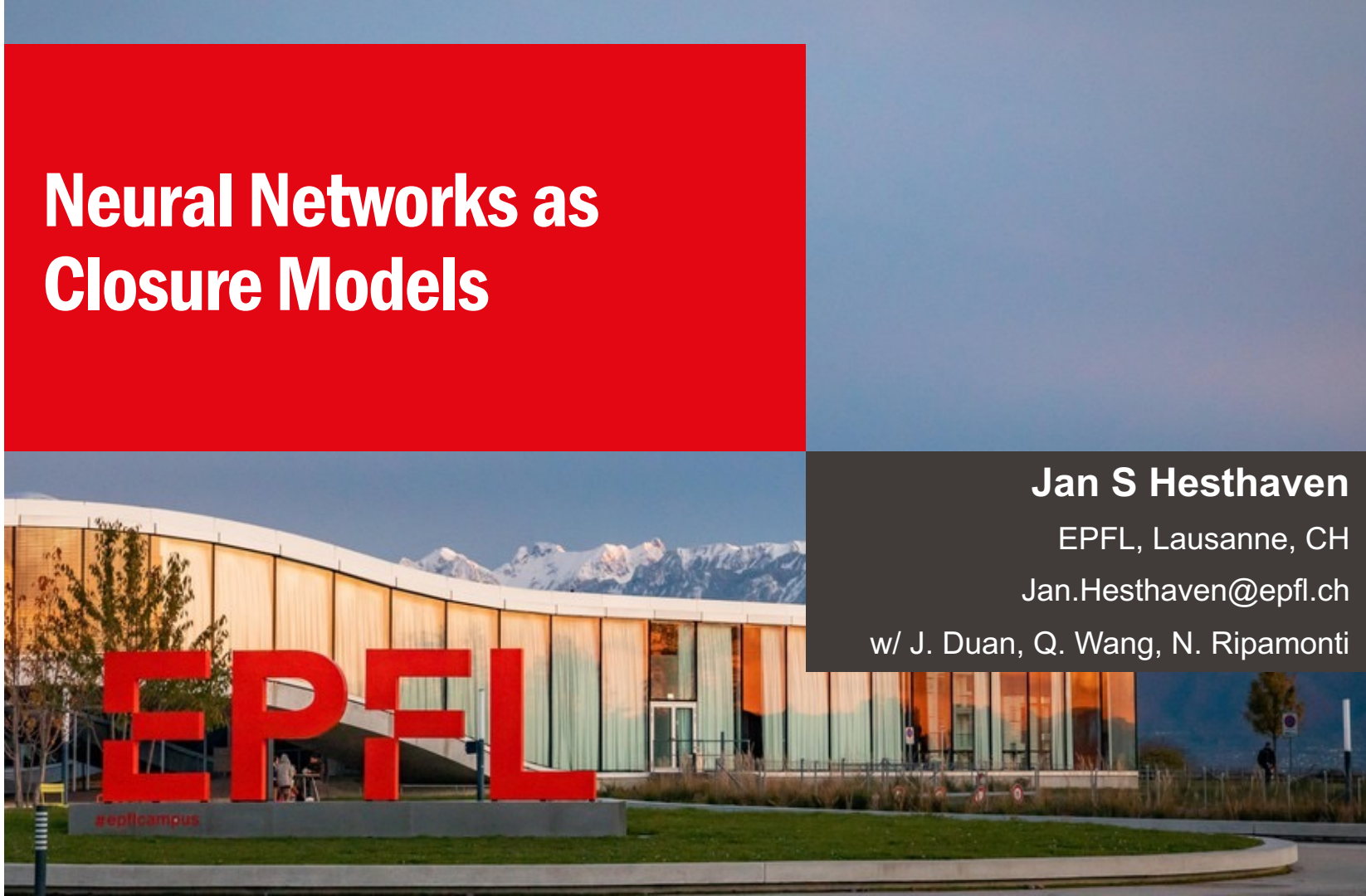
Neural Networks as Closure Models



Jan S Hesthaven

EPFL, Lausanne, CH

Jan.Hesthaven@epfl.ch

w/ J. Duan, Q. Wang, N. Ripamonti

École
polytechnique
fédérale
de Lausanne

Closures in a physical model

Closures appear in different contexts – consider the Navier-Stokes

$$\frac{\partial V_i}{\partial x_i} = 0, \quad \frac{\partial V_i}{\partial t} + V_j \frac{\partial V_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 V_i}{\partial x_j \partial x_j},$$

Assume now that

$$V_i(x,t) = U_i(x,t) + u_i(x,t) \qquad < v >= \frac{1}{T}\int_0^T v dt$$

Take temporal average to obtain

$$\frac{\partial U_i}{\partial x_i} = 0, \quad \frac{\partial U_i}{\partial t} + U_j \frac{\partial U_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial P}{\partial x_i} - \frac{\partial \langle u_i u_j \rangle}{\partial x_j} + \nu \frac{\partial^2 U_i}{\partial x_j \partial x_j},$$

A closure is needed, eg

$$\langle u_i u_j \rangle = -\tau_{ij} = \frac{2}{3}\delta_{ij}k + 2k b_{ij}, \quad k = \frac{1}{2}\langle u_k u_k \rangle,$$

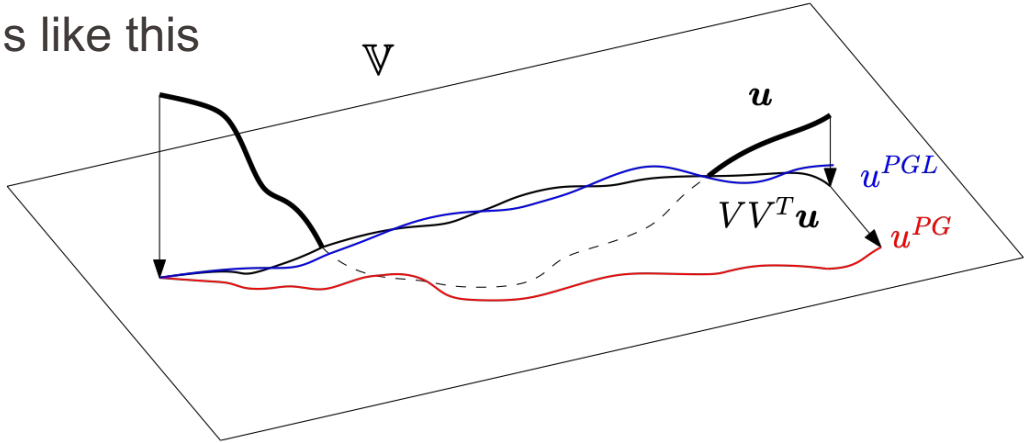
Jan S Hesthaven

Closures in a computational model

Consider the numerical model

$$\frac{d}{dt}\alpha(t;\omega) = V^T f(t, V\alpha + \bar{u}; \omega) = \tilde{f}(t, \alpha; \omega), \qquad u \approx V\alpha + \bar{u},$$

The general situation is like this



How do we predict/estimate the error $VV^T u - u^{PG}$ or

$$\frac{d}{dt}\alpha(t,\omega) = \tilde{f}(t,\alpha;\omega) + \mathcal{M}(t,\alpha;\omega)\,.$$

Jan S Hesthaven

Closures in a computational model

Let's see if we can understand the nature of the term - consider

$$\begin{cases} \frac{d}{dt}y^1 = A_{11}y^1 + A_{12}y^2, \\ \frac{d}{dt}y^2 = A_{21}y^1 + A_{22}y^2, \end{cases}$$

Here $y = (y^1, y^2)$ represent resolved and unresolved scales

By linearity we have

$$y^2(t) = e^{A_{22}t}y^2(0) + \int_0^t e^{A_{22}(t-s)}A_{21}y^1(s)\,ds,$$

Inserting into the first equation yields

$$\frac{d}{dt}y^1 = A_{11}y^1 + A_{12}\int_0^t e^{A_{22}(t-s)}A_{21}y^1(s)\,ds + A_{21}e^{A_{22}t}y^2(0).$$

Resolved scales

Memory term

Impact of unresolved initial conditions

Jan S Hesthaven

Mori-Zwanzig model

Following Chorin (’00), this can be generalized - consider

$$\begin{cases} \frac{d}{dt}y = f(t, y(t)), \quad \forall t \in [0, T] \\ y(0) = y_0, \end{cases}$$

The general MZ formulation becomes

Impact of unresolved initial conditions (=0 in most cases)

$$\frac{d}{dt}\hat{y}(t) = f(t, \hat{y}(t)) + F(t, y_0) + \int_0^t K(t-s, \hat{y}(s))ds.$$

Resolved scales — Memory term

Liouville operator

$$F(t, x) = e^{t\mathcal{Q}\mathcal{L}}\mathcal{Q}\mathcal{L}x, \quad K(t, x) = \mathcal{P}\mathcal{L}F(t, x).$$

where

$$\mathcal{Q} = I - \mathcal{P}$$

Orthogonal dynamics — Projector

$$y = (\hat{y}, \tilde{y})$$

(resolved, unresolved) scales

Jan S Hesthaven

Mori-Zwanzig model

The problem now becomes

$$\begin{cases} \dfrac{d}{dt}\alpha(t;\omega) = V^T f(t, V\alpha + \bar{u}; \omega) + \displaystyle\int_0^t K(t-s, \alpha(s;\omega); \omega)\, ds, \\ \alpha(0;\omega) = 0. \end{cases}$$

The missing element is therefore the estimation of the memory term

$$\mathcal{M}(t, \alpha; \omega) = \int_0^t K(t-s, \alpha(s;\omega); \omega)\, ds$$

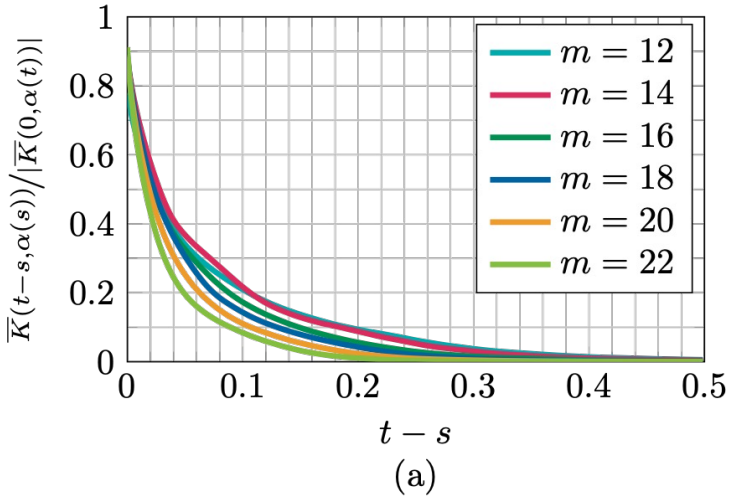
Understanding this would yield insight into the global error of the model - however it is complex to accurately estimate this term

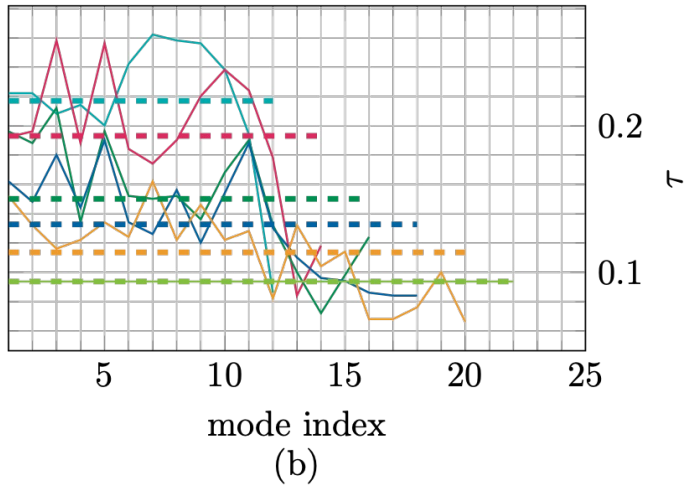
Jan S Hesthaven

Mori-Zwansig Model

Typically, exponential decay of the kernel allows approximations, eg

$$\int_0^t K(t-s,\alpha(s;\omega);\omega)\,ds \overset{(a)}{\approx} \int_{t-\tau}^t K(t-s,\alpha(s;\omega);\omega)\,ds \overset{(b)}{\approx} \frac{1}{2}\tau K(0,\alpha(t;\omega);\omega)$$





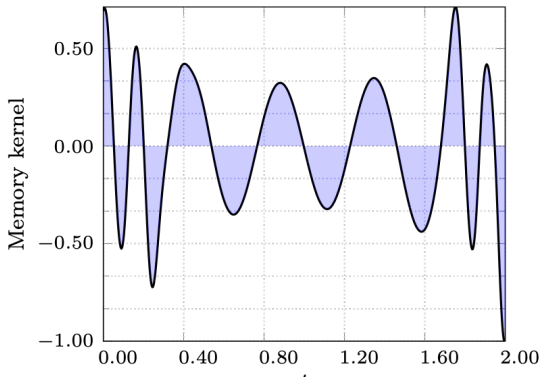
Results for viscous Burgers equation

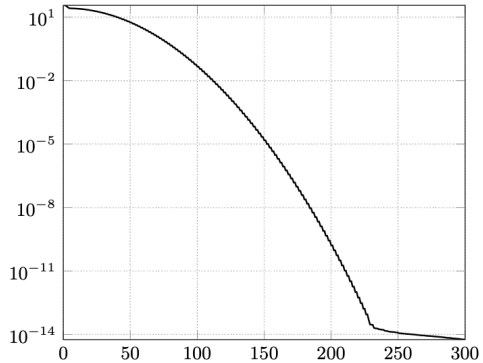
Jan S Hesthaven

Mori-Zwanzig model

Unfortunately not universally valid

$$\begin{cases} \dfrac{\partial u}{\partial t} = -c\nabla u, \quad x \in [0, 16\pi] \\ u(x,0) = e^{-\dfrac{(x-8\pi)^2}{0.8^2}}, \end{cases}$$





For strongly advection dominated problems, closure problem is harder

Jan S Hesthaven

Learning closures

The question we would like to address here is:
Considering the model

$$\frac{\partial u}{\partial t} = f(u, x, t) + M(x, u, t)$$

Can we learn what M should look like rather than trying to model it ? – a notoriously hard problem

- How do we learn ?
- What kind of data can we use ?
- What kind of networks can we use to learn ?

