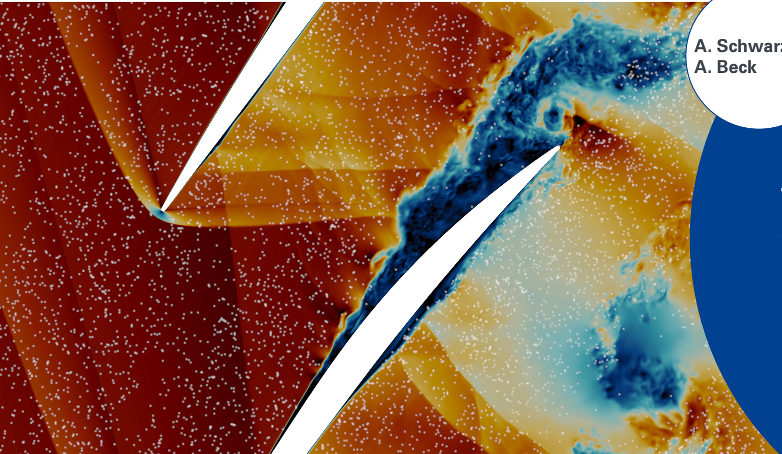
University of Stuttgart
Germany

CEEC

NRG



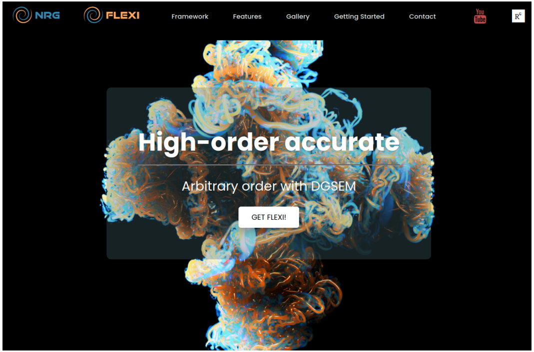
A. Schwarz
A. Beck

Towards Data-Driven Computational Fluid Dynamics

What are we doing…?

Numerics Research Group

Prof. Dr. Andrea Beck



https://numericsresearchgroup.org

What are we doing…?

Numerics Research Group

Prof. Dr. Andrea Beck

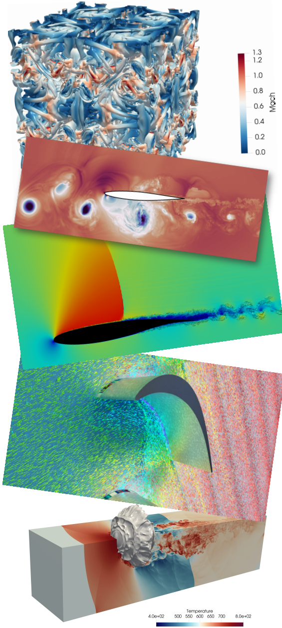
CFD solver FLEXI*:

- OpenSource HPC solver for unsteady compressible Navier–Stokes eq.
- High order discontinuous Galerkin (DG) spectral element method

Applications and features:

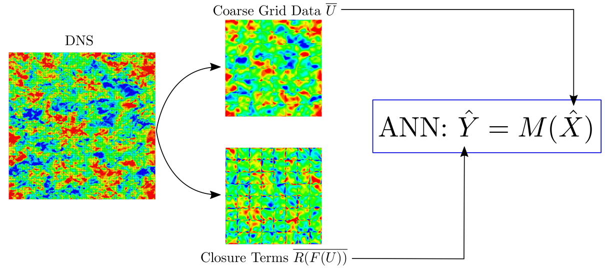
- LES and DNS of multiscale, multiphysics and multiphase flows
- Complex geometries
- Explicit/implicit global time-stepping
- Shock capturing based on FV subcells
- Sharp/diffuse interface methods
- 4-Way Euler–Lagrange particle tracking
- *Relexi*: RL framework for HPC‡
- hp-refinement, …

‡ https://github.com/flexi-framework/relexi



* N. Krais et al. In: *Computers & Mathematics with Applications* 81 (2021)

Machine learning enhanced solution of PDEs

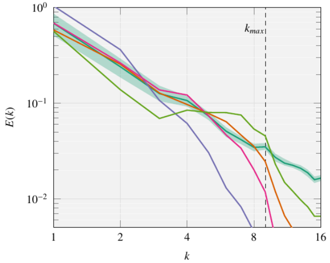


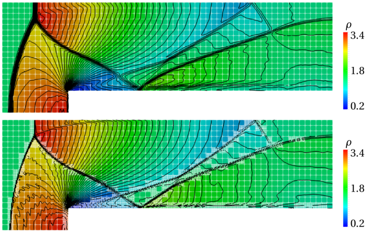
Problem definition

- PDEs are generally non-linear and can fulfil certain constraints: conservation, stability, invariances, symmetries, ...
- PDE solvers can guarantee these, but what about ML models?
  - ML models must converge,
  - have to at least weakly guarantee the physical and mathematical constraints of the underlying PDE
  - and must come with interpretability, error bounds and regions of trustworthiness
- ML methods will not replace PDE solvers
- However, they are useful for
  - abstracting empirical knowledge and improving physical understanding
  - accelerating the solution of PDEs
  - developing enhanced models

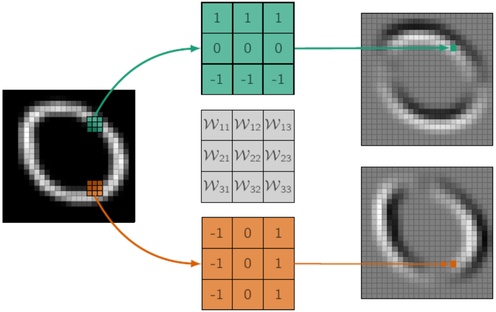
Applications in CFD

- Enhancing closure terms for multiscale problems, e.g., turbulence closure, diffusion processes, ...
- Improving numerical tools, e.g., Riemann solver, iterative solvers, shock capturing ...
- Accelerating solution of PDEs
- Developing enhanced models, e.g., optimal parameter estimation, reduced-models, ...
- Flow control, ...





Short introduction to ML



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+)

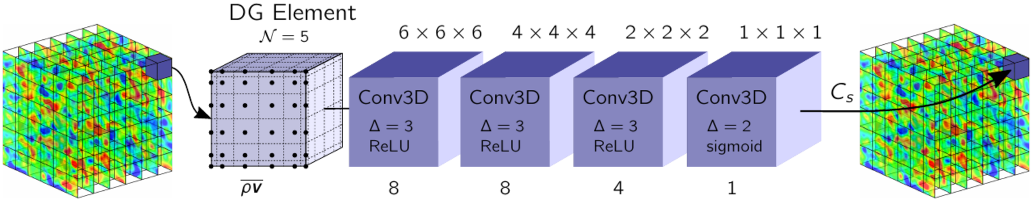
Definitions and concepts

Learning concepts:

- Unsupervised learning
- Supervised learning
- Reinforcement learning

Artificial neural networks:

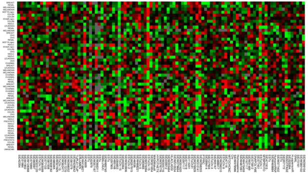
- General function approximators
- Graph neural networks, feed-forward / convolutional / recurrent neural networks, ...
- AlphaGo, Self-Driving Cars, Face recognition
- Incomplete theory, models are difficult to interpret
- NN design: more an art than a science

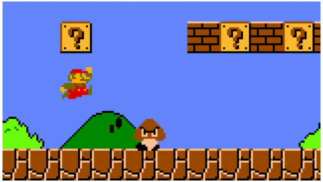


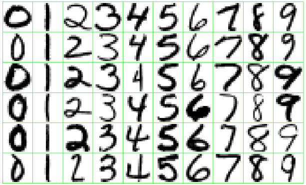
Types of ML

Different types of learning:

- Unsupervised learning:
  Discover a good internal representation of the input. $\Rightarrow$ “segmentation / clustering model”
- Reinforcement learning:
  Learn to select an action to maximize payoff. $\Rightarrow$ “behavioral model”
- Supervised learning:
  Learn to predict an output when given an input vector. $\Rightarrow$ “predictive model”





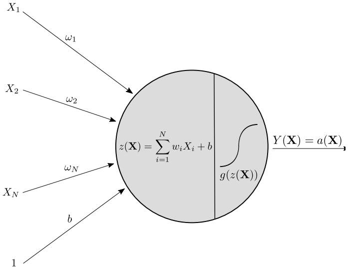


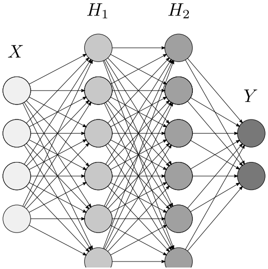
History of ANNs

- 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): An ANN with a single hidden layer and finite neurons can approximate continuous functions
  - Fukushima (1982): Neocognitron: convolutional networks
  - LeCun (1989,1995): “LeNet”, learning 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

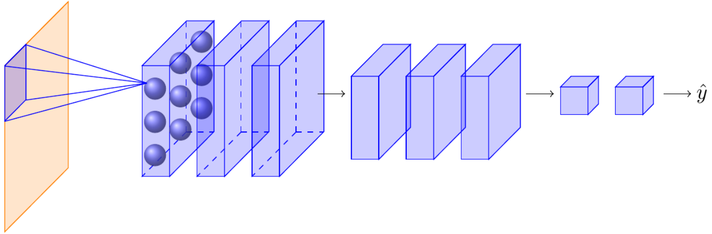
- Artificial Neural Network (ANN): A non-linear mapping from inputs to outputs $\mathbf{M} : \hat{X} \rightarrow \hat{Y}$
- An ANN is a nesting of linear and non-linear functions arranged in a directed acyclic graph:
$$\hat{Y} \approx Y = M(\hat{X}) = \sigma_L \left(W_L \left(\sigma_{L-1} \left(W_{L-1}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





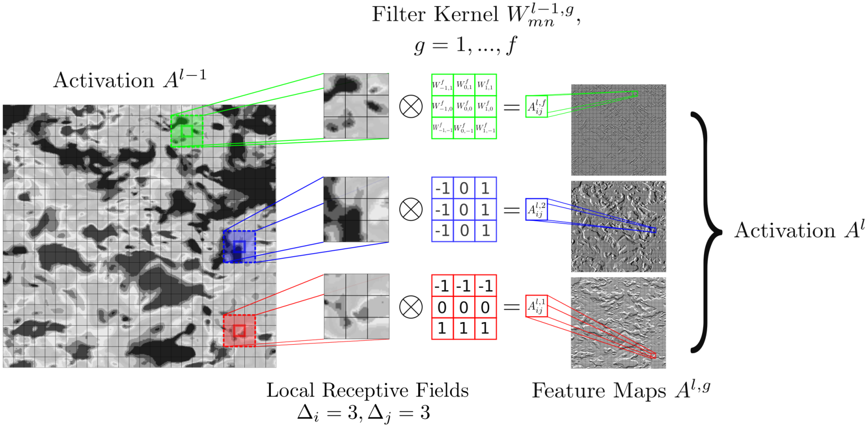
Advanced Architectures

- 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



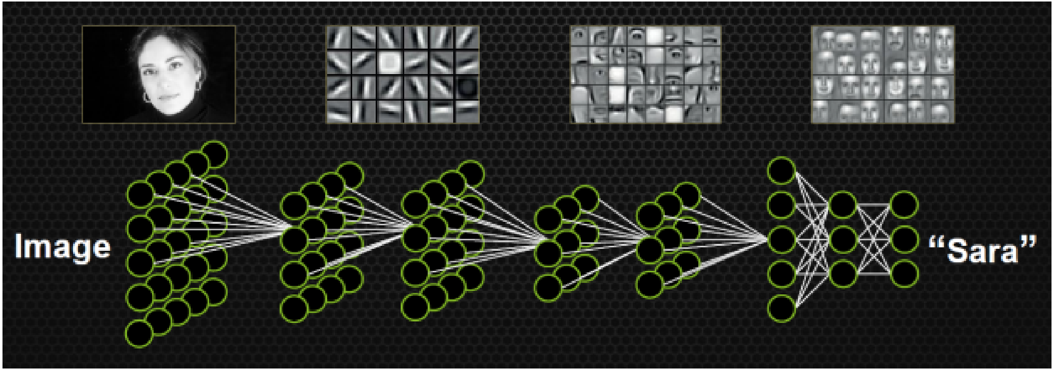
Convolutional Neural Networks

- Filter kernels, feature extraction



What does a CNN learn?

