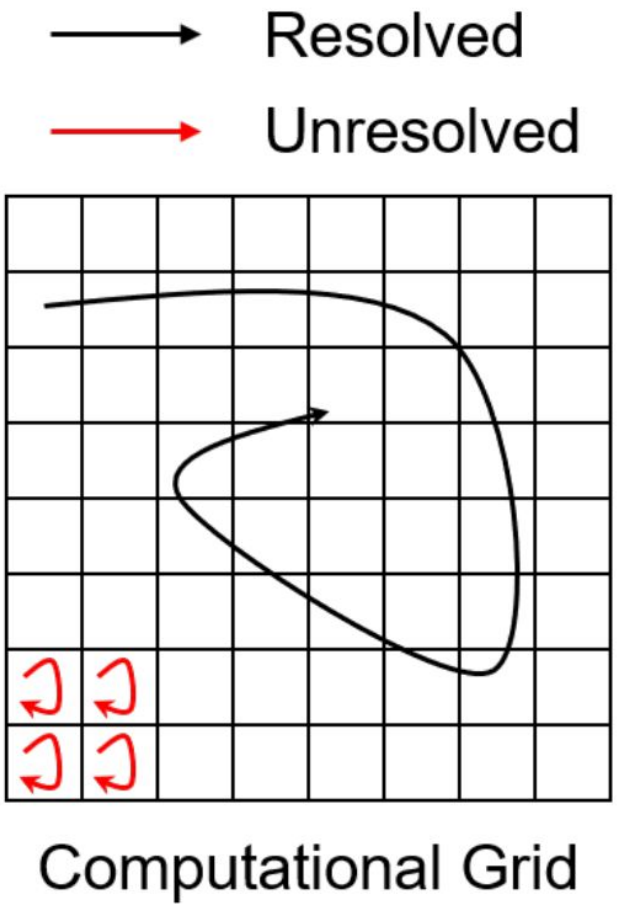
Turbulence modeling for large eddy simulations using data

Romit Maulik
Assistant Professor
Information Science and Technology, Pennsylvania State University
&
Joint Appointment Faculty,
Mathematics and Computer Science Division, Argonne National Laboratory

Large eddy simulations (LES)

LES attempts to reduce resolution requirement by modeling unresolved eddies.



- Resolved scales from 'grid-filtered' Navier-Stokes.
- Effect of unresolved scales from turbulence model.

Resolved and unresolved scales interact nonlinearly

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{1}{Re} \frac{\partial^2 u_i}{\partial x_j \partial x_j}$$

Model choice affects statistics significantly

Model assessment?

- Compare LES statistics to fully resolved Navier-Stokes (DNS).
- Compare LES statistics to experimental observations.

Closure modeling for LES

LES models are applied as a source term to the 'grid-filtered' Navier-Stokes equations.

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} = -\frac{\partial \bar{P}}{\partial x_i} + \frac{1}{Re}\frac{\partial^2 \bar{u}_i}{\partial x_j \partial x_j} + \frac{\partial \tau_{ij}^R}{\partial x_j}$$

Structural closures

Source term approximated as:

$$\tau_{ij}^* = \bar{u}_i \bar{u}_j - \widetilde{u_i^* u_j^*},$$

$$\tilde{u}_i = G * u_i.$$

Layton scale-similarity model:

$$\tau_{ij}^* = \bar{u}_i \bar{u}_j - \widetilde{\bar{u}_i \bar{u}_j}.$$

Approximate deconvolution (AD)[2]:

$$u_i^{*\,0} = \bar{u}_i,$$

$$u_i^{*\,i} = u_i^{*\,i-1} + \left(\bar{u}_i - G * u_i^{*\,i-1}\right), \quad i = 1, 2, 3, \ldots, Q.$$

Key limitation: Definition of G.

Functional closures

Source term approximated as:

$$\tau_{ij}^R = 2\nu_e \bar{S}_{ij},$$

Standard Smagorinsky model:

$$\nu_e = (C_s \bar{\Delta})^2 |\bar{S}|,$$

Dynamic Smagorinsky model[3]:

$$C_s^2 = \frac{1}{2}\frac{\mathbb{L}_{kl}^R \bar{S}_{kl}}{\mathbb{M}_{mn}\bar{S}_{mn}}$$

$$\mathbb{L}_{ij}^R = 2C_s^2\left(\tilde{\bar{\Delta}}^2 |\tilde{\bar{S}}|\tilde{\bar{S}}_{ij} - \bar{\Delta}^2 \widetilde{|\bar{S}_{ij}|\bar{S}_{ij}}\right) = 2C_s^2 \mathbb{M}_{ij}.$$

Key limitation: Definition of $\tilde{\bar{\Delta}}$ and test-filter.

[2] S. Stolz and N. A. Adams. In: Phys. Fluids 11 (1999), pp. 1699-1701.

[3] M. Germano et al. In: Phys. Fluids 3 (1991), pp. 1760-1765.

Data-driven closure modeling

- There is some evidence for the universality of smaller scales in particular flow classes.[5]
- Let coarse-grained Navier-Stokes evolve large-scale structures and let data-driven ideas model sub-grid scales.
- Build generalizable closures through physics-discerning machine learning formulations across different classes.

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} = -\frac{\partial \bar{P}}{\partial x_i} + \frac{1}{Re}\frac{\partial^2 \bar{u}_i}{\partial x_j \partial x_j} + \frac{\partial \tilde{\tau}_{ij}^R}{\partial x_j}$$

Try and predict $\tilde{\tau}_{ij}^R$ using a machine learning framework - three-dimensional turbulence

$$\frac{\partial \bar{\omega}}{\partial t} + J(\bar{\omega}, \bar{\psi}) = \frac{1}{Re}\nabla^2 \bar{\omega} + \tilde{\Pi}$$

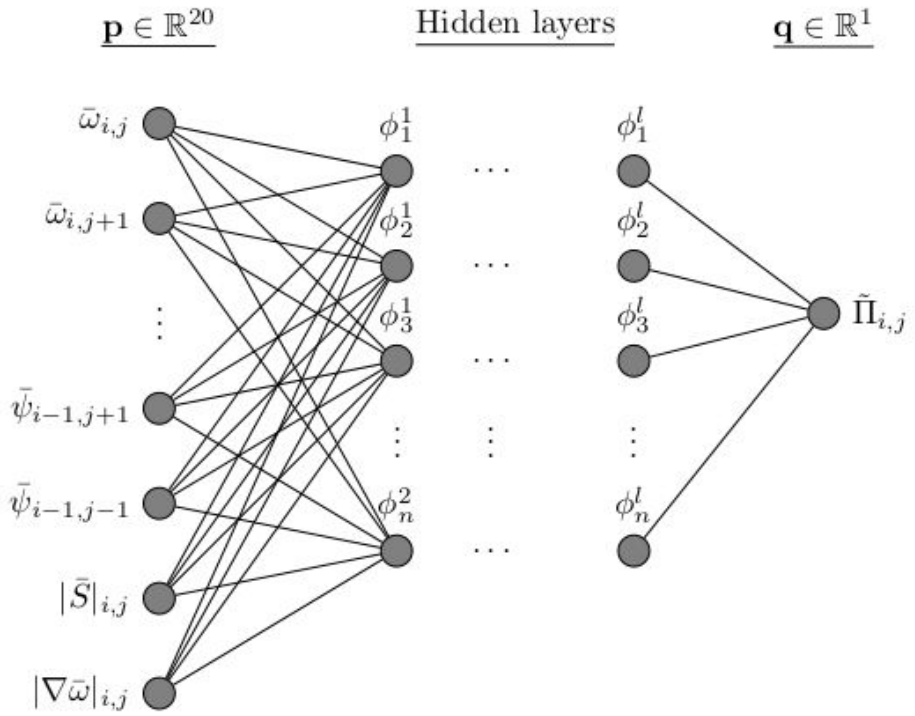
Try and predict $\tilde{\Pi}$ using a machine learning framework - two-dimensional turbulence

[5] Vreman In: Phys. Fluids, 16, 3670-3681.

Data-driven closure modeling

Why not directly predict $\tilde{\Pi}$?

- A single map $\mathbb{M}$ relates an input space to the sub-grid quantity $\tilde{\Pi}$.
- Input space consists of grid-resolved quantities in stencil for data-locality.
- Add grid-resolved information in the form of two extra inputs (the Smagorinsky and Leith eddy-viscosity kernels).[8]



$$\mathbb{M}^{20} : \{\bar{\omega}_{i,j}, \bar{\omega}_{i,j+1}, \bar{\omega}_{i,j-1}, \ldots, \bar{\omega}_{i-1,j-1}, \bar{\psi}_{i,j}, \bar{\psi}_{i,j+1}, \bar{\psi}_{i,j-1}, \ldots, \bar{\psi}_{i-1,j-1}, |\bar{S}|_{i,j}, |\nabla\bar{\omega}|_{i,j}\} \in \mathbb{R}^{20} \rightarrow \{\tilde{\Pi}_{i,j}\} \in \mathbb{R}^{1}.$$

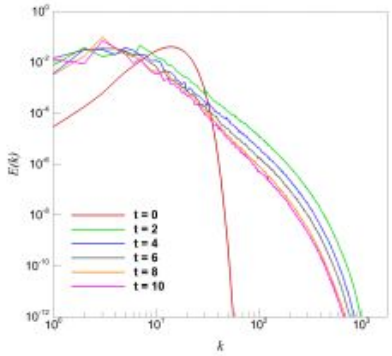
$\overline{\Pi}$ treated as a grid-filtered variable and obtained from DNS.

[8] Maulik et al. in: J. Fluid Mech., 858, 122-144 (2019).

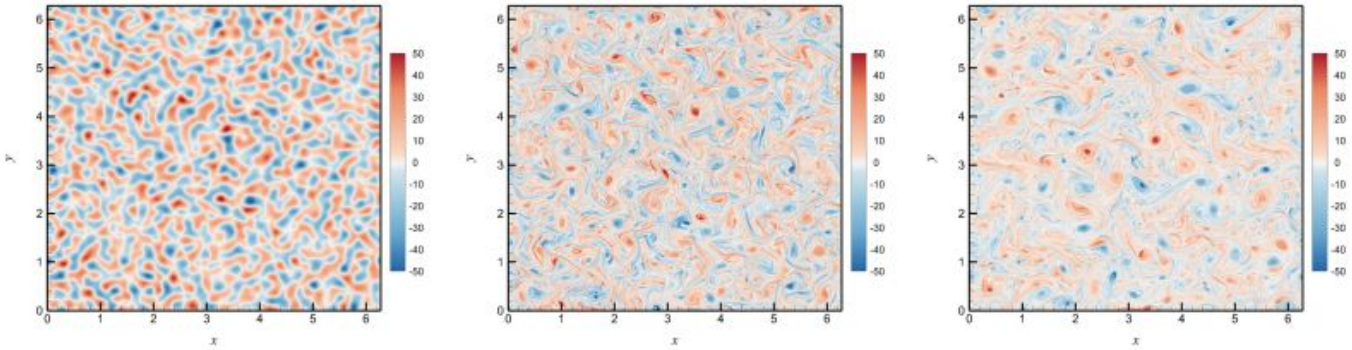
A benchmark problem

- Two-dimensional (Kraichnan) turbulence chosen since turbulence closures for LES are generally devised for three-dimensional turbulence.
- Kraichnan turbulence reveals important characteristics of flows with high aspect-ratios.
- Fundamentally based on the cascade of enstrophy (k^{-3} scaling).

$$\frac{\partial \bar{\omega}}{\partial t} + J(\bar{\omega}, \bar{\psi}) = \frac{1}{Re}\nabla^2 \bar{\omega} + \tilde{\Pi}$$
$$\nabla^2 \bar{\psi} = -\bar{\omega}$$