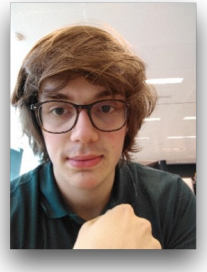
Jan S Hesthaven

Overview of what remains

- Reduced order modeling
- Closures for ROM of time-dependent models
- Corrections of dynamic models of a drone

Dr. Junming Duan
EPFL, CH

Prof. Qian Wang
CSRC, PRC



Dr. Nicolo Ripamonti
EPFL, CH

Jan S Hesthaven

Overview of what remains

- Reduced order modeling
- Closures for ROM of time-dependent models
- Corrections of dynamic models of a drone

Reduced order modeling

Consider a parametrized PDE

$$A(x,\mu)u(x,\mu) = f(x,\mu) \quad x \in \Omega, \mu \in \mathcal{P}$$

and wish to solve it accurately for many values of the parameter μ

We can use our favorite discretization over the spatial domain

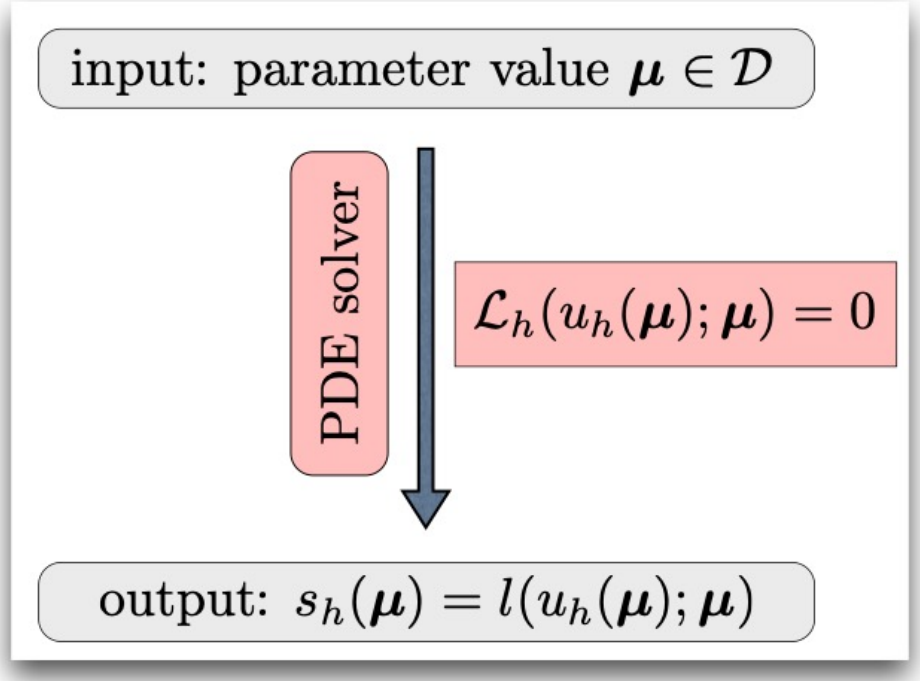
$$\mathbf{A}_h(\mu)\mathbf{u}_h(\mu) = \mathbf{f}_h(\mu) \qquad \dim(\mathbf{u}_h) = N_h \gg 1$$

For many parameter values, this is expensive and slow !

Jan S Hesthaven

Reduced order modeling

An accurate way to evaluate the solution at new parameter values at reduced complexity.



Jan S Hesthaven

Reduced order modeling

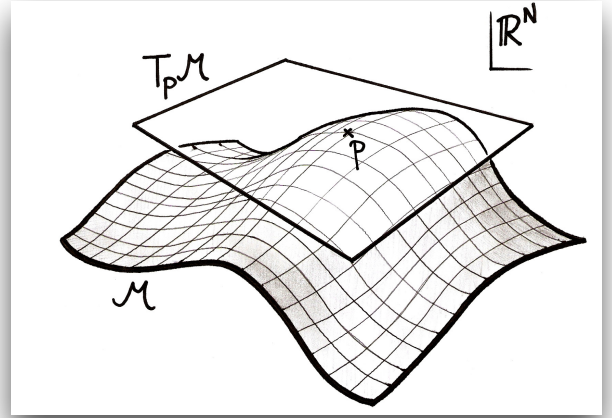
Assume we know

$$\mathbf{u}_h(\mu) \approx \mathbf{V}\mathbf{q}(\mu) \quad \mathbf{V}^T\mathbf{V} = \mathbf{I}$$
$$\dim(\mathbf{V}) = N_h \times N_r \quad N_r \ll N_h$$

Then recover a solution for a new parameter at little cost

$$\underbrace{[\mathbf{V}^T\mathbf{A}_h(\mu)\mathbf{V}]}_{N_r \times N_r}\underbrace{\mathbf{q}(\mu)}_{N_r} = \underbrace{\mathbf{V}^T\mathbf{f}_h(\mu)}_{N_r}$$

Assumption: The solution varies smoothly on a low-dimensional manifold under parameter variation.



This is measured by Kolmogorov N-width

$$d_k(\mathcal{M}) = \inf_A \sup_{u\in\mathcal{M}} \inf_{u_{rb}\in A} \|u - u_{rb}\|$$

For elliptic problems

$$d_k(\mathcal{M}) \sim \exp(-\alpha k)$$

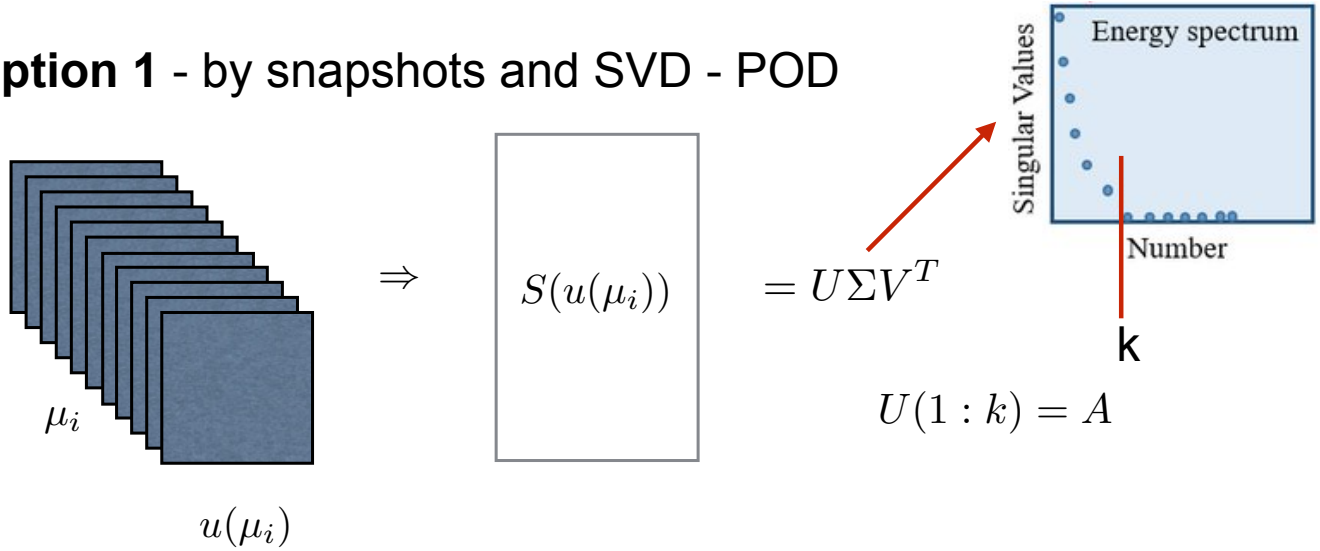
For pure transport

$$d_k(\mathcal{M}) \sim k^{-1/2}$$

Jan S Hesthaven

Reduced basis selection

Option 1 - by snapshots and SVD - POD



Option 2 - by a greedy approach with an error estimator

$$\mu_{k+1} = \arg\max_{\mathcal{D}} |e(\mu, u(\mu_1), ..., u(\mu_k))| \qquad A = [u(\mu_1), ..., u(\mu_{k+1})]$$

Note: subspace created solely based on accuracy

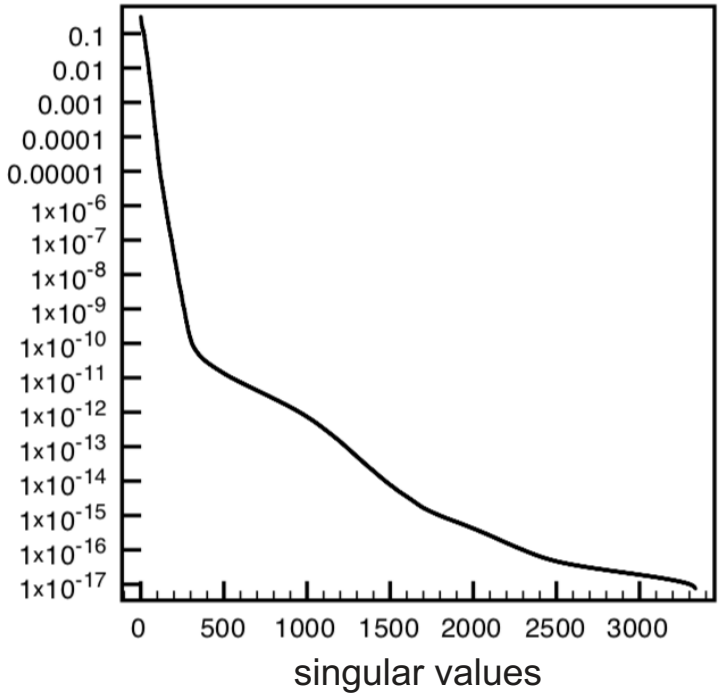
Jan S Hesthaven

Reduced basis construction

Proper Orthogonal Decomposition

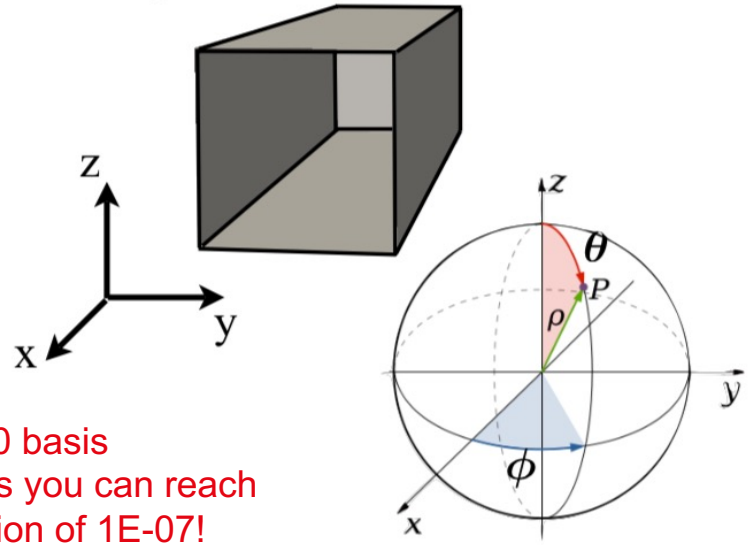
Compute basis through SVD

Two dimensional parametrization with polar angle and frequency



$(k, \theta) \in [1, 25] \times [0, \pi]$, ϕ is fixed

Geometry:



With 200 basis functions you can reach a precision of 1E-07!

Jan S Hesthaven

If you want more

Very successful for many linear problems

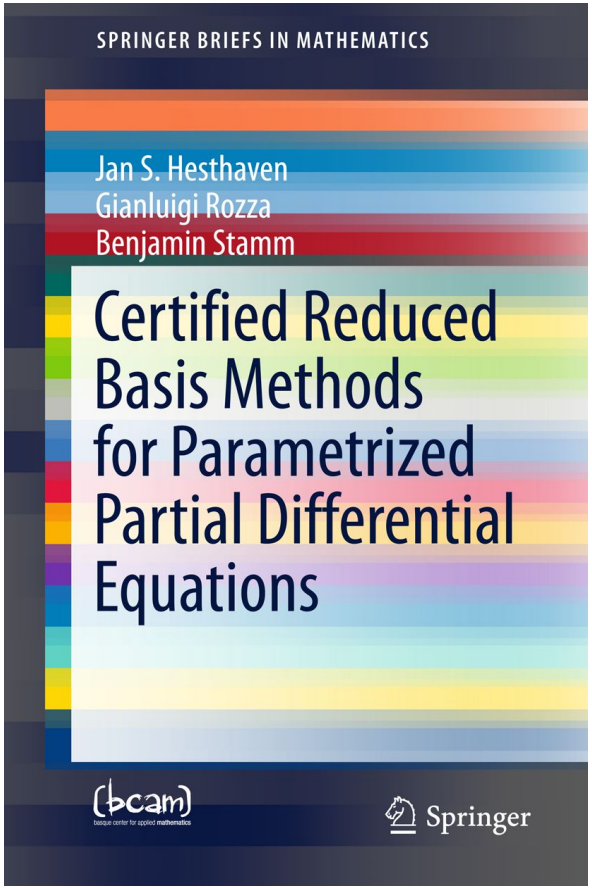
- Acoustics, electromagnetics, elasticity etc

and some nonlinear problems

It can be found on

- *Infoscience.epfl.ch*

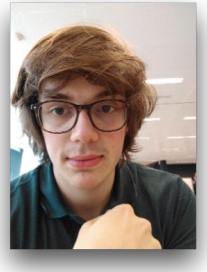
It is free.



