Arbitrary Lagrangian-Eulerian Model Reduction for Advection Dominated Problems and Some Graph Neural Network Ideas

Davide Torlo, Monica Nonino,
Francesco Romor, Gianluigi Rozza

MathLab, Mathematics Area, SISSA International School for Advanced Studies, Trieste, Italy
davidetorlo.it

Lisbon - 5th September 2023

Model Order Reduction for Advection Dominated (Hyperbolic) Problems

First part: ALE one parameter transformations

- Calibration of solutions
- ALE formulation
- PODEIM-Greedy algorithm
- **One parameter** transformation maps
- Regression of calibration

Torlo, D. arXiv preprint arXiv:2003.13735 (2020).

Second part: multiple calibration points

- **Multiple shocks** moving
- Calibration optimization
- Sod shock tube test problem
- Double Mach Reflection

Work in progress with Monica Nonino

Third part: vanishing viscosity and graph Neural Network

- Friedrichs' system
- Vanishing viscosity
- Classical ROMs for high viscosity
- **Graph Neural Network** for low viscosity

Romor, F., Torlo, D., & Rozza, G. (2023). arXiv preprint arXiv:2308.03378.

Table of contents

Motivation: parametric hyperbolic systems

$$\begin{cases} \partial_t u(x,t,\boldsymbol{\mu}) + \nabla \cdot F(u,x,t,\boldsymbol{\mu}) = 0, \quad x \in D,\, t \in \mathbb{R}^+,\, \boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^P \\ \mathbf{B}(u,\boldsymbol{\mu}) = g(t,\boldsymbol{\mu}) \\ u(x,t=0,\boldsymbol{\mu}) = u_0(x,\boldsymbol{\mu}) \end{cases}$$

Properties

- F non linear dependence on $\boldsymbol{\mu}$!
- $\boldsymbol{\mu}$ can influence boundaries, flux, initial conditions

Motivation and solvers

- Why: many physical applications (fluid equations, kinetic models, etc.)
- Classical solvers: **FV, FEM**, FD, **RD** (Slow for high–resolution); **exact solutions**.
- Many query task (UQ, optimization, etc.)

MOR algorithms

- POD (data compression)
- Greedy (data compression)
- EIM (interpolation)
- PODEI-Greedy (data compression and interpolation for nonlinear time-dependent problems)

B. Haasdonk and M. Ohlberger, in Hyperbolic problems: theory, numerics and applications, vol. 67, Amer. Math. Soc., 2009.

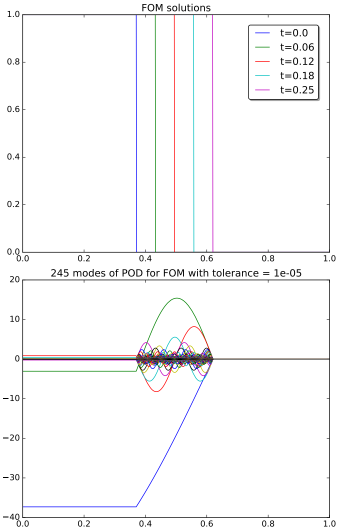
Issues with advection dominated

Issues

- As many basis functions as positions of the shock
- Slow decay of Kolmogorov N–width

$$d_N(\mathcal{S},\mathbb{V}) := \inf_{\mathbb{V}_N\subset\mathbb{V}} \sup_{f\in\mathcal{S}} \inf_{g\in\mathbb{V}_N} ||f-g||$$

- Non linear dependency leads to big EIM and RB space
- 1/2 parameters problem (highly non linear dependence on parameters)



Possible solutions and research directions

Some possibilities to incorporate the advection into RB framework

- Freezing **Ohlberger, M. and Rave, S.**
- Shifted POD **Reiss, J., Schulze, P., Sesterhenn, J., Mehrmann, V., Demo, N., Burela, S., Krah, P.**
- Lagrangian basis method **Mojgani, R. and Balajewicz, M.**
- Advection modes by optimal mass transport **Iollo, A., Lombardi, D., Mula, O., Taddei, T.**
- Calibration (also 2D non–periodic boundaries) **Cagniart, N., Stamm, B. and Maday, Y.**, **Crisovan, R. and Abgrall, R.**
- Online adaptive bases and samplings **Peherstorfer, B.**
- Gradient-preserving DEIM **Pagliantini, C.**
- Transport Reversal **Rim, D., Moe, S. and LeVeque R. J.**
- Registration method **Taddei, T., Ohlberger, M., Kleikamp, H.**
- Optimization based implicit feature tracking **Zahr, M., Mirhoseini, M.A.**
- Preprocessing reduced basis **Karatzas, E., Nonino, M., Ballarin, F., Rozza, G. and Maday, Y.**
- Manifold learning via Neural Network, convolutional autoencoders **Carlberg, K. and Lee, K.; Lye, K., Mishra S. and Ray, D.; Fresca, S., Dedè, L. and Manzoni, A., Venkat, S., Smith, R.C., Kelley, C.T.**
- Dynamic Modes **Lu, H. and Tartakovsky, D. M.**
- Dynamical Low Rank **Kazashi, Y., Nobile, F., Trigo Trindade, T., Vidličková, E., Ceruti, G., Kusch, J., Einkemmer, L., Frank, M.**
- Graph Neural Network **Pichi, F., Moya, B., Hesthaven, J.**
- Sinkhorn Loss and Wasserstein Kernel **Khamlich, M., Pichi, Rozza, G.**

Possible solutions and research directions

Some possibilities to incorporate the advection into RB framework

- Freezing **Ohlberger, M. and Rave, S.**
- Shifted POD **Reiss, J., Schulze, P., Sesterhenn, J., Mehrmann, V., Demo, N., Burela, S., Krah, P.**
- Lagrangian basis method **Mojgani, R. and Balajewicz, M.**
- Advection modes by optimal mass transport **Iollo, A., Lombardi, D., Mula, O., Taddei, T.**
- Calibration (also 2D non–periodic boundaries) **Cagniart, N., Stamm, B. and Maday, Y.**, **Crisovan, R. and Abgrall, R.**
- Online adaptive bases and samplings **Peherstorfer, B.**
- Gradient-preserving DEIM **Pagliantini, C.**
- Transport Reversal **Rim, D., Moe, S. and LeVeque R. J.**
- Registration method **Taddei, T., Ohlberger, M., Kleikamp, H.**
- Optimization based implicit feature tracking **Zahr, M., Mirhoseini, M.A.**
- Preprocessing reduced basis **Karatzas, E., Nonino, M., Ballarin, F., Rozza, G. and Maday, Y.**
- Manifold learning via Neural Network, convolutional autoencoders **Carlberg, K. and Lee, K.; Lye, K., Mishra S. and Ray, D.; Fresca, S., Dedè, L. and Manzoni, A., Venkat, S., Smith, R.C., Kelley, C.T.**
- Dynamic Modes **Lu, H. and Tartakovsky, D. M.**
- Dynamical Low Rank **Kazashi, Y., Nobile, F., Trigo Trindade, T., Vidličková, E., Ceruti, G., Kusch, J., Einkemmer, L., Frank, M.**
- Graph Neural Network **Pichi, F., Moya, B., Hesthaven, J.**
- Sinkhorn Loss and Wasserstein Kernel **Khamlich, M., Pichi, Rozza, G.**

Possible solutions and research directions

Some possibilities to incorporate the advection into RB framework

- Freezing **Ohlberger, M. and Rave, S.**
- Shifted POD **Reiss, J., Schulze, P., Sesterhenn, J., Mehrmann, V., Demo, N., Burela, S., Krah, P.**
- Lagrangian basis method **Mojgani, R. and Balajewicz, M.**
- Advection modes by optimal mass transport **Iollo, A., Lombardi, D., Mula, O., Taddei, T.**
- Calibration (also 2D non–periodic boundaries) **Cagniart, N., Stamm, B. and Maday, Y.**, **Crisovan, R. and Abgrall, R.**
- Online adaptive bases and samplings **Peherstorfer, B.**
- Gradient-preserving DEIM **Pagliantini, C.**
- Transport Reversal **Rim, D., Moe, S. and LeVeque R. J.**
- Registration method **Taddei, T., Ohlberger, M., Kleikamp, H.**
- Optimization based implicit feature tracking **Zahr, M., Mirhoseini, M.A.**
- Preprocessing reduced basis **Karatzas, E., Nonino, M., Ballarin, F., Rozza, G. and Maday, Y.**
- Manifold learning via Neural Network, convolutional autoencoders **Carlberg, K. and Lee, K.; Lye, K., Mishra S. and Ray, D.; Fresca, S., Dedè, L. and Manzoni, A., Venkat, S., Smith, R.C., Kelley, C.T.**
- Dynamic Modes **Lu, H. and Tartakovsky, D. M.**
- Dynamical Low Rank **Kazashi, Y., Nobile, F., Trigo Trindade, T., Vidličková, E., Ceruti, G., Kusch, J., Einkemmer, L., Frank, M.**
- Graph Neural Network **Pichi, F., Moya, B., Hesthaven, J.**
- Sinkhorn Loss and Wasserstein Kernel **Khamlich, M., Pichi, Rozza, G.**

Possible solutions and research directions

Some possibilities to incorporate the advection into RB framework

- Freezing **Ohlberger, M. and Rave, S.**
- Shifted POD **Reiss, J., Schulze, P., Sesterhenn, J., Mehrmann, V., Demo, N., Burela, S., Krah, P.**
- Lagrangian basis method **Mojgani, R. and Balajewicz, M.**
- Advection modes by optimal mass transport **Iollo, A., Lombardi, D., Mula, O., Taddei, T.**
- Calibration (also 2D non–periodic boundaries) **Cagniart, N., Stamm, B. and Maday, Y.**, **Crisovan, R. and Abgrall, R.**
- Online adaptive bases and samplings **Peherstorfer, B.**
- Gradient-preserving DEIM **Pagliantini, C.**
- Transport Reversal **Rim, D., Moe, S. and LeVeque R. J.**
- Registration method **Taddei, T., Ohlberger, M., Kleikamp, H.**
- Optimization based implicit feature tracking **Zahr, M., Mirhoseini, M.A.**
- Preprocessing reduced basis **Karatzas, E., Nonino, M., Ballarin, F., Rozza, G. and Maday, Y.**
- Manifold learning via Neural Network, convolutional autoencoders **Carlberg, K. and Lee, K.; Lye, K., Mishra S. and Ray, D.; Fresca, S., Dedè, L. and Manzoni, A., Venkat, S., Smith, R.C., Kelley, C.T.**
- Dynamic Modes **Lu, H. and Tartakovsky, D. M.**
- Dynamical Low Rank **Kazashi, Y., Nobile, F., Trigo Trindade, T., Vidličková, E., Ceruti, G., Kusch, J., Einkemmer, L., Frank, M.**
- Graph Neural Network **Pichi, F., Moya, B., Hesthaven, J.**
- Sinkhorn Loss and Wasserstein Kernel **Khamlich, M., Pichi, Rozza, G.**

Transfomation of the domain

Geometry map T

$$T : \Theta \times \mathcal{R} \to \Omega \tag{1}$$

- $T(\cdot,\cdot) \in \mathcal{C}^1(\Theta \times \mathcal{R}, \Omega)$,
- $\exists\, T^{-1} : \Theta \times \Omega \to \mathcal{R}$ such that $T^{-1}(\theta, T(\theta, y)) = y$ for $y \in \mathcal{R}$ and $T(\theta, T^{-1}(\theta, x)) = x$ for $x \in \Omega$,
- $T^{-1}(\cdot,\cdot) \in \mathcal{C}^1(\Theta \times \Omega, \mathcal{R})$.

