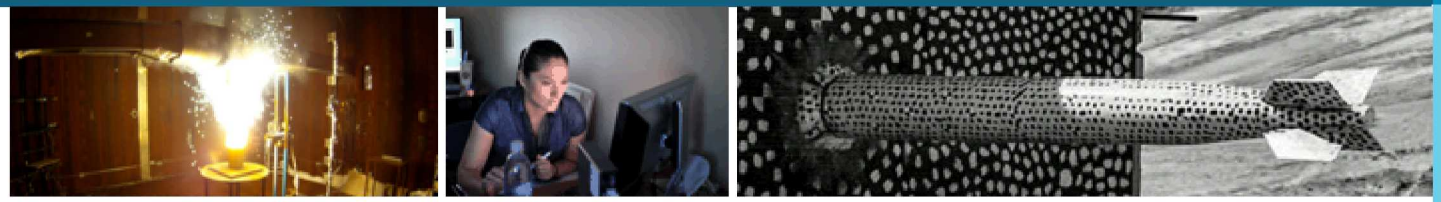
SAND2020-7096C

Enabling Nonlinear Manifold Projection Reduced-Order Models by Extending Convolutional Neural Networks to Unstructured Data



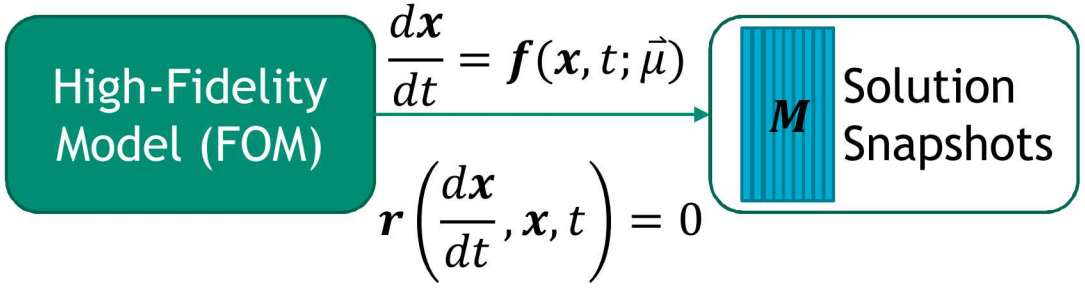
SHTC 2020

John Tencer and Kevin Potter

Supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525.

