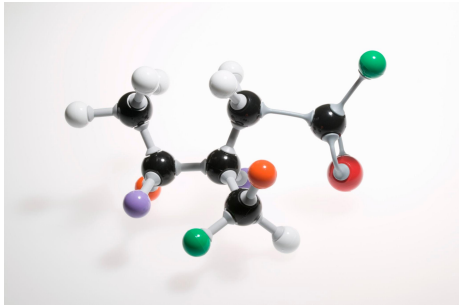
From Numerical Simulators of PDEs to Neural Emulators and Back

RISE ML Seminar | Felix Koehler | 6 Nov 2025

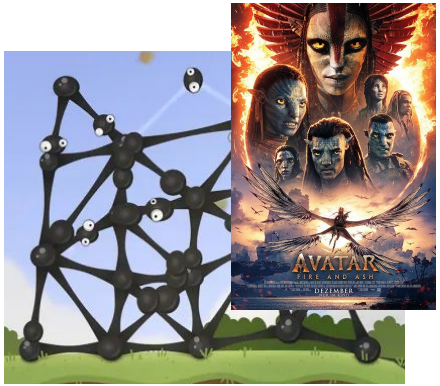
The Simulation of Natural Phenomena is Important!



Molecular/Quantum Simulations

⇩

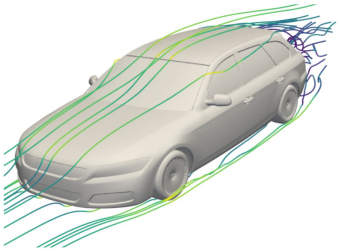
Material & Drug Discovery



Visual Effects & Game Physics

⇩

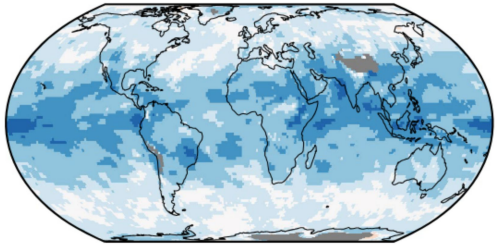
Plausibility, Immersion, Entertainment



Engineering Simulations

⇩

Saver & more efficient designs

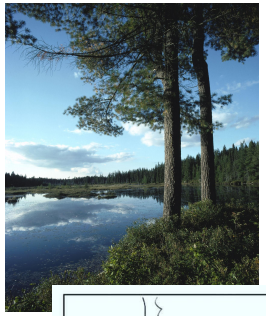


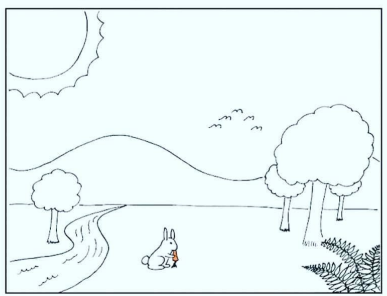
Weather & Climate

⇩

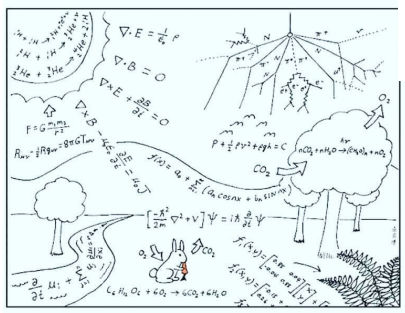
Weather Prediction, Understanding Climate Change

The Path of Simulation

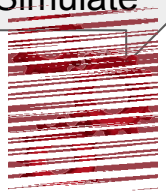


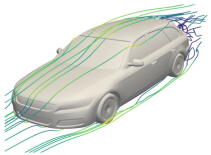


Model



Simulate



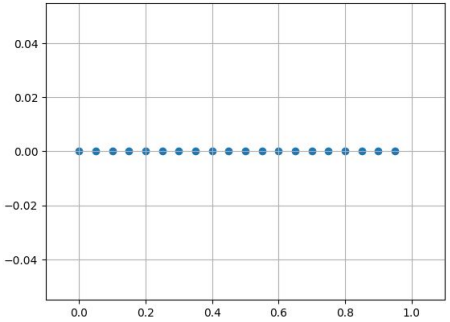


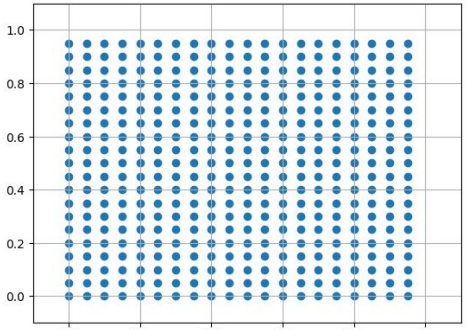
Extract

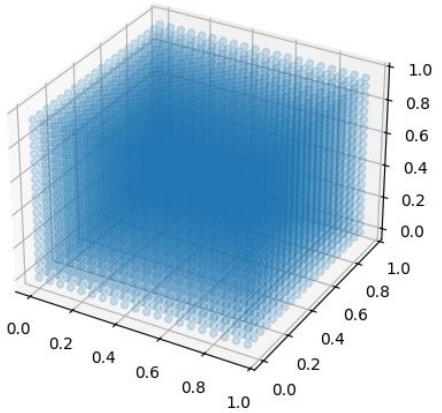
Insight

Can we approximate Simulations with Neural Networks?

Simplification: Abstract PDEs on Regular Periodic Grids

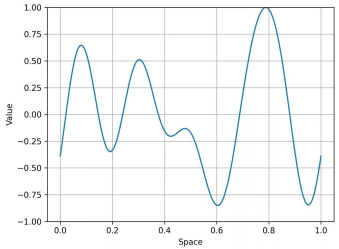


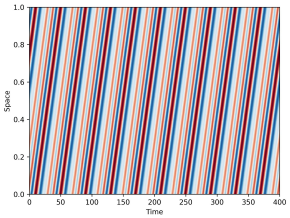


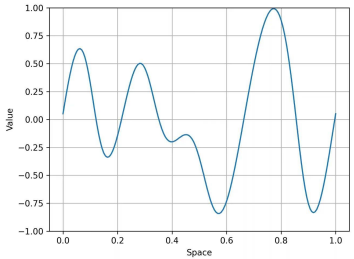


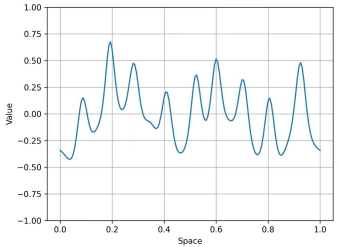
$$\frac{\partial u}{\partial t} = -b\frac{1}{2}\|\nabla u\|^2 - \nu\nabla\cdot\nabla u - \zeta\vec{1}\cdot\nabla^4 u$$

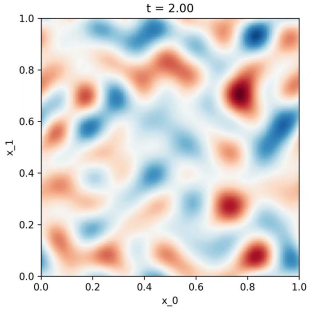
$$\frac{\partial u}{\partial t} = -c\vec{1}\cdot\nabla u \qquad \frac{\partial u}{\partial t} = -b\frac{1}{2}\nabla\cdot(u\otimes u) + \nu\nabla\cdot\nabla u$$

