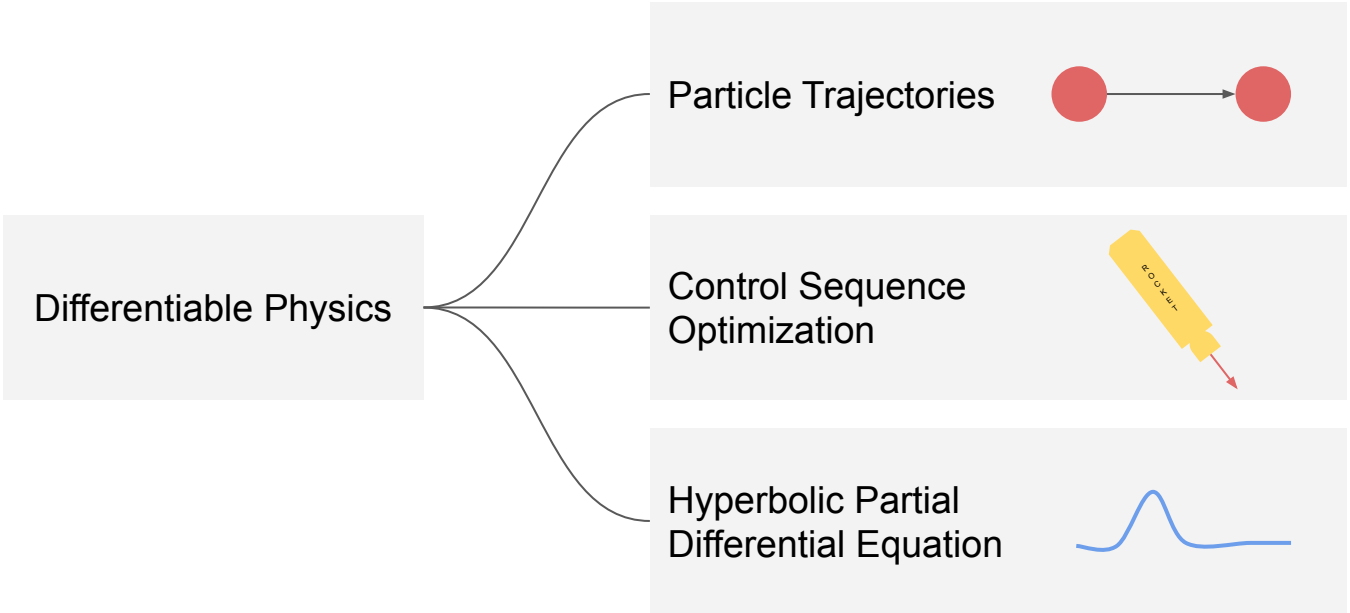
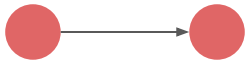
Differentiable Physics Simulations

Andrin Rehmann - Advanced Simulations

Overview

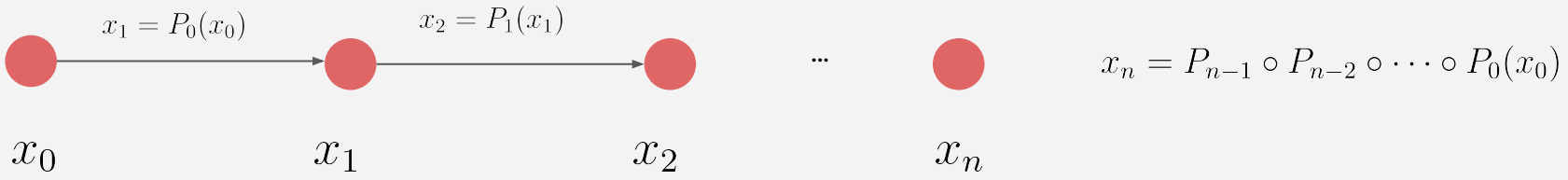

Differentiable Physics
Particle Trajectories
Control Sequence Optimization
Hyperbolic Partial Differential Equation