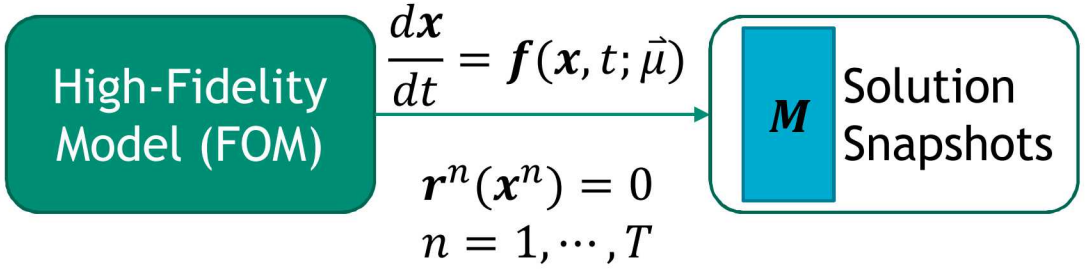
Classical Model Reduction Overview

Generate Solution Snapshots

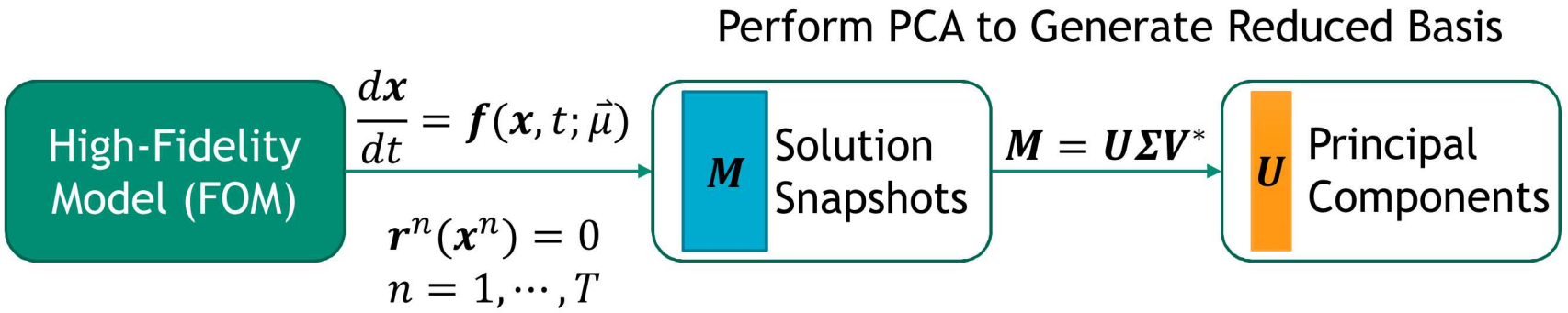


Classical Model Reduction Overview

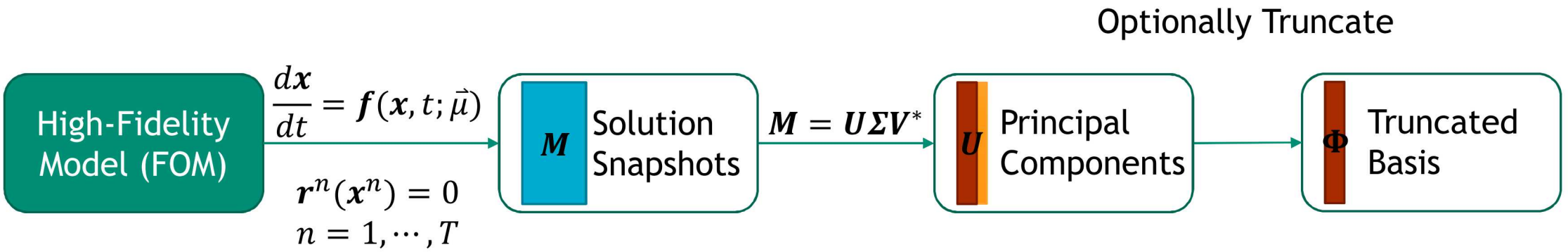
Generate Solution Snapshots



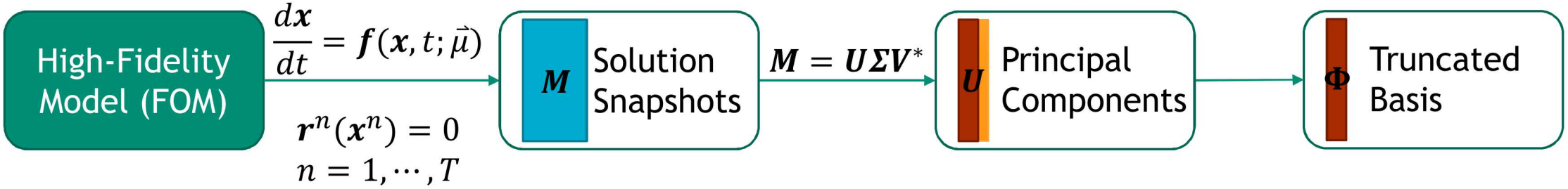
Classical Model Reduction Overview



Classical Model Reduction Overview

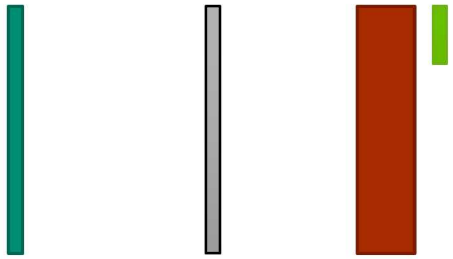


Classical Model Reduction Overview

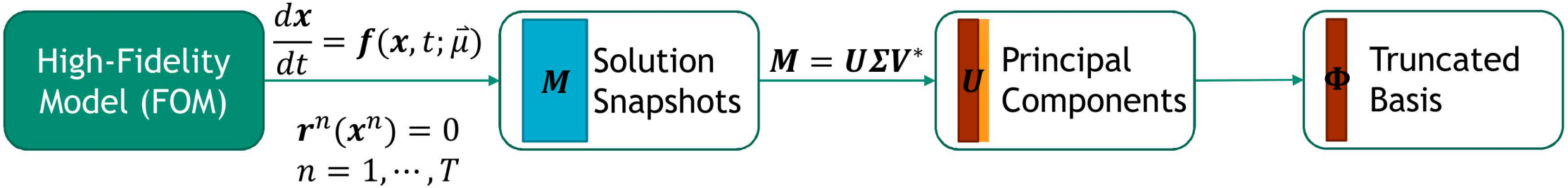


Approximate FOM State

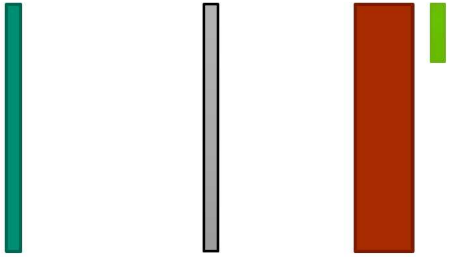
$$\boldsymbol{x}(t; \vec{\mu}) \approx \tilde{\boldsymbol{x}}(t; \vec{\mu}) = \boldsymbol{\Phi}\hat{\boldsymbol{x}}(t; \vec{\mu})$$



Classical Model Reduction Overview



$$\boldsymbol{x}(t;\vec{\mu}) \approx \tilde{\boldsymbol{x}}(t;\vec{\mu}) = \boldsymbol{\Phi}\hat{\boldsymbol{x}}(t;\vec{\mu})$$



Project System Dynamics

Galerkin Projection

$$\boldsymbol{\Phi}^T \boldsymbol{r}^n(\boldsymbol{\Phi}\hat{\boldsymbol{x}}^n) = 0$$
$$n = 1, \cdots, T$$

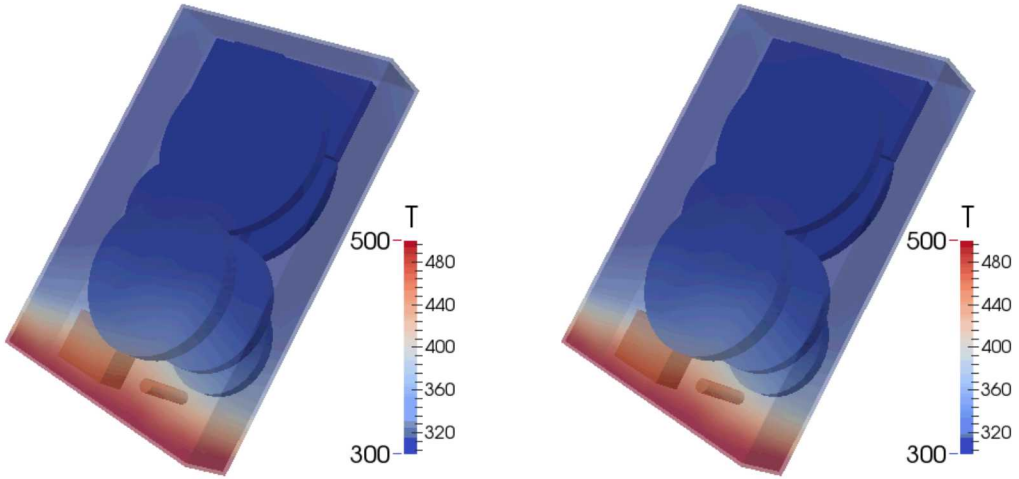
LSPG Projection

$$\boldsymbol{\Phi}\hat{\boldsymbol{x}}^n = \underset{v \in \mathrm{range}(\boldsymbol{\Phi})}{\mathrm{argmin}} \|\boldsymbol{r}^n(v)\|_2$$
$$n = 1, \cdots, T$$

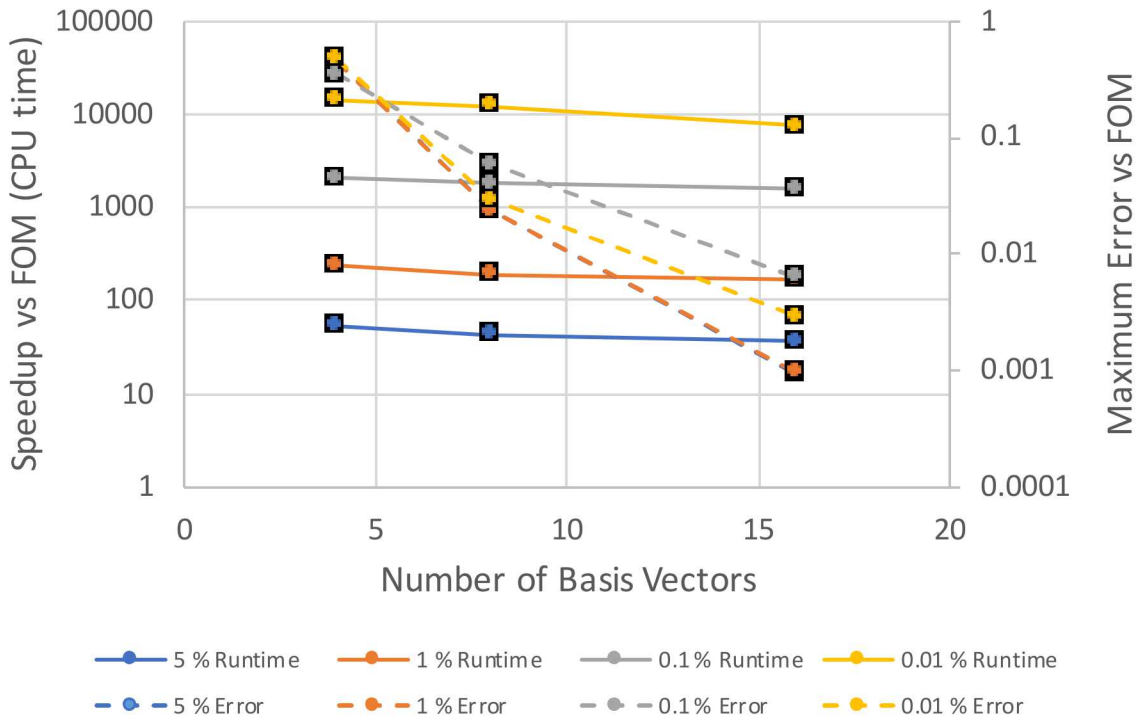
$\boldsymbol{\Phi}$ is a *linear* trial subspace

Highly effective for diffusion dominated problems

- ROM methods are highly effective at reducing cost in (nonlinear) conduction problems.
- Easily applied to complex geometries.
- Hyper-reduction promises further performance benefits.