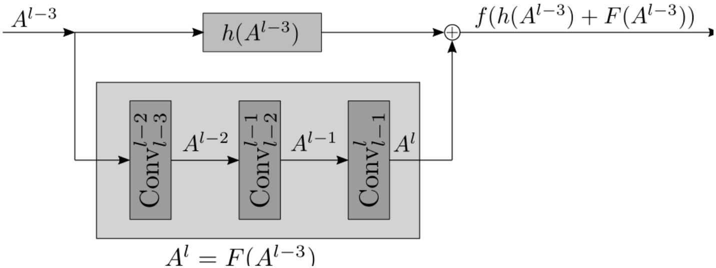
- Representation in hierarchical basis



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 (ResNN)

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

Advances and failures in the ML enhanced solution of PDEs

Examples for ML guided CFD

1. Data-driven shock capturing

High-order methods are superior ...

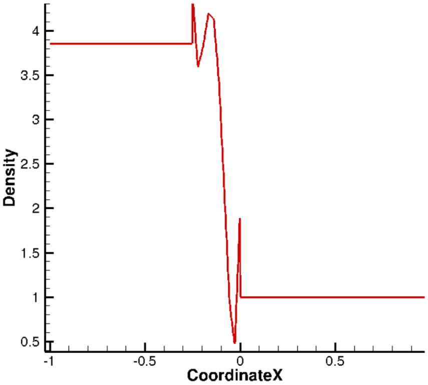
- in smooth regions of the solution
- since they enable an exponential convergence
- for multi-D / smooth multi-scale problems

High-order methods suffer ...

- from spurious oscillations at strong discontinuities (Gibbs' phenomenon)

Solution:

- Adding numerical/artificial viscosity to discontinuities to ensure stability
- Two-step approach: Detecting discontinuities and apply local viscosity



Examples for ML guided CFD

1. Data-driven shock capturing

High-order methods are superior ...

- in smooth regions of the solution
- since they enable an exponential convergence
- for multi-D / smooth multi-scale problems

High-order methods suffer ...

- from spurious oscillations at strong discontinuities (Gibbs' phenomenon)

Solution:

- Adding numerical/artificial viscosity to discontinuities to ensure stability
- Two-step approach: Detecting discontinuities and apply local viscosity

2. Data-driven turbulence closures

Turbulence is a ...

- a multiscale problem in space and time
- non-local, highly non-linear phenomena

Problem:

- No universal closure models
- Aliasing through under-resolved turbulence leads to stability issues
- DNS not feasible for high Re-number flows

Solution:

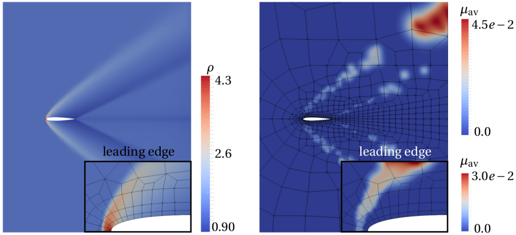
- LES, RANS, ...with "optimal" closure model

Supervised learning

Data-driven shock capturing

Joined work with:

Jonas Zeifang

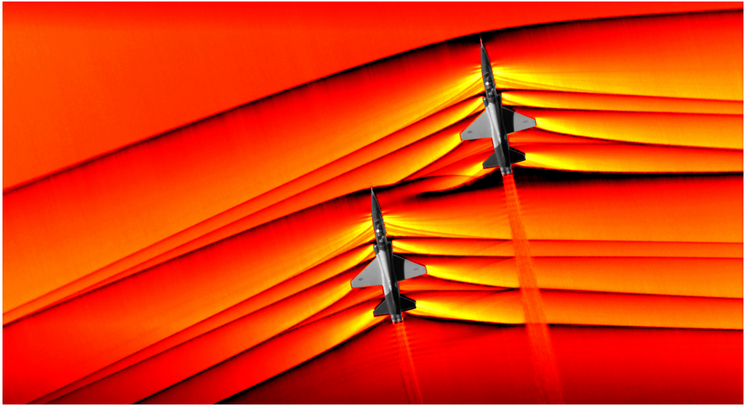


* J. Zeifang et al. In: *Journal of Computational Physics* 441 (2021)

Problem Statement I: Detection of Shock Waves

Shock waves in compressible flow:

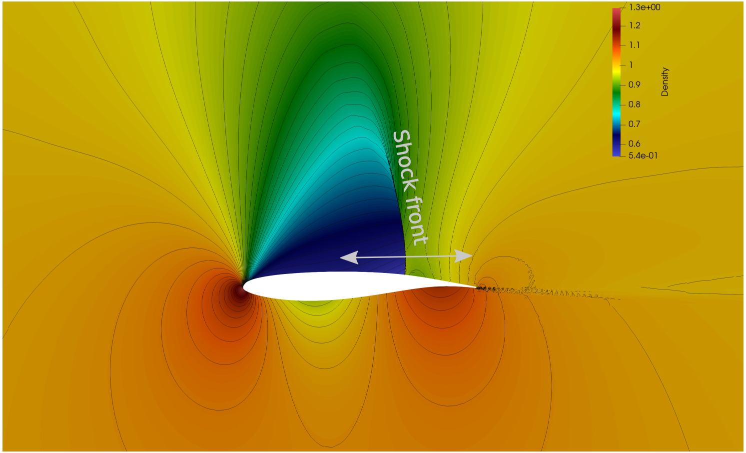
- Govern flow in transonic / supersonic / hypersonic regime



Problem Statement I: Detection of Shock Waves

Shock waves in compressible flow:

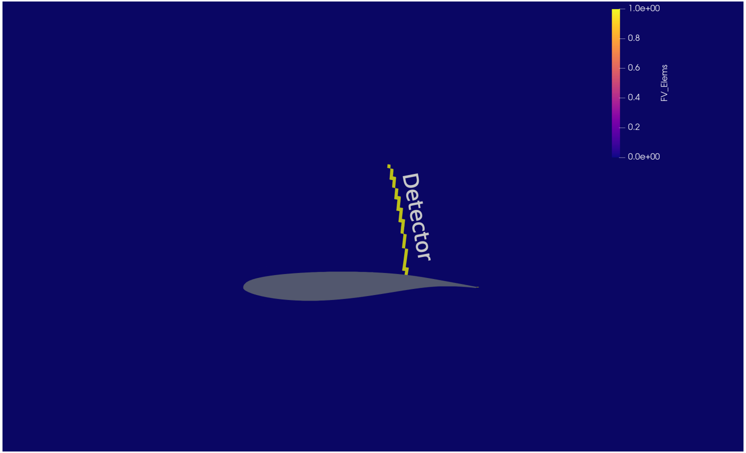
- Require special numerical treatment



Problem Statement I: Detection of Shock Waves

Shock waves in compressible flow:

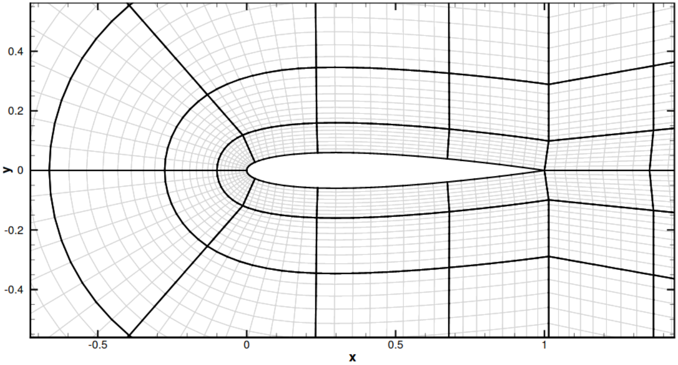
- Must be detected / tracked: empirical, parameter-dependent indicators



Problem Statement II: Localization of Shock Waves

Localizing Shock Waves

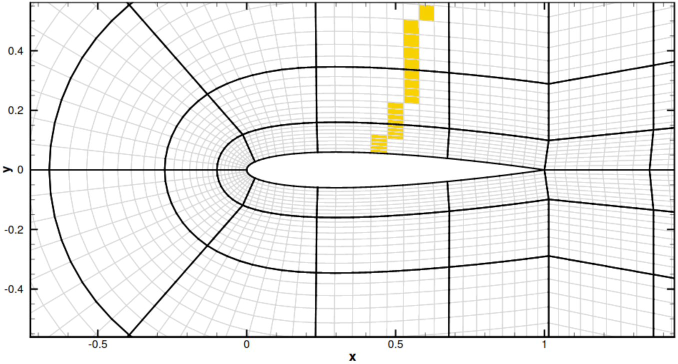
- Grids for low-order (gray) and high-order (black) schemes: large elements



Problem Statement II: Localization of Shock Waves

Localizing Shock Waves

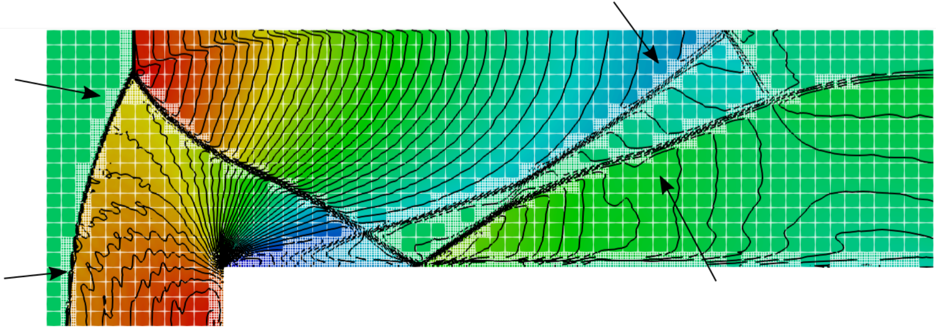
- Inner-element localization: add locally dissipation



Problem Statement III: Shock Capturing / Treatment

Shock capturing strategies for high-order (HO) schemes

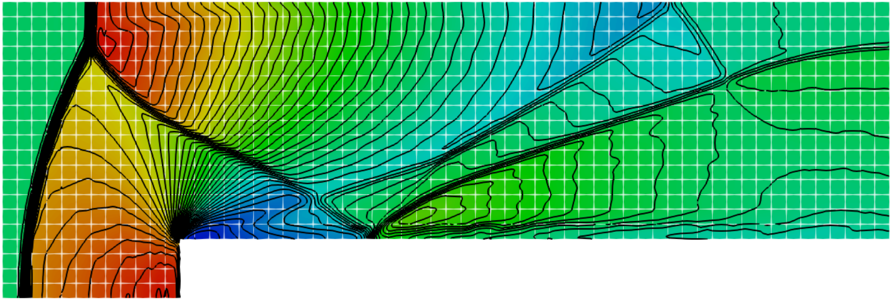
- Operator-based: h/p-schemes, Finite Volume (FV)-hybrid schemes, reconstruction with limiters,...