Calibration map θ

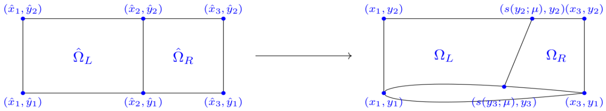
$$\theta : \mathcal{P} \times [0, t_f] \to \Theta$$

- $\theta(\cdot, \boldsymbol{\mu}) \in \mathcal{C}^1([0, t_f], \Theta)$ for all $\boldsymbol{\mu} \in \mathcal{P}$,
- $u_{\mathcal{N}}(T(\theta(t, \boldsymbol{\mu}), y), t, \boldsymbol{\mu}) \approx \bar{v}(y), \quad \forall \boldsymbol{\mu} \in \mathcal{P},\ t \in [0, t_f], y \in \mathcal{R}$

Transformation map for MOR

Examples: θ is the point of maximum height or of steepest solution, or some random points.

- Translation: $T(\theta, y) = y + \theta - 0.5$
- Dilatation: $T(\theta, y) = \frac{y\theta}{(2\theta-1)y+1-\theta}$
- Piece-wise Cubic Hermite Interpolator Polynomial
- Higher degree polynomials
- Gordon-Hall



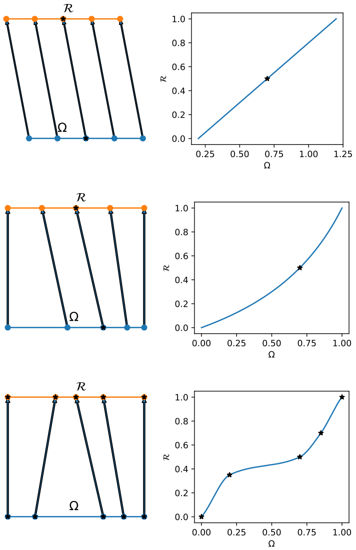


Table of contents

Arbitrary Lagrangian–Eulerian formulation

ALE formulation

$$\frac{\partial}{\partial t} v(y, \boldsymbol{\mu}, t) + \frac{dy}{dx}\frac{d}{dy} F(v, \mu) - \frac{dy}{dx}\frac{dv}{dy}\frac{\partial T}{\partial t} = 0$$

With ALE formulation we can apply the EIM procedure with points on the reference domain $\mathcal{R}$.

What does it imply?

- We must know $T(\theta(t, \boldsymbol{\mu}), y)$
    - Offline phase: detect some interesting points (maxima, steepest gradient) (second part)
    - Offline phase: optimize the transformation in some sense (T. Taddei, Ohlberger et al.) (second part)
    - Online phase: predict the value of the transformation. Regression (polynomials, ANN), projections (first part)
- Compute the Jacobian of the transformation $\frac{dy}{dx}$ and the new flux $\frac{dv}{dy}$ $\Rightarrow$ increasing computational costs also in online phase

Arbitrary Lagrangian–Eulerian formulation

ALE formulation

$$\frac{\partial}{\partial t}v(y,\boldsymbol{\mu},t) + \frac{dy}{dx}\frac{d}{dy}F(v,\mu) - \frac{dy}{dx}\frac{dv}{dy}\frac{\partial T}{\partial t} = 0$$

With ALE formulation we can apply the EIM procedure with points on the reference domain $\mathcal{R}$.

What does it imply?

- We must know $T(\theta(t,\boldsymbol{\mu}),y)$
  - Offline phase: detect some interesting points (maxima, steepest gradient) (second part)
  - Offline phase: optimize the transformation in some sense (T. Taddei, Ohlberger et al.) (second part)
  - Online phase: predict the value of the transformation. Regression (polynomials, ANN), projections (first part)
- Compute the Jacobian of the transformation $\frac{dy}{dx}$ and the new flux $\frac{dv}{dy}$ $\Rightarrow$ increasing computational costs also in online phase

Review of the algorithm: PODEI-Greedy in ALE framework

INITIALIZATION of ALE-PODEI-Greedy:

- Compute some Eulerian FOMs
- Compute or optimize $\theta(\boldsymbol{\mu}_k, t^n)$ for some $\boldsymbol{\mu}_k \in \mathcal{P}$ and $n \leq N_t$
- Build the regression map $\hat{\theta} : \mathcal{P} \times \mathbb{R}^+ \to \mathbb{R}^q$

INITIALIZATION of PODEI-Greedy:

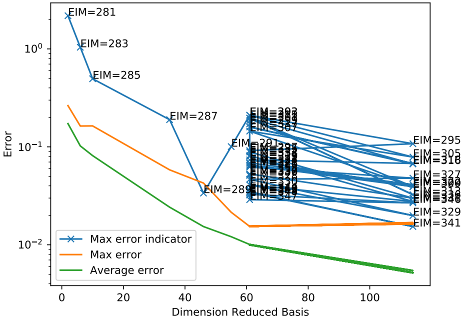
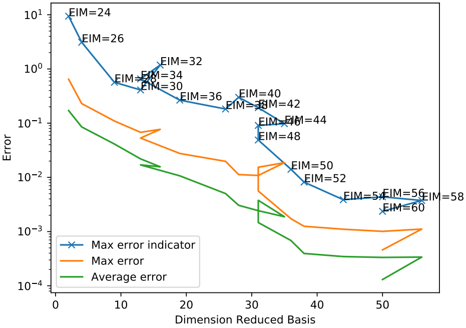
- EIM on ALE-RHS/fluxes of all times for a given parameter $\boldsymbol{\mu}_0$
- $RB = POD(\{v^n(\boldsymbol{\mu}_0)\}_{n=0}^{N_t})$

ITERATION:

- Greedy algorithm spanning over the parameter space $\mathcal{P}_h$, with an error indicator $\varepsilon(v(\boldsymbol{\mu}, t^n, \hat{\theta}(\boldsymbol{\mu}, t^n)))$
- Choose worst parameter as $\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\mu} \in \mathcal{P}_h} \varepsilon(\mathbf{v}(\boldsymbol{\mu}))$
- Apply POD on ALE time evolution of selected solution $POD_{add} = POD\left(\{v^n(\boldsymbol{\mu}^*)\}_{n=1}^{N_t}\right)$
- Update the RB with $RB = POD\left(RB \cup POD_{add}\right)$
- Update EIM basis function with $EIM_{space} = EIM_{space} \cup EIM(\{RHS(\boldsymbol{\mu}^*, t^n, \hat{\theta}(\boldsymbol{\mu}^*, t^n))\}_{n=0}^{N_t})$

Burgers' equation

$$\begin{cases} u_t + \mu_0 (u^2/2)_x = 0, \; D = [0,1], \; T_{max} = 0.6, \; \text{Dirichlet BC} \\ u_0(x, \boldsymbol{\mu}) = \sin(2\pi(x + 0.1\mu_1)) e^{-(60+20\mu_2)(x-0.5)^2} (1 + 0.5\mu_3 x) \\ \mu_0 \sim \mathcal{U}([0,2]), \; \mu_1 \sim \mathcal{U}([0,1]), \; \mu_2, \mu_3 \sim \mathcal{U}([-1,1]) \end{cases}$$

Without calibration	With calibration: Poly3
	

Burgers' equation

$$\begin{cases} u_t + \mu_0 (u^2/2)_x = 0, \; D = [0,1], \; T_{max} = 0.6, \text{ Dirichlet BC} \\ u_0(x, \boldsymbol{\mu}) = \sin(2\pi(x + 0.1\mu_1)) e^{-(60+20\mu_2)(x-0.5)^2} (1 + 0.5\mu_3 x) \\ \mu_0 \sim \mathcal{U}([0,2]), \; \mu_1 \sim \mathcal{U}([0,1]), \; \mu_2, \mu_3 \sim \mathcal{U}([-1,1]) \end{cases}$$

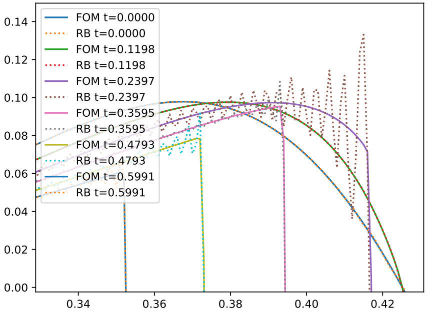
Without calibration[1]		With calibration: Poly3	
RB dim	153	RB dim	50
EIM dim	335	EIM dim	60
FOM time	119 s	FOM time	314 s
RB time	50 s	RB time	35 s
RB/FOM time	42%	RB/FOM time	11%

[1] It does not reach the requested tolerance 10^{-3}

Burgers' equation

$$\begin{cases} u_t + \mu_0 (u^2/2)_x = 0, \; D = [0,1], \; T_{max} = 0.6, \text{ Dirichlet BC} \\ u_0(x,\boldsymbol{\mu}) = \sin(2\pi(x+0.1\mu_1))e^{-(60+20\mu_2)(x-0.5)^2}(1+0.5\mu_3 x) \\ \mu_0 \sim \mathcal{U}([0,2]), \; \mu_1 \sim \mathcal{U}([0,1]), \; \mu_2, \mu_3 \sim \mathcal{U}([-1,1]) \end{cases}$$

Without calibration



With calibration: Poly3

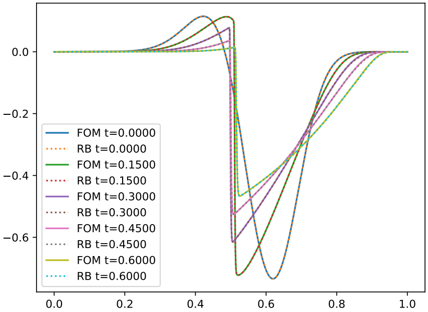
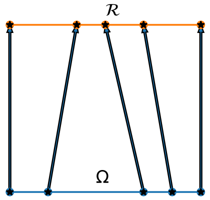


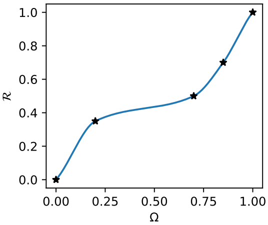
Table of contents

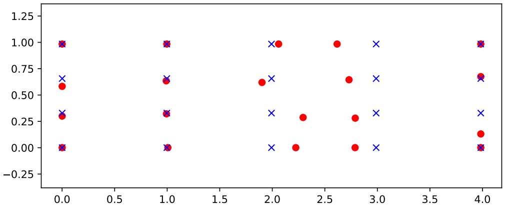
Multiple points to align

Piecewise Cubic Hermite Interpolating Polynomial PCHIP

- Interpolates some points (not optimizing on the whole mesh)
- Maintains monotonicity of the points (always invertible)
- Polynomials (easy to deal with)
- Easy generalization to Cartesian grids
- More complicated meshes ?