Overview of what remains

- Reduced order modeling
- Closures for ROM of time-dependent models
- Corrections of dynamic models of a drone

Prof. Qian Wang
CSRC, PRC



Dr. Nicolo Ripamonti
EPFL, CH

Jan S Hesthaven

Closure problem for ROM

We are left with the general problem

$$\frac{d}{dt}\alpha(t,\omega) = \tilde{f}(t,\alpha;\omega) + \mathcal{M}\left(t,\alpha;\omega\right).$$

Estimating the memory term directly is difficult

Let us consider a different approach - we seek

$$\begin{aligned}\mathcal{M}\left(t_n,\alpha;\omega\right) &= \int_0^{t_n} K\left(t_n - s, \alpha(s;\omega);\omega\right) ds \\ &\approx \int_{t_n-\tau}^{t_n} K\left(t_n - s, \alpha(s;\omega);\omega\right) ds \\ &\approx \mathcal{M}^{num}\left(\alpha_{n-n_{ts}+1},\cdots,\alpha_{n-1},\alpha_n;\omega\right),\end{aligned}$$

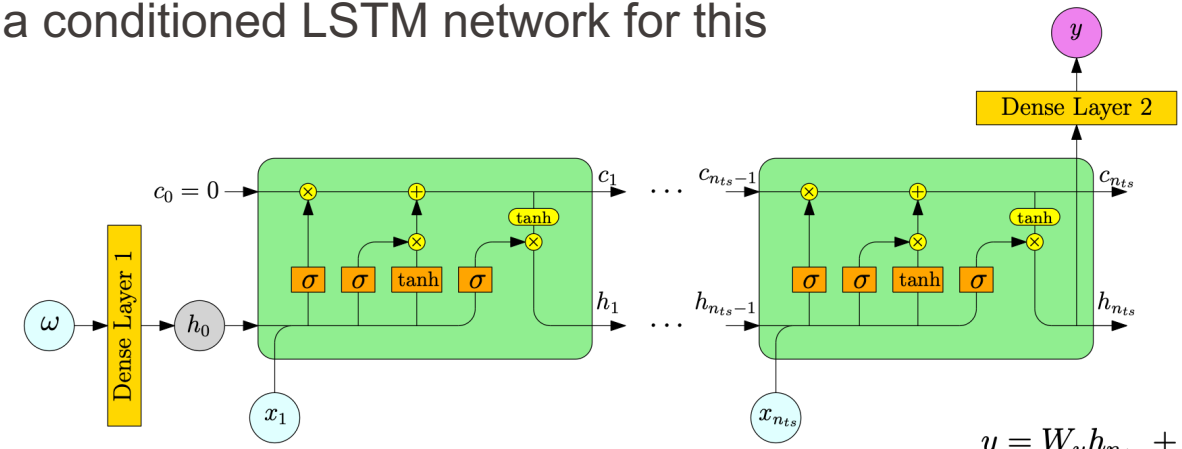
We need the map

$$\left(\alpha_{n-n_{ts}+1},\cdots,\alpha_{n-1},\alpha_n;\omega\right) \mapsto \mathcal{M}\left(t_n,\alpha;\omega\right)$$

Jan S Hesthaven

An LSTM network

We use a conditioned LSTM network for this



$$h_0 = W_{h_0}\omega.$$

$$y = W_y h_{n_{ts}} + b_y.$$

Through the layers

$$\begin{aligned}
f_t &= \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right)\\
i_t &= \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right)\\
o_t &= \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right)\\
\tilde{c}_t &= \sigma\left(W_c x_t + U_c h_{t-1} + b_c\right)\\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t\\
h_t &= o_t \circ \tanh\left(c_t\right)
\end{aligned}$$

$$\sigma(z) = \begin{cases} 0, & z < -2.5,\\ 0.2z + 0.5, & -2.5 \le z \le 2.5,\\ 1, & z > 2.5. \end{cases}$$

Jan S Hesthaven

An LSTM network

Unknowns to learn

$$W_{h_0} \in \mathbb{R}^{n_{hu} \times d}, W = \begin{pmatrix} W_f \\ W_i \\ W_o \\ W_c \end{pmatrix} \in \mathbb{R}^{4n_{hu} \times m}, U = \begin{pmatrix} U_f \\ U_i \\ U_o \\ U_c \end{pmatrix} \in \mathbb{R}^{4n_{hu} \times n_{hu}}, W_y \in \mathbb{R}^{m \times n_{hu}},$$

$$b = \begin{pmatrix} b_f \\ b_i \\ b_o \\ b_c \end{pmatrix} \in \mathbb{R}^{4n_{hu}}, b_y \in \mathbb{R}^m,$$

For testing we also estimate the memory length

$$\mathcal{M}_{\text{col}} = \begin{bmatrix} \mathcal{M}(0, \alpha; \omega) \\ \mathcal{M}(\Delta t, \alpha; \omega) \\ \cdots \\ \mathcal{M}(T, \alpha; \omega) \end{bmatrix}, \; K_{\text{col}} = \begin{bmatrix} K(0, \alpha(0; \omega); \omega) \\ K(0, \alpha(\Delta t; \omega); \omega) \\ \cdots \\ K(0, \alpha(T; \omega); \omega) \end{bmatrix}$$

$$\bar{\tau} \approx \frac{2K_{\text{col}}^T \mathcal{M}_{\text{col}}}{K_{\text{col}}^T K_{\text{col}}}.$$

Jan S Hesthaven

A linear 3D Stokes problem

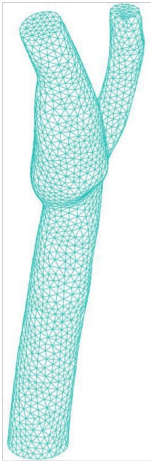
We consider the 3D linear problem

$$\begin{cases} \dfrac{\partial}{\partial t}\mathbf{u} = \nu\Delta\mathbf{u} - \nabla p, \quad t \in [0,4] \\ \nabla \cdot \mathbf{u} = 0, \\ \mathbf{u}(\partial\Omega_{\text{inlet}}, t) = f(t), \\ \mathbf{u}(\partial\Omega_{\text{wall}}, t) = 0, \\ \mathbf{u}\,(\Omega, 0) = 0, \end{cases}$$

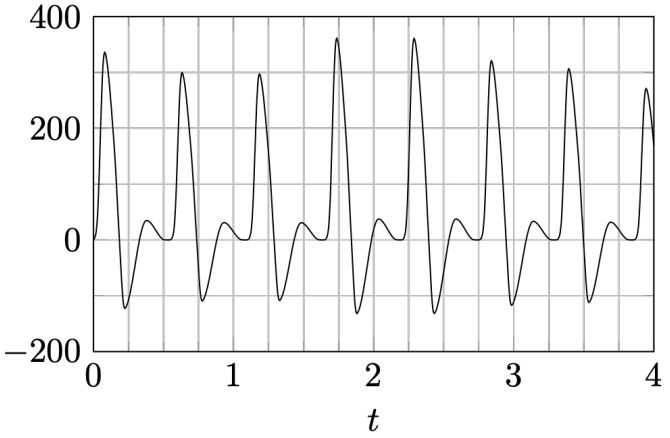
Solved using

$$\beta\frac{\partial}{\partial t}p + \nabla \cdot \mathbf{u} = 0,$$

N = 20’914
K = 1’000
nu = [2,6]