Problem Statement III: Shock Capturing / Treatment

Shock capturing strategies for high-order (HO) schemes

- Artificial viscosity-based: add numerical dissipation



Local shock locator for DG

1. Shock detection / localization:

- For high-order: detecting "troubled cells" is not enough
- Localizing local shock front within a DG element

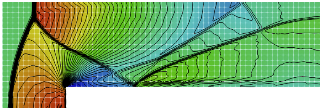
$\Rightarrow$ Shock capturing and detection are interdependent

2. Solution approaches:

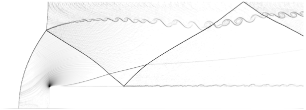
- Artificial viscosity
- Filtering / limiting
- TVD or TVB stable finite volume scheme
- Blending of a high- with a low-order scheme

A priori approach:

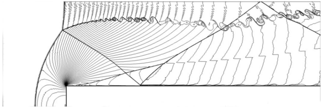
- Based on heuristic indicators
- Linked to numerical scheme, resolution & test cases
- Parameter tuning



Zeifang, Beck (2021)



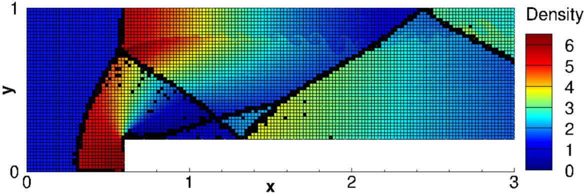
Sonntag, Munz (2017)



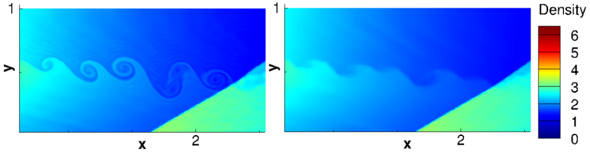
Dumbser, Zanotti, Loubère, Diot (2014)

Hybrid DG/FV operator for shock capturing

- 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
  $\Rightarrow$ Experience / parameter tuning required

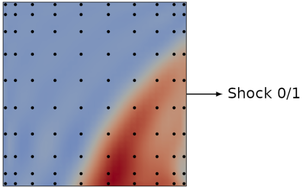


Coupled DG/FV subcell vs. pure FV:

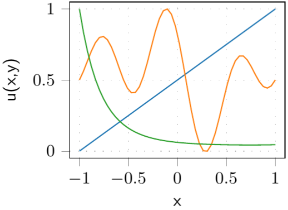


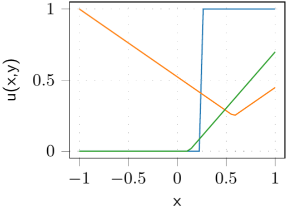
Shock detection/localization through ML*

- **Idea:** Decouple the shock localization and the shock capturing to ameliorate parameter tuning
- **1. Task:** Train a CNN-based binary classifier on element data to detect shocks without regarding their numerical representation
- Training data: Smooth and non-smooth functions



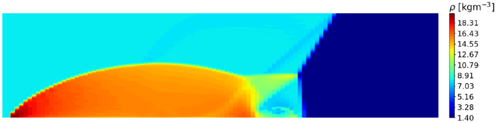
flow situation





* A. D. Beck et al. In: *Journal of Computational Physics* 423 (2020)

Shock detection through ML: Double mach reflection



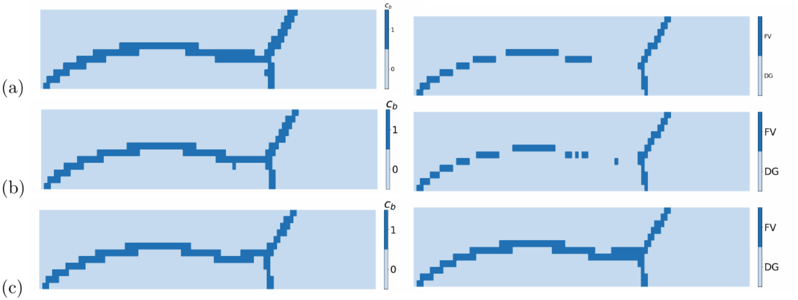
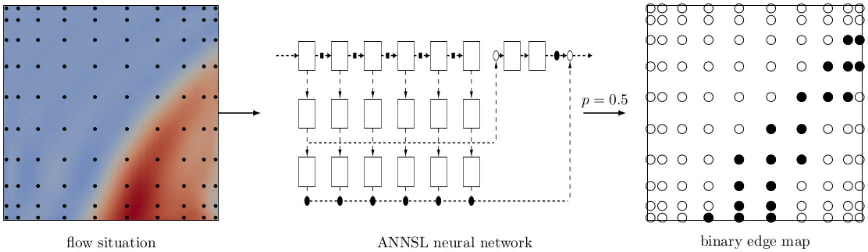


Figure 4.10.: Classification results of models C_{N4}, C_{N5}, and C_{N9} (left) and the Jameson indicator (right) for the DMR on a mesh with 1224 elements at $t_{end} = 0.2$. (a) $N = 4$, (b) $N = 5$, (c) $N = 9$.

Shock localization through ML†

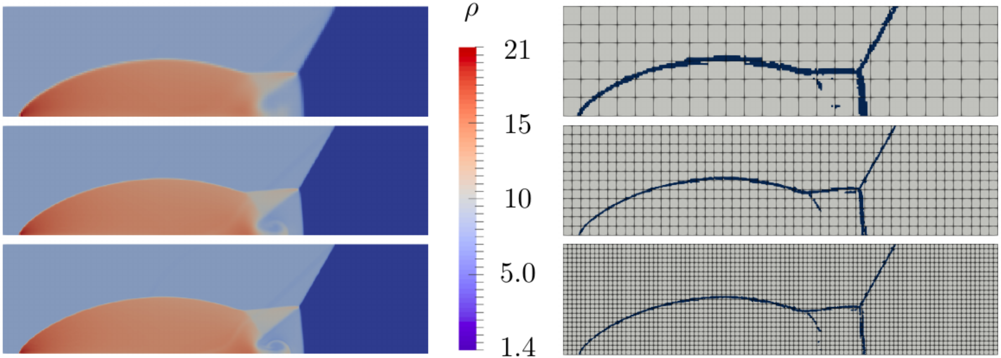
- Shocks can be safely detected by the CNN indicator, without additional parameter tuning
- Consistent detection, which is only weakly dependent on numerical scheme
- **2. Task:** Localize shock within an element: Holistic edge detection*



* Xie2015

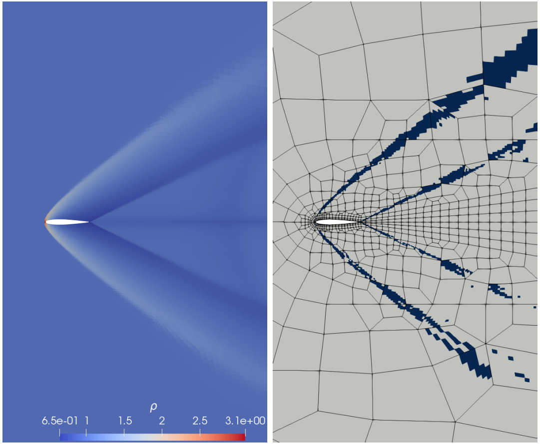
† A. D. Beck et al. In: *Journal of Computational Physics* 423 (2020)

Shock localization through ML: Double mach reflection



Shock localization through ML: Flow around a NACA0012

Works also on real meshes:

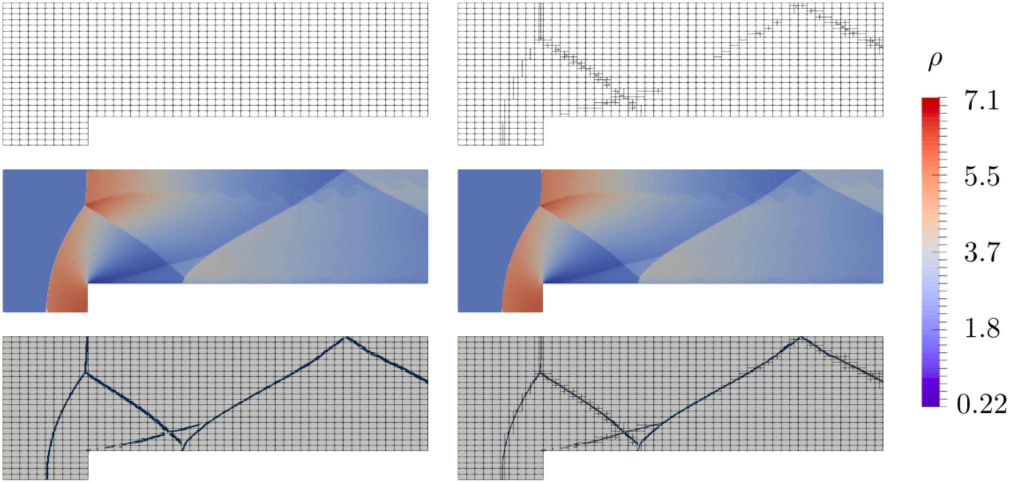


So far...

1. Detection of shock waves: ML $\Rightarrow$ CNN classifier
2. Localization of shock waves: ML $\Rightarrow$ Edge Detector
3. **Guiding mesh refinement: ML-informed (from 1. and 2.) mesh refinement**
4. **Guiding shock capturing: ML-informed (from 1. and 2.) HO artificial viscosity**

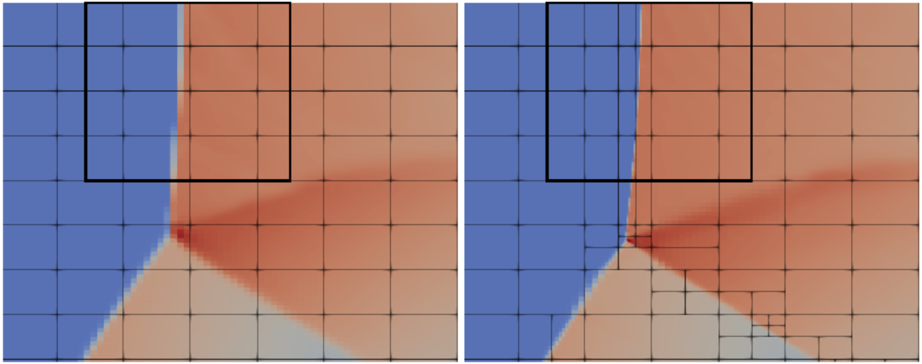
NN-guided mesh adaptation: Double mach reflection

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



NN-guided mesh adaptation: Double mach reflection

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



*A. D. Beck et al. In: *Journal of Computational Physics* 423 (2020)

Shock capturing based on artificial viscosity*

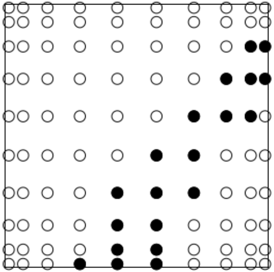
- Artificial viscosity approach: Euler equations with second order term

$$\partial_t \mathbf{w} + \nabla \cdot \mathbf{F}(\mathbf{w}) = \nabla \cdot \mu_a \nabla \mathbf{w}$$

- Shape, amplitude and location of μ_a are subject to user specification
- In DG and related methods: element-wise constant μ_a with linear C^0 continuous reconstruction, PDE- or filter based smoothing methods
- We seek: A highly localized, smooth distribution of μ_a
- Use binary edge map from ANN and smooth with radial basis function (RBF) interpolation

$$\mu_a(\mathbf{x}) = \mu_{a\,\text{scale}} \sum_{i=1}^{n_s} \alpha_i \phi_r \|\mathbf{x} - \mathbf{x}_{s_i}\|_2$$

- Support radius is defined in terms of the length of a grid element Δx

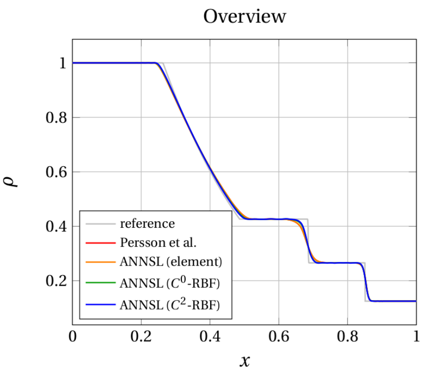


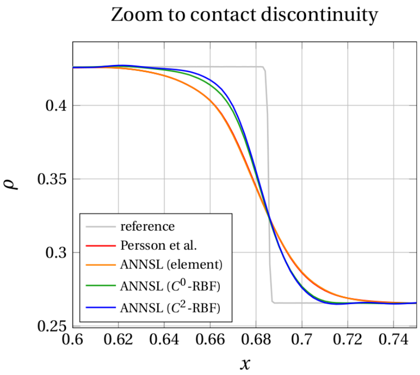
binary edge map

*J. Zeifang et al. In: *Journal of Computational Physics* 441 (2021)

