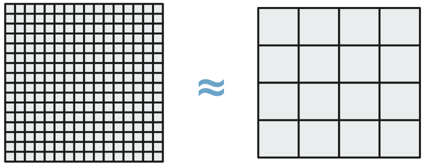
# Learning neural closure models for fluid flows

Benjamin Sanderse, Syver Agdestein
9th October
Autumn School

# Mathematical description of closure modeling

- Model for all scales $\boldsymbol{u}$ (e.g. Burgers, Navier-Stokes):

$$\frac{\partial \boldsymbol{u}}{\partial t} = \boldsymbol{f}(\boldsymbol{u})$$



$$\approx \quad + \; \boldsymbol{c}(\bar{\boldsymbol{u}};\theta)$$

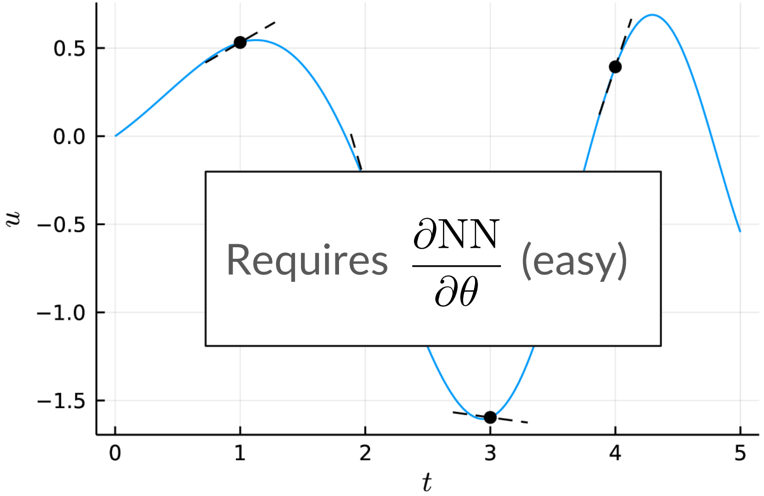
- Model for large scales $\bar{\boldsymbol{u}}$ :

$$\frac{\partial \bar{\boldsymbol{u}}}{\partial t} = \boldsymbol{f}(\bar{\boldsymbol{u}}) + \boldsymbol{c}(\bar{\boldsymbol{u}};\theta)$$

"Closure term":
- Effect of small scales on large scales
- Needs to be "discovered"
- Will be approximated by neural networks

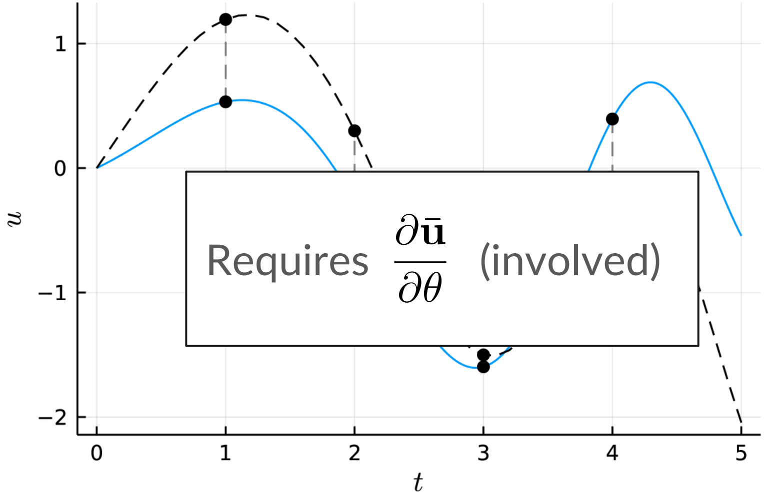
$$\frac{\mathrm{d}\bar{\mathbf{u}}}{\mathrm{d}t} = f(\bar{\mathbf{u}}) + \mathrm{NN}(\bar{\mathbf{u}};\theta)$$

## Operator fitting




$$\text{Loss} = \left\| \left(\frac{\mathrm{d}\bar{\mathbf{u}}}{\mathrm{d}t}\right)_{\mathrm{ref}} - f(\bar{\mathbf{u}}_{\mathrm{ref}}) - \mathrm{NN}(\bar{\mathbf{u}}_{\mathrm{ref}};\theta) \right\|^2$$

## Trajectory fitting




$$\text{Loss} = \sum_{i=1}^{N_t} \left\| \bar{\mathbf{u}}_{\mathrm{ref}}(t_i) - \bar{\mathbf{u}}(t_i) \right\|^2, \ \text{where } \frac{\mathrm{d}\bar{\mathbf{u}}}{\mathrm{d}t} = f(\bar{\mathbf{u}}) + \mathrm{NN}(\bar{\mathbf{u}};\theta)$$