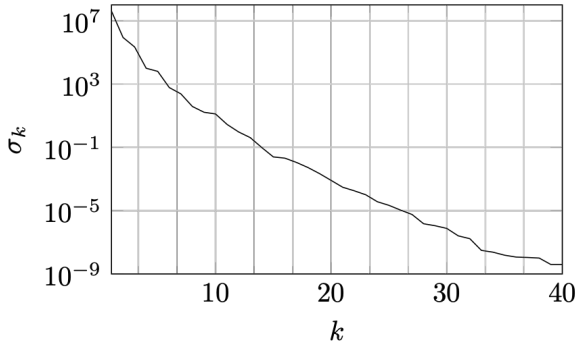


Inflow forcing



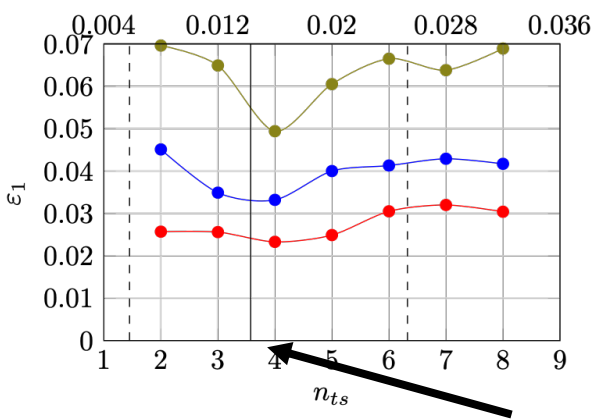
Jan S Hesthaven

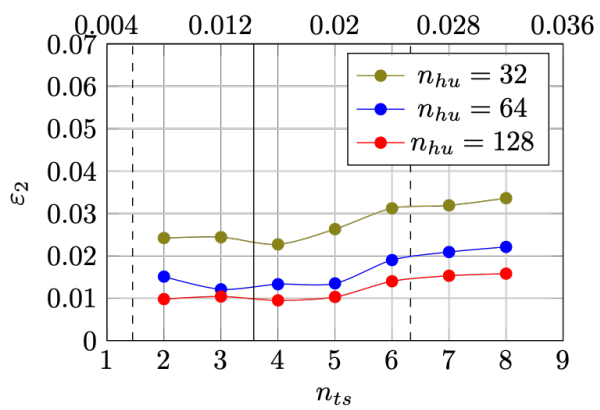
A linear 3D Stokes problem



We use 16 modes

Training using 50 HF solutions, 25 for test, 25 for validation





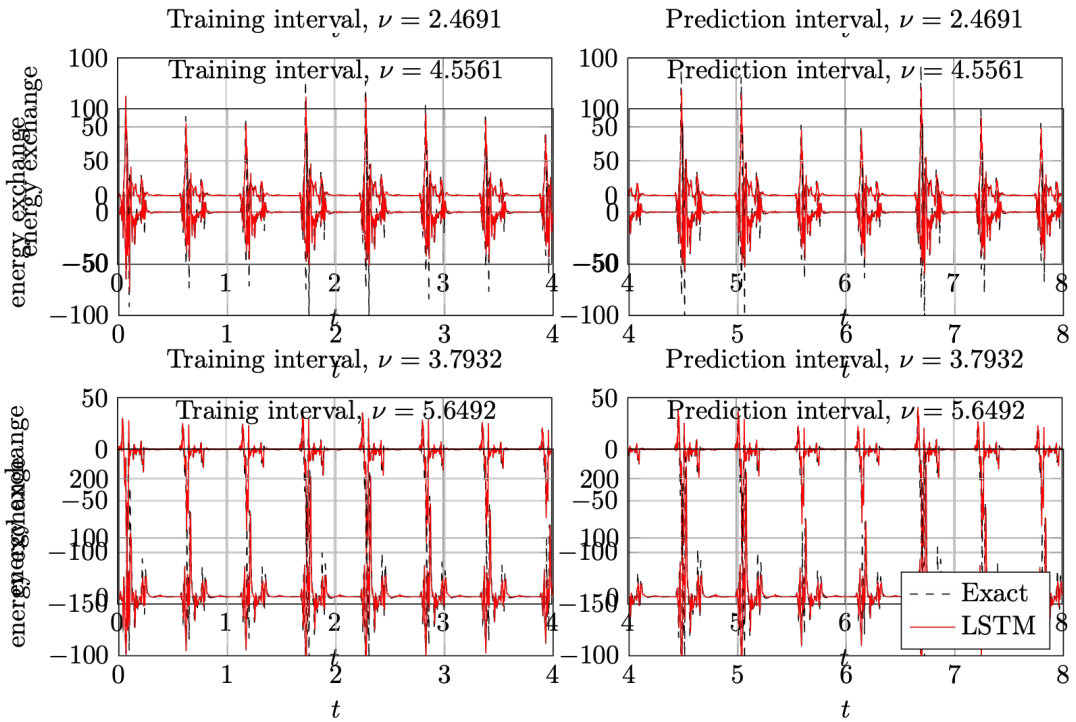
We use 4 layers, 128 hidden units

Jan S Hesthaven

A linear 3D Stokes problem

We consider the energy

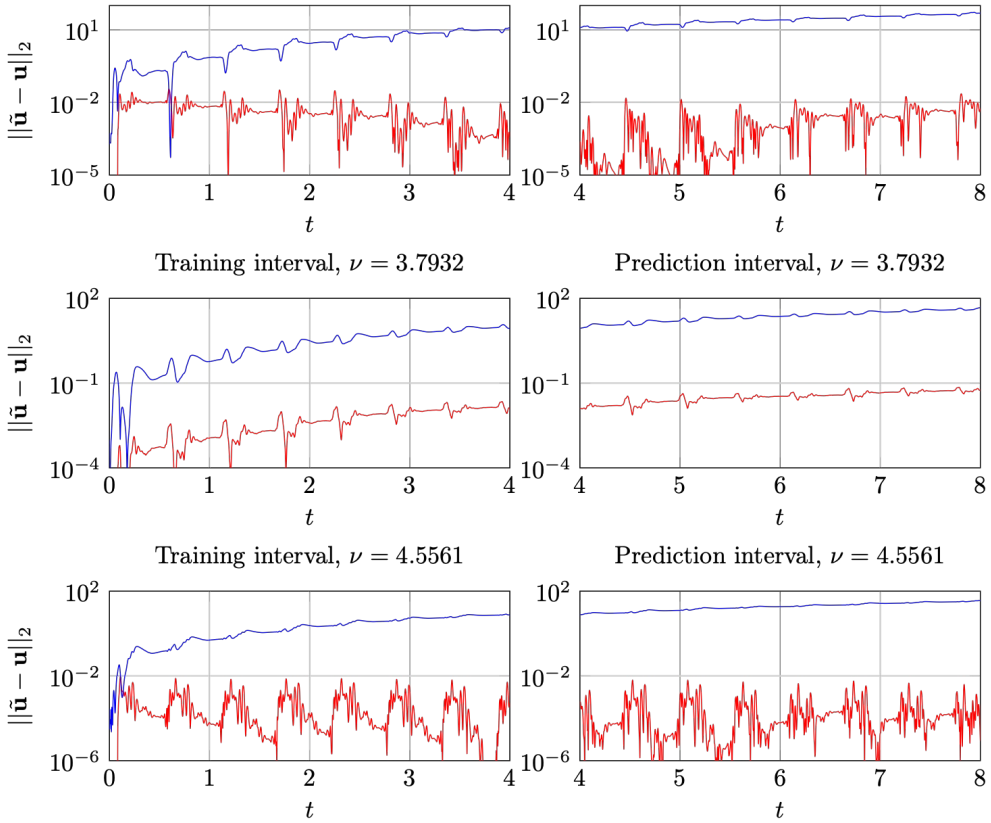
$$\frac{d}{dt}\alpha^T\alpha(t,\omega) = 2\alpha^T\tilde{f}(t,\alpha;\omega) + 2\alpha^T\mathcal{M}(t,\alpha;\omega),$$



Jan S Hesthaven

A linear 3D Stokes problem

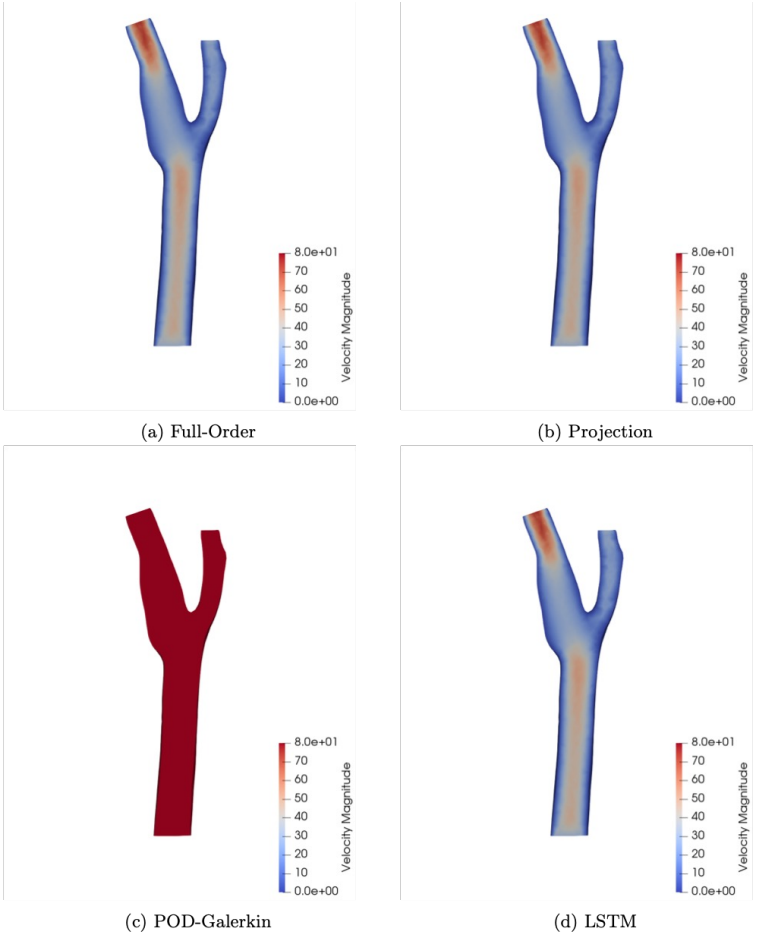
Lets look at the error



Jan S Hesthaven

A linear 3D Stokes problem

Accuracy and stabilization



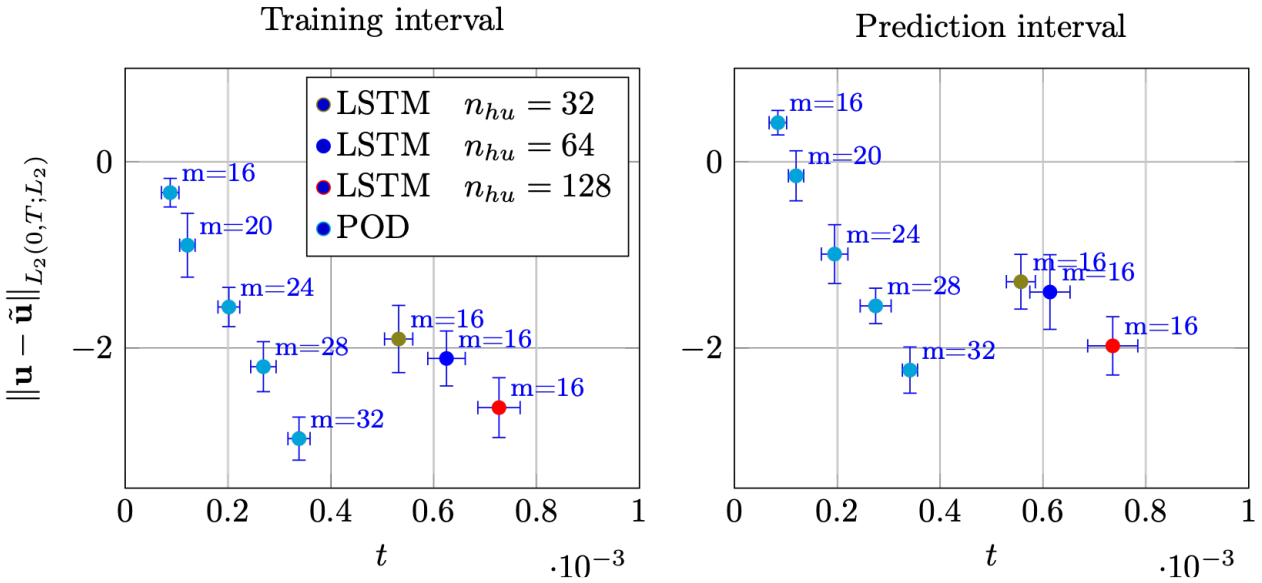
(a) Full-Order (b) Projection

(c) POD-Galerkin (d) LSTM

Jan S Hesthaven

A linear 3D Stokes problem

Cost ?



While it certainly improves accuracy, the cost of the LSTM is too high to be beneficial

Jan S Hesthaven

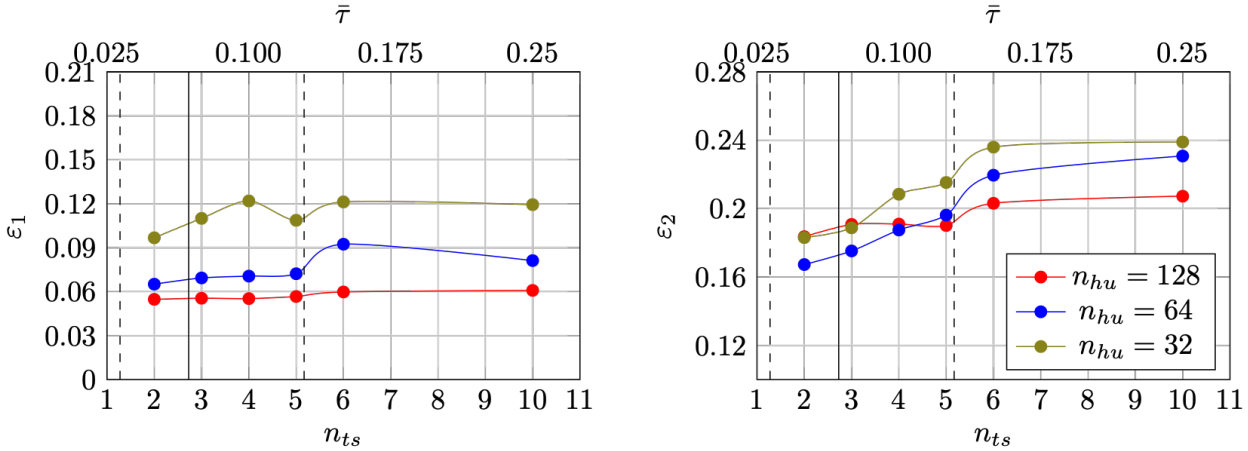
The KS equation

We consider the 1D Kuramoto-Shivashiski equation

$$\begin{cases} \frac{\partial}{\partial t}u = -\frac{1}{2}\nabla \cdot u^2 - \Delta u - \nu \Delta^2 u, \\ u(x+L,t) = u(x,t), \\ u(x,0) = g(x), \end{cases}$$

nu = [0.3, 1.5]

Solved with a Fourier method, N=1’024



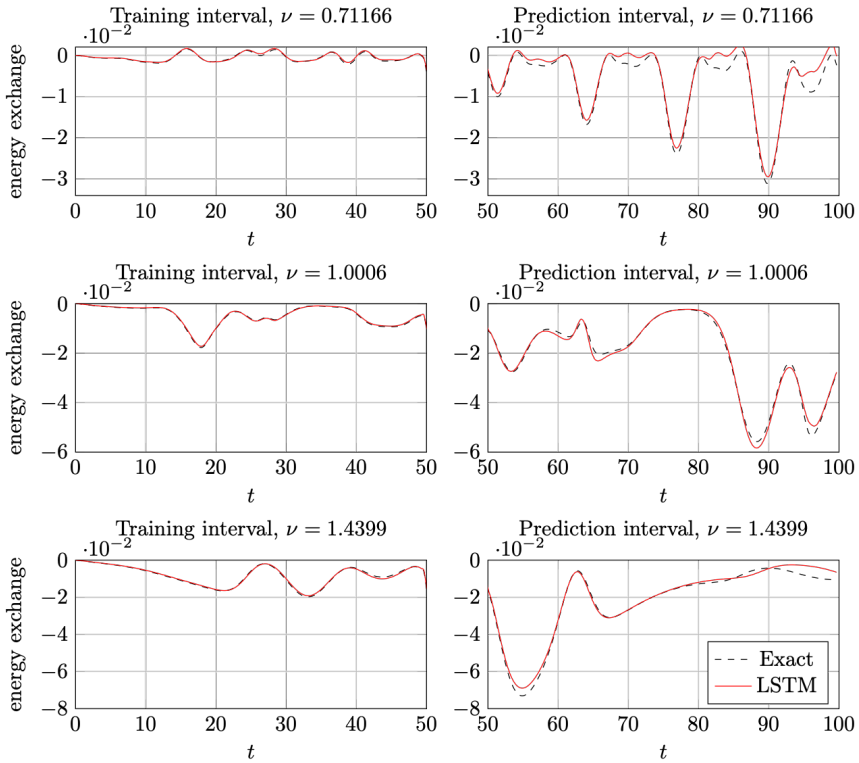
We use 4 layers, 128 hidden units

Jan S Hesthaven

The KS equation

We consider the energy

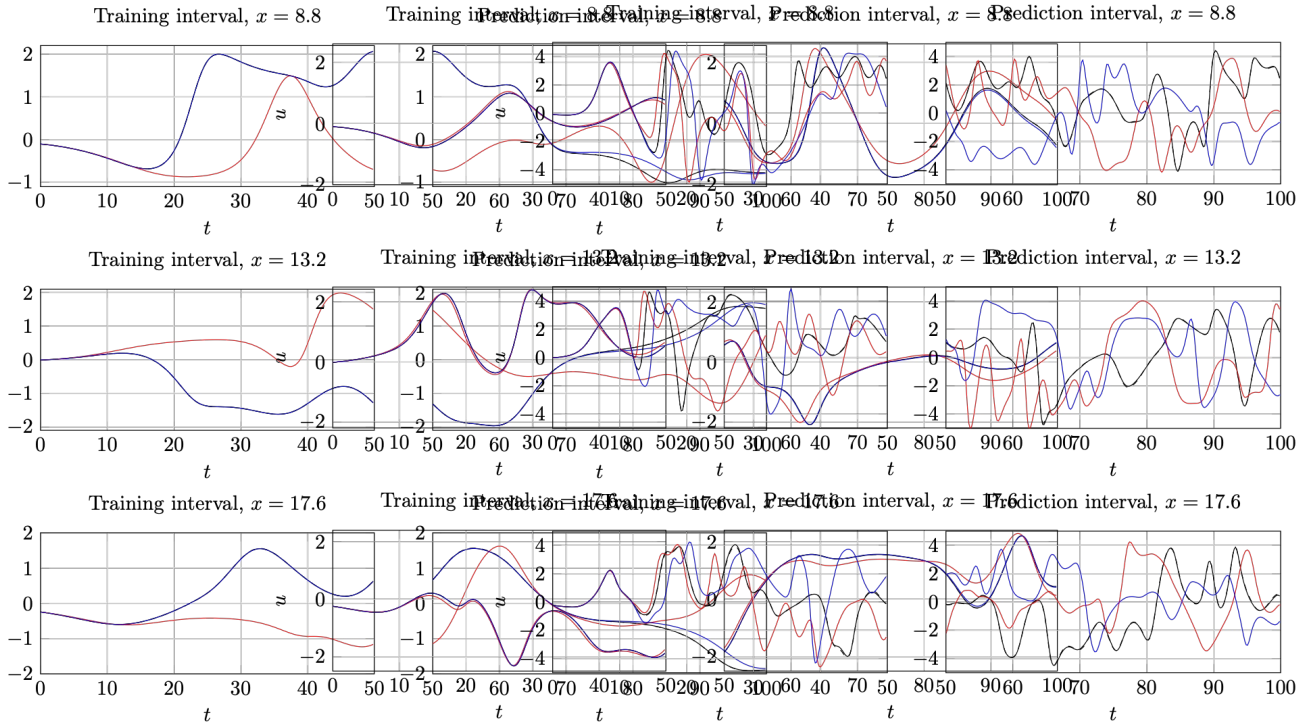
$$\frac{d}{dt}\alpha^T\alpha(t,\omega) = 2\alpha^T\tilde{f}(t,\alpha;\omega) + 2\alpha^T\mathcal{M}(t,\alpha;\omega)\,,$$