High-order artificial viscosity: Sod's shock tube

- Comparing results* with elementwise-constant artificial viscosity† with linear reconstruction‡



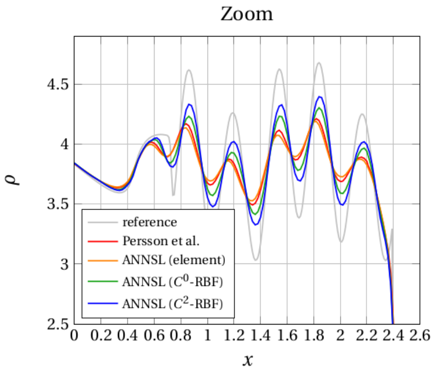


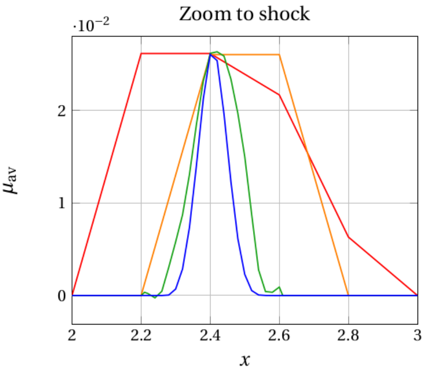
*J. Zeifang et al. In: *Journal of Computational Physics* 441 (2021)

†P.-O. Persson et al. In: *AIAA paper* 2 (2006)

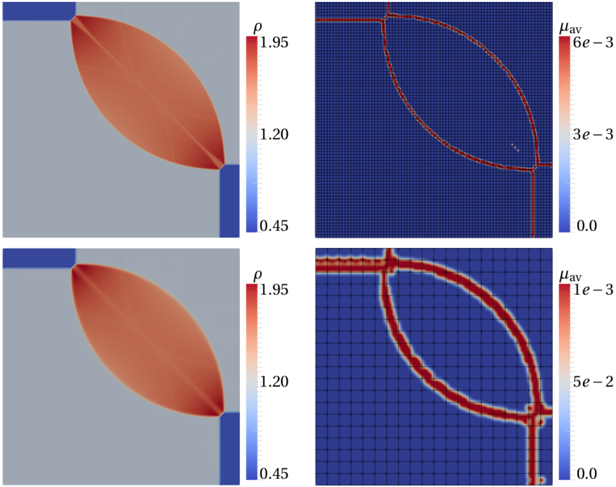
‡A. Klöckner et al. In: *Mathematical Modelling of Natural Phenomena* 6.3 (2011)

High-order artificial viscosity: Shu-Osher shock interaction





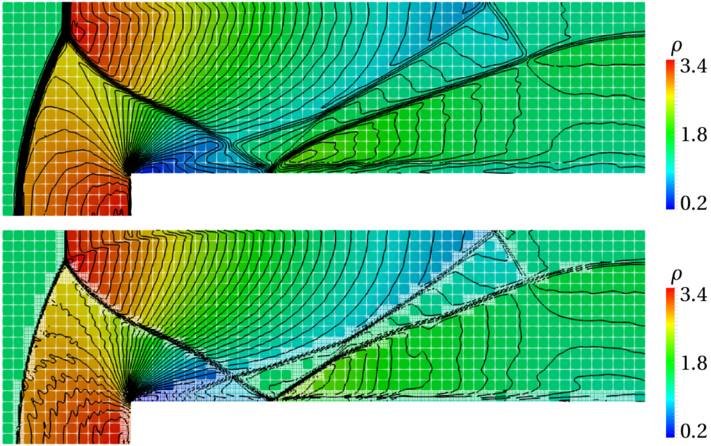
High-order artificial viscosity: 2D Riemann problem - configuration 4*



* C Schulz-Rinne. In: *SIAM Journal on Mathematical Analysis* 24.1 (1993)

High-order artificial viscosity: Double mach reflection

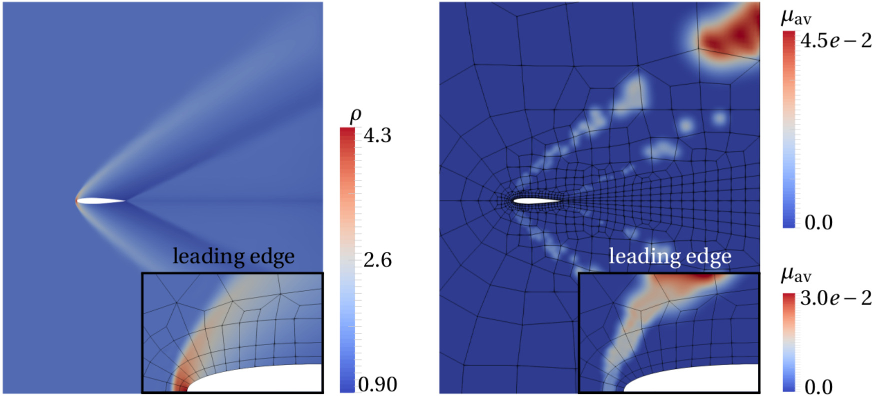
- Hybrid DG/FV scheme vs. artificial viscosity



High-order artificial viscosity: Flow around a NACA0012

Results: Unstructured grid

- Amplitude μ_a proportional to Δx
- Smooth artificial viscosity field also on unstructured grids



To summarize...

Summary:

- Proof-of-Concept: Supervised learning can be used for shock detection / localization and yields accurate results
- Binary edge map of shock can be used for local mesh refinement / artificial viscosity / ...

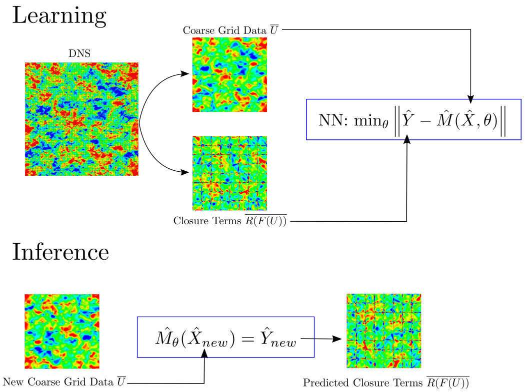
Problems / failures:

- Analytical functions in training set have to be chosen wisely!
- NNs are data hungry and computationally expensive...
- What about generalization to other test cases or polynomial orders?
- And in turn, what about long-term stability, symmetries, ...?

Data-driven turbulence closures

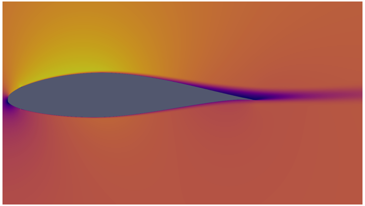
Joined work with:

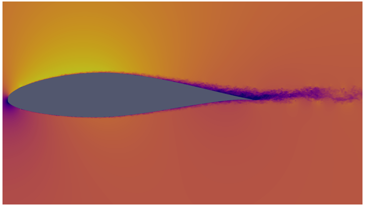
Marius Kurz



Turbulence in a nutshell

- Turbulent flow is a multiscale problem in space and time
- Full scale resolution (DNS) rarely feasible: Coarse scale formulation of NSE is necessary
- Filtering the NSE: Evolution equations for coarse scale quantities, but with a closure term / regularization dependent on the filtered full scale solution
  $\Rightarrow$ Model depending on the coarse scale data needed!
- Two filter concepts: Averaging in time (RANS) or low-pass filter in space (LES)
- Important consequence: RANS can be discretization independent, LES is (typically) not!
- 50 years of research: Still no universal closure model

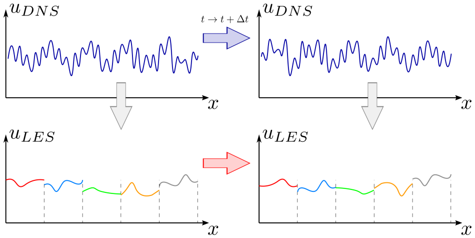




Problem definition

Choice of LES formulations

- Scale separation filter: implicit/explicit, linear/non-linear, isotropic/non-isotropic,...
- Numerical operator part of the LES formulation or negligible
- Subgrid closure: implicit / explicit, deconvolution / stochastic modelling, ...



Essential for ML methods

- Well-defined training data (both input and output)
- Is $\overline{U}$ known explicitly? $\Rightarrow$ For 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

- Filtered NSE:

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

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

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

- Perfect closure with $\overline{U}$ (optimal LES)*

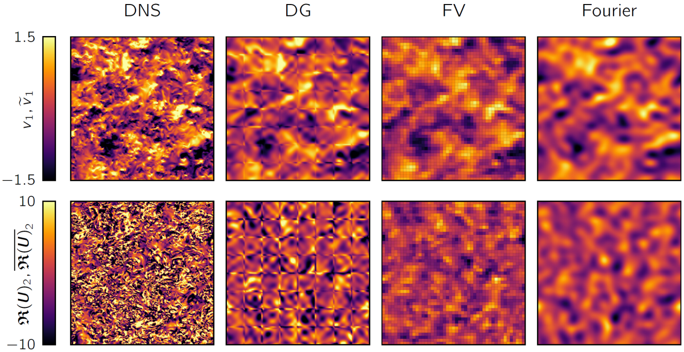
$$\frac{\partial \overline{U}}{\partial t} + \overbrace{\widetilde{R}(F(\overline{U}))}^{\text{coarse grid operator}} = \underbrace{\overbrace{\widetilde{R}(F(\overline{U}))}^{\text{coarse grid operator}} - \overline{R(F(U))}}_{\textbf{perfect closure model}}$$

- 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)

*Moser, R., et al.: "Optimal LES formulations for isotropic turbulence." JFM 398 (1999): 321-346.

Closure terms are discretization-specific!

- The closure terms are a function of the filter
- In implicitly filtered LES, the filter is induced by the discretization
- Hence, the closure terms are a function of the applied discretization

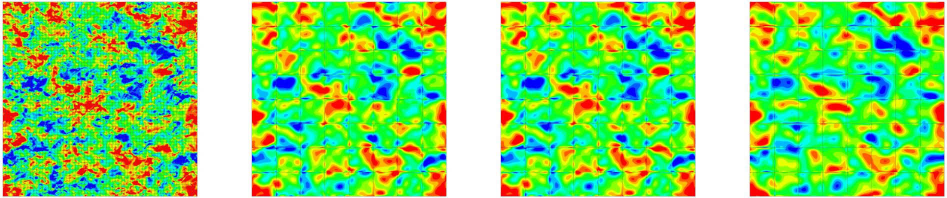


*M. Kurz. PhD thesis. University of Stuttgart, 2023

Perfect LES

- Perfect LES runs with closure term from DNS
- Decaying homogeneous isotropic turbulence
- DNS-to-LES operator $\overline{()}$: L_2 projection from DNS grid onto LES grid via discrete scale-separation filter
- DNS: 64^3 elements with $\mathcal{N} = 7$; LES operator $\widetilde{()}$: 8^3 elements with $\mathcal{N} = 5$ and split flux

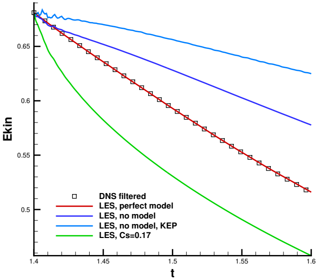
DNS | **Filtered LES** | **Perfect LES** | **LES with Smagorinsky model $C_s = 0.17$**

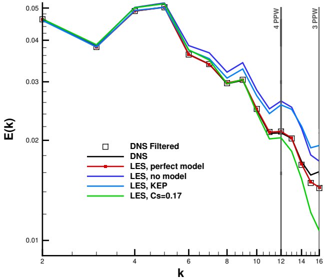


*M. Kurz et al. In: *ETNA - Electronic Transactions on Numerical Analysis* 56 (2022)

Perfect LES

- Perfect LES runs with closure term from DNS
- Decaying homogeneous isotropic turbulence
- DNS-to-LES operator $\overline{()}$: L_2 projection from DNS grid onto LES grid via discrete scale-separation filter
- DNS: 64^3 elements with $\mathcal{N} = 7$; LES operator $\widetilde{()}$: 8^3 elements with $\mathcal{N} = 5$ and split flux



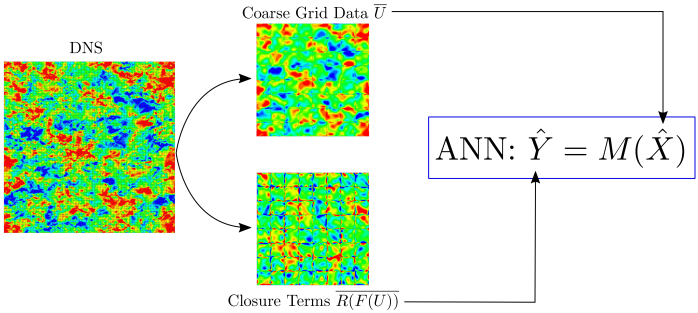


$\Rightarrow$ Perfect LES gives well-defined target and input data for supervised learning with NN

*M. Kurz et al. In: *ETNA - Electronic Transactions on Numerical Analysis* 56 (2022)

Supervised learning of closures

- Approximating an unknown, non-linear and possibly hierarchical mapping from high-dimensional input data to an output $\Rightarrow$ ANN / supervised learning
- Supervised learning from consistent data: predict subfilter terms or fit model constants



However: What to do if the filter is unknown?

