University of Stuttgart
Institute of Aerodynamics
and Gas Dynamics

Andrea Beck

Deep Neural Networks for Data-Driven Turbulence Models

@HLRS-DL 2020

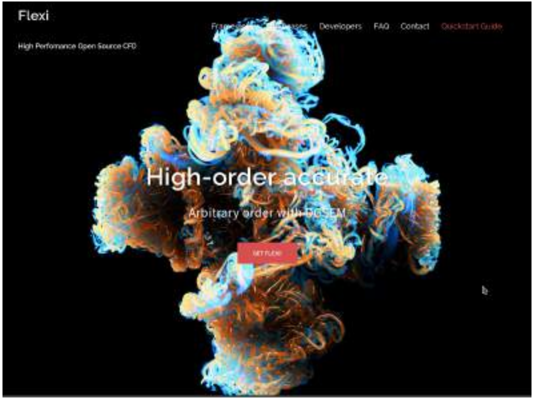
Outline

Introduction

1

Introduction

- Numerics Research Group @ IAG, University of Stuttgart, Germany
- Primary Focus: High Order Discontinuous Galerkin Methods
- OpenSource HPC solver for the compressible Navier-Stokes equations



www.flexi-project.org

DG-SEM in a nutshell

- Hyperbolic/parabolic conservation law , e.g. compressible Navier-Stokes Equations

$$U_t + \vec{\nabla} \cdot \vec{F}(U, \vec{\nabla} U) = 0$$

- Variational formulation and weak DG form per element for the equation system

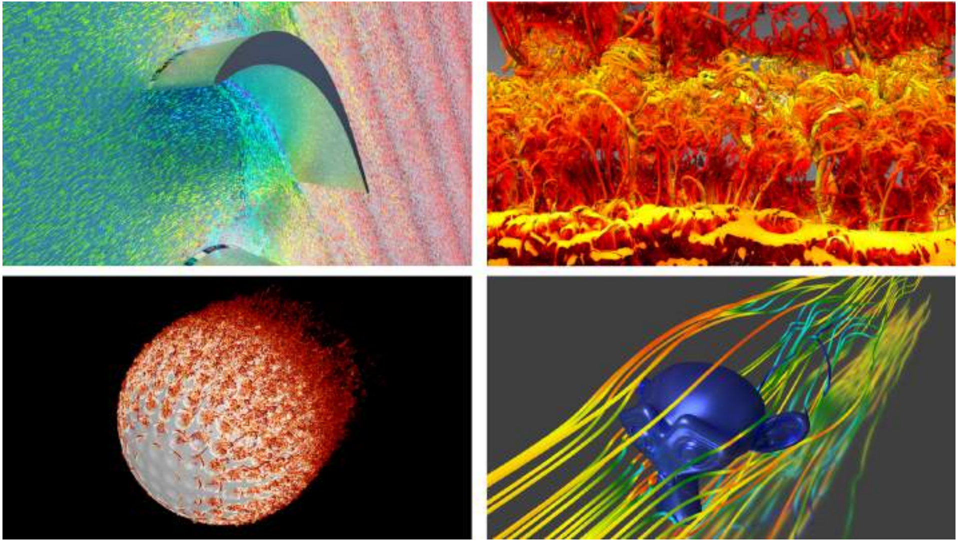
$$\langle J\, U_t, \psi \rangle_E + \left(\widetilde{f}^* \, \vec{n}_\xi, \psi \right)_{\partial E} - \left\langle \tilde{\vec{F}}, \nabla_\xi \psi \right\rangle_E = 0,$$

- Local tensor-product Lagrange polynomials, interpolation nodes equal to quadrature nodes
- Tensor-product structure in multi-D: line-by-line operations

$$(U_{ij})_t + \frac{1}{J_{ij}} \left[\widetilde{f}^*(1, \eta_j) \hat{\psi}_i(1) - \widetilde{f}^*(-1, \eta_j) \hat{\psi}_i(-1) + \sum_{k=0}^{N} \hat{D}_{ik} \, \widetilde{F}_{kj} \right] + \underbrace{\frac{1}{J_{ij}} \left[\widetilde{g}^*(\xi_i, 1) \hat{\psi}_j(1) - \widetilde{g}^*(\xi_i, -1) \hat{\psi}_j(-1) + \sum_{k=0}^{N} \hat{D}_{jk} \, \widetilde{G}_{ik} \right]}_{\text{1D DGSEM Operator}} = 0$$

- BR1/2 lifting for viscous fluxes, Roe/LF/HLL-type inviscid fluxes, explicit in time by RK/ Legendre-Gauss or LGL-nodes

Applications: LES, moving meshes, acoustics, multiphase, UQ, particle-laden flows...



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:140+)

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
- NN design: more an art than a science

Types of ML

Different Types of Learning:

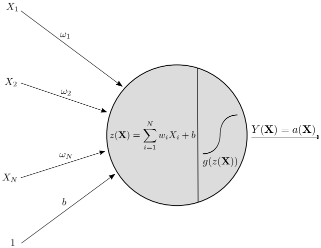
- Unsupervised learning:
  Discover a good internal representation of the input. ⇒ “Segmentation / Clustering Model”
- Reinforcement learning:
  Learn to select an action to maximize payoff. ⇒ “Behavioral Model”
- Supervised learning:
  Learn to predict an output when given an input vector. ⇒ “Predictive Model”

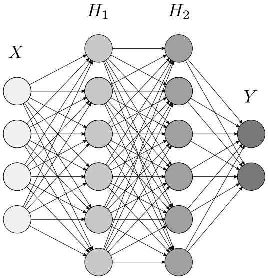
History of ANNs

- 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
  - Cybenko (1989): A NN with a single hidden layer and finite neurons can approximate continuous functions
  - 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...

Neural Networks

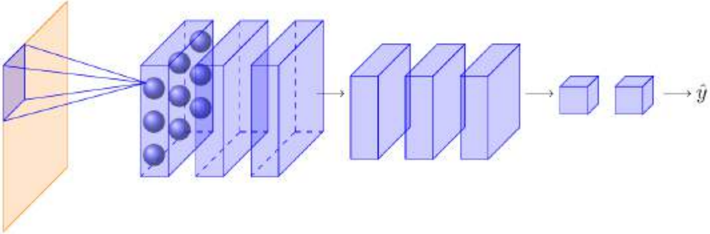
- Artificial Neural Network (ANN): A non-linear mapping from inputs to ouputs: $\mathbf{M} : \hat{X} \rightarrow \hat{Y}$
- An ANN is a nesting of linear and non-linear functions arranged in a directed acyclic graph:
$$\hat{Y} \approx Y = M(\hat{X}) = \sigma_L \left(W_L \left(\sigma_{L-1} \left(W_{L-1} \left(\sigma_{L-2} \left(...W_1(\hat{X}) \right) \right) \right) \right) \right), \tag{1}$$
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