Particle Trajectories

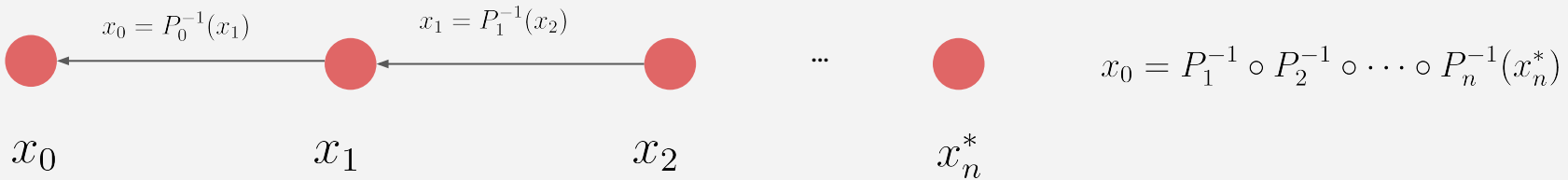


Inverse Simulation

Forward: Given x_0 find x_n



Inverse: Given x_n^* find x_0

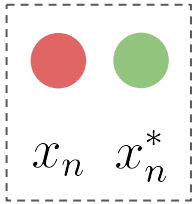


But can we find P^{-1}? - Yes but not feasible for complex forces.

Better use: $\frac{\partial P_i}{\partial x_i}$ Gradient with respect to output of numerical solver.

Optimization

1. Compute the loss given the desired final state and the simulated final state.



$$L = || \left(P_{n-1} \circ P_{n-2} \circ \cdots \circ P_0(x_0)\right) - s_n^* ||^2$$

2. Get loss with respect to initial state.

$$\frac{\partial L}{\partial x_0} = \frac{\partial L}{\partial x_n} \frac{\partial P_{n-1}}{\partial x_{n-1}} \cdots \frac{\partial P_0}{\partial s_0}$$

3. Update initial state with gradient descent.

$$x_0 = x_0 - \alpha \frac{\partial L}{\partial x_0}$$

Verlet Integration

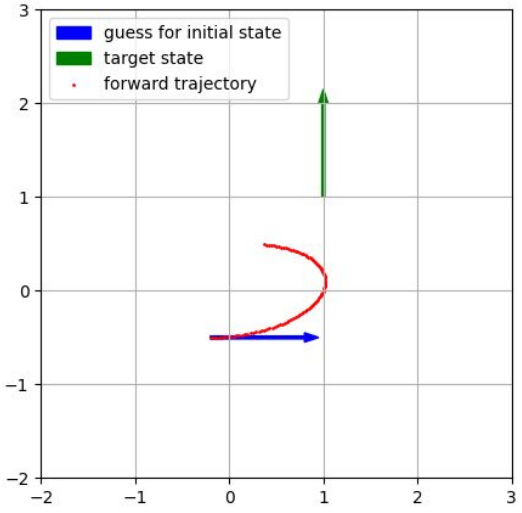
Forward

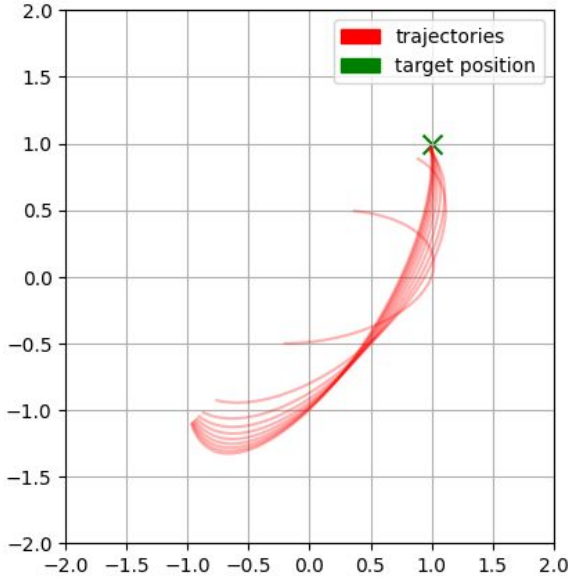
$$\begin{bmatrix} x_{i+1} \\ v_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ v_i \end{bmatrix} + \Delta t \begin{bmatrix} v_i \\ 0.5 \times (a_i + a_{i+1}) \end{bmatrix} + \frac{\Delta t^2}{2} \begin{bmatrix} a_i \\ 0 \end{bmatrix}$$