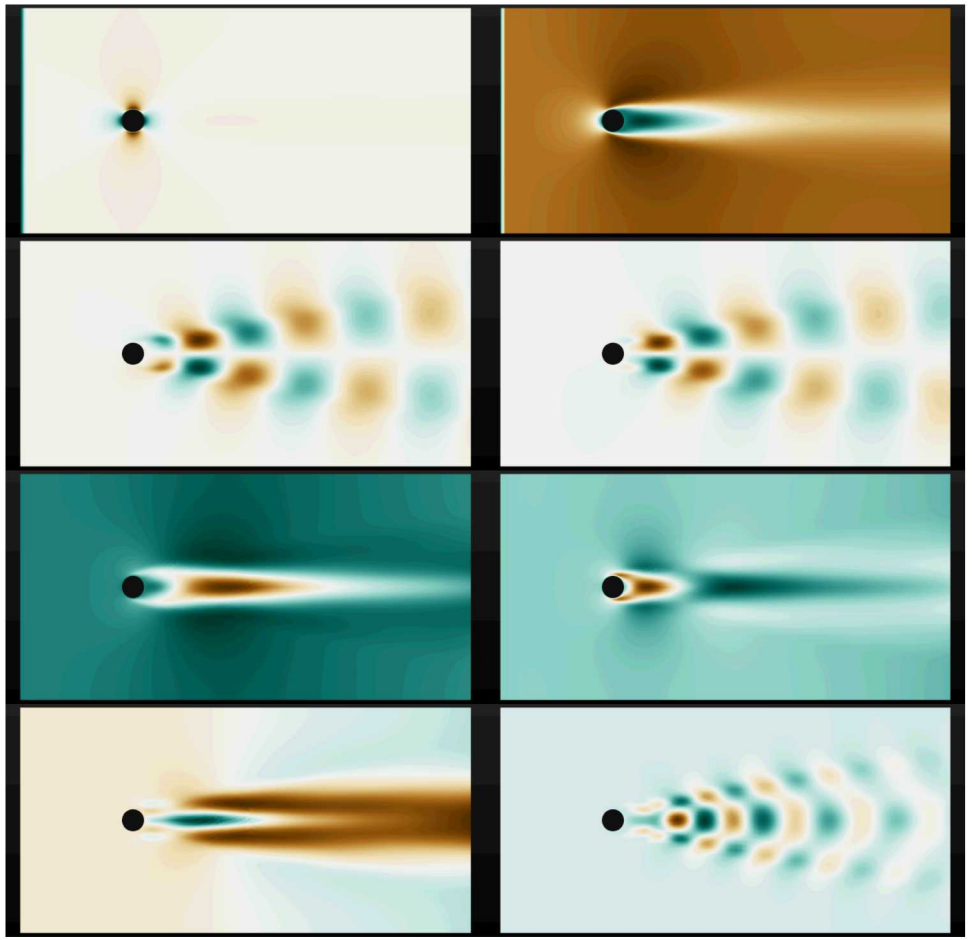


Comparison between FOM (left) and ROM (right) with only 4 modes retained.



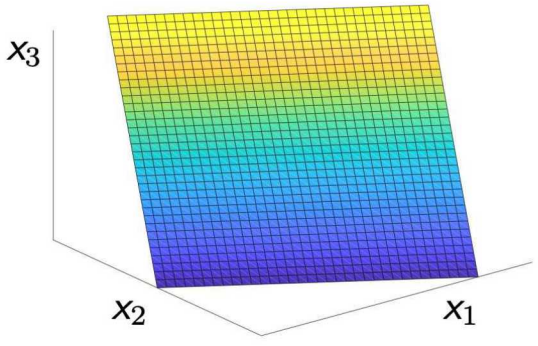
Kolmogorov-width limitation

- Linear trial subspace sufficient for problems where the singular values of $\boldsymbol{M}$ decay rapidly (i.e. diffusion dominated problems).
- For other problems the singular values decay slowly and many of the columns from $\boldsymbol{U}$ are required to be retained in $\boldsymbol{\Phi}$ to achieve accurate solutions.
- ROM computational cost is closely tied to the number of modes retained
  - Newton-Raphson iteration costs scale quadratically with the trial subspace dimension



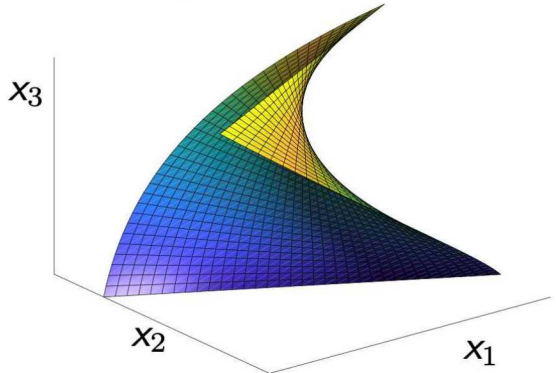
Linear trial subspace

$\text{range}(\boldsymbol{\Phi}) := \{\boldsymbol{\Phi}\hat{\mathbf{x}} \,|\, \hat{\mathbf{x}} \in \mathbb{R}^p\}$

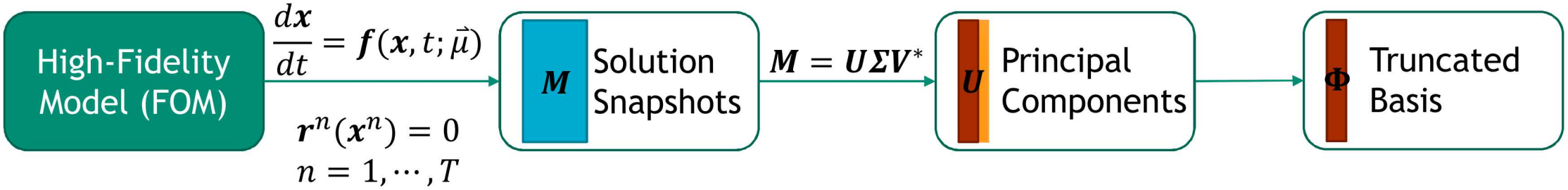


Nonlinear trial manifold

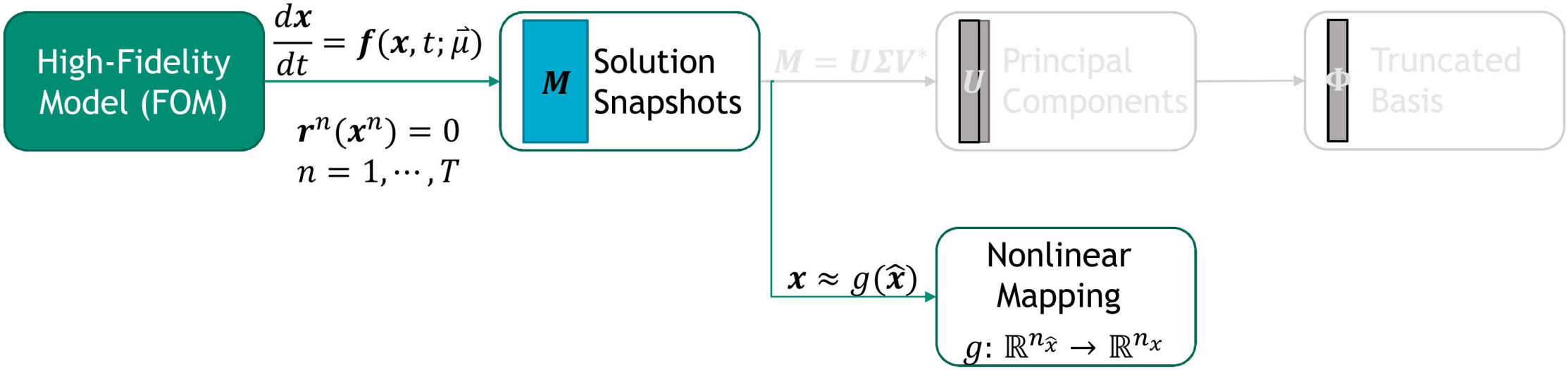
$\mathcal{S} := \{\mathbf{g}(\hat{\mathbf{x}}) \,|\, \hat{\mathbf{x}} \in \mathbb{R}^p\}$