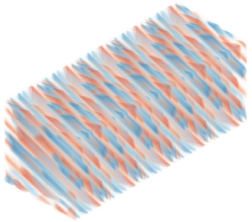

Value
Space

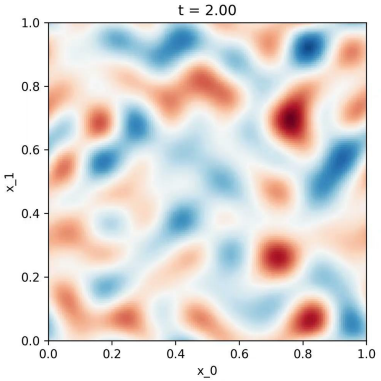

Space
Time

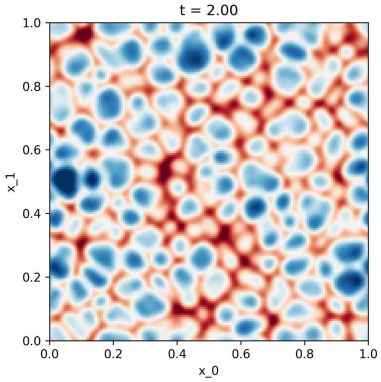

Value
Space

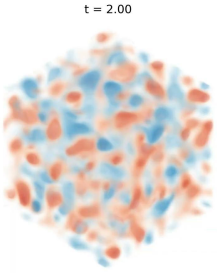

Value
Space

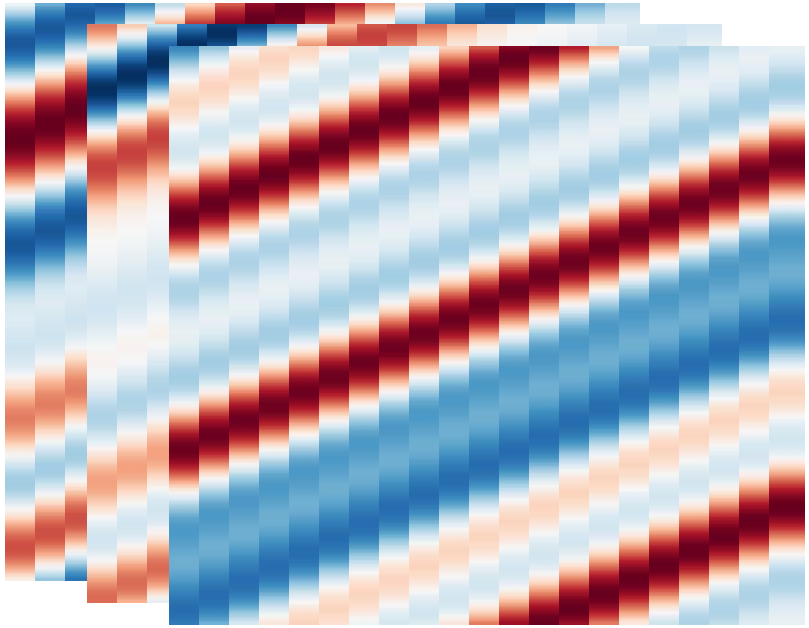

t = 2.00
x_1
x_0

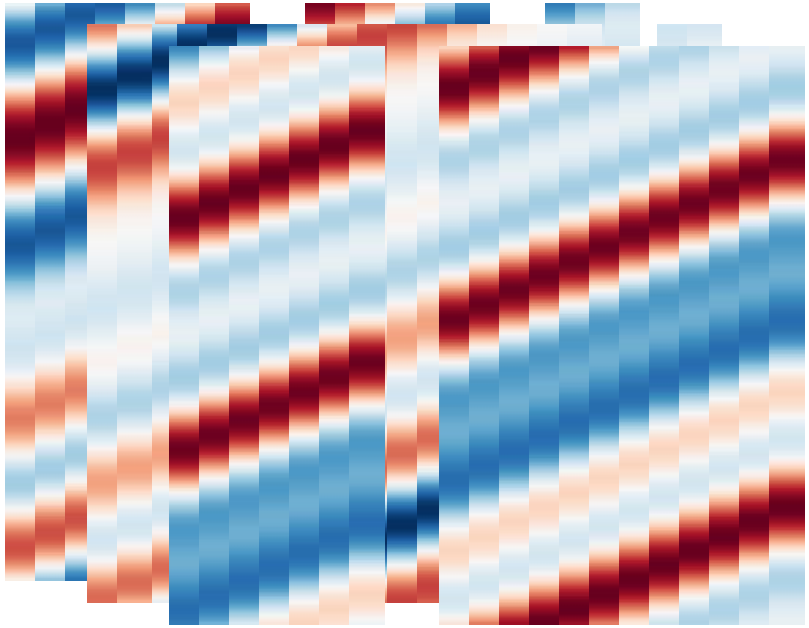


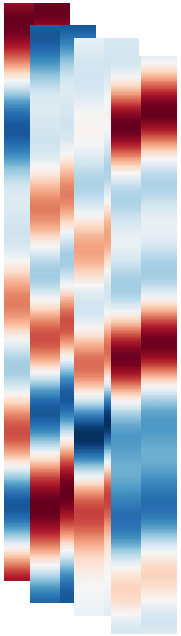

t = 2.00
x_1
x_0

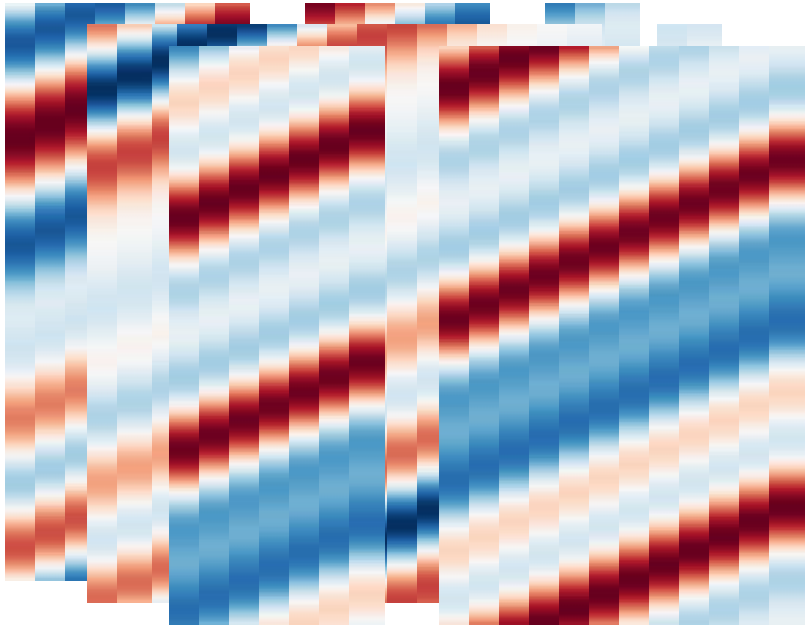

t = 2.00
x_1
x_0

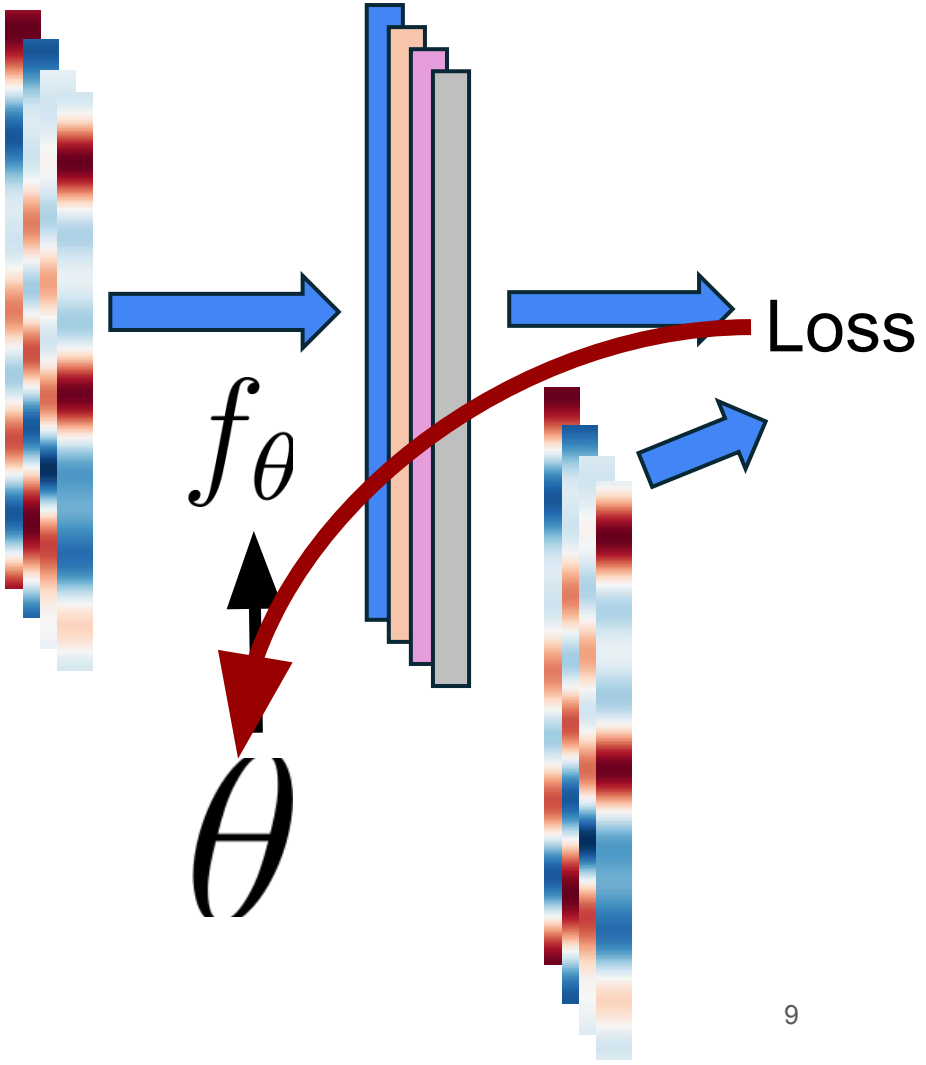

t = 2.00



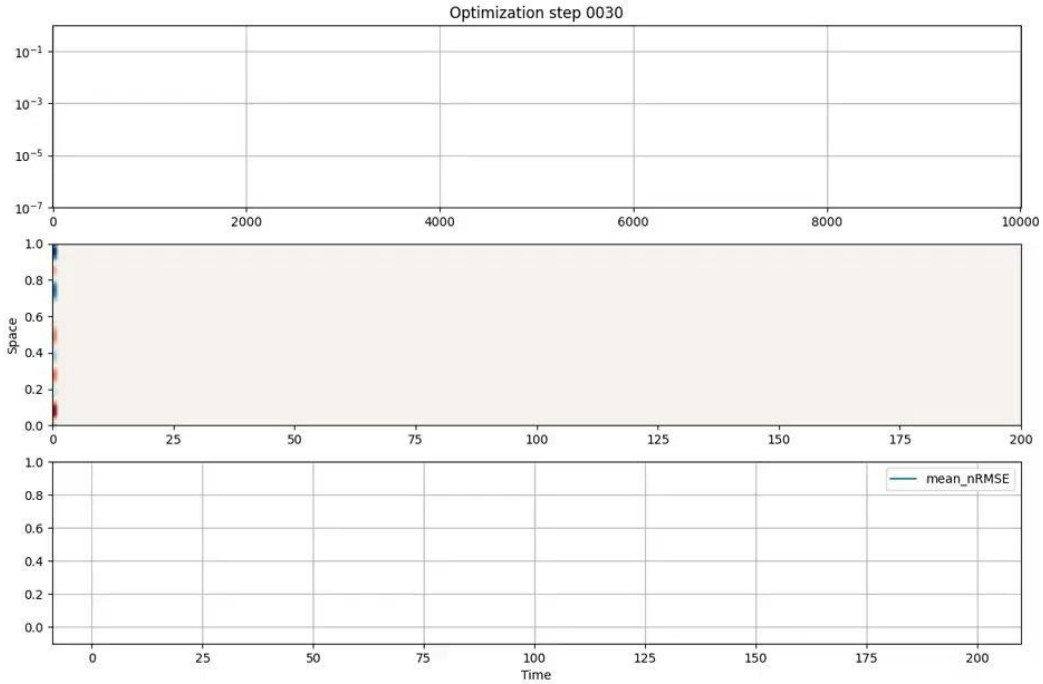




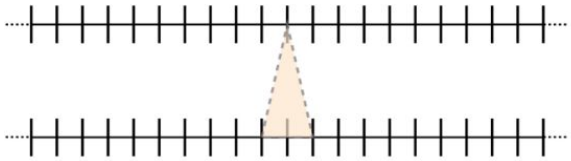



Loss
f_θ
θ

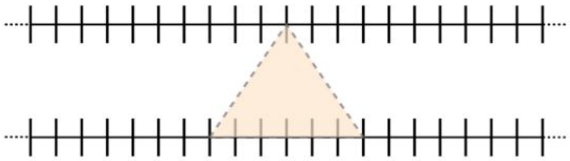
Advection Learning Animation



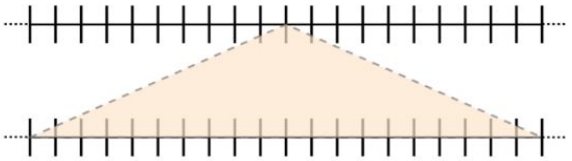
Different Parameterizations of PDEs