Optimal θs

Optimization

- Find θ that minimizes

$$||u(T(\theta,\cdot),t,\boldsymbol{\mu}) - \bar{u}(\cdot)||_{\mathcal{R}}$$

- Penalization $|\partial_t \theta|$
- Penalization $\max_{y\in\mathcal{R}} \partial_y T(\theta, y)$ and $\max_{x\in\Omega} \partial_x T^{-1}(\theta, x)$
- Constraint on order $\theta_i < \theta_{i+1}$
- Initial guess, order of optimization
- Algorithm: Sequential Least Squares Programming

Optimal θs

What is $\bar{u}$?

- In some tests $u(t^{end}, \boldsymbol{\mu})$ is a good choice
- What if more parameters with different u values?

Optimization

- Find θ that minimizes
$$||u(T(\theta,\cdot), t, \boldsymbol{\mu}) - \bar{u}(\cdot)||_{\mathcal{R}}$$
- Penalization $|\partial_t \theta|$
- Penalization $\max_{y\in\mathcal{R}} \partial_y T(\theta, y)$ and $\max_{x\in\Omega} \partial_x T^{-1}(\theta, x)$
- Constraint on order $\theta_i < \theta_{i+1}$
- Initial guess, order of optimization
- Algorithm: Sequential Least Squares Programming

Optimal θs

Optimization

- Find θ that minimizes

$$||u(T(\theta,\cdot),t,\boldsymbol{\mu}) - \bar{u}(\cdot)||_{\mathcal{R}}$$

- Penalization $|\partial_t \theta|$
- Penalization $\max_{y\in\mathcal{R}} \partial_y T(\theta,y)$ and $\max_{x\in\Omega} \partial_x T^{-1}(\theta,x)$
- Constraint on order $\theta_i < \theta_{i+1}$
- Initial guess, order of optimization
- Algorithm: Sequential Least Squares Programming

What is $\bar{u}$?

- In some tests $u(t^{end},\boldsymbol{\mu})$ is a good choice
- What if more parameters with different u values?

Generalization for more u values

- Find θ that minimizes

$$||u(T(\theta,\cdot),t,\boldsymbol{\mu}) - \mathcal{P}_{\mathbb{V}_{N_{RB}}}(u(T(\theta,\cdot),t,\boldsymbol{\mu}))||_{\mathcal{R}}$$

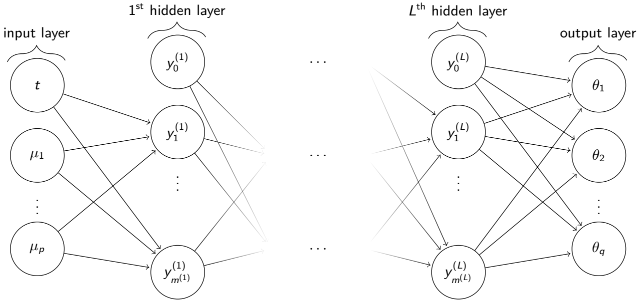
- What is $\mathbb{V}_{N_{RB}}$ at this point? Using few snapshots $\{u(\boldsymbol{\mu}_j,t^{end})\}_{j=1}^M$ for different parameters optimized
  - **Manually** (if possible multiple features detecting)
  - **Iteratively optimizing** $\theta(\boldsymbol{\mu}_j)$ using

$$\mathbb{V}_{N_{RB}} = POD(\{u(\boldsymbol{\mu}_j,t^{end})\circ T(\theta(\boldsymbol{\mu}_j))\}_{j=1}^M)$$

Learning

Learning θ

- Use calibrated θ to get an estimator $\widehat{\theta}(t, \boldsymbol{\mu})$
- ANN with 4 hidden layers, 16 neurons each, tanh activation
- Enforcing $\theta_i < \theta_{i+1}$ with Softplus final activation function



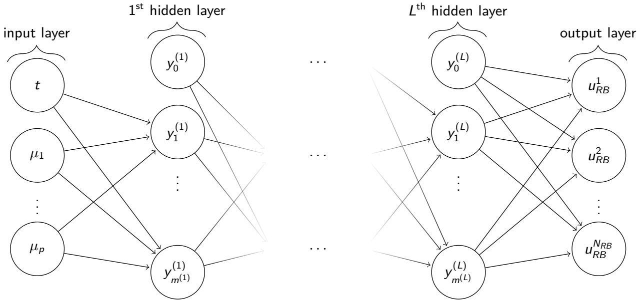
Learning

Learning θ

- Use calibrated θ to get an estimator $\widehat{\theta}(t,\boldsymbol{\mu})$
- ANN with 4 hidden layers, 16 neurons each, tanh activation
- Enforcing $\theta_i < \theta_{i+1}$ with Softplus final activation function

Learning u_{RB} (POD-NN)

- Compute POD on $\{u \circ T(\widehat{\theta}, t^n, \boldsymbol{\mu}_j)\}_{j,n}$ extract $\mathbb{V}_{N_{RB}}$
- Project $\{u \circ T(\widehat{\theta}, t^n, \boldsymbol{\mu}_j)\}_{j,n}$ onto $\mathbb{V}_{N_{RB}}$ obtaining the reduced coefficients $u_{RB}(t^n, \boldsymbol{\mu}_j)$
- Learn the map $\widehat{u}_{RB}(t,\boldsymbol{\mu})$ with an ANN with 4 hidden layers, 16 neurons and tanh activation



Sod shock tube test case: no parameters only time dependence

Euler Equations

$$\begin{cases} \partial_t \rho + \partial_x(\rho u) = 0 \\ \partial_t(\rho u) + \partial_x(\rho u^2 + p) = 0 \\ \partial_t(\rho E) + \partial_x(u(\rho E + p)) = 0 \\ + \text{EOS: } E = \frac{p}{\rho(\gamma-1)} + \frac{u^2}{2} \end{cases}$$

Riemann Problem: Sod Shock tube

$$\begin{pmatrix} \rho \\ u \\ p \end{pmatrix} = \begin{cases} \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}^T & \text{if } x < 0.5 \\ \begin{pmatrix} 0.1 & 0 & 0.125 \end{pmatrix}^T & \text{if } x > 0.5 \end{cases}$$

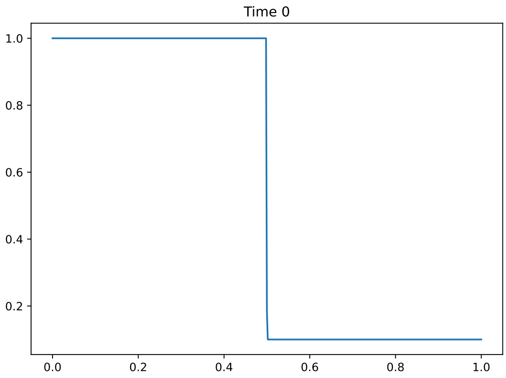
Sod shock tube test case: no parameters only time dependence

Euler Equations

$$\begin{cases} \partial_t \rho + \partial_x(\rho u) = 0 \\ \partial_t(\rho u) + \partial_x(\rho u^2 + p) = 0 \\ \partial_t(\rho E) + \partial_x(u(\rho E + p)) = 0 \\ + \text{ EOS: } E = \frac{p}{\rho(\gamma-1)} + \frac{u^2}{2} \end{cases}$$

Riemann Problem: Sod Shock tube

$$\begin{pmatrix} \rho \\ u \\ p \end{pmatrix} = \begin{cases} \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}^T & \text{if } x < 0.5 \\ \begin{pmatrix} 0.1 & 0 & 0.125 \end{pmatrix}^T & \text{if } x > 0.5 \end{cases}$$



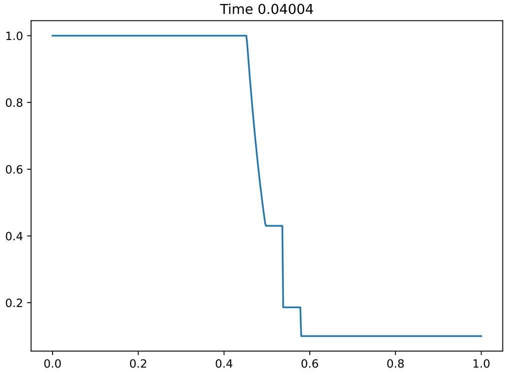
Sod shock tube test case: no parameters only time dependence

Euler Equations

$$\begin{cases} \partial_t \rho + \partial_x(\rho u) = 0 \\ \partial_t(\rho u) + \partial_x(\rho u^2 + p) = 0 \\ \partial_t(\rho E) + \partial_x(u(\rho E + p)) = 0 \\ + \text{ EOS: } E = \frac{p}{\rho(\gamma-1)} + \frac{u^2}{2} \end{cases}$$

Riemann Problem: Sod Shock tube

$$\begin{pmatrix} \rho \\ u \\ p \end{pmatrix} = \begin{cases} \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}^T & \text{if } x < 0.5 \\ \begin{pmatrix} 0.1 & 0 & 0.125 \end{pmatrix}^T & \text{if } x > 0.5 \end{cases}$$



