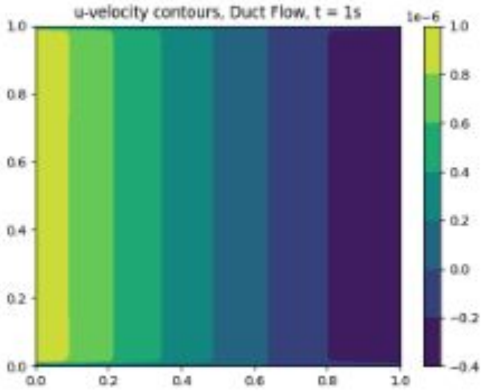


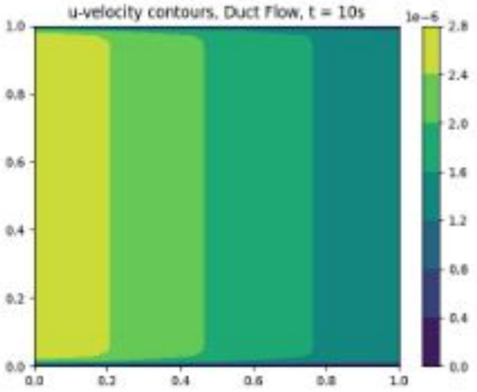
Results: Lid Driven Cavity Flow

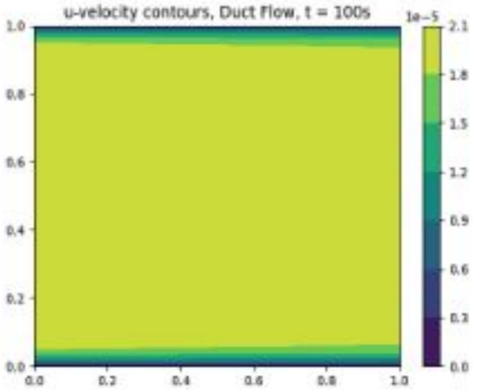


Duct Flow

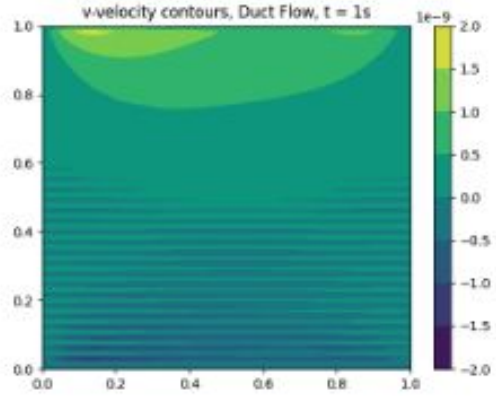
Results: Duct Flow

