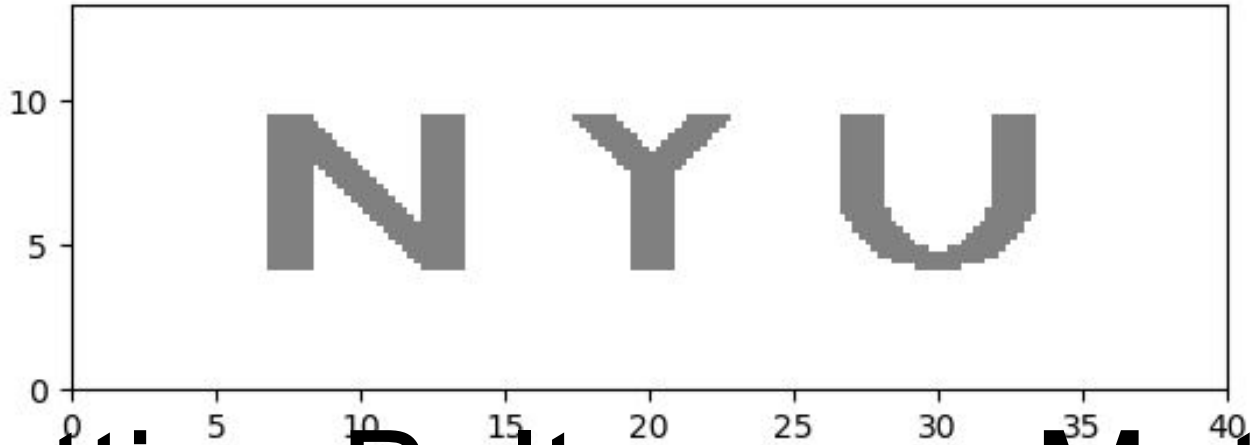


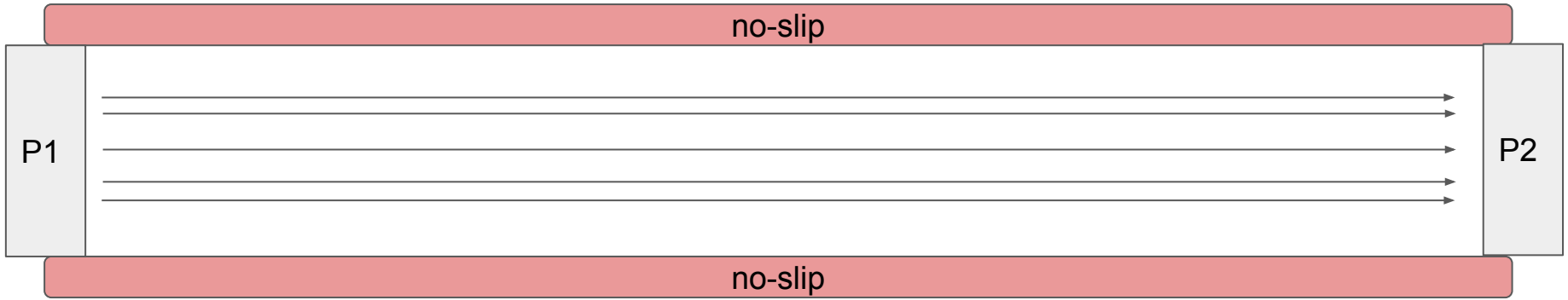
Lattice Boltzmann Methods

Zachary Sustiel

Recall: The Standard Hydro Code

To compute a flow…

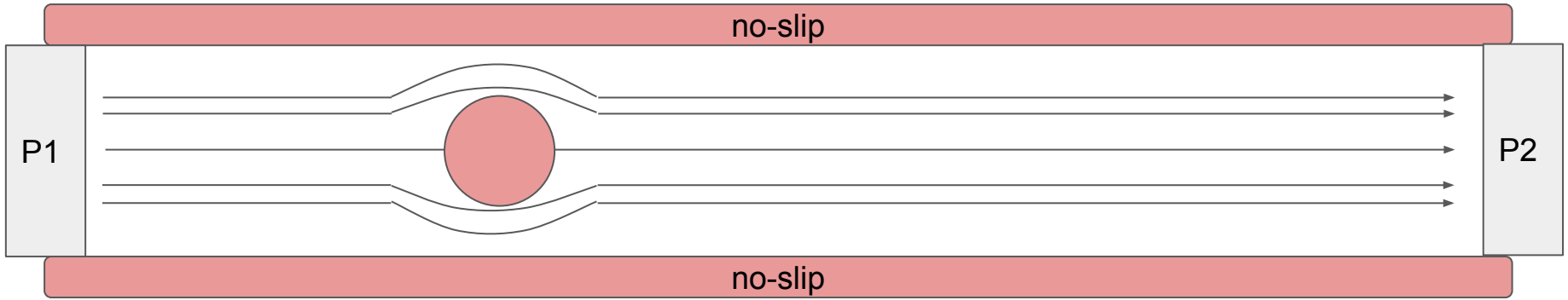
1) Initialize macroscopic, coarse-grained variables $\rho, \vec{v}, P$ at t=0
2) Use continuity equations (mass conservation, Navier Stokes, etc) for $\partial_t\rho, \partial_t\vec{v}, \partial_t P$
3) Use finite difference methods with continuity equations to find $\rho, \vec{v}, P$ at t=dt
4) Repeat



...but what if there's a complex boundary?