# Our setting compared to the morning presentations

- We use supervised learning (not reinforcement learning) with differentiable solver
- We use Julia and can do everything on GPU (PDE solver + ML)
- We use an energy-conserving ML that is stable by design
- We train entire subgrid-stress term (no eddy viscosity assumption)
- We show Burgers, you can do Navier-Stokes yourself with the provided codes

# Key learning points for today

- Learn how extend a PDE with a neural network term, and train it
- Understand the idea of differentiable programming and the two options of learning: operator fitting or trajectory fitting
- Experience the benefits of working in a language like Julia
- Get triggered to use our incompressible Navier Stokes Julia code

# Cheat sheet

DNS: $$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{u})$$

Filter: $$\bar{\boldsymbol{u}} = \Phi \boldsymbol{u}$$

LES exact: $$\frac{\mathrm{d}\bar{\boldsymbol{u}}}{\mathrm{d}t} = \boldsymbol{f}(\bar{\boldsymbol{u}}) + \underbrace{\bar{\boldsymbol{f}}(\boldsymbol{u}) - \boldsymbol{f}(\bar{\boldsymbol{u}})}_{\boldsymbol{c}(\boldsymbol{u},\bar{\boldsymbol{u}})}$$

LES with NN: $$\frac{\mathrm{d}\bar{\boldsymbol{v}}}{\mathrm{d}t} = \boldsymbol{f}(\bar{\boldsymbol{v}}) + \boldsymbol{m}(\bar{\boldsymbol{v}}, \theta)$$

LES no closure: $$\frac{\mathrm{d}\bar{\boldsymbol{v}}}{\mathrm{d}t} = \boldsymbol{f}(\bar{\boldsymbol{v}})$$

Operator fitting: $$\boldsymbol{m}(\bar{\boldsymbol{u}}, \theta) \approx \boldsymbol{c}(\boldsymbol{u}, \bar{\boldsymbol{u}})$$

→ A-priori loss function: $$L^{\text{prior}}(\theta) = \sum \|\boldsymbol{m}(\bar{\boldsymbol{u}}, \theta) - \boldsymbol{c}(\boldsymbol{u}, \bar{\boldsymbol{u}})\|^2 + \lambda \|\theta\|^2$$

Trajectory fitting: $$\bar{\boldsymbol{v}}_\theta \approx \bar{\boldsymbol{u}}$$

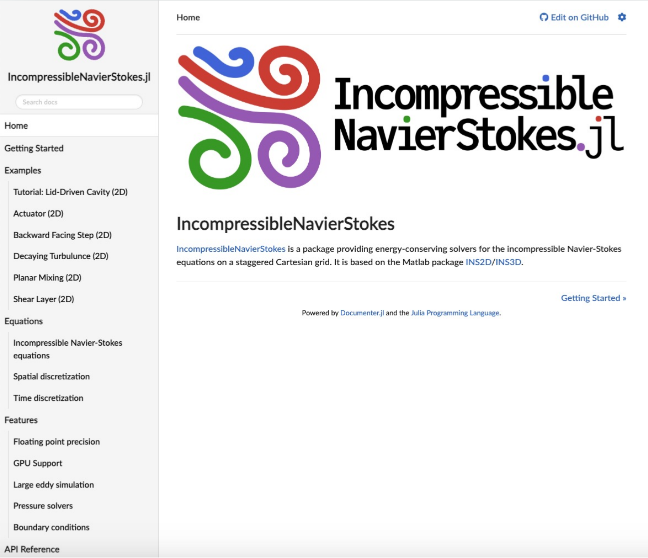
→ A-posteriori loss function: $$L^{\text{post}}(\theta) = \sum \|\bar{\boldsymbol{v}}_\theta - \bar{\boldsymbol{u}}\|^2$$

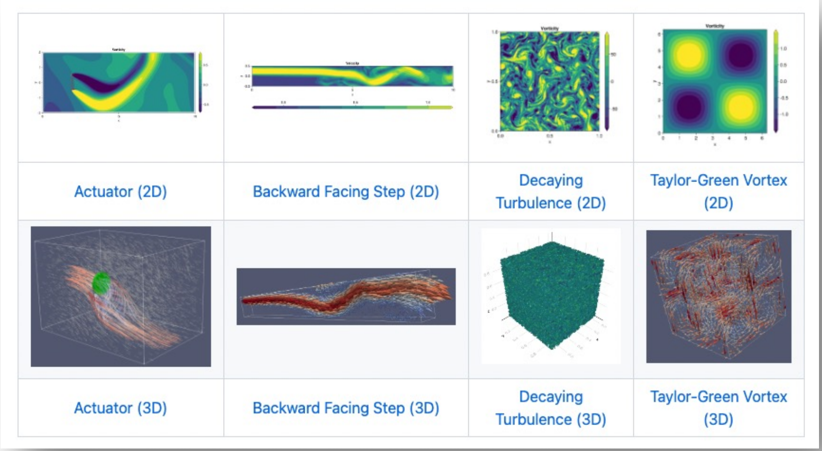
Parameters to play with:

