University of Stuttgart
Institute of Aerodynamics
and Gas Dynamics

Andrea Beck

Deep Neural Networks for Data-Driven Turbulence Models

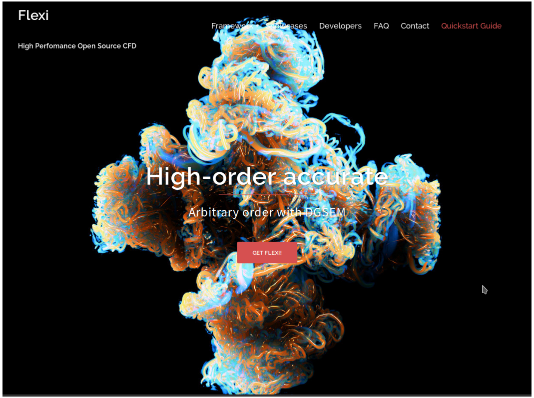
28th Workshop for Sustained Simulation Performance

Introduction

1

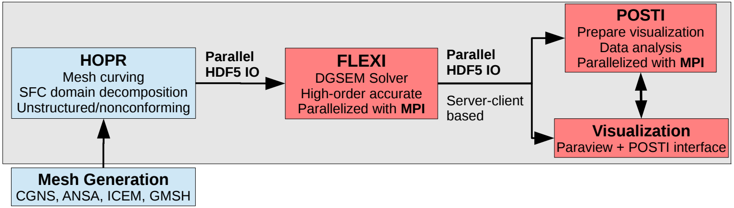
Introduction

- Numerics Research Group IAG, Uni Stuttgart, Prof. Munz
- Primary Focus: High Order Discontinuous Galerkin Methods
- OpenSource HPC solver for the compressible Navier-Stokes equations



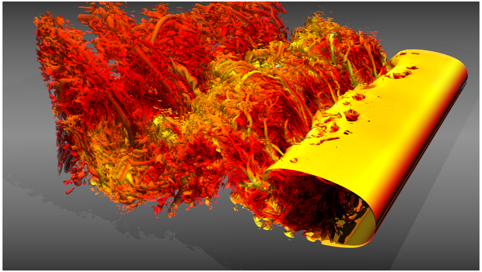
www.flexi-project.org

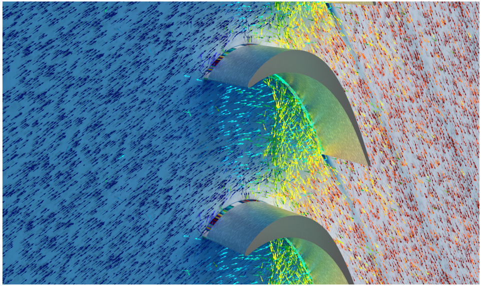
Framework

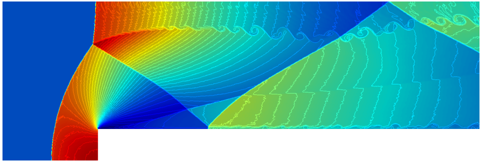


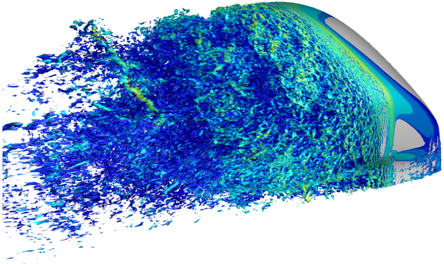
- FLEXI: Designed for solving unsteady compressible flows using the Discontinuous Galerkin Spectral Element Method (DGSEM)
- Very high orders possible ($\mathcal{O}16+$)
- Use explicit RK global time-stepping approach
- FLEXI comes with a variety of flux functions, RK schemes, Lifting procedures and boundary conditions implemented
- Support for relatively complex geometries using unstructured, non-conforming grids
- Shock capturing based on finite volume sub cells
- Highly parallel and scalable due to compact operator: DG is ”embarrassingly parallel”

Applications: DNS, LES, high Mach flows, direct aeroacustics, particle-laden flows...