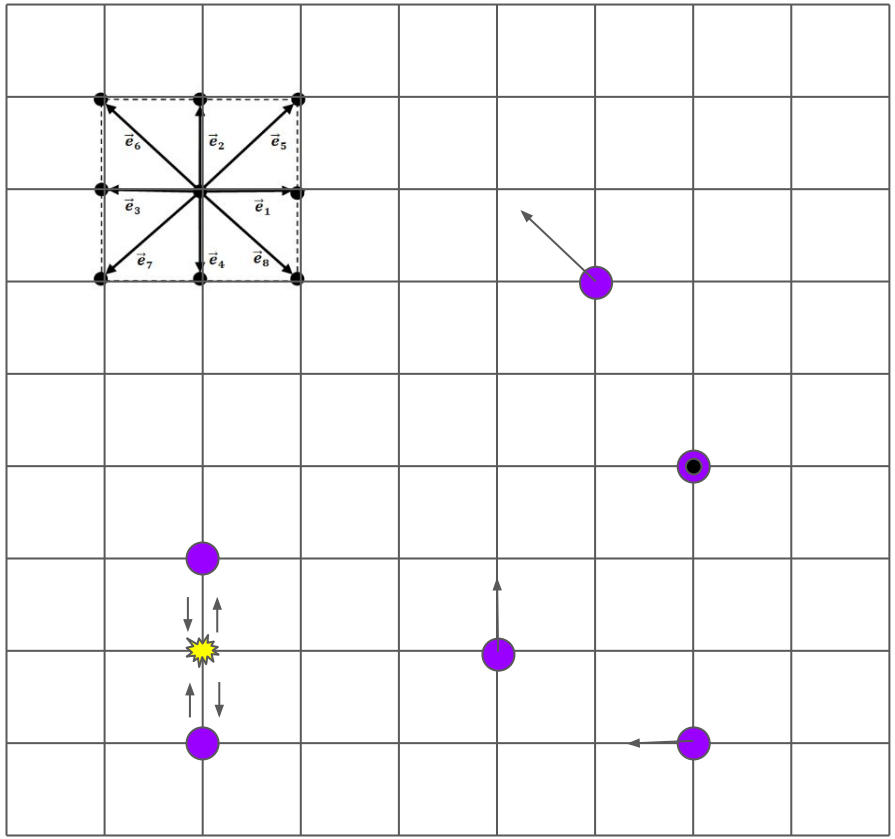
To solve for the flow, we need to find the analytic solution corresponding to the boundary conditions

This can be tough, especially if the boundary doesn't have an obvious symmetry



An easier way? Lattice Gas Automata

1) Keep track of individual particle x, v
2) Particles confined to nodes on a grid
3) Particles can have one of 9 discrete velocities
4) At each timestep, particles stream to adjacent node
5) Collisions handled with conservation of momentum



Lattice Gas Automata Cont.