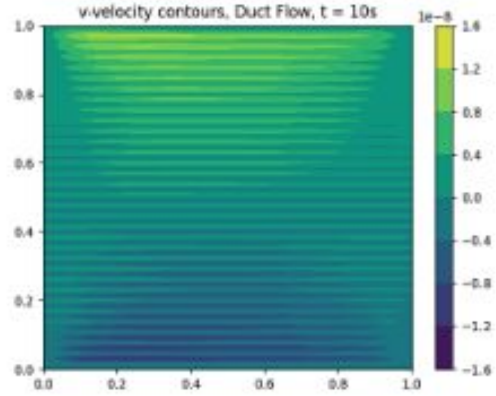


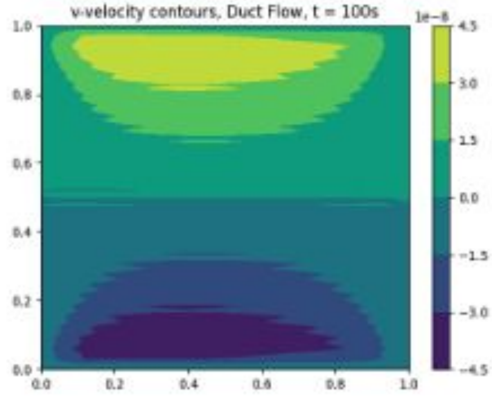




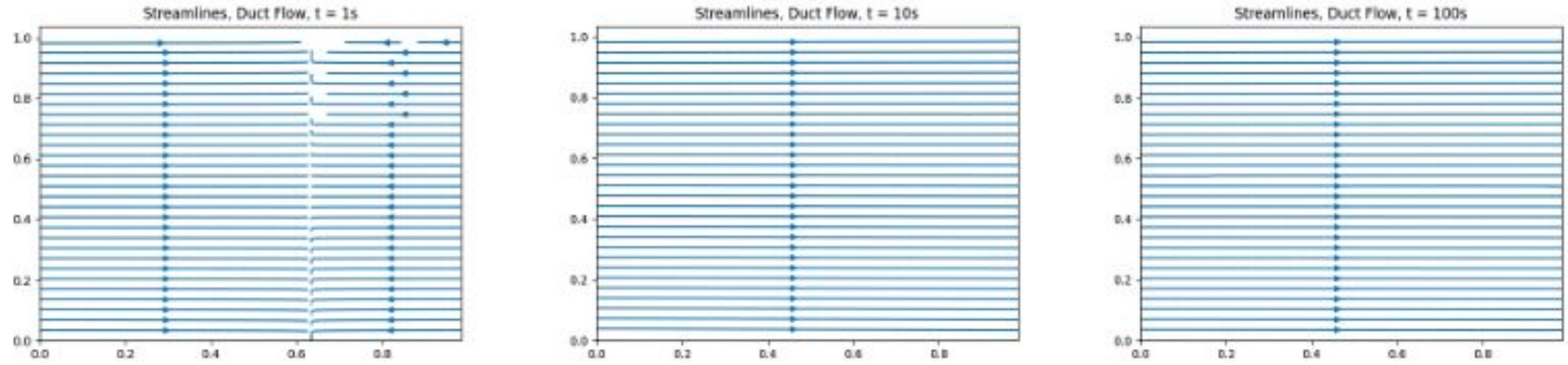
Results: Duct Flow





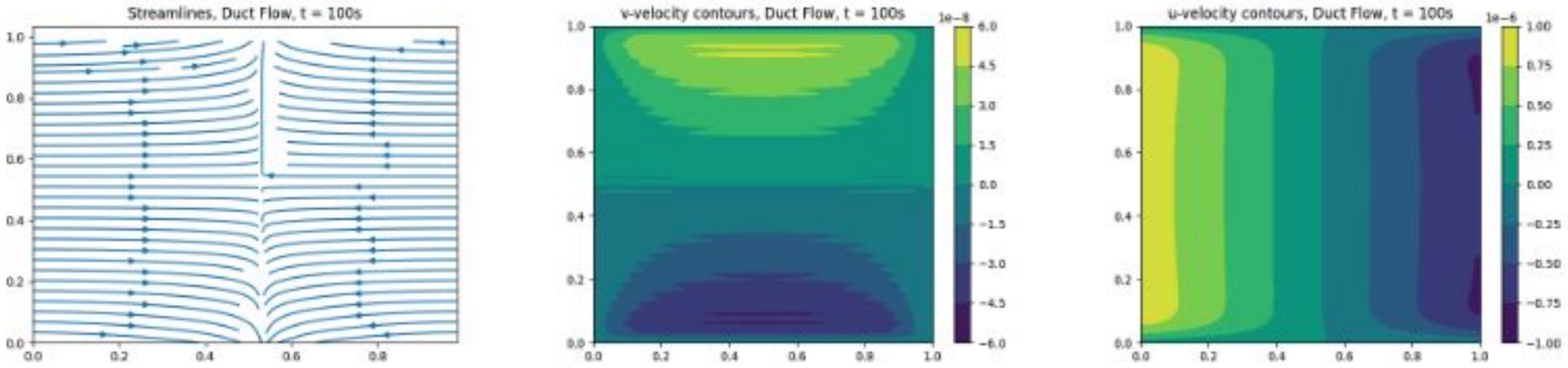


Results: Duct Flow



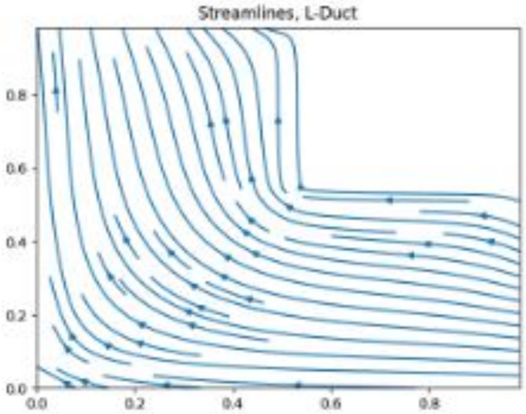