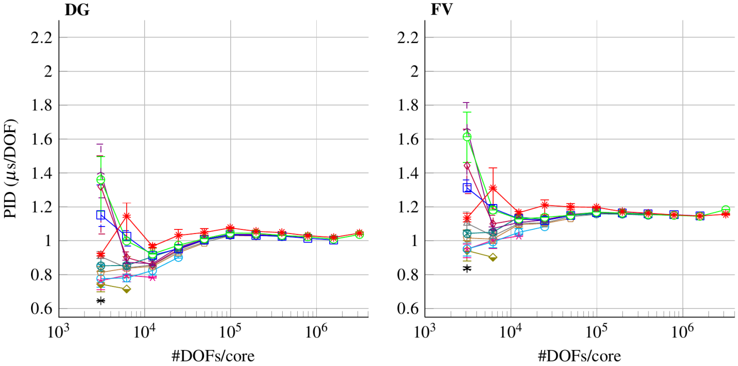






Parallel Efficiency

- Communication based on MPI
- Compact stencil in combination with latency hiding and optimized communication patters allow for strong scaling down to $\mathcal{O}(10^3)$ DOFs per core
- Efficiency still intact for combined FV/DG calculations
- Proven efficiency up to 100.000 cores
- Parallel I/O based on HDF5



Machine Learning with Neural Networks

2

Rationale for Machine Learning

"It is very hard to write programs that solve problems like recognizing a three-dimensional object from a novel viewpoint in new lighting conditions in a cluttered scene.

- We don't know what program to write because we don't know how its done in our brain.
- Even if we had a good idea about how to do it, the program might be horrendously complicated."

Geoffrey Hinton, computer scientist and cognitive psychologist (h-index:131)

Definitions and Concepts

An attempt at a definition:

Machine learning describes algorithms and techniques that progressively improve performance on a specific task through data without being explicitly programmed.

Learning Concepts

- Unspervised Learning
- Supervised Learning
- Reinforcement Learning

Artificial Neural Networks

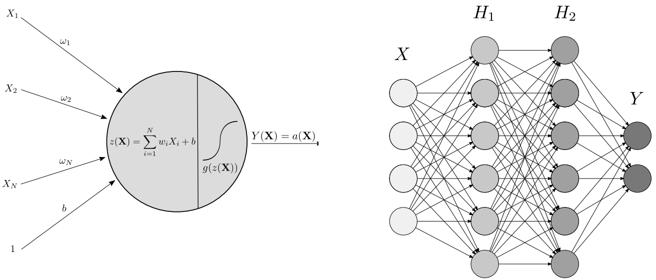
- General Function Approximators
- AlphaGo, Self-Driving Cars, Face recognition, NLP
- Incomplete Theory, models difficult to interpret

Neural Networks

- Artificial Neural Network (ANN): A non-linear mapping from inputs to ouputs: $\mathbf{M} : \hat{X} \rightarrow \hat{Y}$
- An ANN is nesting of linear and non-linear functions arranged in a directed acyclic graph:

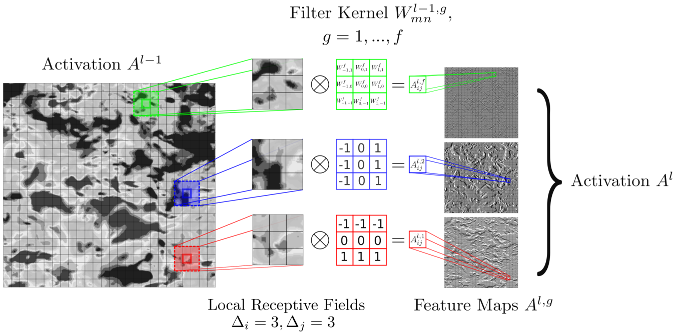
$$\hat{Y} \approx Y = M(\hat{X}) = \sigma_L \left(W_L \left(\sigma_{L-1} \left(W_{L-1} \left(\sigma_{L-2} \left(...W_1(\hat{X}) \right) \right) \right) \right) \right), \quad (1)$$

- with W being an affine mapping and σ a non-linear function
- The entries of the mapping matrices W are the parameters or weights of the network: improved by training
- Cost function C as a measure for $\left|\hat{Y} - Y\right|$, (MSE / L_2 error) convex w.r.t to Y, but not w.r.t W: $\Rightarrow$ non-convex optimization problem requires a lot of data