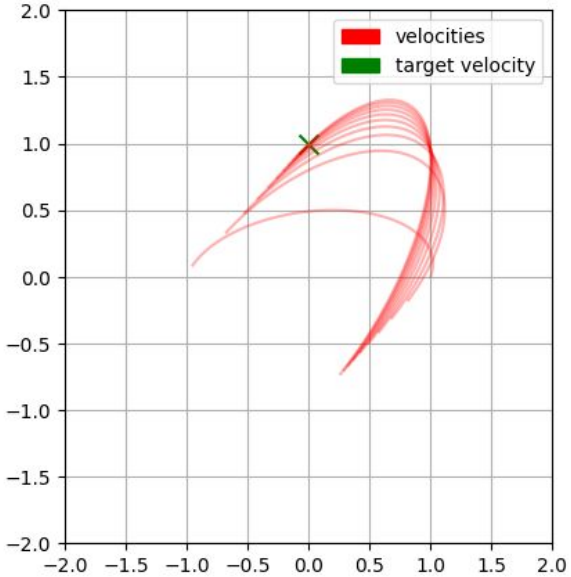
Simulation Derivative

$$\frac{\partial P(s_i)}{\partial s_i} = \begin{bmatrix} \frac{\partial P(s_i)}{\partial x_i} & \frac{\partial P(s_i)}{\partial v_i} \end{bmatrix} = \begin{bmatrix} 1 + \frac{\Delta t^2}{2m} \frac{\partial F(x_i)}{\partial x_i} & \Delta t \\ \frac{\Delta t}{2m} \left(\frac{\partial F(x_i)}{\partial x_i} + \frac{\partial F(x_{i+1})}{\partial x_i} \right) & 1 \end{bmatrix}$$