Benefit:

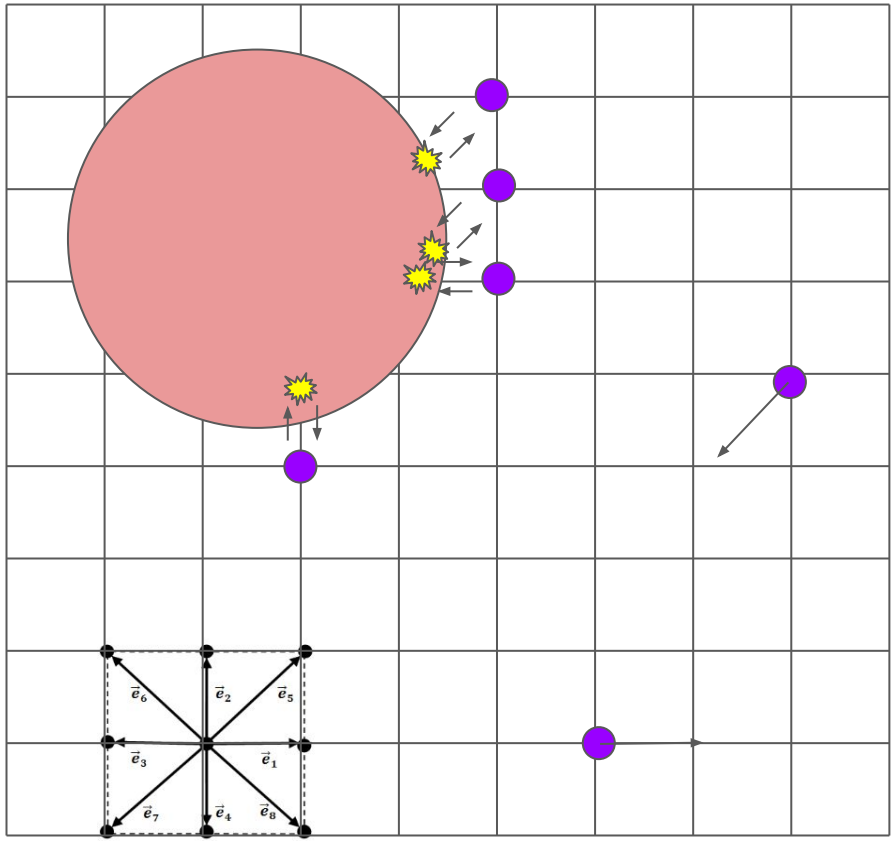
Very easy to immerse an object in flow!

(When you hit the object, v → -v)

Drawback:

High statistical noise

(Remember: gases don't actually live on grids)



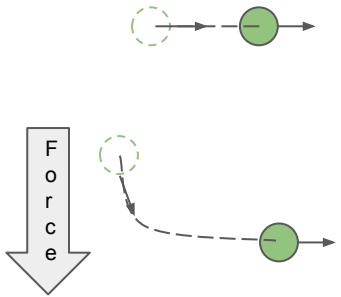
Lattice Boltzmann Method: The Boltzmann Equation

Replace particles by particle distribution functions $f(\vec{x}, \vec{v}, t)$

$f(\vec{x}, \vec{v}, t)$ = Probability of finding a particle at position x with velocity v at time t

$f(\vec{x}, \vec{v}, t)$ obeys the Boltzmann Equation:

$$\underbrace{\frac{\partial f}{\partial t} + \vec{u} \cdot \vec{\nabla} f}_{\text{Streaming Term}} + \underbrace{\frac{F}{m} \frac{\partial f}{\partial \vec{u}}}_{\text{External Forcing}} = \underbrace{\Omega}_{\text{Collision Operator}}$$



Intuitively:

P(particle is here now) = P(particle came from over there) + P(Particle got pushed here) +
P(particle was going somewhere else but somebody bumped it over here)

LBM: Discretized Boltzmann Equation

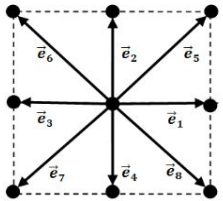
Letting $f_i(\vec{x},t) \equiv f(\vec{x},\vec{e_i},t)$, the discretized Boltzmann Equation (with no ext force) is

$$\underbrace{f_i(\vec{x}+c\vec{e_i}\Delta t, t+\Delta t) - f_i(\vec{x},t)}_{\text{Streaming}} = \underbrace{\frac{\Delta t}{\tau}\left(f_i^{eq}(\vec{x},t) - f_i(\vec{x},t)\right)}_{\text{BGK Collision Operator}}$$