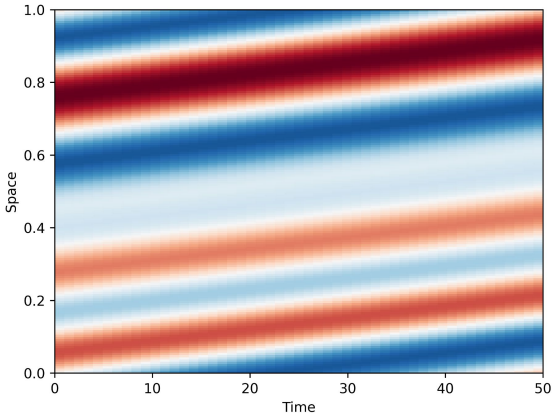
$\gamma_1 = 0.5$

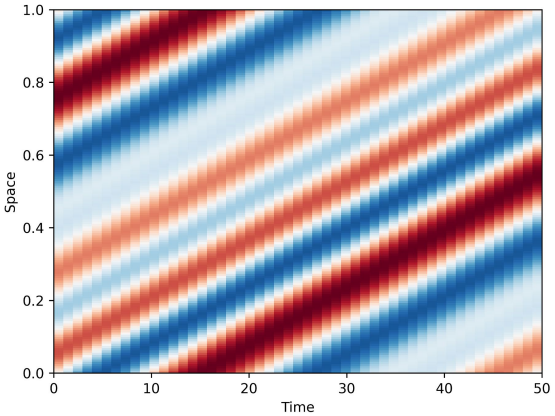


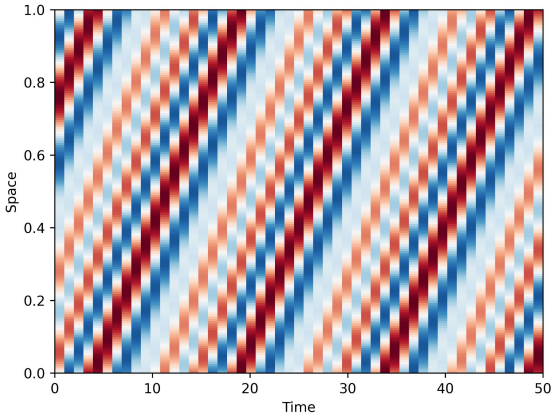
$\gamma_1 = 2.5$



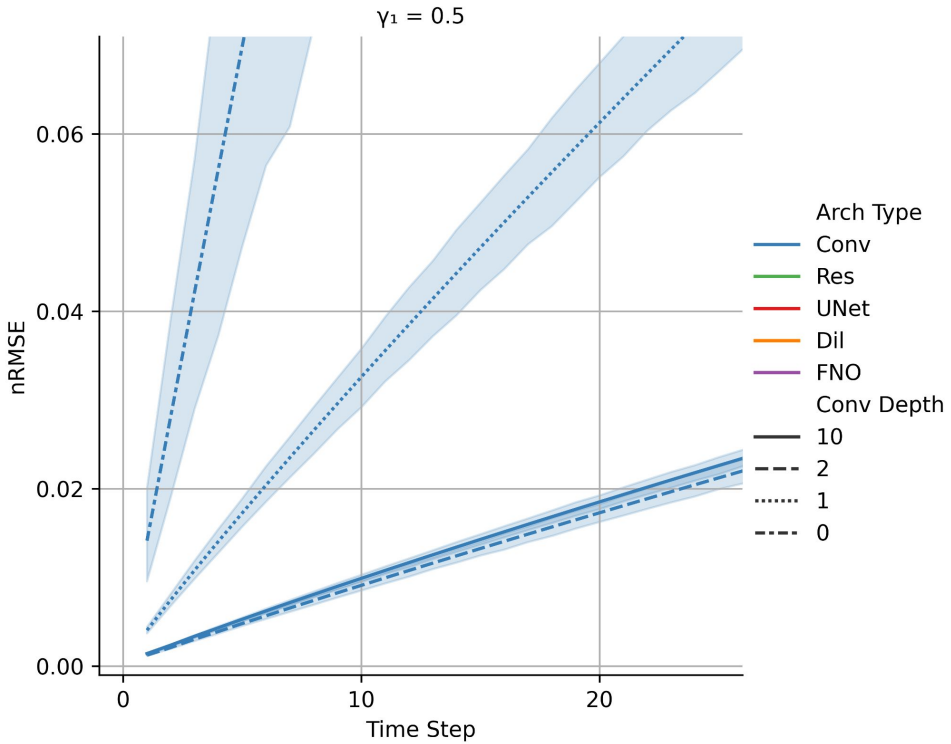
$\gamma_1 = 10.5$



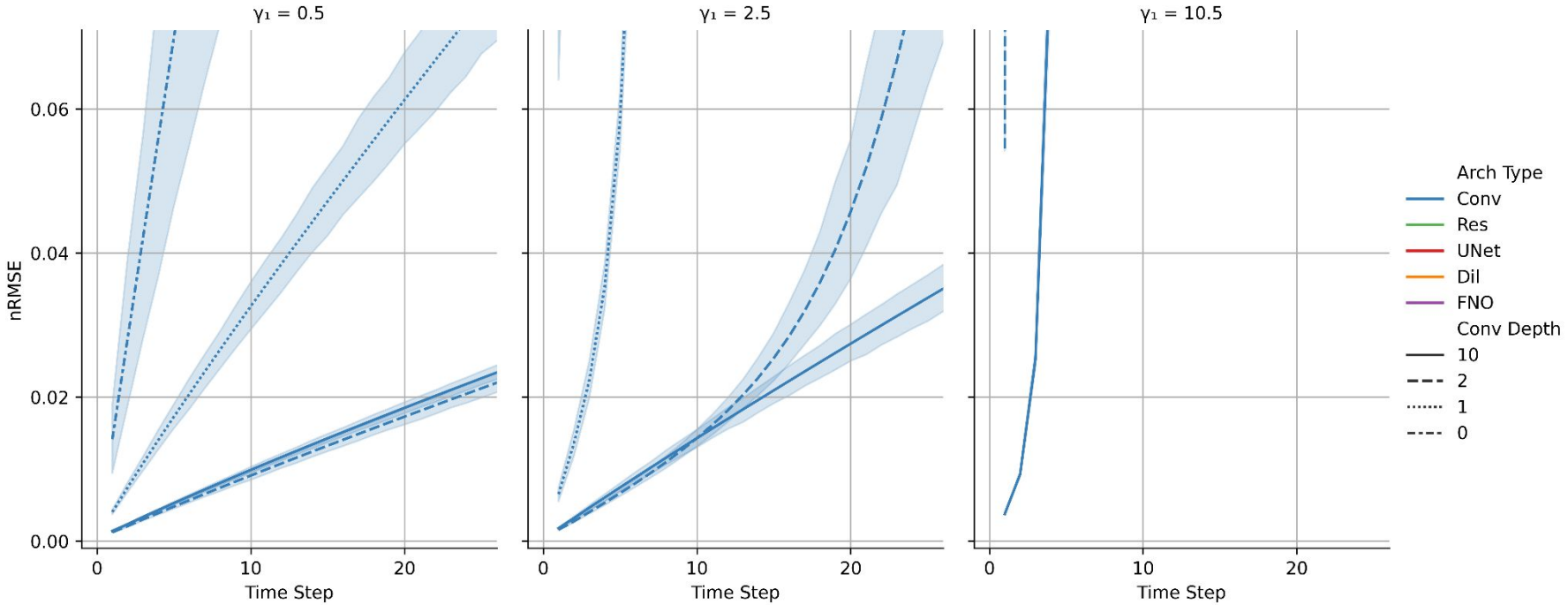




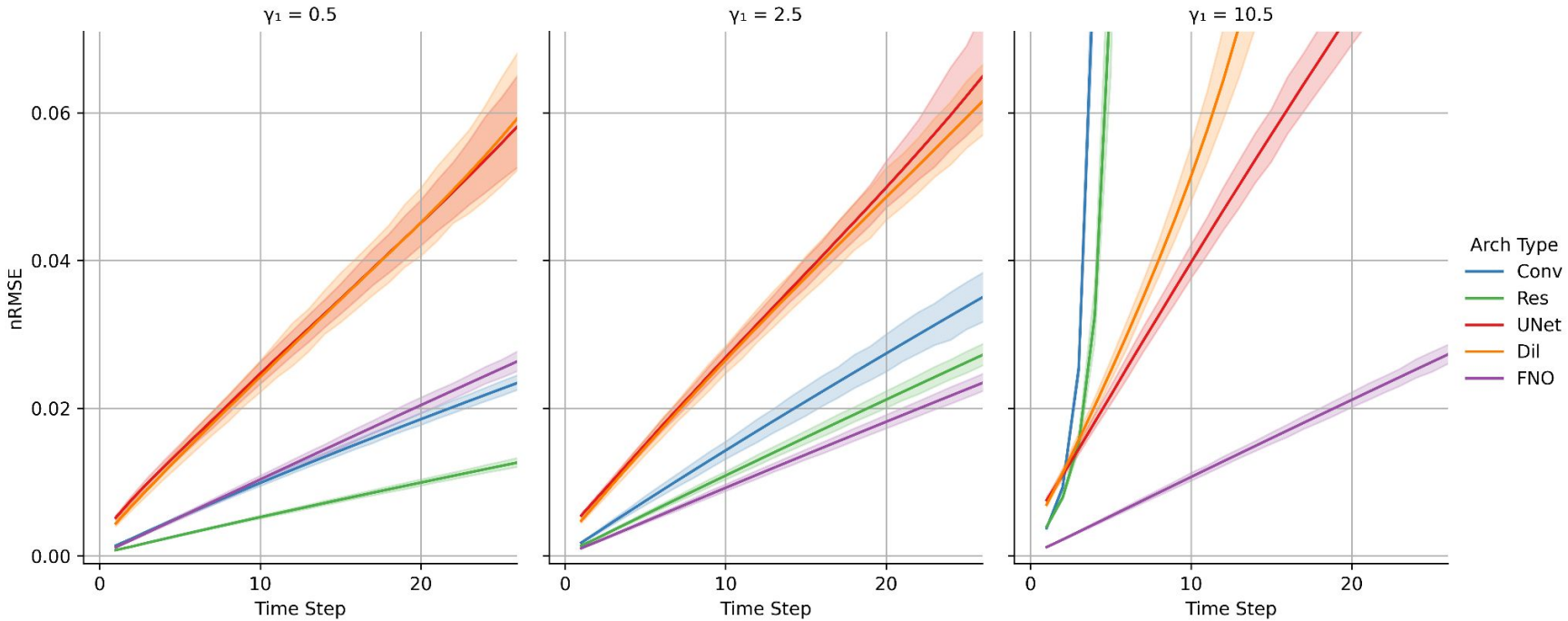
Low-Speed Advection Emulation



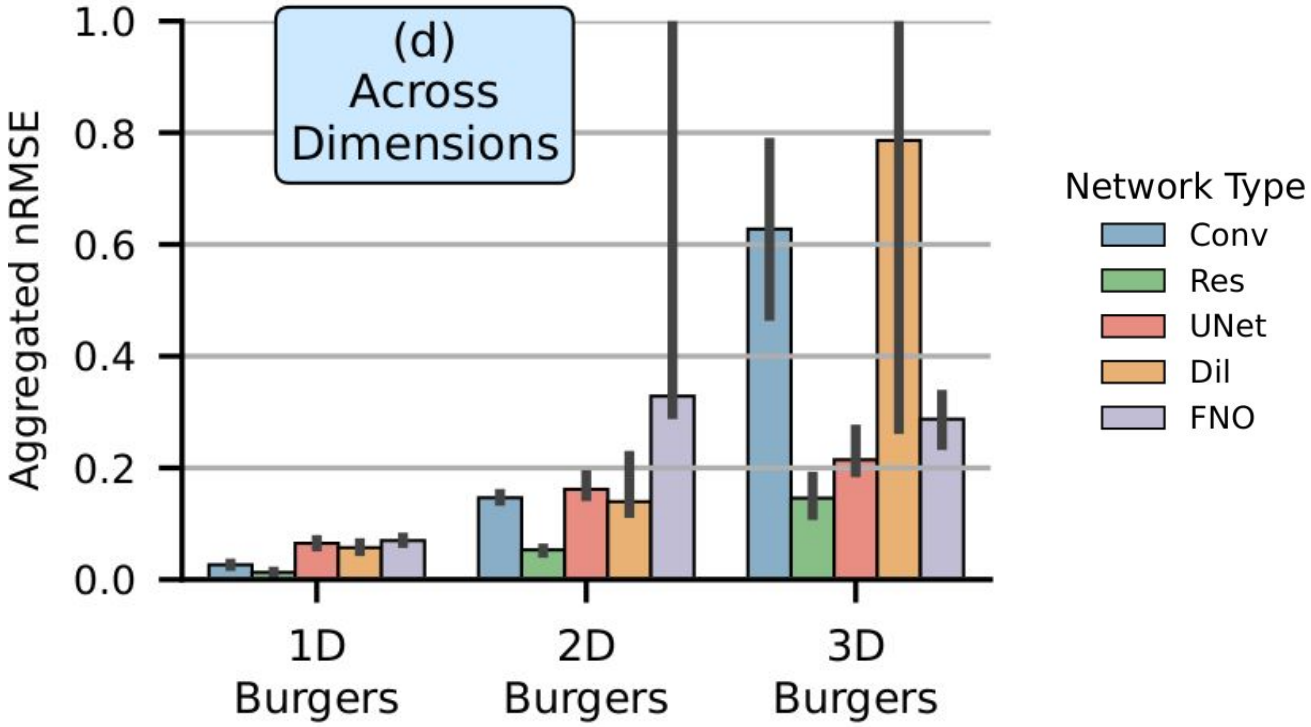
Advection Emulation at different speeds

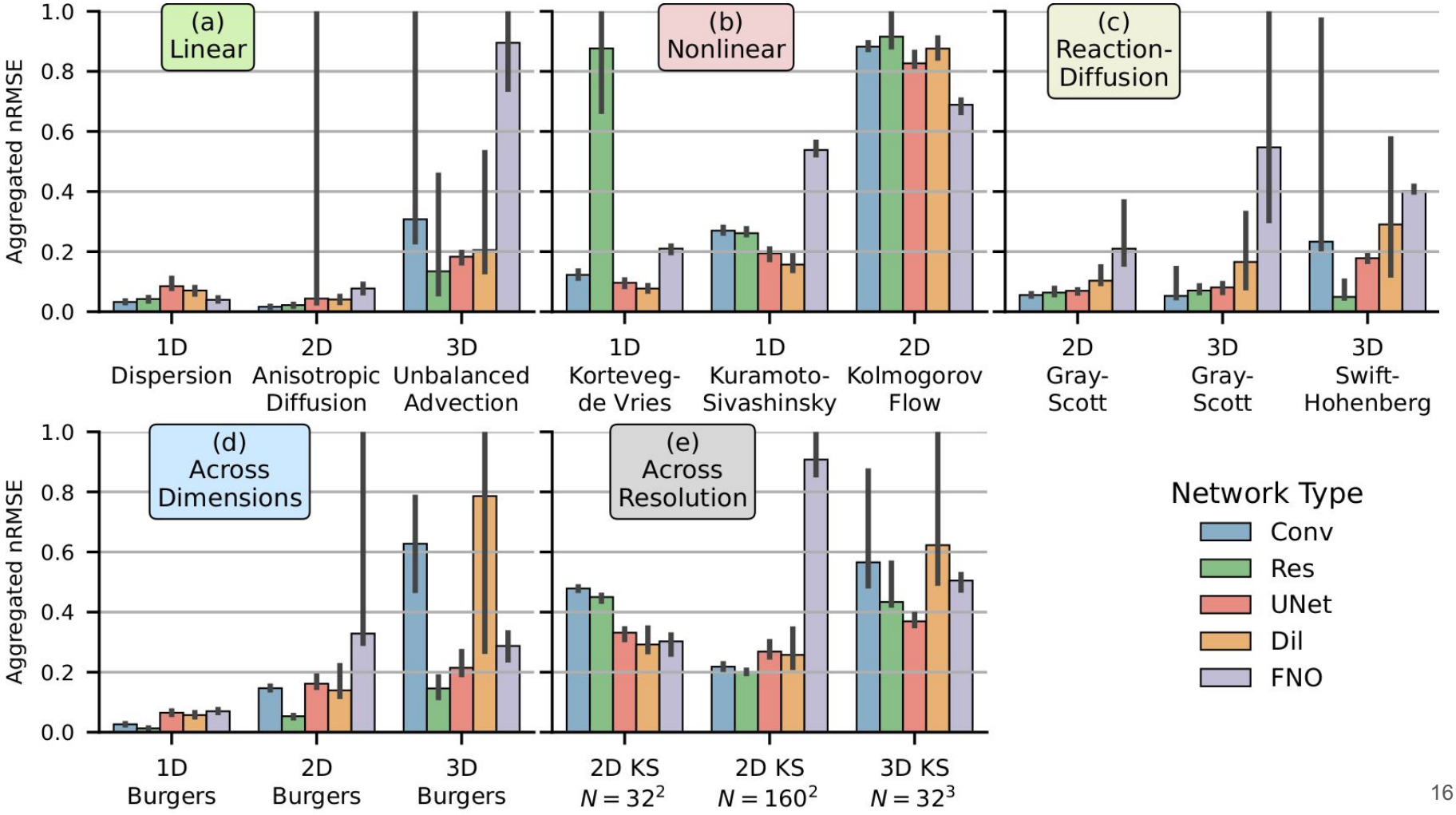


Advection Emulation at different speeds w/ other Archs

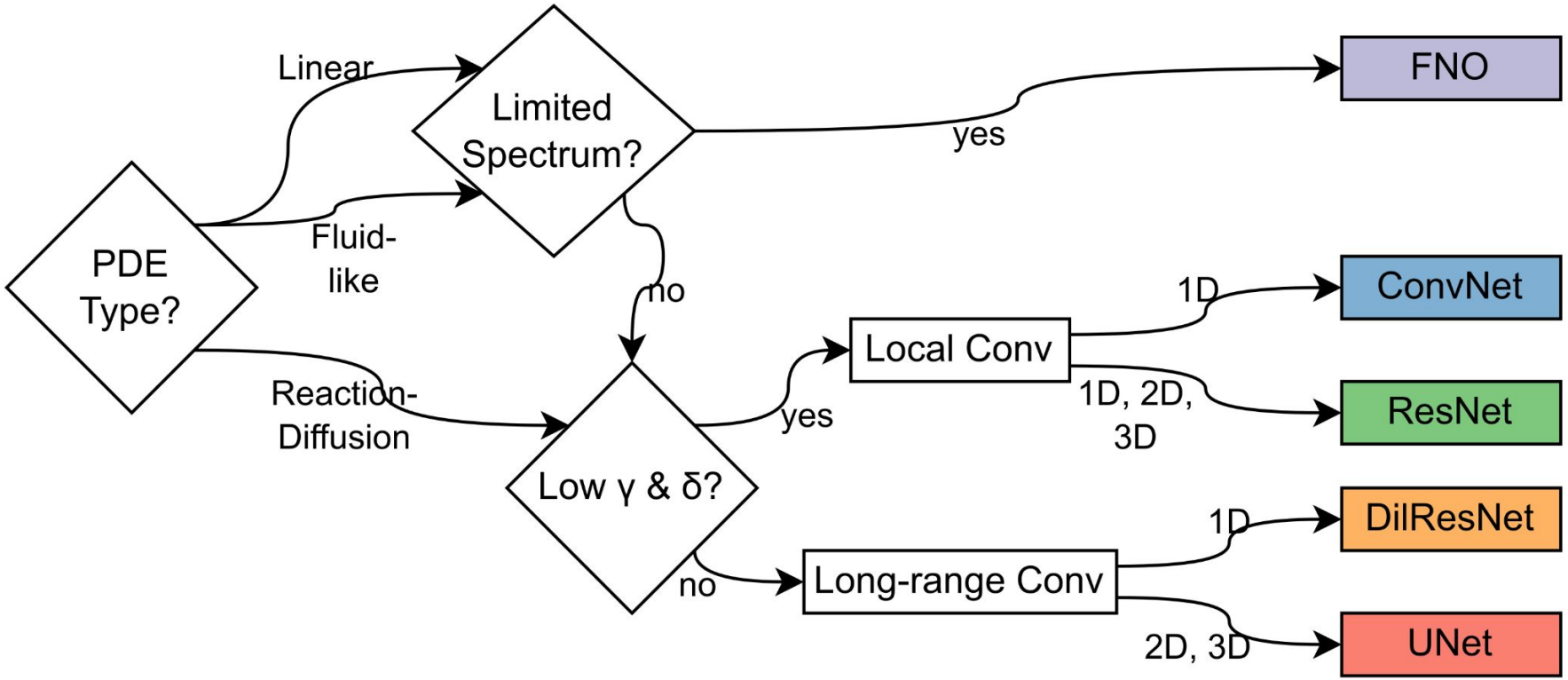


Burgers Emulation Across Dimensionality




(a) Linear
(b) Nonlinear
(c) Reaction-Diffusion
(d) Across Dimensions
(e) Across Resolution
Aggregated nRMSE
1.0
0.8
0.6
0.4
0.2
0.0
1D Dispersion