Jan S Hesthaven

The KS equation

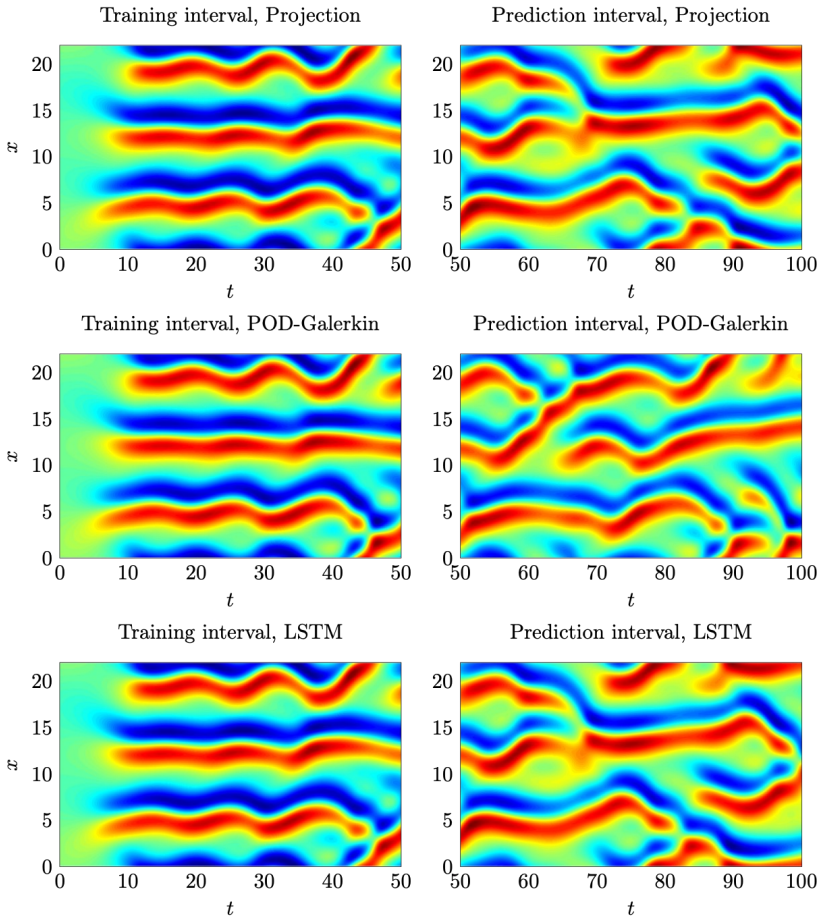
Accuracy and prediction



nu = 1.4399 nu = 1.0006 nu = 0.34756

Jan S Hesthaven

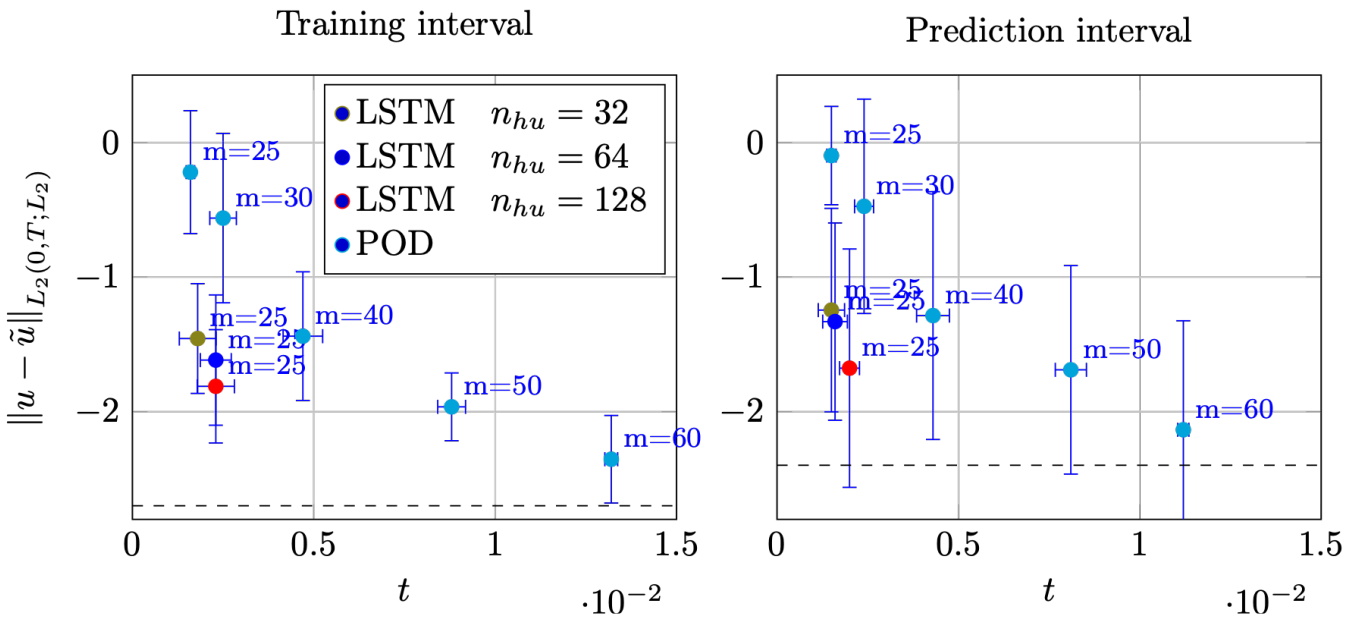
The KS equation



Jan S Hesthaven

The KS equation

Cost ?



Now the benefits are clear

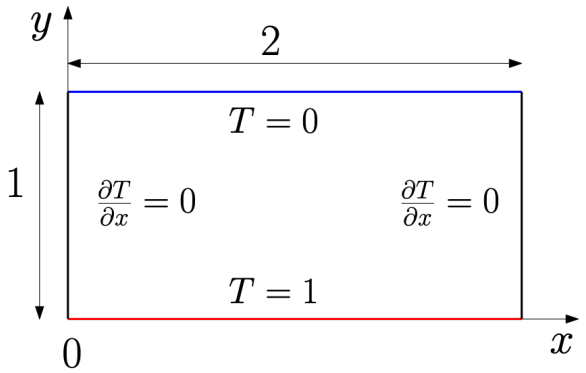
Jan S Hesthaven

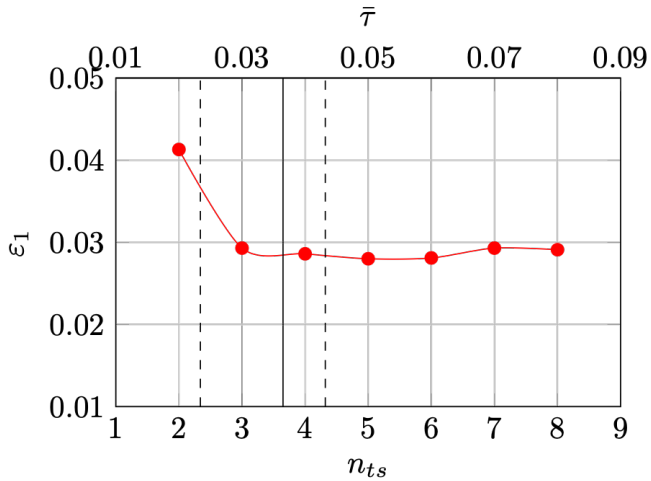
Rayleigh-Bernard convection

We consider the 2D Rayleigh-Bernard

$$\begin{cases} \nabla \cdot \mathbf{u} = 0, \\ \frac{\partial}{\partial t}\mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \sqrt{\frac{Pr}{Ra}}\Delta \mathbf{u} + T\mathbf{e}_y, \\ \frac{\partial}{\partial t}T + \mathbf{u} \cdot \nabla T = \frac{1}{\sqrt{PrRa}}\Delta T, \end{cases}$$

Two parameters - Ra and Pr

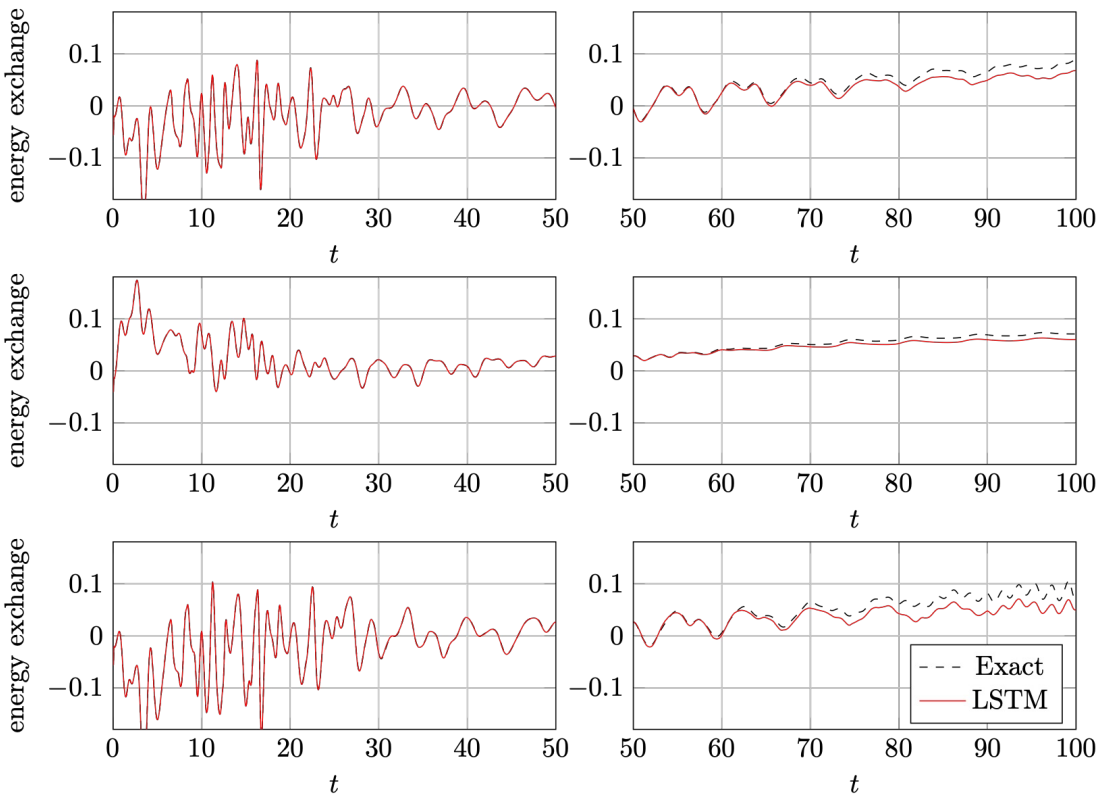




We use 3 layers, 128 hidden units

Jan S Hesthaven

Rayleigh-Bernard convection



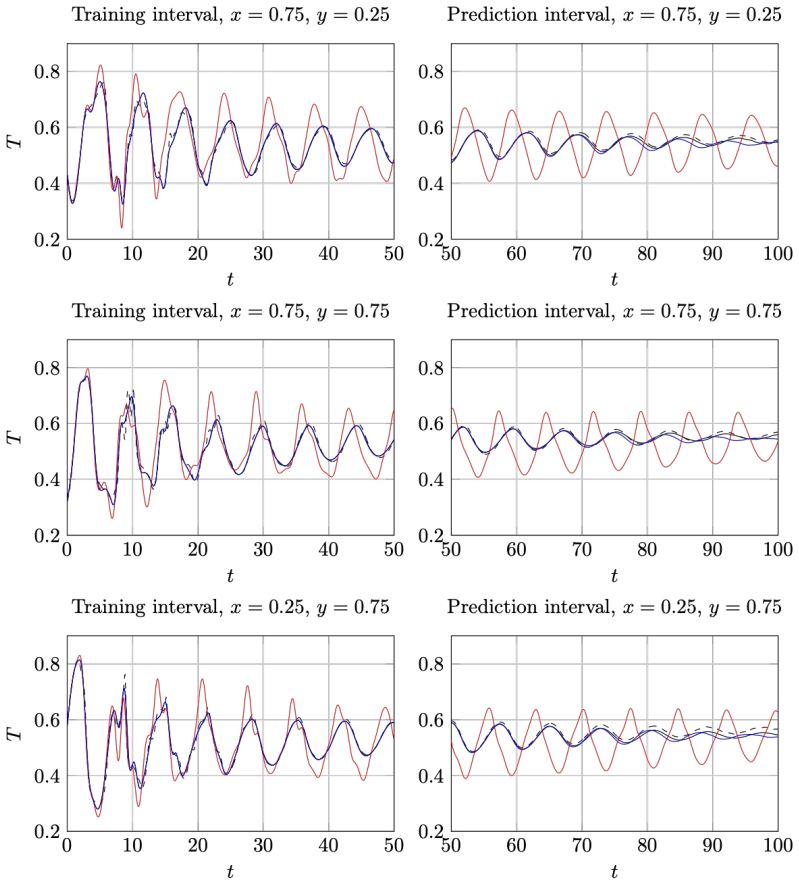
(12317729.8502, 0.86391)

(5793019.4994, 0.92232).