For the collision term, we use the BGK Collision Operator: τ is a relaxation time

$$f_i^{eq} = \rho\omega_i\left(1 + 3\frac{\vec{e_i}\cdot\vec{v}}{c} + \frac{9}{2}\frac{(\vec{e_i}\cdot\vec{v})^2}{c^2} - \frac{3}{2}\frac{(\vec{v_i}\cdot\vec{v})^2}{c^2}\right)$$

with weights $\omega_i = \begin{cases} 4/9, & i=0 \\ 1/9, & i=1,2,3,4 \\ 1/36, & i=5,6,7,8 \end{cases}$



Lattice Boltzmann Method Cont.

Note: the BGK operator is *mesoscopic*: f_i^{eq} depends on both **microscopic** f_i AND on the macroscopic fields

$$\rho(\vec{x},t) = \sum_{i=0}^{8} f_i(\vec{x},t) \text{ and } \vec{v}(\vec{x},t) = \frac{1}{\rho}\sum_{i=0}^{8} c\vec{e}_i f_i(\vec{x},t)$$

Algorithm:

1) Initialize/calculate macroscopic ρ and $\vec{v}$, and compute f_i^{eq}
2) Use f_i^{eq} to compute post-collision f_i^*
3) Stream the post-collision f_i^* in the direction e_i to obtain f_i at t+dt
4) Repeat

Boundary Conditions: Full Bounce Back

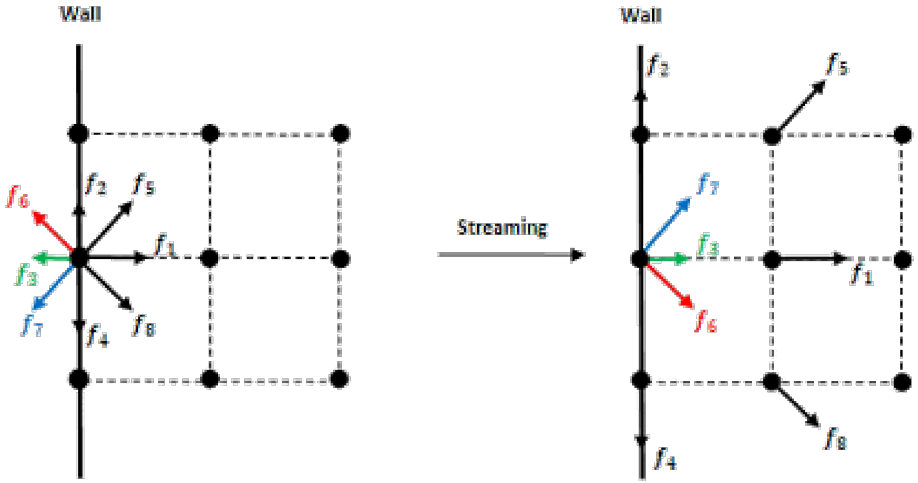
At the boundary, not all f_i are known after streaming, so we impose macroscopic boundary conditions instead.

Ex: *No-Slip Surface:*

$$\vec{v_x} = (f_1 - f_3) + (f_5 - f_7) + (f_8 - f_6) = 0$$
$$v_y = (f_2 - f_4) + (f_5 - f_7) + (f_6 - f_8) = 0$$

We can satisfy this by just “bouncing” off the wall in the opposite direction:

$$\left.\begin{aligned} f_1 &= f_3 \\ f_5 &= f_7 \\ f_8 &= f_6 \end{aligned}\right\}$$