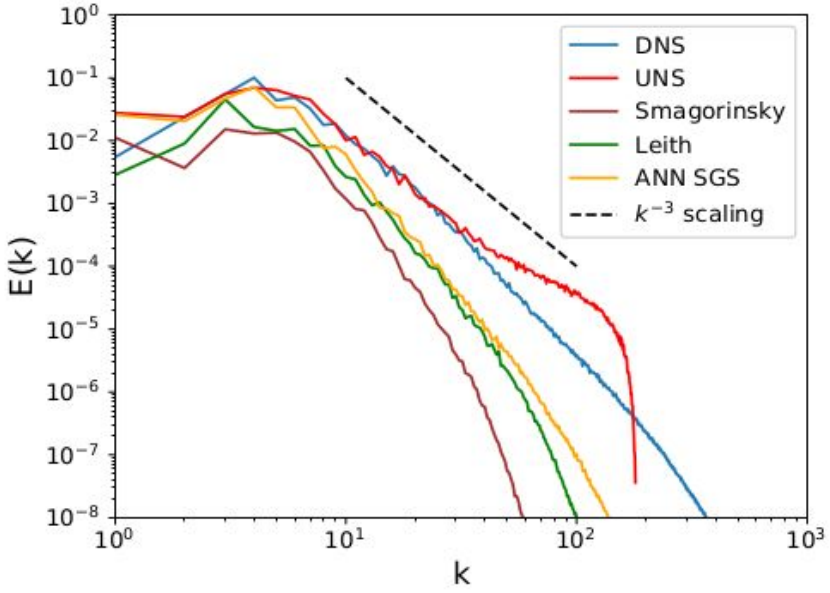
Angle-averaged energy-spectra



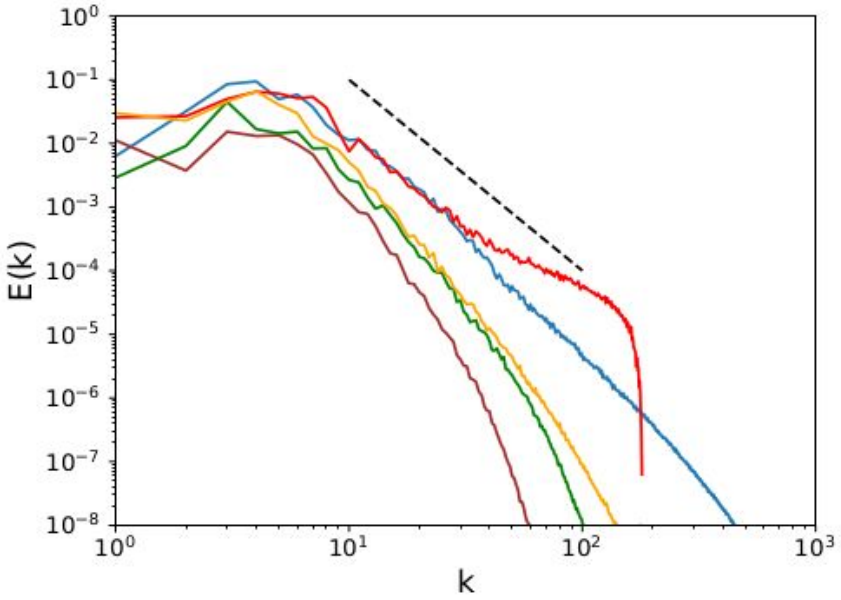
Turbulence decays with time $\rightarrow$

Results on Kraichnan turbulence

- Deploy in an a-posteriori forward simulation with associated numerical errors.
- Validated at $Re = 32000$ (similar to training data) and $Re = 64000$ for testing generalizability.
- Higher fidelity at lower wavenumbers but 'Leith' like behavior in the inertial range.



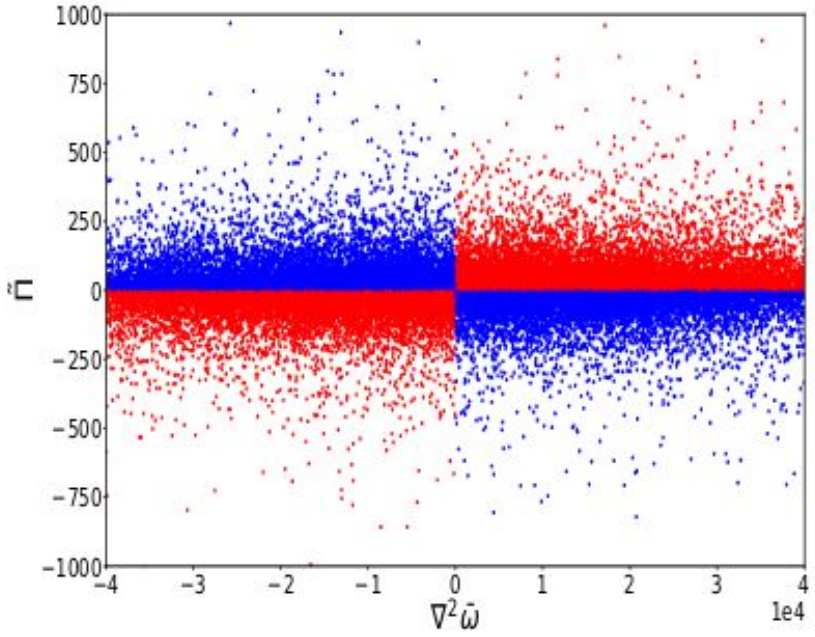
Training Reynolds number - 32000



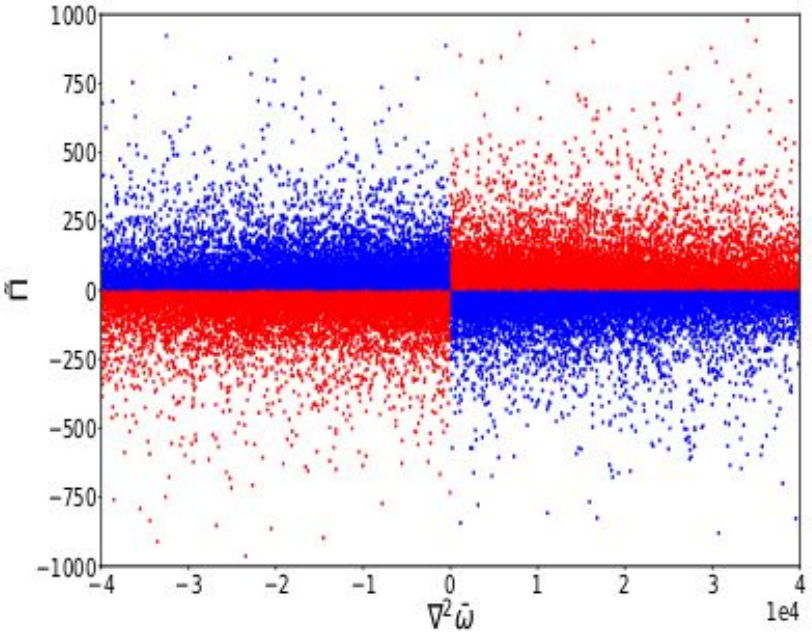
Testing Reynolds number - 64000

The issue of a-posteriori stability

- Age-old issue with DNS based closure development[9].
- Truncation of ML predictions for viscous stability and interaction with numerical errors in deployment cause perturbed trends in a-posteriori.
- Truncation can be visualized in a quadrant-based view according to viscous stability requirements



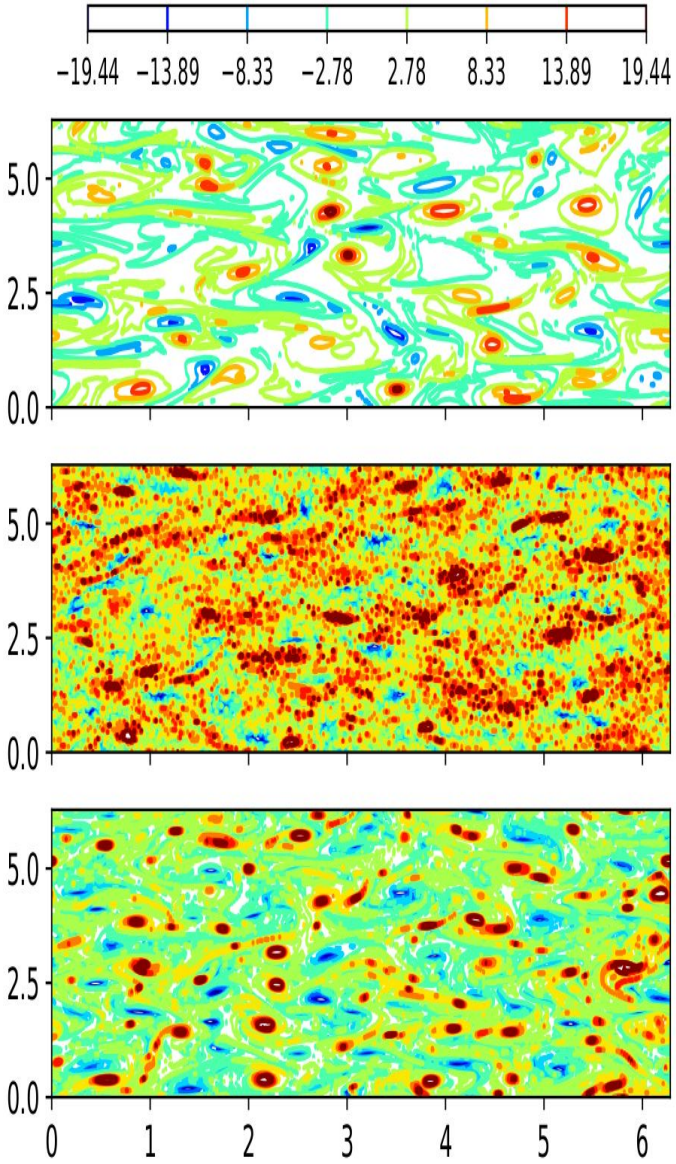
$Re = 32000$



$Re = 64000$

[9] Piomelli et al. In: Phys. Fluids, 7 PP. 839-848

Results on Kraichnan turbulence



Vorticity magnitude contours

Prediction from proposed framework

No-model simulation

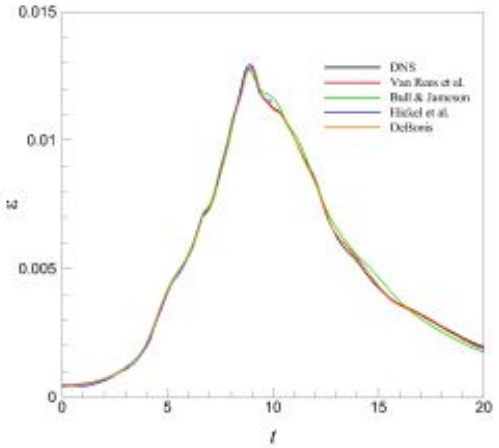
Filtered-DNS

All-in-all - not very satisfactory

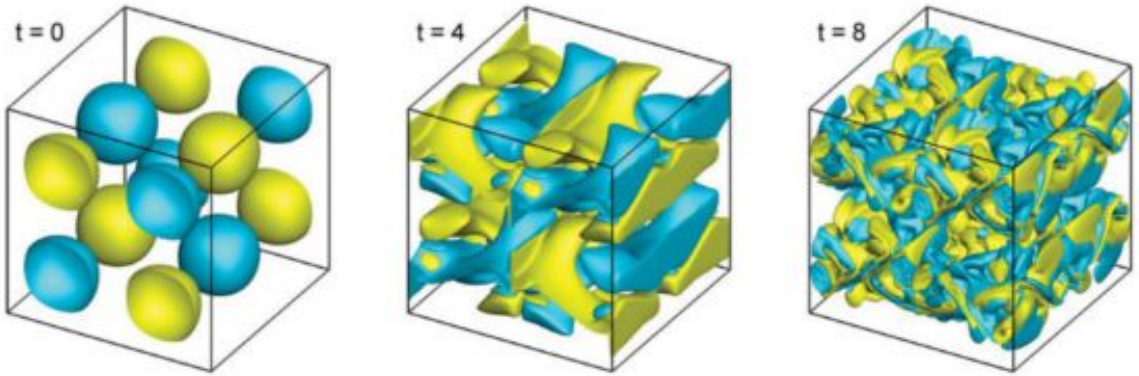
Extension to three-dimensions

- Three-dimensional Taylor-Green Vortex turbulence chosen for validation of our proposed ideas in classical turbulence modeling applications
- This problem displays an unsteady decaying turbulence behavior with vortex stretching
- Fundamentally based on the cascade of kinetic energy ($k^{-5/3}$ scaling).

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} = -\frac{\partial \bar{P}}{\partial x_i} + \frac{1}{Re}\frac{\partial^2 \bar{u}_i}{\partial x_j \partial x_j} + \frac{\partial \tilde{\tau}_{ij}^R}{\partial x_j}$$



Validation (Energy dissipation rate)



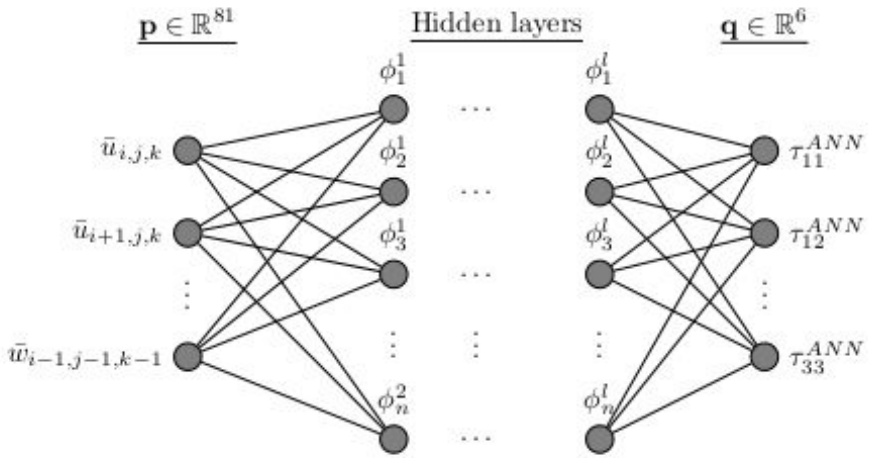
Vortex stretching →

Extension to three-dimensions

Use local stencil of grid resolved variables to predict τ_{ij}[10].

$$\tau_{ij} = \overline{u_i u_j} - \bar{u}_i \bar{u}_j$$

generated from grid-filtered DNS of Kolmogorov turbulence (4 snapshots temporally).