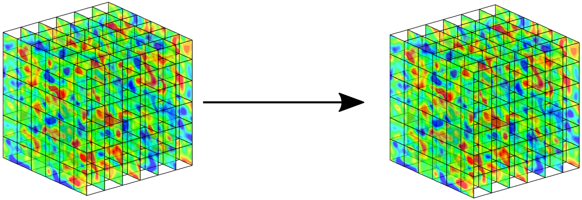
Supervised learning of closures

Dataset:

- Ensemble of DNS runs of forced homogeneous isotropic turbulence ("Turbulence-in-a-box")
- 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 $\overline{R(F(U))}$

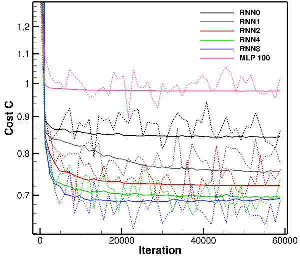


Iso-contours of the ${}_2$-criterion*

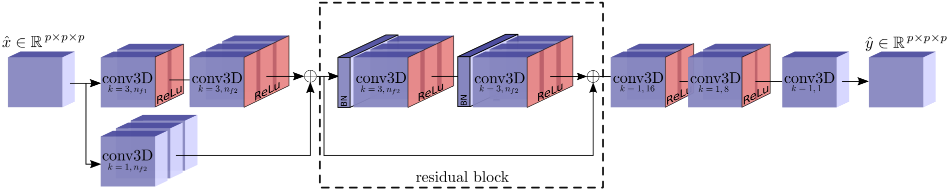
* M. Kurz. PhD thesis. University of Stuttgart, 2023

Networks and training

- CNNs with skip connections (RNN), ADAM optimizer, ...
- 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 set



Cost function: RNNs outperform MLP, deeper networks learn better

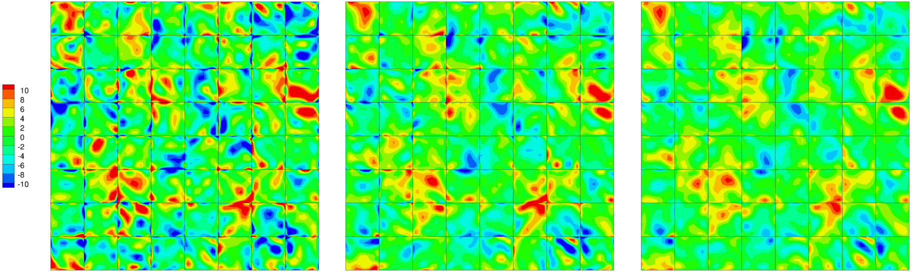


*Gamahara et al.: "Searching for turbulence models by artificial neural network." Physical Review Fluids 2.5 (2017)

Homogeneous isotropic turbulence

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

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



$\mathcal{CC} \approx 0.47$ | $\mathcal{CC} \approx 0.34$

*M. Kurz et al. In: *ETNA - Electronic Transactions on Numerical Analysis* 56 (2022)

Can we do better?

So far:

- Neglecting the temporal evolution of turbulence and the closure terms

Solution:

- NNs that model dynamic temporal behaviours are called sequence models or recurrent NNs
- General form (of a uni-directional RecNN):

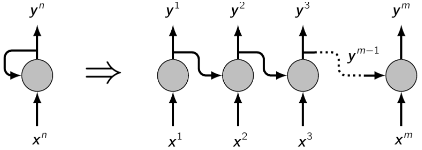
$$\hat{Y}^{t+1} = f(\underbrace{X^{t+1}}_{\text{input}}, \underbrace{m(\hat{Y}^{t}, \hat{Y}^{t-1}, \dots)}_{\text{"memory"}})$$

- RecNN-Architectures: Gated Recurrent Unit (GRU) / Long Short Term Memory (LSTM)

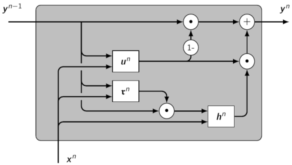
Drawback:

- Predicting long term sequential input can lead to exponential error growth

$\Rightarrow$ Long term stability is currently a problem



General layout of an RNN



Outline of a GRU cell

*M. Kurz. PhD thesis. University of Stuttgart, 2023

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.

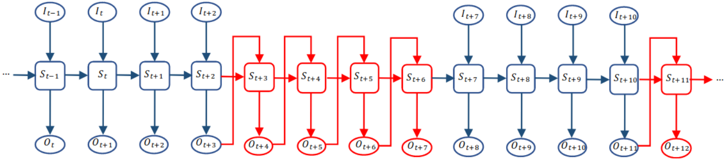


Figure 1: Visual diagram of an unrolled Auto-Conditioned RNN (right) with condition length $v = 4$ and ground truth length $u = 4$. I_t is the input at time step t. S_t is the hidden state. O_t is the output.

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.

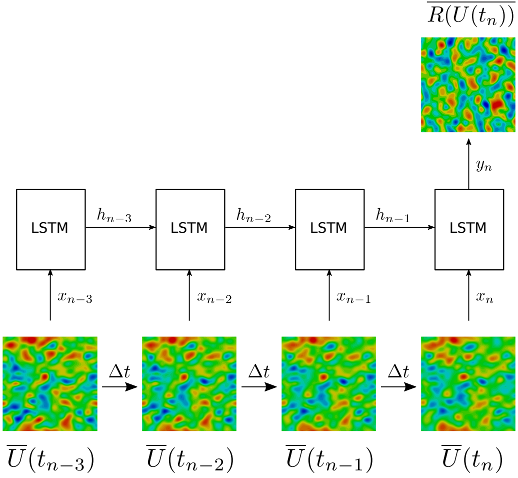
Back to LES closure predictions

Dataset:

- Ensemble of DNS runs of forced homogeneous isotropic turbulence ("Turbulence-in-a-box")
- 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: time series of velocities $\overline{U_i}$
- Output labels: DNS closure terms on the LES grid $\overline{R(F(U))}$

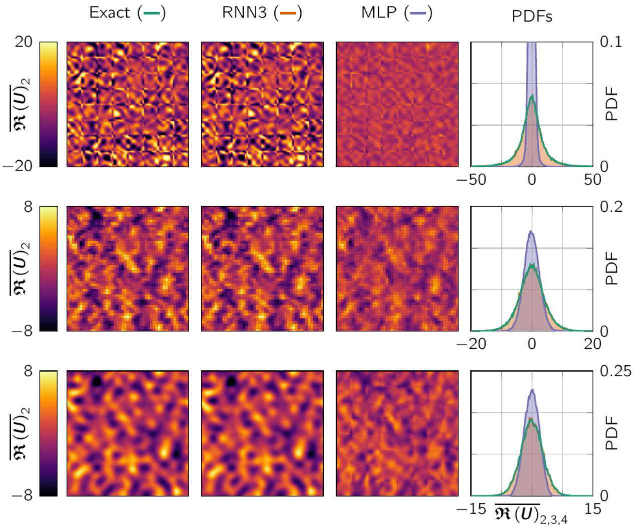


Performance of network architectures

- RNNs outperform MLP and CNN architectures by a lot!

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



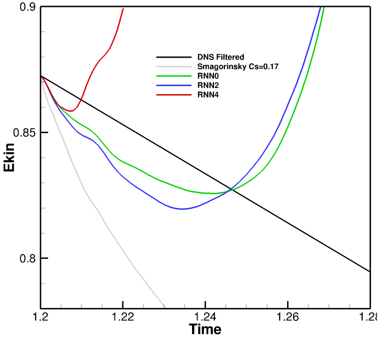
* M. Kurz et al. In: *ETNA - Electronic Transactions on Numerical Analysis* 56 (2022)

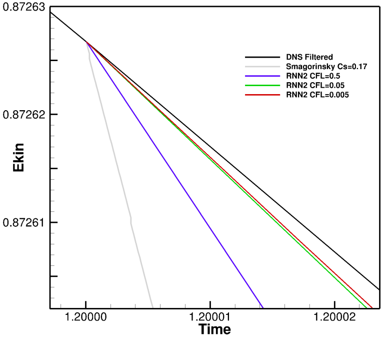
However ...

- Perfect LES is possible, but the NN-learned mappings are approximate

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

- Our process is data-limited, i.e., learning can be improved with more data
- No long term stability, but short term stability and dissipation

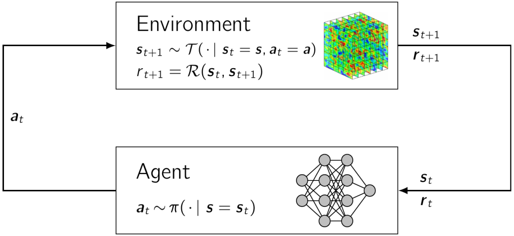




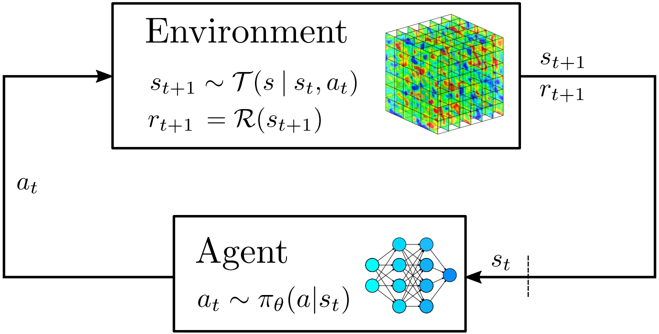
Issues with supervised learning

- Discretization not inherently considered by supervised learning
- Approach is data-limited! NNs are very data-hungry!
- Bad prediction, bad training set
- For multiscale models: data-dependence on the scale definition
- Works well with static, independent snapshots, but not in dynamical feedback systems
  -> stability problems
- Instead: Reinforcement learning

Reinforcement learning



Reinforcement learning



- a_t: Action at step t
- s_t: State at step t
- r_t: Reward at step t
- $\pi_\theta(a_t|s_t)$: Policy with parameters θ

Reinforcement Learning

Finding a policy

- π: a "control strategy" or a behavioral model
- Many strategies for finding π: policy-based or value (Q-)based: We use a policy-based approach (allows continuous state/action spaces and is more robust)
- $\pi = \pi_\theta$ and model it as a neural network ("policy net" with parameters θ).
- The objective for the MDP can be defined as the expected discounted reward per episode (if started at state s_0 with a fixed policy π.)