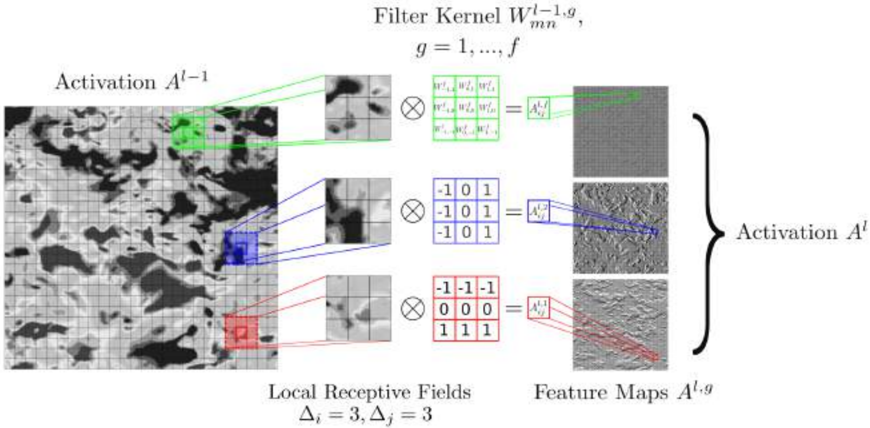
Advanced Architectures

- Convolutional Neural Networks
  - Local connectivity, multidimensional trainable filter kernels, discrete convolution, shift invariance, hierarchical representation
  - Current state of the art for multi-D data and segmentation



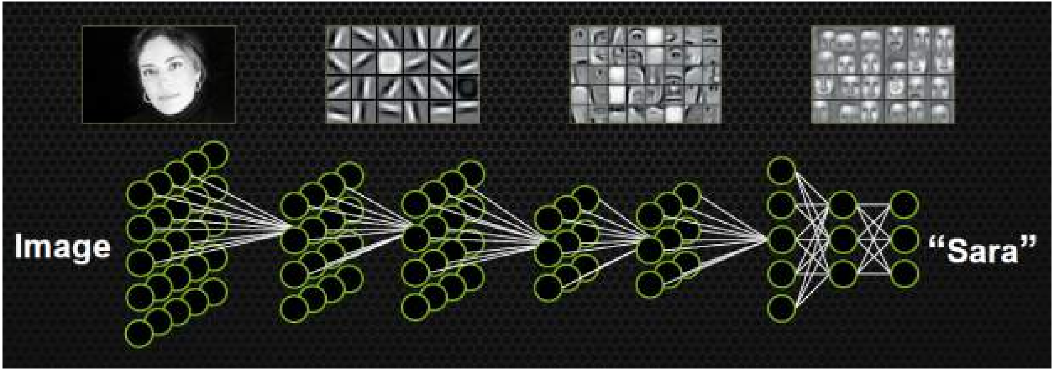
Advanced Architectures

- Convolutional Neural Networks



What does a CNN learn?

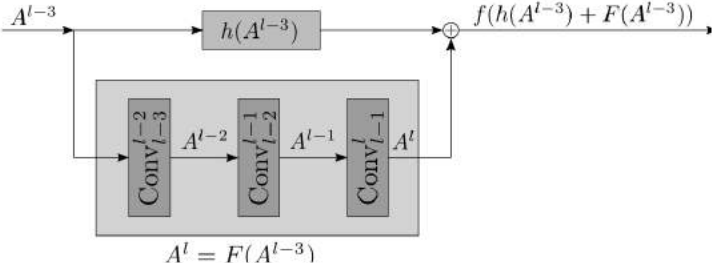
- Representation in hierarchical basis



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.

Residual Neural Networks

- He et al. recognized that the prediction performance of CNNs may deteriorate with depths (not an overfitting problem)
- Introduction of skip connectors or shortcuts, most often identity mappings
- A sought mapping, e.g. $G(A^{l-3})$ is split into a linear and non-linear (residual) part
- Fast passage of the linear part through the network: hundreds of CNN layers possible
- More robust identity mapping



He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

Turbulence Models from Data

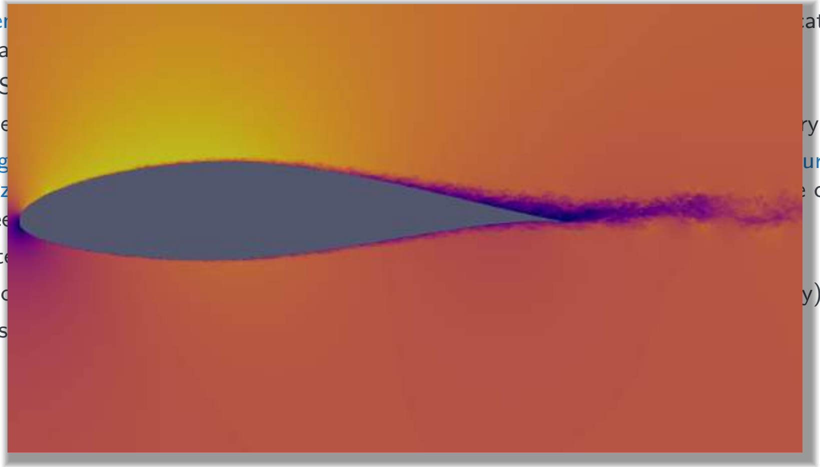
3

Turbulence in a nutshell

- Turbulent fluid motion is prevalent in naturally occurring flows and 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 / regularization dependent on the filtered full scale solution ⇒ Model depending on the coarse scale data needed!
- Two filter concepts: Averaging in time (RANS) or low-pass filter in space (LES)
- An important consequence: RANS can be discretization independent, LES is (typically) not!
- 50 years of research: Still no universal closure model

Turbulence in a nutshell

- Turbuler ations: multisca
- Navier-S
- Fullscale ry
- Filtering ure term / regulariz coarse scale data nee
- Two filte
- An impo y) not!
- 50 years

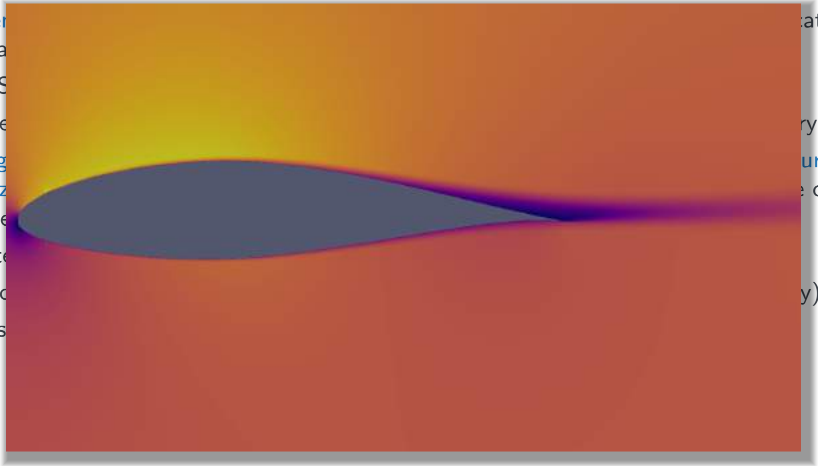


Turbulence in a nutshell

- Turbulent fluid motion is prevalent in naturally occurring flows and 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 / regularization dependent on the filtered full scale solution ⇒ Model depending on the coarse scale data needed!
- Two filter concepts: Averaging in time (RANS) or low-pass filter in space (LES)
- An important consequence: RANS can be discretization independent, LES is (typically) not!
- 50 years of research: Still no universal closure model

Turbulence in a nutshell

- Turbuler ations:
  multisca
- Navier-S
- Fullscale ry
- Filtering ure term /
  regulariz coarse scale
  data nee
- Two filte
- An impo y) not!
- 50 years