Added - projection for stability.

The linear eddy-viscosity hypothesis is utilized for a-posteriori projection of τ_{ij}^{ANN}.
This is to ensure that the predictions have uni-directional energy transfer (super-grid to sub-grid).

$$\Pi_{ij} = \frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} - \frac{2}{3}\frac{\partial \bar{u}_k}{\partial x_k}\delta_{ij}$$

$$J(\nu_e) = \sum_{i=1}^{3}\sum_{j=1}^{3}(\tau_{ij}^{ANN} - \nu_e \Pi_{ij})^2$$

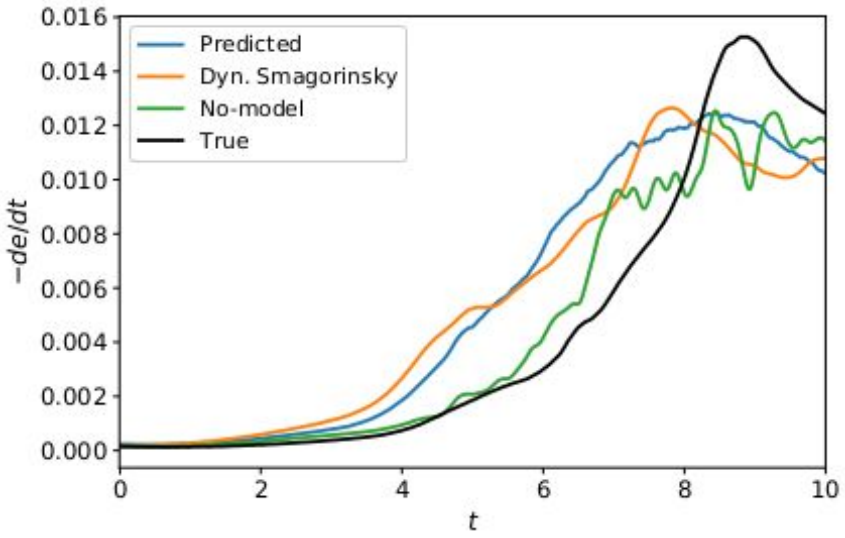
$$\nu_e = \sum_{i=1}^{3}\sum_{j=1}^{3}\frac{\tau_{ij}^{ANN}\Pi_{ij}}{\Pi_{ij}\Pi_{ij}}$$

Projection is truncated to ensure only positive eddy-viscosities (rather ad-hoc but stable)

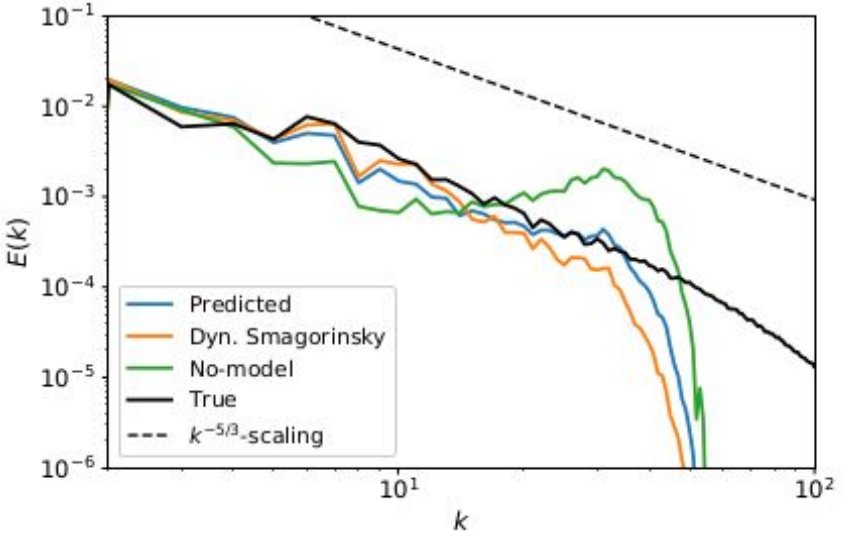
A-priori training requires a lot of ad-hoc treatments for a-posteriori stability and accuracy

Extension to three-dimensions

Trained on $Re = 1600$ with DNS snapshots at $t = 5, 10, 15, 20$. Deployed at $Re = 5000$. The linear-eddy viscosity projection ensures time, pressure and Galilean invariance (Silvis et al., Phys. Fluids, 2017).



Energy dissipation rate



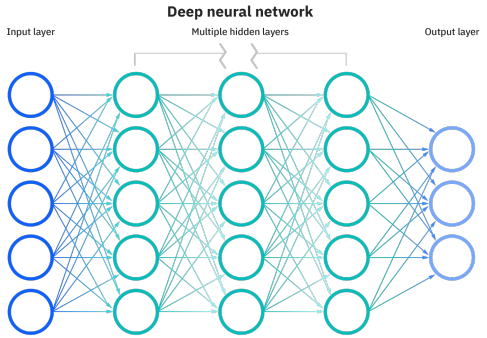
Spectra

A-posteriori turbulence modeling

Background: Neural ordinary differential equations

Given snapshots of *u* and an assumption of data being generated from autonomous systems - our goal is to identify *f* in:

$$\frac{du}{dt} = f(u(t))$$



With snapshots of training data

$$\{u_0, u_1, \ldots, u_M\}$$

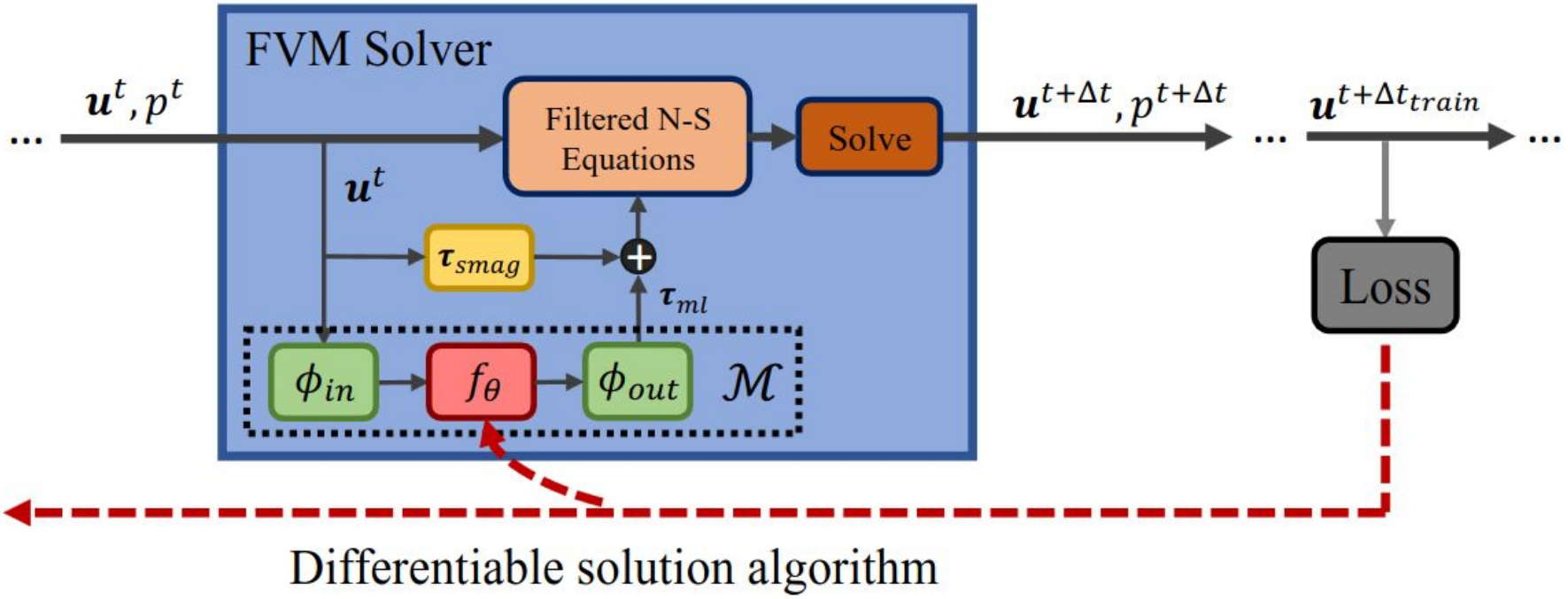
Using a loss-function as follows for various \tau:

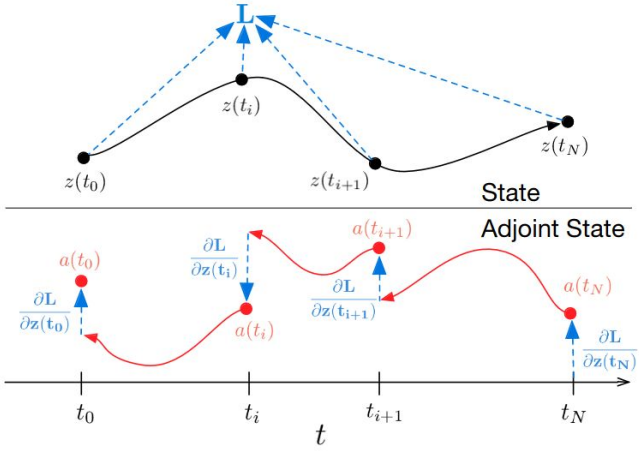
$$\left\langle \sum_{j=1}^{K} ||u_{i+j} - u(t_{i+j})|| \right\rangle$$

True trajectory

Predicted trajectory

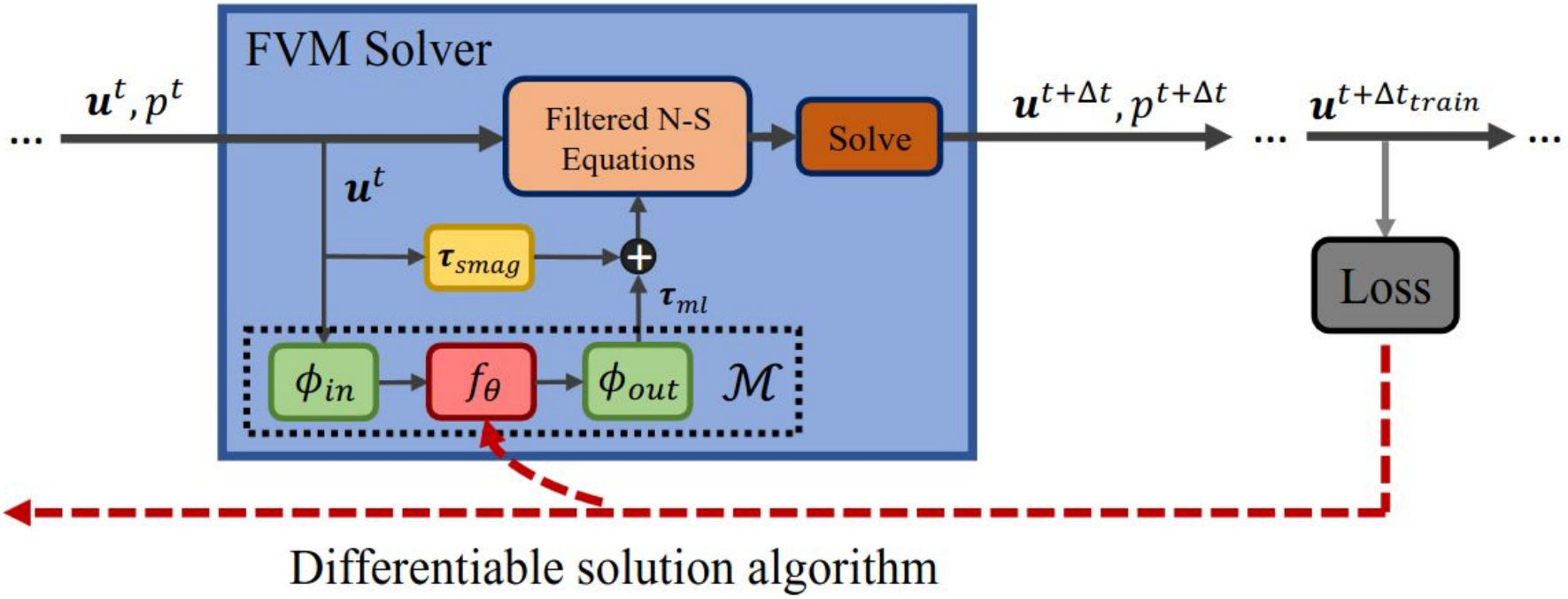
The *differentiable physics* approach



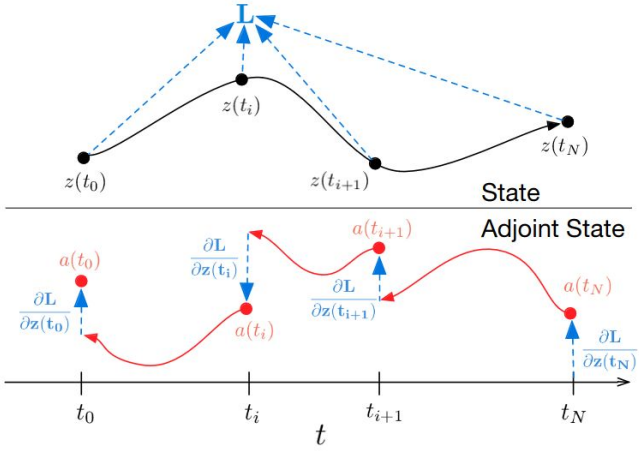


Goal:
Leverage rich developments in adjoint-based techniques for numerical solutions of partial differential equations.