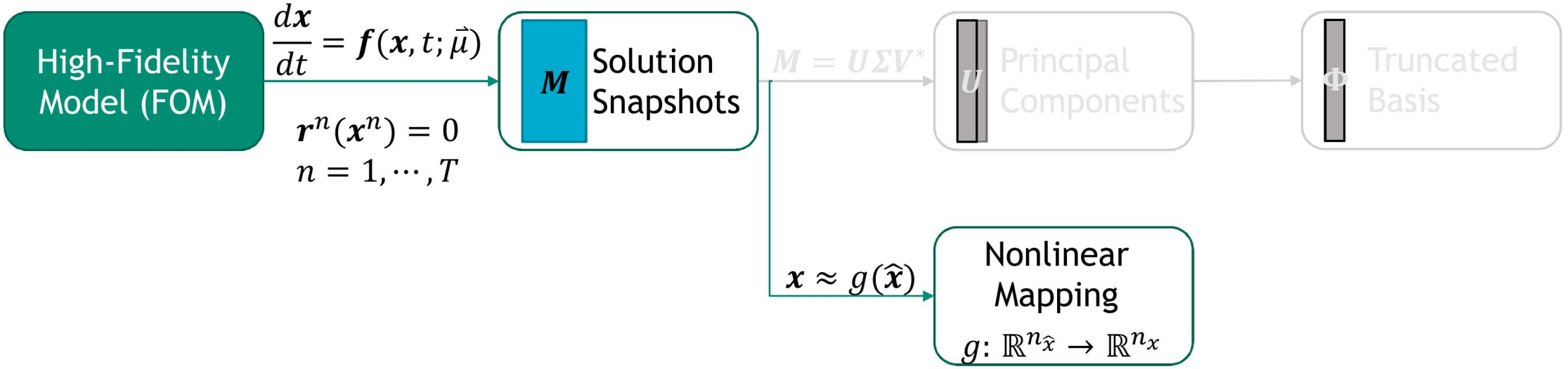
Nonlinear Model Reduction Overview



Nonlinear Model Reduction Overview

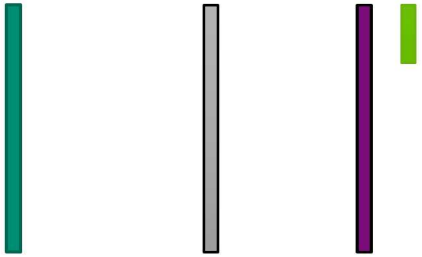


Nonlinear Model Reduction Overview



Approximate FOM State

$$\boldsymbol{x}(t; \vec{\mu}) \approx \tilde{\boldsymbol{x}}(t; \vec{\mu}) = g(\hat{\boldsymbol{x}}(t; \vec{\mu}), t; \vec{\mu})$$



Project System Dynamics

Galerkin Projection

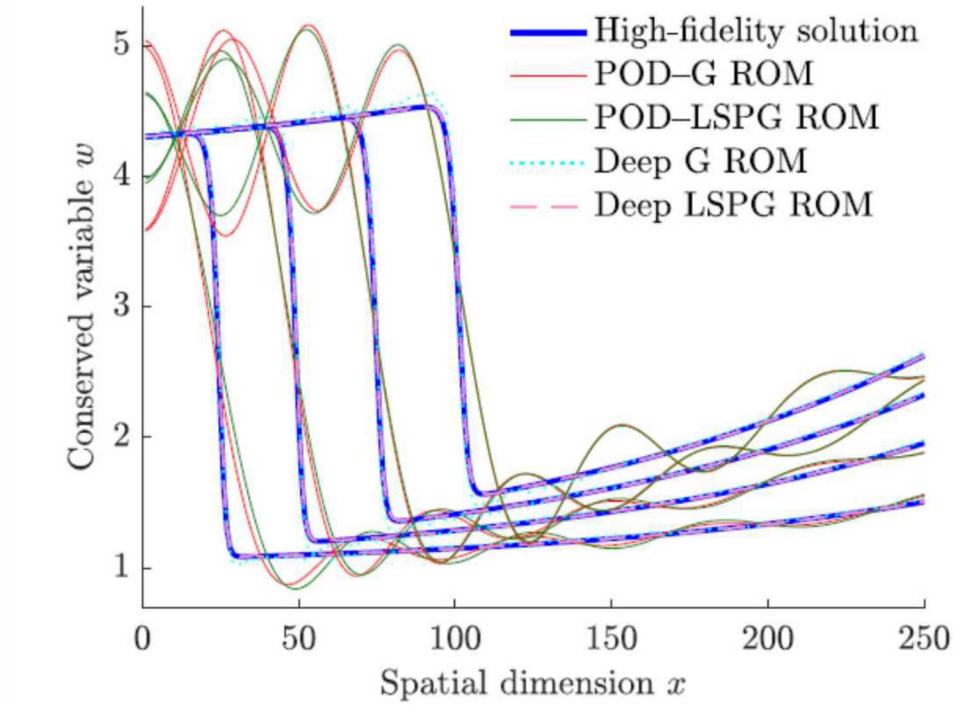
$$\frac{\partial \boldsymbol{x}}{\partial t} = \underset{v \in \mathbb{R}^{n_{\hat{x}}}}{\operatorname{argmin}} \| \boldsymbol{r}(\boldsymbol{J}(\hat{\boldsymbol{x}})v, \boldsymbol{g}(\hat{\boldsymbol{x}}), t; \vec{\mu}) \|_2$$

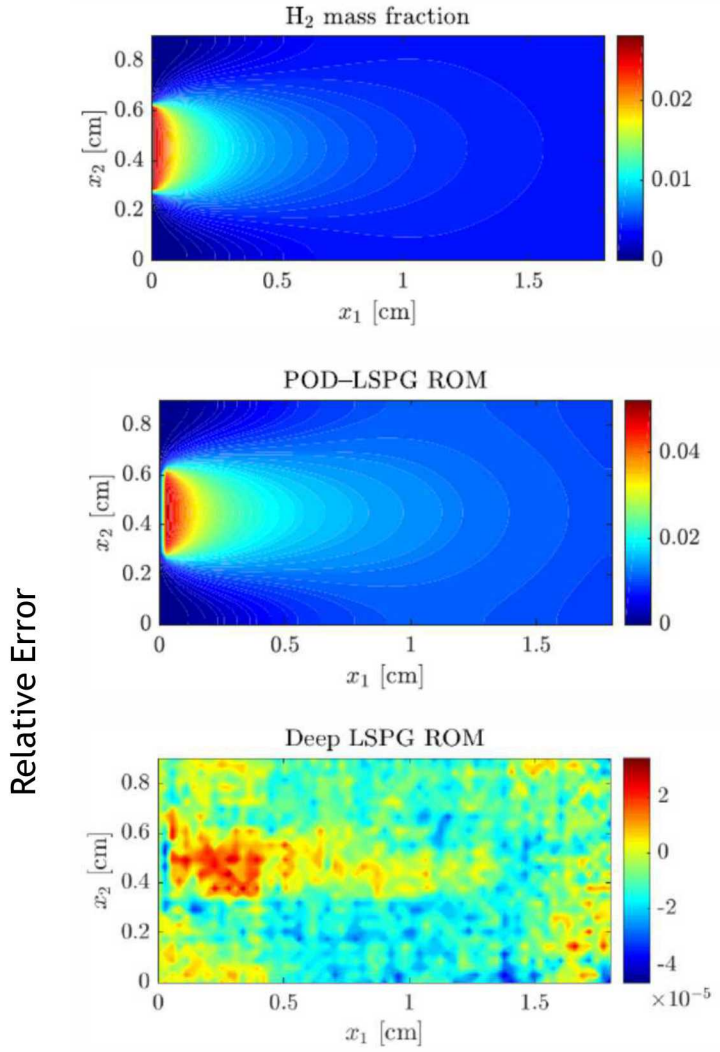
LSPG Projection

$$g(\hat{\boldsymbol{x}}^n) = \underset{v \in \mathbb{R}^{n_{\hat{x}}}}{\operatorname{argmin}} \| \boldsymbol{r}^n(g(v)) \|_2$$
$$n = 1, \cdots, T$$

Manifold Projection ROMs

For advection dominated flows, nonlinear manifold projection techniques have been shown to consistently outperform linear subspace methods.