Turbulence in a nutshell

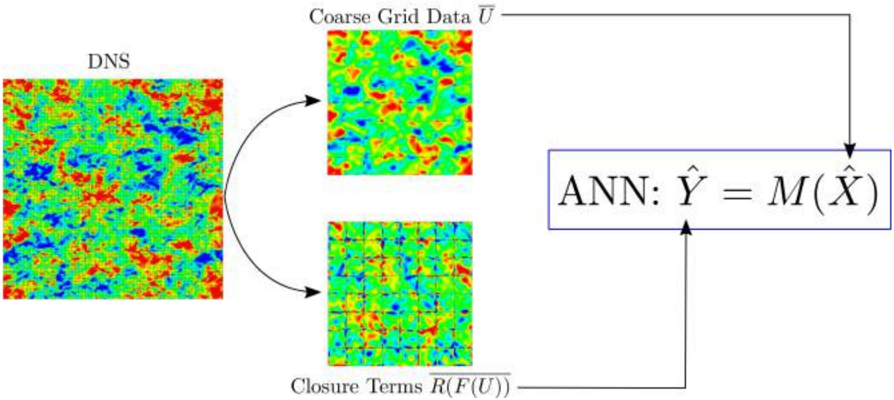
- Turbulent fluid motion is prevalent in naturally occurring flows and engineering applications: multiscale problem in space and time
- Navier-Stokes equations: system of non-linear PDEs (hyp. / parab.)
- Fullscale resolution (DNS) rarely feasible: Coarse scale formulation of NSE is necessary
- Filtering the NSE: Evolution equations for the coarse scale quantities, but with a closure term / regularization dependent on the filtered full scale solution ⇒ Model depending on the coarse scale data needed!
- Two filter concepts: Averaging in time (RANS) or low-pass filter in space (LES)
- An important consequence: RANS can be discretization independent, LES is (typically) not!
- 50 years of research: Still no universal closure model

Idea

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

Idea

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

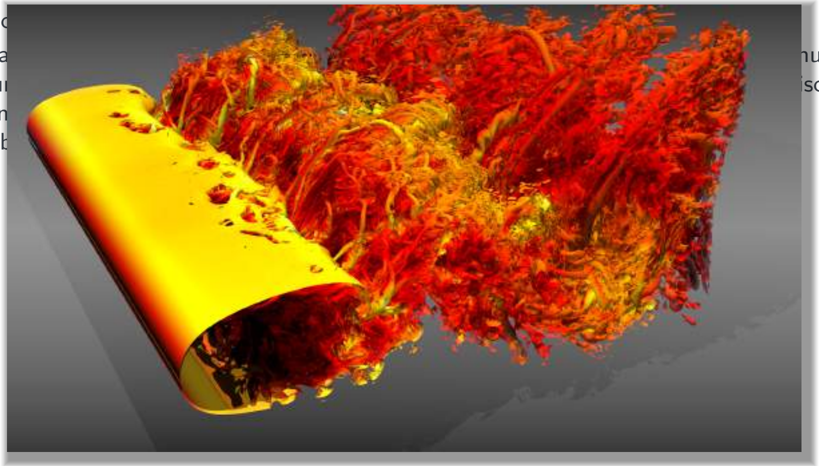


Problem Definition

- Choice of LES formulations:
  - Scale separation filter: implicit ⇔ explicit, linear ⇔ non-linear, discrete ⇔ continuous...
  - Numerical operator: negligible ⇔ part of the LES formulation, isotropic ⇔ non-isotropic, commutation with filter...
  - Subgrid closure: implicit ⇔ explicit, deconvolution ⇔ stochastic modelling,...

Problem Definition

- Choice c
  - Sca   nuous...
  - Nu   isotropic, con
  - Sub



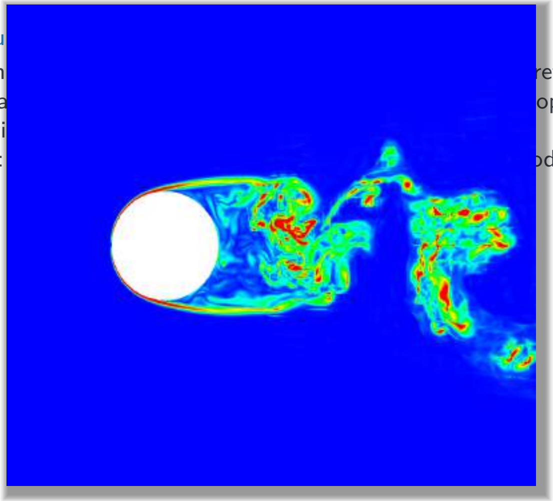
Problem Definition

- Choice of LES formu
  - Scale separation
  - Numerical opera
    commutation wi
  - Subgrid closure:

rete ⇔ continuous...

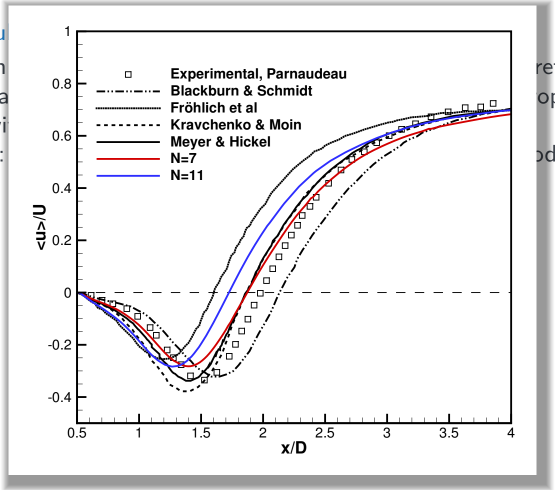
opic ⇔ non-isotropic,

odelling,...



Problem Definition

- Choice of LES formu
  - Scale separation … rete ⇔ continuous...
  - Numerical opera … opic ⇔ non-isotropic, commutation wi
  - Subgrid closure: … odelling,...



Problem Definition

- Choice of LES formulations:
  - Scale separation filter: implicit ⇔ explicit, linear ⇔ non-linear, discrete ⇔ continuous…
  - Numerical operator: negligible ⇔ part of the LES formulation, isotropic ⇔ non-isotropic, commutation with filter…
  - Subgrid closure: implicit ⇔ explicit, deconvolution ⇔ stochastic modelling,…

- Essential for ML methods: Well-defined training data (both input and output)
- Is $\overline{U}$ known explicitly? ⇒ For practical LES, i.e. grid-dependent LES, it is not most of the time!

Problem Definition

- Choice of LES formulations:
  - Scale separation filter: implicit ⇔ explicit, linear ⇔ non-linear, discrete ⇔ continuous...
  - Numerical operator: negligible ⇔ part of the LES formulation, isotropic ⇔ non-isotropic, commutation with filter...
  - Subgrid closure: implicit ⇔ explicit, deconvolution ⇔ stochastic modelling,...

- Essential for ML methods: Well-defined training data (both input and output)
- Is $\overline{U}$ known explicitly? ⇒ For practical LES, i.e. grid-dependent LES, it is not most of the time!