Duct Flow w/ Equal Pressure

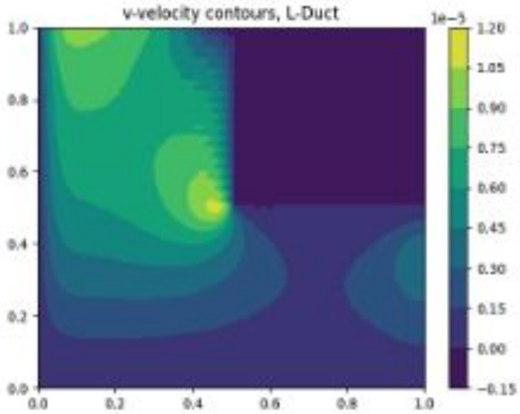
Results: Duct Flow w/ Equal Pressure

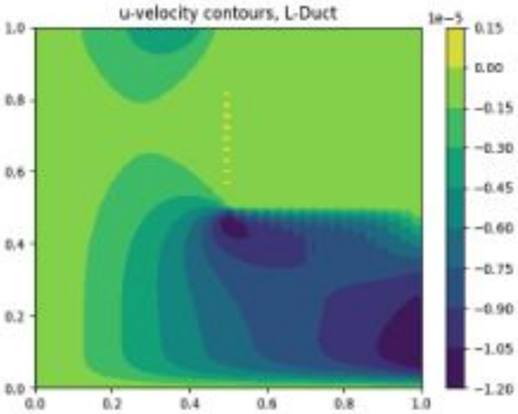


L-Duct Flow

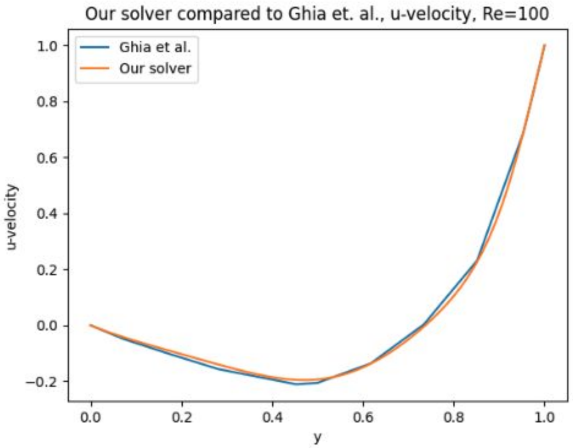
Results: L-Duct Flow