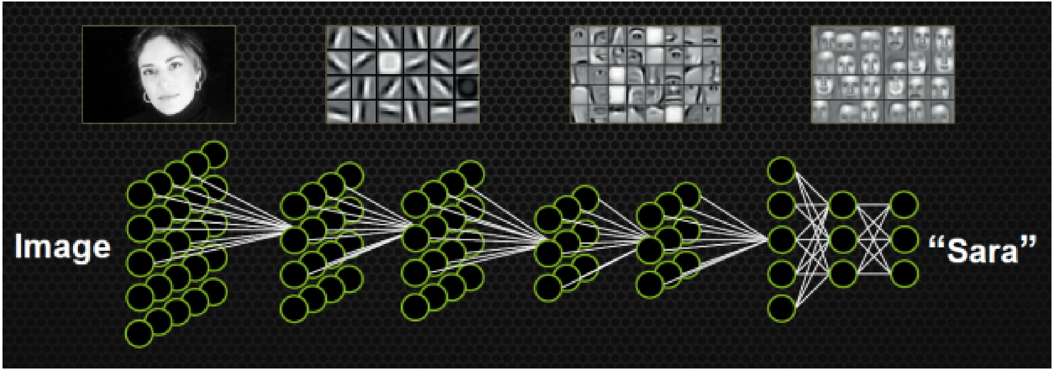
Advanced Architectures

- Convolutional Neural Networks
  - Local connectivity, multidimensional trainable filter kernels, discrete convolution, shift invariance
  - Current State of the Art for multi-D data



What does a CNN learn?

- Hierarchical representation



from: H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." In ICML 2009.

3

Turbulence Models from Data

Turbulence Closure

- Turbulent fluid motion is prevalent in engineering applications: multiscale problem in space and time
- Navier-Stokes equations: system of non-linear PDEs (hyp. / parab.)
- Fullscale resolution (DNS) rarely feasible: Coarse scale formulation of NSE is necessary
- Filtering the NSE: Evolution equations for the coarse scale quantities, but with a closure term dependent on the filtered full scale solution $\Rightarrow$ Model depending on the coarse scale data needed!

40+ years of research in Turbulence Modeling:
"All models are wrong, some models are useful"

- Filtered NSE:

$$\frac{\partial \overline{U}}{\partial t} + \overline{R(F(U))} = 0 \tag{2}$$

- Imperfect closure with $\hat{U} \neq \overline{U}$:

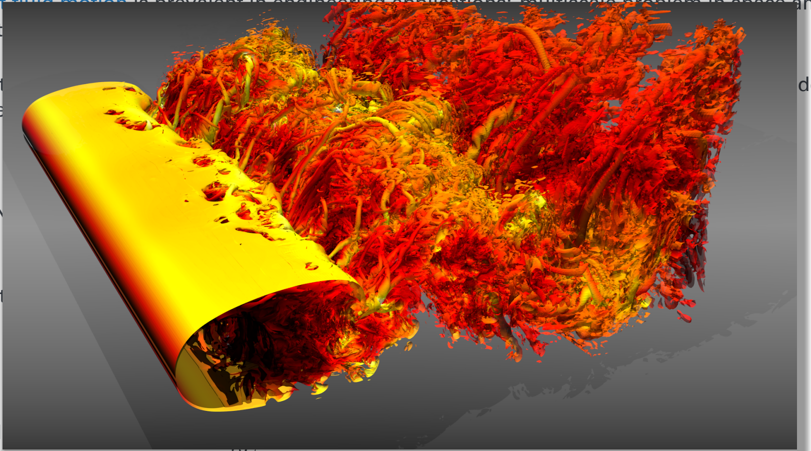
$$\frac{\partial \hat{U}}{\partial t} + \widetilde{R}(F(\hat{U})) = \underbrace{\widetilde{M}(\hat{U}, C_k)}_{\textbf{imperfect closure model}} , \tag{3}$$

- Perfect closure with $\overline{U}$

$$\frac{\partial \overline{U}}{\partial t} + \widetilde{R}(F(\overline{U})) = \underbrace{\widetilde{R}(F(\overline{U})) - \overline{R(F(U))}}_{\textbf{perfect closure model}} . \tag{4}$$

Turbulence Closure

- Turbulent fluid motion is prevalent in engineering applications: multiscale problem in space and time
- Navier-St
- Fullscale
- Filtering t ... dependent on the filtere



- Filtered N

(2)

- Imperfect

(3)

- Perfect cl

$$\frac{\partial \overline{U}}{\partial t} + \widetilde{R}(F(\overline{U})) = \underbrace{\widetilde{R}(F(\overline{U})) - \overline{R(F(U))}}_{\textbf{perfect closure model}}. \quad (4)$$

Turbulence Closure

- Turbulent fluid motion is prevalent in engineering applications: multiscale problem in space and time
- Navier-Stokes equations: system of non-linear PDEs (hyp. / parab.)
- Fullscale resolution (DNS) rarely feasible: Coarse scale formulation of NSE is necessary
- Filtering the NSE: Evolution equations for the coarse scale quantities, but with a closure term dependent on the filtered full scale solution $\Rightarrow$ Model depending on the coarse scale data needed!