(14024512.0002, 0.86976)

Jan S Hesthaven

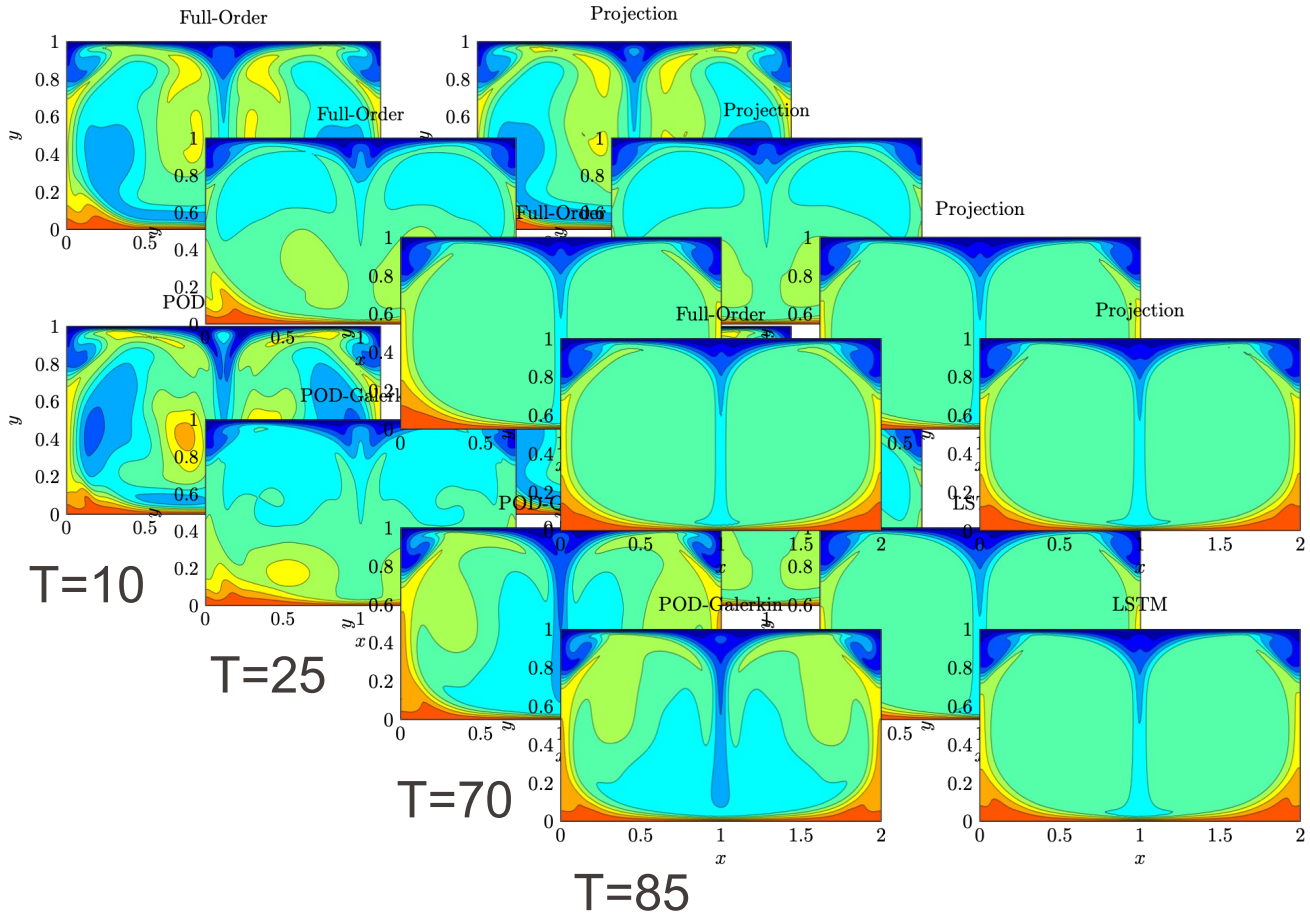
Rayleigh-Bernard convection



$(14024512.0002, 0.86976)$

Jan S Hesthaven

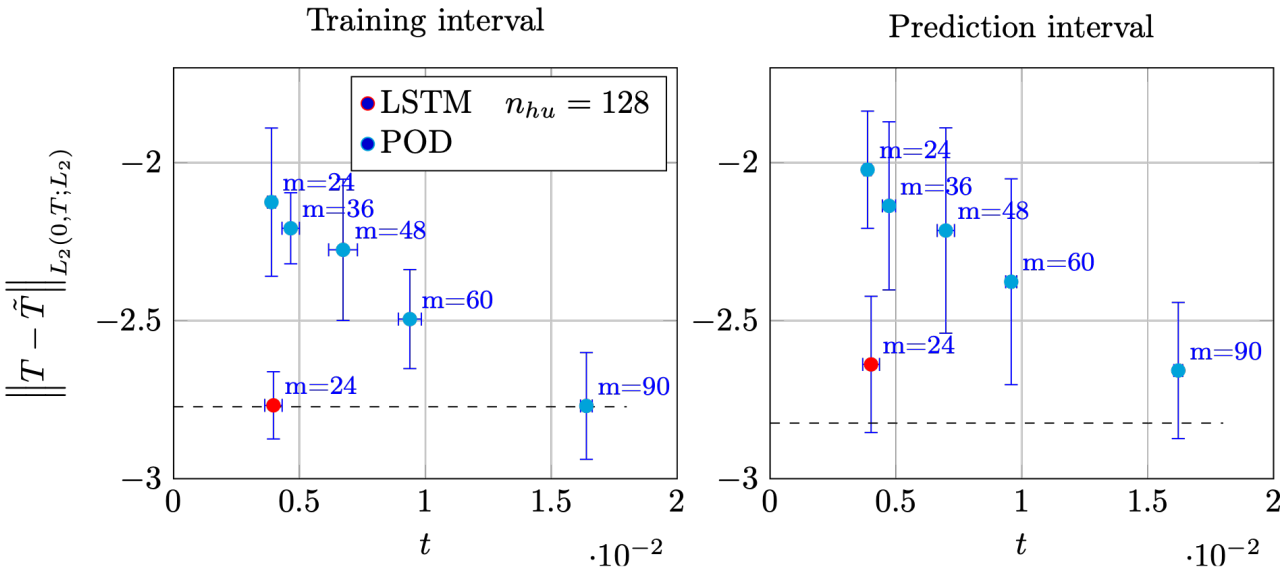
Rayleigh-Bernard convection



Jan S Hesthaven

Rayleigh-Bernard convection

Cost ?



Substantial benefits in both accuracy and cost

Jan S Hesthaven

Overview of what remains

- Reduced order modeling
- Closures for ROM of time-dependent models
- Corrections of dynamic models of a drone

Dr. Junming Duan
EPFL, CH

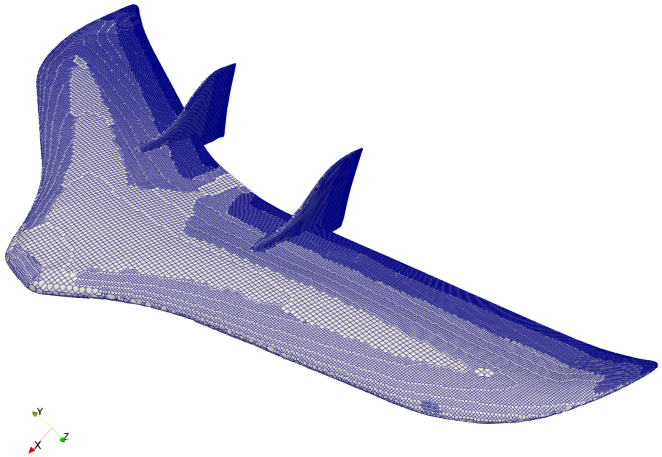
Prof. Qian Wang
CSRC, PRC

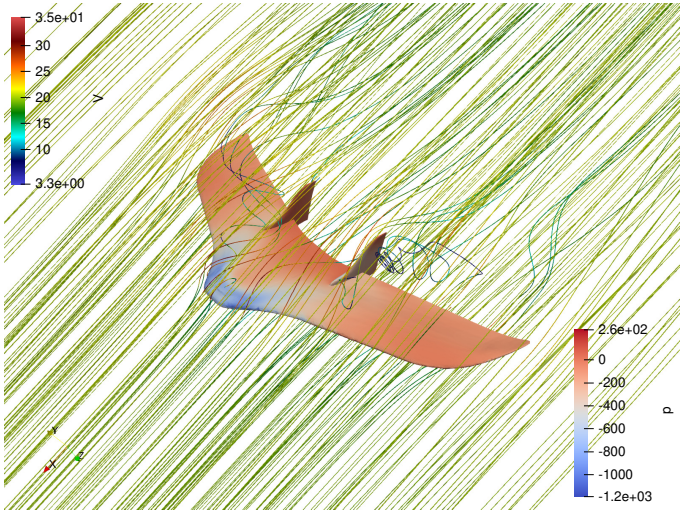
Jan S Hesthaven

Using NN as a closure – a drone example

Goal – short term navigation of a drone in a GPS deprived environment

Need – surface pressure and forces to predict position and correct course





We consider forces under angle of attack variations $\alpha(t) = \alpha_0 + A\sin(2\pi f t).$

Jan S Hesthaven

Using NN as a closure – a drone example

We express the surface pressure using a reduced basis

$$\boldsymbol{C}_p \approx \boldsymbol{C}_p^{\mathrm{rb}} = \overline{\boldsymbol{C}_p} + \sum_{i=1}^{n_b} b_i \boldsymbol{u}_i = \overline{\boldsymbol{C}_p} + \boldsymbol{U}\boldsymbol{b},$$

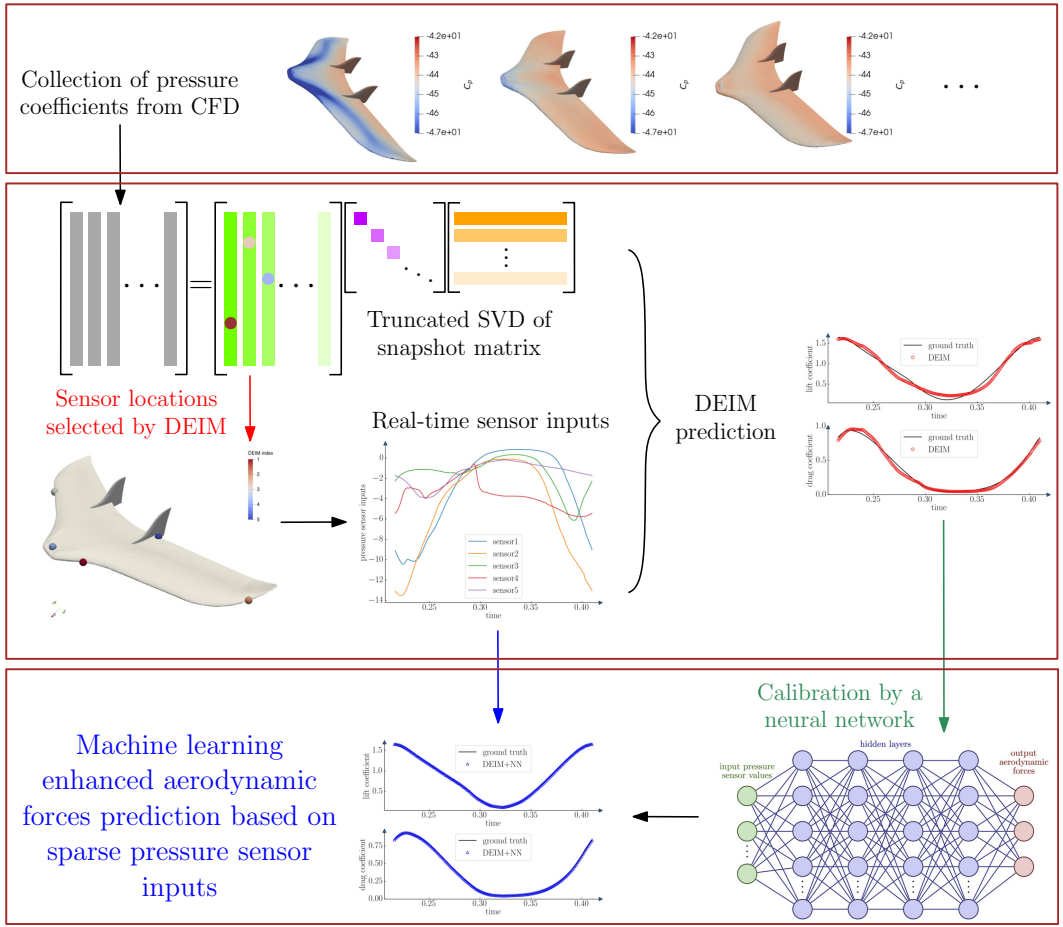
Coefficients are found from pressure measurements

$$\overline{\boldsymbol{C}_p}\,(\boldsymbol{\mathcal{I}}) + \boldsymbol{U}\,(\boldsymbol{\mathcal{I}}, :)\,\boldsymbol{b} = \boldsymbol{C}_p^{\mathrm{s}},$$