The *differentiable physics* approach



Differentiable solution algorithm

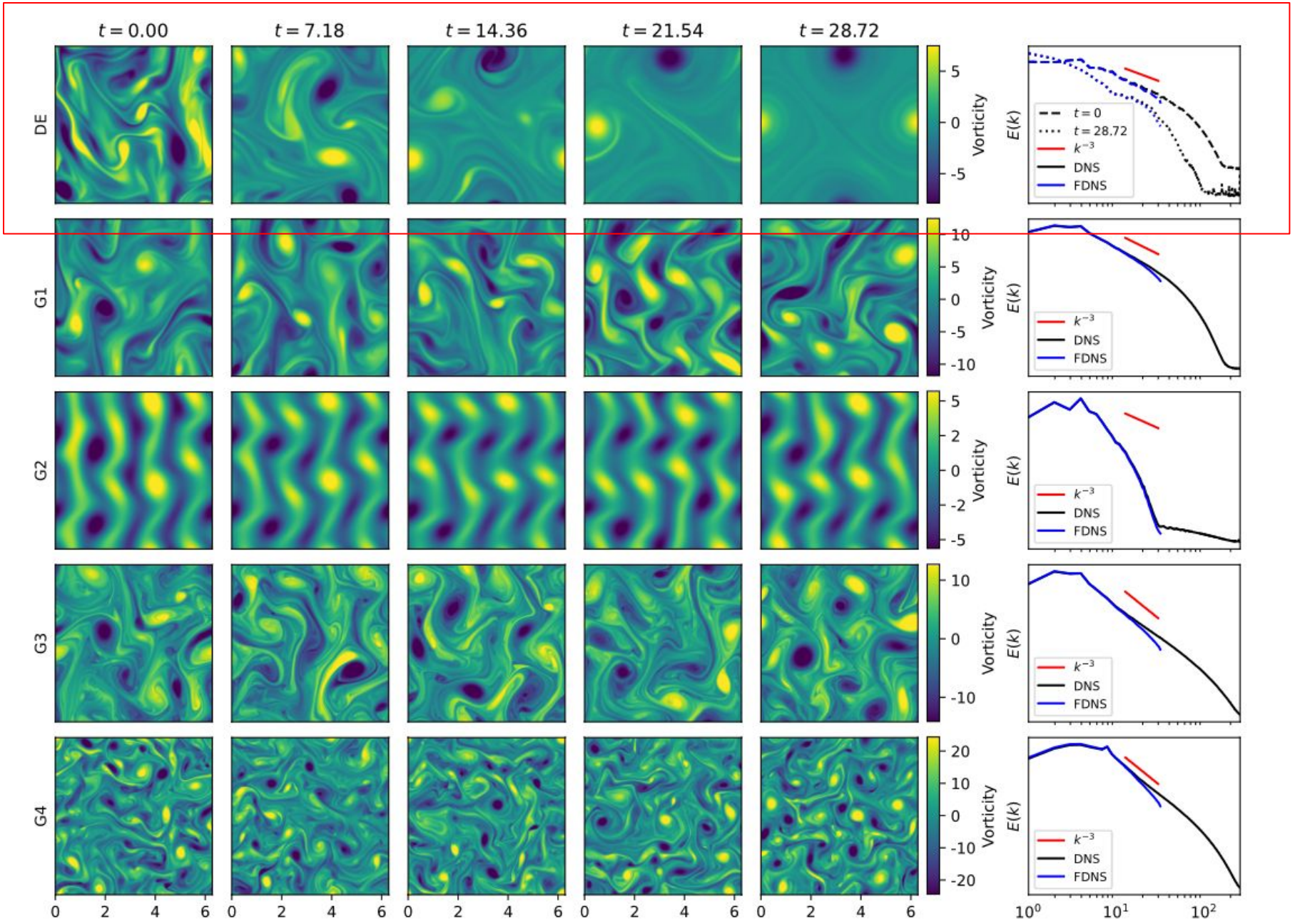


$$\hat{\overline{\mathbf{u}}}^0 \dots \hat{\overline{\mathbf{u}}}^t = \mathcal{S}_{ML}(\overline{\mathbf{u}}^0, C_s, \mathcal{M}(\theta)),$$

$$\overline{\mathbf{u}}^0 \dots \overline{\mathbf{u}}^t = G_\Delta \star \mathcal{S}_{DNS}(\mathbf{u}^0),$$

$$\min_{C_s,\theta} \mathcal{L}(\overline{\mathbf{u}}^0 \dots \overline{\mathbf{u}}^t, \mathcal{S}_{ML}(\overline{\mathbf{u}}^0, C_s, \mathcal{M}(\theta))),$$

Assessments (a *bit* more challenging now)



Assessments

$$\nu_t = (C_s\Delta)^2|\overline{\mathbf{S}}|$$
$$\boldsymbol{\tau}_{smag} = -2\nu_t\overline{\mathbf{S}},$$
$$\hat{\boldsymbol{\tau}} = \boldsymbol{\tau}_{smag} + \boldsymbol{\tau}_{ml},$$
$$\boldsymbol{\tau}_{ml} = \mathcal{M}(\overline{\mathbf{u}}, f_\theta, \phi_{in}, \phi_{out}),$$

$$\phi_{in}(\overline{\mathbf{u}}) = \left\{\overline{\mathbf{S}}^2\right\}, \left\{\overline{\mathbf{R}}^2\right\}$$
$$f_\theta\left(\left\{\overline{\mathbf{S}}^2\right\}, \left\{\overline{\mathbf{R}}^2\right\}\right) = \alpha$$
$$\phi_{out}(\alpha) = \sum_{n=0}^{2} \alpha^{(n)}\mathbf{T}^{(n)},$$

TABLE I. Model names

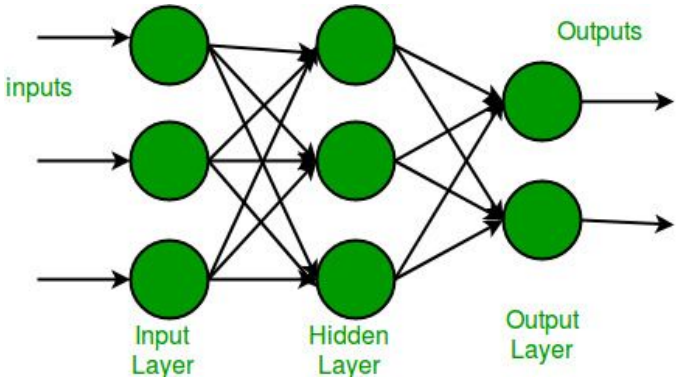
Name	ϕ_{out}
Linear (LIN)	$\sum_{n=0}^{1} \alpha^{(n)}\mathbf{T}^{(n)}$
Non-linear (NL)	$\sum_{n=0}^{2} \alpha^{(n)}\mathbf{T}^{(n)}$
Non-linear asymmetric (NLA)	$\sum_{n=0}^{3} \alpha^{(n)}\mathbf{T}^{(n)}$
Model-free (MF)	$\alpha\mathbf{I} + \mathbf{D} + \mathbf{A}$
Model-free symmetric (MFS)	$\alpha\mathbf{I} + \mathbf{D}$

Training performance assessed in a-posteriori deployments using correlations with filtered-DNS and statistical error

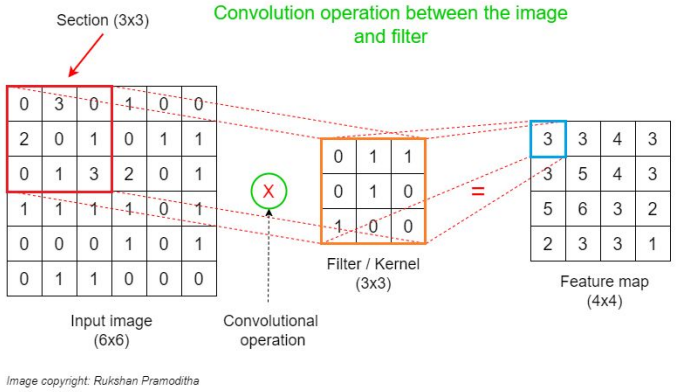
Statistical error in wavenumber space →

$$\left(\frac{1}{K}\sum_{k=1}^{K}\log\left(\frac{\hat{E}(k)}{E(k)}\right)^2\right)^{1/2},$$

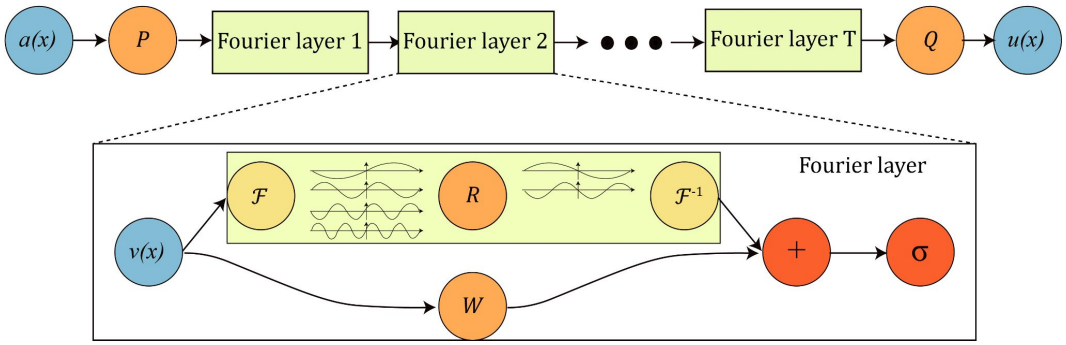
Network architectures



MLP: Perfectly local inputs for perfectly local outputs

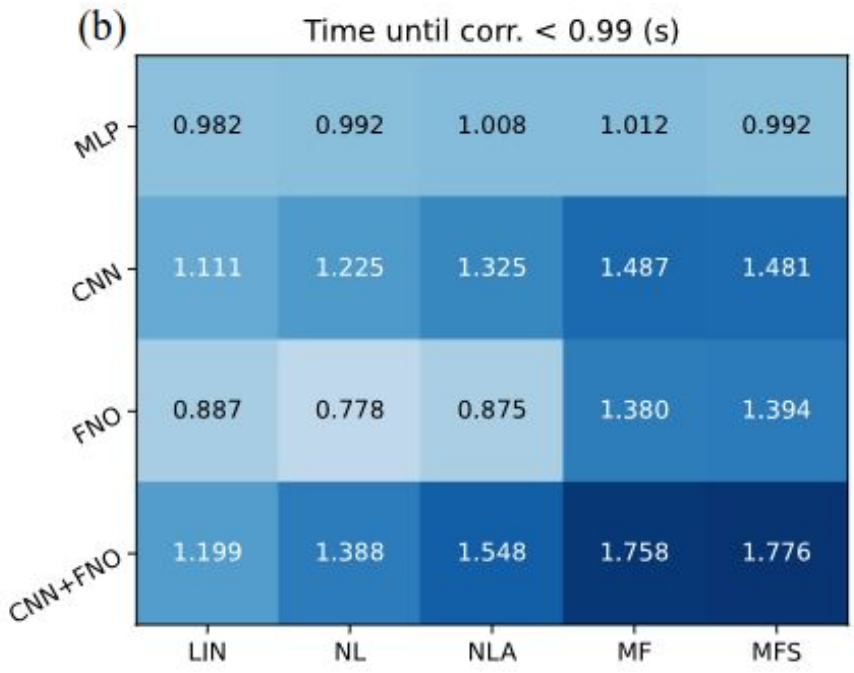


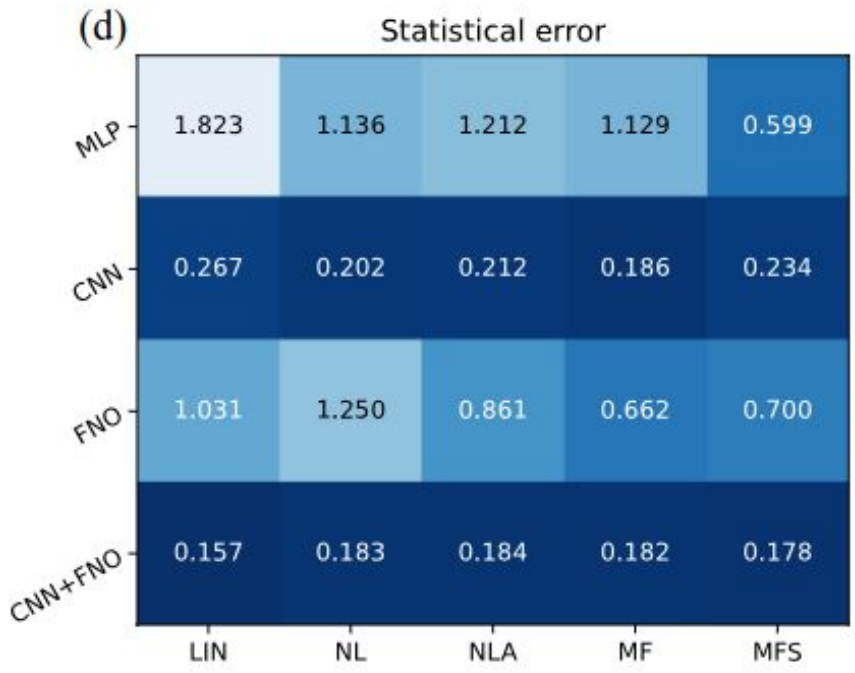
CNN: Iterative convolution operations for non-local influence on predictions - LeNet (Le-Cun)



FNO: Fourier Neural Operator that learns a function approximation that is perfectly global (Li et al.)