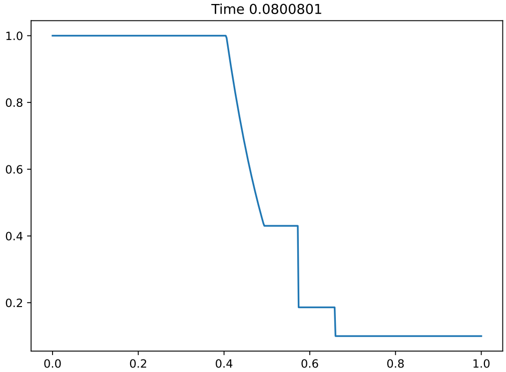
Sod shock tube test case: no parameters only time dependence

Euler Equations

$$\begin{cases} \partial_t \rho + \partial_x(\rho u) = 0 \\ \partial_t(\rho u) + \partial_x(\rho u^2 + p) = 0 \\ \partial_t(\rho E) + \partial_x(u(\rho E + p)) = 0 \\ + \text{EOS: } E = \frac{p}{\rho(\gamma-1)} + \frac{u^2}{2} \end{cases}$$

Riemann Problem: Sod Shock tube

$$\begin{pmatrix} \rho \\ u \\ p \end{pmatrix} = \begin{cases} \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}^T & \text{if } x < 0.5 \\ \begin{pmatrix} 0.1 & 0 & 0.125 \end{pmatrix}^T & \text{if } x > 0.5 \end{cases}$$



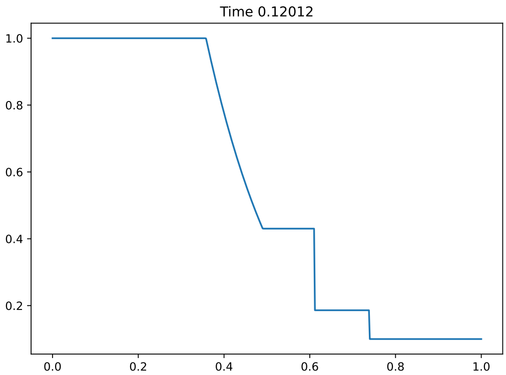
Sod shock tube test case: no parameters only time dependence

Euler Equations

$$\begin{cases} \partial_t \rho + \partial_x(\rho u) = 0 \\ \partial_t(\rho u) + \partial_x(\rho u^2 + p) = 0 \\ \partial_t(\rho E) + \partial_x(u(\rho E + p)) = 0 \\ + \text{ EOS: } E = \frac{p}{\rho(\gamma-1)} + \frac{u^2}{2} \end{cases}$$

Riemann Problem: Sod Shock tube

$$\begin{pmatrix} \rho \\ u \\ p \end{pmatrix} = \begin{cases} \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}^T & \text{if } x < 0.5 \\ \begin{pmatrix} 0.1 & 0 & 0.125 \end{pmatrix}^T & \text{if } x > 0.5 \end{cases}$$



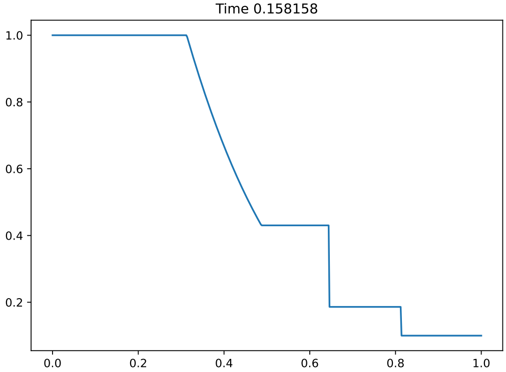
Sod shock tube test case: no parameters only time dependence

Euler Equations

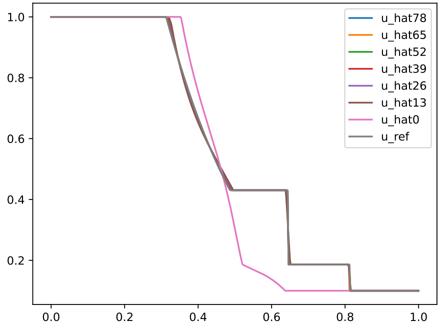
$$\begin{cases} \partial_t \rho + \partial_x(\rho u) = 0 \\ \partial_t(\rho u) + \partial_x(\rho u^2 + p) = 0 \\ \partial_t(\rho E) + \partial_x(u(\rho E + p)) = 0 \\ + \text{ EOS: } E = \frac{p}{\rho(\gamma-1)} + \frac{u^2}{2} \end{cases}$$

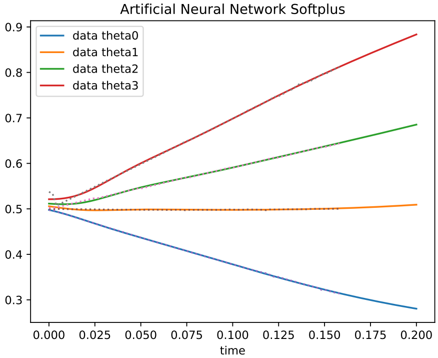
Riemann Problem: Sod Shock tube

$$\begin{pmatrix} \rho \\ u \\ p \end{pmatrix} = \begin{cases} \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}^T & \text{if } x < 0.5 \\ \begin{pmatrix} 0.1 & 0 & 0.125 \end{pmatrix}^T & \text{if } x > 0.5 \end{cases}$$



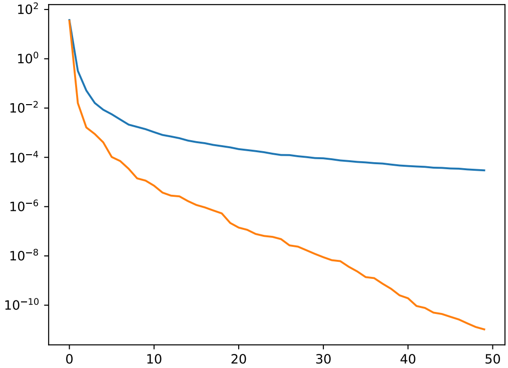
Transformations and calibration



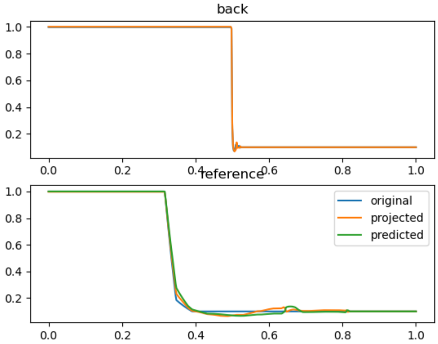


Results: online with POD-NN

Singular values decay for original problem (blue) and transformed problem (orange)

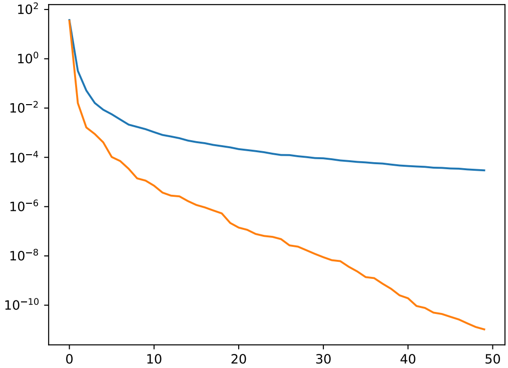


Multilayer Perceptron Regression for θ

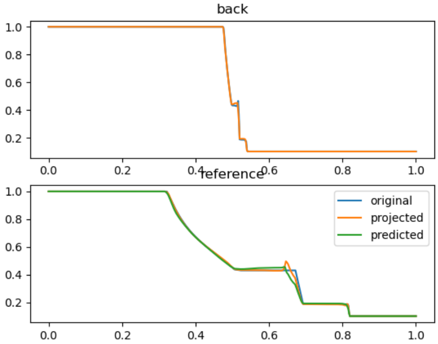


Results: online with POD-NN

Singular values decay for original problem (blue) and transformed problem (orange)

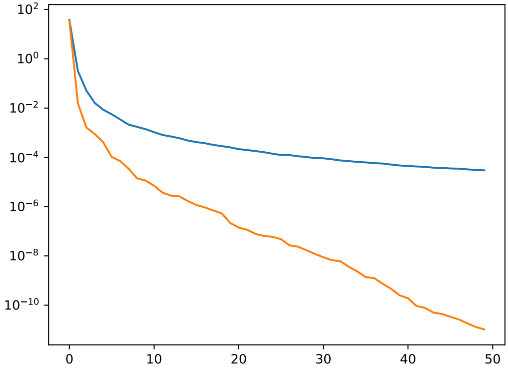


Multilayer Perceptron Regression for θ

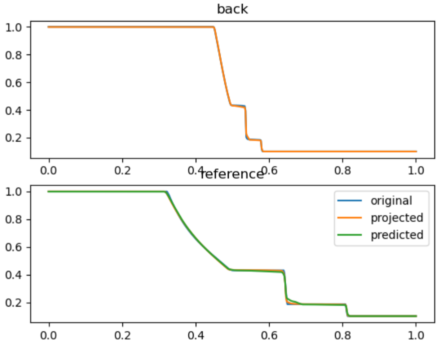


Results: online with POD-NN

Singular values decay for original problem (blue) and transformed problem (orange)

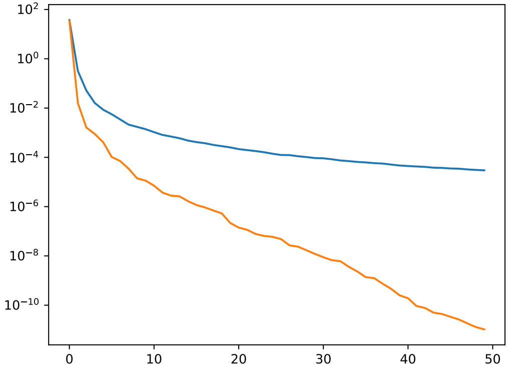


Multilayer Perceptron Regression for θ

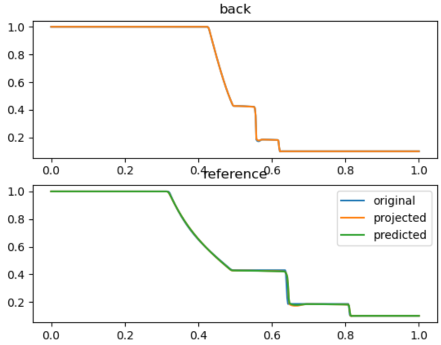


Results: online with POD-NN

Singular values decay for original problem (blue) and transformed problem (orange)