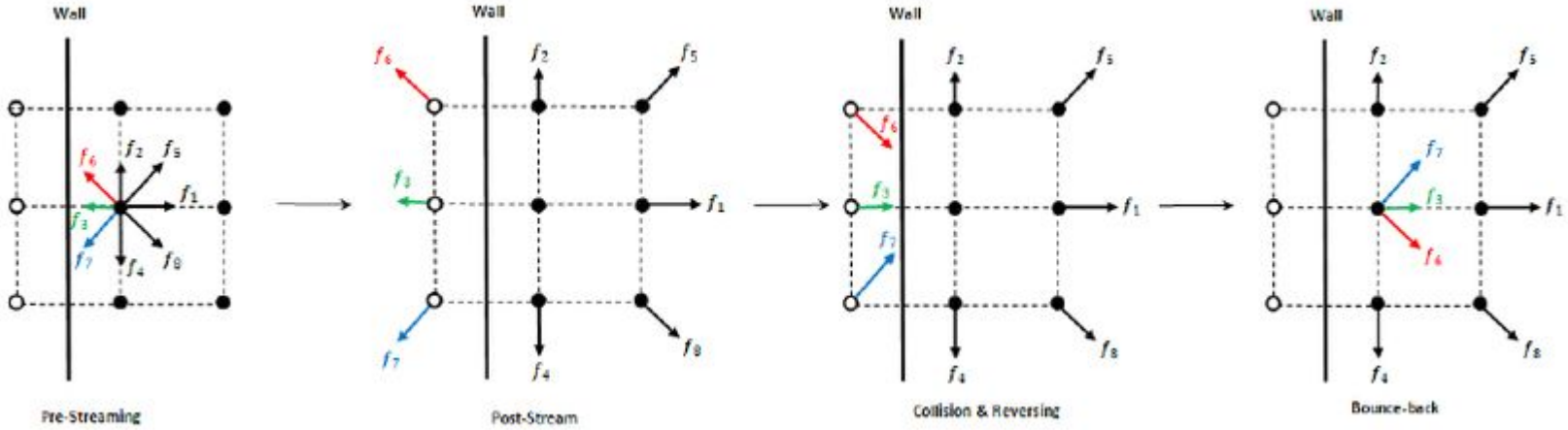
<u>Full Bounce Back:</u>

- First Order
- Independent of Boundary Orientation!!!
- Great for immersed objects

Boundary Conditions: Half Bounce Back

The Halfway Bounce-Back is a second order method for no-slip walls:

- Place the wall halfway between nodes and initialize ghost cells outside the fluid
- Perform reversal on ghost nodes during collision step, and stream back into fluid



- Second Order
- More difficult to implement than Full Bounce Back

Zou-He Boundary Conditions

How to make a pressure gradient?

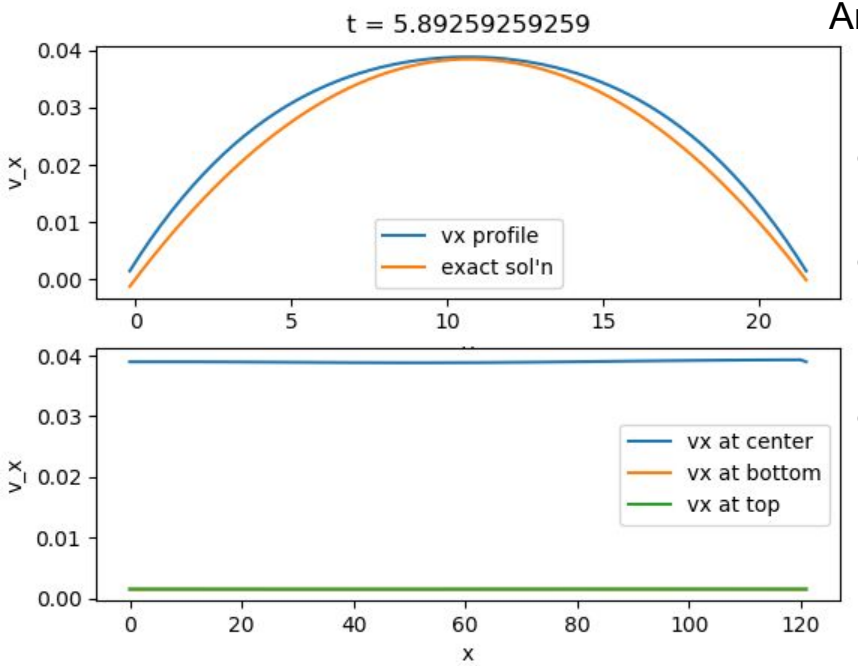
- @ inlet/outlet, $v_y = 0$
- $\frac{P}{\rho_0} = c_s^2 \rho$ (where $c_s = \frac{\Delta x}{\Delta t}$ is the lattice speed)