Assessments





On the left - the amount of time it takes to decorrelate from the DNS - higher is better.
On the right - the time-averaged statistical error after complete decorrelation - lower is better.
These statistics computed for each test case and averaged.

$$\left(\frac{1}{K} \sum_{k=1}^{K} \log \left(\frac{\hat{E}(k)}{E(k)} \right)^2 \right)^{1/2},$$

Assessments

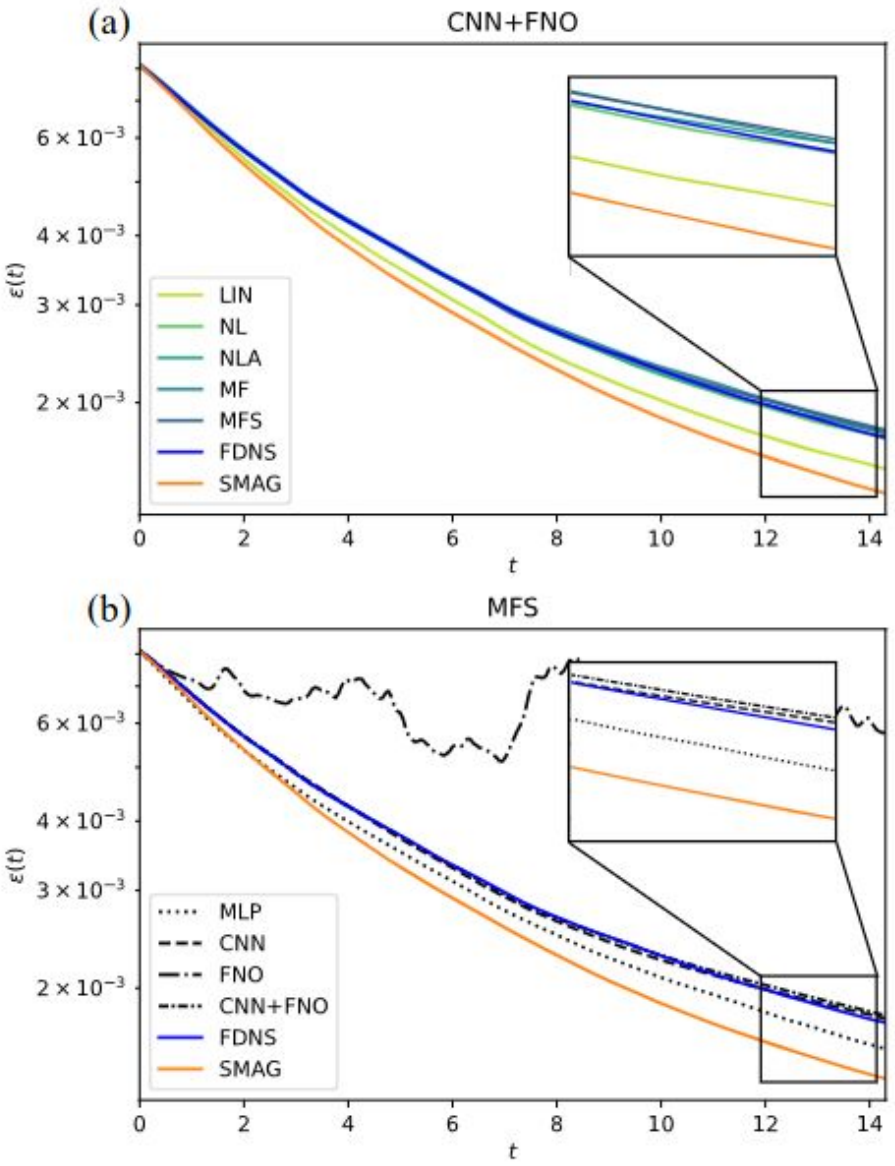
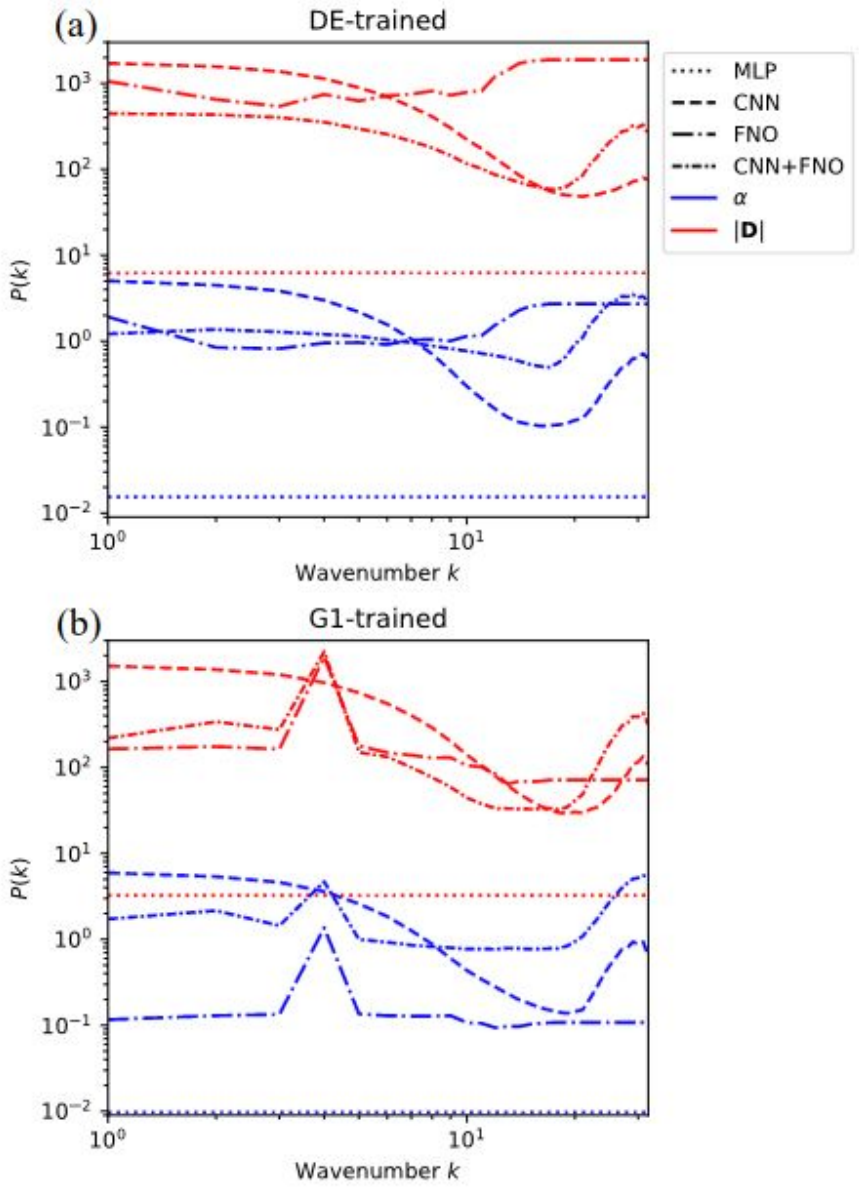


TABLE I. Model names

Name	ϕ_{out}
Linear (LIN)	$\sum_{n=0}^{1}\alpha^{(n)}\mathbf{T}^{(n)}$
Non-linear (NL)	$\sum_{n=0}^{2}\alpha^{(n)}\mathbf{T}^{(n)}$
Non-linear asymmetric (NLA)	$\sum_{n=0}^{3}\alpha^{(n)}\mathbf{T}^{(n)}$
Model-free (MF)	$\alpha\mathbf{I}+\mathbf{D}+\mathbf{A}$
Model-free symmetric (MFS)	$\alpha\mathbf{I}+\mathbf{D}$

The “MFS” model with CNN+FNO architecture for the subgrid stress defect is most accurate for our predictions - based on the energy dissipation rate.

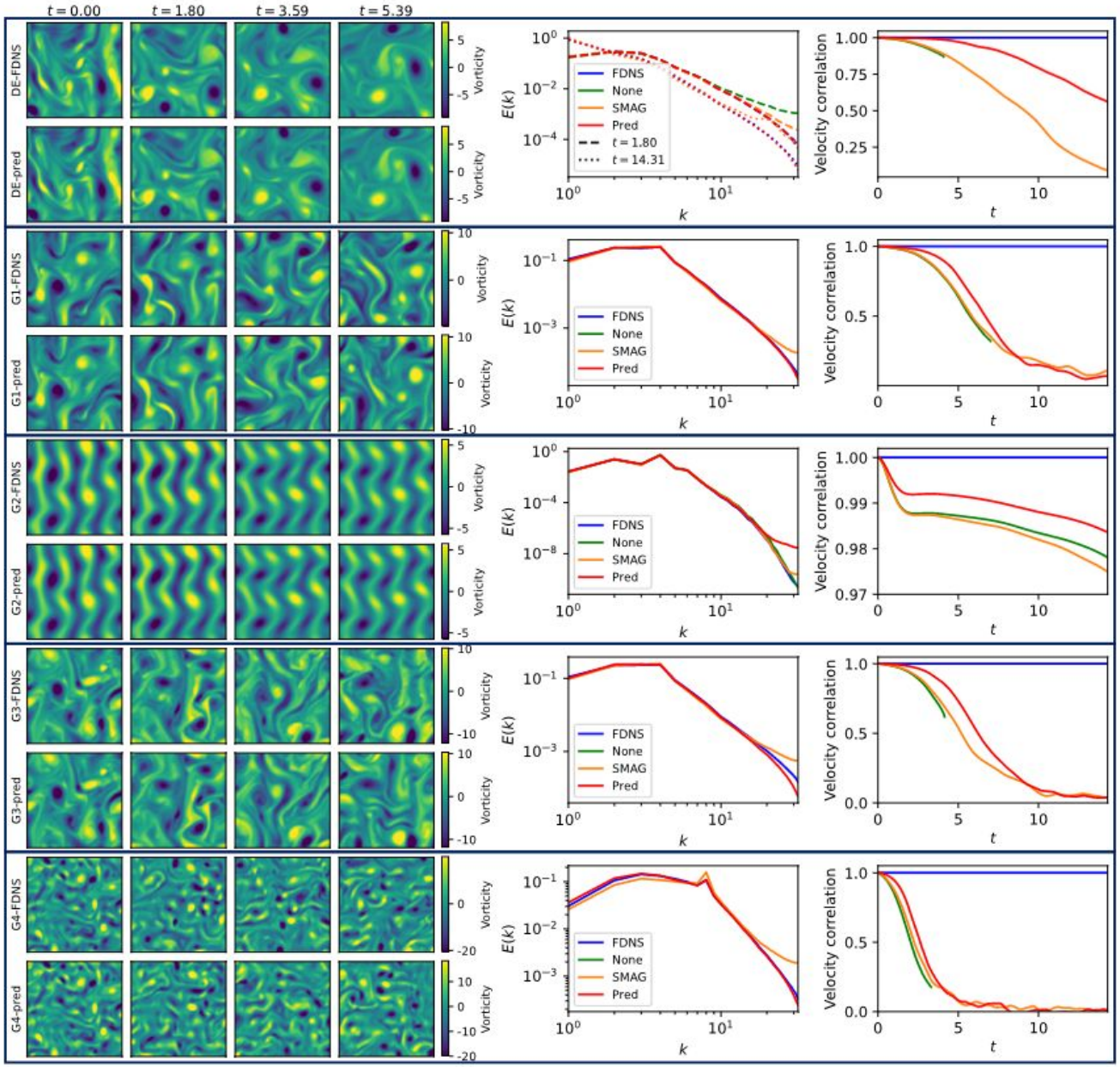
Assessments



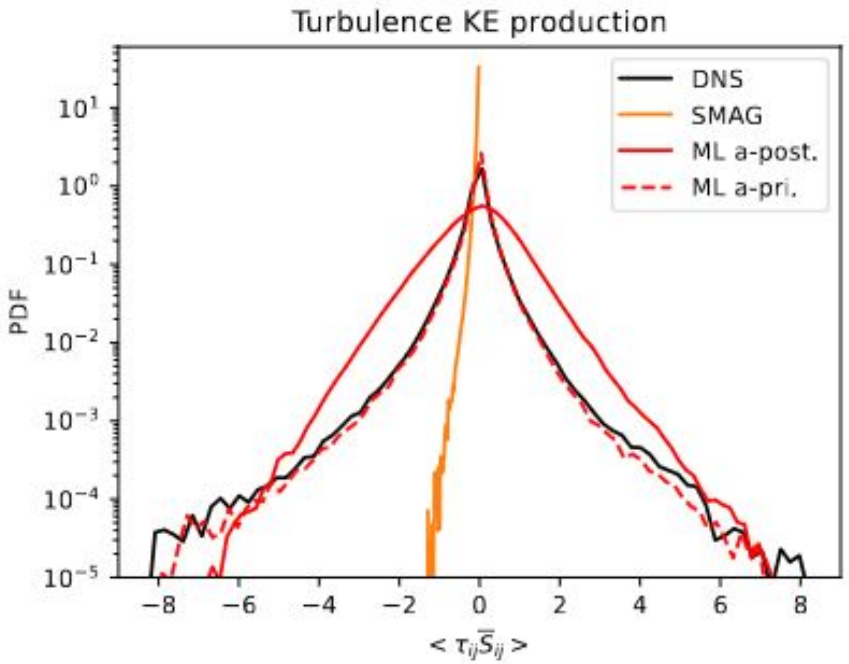
We look at power spectral densities of responses to white noise by the trained models. The first case (top) when it is trained on a decaying turbulence case. The second case (below) is when it is trained on the forced turbulence case. The colors indicate which output is being looked at in the MFS model