How to generate nonlinear manifold?

K. Lee and K. Carlberg, *Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders*, 2018

Autoencoders for Manifold Learning

Unsupervised learning method

$$h: x \mapsto h_{dec}(h_{enc}(x))$$

Two parts:

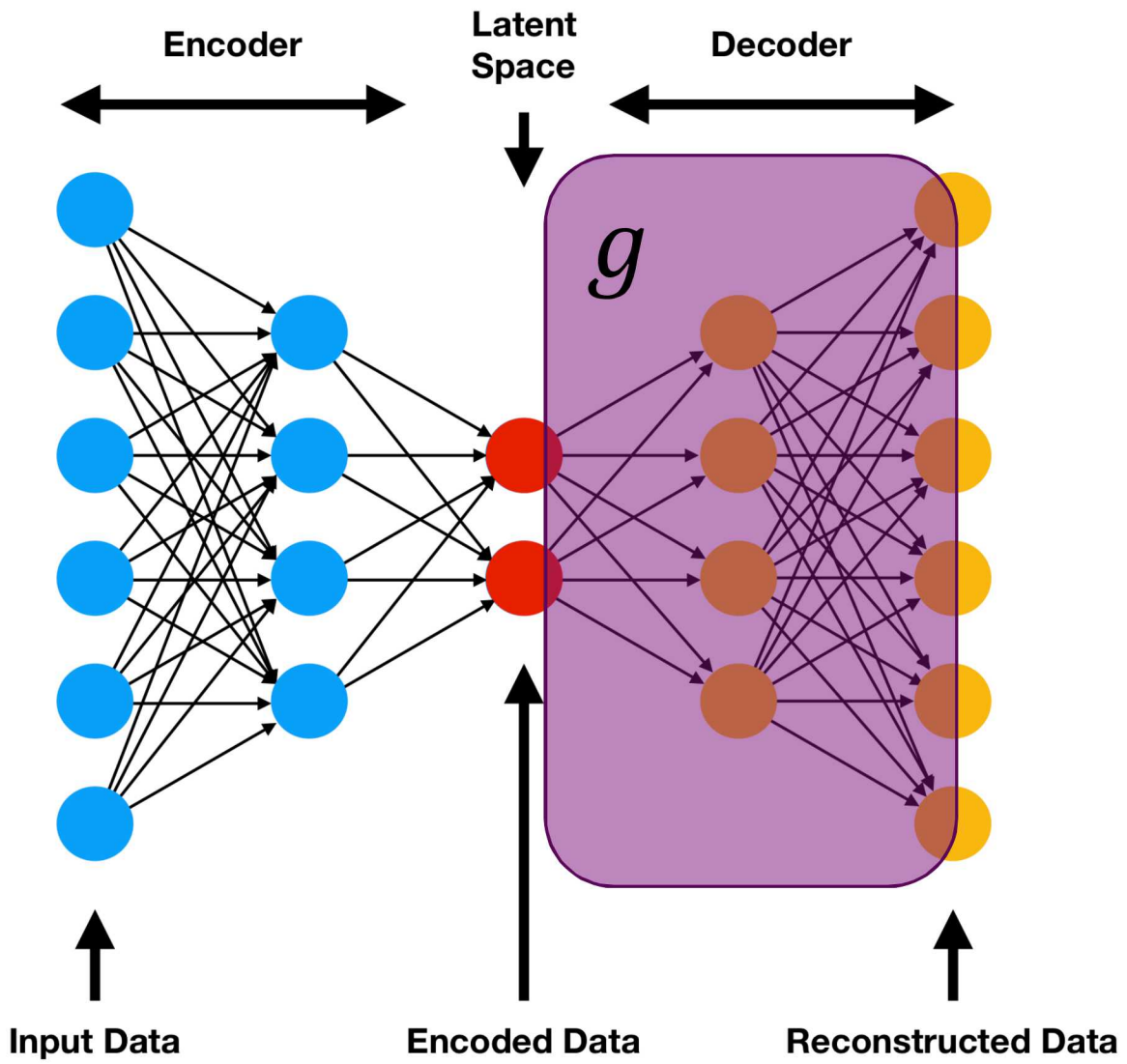
Encoder $h_{enc}: \mathbb{R}^{n_x} \to \mathbb{R}^{n_{\hat{x}}}$

Decoder $h_{dec}: \mathbb{R}^{n_{\hat{x}}} \to \mathbb{R}^{n_x}$

Each layer consists of a dense matrix-vector product and a non-linear activation function.

Large state space dimension ($\mathbb{R}^{n_x}$) leads to impractically large number of parameters.

Restricted to small states



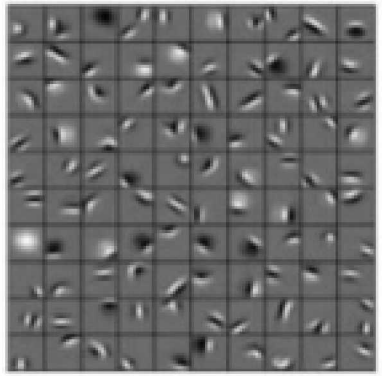
Convolutional Autoencoders

Network architecture common in computer vision.

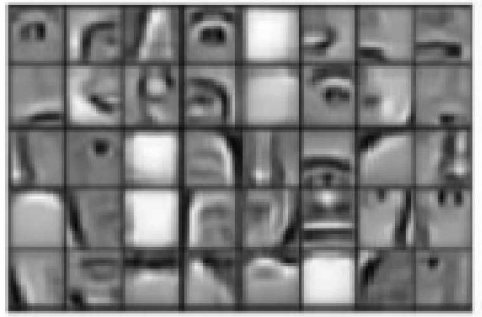
Reduces cost by sharing learned weights across the domain.

Matrix products are now sparse rather than dense.

Low-level feature



Mid-level feature



High-level feature

Source : Deep Learning in a Nutshell: Core Concepts, Nvidia
https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

114				

Restricted to structured data

Convolutional Layers for Unstructured Data

Commonly learned filters are often closely related to differential operators

∇_x

-1	0	1
-4	0	4
-1	0	1

x- Sobel Filter

∇_y

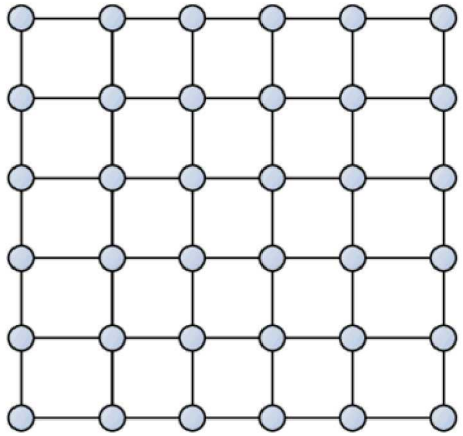
1	4	1
0	0	0
-1	-4	-1

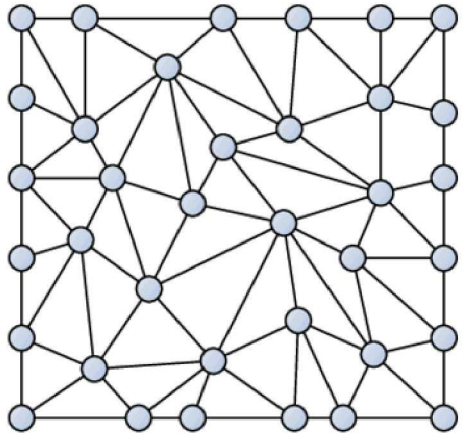
y- Sobel Filter

Δ

-1	-1	-1
-1	8	-1
-1	-1	-1

Edge Detection





Use differential operators defined by the underlying spatial discretization to propagate information.