40+ years of research in Turbulence Modeling:
"All models are wrong, some models are useful"

- Filtered NSE:

$$\frac{\partial \overline{U}}{\partial t} + \overline{R(F(U))} = 0 \tag{2}$$

- Imperfect closure with $\hat{U} \neq \overline{U}$:

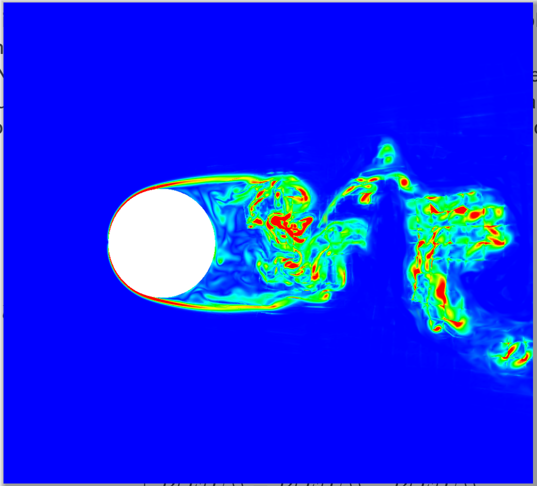
$$\frac{\partial \hat{U}}{\partial t} + \widetilde{R}(F(\hat{U})) = \underbrace{\widetilde{M}(\hat{U}, C_k)}_{\textbf{imperfect closure model}} , \tag{3}$$

- Perfect closure with $\overline{U}$

$$\frac{\partial \overline{U}}{\partial t} + \widetilde{R}(F(\overline{U})) = \underbrace{\widetilde{R}(F(\overline{U})) - \overline{R(F(U))}}_{\textbf{perfect closure model}} . \tag{4}$$

Turbulence Closure

- Turbulent fluid motion [...]lem in space and time
- Navier-Stokes equation[...]
- Fullscale resolution (DN[...]ecessary
- Filtering the NSE: Evolu[...] closure term dependent on the filtered full scale so[...]ded!



- Filtered NSE:

(2)

- Imperfect closure with [...]

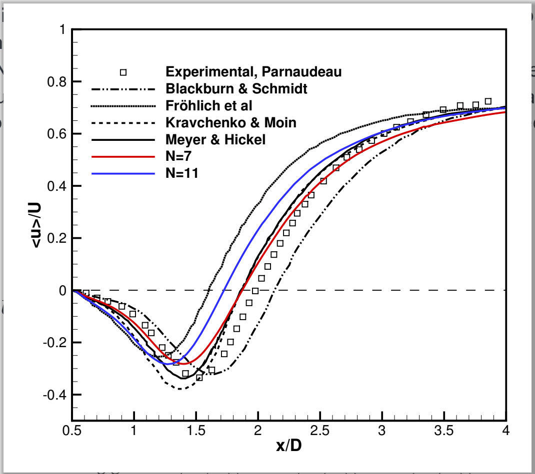
(3)

- Perfect closure with $\overline{U}$

$$\frac{\partial \overline{U}}{\partial t} + R(F(\overline{U})) = \underbrace{R(F(\overline{U})) - \overline{R(F(U))}}_{\text{perfect closure model}}. \quad (4)$$

Turbulence Closure

- Turbulent fluid motion i… …lem in space and time
- Navier-Stokes equation…
- Fullscale resolution (DN… …ecessary
- Filtering the NSE: Evolu… …a closure term dependent on the filtered full scale so… …ded!



- Filtered NSE: (2)

- Imperfect closure with … (3)

- Perfect closure with $\overline{U}$

$$\frac{\partial \overline{U}}{\partial t} + R(F(\overline{U})) = \underbrace{R(F(\overline{U})) - \overline{R(F(U))}}_{\text{perfect closure model}}. \quad (4)$$

Turbulence Closure

- Turbulent fluid motion is prevalent in engineering applications: multiscale problem in space and time
- Navier-Stokes equations: system of non-linear PDEs (hyp. / parab.)
- Fullscale resolution (DNS) rarely feasible: Coarse scale formulation of NSE is necessary
- Filtering the NSE: Evolution equations for the coarse scale quantities, but with a closure term dependent on the filtered full scale solution $\Rightarrow$ Model depending on the coarse scale data needed!