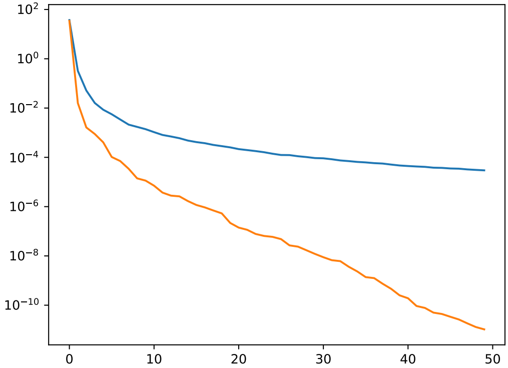


Multilayer Perceptron Regression for θ

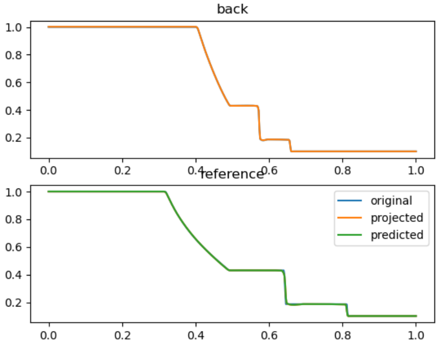


Results: online with POD-NN

Singular values decay for original problem (blue) and transformed problem (orange)

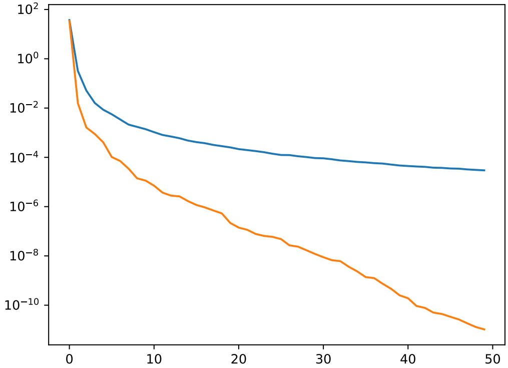


Multilayer Perceptron Regression for θ

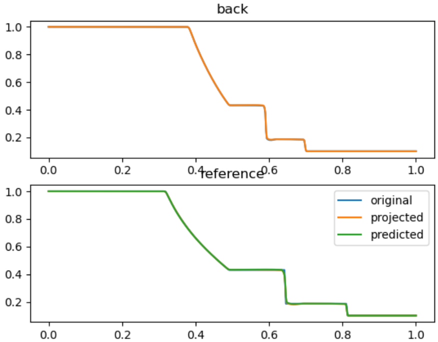


Results: online with POD-NN

Singular values decay for original problem (blue) and transformed problem (orange)

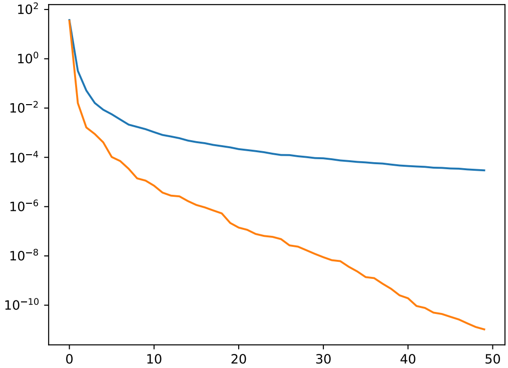


Multilayer Perceptron Regression for θ

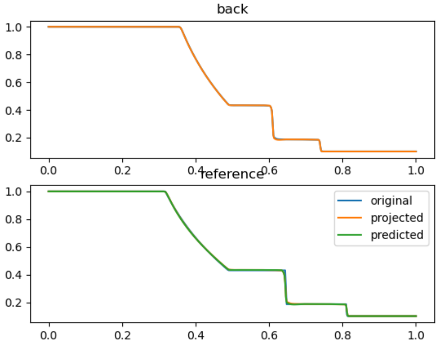


Results: online with POD-NN

Singular values decay for original problem (blue) and transformed problem (orange)

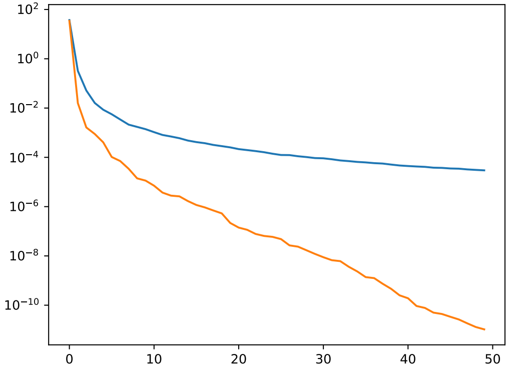


Multilayer Perceptron Regression for θ

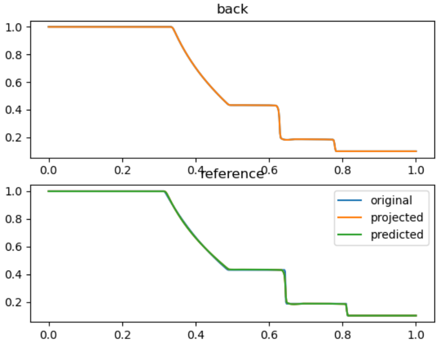


Results: online with POD-NN

Singular values decay for original problem (blue) and transformed problem (orange)

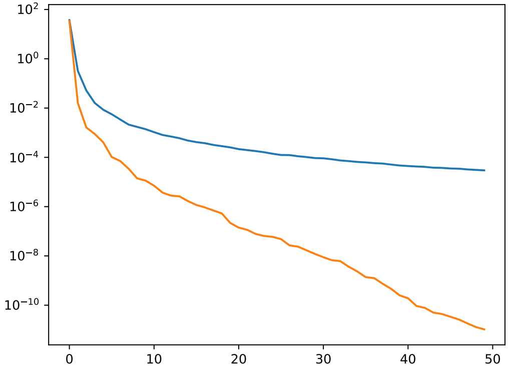


Multilayer Perceptron Regression for θ

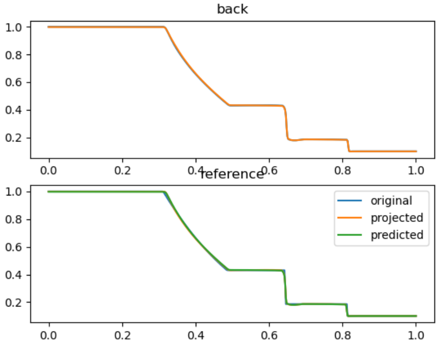


Results: online with POD-NN

Singular values decay for original problem (blue) and transformed problem (orange)

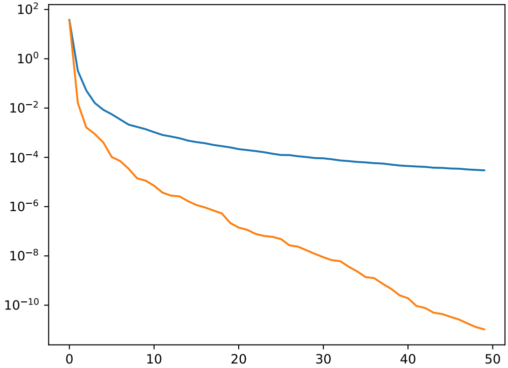


Multilayer Perceptron Regression for θ

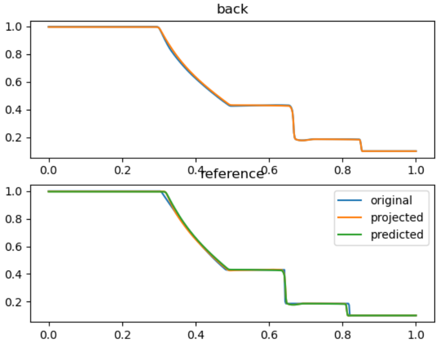


Results: online with POD-NN

Singular values decay for original problem (blue) and transformed problem (orange)

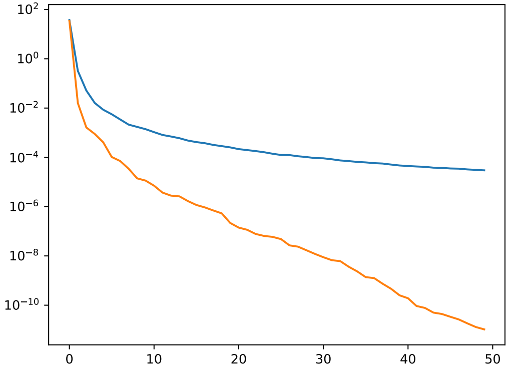


Multilayer Perceptron Regression for θ

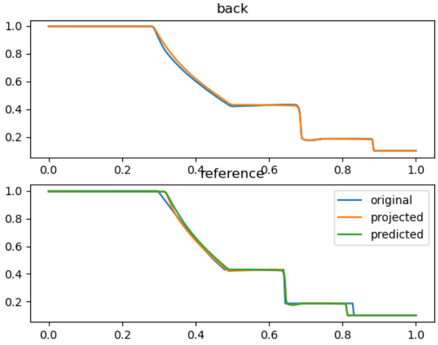


Results: online with POD-NN

Singular values decay for original problem (blue) and transformed problem (orange)



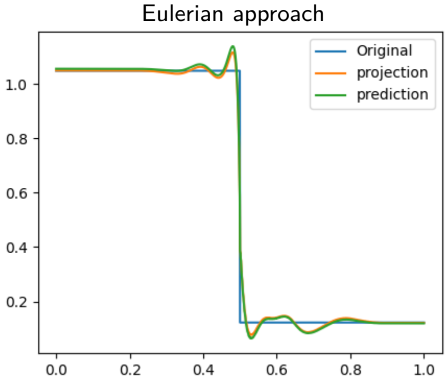
Multilayer Perceptron Regression for θ

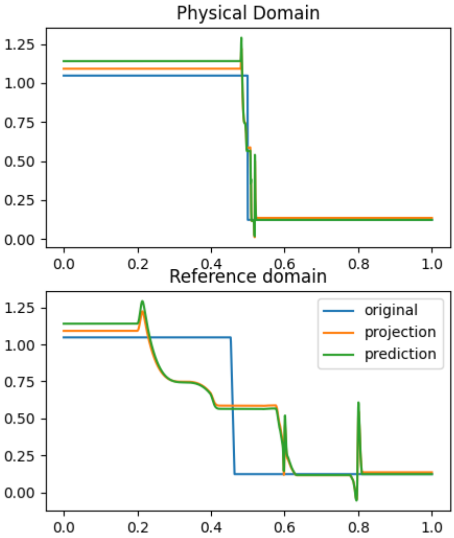


Parameter dependent Sod: Eulerian vs ALE

Parameters

- $\rho_L \in [0.7, 1.3]$, $\rho_R = [0.1, 0.15]$
- $p_L \in [0.7, 1.3]$, $p_R = [0.05, 0.15]$