Operators can be computed offline or on-the-fly.

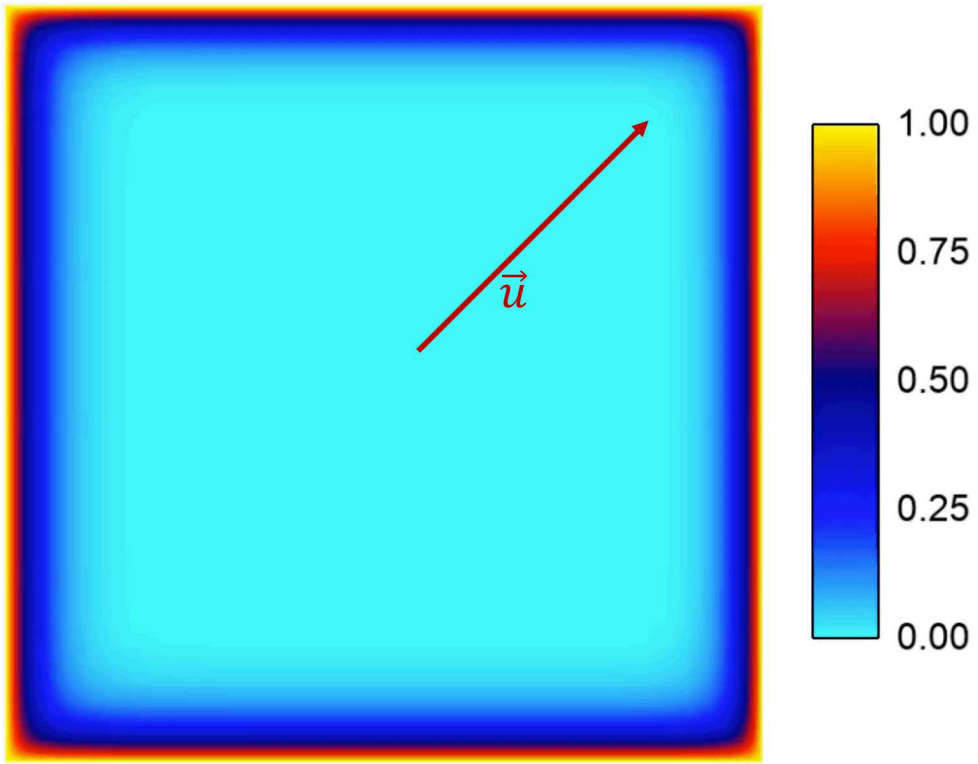
Resulting learned weights will be discretization independent.

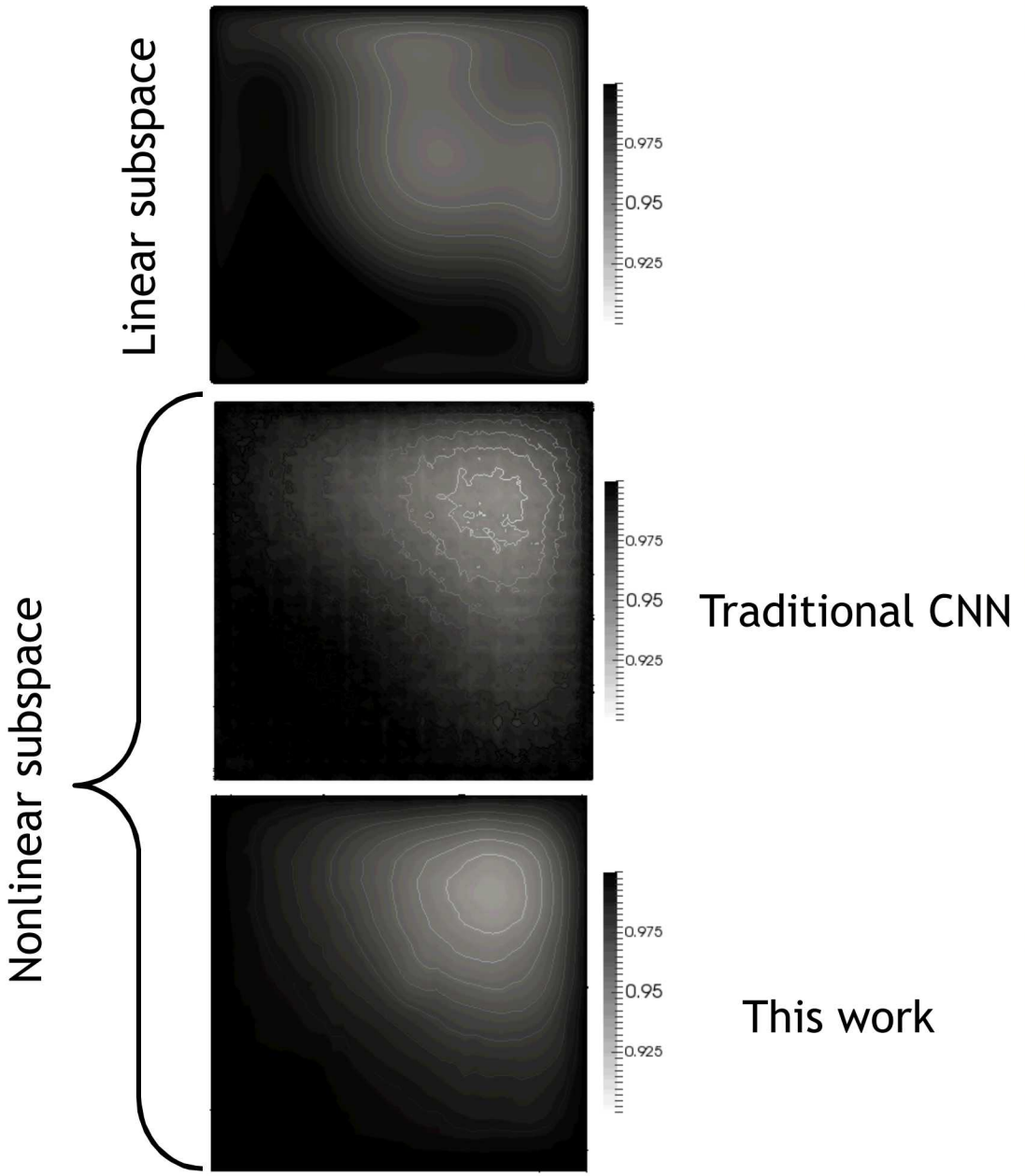
Drop-in replacement for convolutional layers in autoencoder networks.