2D Anisotropic Diffusion
3D Unbalanced Advection
1D Korteveg-de Vries
1D Kuramoto-Sivashinsky
2D Kolmogorov Flow
2D Gray-Scott
3D Gray-Scott
3D Swift-Hohenberg
1D Burgers
2D Burgers
3D Burgers
2D KS $N = 32^2$
2D KS $N = 160^2$
3D KS $N = 32^3$
Network Type
Conv
Res
UNet
Dil
FNO

Architecture Decision Tree



APEBench: A Benchmark for Autoregressive Neural Emulators of PDEs

Felix Koehler
Technical University of Munich
Munich Center for Machine Learning
f.koehler@tum.de

Simon Niedermayr
Technical University of Munich
simon.niedermayr@tum.de

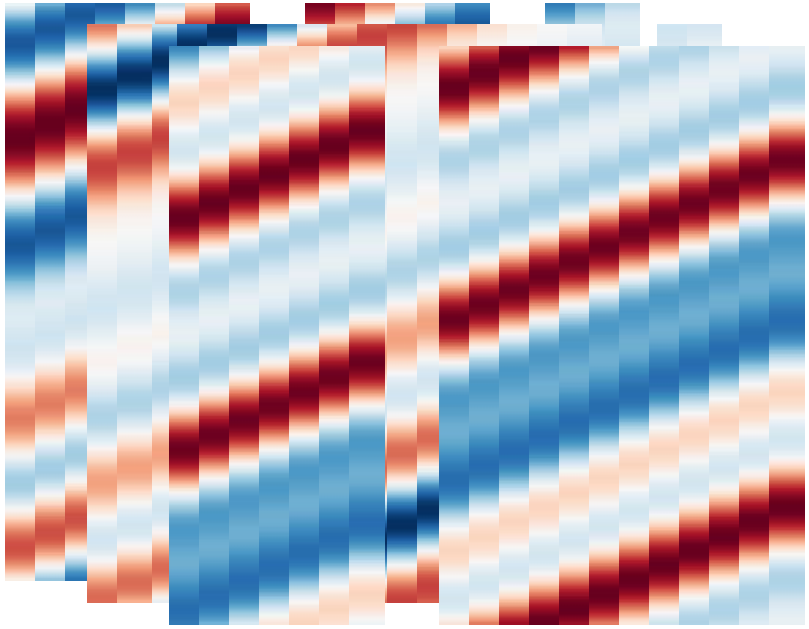
Rüdiger Westermann
Technical University of Munich
westermann@tum.de

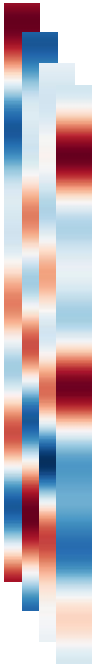
Nils Thuerey
Technical University of Munich
nils.thuerey@tum.de