Sensor positions (I) are found by a DIEM process on CFD

Jan S Hesthaven

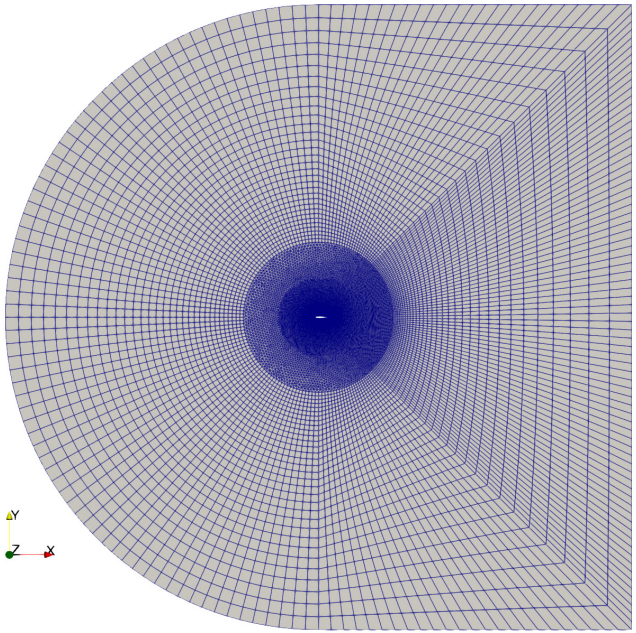
Using NN as a closure – a drone example

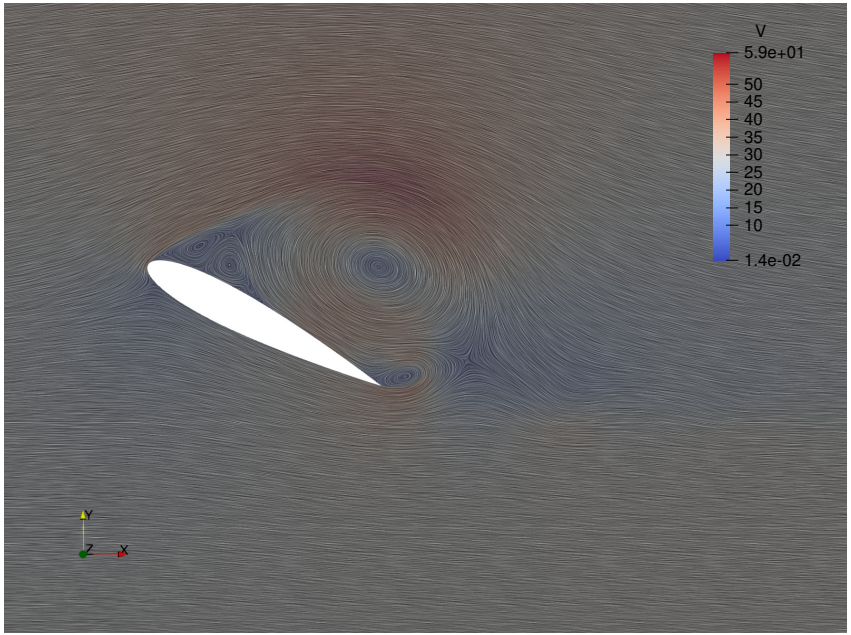


Jan S Hesthaven

Using NN as a closure – a drone example

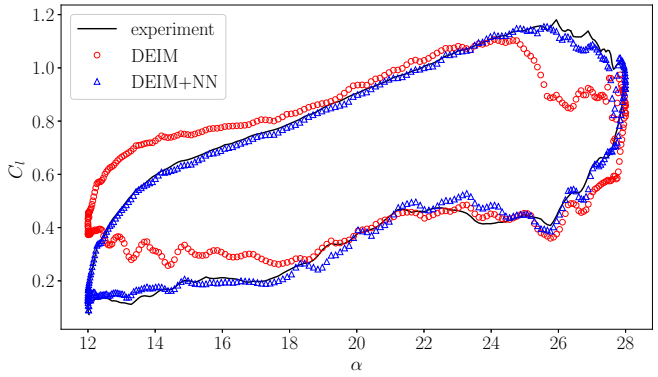
Let’s first consider a 2D airfoil



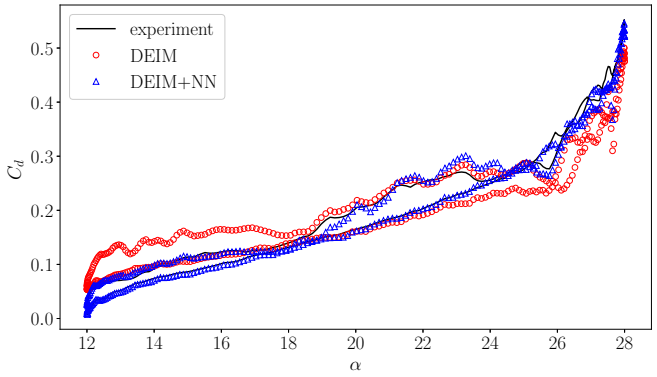


Jan S Hesthaven

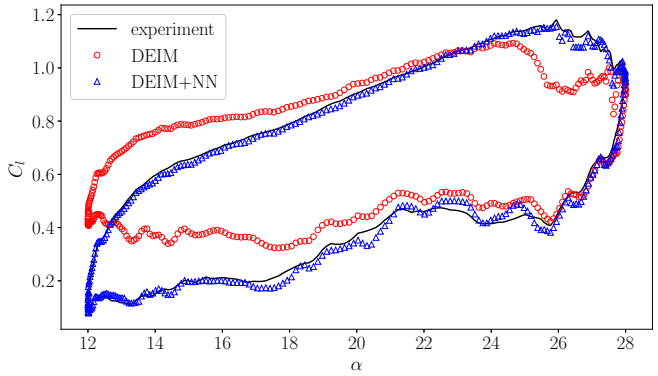
Using NN as a closure – a drone example



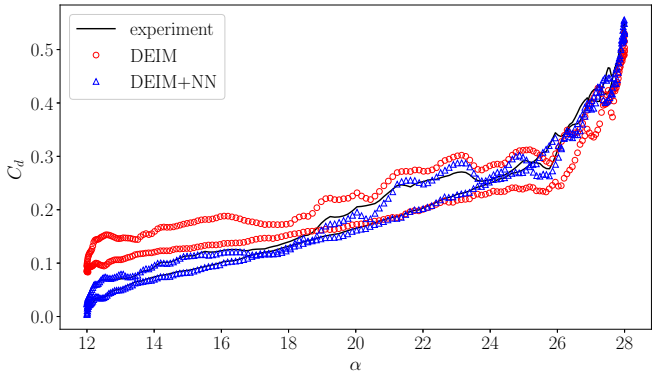
(c) $n_s = 8$, C_l



(d) $n_s = 8$, C_d



(e) $n_s = 10$, C_l



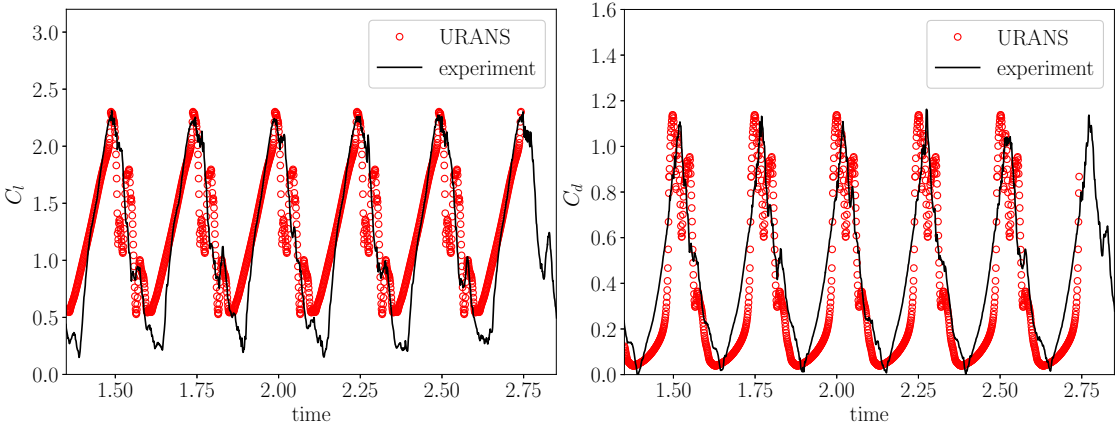
(f) $n_s = 10$, C_d

2D airfoil

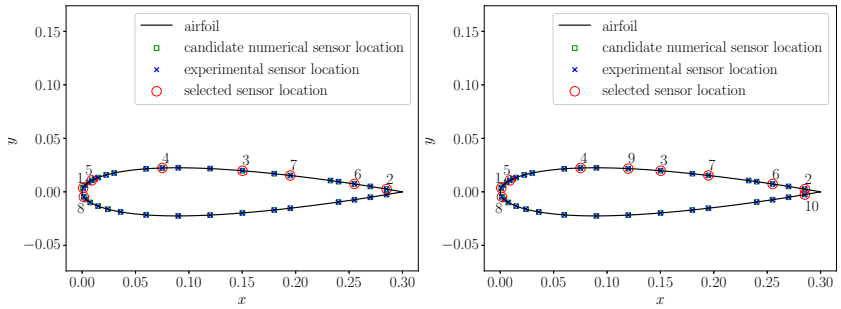
Jan S Hesthaven

Using NN as a closure – a drone example

We have access to validated URANS data and experimental data



(a) C_l **(b)** C_d



(b) $n_s = 8$ **(c)** $n_s = 10$

This is used to train an NN closure

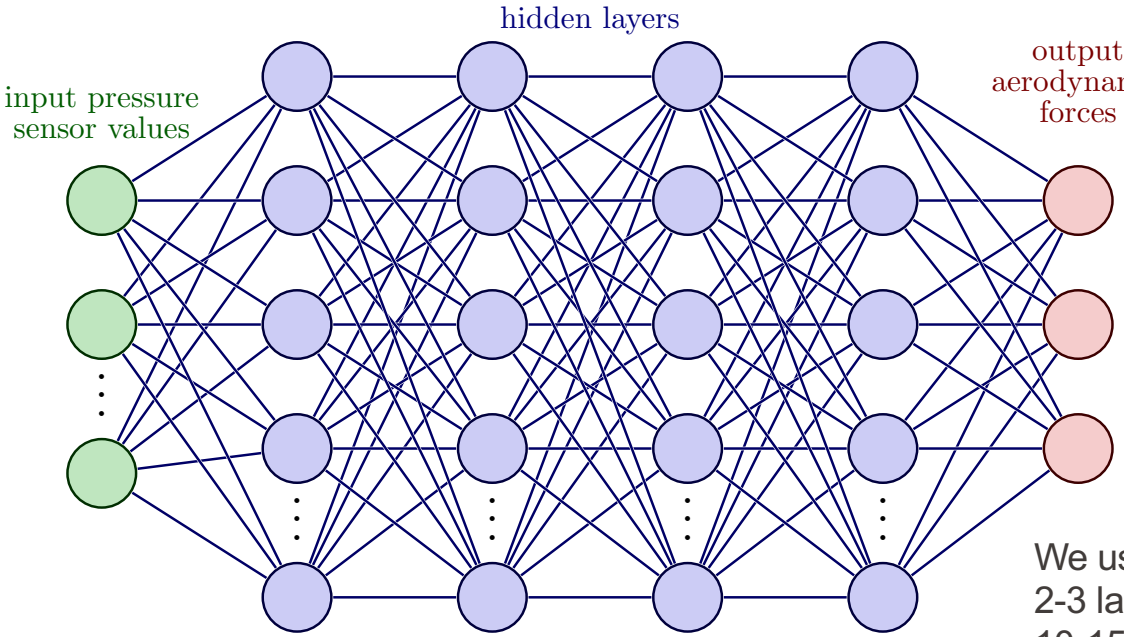
$$C_l^{\mathtt{NN}} = \left(\boldsymbol{F}^{\mathtt{DEIM}} + \mathrm{NN}(\boldsymbol{C}_p^{\mathtt{s}}, \Theta) \right) \cdot \boldsymbol{n}_l,$$

$$C_d^{\mathtt{NN}} = \left(\boldsymbol{F}^{\mathtt{DEIM}} + \mathrm{NN}(\boldsymbol{C}_p^{\mathtt{s}}, \Theta) \right) \cdot \boldsymbol{n}_d.$$