2D Harmonic Oscillator

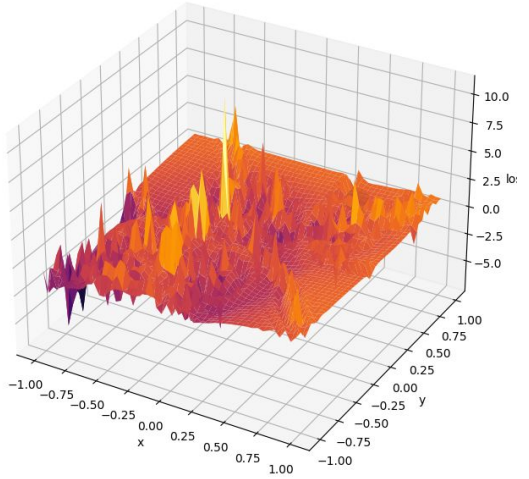


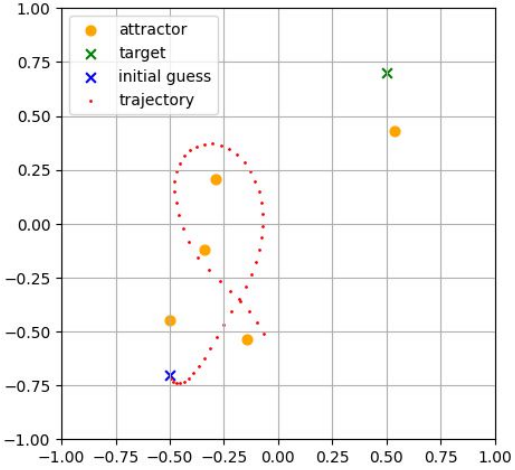


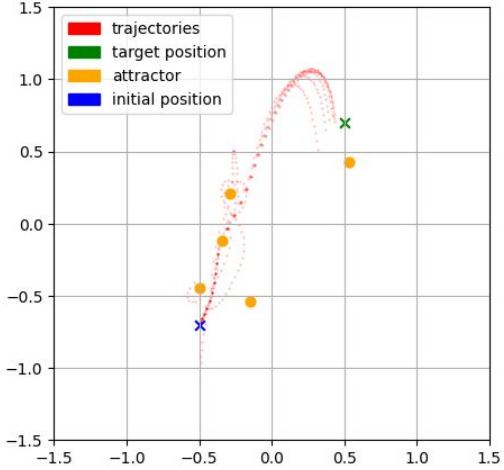


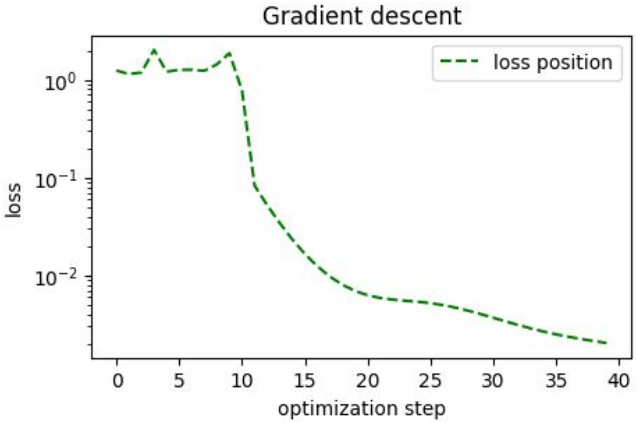
Automatic Differentiation

- Finding the simulation derivative is complex
- JAX combines automatic differentiation and just in time compilation
- adaptive learning rate and momentum due to complex surface

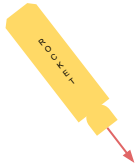






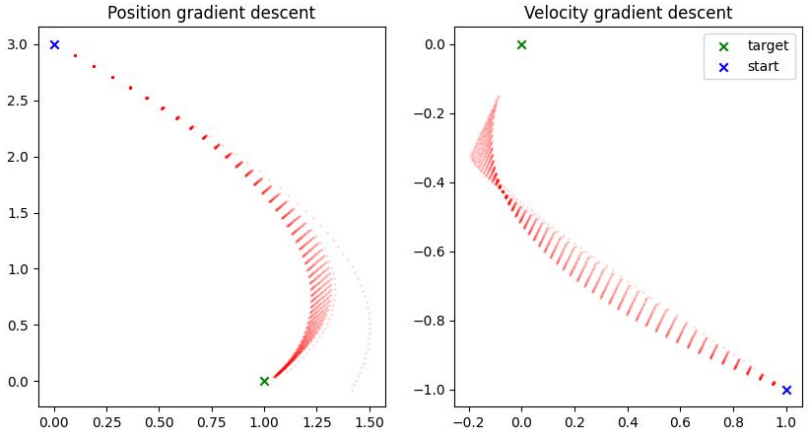


Control Sequence
Optimization



Control Sequence Learning: Liquid Rocket Engine

- Control sequence for throttle and direction of thrust
- Find optimal path to target position and velocity
- Minimal fuel consumption
- Initial conditions are given
- Is this Optimal Control?