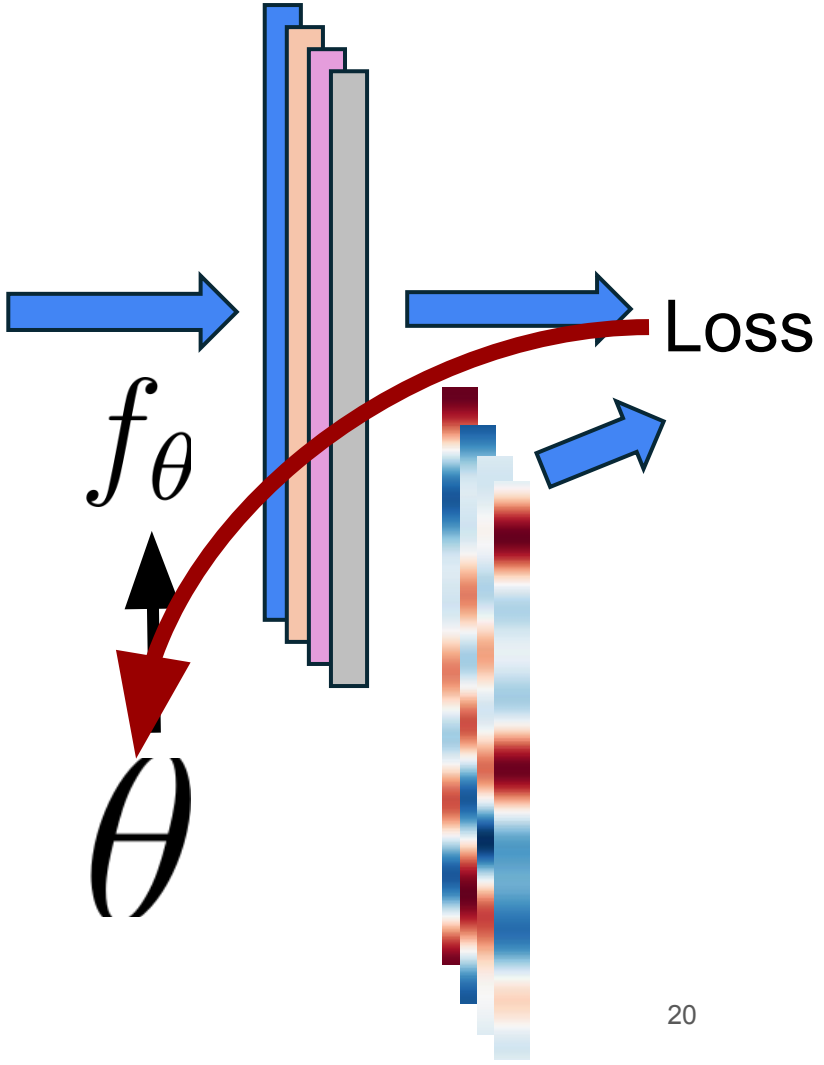
```
pip install apebench
```

tum-pbs.github.io/apebench

Let’s dive deeper





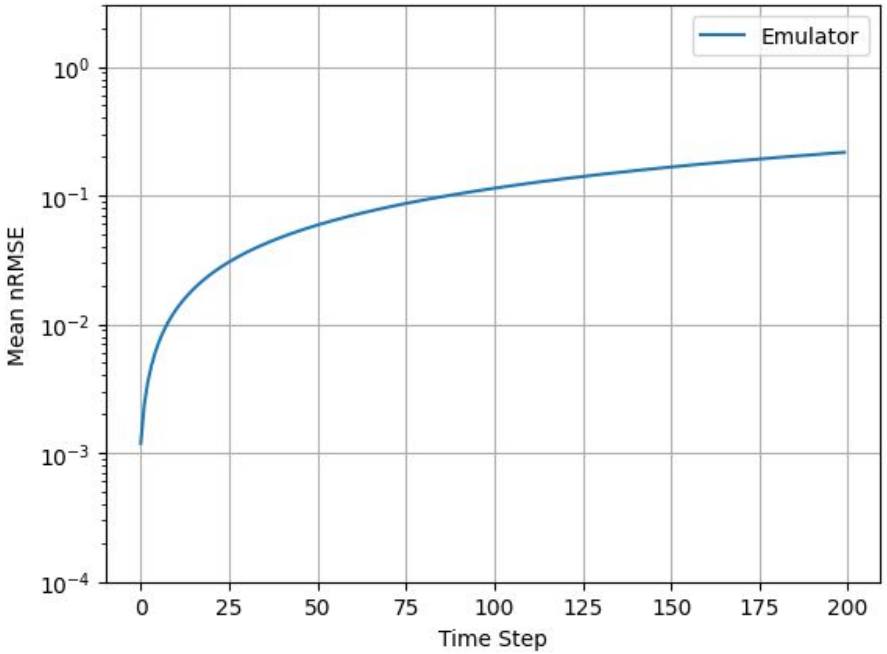


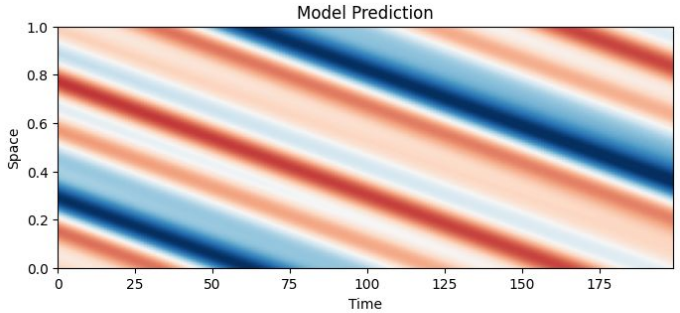
$$f_\theta(u_h) = w_\theta \star u_h$$

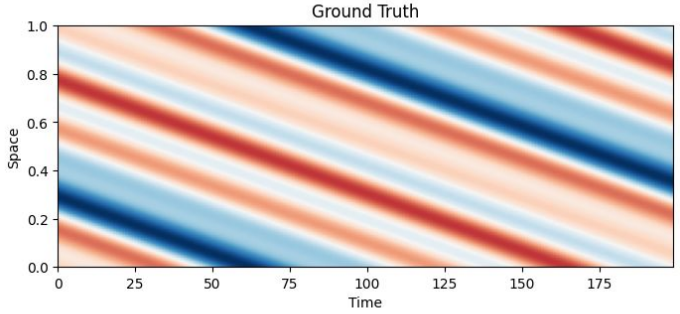
$$w_\theta = [\theta_{\text{center}}, \theta_{\text{right}}]^T \in \mathbb{R}^2$$