$$u_x = 1 - \frac{[f_0 + f_2 + f_4 + 2(f_3 + f_6 + f_7)]}{\rho_{in}}.$$

$$\begin{aligned} f_1 &= f_3 + \frac{2}{3}\rho_{in} u_x, \\ f_5 &= f_7 - \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho_{in} u_x, \\ f_8 &= f_6 + \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho_{in} u_x. \end{aligned}$$

Pressure Driven Flow in a No-Slip Channel

If we assume symmetry in the x direction, the analytical solution is Poiseuille Flow:

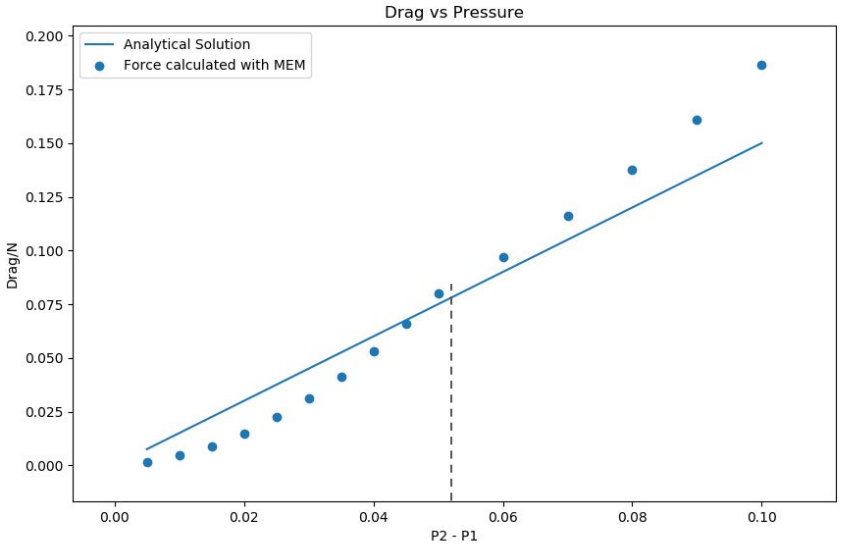


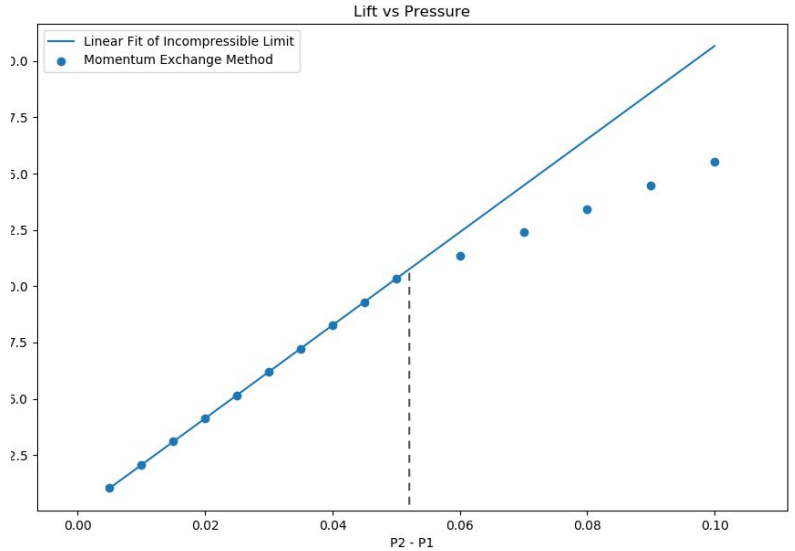
Analytic Solution: $\vec{v} = \frac{\Delta P}{2\eta} y(H - y)\hat{x}$

- The simulation doesn't converge to the exact solution because the model is incompressible
- To test convergence to the analytical solution, we must either run at low velocity (i.e. $v << \frac{h^2}{3s^2}$ or use d2q9i model
- At the channel walls, the velocity is close to zero. I.e, bounce-back works nicely