- DNS: resolution `nx`, viscosity $\mu$, time span `nt`, time step `dt`, discretization method (`f_central`, `f_shock`)
- LES: resolution, filter choice $\Phi$
- Training: training/testing/validation data set, closure model (FNO or CNN), architecture, hyperparameters, optimizer

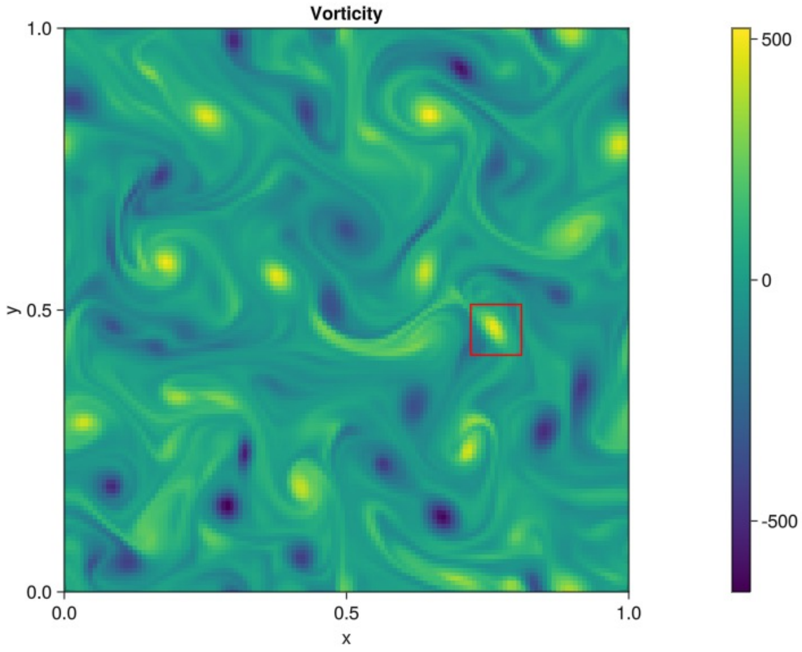
# Software: fluid flow simulator

- Computational fluid dynamics code written in Julia
- Github: https://github.com/agdestein/IncompressibleNavierStokes.jl
- 2D/3D
- Automatic differentiation with Zygote
- CPU and GPU implementation
- Range of test cases

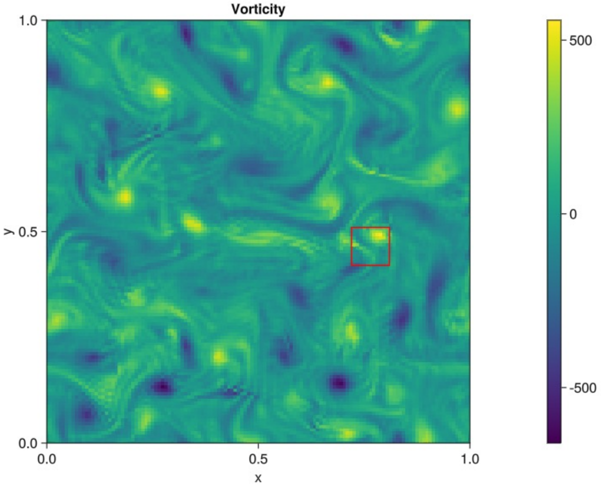


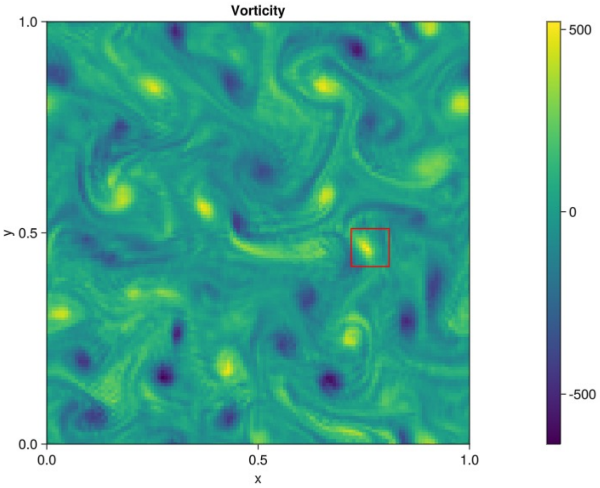
# Example results (2D)




Reference




No model




FNO