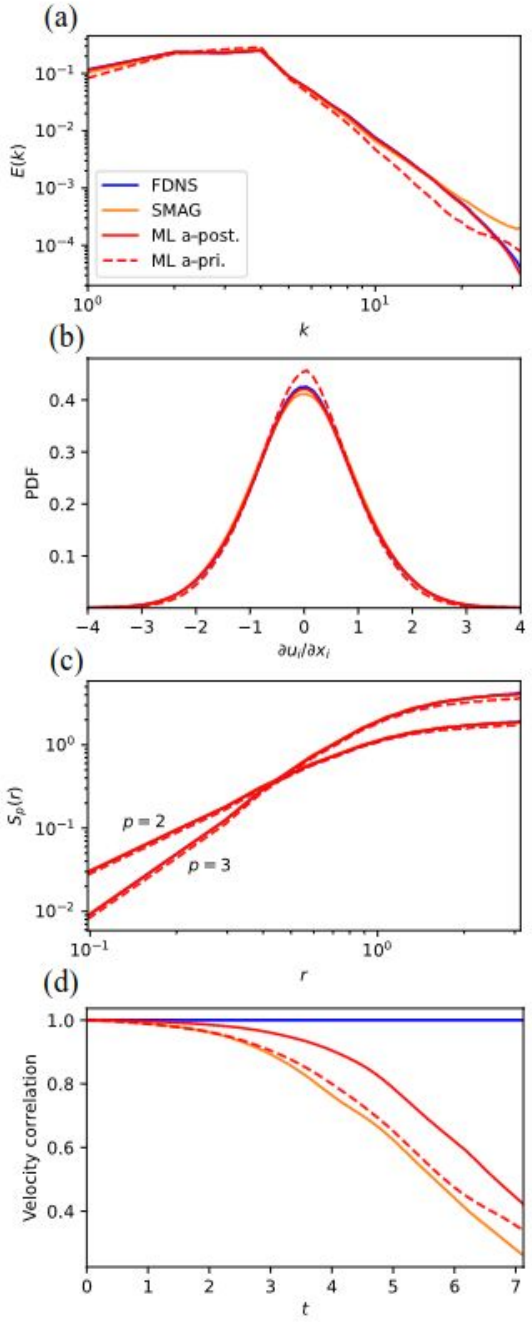
MLP responses are **flat** - this makes sense - it is purely local. CNN responses are not flat - but can't really get the peak at the forcing wavenumber. The FNO picks this up - but tends to remain flat at other wavenumbers (perhaps due to spectral truncation) - the CNN+FNO work together well.

Generalization



A-priori and a-posteriori

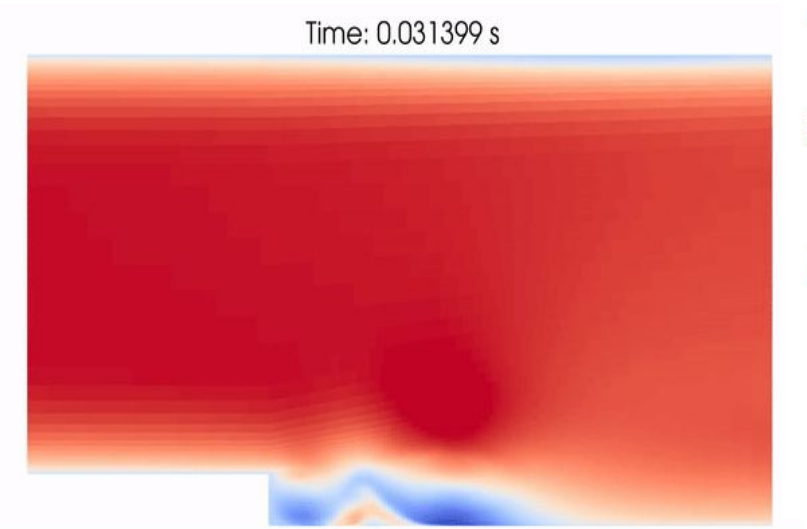


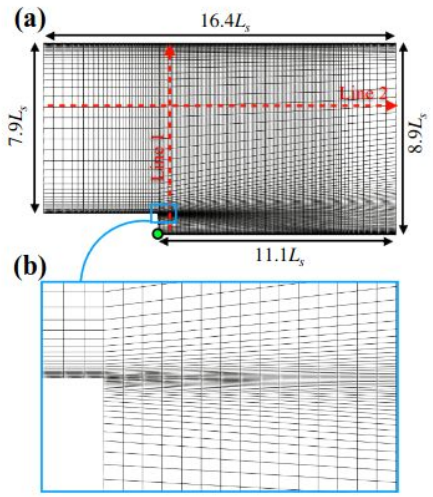


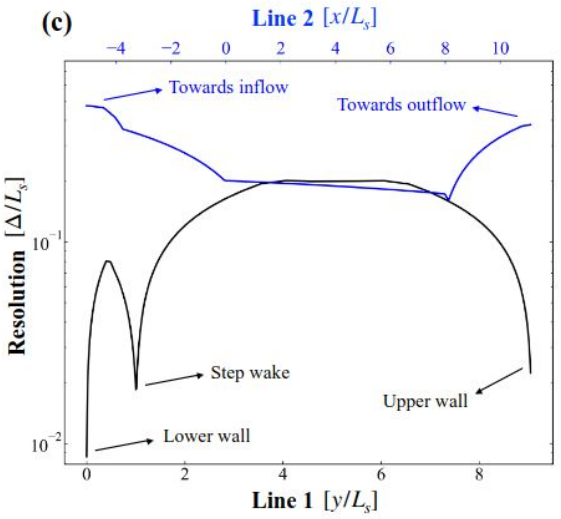
Typically most a-priori methods try to match statistics in SGS - which differentiable physics does not seem to care for. However, is it even necessary?

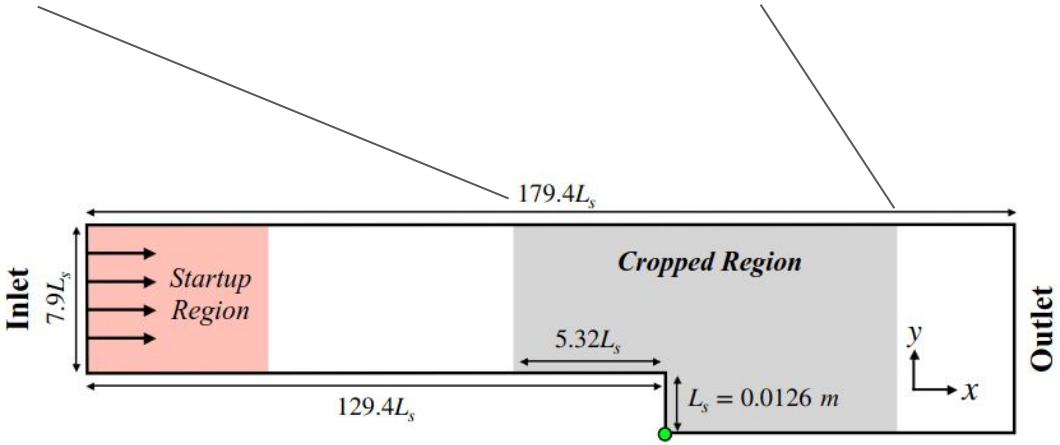
V. Shankar, R. Maulik, V. Vishwanathan: Differentiable turbulence I, arXiv:2307.03683

A more realistic benchmark



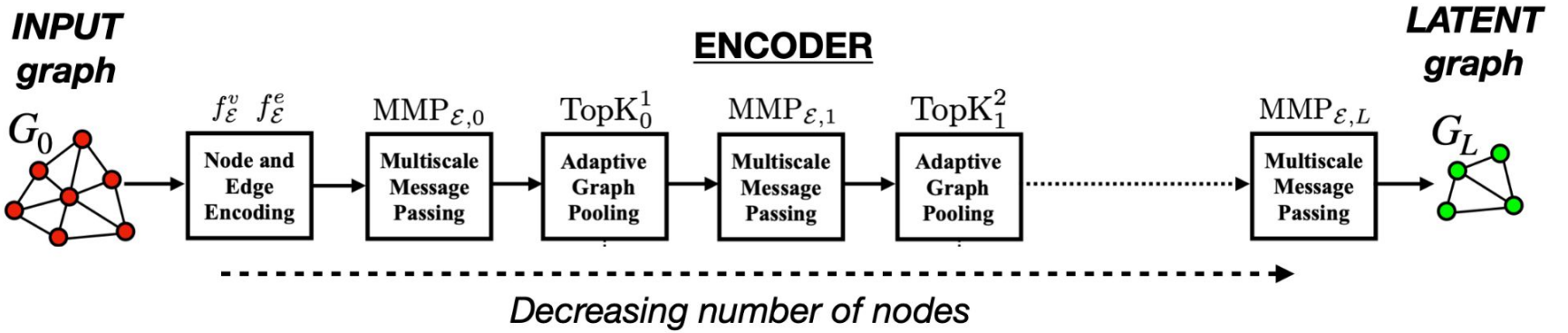


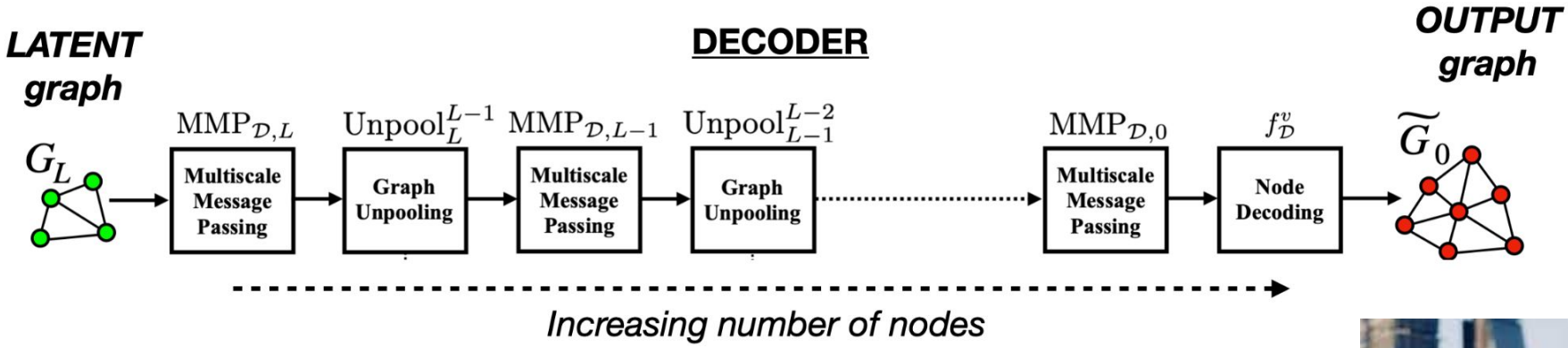




A more realistic test-case for data-driven closure modeling. Characterized by separation, anisotropy, sharp gradients -> not amenable to structured-grid neural net architecures!