$$\max_\theta J(\theta) = \max_\theta \mathbf{E}_\theta \sum_{k=1}^{T} \gamma^k r_k \tag{2}$$

- This can be solved by a "gradient ascent" method: $\theta' = \theta + \lambda \nabla_\theta J(\theta)$

Reinforcement learning - training

- **Task**: find θ^* such that $\pi_{\theta^*}(a_t|s_t)$ maximizes the collected reward r_t

- Use good old gradient ascent!

$$\theta^{new} = \theta^{old} + \alpha \nabla_\theta J(\theta)$$

- We have to estimate the steepest gradient with respect to the weights θ: *

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta}\left[\underbrace{\left(\sum_{k=1}^{N} \gamma^k r_k\right)}_{\text{Cum. reward over } \tau} \underbrace{\nabla_\theta \log \pi_\theta(\tau)}_{\text{Grad. of the policy}} \right]$$

- Approximate $E_{\tau\sim\pi_\theta}$ by sampling some discrete trajectories $\tau^{(i)}$:

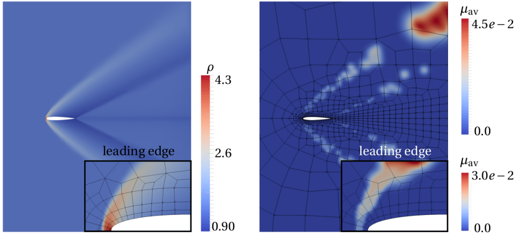
$$\tau^{(1)} = \{(s_0, a_0), (s_1, a_1, r_1), \ldots, (s_N, a_N, r_N)\}$$

* D. Silver et al. In: *31st International Conference on Machine Learning*. Vol. 32. PMLR, 2014

Data-driven shock capturing

Joint work with:

Jens Keim



* J. Zeifang et al. In: *Journal of Computational Physics* 441 (2021)

Motivation

Solution approaches:

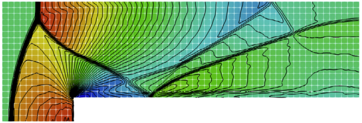
- Artificial viscosity
- Filtering/Limiting
- TVD or TVB stable finite volume scheme
- Blending of a high- with a low-order scheme

A priori:

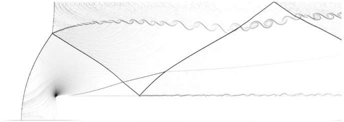
- Based on heuristic indicators
- Test case and setup dependent
- Parameter tuning

A posteriori:

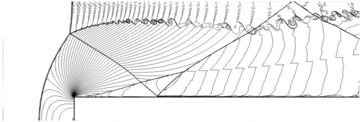
- Based on the admissibility of the solution
- Re-computation of invalid solutions



Zeifang, Beck, (2021)



Sonntag, Munz, (2017)



Dumbser, Zanotti, Loubère, Diot, (2014)

Second-Order Finite Volume Schemes

Conservation law:

$$q_t + f(q)_x = 0$$

MUSCL-Hancock:

$$Q_i^* = Q_i^n - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(f(q_i^+) - f(q_i^-)\right),$$

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left(g^*_{i+\frac{1}{2}} - g^*_{i-\frac{1}{2}}\right)$$

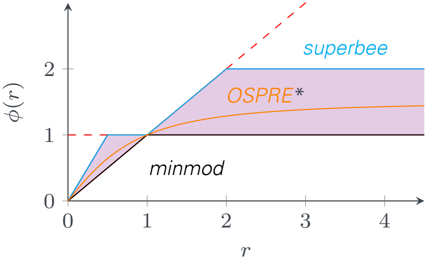
Reconstruction:

$$q_i^{\pm} = Q_i^n \pm \frac{\Delta x}{2} s_i^n,$$

$$s_i^n = \frac{q_{i+1}^n - q_i^n}{\Delta x}\phi(r_i^n), \quad r_i^n = \frac{q_i^n - q_{i-1}^n}{q_{i+1}^n - q_i^n}$$

*Waterson, Deconinck, Num. Meth. in Laminar and Turb. Flow 9 (1995)

TVD region:



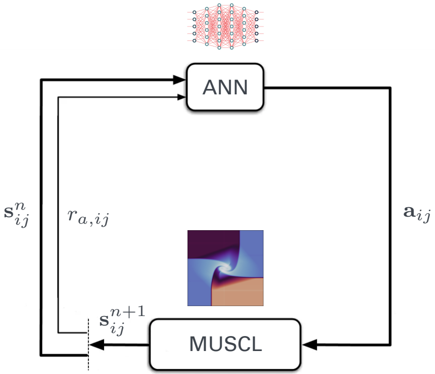
Goal:

- An a priori limiter which has ...
- the properties of an a posteriori limiter, ...
- following the idea of the MOOD approach, ...
- by the use of reinforcement learning.

Reinforcement Learning

Markov decision process:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r_a)$$



Environment:

- Second-order MUSCL-Hancock scheme applied to the Euler equations

Agent:

- "Learns"/predict an optimal/admissible slope

State:

- Integral mean values of the present, the adjacent and the diagonal cells

Action:

- Blending parameters between a fully right and a fully left sided slope

Reward:

- Immediate value of the present action as a function of the state and the action

State

- The state at t^n is composed of an extended nine-point stencil, given as

$$\mathbf{s}^n_{i,j} = \begin{pmatrix} (\mathcal{V}_{i-,j+}, \mathcal{V}_{i,j+}, \mathcal{V}_{i+,j+}) \\ (\mathcal{V}_{i-,j}\ \ , \mathcal{V}_{i,j}\ \ , \mathcal{V}_{i+,j}\ \ ) \\ (\mathcal{V}_{i-,j-}, \mathcal{V}_{i,j-}, \mathcal{V}_{i+,j-}) \end{pmatrix}$$

with

$$i = 1, \ldots, \mathcal{N}_x \quad \text{and} \quad j = 1, \ldots, \mathcal{N}_y$$

and

$$(\cdot)^- := (\cdot) - 1 \quad \text{and} \quad (\cdot)^+ := (\cdot) + 1.$$

- The components of $\mathcal{V} = \{\rho, p\}$ are defined as the density and the pressure, respectively.

- Use of a min-max normalization, defined as

$$\mathrm{NORMALIZE}(\mathbf{s}_{ij}) = \begin{cases} \frac{\mathbf{s}_{ij} - \min(\mathbf{s}_{ij})}{\max(\mathbf{s}_{ij}) - \min(\mathbf{s}_{ij})} & : \max(\mathbf{s}_{ij}) - \min(\mathbf{s}_{ij}) > \epsilon_1 \max(\mathbf{s}_{ij}), \\ 1 & : \text{otherwise} \end{cases}$$

  which maps the state space to a bounded interval $\tilde{\mathbf{s}}_{ij} \in [0, 1]^{3\times3\times2}$.

- Distinguish constant from non-constant states by the additional parameter $\epsilon_1 = 10^{-5}$.

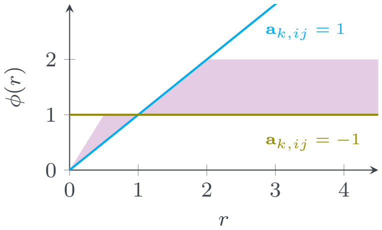
Action

Goal:

- The definition of the action has to maintain the second-order character of the scheme.

Idea:

- Use a convex blending of a fully left- and a fully right-sided slope.



- The slopes of the primitive variables $\mathbf{V}_k = (\rho, \mathbf{u}_k, p)^{\mathsf{T}}$ in each dimension are defined as

$$\delta \mathbf{V}_{k,ij} = \frac{1}{2}\left[\left(1 - \mathbf{a}_{k,ij}\right)\frac{\mathbf{V}_{l+} - \mathbf{V}_l}{\Delta \mathbf{x}_k} + \left(1 + \mathbf{a}_{k,ij}\right)\frac{\mathbf{V}_l - \mathbf{V}_{l-}}{\Delta \mathbf{x}_k}\right] \quad \text{with} \quad l = \begin{cases} i & : k = 1, \\ j & : k = 2, \end{cases}$$

  where $\mathbf{a}_{k,ij} \in [-1, 1]$.

- Constant states are always treated with $\mathbf{a}_{k,ij} = 1$.

Reward

Goal:

- Design of an a priori limiter which has the properties of an a posteriori limiter based on the MOOD approach*†
- Check the admissibility of the solution in terms of the positivity and the boundedness.

Positivity:

$$\mathbb{S}_{ij} = \mathrm{SANITY}(\mathbf{s}_{ij}^{n+1}) = \begin{cases} 1 & : \min(\mathbf{s}_{ij}^{n+1}) < \epsilon_2 \\ 0 & : \text{otherwise} \end{cases} \quad \text{with} \quad \epsilon_2 = 10^{-6}$$

Boundedness:

$$\begin{aligned} \mathbb{M}_{ij} &= \mathbb{M}(\mathbf{s}_{ij}^{n}, \mathbf{s}_{ij}^{n+1}) \\ &= \begin{cases} 1 & : \mathbf{s}_{ij}^{n+1} < (1-\epsilon_3)\min(\mathbf{s}_{ij}^{n}) \;\vee\; \mathbf{s}_{ij}^{n+1} > (1+\epsilon_3)\max(\mathbf{s}_{ij}^{n}) \\ 0 & : \text{otherwise} \end{cases} \quad \text{with} \quad \epsilon_3 = 10^{-3} \end{aligned}$$

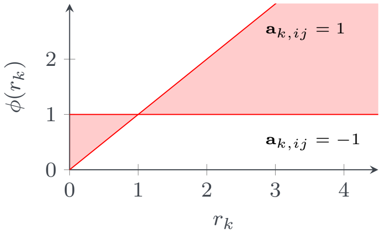
*Clain, Diot, Loubère, J. Comput. Phys. 230(10), (2011)

†Dumbser, Zanotti, Loubère, Diot, J. Comput. Phys. 278 (2014)

Reward

- This enables the definition of a reward in each direction $\mathrm{REW} = \mathrm{REW}(\mathbf{s}_{ij}^{n}, \mathbf{s}_{ij}^{n+1}, \mathbf{a}_{ij}, k)$ as

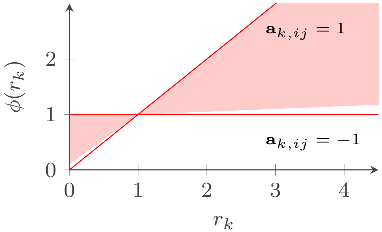
$$\mathrm{REW} = \begin{cases} c_1 & : \mathbb{S}_{ij} = 1, \\ & \\ & \end{cases}$$



Reward

- This enables the definition of a reward in each direction $\mathrm{REW} = \mathrm{REW}(\mathbf{s}_{ij}^{n}, \mathbf{s}_{ij}^{n+1}, \mathbf{a}_{ij}, k)$ as

$$\mathrm{REW} = \begin{cases} c_1 & : \mathbb{S}_{ij} = 1, \\ c_2 |\mathbf{a}_{k,ij}| & : \mathbb{M}_{ij} = 1 \land \neg \mathbb{J}_k, \\ & \\ & \end{cases}$$



- The additional criterion

$$\mathbb{J}_k = (\mathbf{a}_{k,ij} < -1 + \epsilon_4 : |r_k| > 1) \quad \vee \quad (\mathbf{a}_{k,ij} > 1 - \epsilon_4 : |r_k| < 1) \quad \text{with} \quad \epsilon_4 = 0.1$$

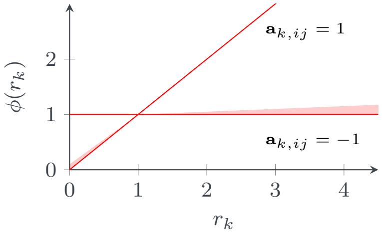
  is used to avoid an invalid penalization in the limit case of the *minmod*.