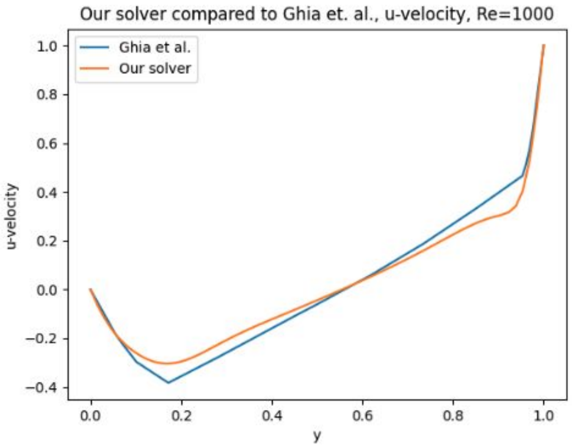


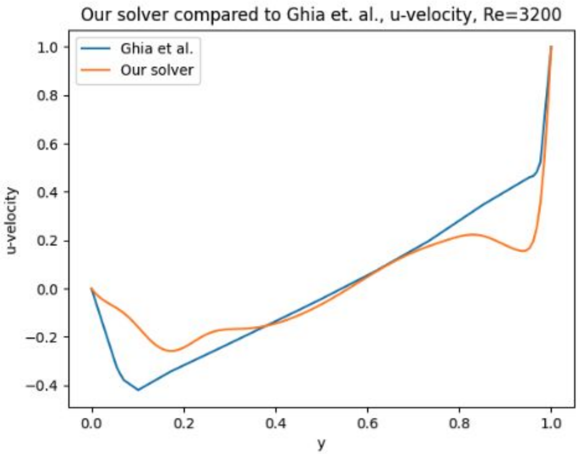




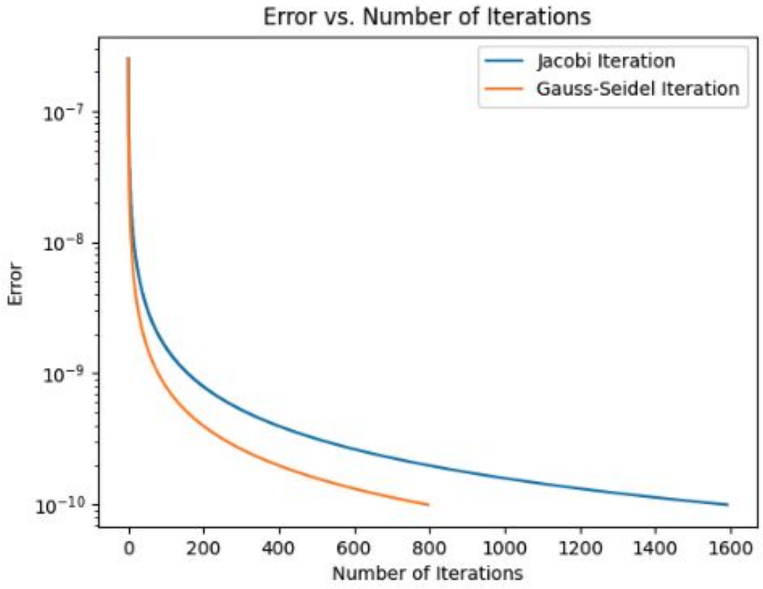
Performance: Accuracy

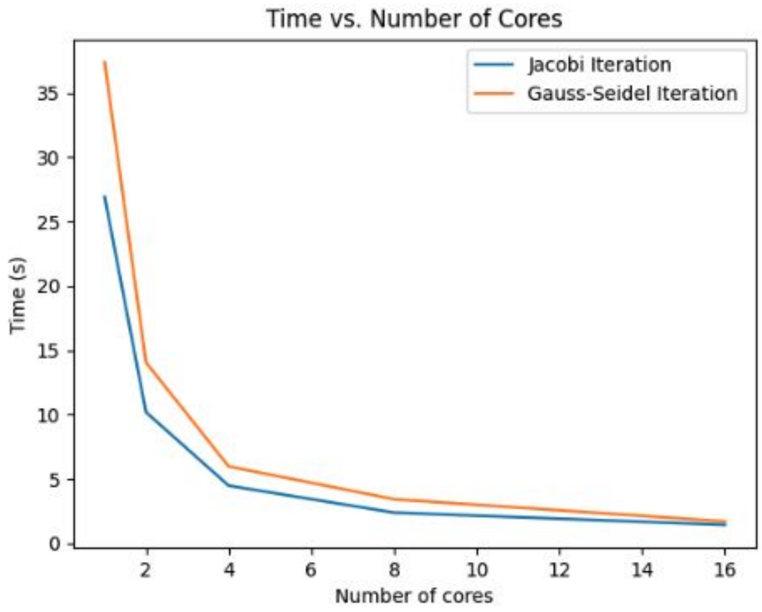






Performance: Parallel Algorithms





Thank you!