40+ years of research in Turbulence Modeling:
"All models are wrong, some models are useful"

- Filtered NSE:

$$\frac{\partial \overline{U}}{\partial t} + \overline{R(F(U))} = 0 \tag{2}$$

- Imperfect closure with $\hat{U} \neq \overline{U}$:

$$\frac{\partial \hat{U}}{\partial t} + \widetilde{R}(F(\hat{U})) = \underbrace{\widetilde{M}(\hat{U}, C_k)}_{\textbf{imperfect closure model}} \quad , \tag{3}$$

- Perfect closure with $\overline{U}$

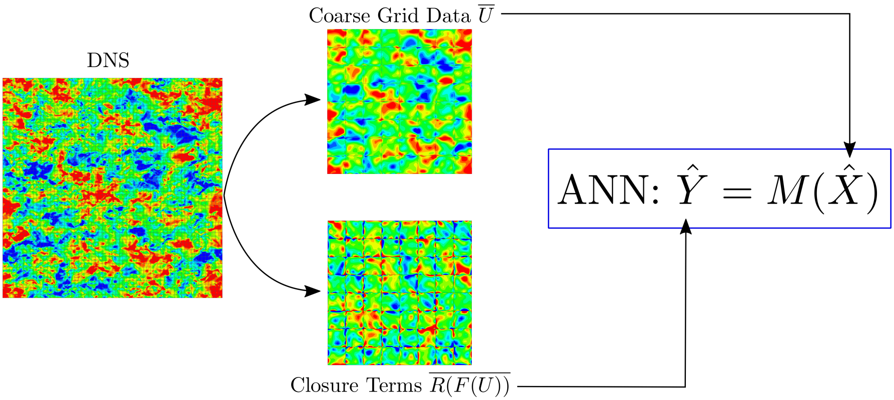
$$\frac{\partial \overline{U}}{\partial t} + \widetilde{R}(F(\overline{U})) = \underbrace{\widetilde{R}(F(\overline{U})) - \overline{R(F(U))}}_{\textbf{perfect closure model}} . \tag{4}$$

Idea

- Approximating an unknown, non-linear and possibly hierarchical mapping from high-dimensional input data to an output $\Rightarrow$ ANN

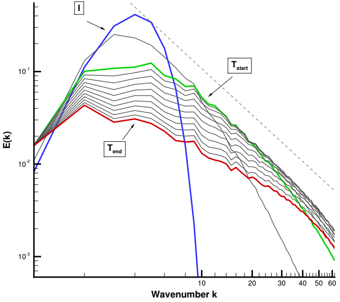
Idea

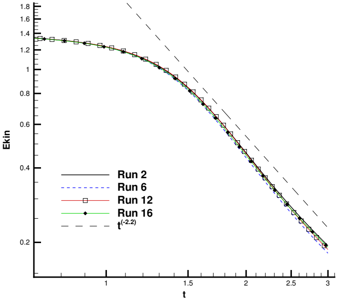
- Approximating an unknown, non-linear and possibly hierarchical mapping from high-dimensional input data to an output $\Rightarrow$ LES closure



Data Acquisition: Decaying Homogeneous Isotropic Turbulence

- DNS of decaying homogeneous isotropic turbulence (DHIT) with initial spectrum defined by Chasnov (1995) initialized by Rogallo (1981) procedure
- Data collection in the range of exponential energy decay: 25 DHIT realizations with 134 Mio DOF each computed on Hazel Hen (approx. 400,000 CPUh, 8200 cores)
- Compute coarse grid terms on LES grid with filter definition

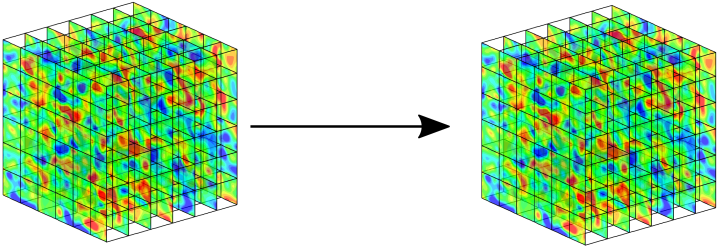




Features and Labels

- Each sample: A single LES grid cell with 6^3 solution points
- Input features: velocities and LES operator: $\overline{u_i}, \widetilde{R}(F(\overline{U}))$
- Output labels: DNS closure terms on the LES grid $\overline{R(F(U))}$