Definition: Perfect LES

- All terms must be computed on the coarse grid
- Given $\overline{U}(t_0, x) = \overline{U^{DNS}}(t_0, x)\ \forall x$, then $\overline{U}(t, x) = \overline{U^{DNS}}(t, x)\ \forall x$ and $\forall t > 0$

Turbulence Closure

- Filter

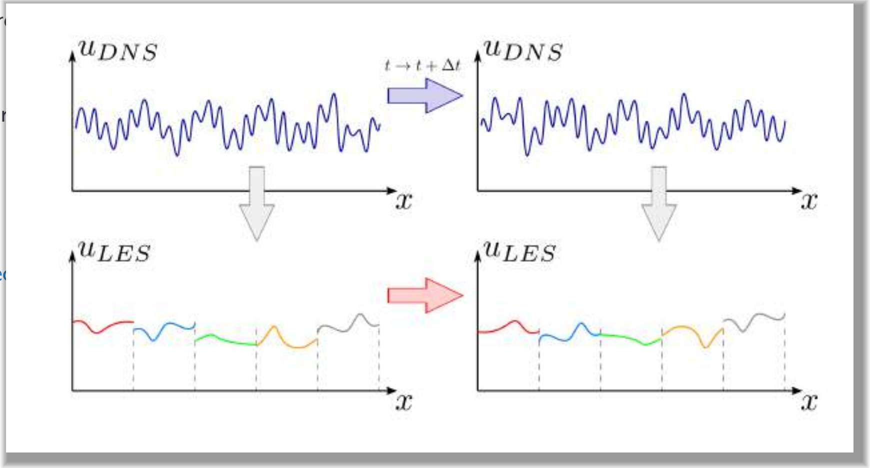
(2)

- Imper

(3)

- Perfe

(4)



Turbulence Closure

- 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 $\overline{U}$

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

- Note $\widetilde{R}(F(\overline{U}))$ is necessarily a part of the closure, but it is known
- Perfect LES and perfect closure are not new concepts: introduced by R. Moser et al in a series of papers*, termed ideal / optimal LES

---

*Langford, Jacob A. & Robert D. Moser. "Optimal LES formulations for isotropic turbulence." JFM 398 (1999): 321-346.

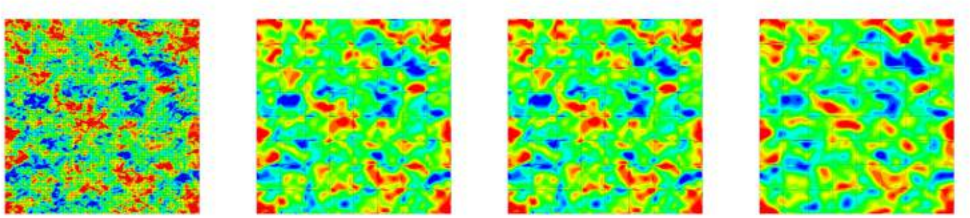
Perfect LES

$$\frac{\partial \overline{U}}{\partial t} + \overbrace{\widetilde{R}(F(\overline{U}))}^{\text{coarse grid operator}} = \underbrace{\overbrace{\widetilde{R}(F(\overline{U}))}^{\text{coarse grid operator}} - \overline{R(F(U))}}_{\textbf{perfect closure model}}.$$

- The specific operator and filter choices are not relevant for the perfect LES
- Note that the coarse grid operator is part of the closure (and cancels with the LHS)
- We choose:
  - DNS-to-LES operator $\overline{()}$: L_2 projection from DNS grid onto LES grid: We choose a discrete scale-separation filter
  - LES operator $\widetilde{()}$: 6th order DG method with split flux formulation and low dissipation Roe flux

Perfect LES

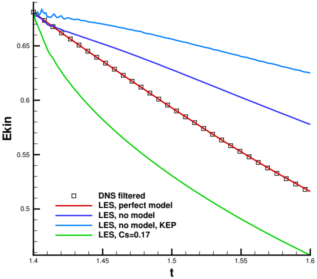
- Perfect LES runs with closure term from DNS
- Decaying homogeneous isotropic turbulence
- DNS grid: 64^3 elements, N = 7 ; LES grid: 8^3 elements, N = 5 ;

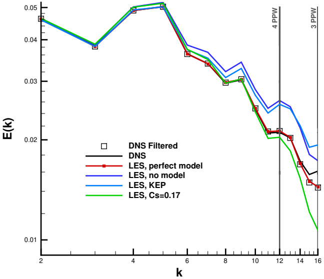


Left to right: a) DNS, b) filtered DNS, c) computed perfect LES d) LES with Smagorinsky model $C_s = 0.17$

Perfect LES

- Perfect LES runs with closure term from DNS
- Decaying homogeneous isotropic turbulence
- DNS grid: 64^3 elements, N $= 7$; LES grid: 8^3 elements, N $= 5$;





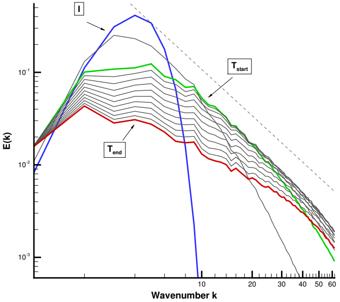
- ⇒ Perfect LES gives well-defined target and input data for supervised with NN

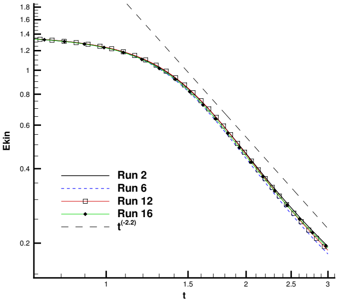
Training and Results

4

Data Acquisition: Decaying Homogeneous Isotropic Turbulence

- Ensemble of DNS runs of decaying homogeneous isotropic turbulence with initial spectrum defined by Chasnov (1995) initialized by Rogallo (1981) procedure and $Re_\lambda = 180$ at start
- Data collection in the range of exponential energy decay: 25 DHIT realizations with 134 Mio DOF each computed on CRAY XC40 (approx. 400,000 CPUh, 8200 cores)
- Compute coarse grid terms from DNS-to-LES operator

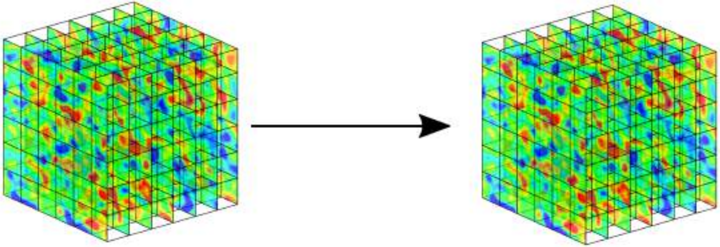




Features and Labels

- 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 $\overline{R(F(U))}$

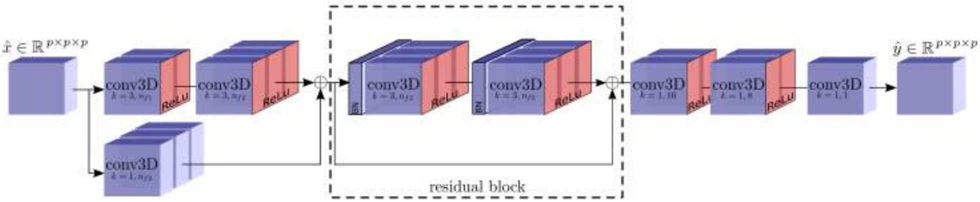
$$\hat{X} = \left\{ \hat{x} \in \mathbb{R}^{6 \times p \times p \times p} \mid \hat{x} = (\overline{u}_{ijk}, \overline{v}_{ijk}, \overline{w}_{ijk}, \widetilde{R}(F(\overline{U^1}))_{ijk}, \widetilde{R}((F(\overline{U^2}))_{ijk}, \widetilde{R}(F(\overline{U^3}))_{ijk}), \quad \text{with } i,j,k = 0,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