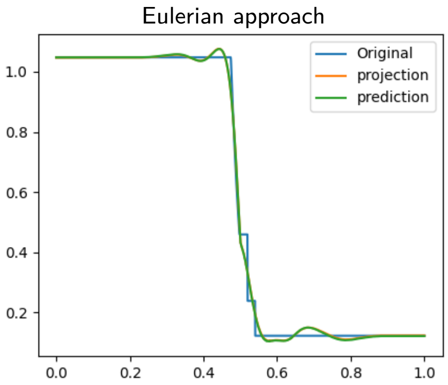


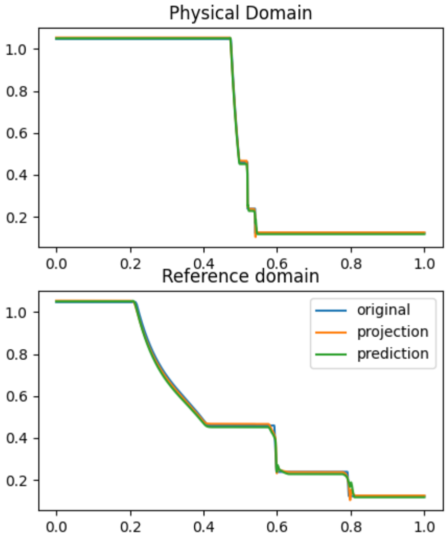


Parameter dependent Sod: Eulerian vs ALE

Parameters

- $\rho_L \in [0.7, 1.3]$, $\rho_R = [0.1, 0.15]$
- $p_L \in [0.7, 1.3]$, $p_R = [0.05, 0.15]$

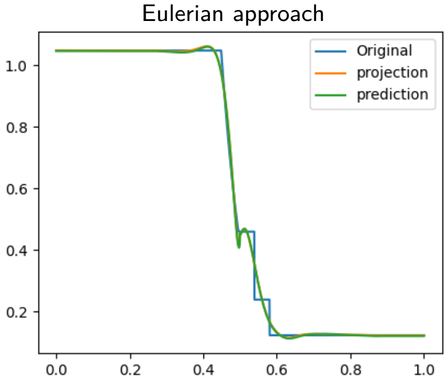


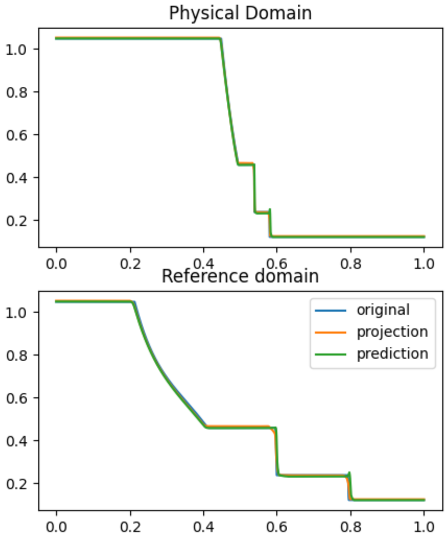


Parameter dependent Sod: Eulerian vs ALE

Parameters

- $\rho_L \in [0.7, 1.3]$, $\rho_R = [0.1, 0.15]$
- $p_L \in [0.7, 1.3]$, $p_R = [0.05, 0.15]$

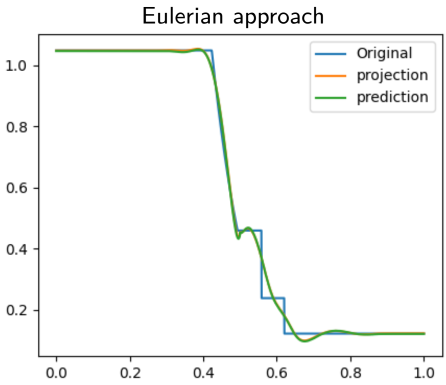


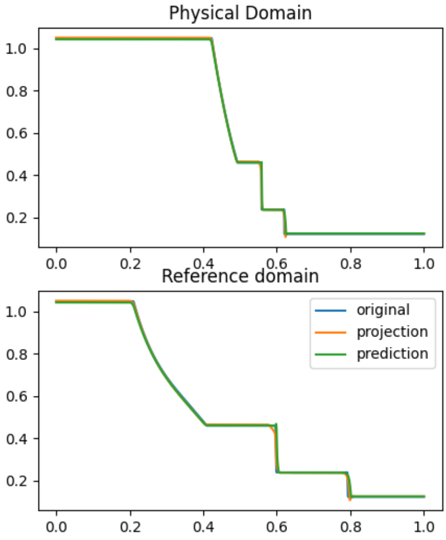


Parameter dependent Sod: Eulerian vs ALE

Parameters

- $\rho_L \in [0.7, 1.3]$, $\rho_R = [0.1, 0.15]$
- $p_L \in [0.7, 1.3]$, $p_R = [0.05, 0.15]$

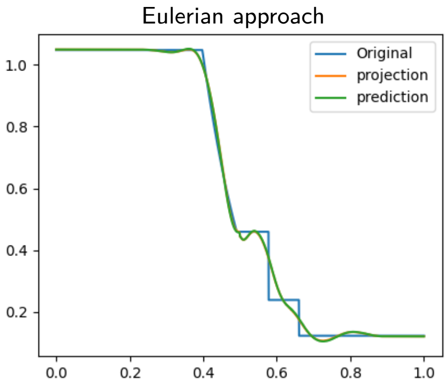


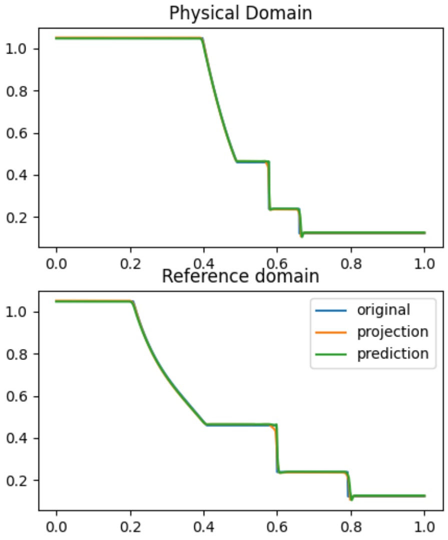


Parameter dependent Sod: Eulerian vs ALE

Parameters

- $\rho_L \in [0.7, 1.3]$, $\rho_R = [0.1, 0.15]$
- $p_L \in [0.7, 1.3]$, $p_R = [0.05, 0.15]$

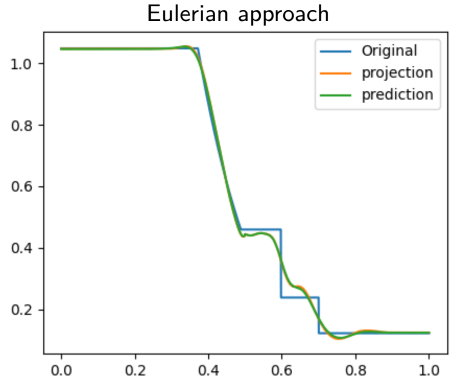


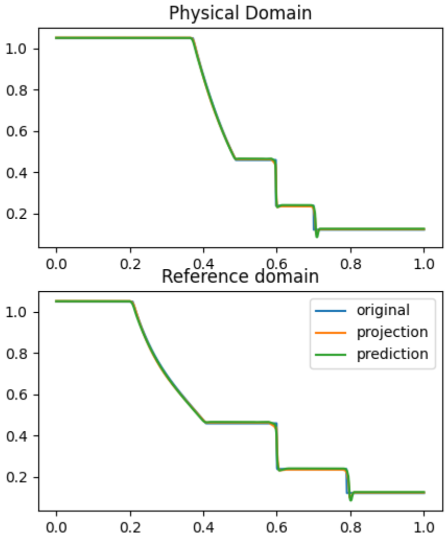


Parameter dependent Sod: Eulerian vs ALE

Parameters

- $\rho_L \in [0.7, 1.3],\ \rho_R = [0.1, 0.15]$
- $p_L \in [0.7, 1.3],\ p_R = [0.05, 0.15]$

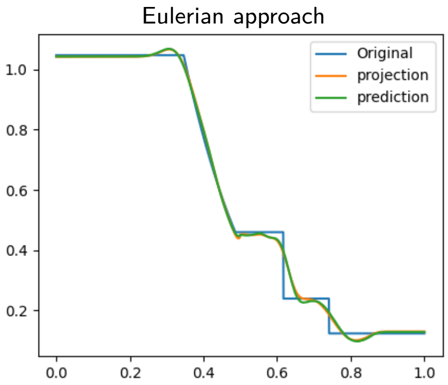


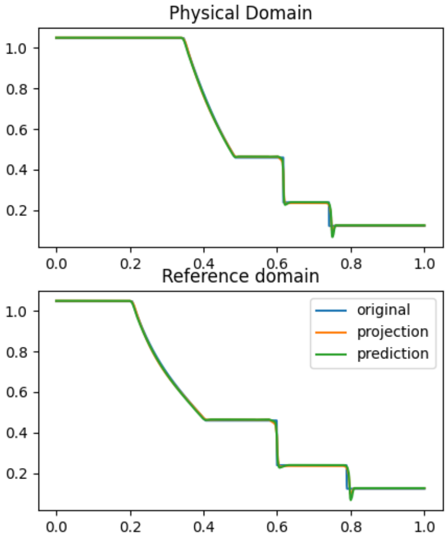


Parameter dependent Sod: Eulerian vs ALE

Parameters

- $\rho_L \in [0.7, 1.3]$, $\rho_R = [0.1, 0.15]$
- $p_L \in [0.7, 1.3]$, $p_R = [0.05, 0.15]$