Results (Transient Advection-Diffusion)

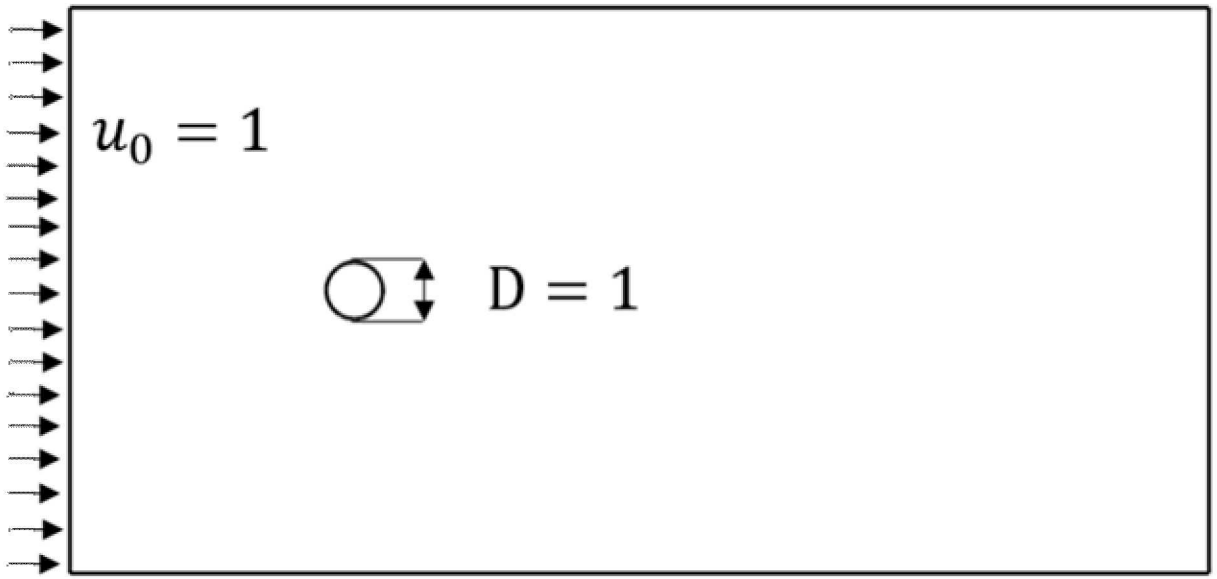
$$\frac{\partial \alpha}{\partial t} = \nabla \cdot (D \nabla \alpha) - \vec{u} \cdot \nabla \alpha$$

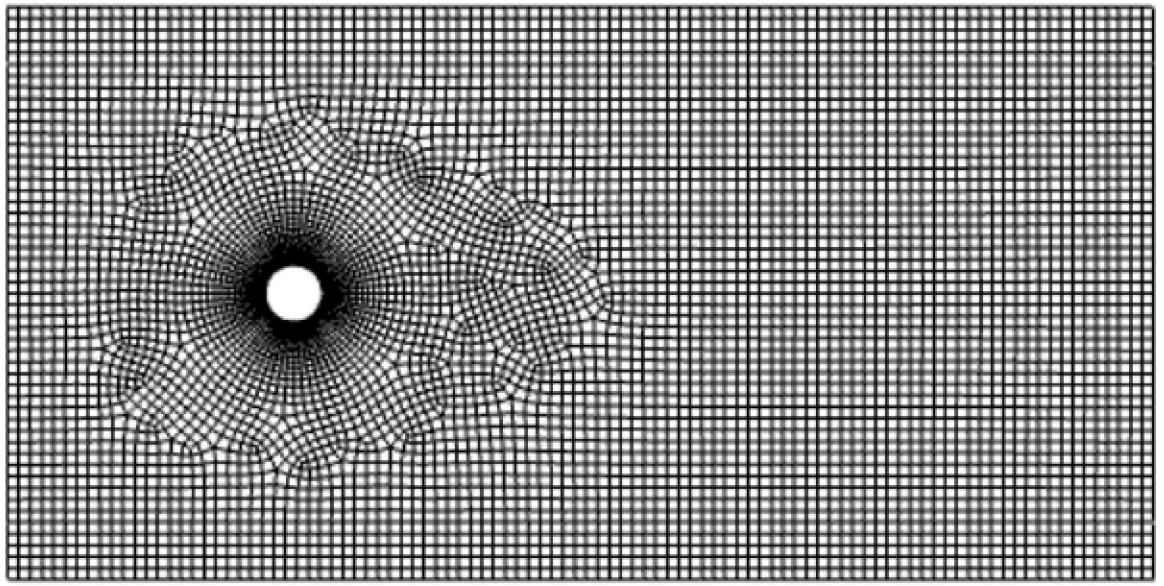
$$\alpha(\vec{r}, 0) = 0$$
$$\alpha(\vec{r}, t) = 1 \text{ for } \vec{r} \in \partial\Omega$$





Results (Incompressible Navier-Stokes)