A multiscale graph neural network

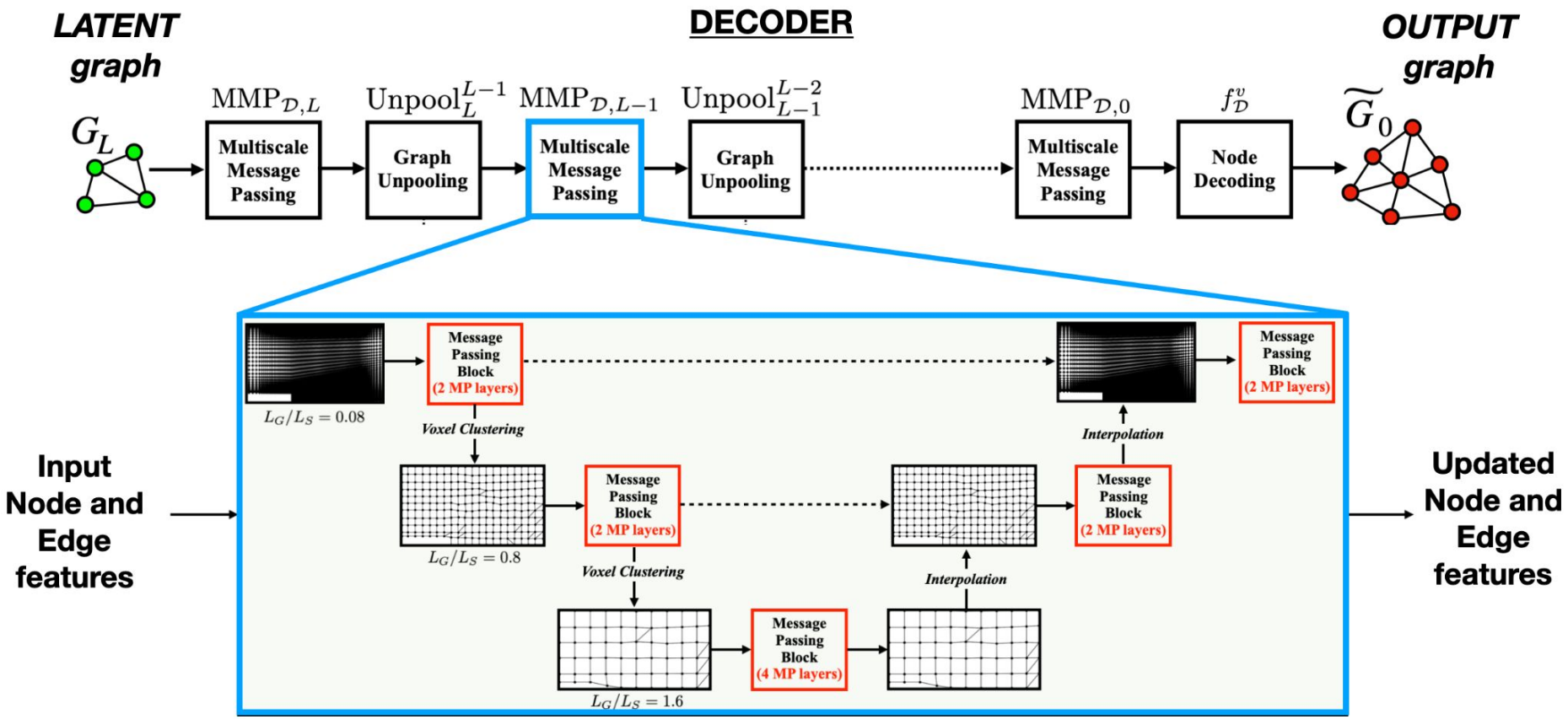




Goal: A *scalable* and *interpretable* geometric deep learning paradigm for computational fluid dynamics

Shivam Barwey.
Postdoctoral Fellow
Argonne National Laboratory.

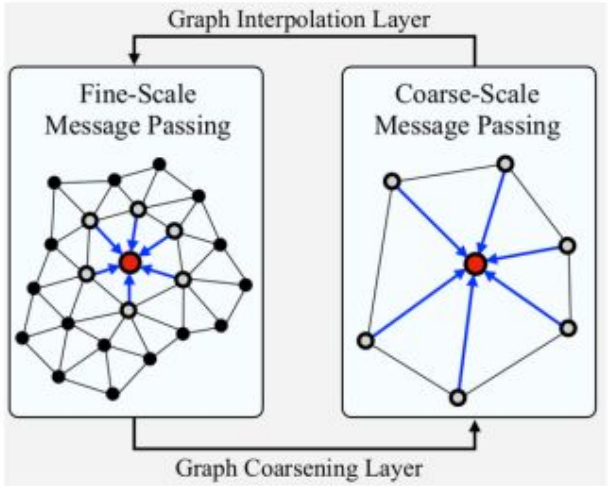
A multiscale graph neural network



Multiscale Graph Neural Network closure for unstructured grids

S. Barwey, V. Shankar, V. Vishwanathan, R. Maulik: Multiscale graph neural network autoencoders for interpretable scientific machine learning, Journal of Computational Physics, 2023

End-to-end differentiable modeling



$\boldsymbol{\tau}$

$$\nabla \bar{u} = 0$$

$$\frac{\partial \overline{\mathbf{u}}}{\partial t} + \nabla \cdot (\overline{\mathbf{u}} \otimes \overline{\mathbf{u}}) = \frac{1}{Re} \nabla^2 \overline{\mathbf{u}} - \frac{1}{\rho} \nabla \overline{p} + \overline{\mathbf{f}} + \nabla \cdot \boldsymbol{\tau}$$

$$J = J(\bar{u})$$

Observation operator

$$\frac{\partial J}{\partial \boldsymbol{\theta}}$$ ← Backpropagation $$\frac{\partial J}{\partial \boldsymbol{\tau}}$$ ← Adjoint model J

Goes to a stochastic gradient descent type optimizer (here ADAM)

FENICS + Pytorch

Algorithm1 IPCS method for N-S

Initialize:

$\texttt{GNN}_\theta$
`tentative_vel`
`pres_correct`
`vel_correct`
$\mathbf{u}, p$
$t \leftarrow 0$

while $t < T$ **do**

$\alpha_\tau \leftarrow \texttt{GNN}_\theta(\mathbf{u}, p)$
$\mathbf{u}^* \leftarrow \texttt{tentative\_vel}(\mathbf{u}, p, \alpha_\tau)$
$p^* \leftarrow \texttt{pres\_correct}(\mathbf{u}^*, p)$
$\mathbf{u} \leftarrow \texttt{vel\_correct}(\mathbf{u}^*, p^*, p)$
$p \leftarrow p^*$
$t \leftarrow t + \Delta t$

end while

Algorithm2 Custom gradient for PDE solve

function PDE_SOLVE($\mathbf{m}$; F)
  $A, b = \text{assemble}(\text{F}(\mathbf{m}))$ ▷ LHS/RHS of form F
  $\mathbf{x} = \text{solve}(A, b)$ ▷ Linear solve
  return $\mathbf{x}$
end function
function PDE_SOLVE_VJP($y_\mathbf{x}, \mathbf{m}$; F)
  $\boldsymbol{\lambda} = \text{solve}(A^T, y_\mathbf{x}^T)$ ▷ Adjoint solution
  $F_\mathbf{m} = \text{derivative}(\text{F}(\mathbf{m}; \mathbf{x}), \mathbf{m})$ ▷ Residual gradient
  $y_\mathbf{m} = \text{assemble}(-\boldsymbol{\lambda}^T F_\mathbf{m})$ ▷ Assembled gradient
  return $y_\mathbf{m}$
end function

The PDE_SOLVE_VJP function is manually constructed to tell Pytorch what the gradient of the forward-pass through fenics is. Pytorch calls this (without backpropagation) and then links it to the AD-based gradient using the chain-rule.

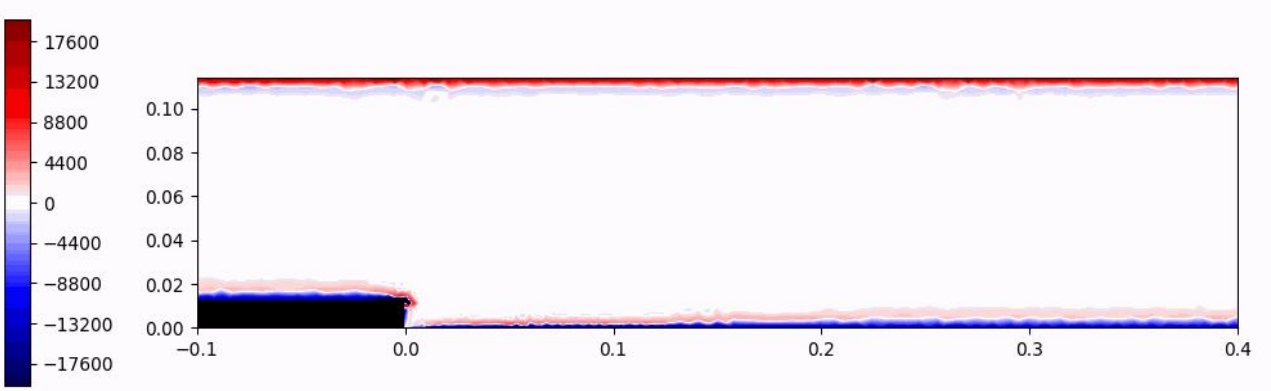
If your function is out = f(g(x)), and the loss is y; the backward-pass is g_VJP(f_VJP(dy/dout)).

Some notation abuse here:
y is the objective, *x* is the state. *m* are trainable parameters.

We use a discrete adjoint for a nonlinear system solve.

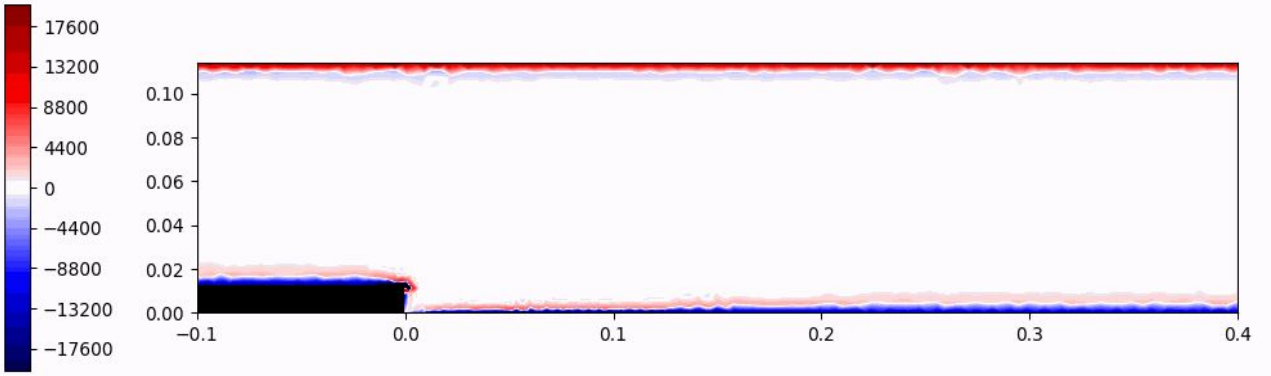
PyTorch

The backward facing step (at a different height)



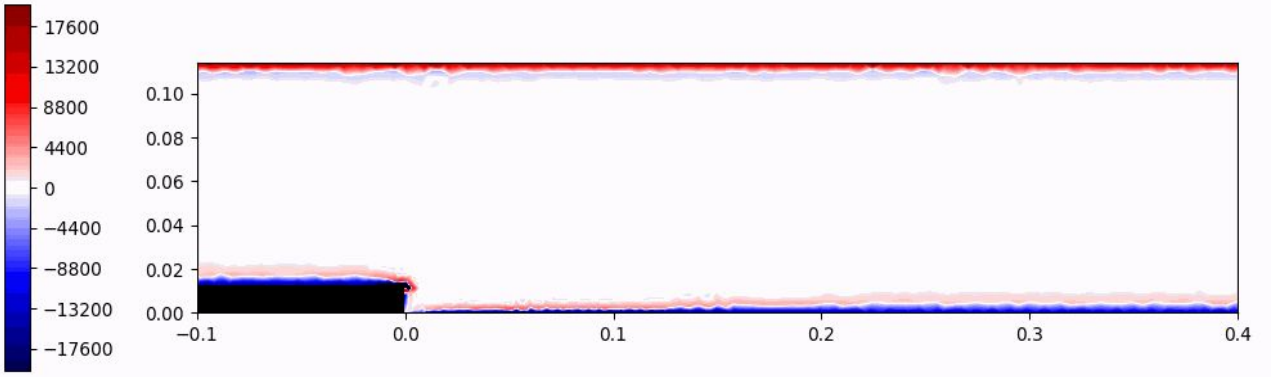
True

Truth from finer grid (1 order of magnitude higher points).