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

Networks and Training

- CNNs with skip connections (RNN), batch normalization, ADAM optimizer, data augmentation
- Different network depths (no. of residual blocks)
- For comparison: MLP with 100 neurons in 1 hidden layer*
- Implementation in Python / Tensorflow, Training on K40c and P100 at HLRS
- Split in training, semi-blind validation and blind test DHIT runs

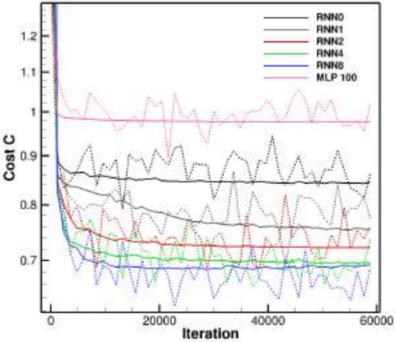


---

*Gamahara & Hattori. "Searching for turbulence models by artificial neural network." Physical Review Fluids 2.5 (2017)

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

- High correlation achievable with deep networks
- For surfaces: one-sidedness of data / filter kernels

Training Results III: Feature Sensitivity

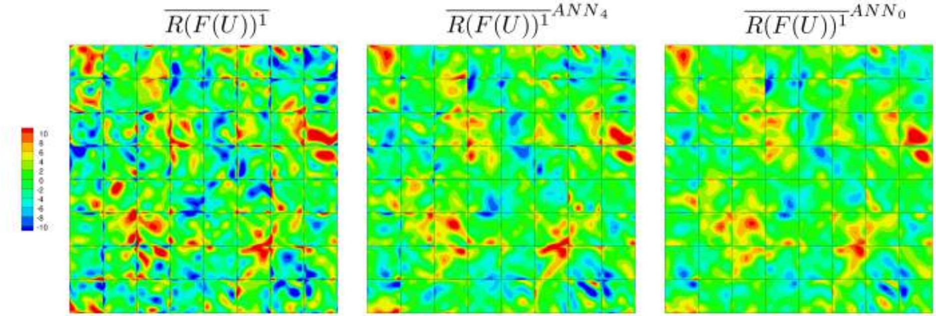
Set	Features	$\mathcal{CC}^1$	$\mathcal{CC}^2$	$\mathcal{CC}^3$
1	$u_i,\ \widetilde{R}(F(\overline{U^i})),\ i=1,2,3$	0.4706	0.4505	0.4499
2	$u_i,\ i=1,2,3$	0.3665	0.3825	0.3840
3	$\widetilde{R}(F(\overline{U^i})),\ i=1,2,3$	0.3358	0.3066	0.3031
4	$\rho, p, e, u_i,\ \widetilde{R}(F(\overline{U^i})),\ i=1,2,3$	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 3D cell data than pointwise data

Training Results IV: Visualization

- "Blind" application of the trained network to unknown test data
- Cut-off filter: no filter inversion / approximate deconvolution



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

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

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

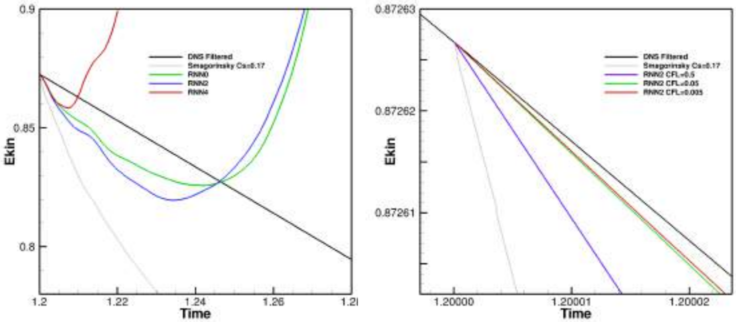
$\mathcal{CC} \approx 0.47$

$\mathcal{CC} \approx 0.34$

LES with NN-trained model I

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

- Perfect LES is possible, but the NN-learned mappings are approximate
- No long term stability, but short term stability and dissipation

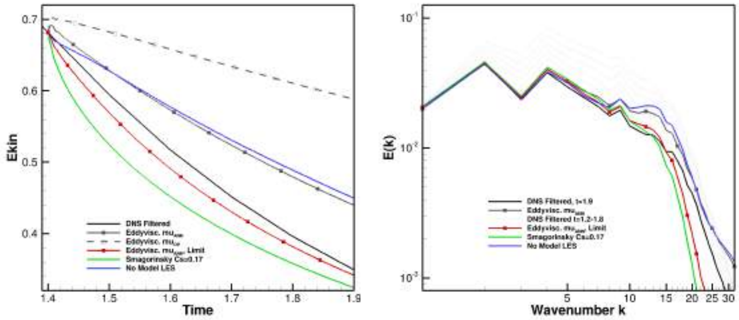


LES with NN-trained model II

$$\frac{\partial \overline{U}}{\partial t} + \widetilde{R}(F(\overline{U})) = \underbrace{\widetilde{R}(F(\overline{U})) - \overline{R(F(U))}}_{\textbf{data-based eddy viscosity model}} .$$

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

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



Marius Kurz: Sequence Learning

5

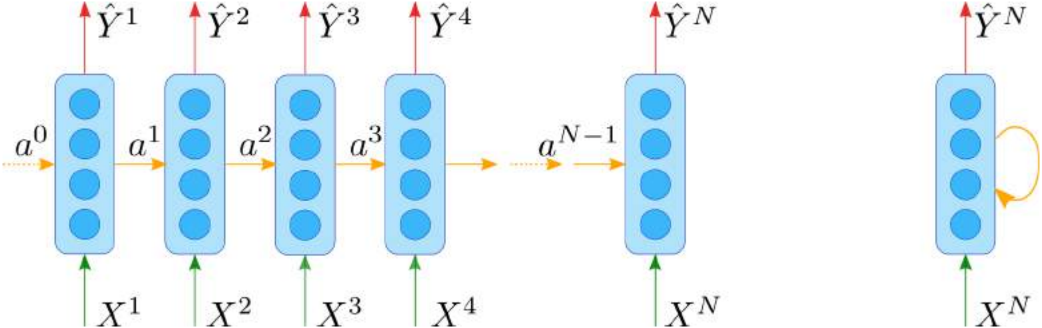
Can we do better?

- So far, we have not taken the temporal evolution of turbulence and the closure terms into account
- NN architectures that make use of ordered, consecutive information are called sequence models or recurrent NNs: Models dynamic temporal behaviours
- Examples of sequence data: Sensor data, spoken language, translation, stock prizes, ...
- There are many different architectures and flavours of RecNN, so let us just discuss the basic ideas!
- The general form (of a uni-directional RecNN): an autoregressive non-linear model

$$\hat{Y}^{t+1} = f(\underbrace{X^{t+1}}_{\text{input}}, \underbrace{m(\hat{Y}^{t}, \hat{Y}^{t-1}, ...))}_{\text{"memory"}} \tag{6}$$

Recurrent NNs

- Architecture:

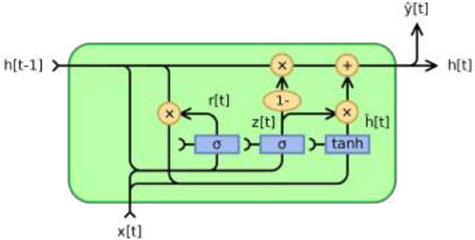


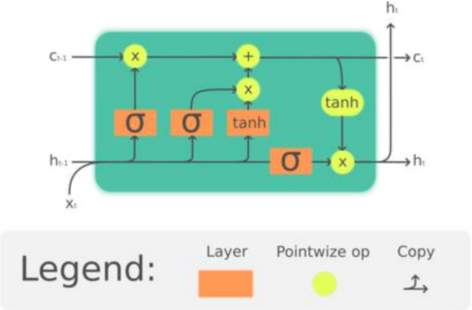
- Forward pass:

$$\begin{aligned} a^t &= \sigma(W_{aa}\, a^{t-1} + W_{ax} X^t + b_a) \\ \hat{Y}^t &= \sigma(W_{ya}, a^t + b_y) \end{aligned} \tag{7}$$