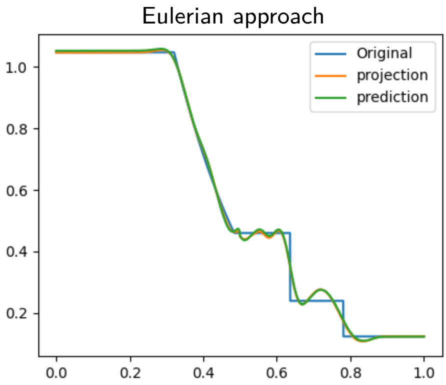


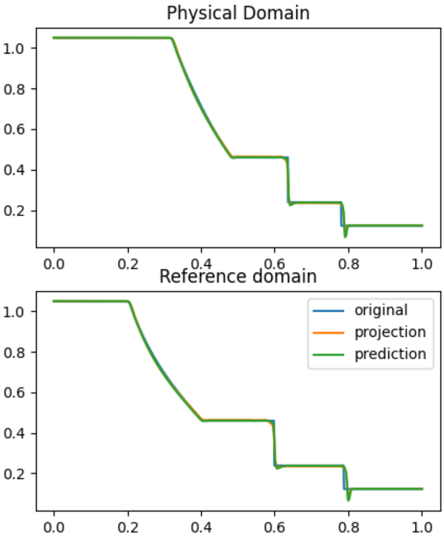


Parameter dependent Sod: Eulerian vs ALE

Parameters

- $\rho_L \in [0.7, 1.3],\ \rho_R = [0.1, 0.15]$
- $p_L \in [0.7, 1.3],\ p_R = [0.05, 0.15]$

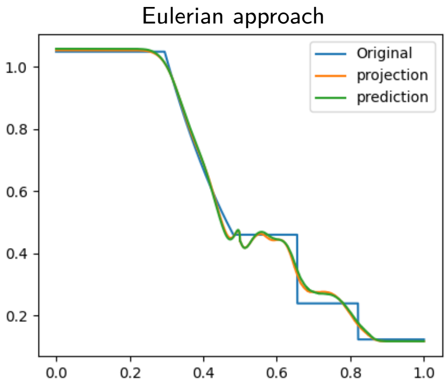


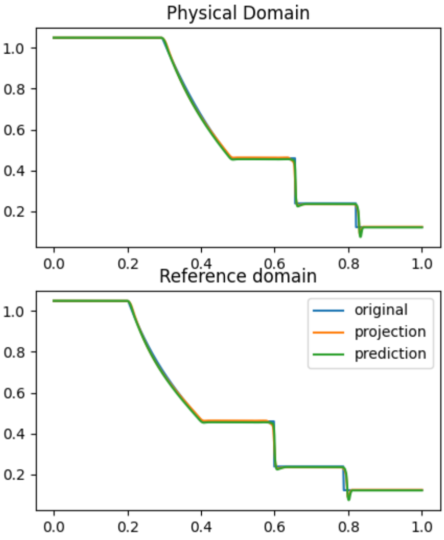


Parameter dependent Sod: Eulerian vs ALE

Parameters

- $\rho_L \in [0.7, 1.3]$, $\rho_R = [0.1, 0.15]$
- $p_L \in [0.7, 1.3]$, $p_R = [0.05, 0.15]$

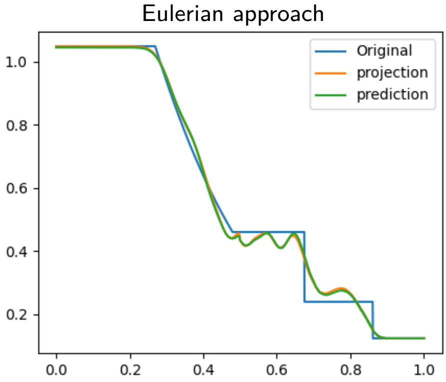


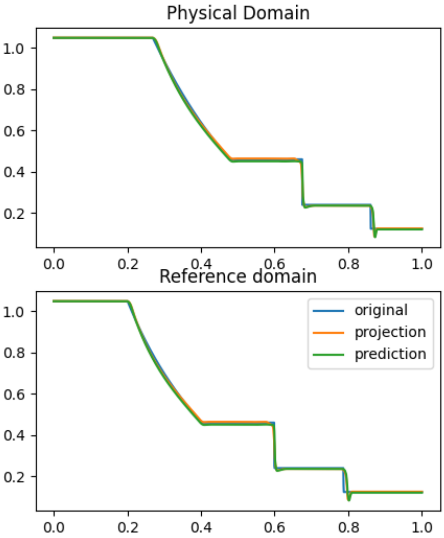


Parameter dependent Sod: Eulerian vs ALE

Parameters

- $\rho_L \in [0.7, 1.3]$, $\rho_R = [0.1, 0.15]$
- $p_L \in [0.7, 1.3]$, $p_R = [0.05, 0.15]$

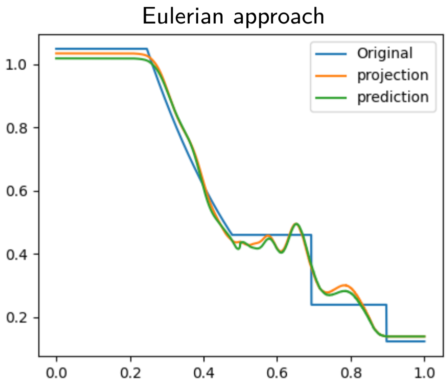


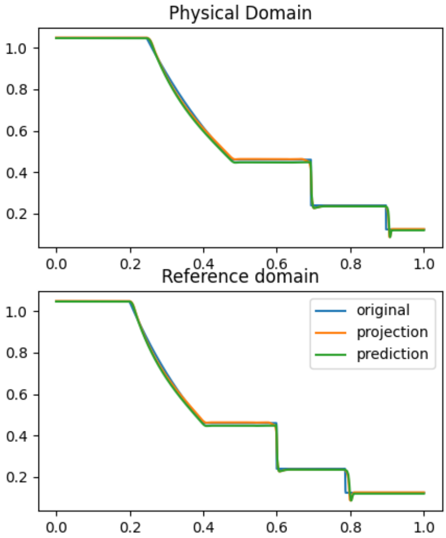


Parameter dependent Sod: Eulerian vs ALE

Parameters

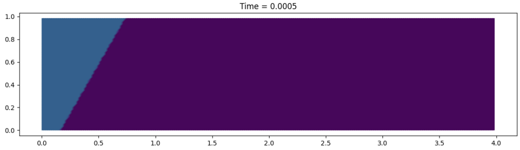
- $\rho_L \in [0.7, 1.3]$, $\rho_R = [0.1, 0.15]$
- $p_L \in [0.7, 1.3]$, $p_R = [0.05, 0.15]$



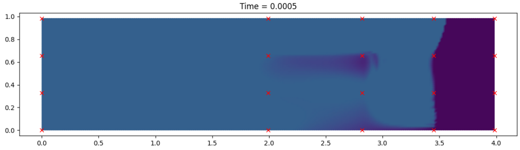


Double Mach Reflection 2D

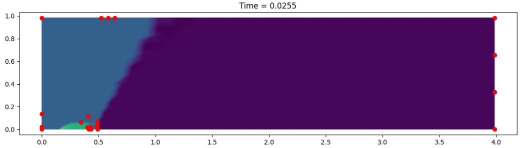
FOM on Ω



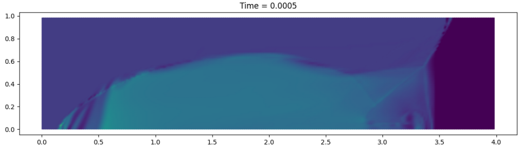
FOM on $\mathcal{R}$



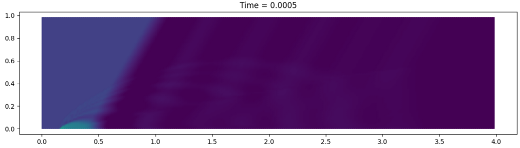
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

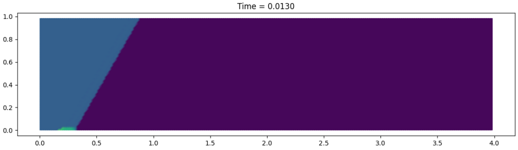


Problem data

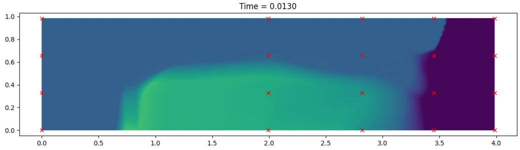
- Oblique shock hitting the bottom wall
- $\rho_L = 8,\ |\underline{u}_L| = 8.25,\ \arctan \underline{u} = \frac{\pi}{6},\ p_L = 116.5$
- $\rho_R = 1.4,\ u_R = 0,\ v_R = 0,\ p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

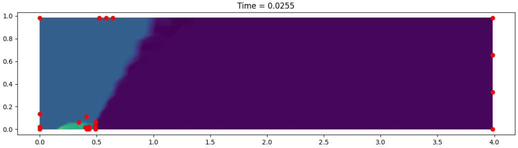
FOM on Ω



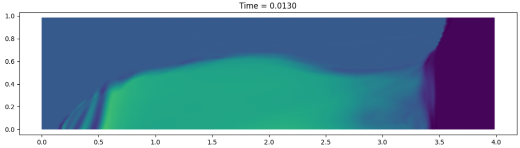
FOM on $\mathcal{R}$



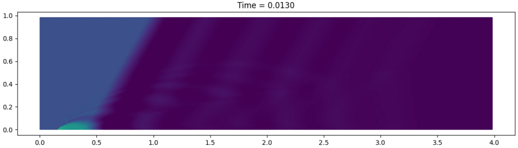
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

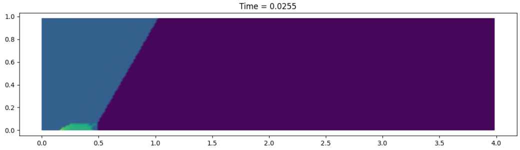


Problem data

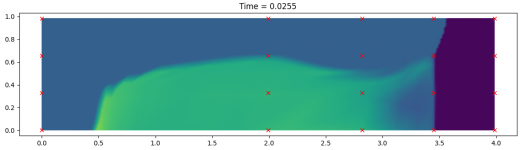
- Oblique shock hitting the bottom wall
- $\rho_L = 8,\ |\underline{u}_L| = 8.25,\ \arctan \underline{u} = \frac{\pi}{6},\ p_L = 116.5$
- $\rho_R = 1.4,\ u_R = 0,\ v_R = 0,\ p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