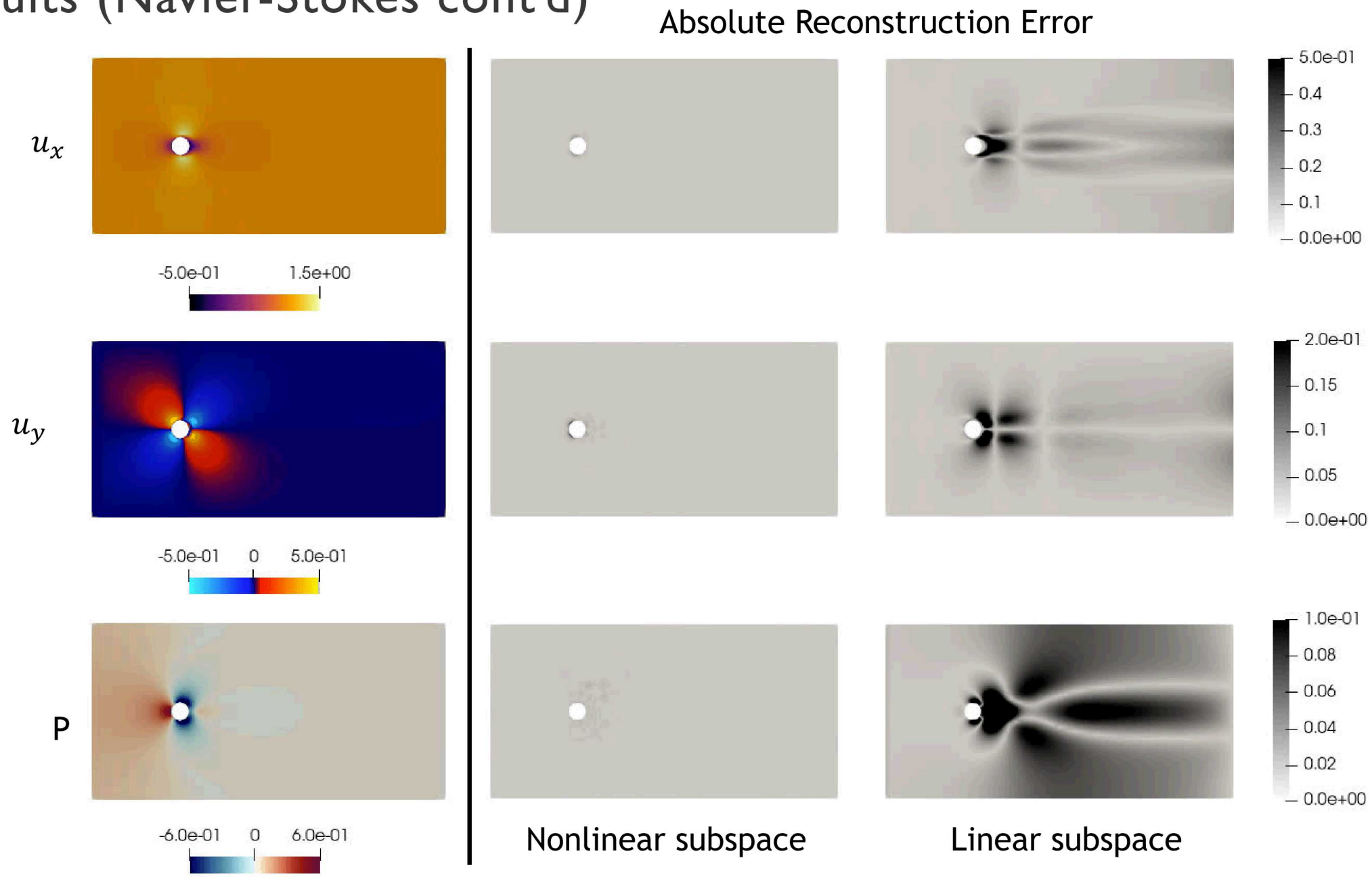




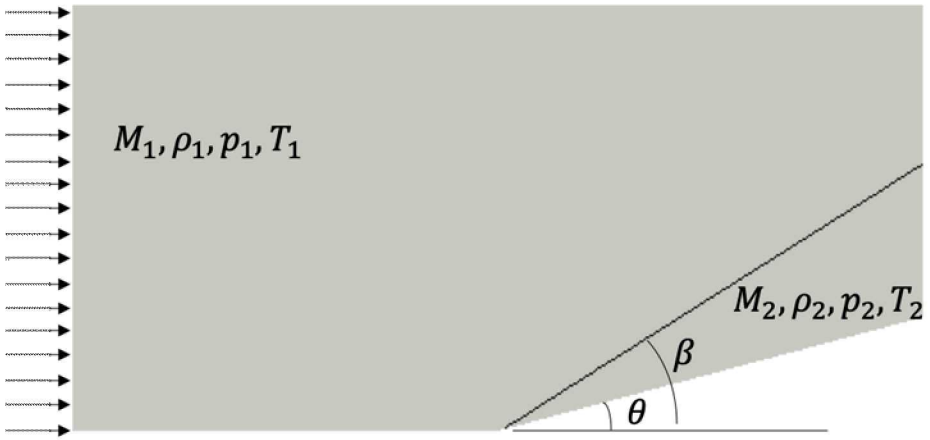
$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} - \upsilon \nabla^2 \vec{u} = 0$$

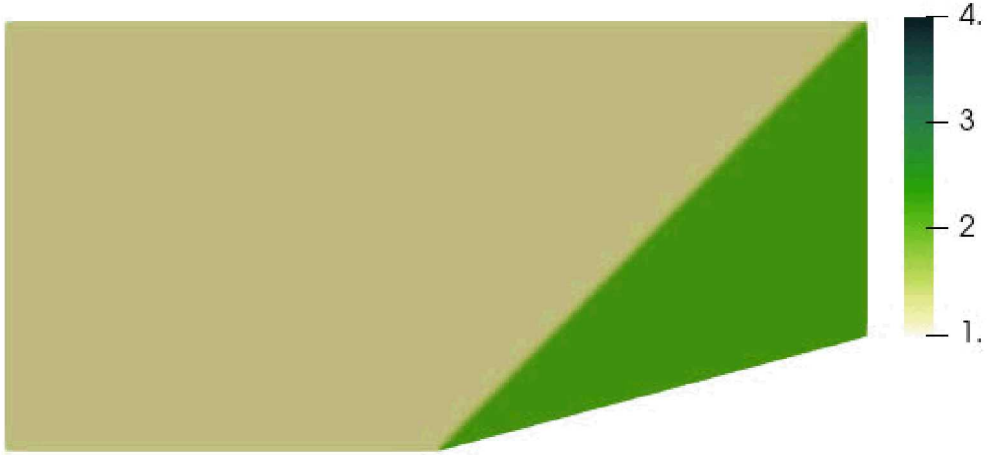
$$\nabla \cdot \vec{u} = 0$$

Results (Navier-Stokes cont'd)



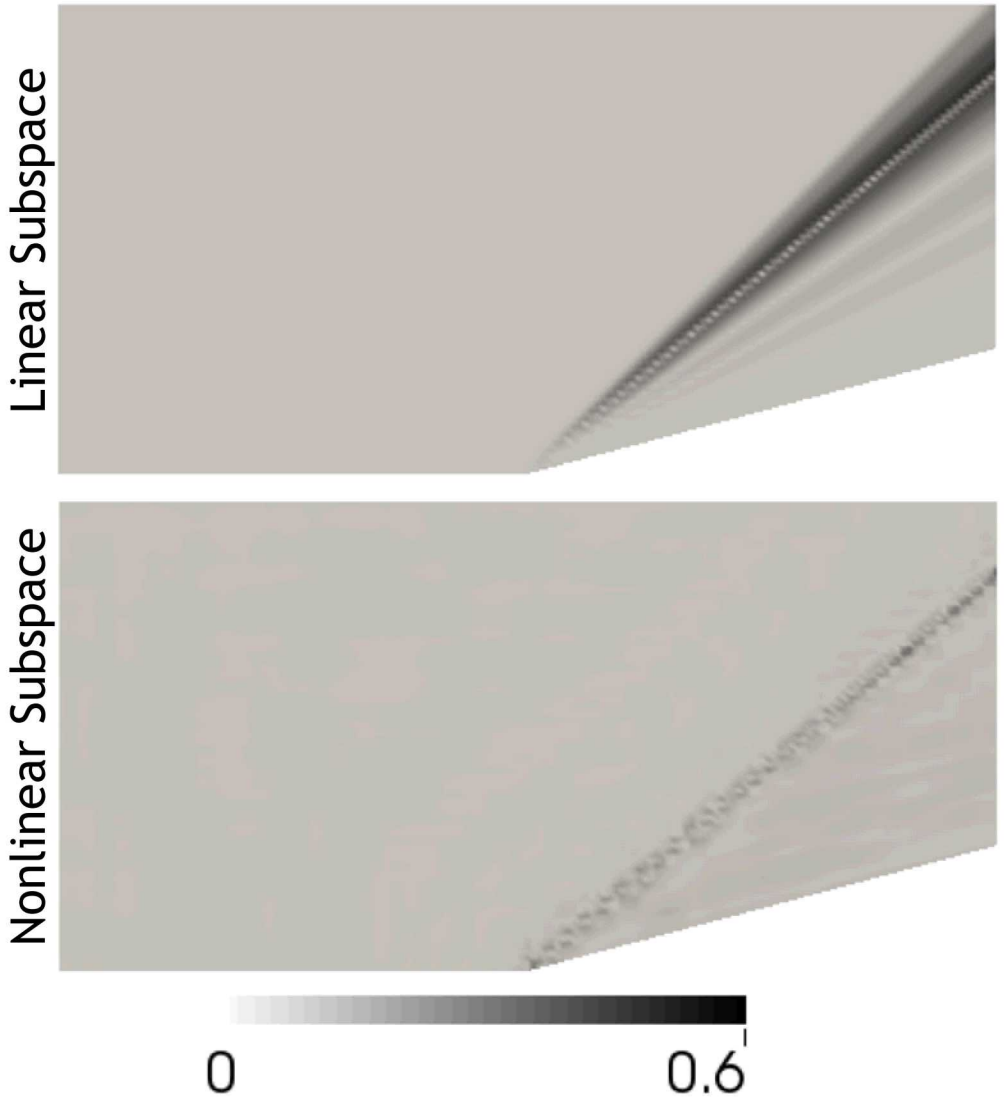
Results (Inviscid Euler)





$M_1 \in \{2, 2.1, \cdots, 5.9, 6.0\}$

$M_1 = 2.25$



Error in density reconstruction

Questions?

Supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525.

Pressio Integration Status

Pressio integration is functional provided:

1. The model is generated/trained using PyTorch exported through TorchVision
2. A finite difference decoder Jacobian is sufficient

I've encountered some difficulties exporting the more complex decoders with TorchVision, so 1 is not a given.