Recurrent NNs

- RecNN-Architectures differ in the way the hidden layers are structured
- Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM)





By Jeblad - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=66225938 and Guillaume Chevalier, https://upload.wikimedia.org/wikipedia/commons/3/3b/The_LSTM_cell.png

Recurrent NNs

- GRU and LSTM: learning long range connections through memory lanes
- Differ in terms of gates: How and when the memory lane is written, updated or forgotten:
  - Update gate (GRU, LSTM): How much of the past should matter now?
  - Relevance gate (GRU, LSTM): Drop previous information?
  - Forget gate (LSTM): Erase memory?
  - Output gate (LSTM): How much to reveal of a cell?
- Many more technical details, here are some suggestions:
  - https://stanford.edu/ shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
  - Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short term memory." Neural computation 9.8 (1997): 1735-1780.
  - Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).
  - Greff, Klaus, et al. "LSTM: A search space odyssey." IEEE transactions on neural networks and learning systems 28.10 (2016): 2222-2232.

Stability of Recurrent NNs

- Recurrency introduces possible source of trouble: predicting long term sequential input can lead to exponential error growth.
- Simplified: $\hat{Y}^T = A(\hat{Y}^{T-1}, X^T)$, of course $\hat{Y}^{T-1} = A(\hat{Y}^{T-2}, X^{T-1})$, ...: A^D stability w.r.t. to small errors?
- Long term stability is currently a problem, some fixes are:
  - ”Scheduled Sampling” by Bengio et al.
  - ”Auto-conditioned recurrent networks” by Zhou et al.
  - ”Stability Training” by Goodfellow et at.

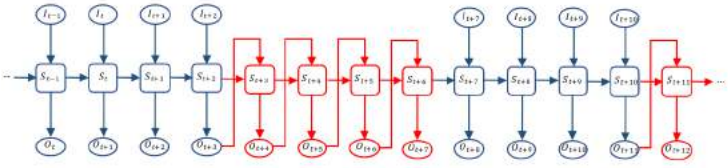
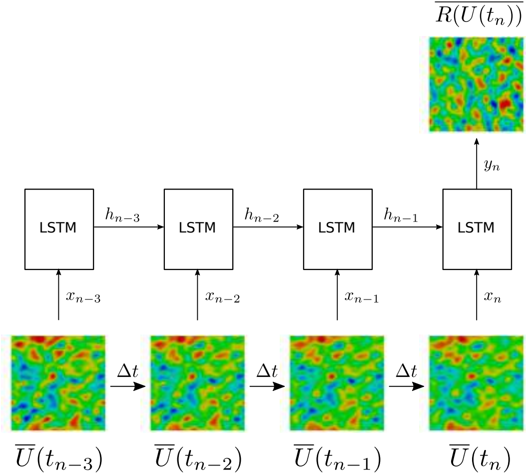


Figure 1: Visual diagram of an unrolled Auto-Conditioned RNN (right) with condition length $v = 4$ and ground truth length $u = 4$. I_t is the input at time step t. S_t is the hidden state. O_t is the output.

from: Li, Z., Zhou, Y., Xiao, S., He, C., Huang, Z., & Li, H. (2017). Auto-conditioned recurrent networks for extended complex human motion synthesis. arXiv preprint arXiv:1707.05363.

Back to LES Closure Preditions

- Predict closure terms from time series data
- Prediction mode: many-to-one



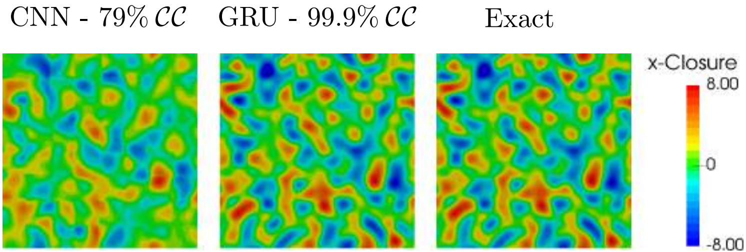
Performance of Network Architectures

- RNNs outperform MLP and CNN architectures by a lot!
- LSTMs and GRUs give similar results

Network	# Parameter	Time (GPU)	Time (CPU)	L_2-Error	$\mathcal{CC}$
MLP	$6,720$	6 ms	28 ms	$3.0 \cdot 10^{+1}$	66.0%
CNN	$187,088$	72 ms	198 ms	$2.1 \cdot 10^{+1}$	78.7%
LSTM $(3\Delta t)$	$39,744$	62 ms	340 ms	$1.3 \cdot 10^{-1}$	99.9%
GRU $(3\Delta t)$	$31,578$	59 ms	319 ms	$1.1 \cdot 10^{-1}$	99.9%

Performance of Network Architectures

- RNNs outperform MLP and CNN architectures by a lot!
- LSTMs and GRUs give similar results



CNN - 79% $\mathcal{CC}$ GRU - 99.9% $\mathcal{CC}$ Exact

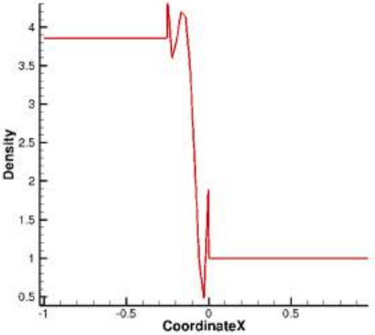
Summary

- Perfect / optimal LES framework: well-defined target quantities for learning
- 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
- Sequence models show superior performance
- Achievable $\mathcal{CC} \approx 99\%$, with up to $\approx 79\%$ for CNN
- Currently 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
- More Info: Beck, Flad, Munz. "Deep neural networks for data-driven LES closure models." Journal of Computational Physics 398 (2019): 108910.

Anna Schwarz: Detecting Shocks

Shock Localization through Holistic Edge Detection

- Another quick example of combining CFD + ML
- Shocks and sharp discontinuities cause Gibb's oscillations in high order methods due to non-smoothness
- These features need to be treated with special numerical methods to ensure stability



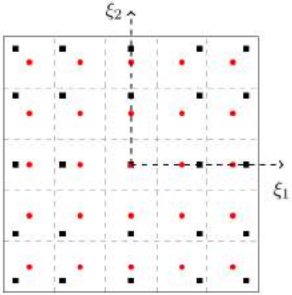
Shock capturing

- A classical approach:
  1. Choose some numerical method for the stable approximation of discontinuities (e.g. FV subcells, p-reduction, artificial viscosity)
  2. Define a "troubled cell" indicator with empirical parameters
  3. Apply the method from (1) in the troubled cells
  4. Find "good" parameters for (2), where good means both stable and as sharp as possible
  5. Rinse and repeat for different physics, numerics, etc.
- Note that the indicator and the numerics are closely linked
- An indicator that leads to a stable simulation for one case (e.g. for one Riemann solver, N, Mach number) will fail for another case
- The troubled cell indicator is an empirically tuned "tolerance level" fitted to the numerical scheme: How strong can the discontinuity be for the scheme to survive?