Smagorinsky (C=0.2)

$$\nu_t = (C_s\Delta)^2|\overline{\mathbf{S}}|$$
$$\boldsymbol{\tau}^{EV} = -2\nu_t\overline{\mathbf{S}},$$

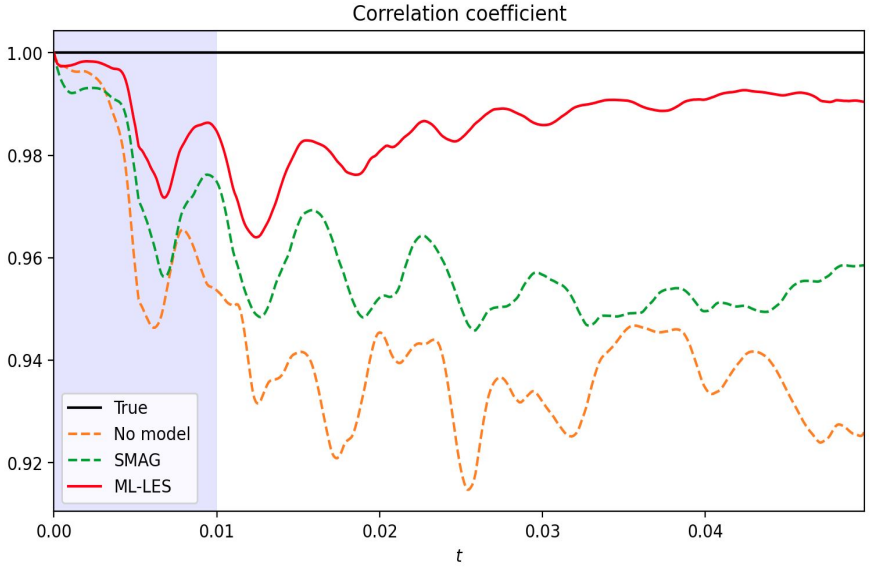


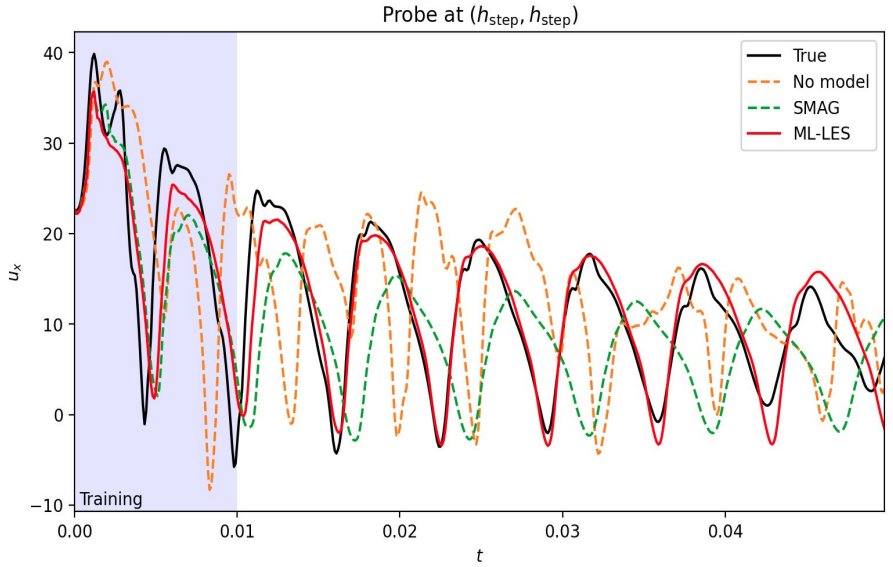
Differentiable physics ML

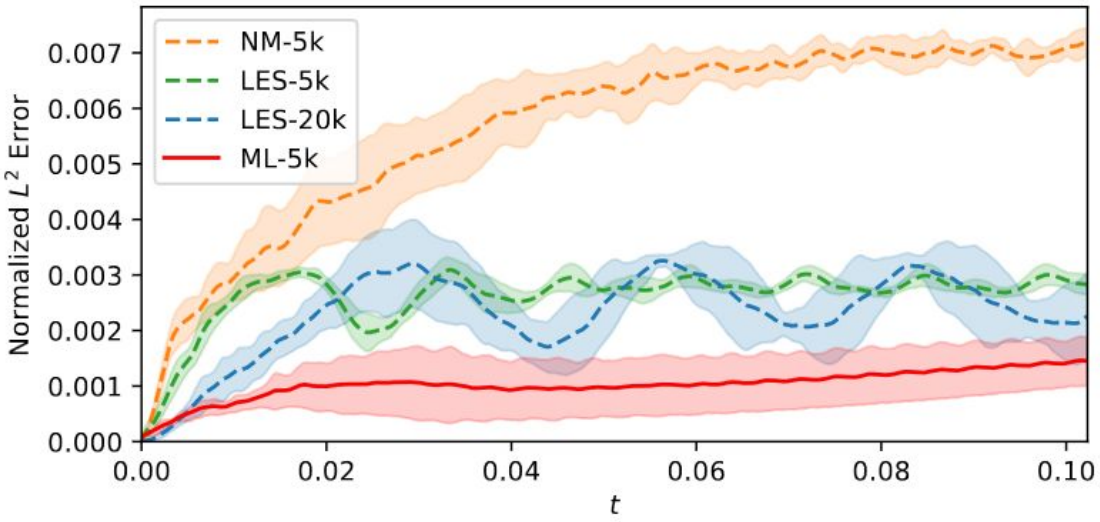
$$\hat{\boldsymbol{\tau}} = \boldsymbol{\tau}^{EV} + \boldsymbol{\tau}^{NL},$$

Varying C

Quantitative metrics



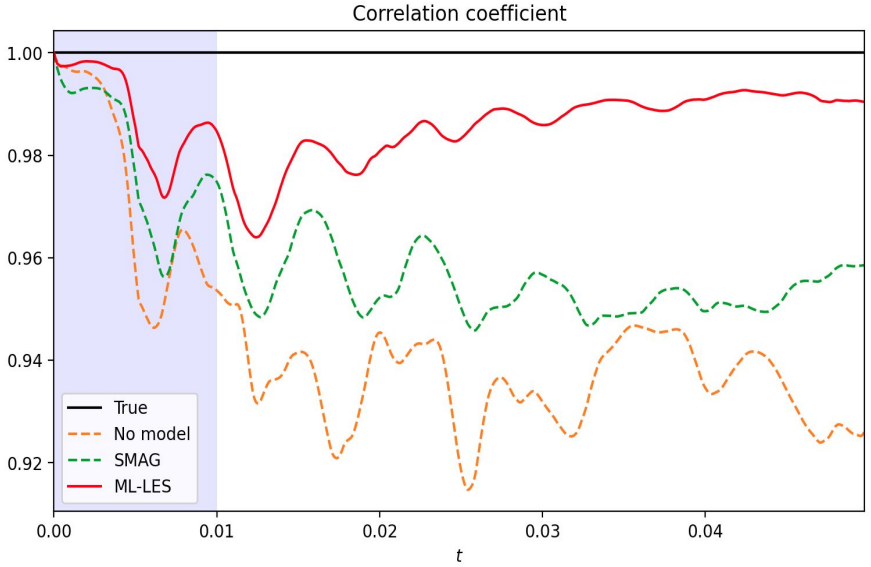




$$L = \frac{1}{T}\sum_{t=0}^{T}\int_{\Omega} ||\overline{\mathbf{u}}_t - \hat{\mathbf{u}}_t||_2^2 \, d\Omega,$$

Training time: 6 hours on 2 V100s
(Bottleneck - CPU-GPU transfer.)

Quantitative metrics



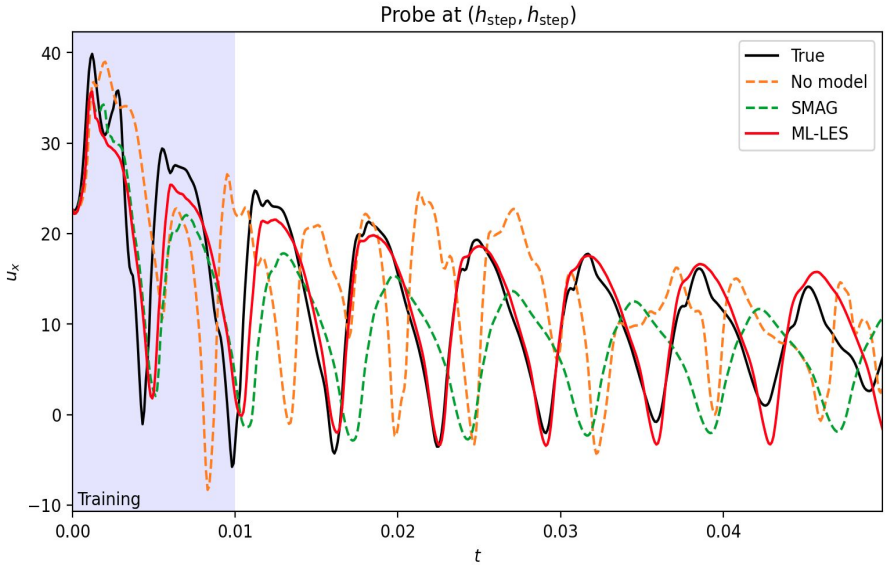


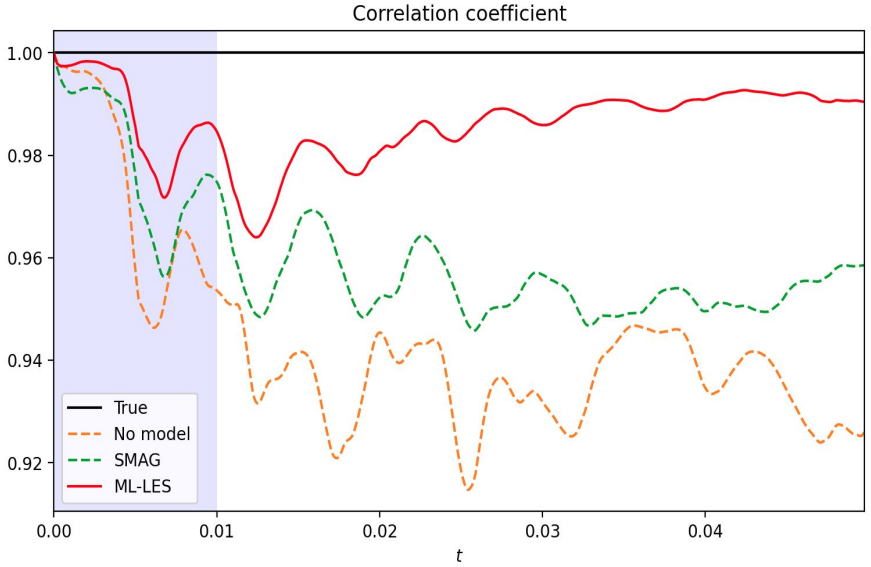
TABLE II. Core/GPU-s per 10^{-4} s of physical time in simulation

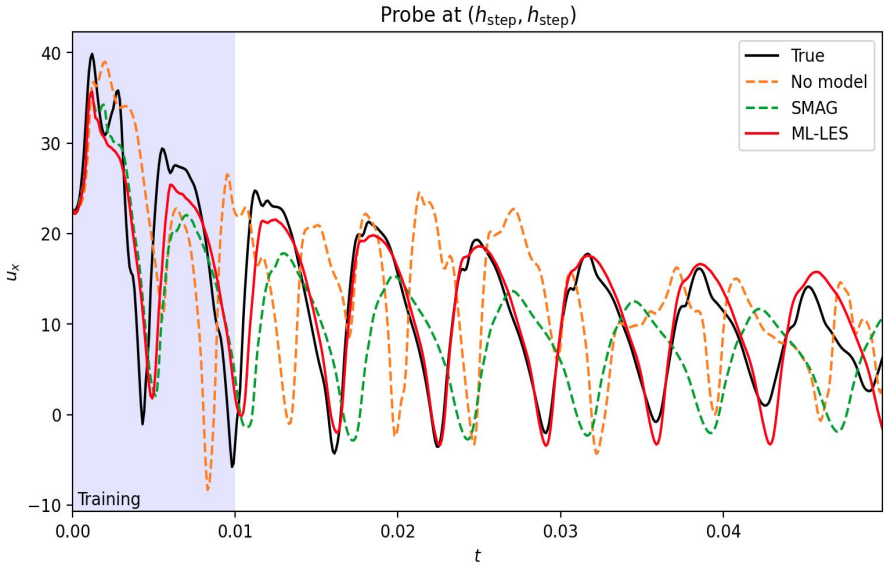
Model	Cost
LES-80k	177 s
LES-20k	7.63 s
LES-5k	0.557 s
ML-5k	0.711 s

$$L = \frac{1}{T}\sum_{t=0}^{T}\int_{\Omega} ||\overline{\mathbf{u}}_t - \hat{\mathbf{u}}_t||_2^2 \, d\Omega,$$

Shankar, Varun, Romit Maulik, and Venkatasubramanian Viswanathan. "Differentiable Turbulence II." *arXiv preprint arXiv:2307.13533* (2023).

Quantitative metrics





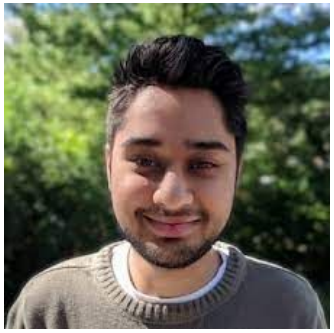
Predictions of Smagorinsky coefficient from graph multiscale NN

What we're working on?

1. Scaling up infrastructure for in-situ/adaptive training - CFD solvers and Pytorch don't play well.
2. Deployment for 3D problems.
3. Expanding applications.
4. Data and model fusion during training.
5. Long rollout times with chaotic systems.

Happy to chat some more!

Acknowledgements: DOE ASCR Data-intensive SciML, NSF GRFP



Dr. Varun Shankar, NSF Graduate Research Fellow