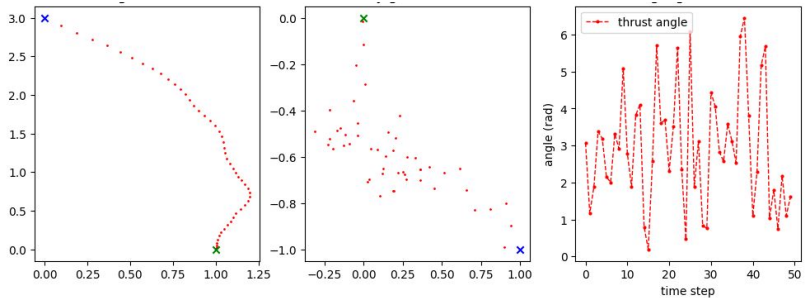


$$L = \underbrace{MSE(x_n^* - x_n)}_{\text{position}} + \underbrace{MSE(v_n^* - v_n)}_{\text{velocity}} + \underbrace{MSE(v_0, v_1, , v_n)}_{\text{fuel}}$$

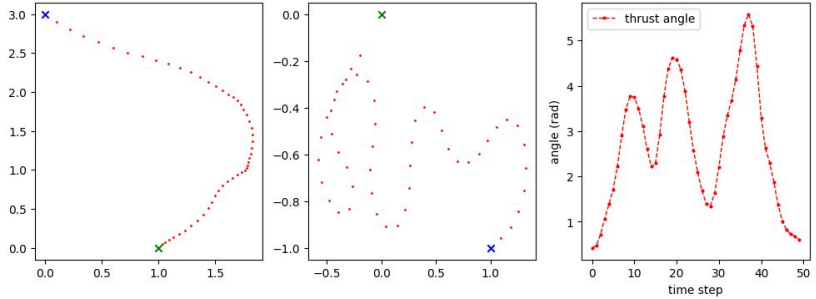
Control Sequence Learning: Solid Rocket Engine

- Control only for direction of thrust
- Downwards force from rocket can be reduced by oscillation
- Minimize change of angle

$$L = \underbrace{MSE(x_n^* - x_n)}_{\text{position}} + \underbrace{MSE(v_n^* - v_n)}_{\text{velocity}} + \underbrace{MSE(a_1 - a_0, a_2 - a_1, , a_n - a_{n-1})}_{\text{change of angle}}$$

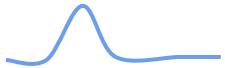


Only position / velocity based loss



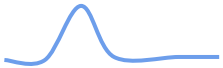
Minimize Change of angle

Hyperbolic Partial
Differential Equation



Hyperbolic PDE: 1D Burgers Equation

- Hyperbolic PDE's describe waves.



- Burger's Equation without diffusion is a conservation equation.

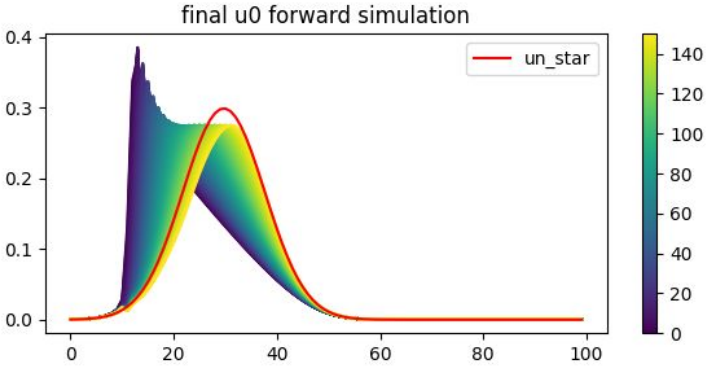
$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0 \quad f(u) = \frac{u^2}{2}$$

- Find initial state such that final distribution is reached at the end of the simulation.