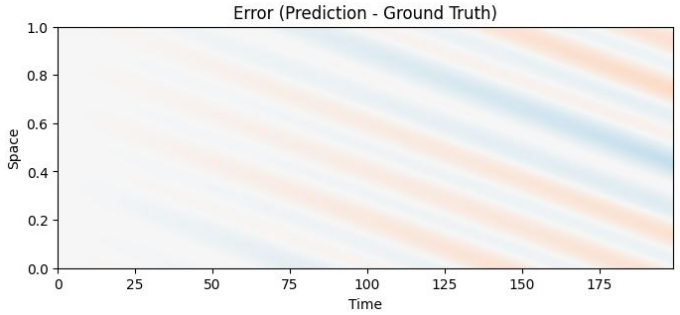
Working decently at

$$w_\theta^* = [0.2504286, 0.7508612]$$



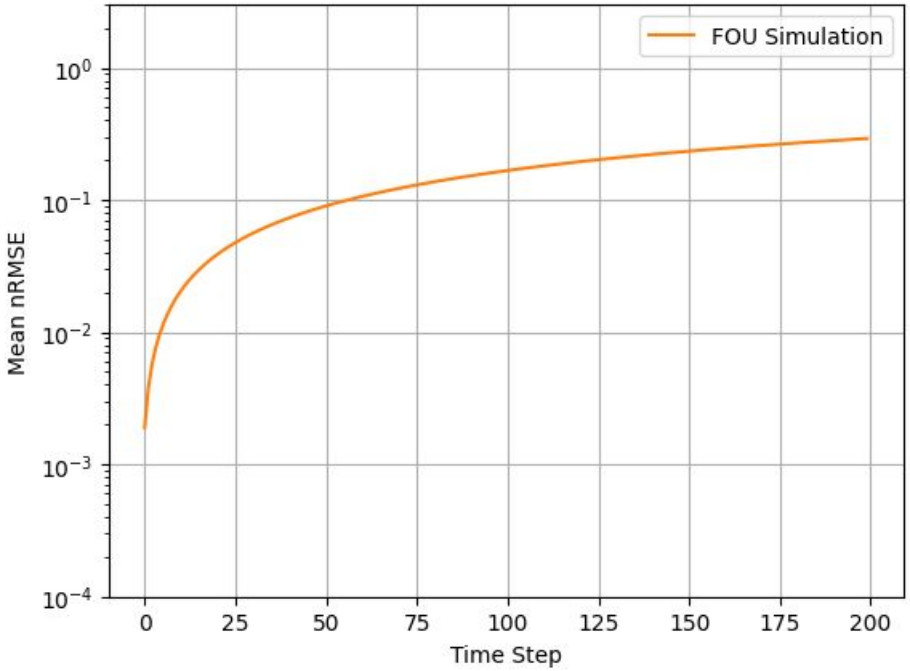


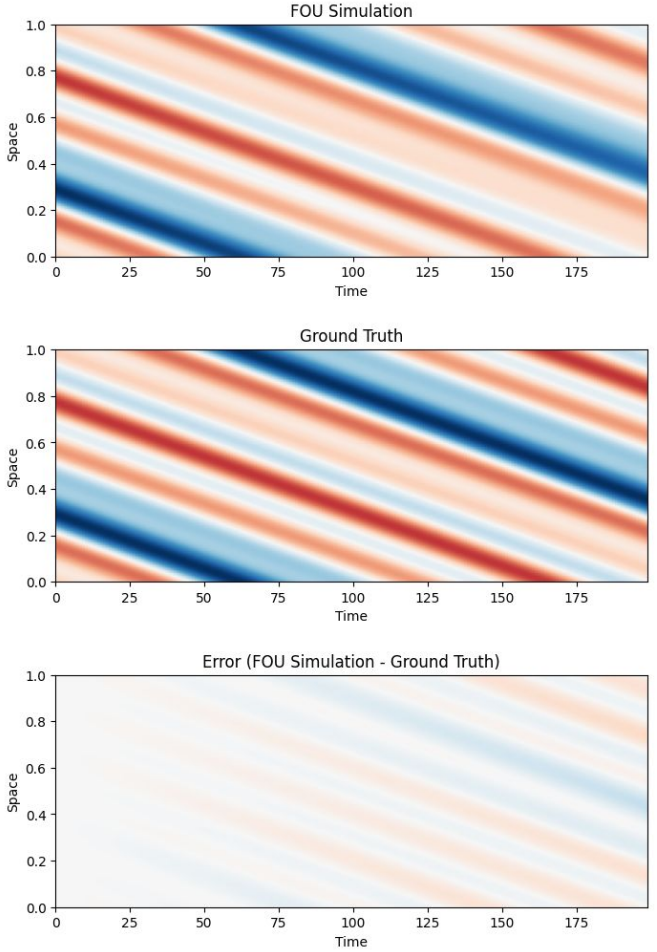




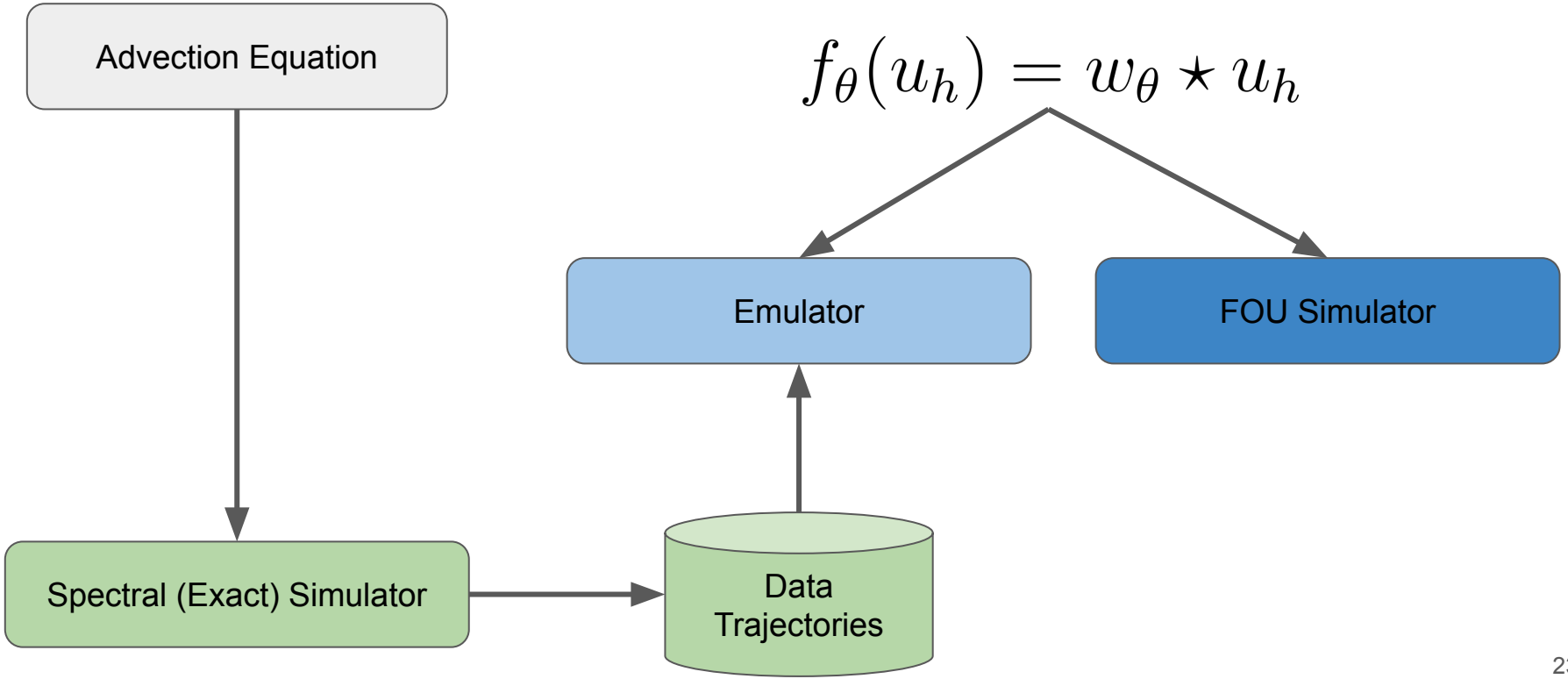
What about another numerical solver

$$w_{\mathrm{FOU}} = [0.25, 0.75]$$

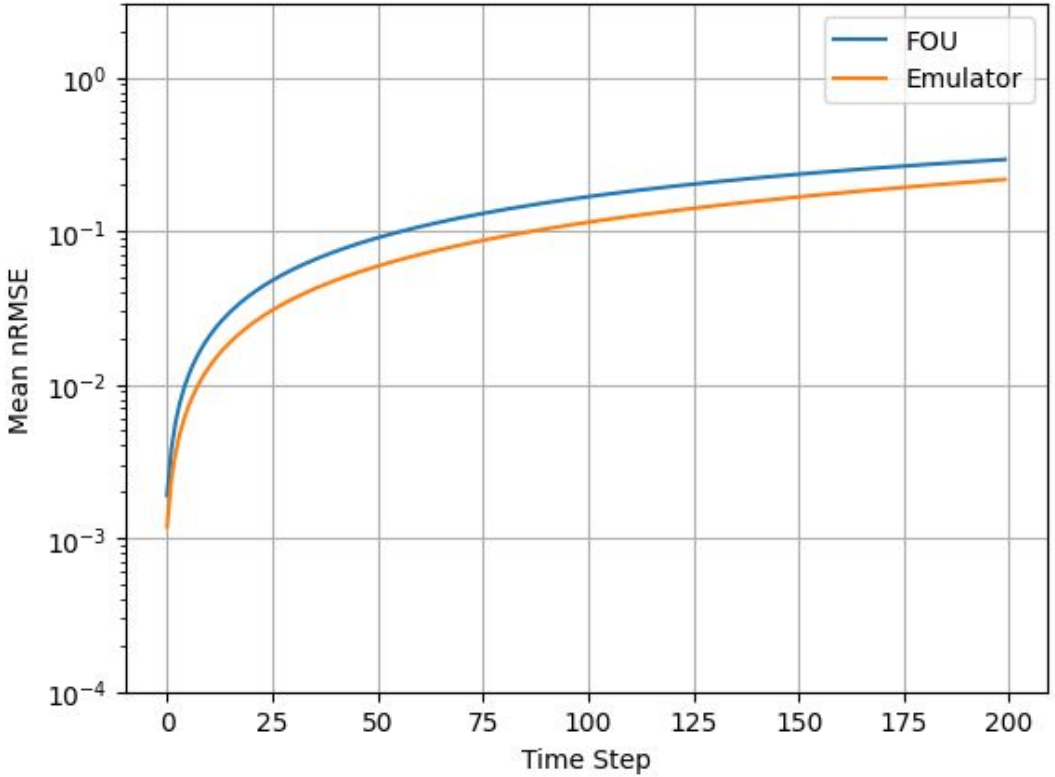




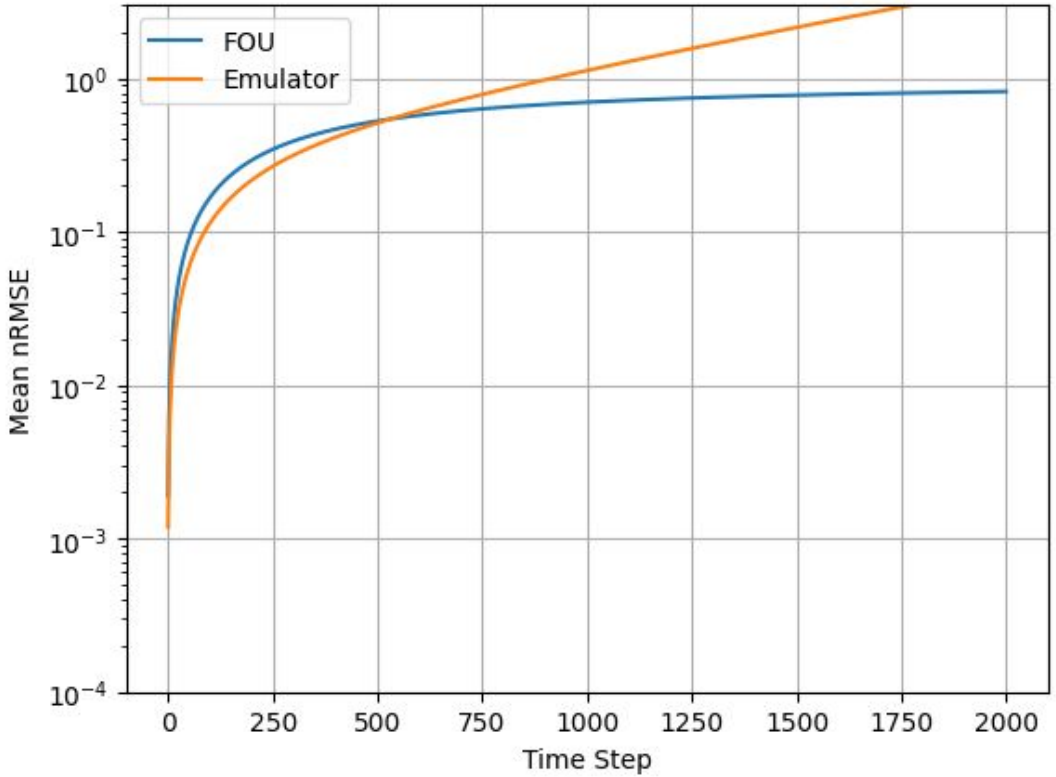
The three timesteppers in more context



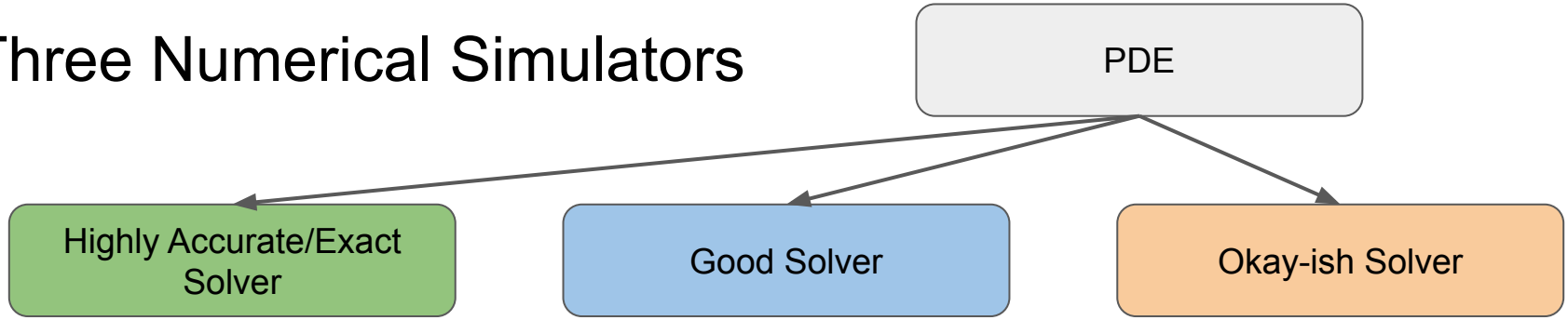
Emulator better under the same functional form?



What's the catch? Emulator is not consistent!