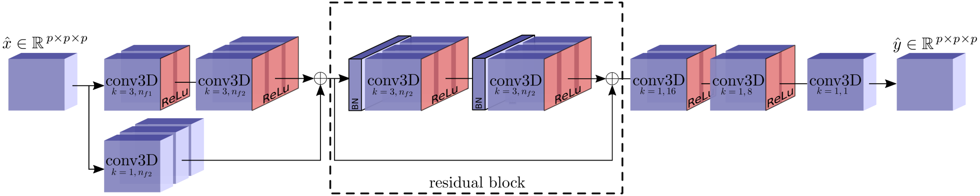
$$\hat{X} = \left\{ \hat{x} \in \mathbb{R}^{6 \times p \times p \times p} \mid \hat{x} = (\overline{u}_{ijk}, \overline{v}_{ijk}, \overline{w}_{ijk}, \widetilde{R}(F(\overline{U^1}))_{ijk}, \widetilde{R}((F(\overline{U^2}))_{ijk}, \widetilde{R}(F(\overline{U^3}))_{ijk}), \quad \text{with } i,j,k = 0, ..., p-1 \right\}$$



$$\hat{Y} = \left\{ \hat{y} \in \mathbb{R}^{3 \times p \times p \times p} \mid \hat{y} = \overline{R(F(U))}^{n}_{ijk}, \ \text{ with } n = 1, ..., 3; \ i,j,k = 0, ..., p-1 \right\}$$

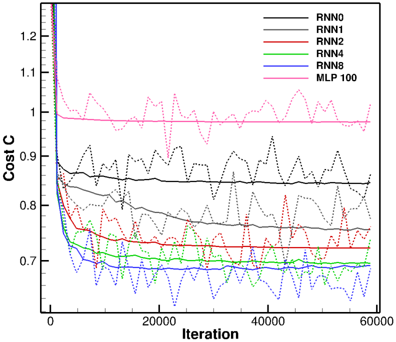
Networks and Training

- CNNs with skip connections (RNN), batch normalization, ADAM optimizer, data augmentation
- Different network depths (no. of residual blocks)
- Implementation in Python / Tensorflow, Training on K40c and P100 at HLRS
- Split in training, semi-blind validation and blind test DHIT runs



Training Results I: Costs

- Cost function for different network depths
- RNNs outperform MLP, deeper networks learn better
- The approach is data-limited! NNs are very data-hungry!



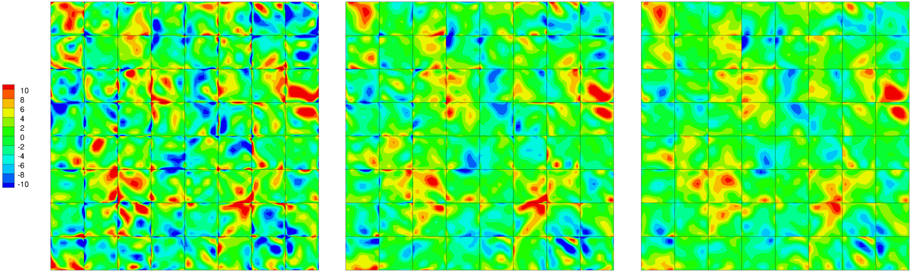
Training Results II: Correlation

Network	a, b	$\mathcal{CC}(a,b)$	$\mathcal{CC}^{inner}(a,b)$	$\mathcal{CC}^{surf}(a,b)$
RNN0	$\overline{R(F(U))^1}, \overline{R(F(U))^1}^{ANN}$	0.347676	0.712184	0.149090
	$\overline{R(F(U))^2}, \overline{R(F(U))^2}^{ANN}$	0.319793	0.663664	0.134267
	$\overline{R(F(U))^3}, \overline{R(F(U))^3}^{ANN}$	0.326906	0.669931	0.101801
RNN4	$\overline{R(F(U))^1}, \overline{R(F(U))^1}^{ANN}$	0.470610	0.766688	0.253925
	$\overline{R(F(U))^2}, \overline{R(F(U))^2}^{ANN}$	0.450476	0.729371	0.337032
	$\overline{R(F(U))^3}, \overline{R(F(U))^3}^{ANN}$	0.449879	0.730491	0.269407

Training Results III: Visualization

- "Blind" application of the trained network to unknown test data

$\overline{R(F(U))^1}$ $\qquad$ $\overline{R(F(U))^1}^{ANN_4}$ $\qquad$ $\overline{R(F(U))^1}^{ANN_0}$