- The tuning parameters are fixed to $c_1 = -50$ and $c_2 = -10$.

Reward

- This enables the definition of a reward in each direction $\mathrm{REW} = \mathrm{REW}(\mathbf{s}_{ij}^n, \mathbf{s}_{ij}^{n+1}, \mathbf{a}_{ij}, k)$ as

$$\mathrm{REW} = \begin{cases} c_1 & : \mathbb{S}_{ij} = 1, \\ c_2 |\mathbf{a}_{k,ij}| & : \mathbb{M}_{ij} = 1 \wedge \neg \mathbb{J}_k, \\ |\mathbf{a}_{k,ij}| & : \mathbb{M}_{ij} = 1 \wedge \mathbb{J}_k, \end{cases}$$



- The additional criterion

$$\mathbb{J}_k = (\mathbf{a}_{k,ij} < -1 + \epsilon_4 : |r_k| > 1) \quad \vee \quad (\mathbf{a}_{k,ij} > 1 - \epsilon_4 : |r_k| < 1) \quad \text{with} \quad \epsilon_4 = 0.1$$

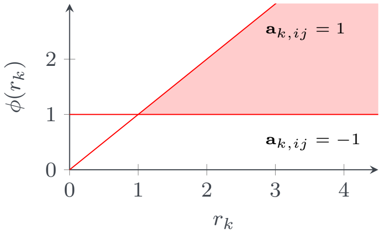
is used to avoid an invalid penalization in the limit case of the *minmod*.

- The tuning parameters are fixed to $c_1 = -50$ and $c_2 = -10$.

Reward

- This enables the definition of a reward in each direction $\mathrm{REW} = \mathrm{REW}(\mathbf{s}_{ij}^n, \mathbf{s}_{ij}^{n+1}, \mathbf{a}_{ij}, k)$ as

$$\mathrm{REW} = \begin{cases} c_1 & : \mathbb{S}_{ij} = 1, \\ c_2|\mathbf{a}_{k,ij}| & : \mathbb{M}_{ij} = 1 \wedge \neg\mathbb{J}_k, \\ |\mathbf{a}_{k,ij}| & : \mathbb{M}_{ij} = 1 \wedge \mathbb{J}_k, \\ \mathbf{a}_{k,ij} & : \mathbb{M}_{ij} = 0 \wedge |r_k| > 1, \\ \end{cases}$$



- The additional criterion

$$\mathbb{J}_k = (\mathbf{a}_{k,ij} < -1 + \epsilon_4 : |r_k| > 1) \quad \vee \quad (\mathbf{a}_{k,ij} > 1 - \epsilon_4 : |r_k| < 1) \quad \text{with} \quad \epsilon_4 = 0.1$$

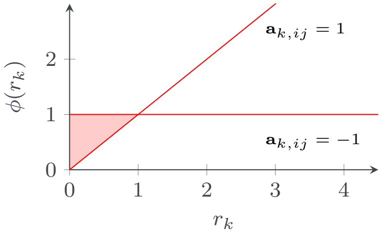
is used to avoid an invalid penalization in the limit case of the *minmod*.

- The tuning parameters are fixed to $c_1 = -50$ and $c_2 = -10$.

Reward

- This enables the definition of a reward in each direction $\mathrm{REW} = \mathrm{REW}(\mathbf{s}_{ij}^n, \mathbf{s}_{ij}^{n+1}, \mathbf{a}_{ij}, k)$ as

$$\mathrm{REW} = \begin{cases} c_1 & : \mathbb{S}_{ij} = 1, \\ c_2 |\mathbf{a}_{k,ij}| & : \mathbb{M}_{ij} = 1 \wedge \neg \mathbb{J}_k, \\ |\mathbf{a}_{k,ij}| & : \mathbb{M}_{ij} = 1 \wedge \mathbb{J}_k, \\ \mathbf{a}_{k,ij} & : \mathbb{M}_{ij} = 0 \wedge |r_k| > 1, \\ -\mathbf{a}_{k,ij} & : \mathbb{M}_{ij} = 0 \wedge |r_k| < 1. \end{cases}$$



- The additional criterion

$$\mathbb{J}_k = (\mathbf{a}_{k,ij} < -1 + \epsilon_4 : |r_k| > 1) \quad \vee \quad (\mathbf{a}_{k,ij} > 1 - \epsilon_4 : |r_k| < 1) \quad \text{with} \quad \epsilon_4 = 0.1$$

  is used to avoid an invalid penalization in the limit case of the *minmod*.

- The tuning parameters are fixed to $c_1 = -50$ and $c_2 = -10$.

- The final reward reads as $r_{a,ij} = \mathrm{REWARD}(\mathbf{s}_{ij}^n, \mathbf{s}_{ij}^{n+1}, \mathbf{a}_{ij}) = \sum_k \mathrm{REW}(\mathbf{s}_{ij}^n, \mathbf{s}_{ij}^{n+1}, \mathbf{a}_{ij}, k)$

Training

2D Riemann problems: *

- Euler equations + ideal gas EoS ($\gamma = 1.4$)
- $\Omega = [0,1]^2$, $t_{\text{end}} = 0.2$
- $\mathcal{N}_{\text{el}} = \mathcal{N}_x \times \mathcal{N}_x = 50 \times 50$
- CFL = 0.99, *HLL* flux

Initialization:

$$\rho \sim \mathcal{U}(0.01, 4),$$
$$u_1, u_2 \sim \mathcal{U}(-2, 2),$$
$$p \sim \mathcal{U}(0.01, 10)$$

Training:

- Epochs: 1000 for 2h
- GPU: NVIDIA Tesla K40c

Multi-agent twin-delayed deep deterministic policy gradient (TD3, off-policy): *

Actor:

- Convolutional neural network (CNN)
- Two input channels: (p, ρ)
- Window kernel size: 2×2
- Output: $\mathbf{a_{ij}} = (a_{1,ij}, a_{1,ij})^{\mathsf{T}}$
- Trainable parameters: 4482

Critic:

- Multilayer perceptron (MLP)
- Output: $Q(\tilde{\mathbf{s}}^n, \mathbf{a})$
- Trainable parameters: 96901

* C Schulz-Rinne. In: *SIAM Journal on Mathematical Analysis* 24.1 (1993)

† G. Fu et al. In: *Journal of Computational Physics* 347.347 (2017)

2D Riemann Problem: Configuration 4

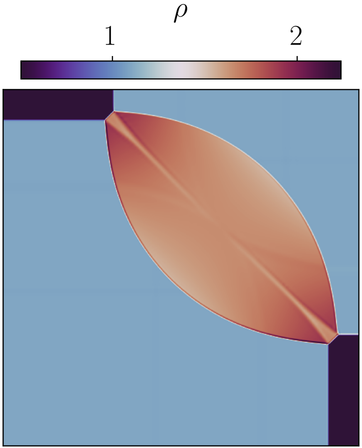


Figure: *RLindi*

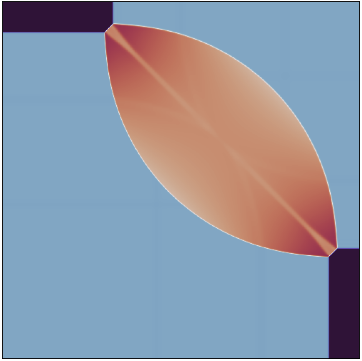


Figure: *OSPRE*

2D Riemann Problem: Configuration J

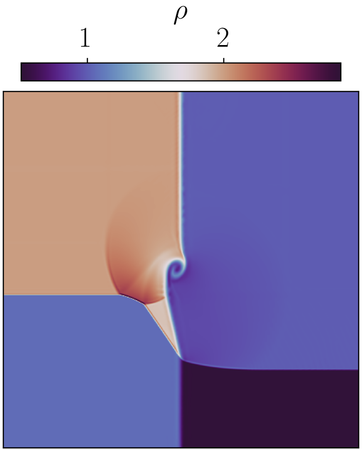


Figure: *RLindi*

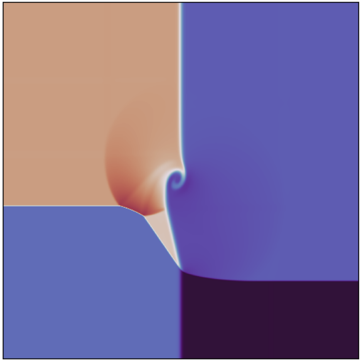


Figure: *OSPRE*

2D Riemann Problem: Configuration B

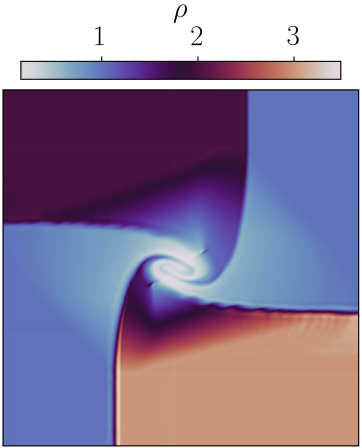


Figure: *RLindi*

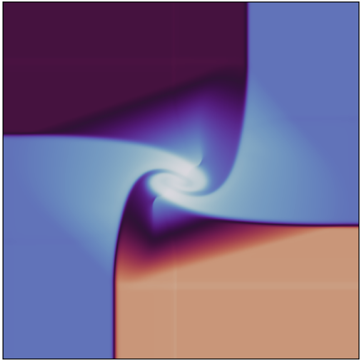


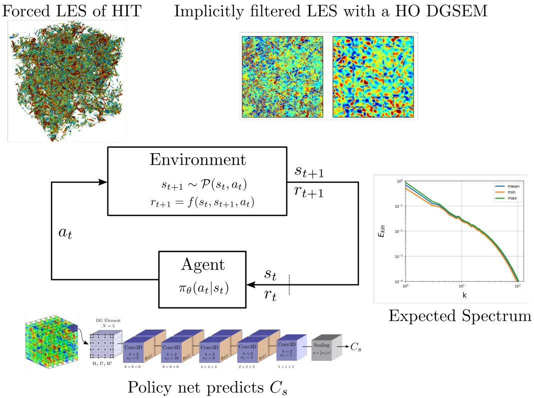
Figure: *OSPRE*

$\Rightarrow$ In reinforcement learning the definition of the reward is crucial!

Data-driven turbulence closures

Joint work with:

Marius Kurz



Explicit closure model

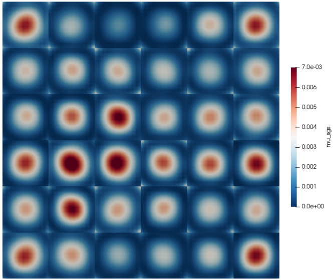
- **Baseline**: Smagorinsky's model:

$$\nu_t = (C_s\Delta)^2\sqrt{2\overline{S}_{ij}\overline{S}_{ij}} \quad \text{with } \overline{S} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right)$$

- Adapt parameter dynamically in space and time: $C_s = C_s(x,t)$

- First step: elementwise constant C_s

- Second step: elementwise quadratically distributed C_s



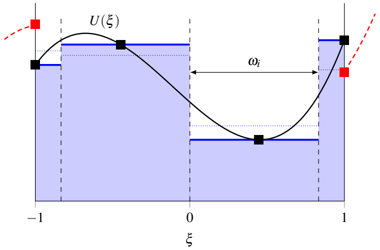
*Kurz2023

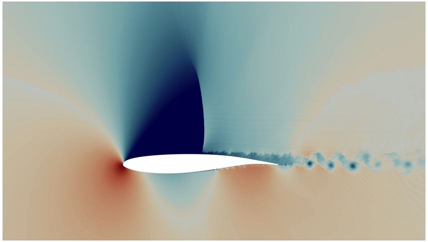
Implicit closure model*

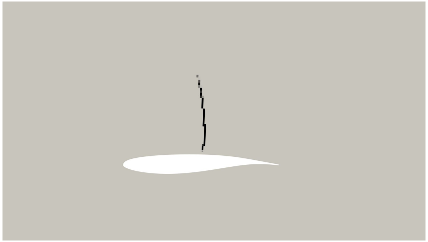
- Elementwise, convex blending between DG and FV operator

$$\hat{U}_t = \alpha\,\mathcal{R}^{FV}(\hat{U}) + (1-\alpha)\,\mathcal{R}^{DG}(\hat{U})$$