Mac Cormack Method

$$\bar{u}^{n+1} = u^n - \frac{\Delta t}{\Delta x}\left(f(u_i) - f(u_{i-1})\right)$$

$$u^{n+1} = \frac{1}{2}\left(u^n + \bar{u}^{n+1}\right) - \frac{\Delta t}{2\Delta x}\left(f(\bar{u}_i^{n+1}) - f(\bar{u}_{i-1}^{n+1})\right)$$



Hyperbolic PDE: 2D Wave Equation

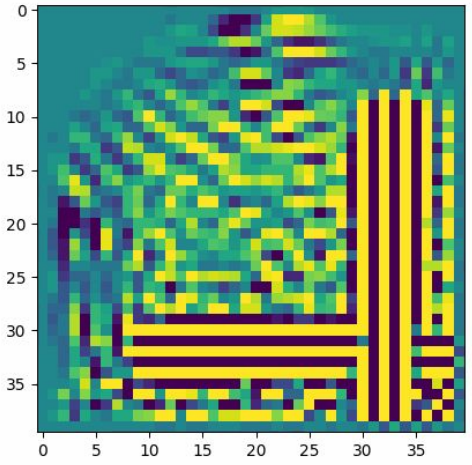
- Linear 2D Wave Equation (Or linear Convection Equation)

$$\frac{\partial}{\partial t} = c\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right)$$

- Same Mac Cormack scheme as before with flux function:

$$f(u) = c \cdot u$$

- Gradient Descent with 300 optimization steps. Slow convergence



Attempt at Compressible Navier Stokes

- Mac Cormack method with alternating backward / forward differences

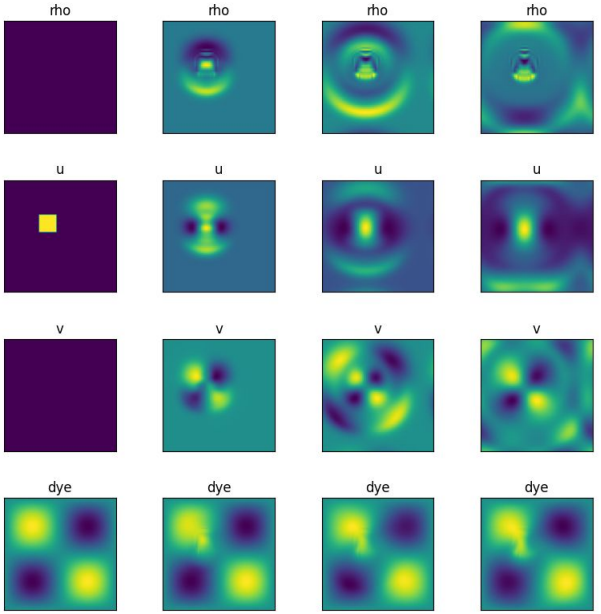
$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix} \quad E = \begin{bmatrix} \rho u \\ \rho u^2 + p - \tau_{xx} \\ \rho uv - \tau_{xy} \\ (E+p)u - u\tau_{xx} - v\tau_{xy} + q_x \end{bmatrix} \quad F = \begin{bmatrix} \rho v \\ \rho uv - \tau_{xy} \\ \rho v^2 + p - \tau_{yy} \\ (E+p)v - u\tau_{xy} - v\tau_{yy} + q_y \end{bmatrix}$$

$$\tau_{xx} = \mu\left(2\frac{\partial u}{\partial x} - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\right) \quad \tau_{xy} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \quad \tau_{yy} = \mu\left(2\frac{\partial v}{\partial y} - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\right)$$

$$q_x = -\lambda\frac{\partial T}{\partial x} \quad q_y = -\lambda\frac{\partial T}{\partial y}$$

$$\bar{U}_{i,j}^{n+1} = U_{i,j}^n - \frac{\Delta t}{\Delta x}\left(E_{i+1,j}^n - E_{i,j}^n\right) - \frac{\Delta t}{\Delta y}\left(F_{i,j+1}^n - F_{i,j}^n\right)$$

$$U_{i,j}^{n+1} = \frac{1}{2}\left[\left(U_{i,j}^n + \bar{U}_{i,j}^{n+1}\right) - \frac{\Delta t}{\Delta x}\left(\bar{E}_{i,j}^{n+1} - \bar{E}_{i-1,j}^{n+1}\right) - \frac{\Delta t}{\Delta y}\left(\bar{F}_{i,j}^{n+1} - \bar{F}_{i,j-1}^{n+1}\right)\right]$$