Training Results IV: Feature Sensitivity

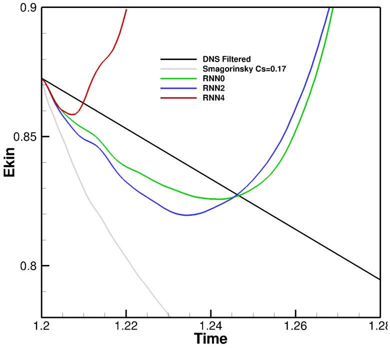
Set	Features	$\mathcal{CC}^1$	$\mathcal{CC}^2$	$\mathcal{CC}^3$
1	$u_i,\ \widetilde{R}(F(\overline{U^i})),\ i=1,2,3$	0.4706	0.4505	0.4499
2	$u_i,\ i=1,2,3$	0.3665	0.3825	0.3840
3	$\widetilde{R}(F(\overline{U^i})),\ i=1,2,3$	0.3358	0.3066	0.3031
4	$\rho, p, e, u_i,\ \widetilde{R}(F(\overline{U^i})),\ i=1,2,3$	0.4764	0.4609	0.4580
5	$u_1,\ \widetilde{R}(F(\overline{U^1}))$	0.3913		

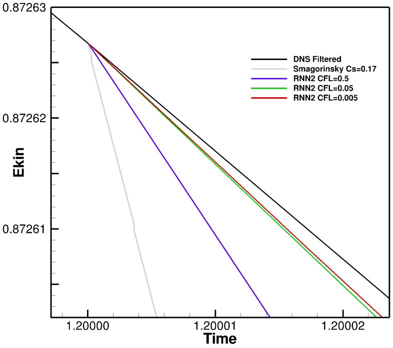
Feature sets and resulting test correlations. $\mathcal{CC}^i$ with $i=1,2,3$ denotes the cross correlation between the targets and network outputs $\mathcal{CC}(\overline{R(F(U)^i)}, \overline{R(F(U))^i}^{ANN})$. Set 1 corresponds to the original feature choice; Set 5 corresponds to the RNN4 architecture, but with features and labels for the $u-$momentum component only.

- Both the coarse grid primitive quantities as well as the coarse grid operator contribute strongly to the learning success
- Better learning for full 3D data than 1D data only

LES with NN-trained model I

- Perfect LES is possible (see above), but the NN-learned mappings are approximate ⇒ Direct application in the sense of Eqn. 4 not long-term stable!
- Short term stability and dissipation only



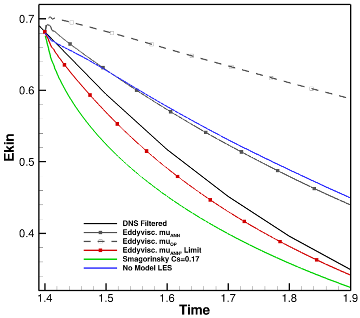


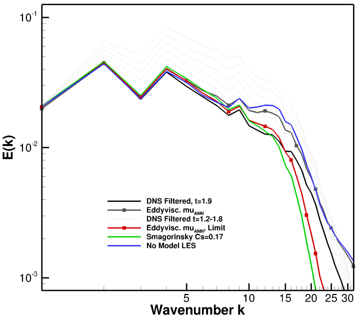
LES with NN-trained model II

- Simplest model: Eddy viscosity approach with μ_{ANN} from

$$\widetilde{R}(F(\overline{U^i})) - \overline{R(F(U^i))} \approx \mu_{ANN}\, \widetilde{R}(F^{visc}(\overline{U^i}, \nabla\overline{U^i})) \tag{5}$$

- Limit: $-\mu_0 \leq \mu_{ANN} \leq 20\mu_0$





Summary

- Learning the exact closure terms from data is possible
- Deeper RNNs learn better
- Our process is data-limited, i.e. learning can be improved with more data
- Achievable $\mathcal{CC} \approx 45\%$, with up to $\approx 75\%$ for inner points
- Both the coarse grid velocities and the coarse grid operator contribute strongly to learning (backup slides)
- The resulting ANN models are dissipative (not shown)
- No long term stability due to approximate model
- Simplest way to construct a stable model: Data-informed, local eddy-viscosity
- Other approaches to construct models from prediction of closure terms under investigation