Three Numerical Simulators



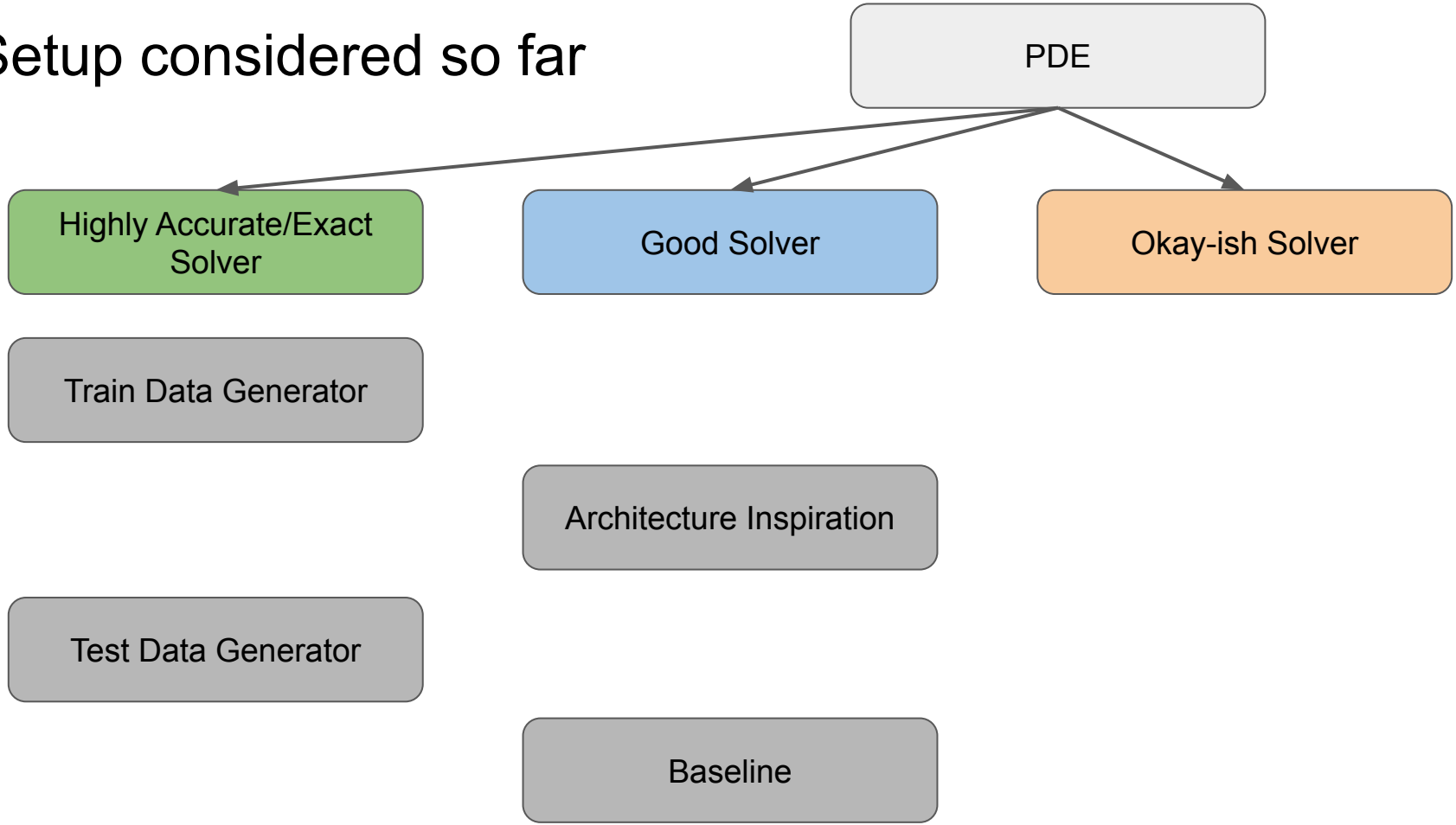
Train Data Generator

Architecture Inspiration

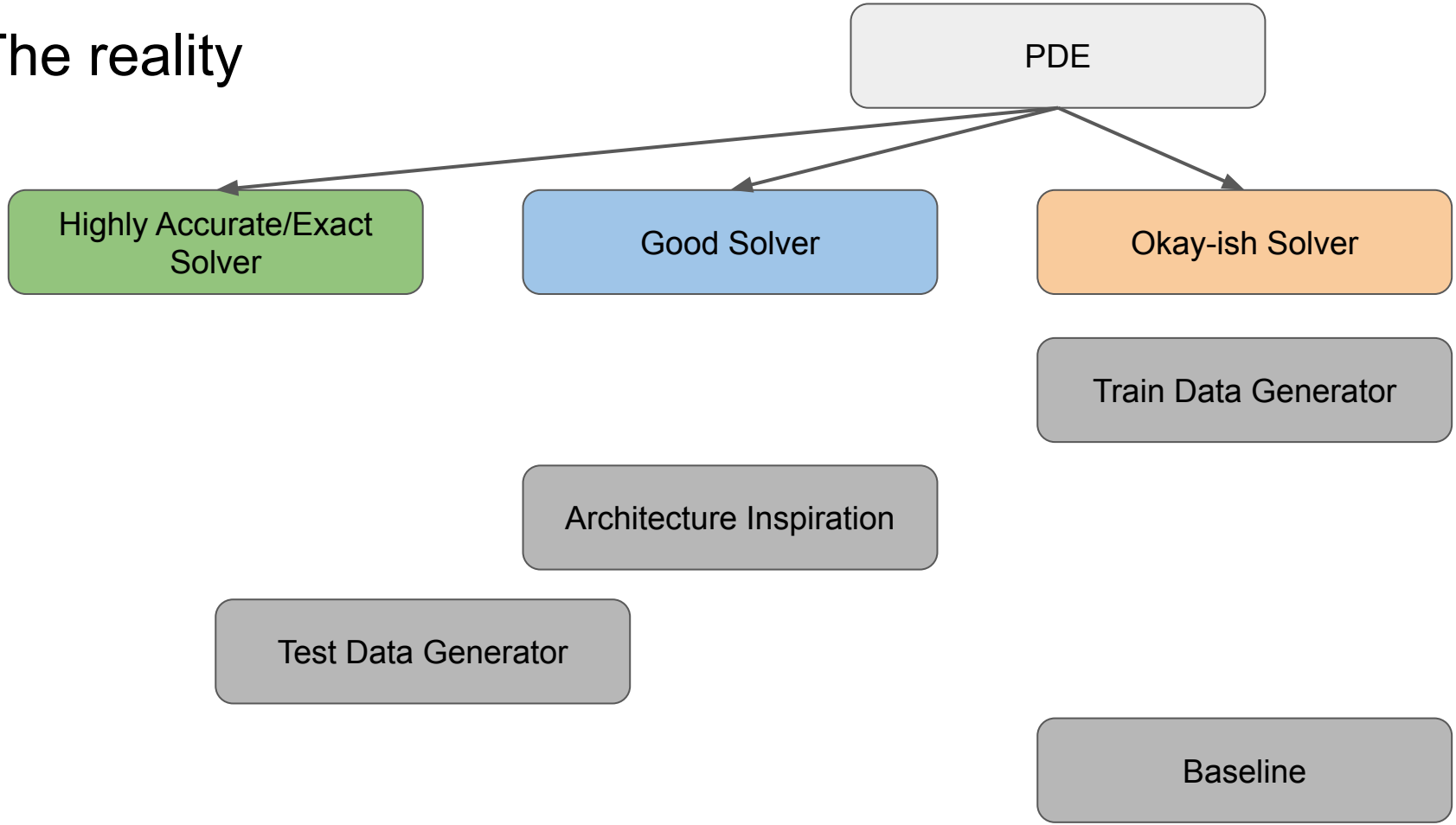
Test Data Generator

Baseline

Setup considered so far



The reality

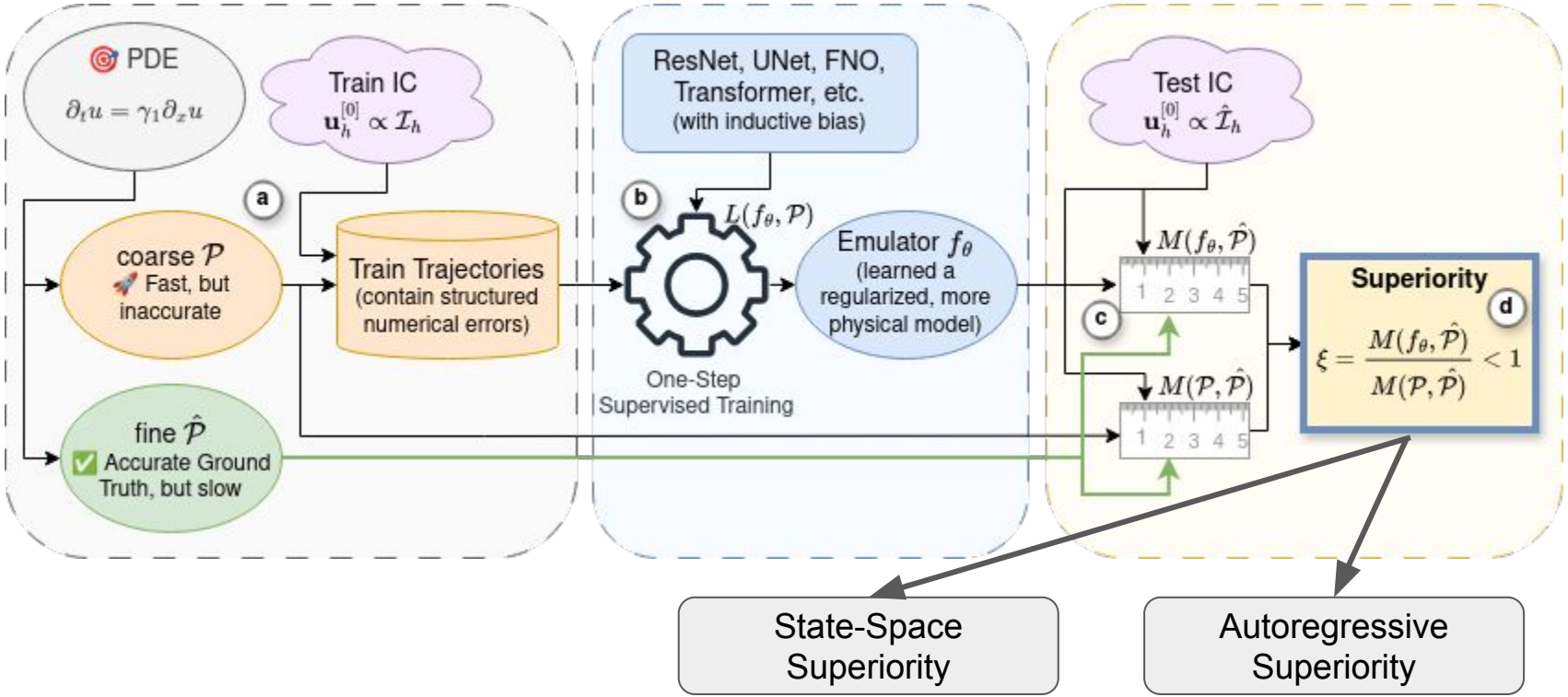

PDE
Highly Accurate/Exact Solver
Good Solver
Okay-ish Solver
Train Data Generator
Architecture Inspiration
Test Data Generator
Baseline

Understanding Difference Between Training and Testing

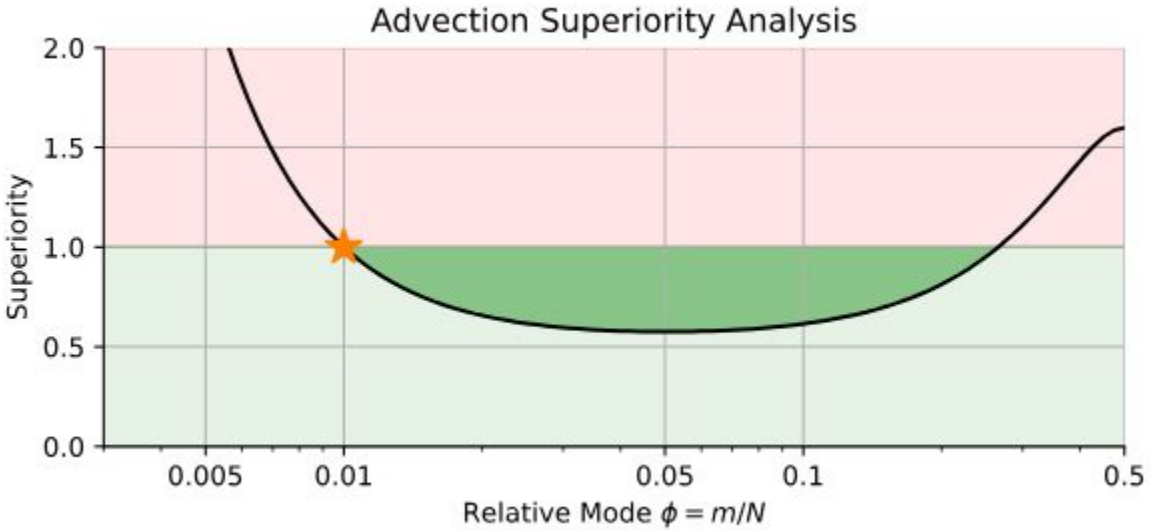
$$L(\theta) = \mathbb{E}_{\{\mathbf{u}_h^{[t]}, \mathbf{u}_h^{[t+1]}\} \sim \mathcal{T}_{\mathcal{P}_h}(\mathbf{u}_h^{[0]}),\, \mathbf{u}_h^{[0]} \sim \mathcal{I}_h} \left[\zeta \left(f_\theta(\mathbf{u}_h^{[t]}), \mathbf{u}_h^{[t+1]} \right) \right]$$

$$e^{[t]} = \mathbb{E}_{\mathbf{u}_h^{[0]} \sim \tilde{\mathcal{I}}_h} \left[\tilde{\zeta} \left(f_\theta^t(\mathbf{u}_h^{[0]}), \tilde{\mathcal{P}}_h^t(\mathbf{u}_h^{[0]}) \right) \right]$$

The missing piece: Where is the training data from?



Theoretically Provable -> Forward Superiority



Let’s loop back to what we learned

- Spectral-Solver (Perfect) Data -> Functional Form of Explicit Method -> Better than the Explicit Method
- FOU (good) Data -> Functional Form of Explicit Method -> Recover Explicit Method
- IFOU (mediocre) Data -> Functional Form of Explicit Method -> Better than Implicit Method: **Learned Emulator can be Superior!**