- Blending parameter $\alpha \in [0,1]$
- Originally proposed for shock capturing purposes[†]





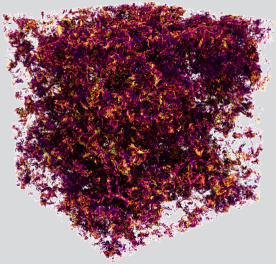


[*] A. Beck et al. In: *Physics of Fluids* 35.12 (2023)
[†] S. Hennemann et al. In: *Journal of Computational Physics* 426 (2021)
[‡] **Kurz2023**

Stating the RL problem

Environment

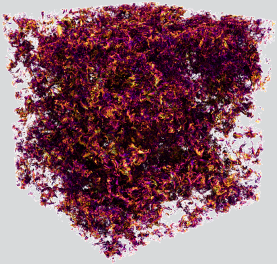
- Implicitly filtered LES with high-order DG scheme
- Homogeneous isotropic turbulence ("Turbulence-in-a-box")
- Periodic boundaries
- Forcing for statistically stationary flow



Stating the RL problem

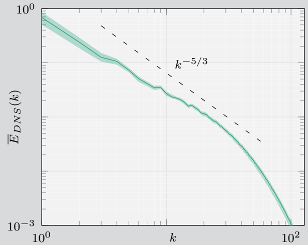
Environment

- Implicitly filtered LES with high-order DG scheme
- Homogeneous isotropic turbulence ("Turbulence-in-a-box")
- Periodic boundaries
- Forcing for statistically stationary flow



Reward

- Reward based on error in spectrum of turbulent kinetic energy
- Spectrum of precomputed DNS as target
- Reward scaled to $\mathbf{r_t} \in [-\mathbf{1}, \mathbf{1}]$ with exponential function



Stating the RL problem

Actions

Actions are elementwise and either

1. C_s constant
2. C_s quadratic
3. α constant

Stating the RL problem

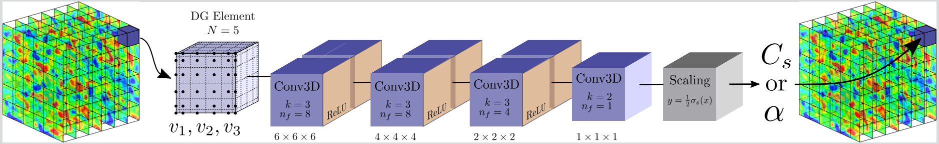
Actions

Actions are elementwise and either

1. C_s constant
2. C_s quadratic
3. α constant

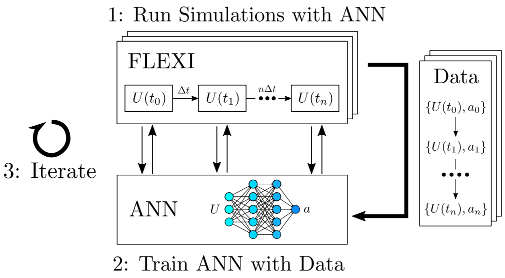
Policy

- Elementwise convolutional architecture



Computational setup

- RL training loop implemented within Relexi[†]
- RL algorithm: Proximal Policy Optimization (PPO, on-policy)*
- LES computed with high-order DG code FLEXI[‡]
- Different resolutions: 24, 32, 36, 48 DOF per directions
- Different pol. degrees: $\mathcal{N} = 3, 5, 8$
- Using up to 1024 cores in parallel for simulations



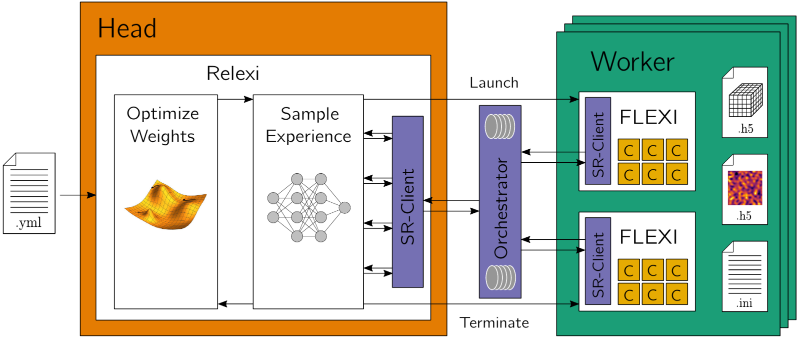
*J. Schulman et al. In: (2017)

† https://github.com/flexi-framework/relexi

‡ https://github.com/flexi-framework/flexi

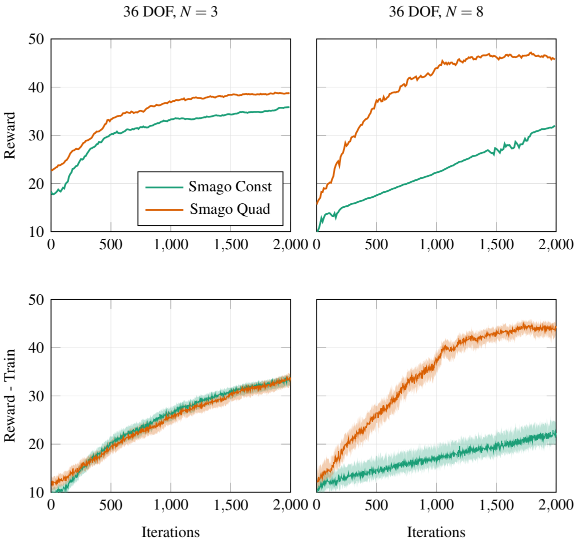
Relexi* - An RL framework for high-performance computing

- Implemented in **cooperation with HLRS and HPE**
- Distribution on hybrid HPC systems via the **SmartSim** library
- **Dedicated GPU node** for training and model evaluation with TensorFlow
- FLEXI instances distributed **across multiple CPU nodes** („Workers“)



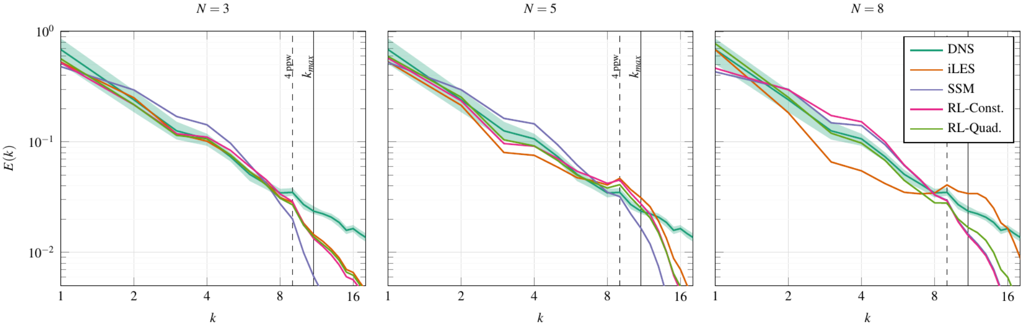
* M. Kurz et al. In: *Software Impacts* (2022)

Results - training



- The agent's policy improves and converges
- Policy improves steadily and consistently
- Improved performance for the quadratic model
- Larger elements profit more from quadratic C_s

Results - Explicit model*

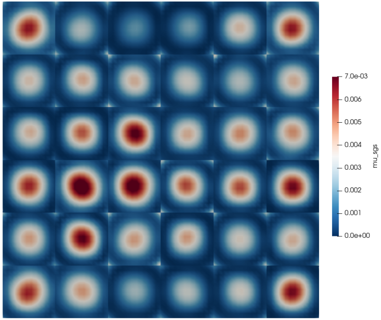


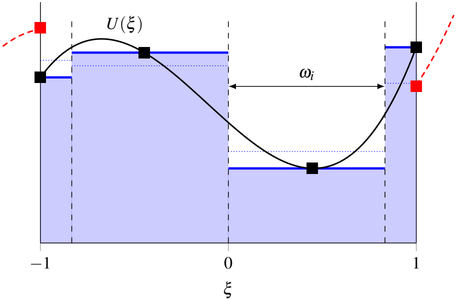
* A. Beck et al. In: *Physics of Fluids* 35.12 (2023)

Results - Explicit model*

Observation: The RL model consistently adds more dissipation within the DG element and none at the faces.

But Why?



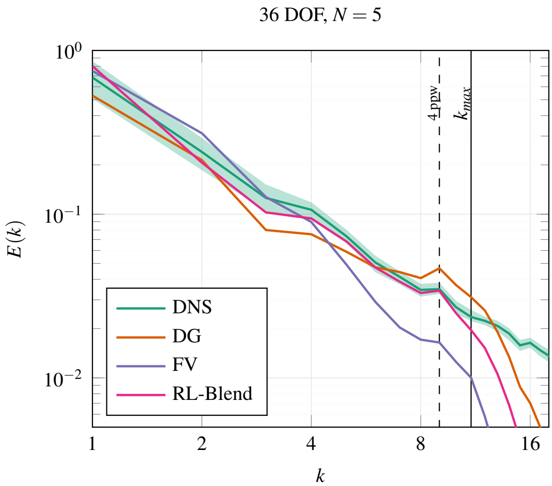


Hypothesis: This homogenizes the dissipation, which is only added at the element faces by the Riemann solver!

* A. Beck et al. In: *Physics of Fluids* 35.12 (2023)

Results - Implicit model*

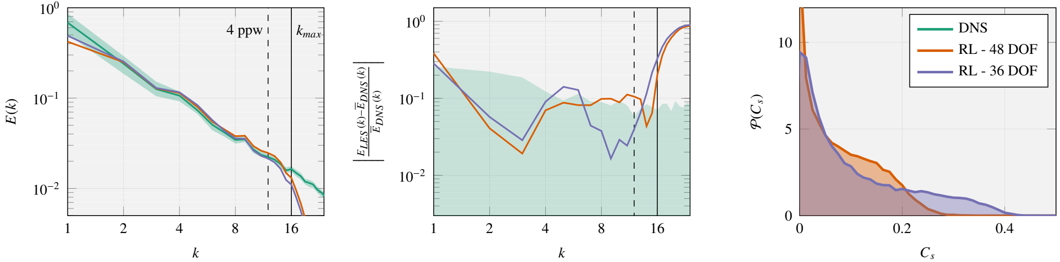
- Agent learns successfully blending between DG and FV methods
- RL-informed hybrid scheme yields better energy spectrum than both individual schemes
- Performance at minimum on par with dynamic Smagorinsky model



* A. Beck et al. In: *Physics of Fluids* 35.12 (2023)

To summarize ...

- Proof-of-Concept: Reinforcement learning can be applied for non-linear phenomena such as turbulence modeling / shock capturing and can give accurate and long-term stable results
- What about generalization? RL model trained on 36 DOF case applied to 48 DOF case:*



→ Systematic difference in predictions!

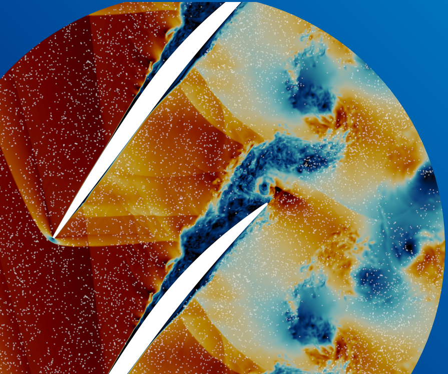
→ Reinforcement learning better than supervised learning, however, still good enough in terms of stability, accuracy and efficiency?

* Kurz2023

University of Stuttgart
Germany

FLEXI

NRG Group



Thank you for your attention!

Numerics Research Group (NRG), Prof. Dr. Andrea Beck
Institute of Aerodynamics and Gas Dynamics (IAG)
University of Stuttgart

email schwarz@iag.uni-stuttgart.de