Jan S Hesthaven

NN in two slides

We consider a simple feedforward NN and supervised training

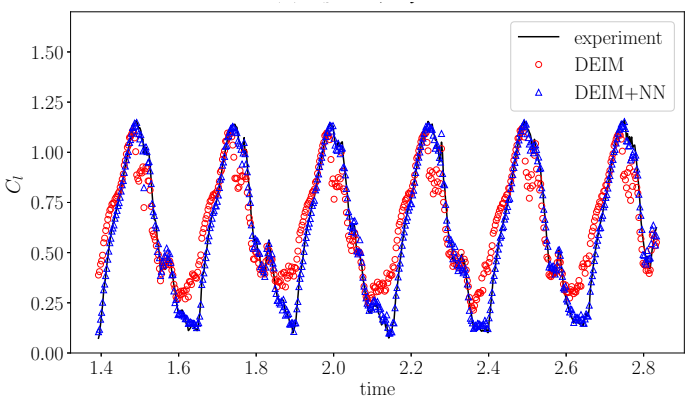


We use small networks
2-3 layers
10-15 neurons

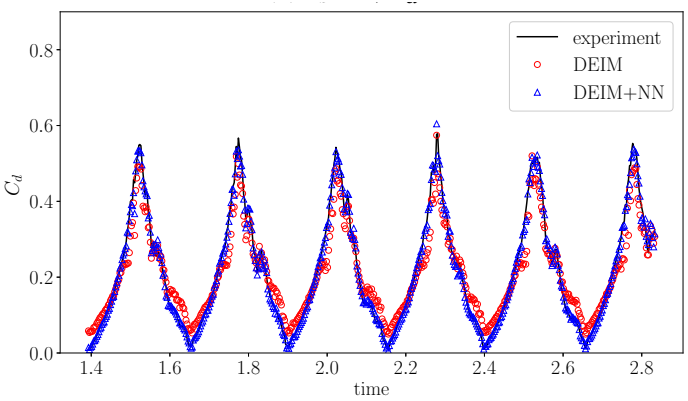
Creates a (nonlinear) map between input and output

Jan S Hesthaven

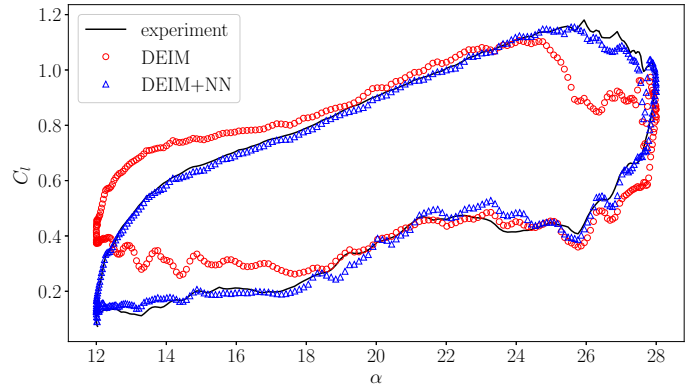
Using NN as a closure – a drone example



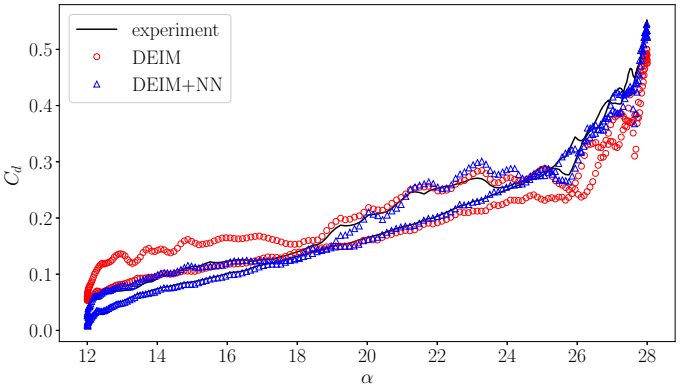
(c) $n_s = 8$, C_l



(d) $n_s = 8$, C_d



(c) $n_s = 8$, C_l



(d) $n_s = 8$, C_d

Same results if noise added to pressure signals

$\alpha_0 = 20\,\text{deg}$

$A = 8\,\text{deg}$

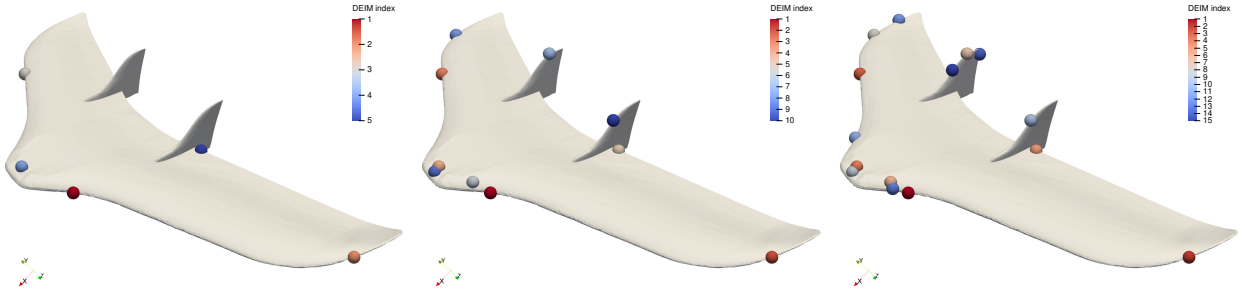
$Re = 5.4 \times 10^5$

Jan S Hesthaven

Using NN as a closure – a drone example

Full 3D drone – no experimental results – all based on URANS

Jan S Hesthaven



(a) $n_s = 5$ **(b)** $n_s = 10$ **(c)** $n_s = 15$

2D airfoil			3D drone		
n_s	DEIM	NN	n_s	DEIM	NN
5	2.41×10^{-6}	7.89×10^{-5}	5	3.60×10^{-6}	1.19×10^{-4}
8	2.28×10^{-6}	7.05×10^{-5}	10	2.41×10^{-6}	7.89×10^{-5}
10	2.86×10^{-6}	7.30×10^{-5}	15	2.41×10^{-6}	7.89×10^{-5}

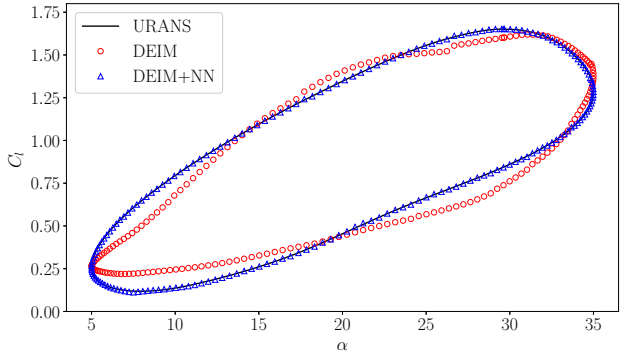
$\alpha_0 = 20\,\text{deg}$

$A = 15\,\text{deg}$

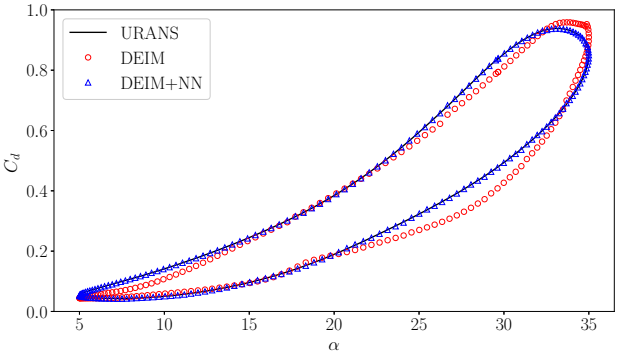
$Re = 3.6 \times 10^5$

Fast enough for real time

Using NN as a closure – a drone example

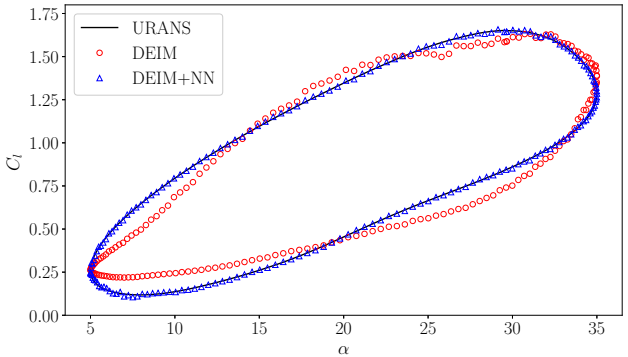


(a) $n_s = 5$, C_l

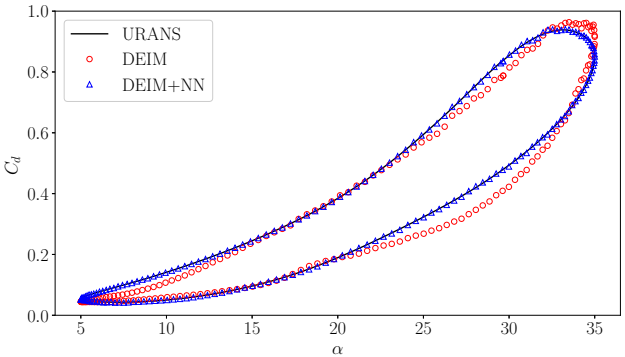


(b) $n_s = 5$, C_d

No added noise in pressure signal



(a) $n_s = 5$, C_l



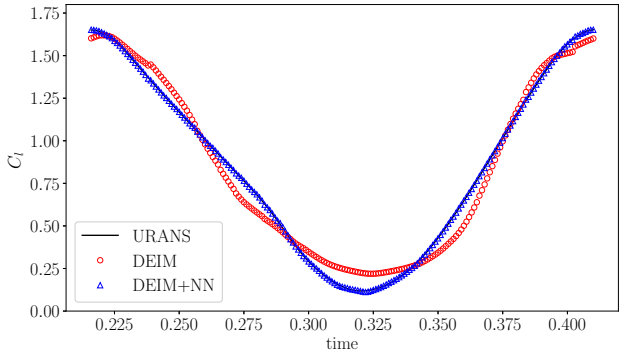
(b) $n_s = 5$, C_d

1.5% added noise in pressure signal

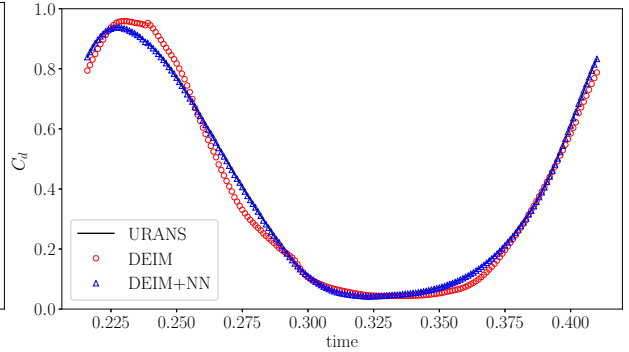
Jan S Hesthaven

Using NN as a closure – a drone example

Jan S Hesthaven

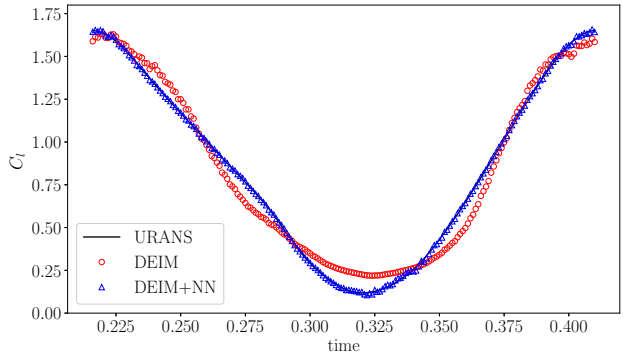


(a) $n_s = 5$, C_l

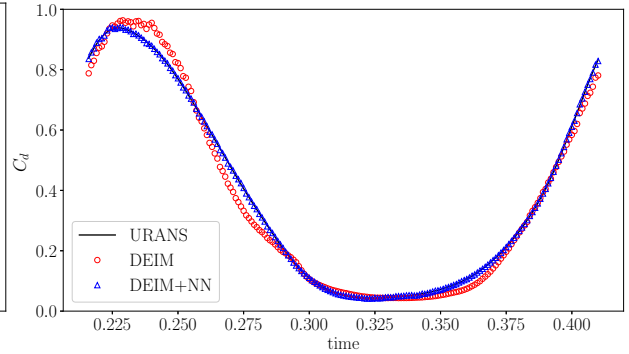


(b) $n_s = 5$, C_d

No added noise in pressure signal



(a) $n_s = 5$, C_l



(b) $n_s = 5$, C_d

1.5% added noise in pressure signal

A few remarks

- The closure problem is a general challenge which has proven hard to address through modeling
- Neural networks are not very accurate – but good enough for corrections
- Learning the closure terms open up for interesting opportunities.
- Both computational and experimental data can be used for this
- Even for simple problems, benefits can be demonstrated
- There are many open problems, eg transformers (ChatGPT) may be interesting to explore

Jan S Hesthaven

Some references

- J. S. Hesthaven, G. Rozza, and B. Stamm, 2016, *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, Springer Brief in Mathematics, Springer Verlag, Berlin, Germany,

- Q. Wang, N. Ripamonti and J.S. Hesthaven, 2022, Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori-Zwanzig formalism, J. Comput. Phys. 410, 109402.

- J. Duan, Q. Wang and J.S. Hesthaven, 2023, *Machine learning enhanced real-time aerodynamic force prediction based on sparse pressure point sensor input* - submitted


Thank you!