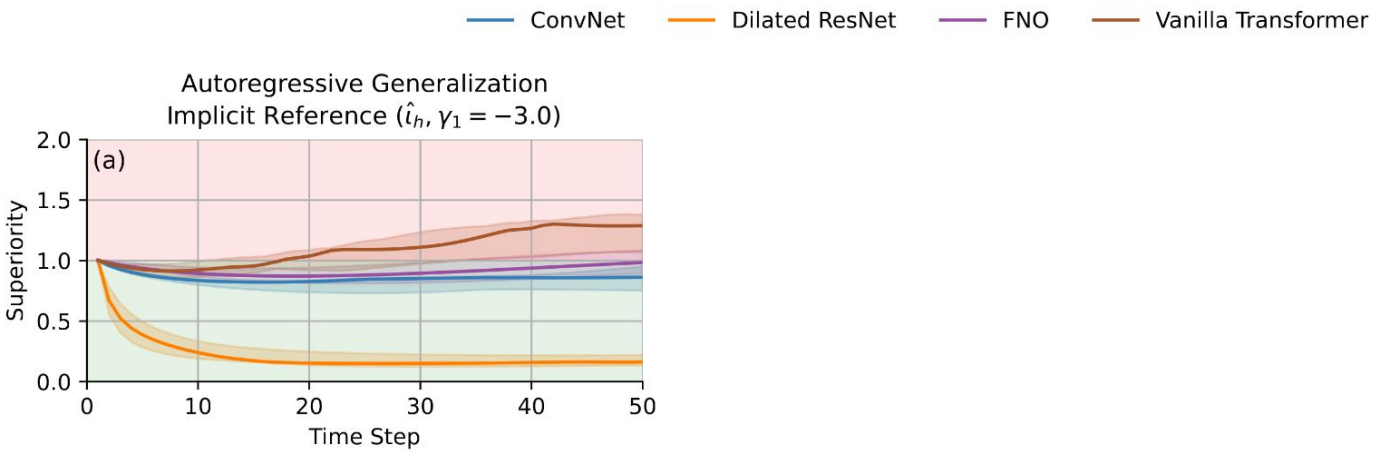
Let’s go Nonlinear

Analytic Solution

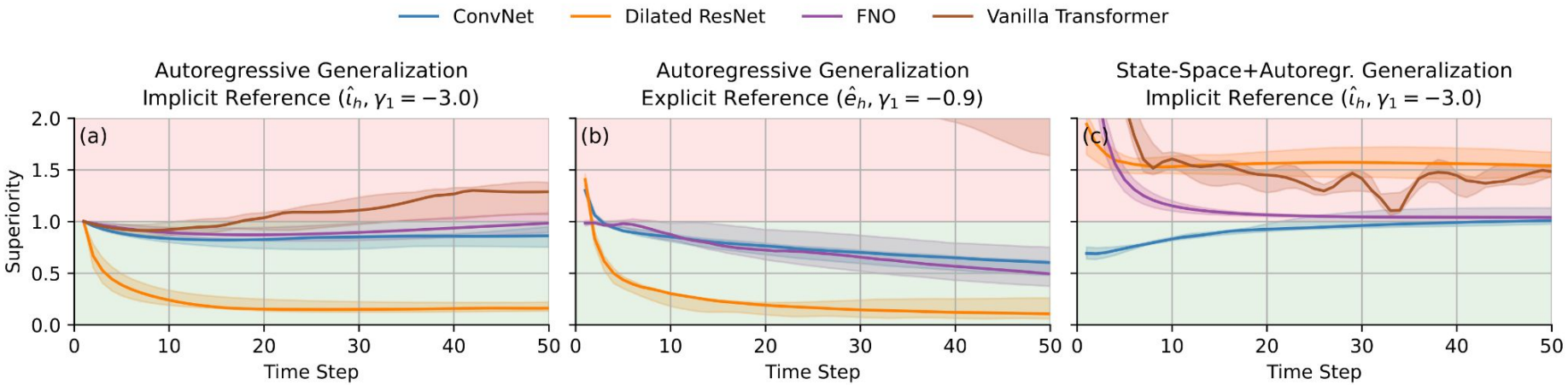
Neural Emulator
(trained on coarse simulator ➡)

Coarse Simulator

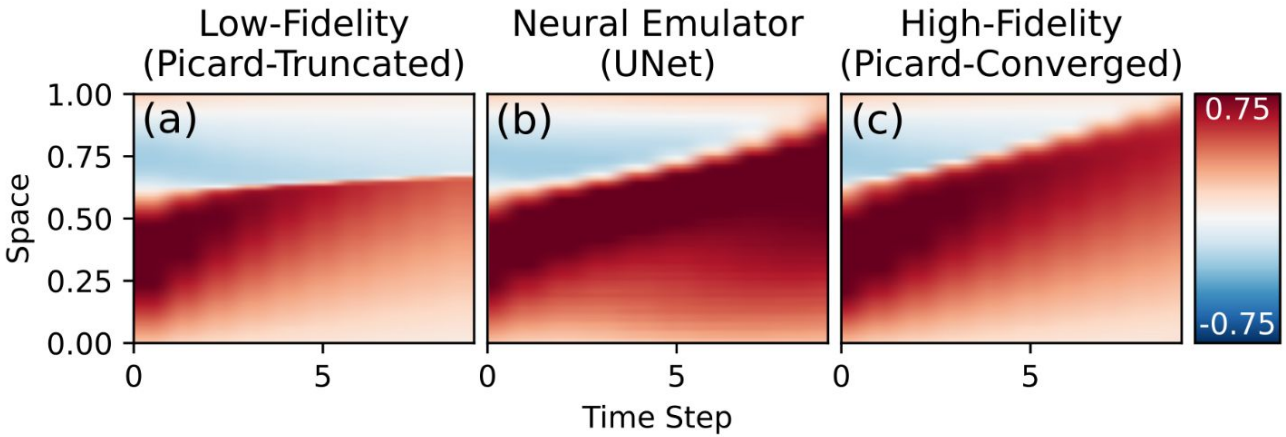
More Architectures On Advection

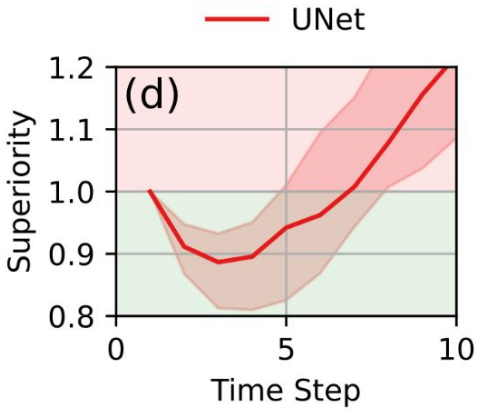


More Architectures On Advection



Interesting Example: Burgers

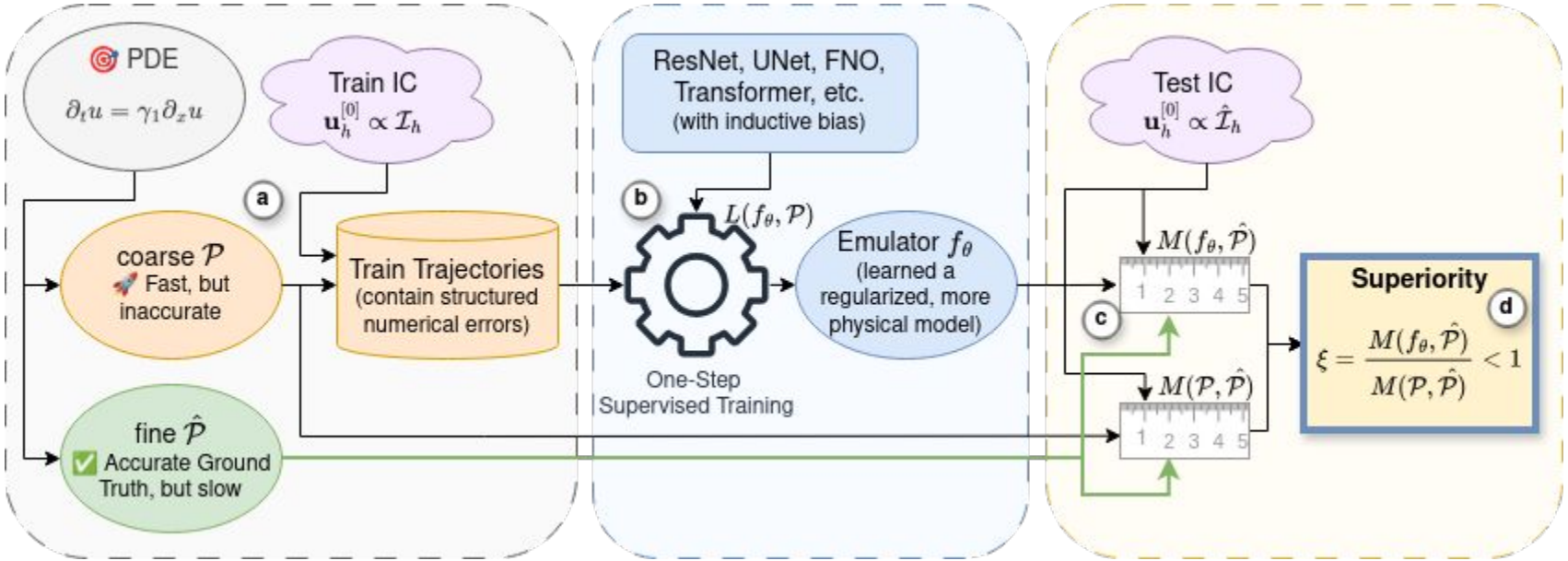




More Architectures on Burgers

	Superiority $\xi^{[t]}$ at Time Step									
Arch/Time Step	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
ConvNet	1.00	0.97	0.87	0.82	0.80	0.80	0.80	0.81	0.81	0.81
Dilated ResNet	1.00	0.91	0.79	0.73	0.69	0.66	0.65	0.67	0.66	0.65
FNO	1.00	1.01	0.91	0.87	0.86	0.89	0.93	1.00	1.08	1.12
Transformer	1.00	1.06	1.13	1.24	1.35	1.45	1.52	1.58	1.63	1.68
UNet	1.00	0.90	0.86	0.88	0.91	0.96	1.00	1.04	1.07	1.11

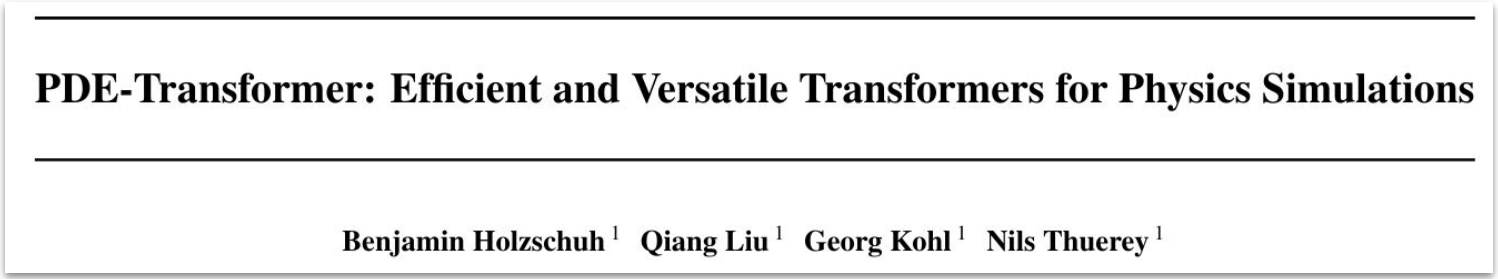
What does Emulator Superiority mean in practice?



What’s Next in this field?

- Emulators for Larger 3D problems, especially on unstructured grids
- Foundation Models for a wider range of Dynamics
    - Here it seems that we need lower specific inductive bias to be applicable to all dynamics

For all this, it seems to crystallize that transformer models are best at scaling



Summary and Take-Aways

- Neural Networks can be trained to approximate time-dependent PDE problems autoregressively 🧠:
  - For a specified time horizon 🕓, they can achieve decent accuracy ✅
  - But, we do have little guarantees on them 🤔
  - For non-trivial simulations, they can speed up the solution process, especially in a more holistic setting ⏱️
- Inductive biases are important for model performance 🦿:
  - Chose the architecture that matches the best numerical scheme for the specific PDE 🤝
  - Sometimes, it is possible to have the inductive bias outlearn structured numerical errors from imperfect simulation data 🔍

Thanks for listening!

NeurIPS 2024

APEBench: A Benchmark for Autoregressive Neural Emulators of PDEs

Felix Koehler
Technical University of Munich
Munich Center for Machine Learning
f.koehler@tum.de

Simon Niedermayr
Technical University of Munich
simon.niedermayr@tum.de

Rüdiger Westermann
Technical University of Munich
westermann@tum.de

Nils Thuerey
Technical University of Munich
nils.thuerey@tum.de


```
pip install apebench
```

tum-pbs.github.io/apebench

NeurIPS 2025

Neural Emulator Superiority: When Machine Learning for PDEs Surpasses its Training Data

Felix Koehler & **Nils Thuerey**
Technical University of Munich
Munich Center for Machine Learning (MCML)
{f.koehler,nils.thuerey}@tum.de