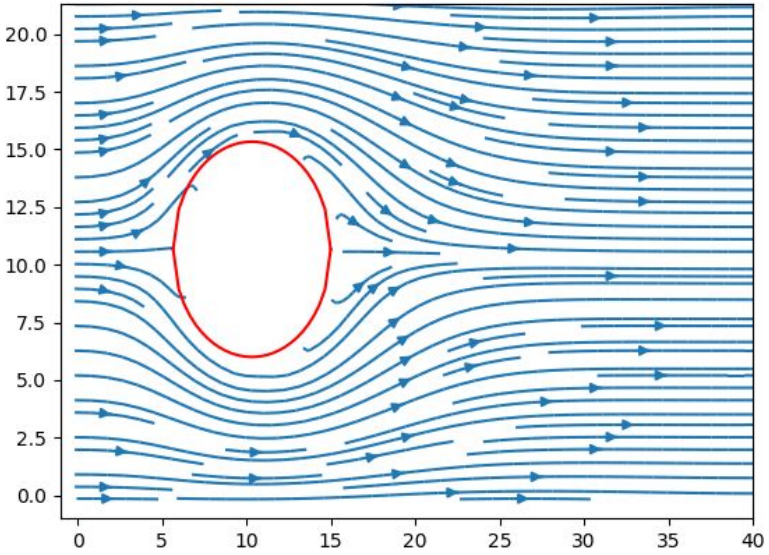
Pressure Driven Flow cont: Force Evaluation

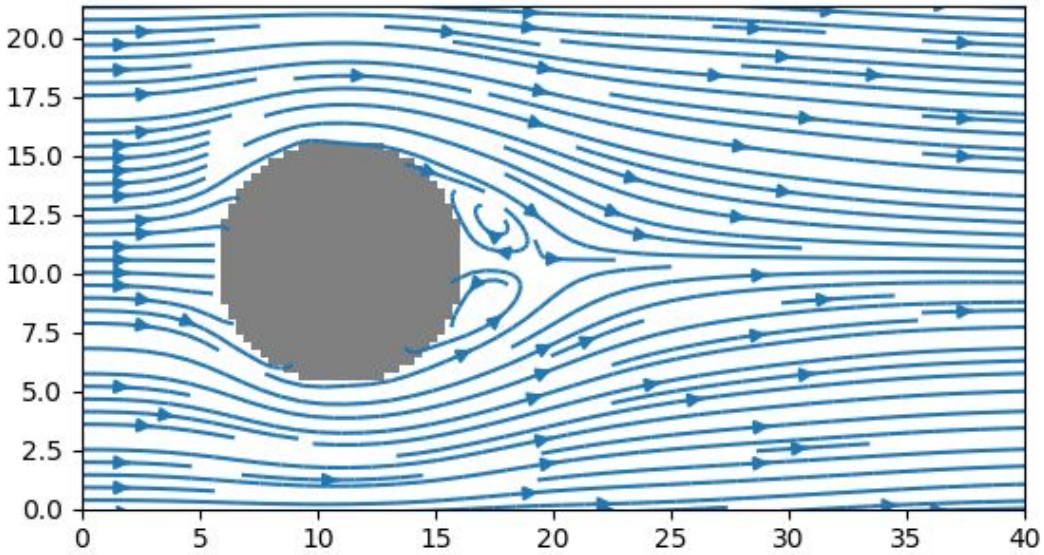
Using the Momentum Exchange Method, we can approximate the force on a bounce-back boundary as $\Delta\vec{F} = [\sum f_i + \tilde{f}_i^{-}]\vec{e}$





Flow around a Cylinder





Future Directions

- Lift on a cylinder: very easy to calculate using bounce-back, but higher order methods might be needed for curved boundaries to get desired results

- Flow around a triangle
- Flow around two cylinders
- Flow around… literally anything. If you can draw it on a piece of grid paper, this code can give you the flow.

- Calculate convergence of bounce-back method/implement d2q9i model

- Implement velocity at the boundaries, or boundaries that can move (i.e. a large sphere immersed in a fluid)