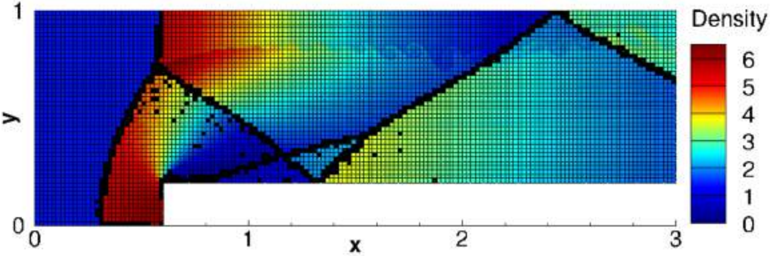
⇒ Shock capturing and shock detection are interdependent
⇒ Experience / Parameter Tuning required

A DG method for shock capturing

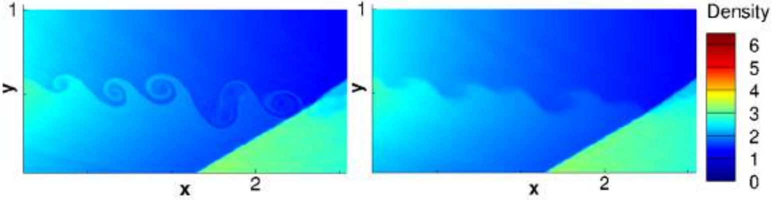
- Hybrid DG / Finite Volume operator
- Interpret solution polynomial differently
- Introduce virtual FV grid within each DG element
- Solve a TVD Finite volume method in troubled cells
- Keep high order accuracy wherever possible
- Switch DG2FV and vice versa ⇒ Experience / Parameter tuning required



A DG method for shock capturing

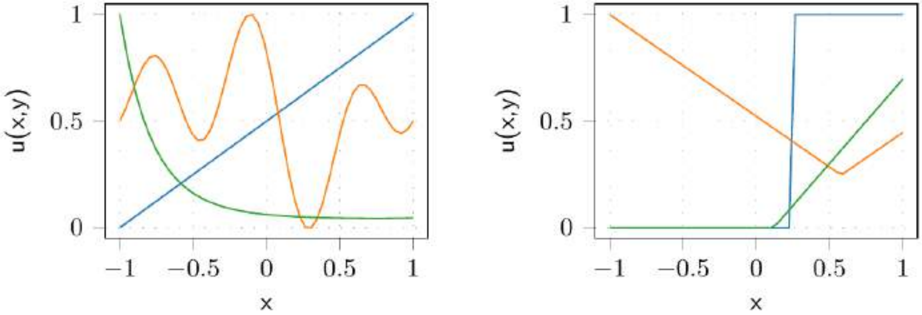


Coupled DG/FV subcell vs. pure FV:



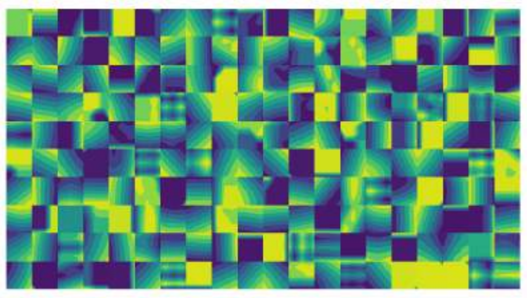
Shock Detection through ML

- General idea: Decouple the shock localization and the shock capturing to ameliorate parameter tuning
- First task: Train a CNN-based binary classifier on element data to detect shocks without regarding their numerical representation
- Second task: Localize the shock within an element
- Training data: Smooth and non-smooth functions

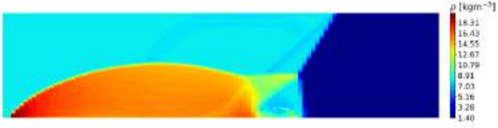


Shock Detection through ML

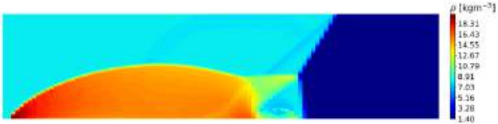
- General idea: Decouple the shock localization and the shock capturing to ameliorate parameter tuning
- First task: Train a CNN-based binary classifier on element data to detect shocks without regarding their numerical representation
- Second task: Localize the shock within an element
- Training data: Smooth and non-smooth functions



Shock Detection through ML



Shock Detection through ML



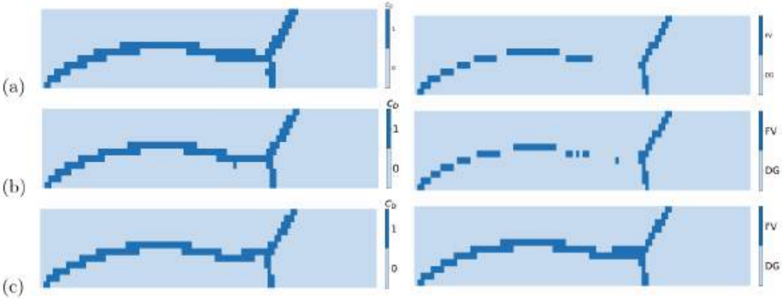
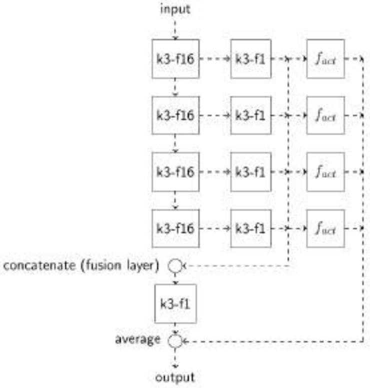
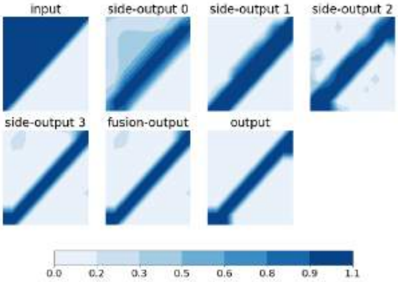


Figure 4.10.: Classification results of models C_{N4}, C_{N5}, and C_{N9} (left) and the Jameson indicator (right) for the DMR on a mesh with 1224 elements at $t_{end} = 0.2$. (a) $N = 4$, (b) $N = 5$, (c) $N = 9$.

Shock Detection through ML

- Shocks can be safely detected by the NN indicator, without additional parameter tuning
- Consistent detection, not dependent on numerical scheme: not a troubled cell indicator!
- Task 2: Localize the shock within an element: Holistic Edge Detection

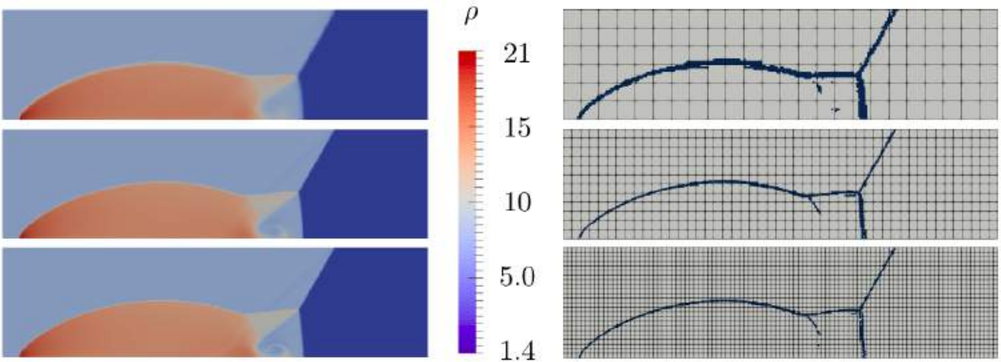




(d) Model C_8 ($N = 9$).