Discussion

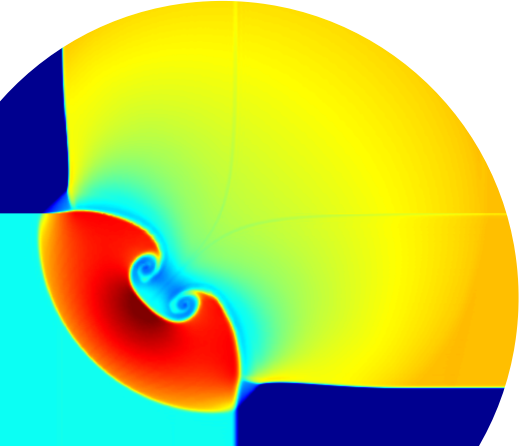
4

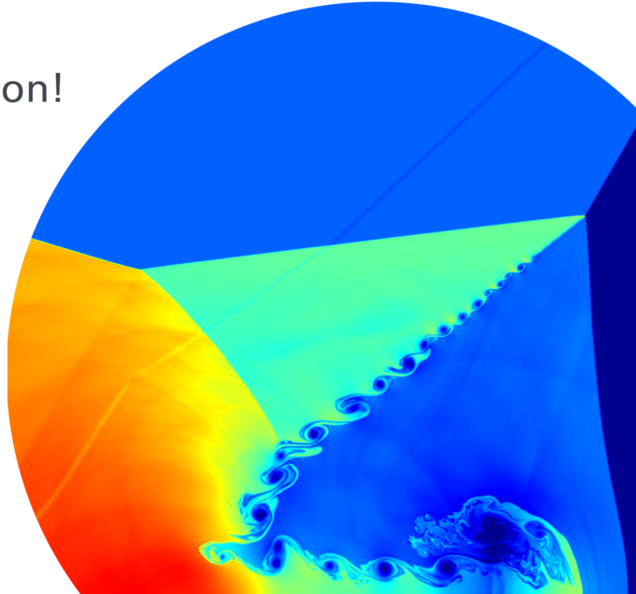
Some thoughts on data-informed models, engineering and HPC

- Machine Learning is not a silver bullet
- First successes: ML can help build subscale models from data, not just for turbulence
- A lot of representative data is needed... maybe we already have the data? Computations, experiments...
- In this work, the computational times were: DNS: $\mathcal{O}(10^5)$ CPUh, data preparation $\mathcal{O}(10^3)$, Training the RNN: $\mathcal{O}(10^1 - 10^2)$: Is it worth it?
- Incorporating physical constraints (e.g. realizability, positivity) field of research
- Self-learning algorithms: Reinforcement learning
- "Philosophical aspects": Interpretability of the models and "who should learn what?"
- HPC: Training has to done on GPUs (easy for supervised learning, bit more complicated for reinforcement learning), but ...
- What about model deployment? GPU (native) or CPU (export model)?
- Coupling of CFD solver (Fortran) to Neural Network (python): In our case, f2py is a very cumbersome solution
- Hybrid CPU/GPU codes, or rewrite it all for the GPU?

flexi-project.org

Thank you for your attention!





History of ANNs

- Some important publications:
  - McCulloch-Pitts (1943): First compute a weighted sum of the inputs from other neurons plus a bias: the perceptron
  - Rosenblatt (1958): First to generate MLP from perceptrons
  - Rosenblatt (1962): Perceptron Convergence Theorem
  - Minsky and Papert (1969): Limitations of perceptrons
  - Rumelhart and Hinton (1986): Backpropagation by gradient descent
  - LeCun (1995): “LeNet”, convolutional networks
  - Hinton (2006): Speed-up of backpropagation
  - Krizhevsky (2012): Convolutional networks for image classification
  - Ioffe (2015): Batch normalization
  - He et al. (2016): Residual networks
  - AlphaGo, DeepMind…

Closure Terms for LES

- For grid dependent LES: coarse grid operator is part of the closure
- Dual role of closure: cancel operator effects and model unknown term
- DNS grid: 64^3 elements, $N = 7$; LES grid: 8^3 elements, $N = 5$;

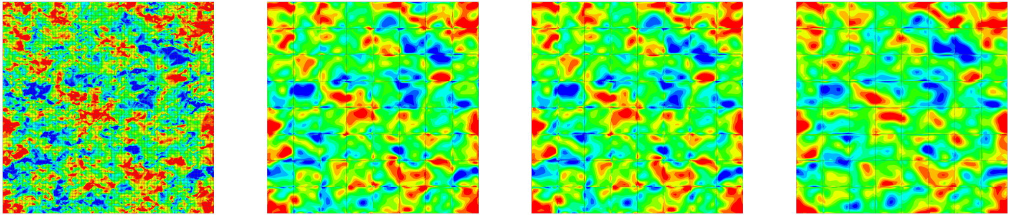


Figure: Left to right: a) DNS, b) filtered DNS, c) computed perfect LES d) LES with Smagorinsky model $C_s = 0.17$