- No success at inverse problem

Attempt at Incompressible Navier Stokes

Poisson equation for stream function:

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\zeta$$

Vorticity Transport Equation:

$$\frac{\partial \zeta}{\partial t} + u\frac{\partial \zeta}{\partial x} + v\frac{\partial \zeta}{\partial y} = \nu\left(\frac{\partial^2 \zeta}{\partial x^2} + \frac{\partial^2 \zeta}{\partial y^2}\right)$$

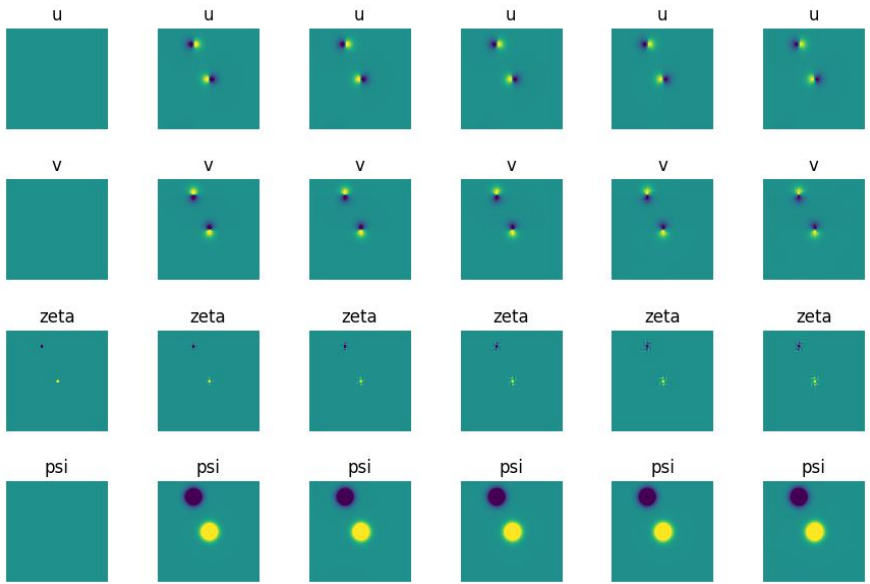
Solve with Mac Cormack:

$$E = u\zeta - \nu\frac{\partial \zeta}{\partial x} \qquad F = v\zeta - \nu\frac{\partial \zeta}{\partial y}$$

$$\bar{V}_{i,j}^{n+1} = V_{i,j}^{n} - \frac{\Delta t}{\Delta x}\left(E_{i+1,j}^{n} - E_{i,j}^{n}\right) - \frac{\Delta t}{\Delta y}\left(F_{i,j+1}^{n} - F_{i,j}^{n}\right)$$

$$V_{i,j}^{n+1} = \frac{1}{2}\left[\left(V_{i,j}^{n} + \bar{V}_{i,j}^{n+1}\right) - \frac{\Delta t}{\Delta x}\left(\bar{E}_{i,j}^{n+1} - \bar{E}_{i-1,j}^{n+1}\right) - \frac{\Delta t}{\Delta y}\left(\bar{F}_{i,j}^{n+1} - \bar{F}_{i,j-1}^{n+1}\right)\right]$$

Used with pressure correction method.



Unsuccessful forward implementation 🥲

Final Remarks:

- Differentiable Physics easy and fast with JAX
- Control Sequence Optimization works surprisingly well
- CFD is hard
- PDE inverse problems could benefit from NN layers

- What are the limitations of inverting functions?
- Optimal Control in Robotics, how is it related?

Main sources:

Physics Based Deep Learning Book