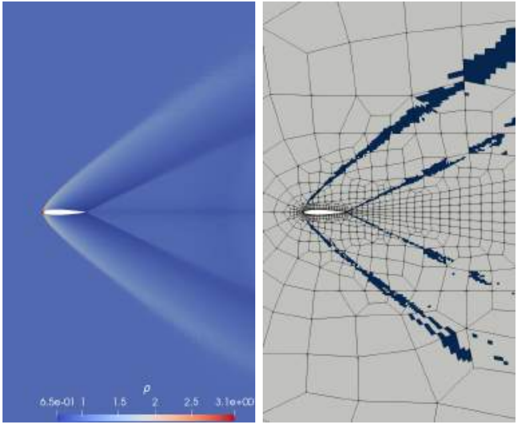
Shock Localization through ML

- Task 2: Localize the shock within an element: This is especially beneficial for high order schemes!



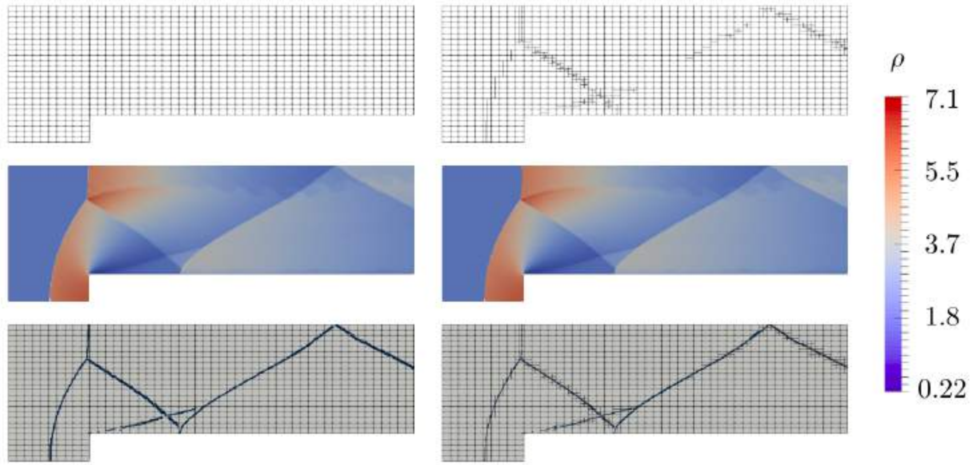
Shock Localization through ML

- Works also on real meshes!



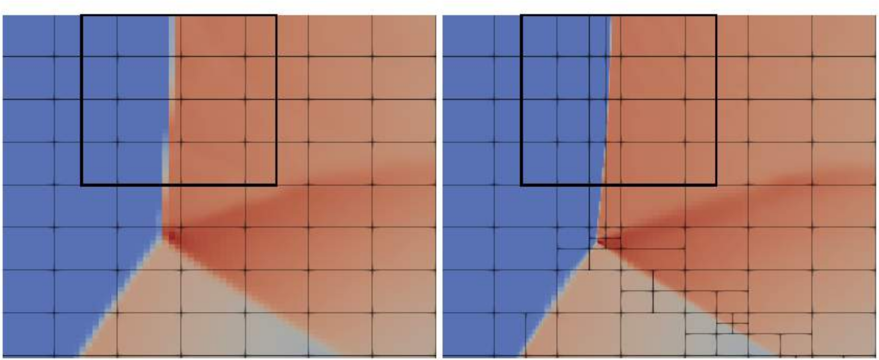
NN-guided mesh adaptation

- Evaluate indicator on baseline grid (left), then refine accordingly (right)



NN-guided mesh adaptation

- Evaluate indicator on baseline grid (left), then refine accordingly (right)



- Beck et al. ”A Neural Network based Shock Detection and Localization Approach for Discontinuous Galerkin Methods.” arXiv preprint arXiv:2001.08201 (2020).

Summary

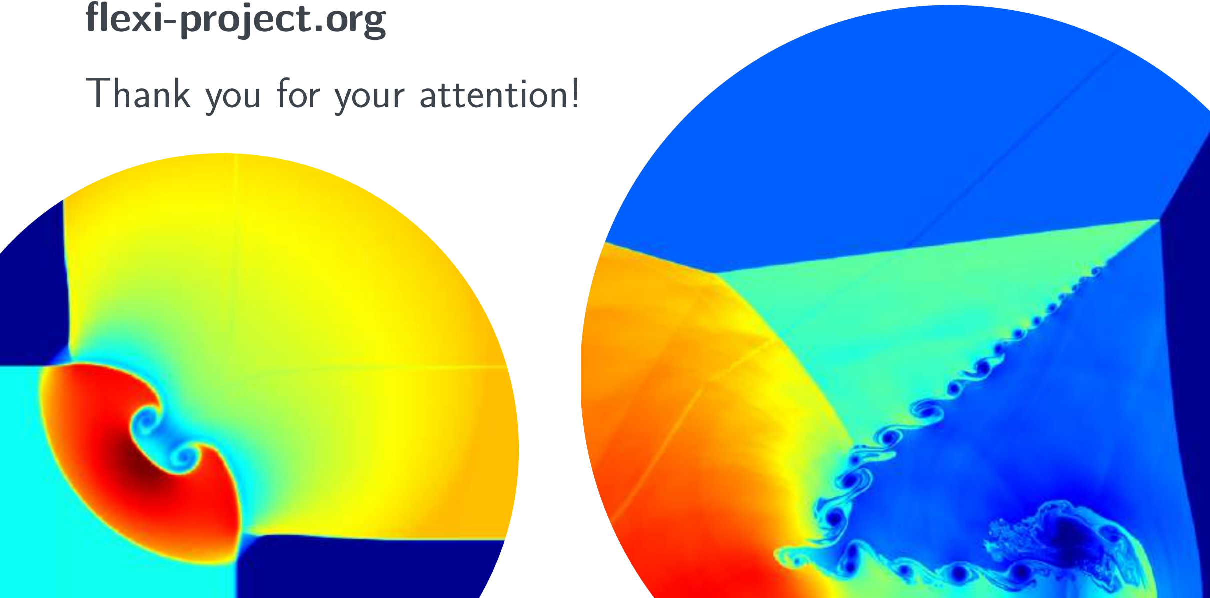
7

Some final thoughts on data-informed models, engineering and HPC

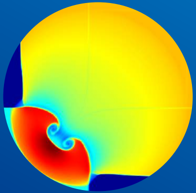
- Machine Learning is not a silver bullet
- First successes: ML can help build subscale models from data, or improve replace parameter-dependent empirical models
- A lot of representative data is needed... maybe we already have the data? Computations, experiments...
- In this work, the computational 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
- "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)
- 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?
- Data storage policy: where to compute/store the data (reproducibility)

flexi-project.org

Thank you for your attention!



University of Stuttgart
Institute of Aerodynamics
and Gas Dynamics



Andrea Beck

eMail	beck@iag.uni-stuttgart.de
Telefon	+49-711-685 60218
Web	nrg.iag.uni-stuttgart.de