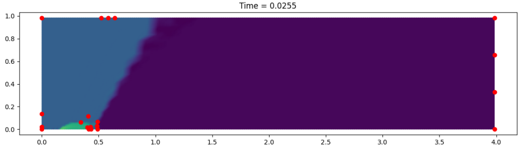
FOM on Ω



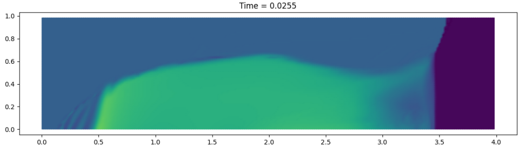
FOM on $\mathcal{R}$



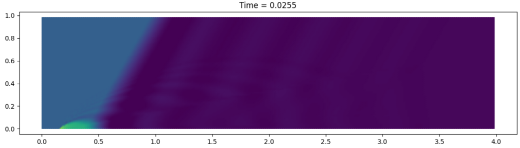
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

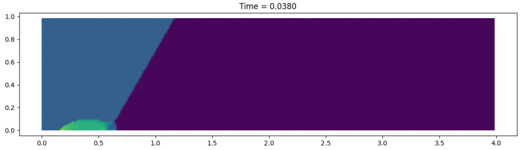


Problem data

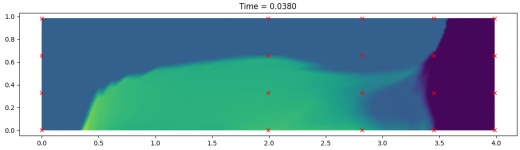
- Oblique shock hitting the bottom wall
- $\rho_L = 8, \; |\underline{u}_L| = 8.25, \; \arctan \underline{u} = \frac{\pi}{6}, \; p_L = 116.5$
- $\rho_R = 1.4, \; u_R = 0, \; v_R = 0, \; p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

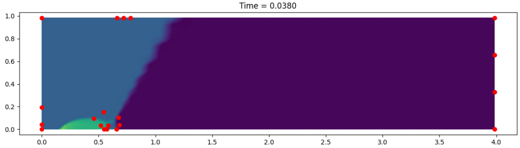
FOM on Ω



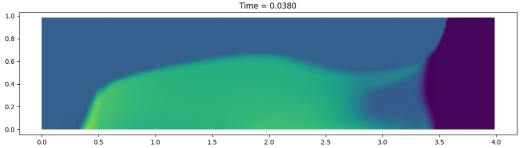
FOM on $\mathcal{R}$



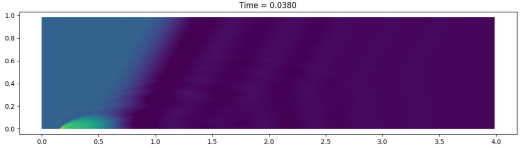
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

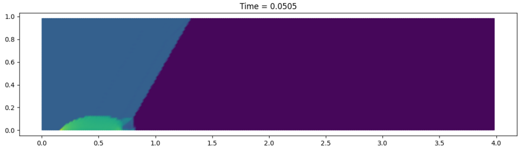


Problem data

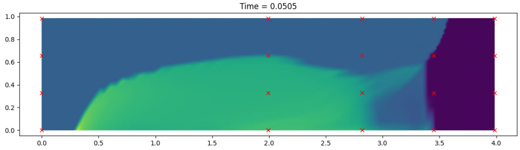
- Oblique shock hitting the bottom wall
- $\rho_L = 8,\ |\underline{u}_L| = 8.25,\ \arctan \underline{u} = \frac{\pi}{6},\ p_L = 116.5$
- $\rho_R = 1.4,\ u_R = 0,\ v_R = 0,\ p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

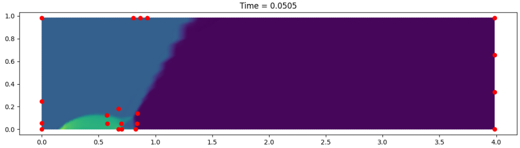
FOM on Ω



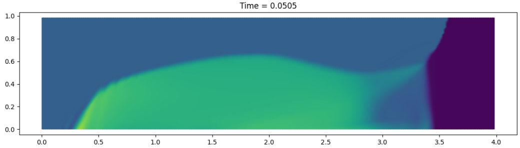
FOM on $\mathcal{R}$



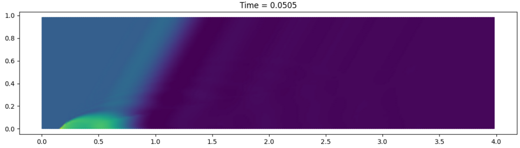
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

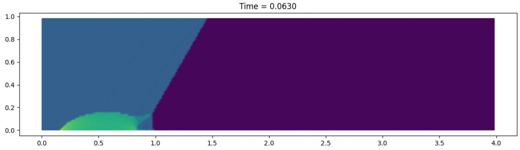


Problem data

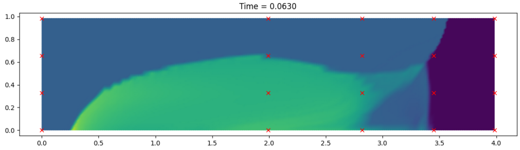
- Oblique shock hitting the bottom wall
- $\rho_L = 8, \ |\underline{u}_L| = 8.25, \ \arctan \underline{u} = \frac{\pi}{6}, \ p_L = 116.5$
- $\rho_R = 1.4, \ u_R = 0, \ v_R = 0, \ p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

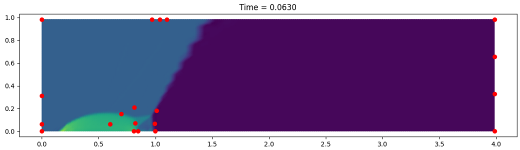
FOM on Ω



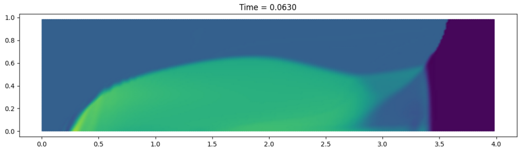
FOM on $\mathcal{R}$



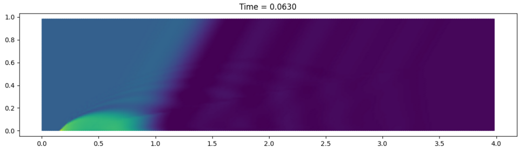
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

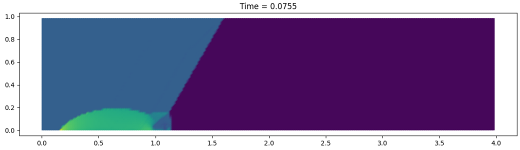


Problem data

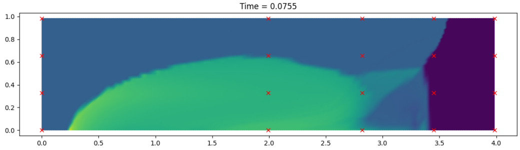
- Oblique shock hitting the bottom wall
- $\rho_L = 8,\ |\underline{u}_L| = 8.25,\ \arctan \underline{u} = \frac{\pi}{6},\ p_L = 116.5$
- $\rho_R = 1.4,\ u_R = 0,\ v_R = 0,\ p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

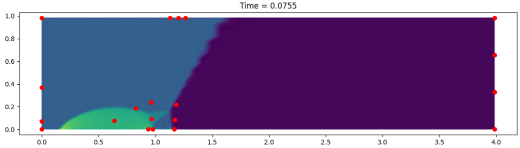
FOM on Ω



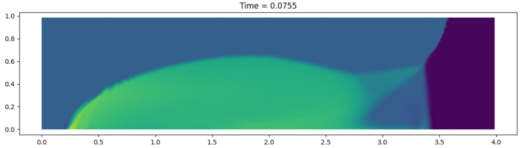
FOM on $\mathcal{R}$



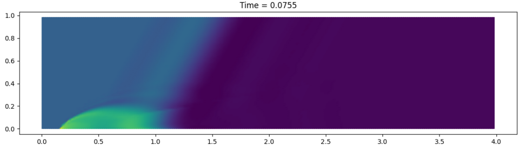
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

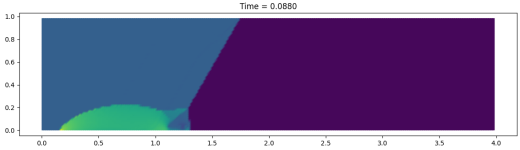


Problem data

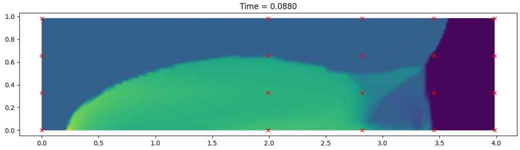
- Oblique shock hitting the bottom wall
- $\rho_L = 8,\ |\underline{u}_L| = 8.25,\ \arctan \underline{u} = \frac{\pi}{6},\ p_L = 116.5$
- $\rho_R = 1.4,\ u_R = 0,\ v_R = 0,\ p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

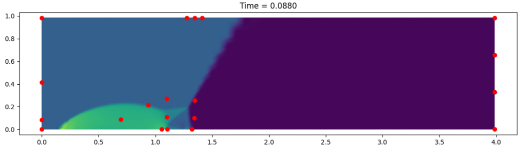
FOM on Ω



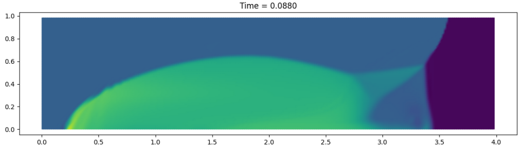
FOM on $\mathcal{R}$



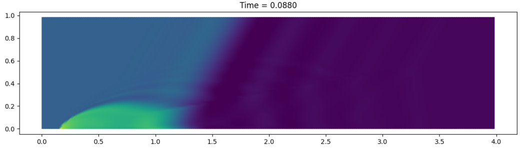
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

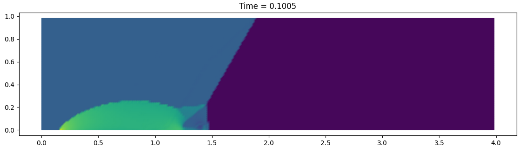


Problem data

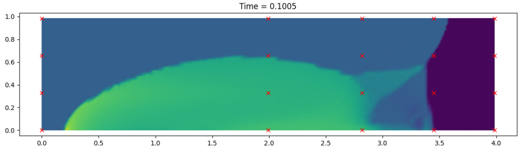
- Oblique shock hitting the bottom wall
- $\rho_L = 8, \; |\underline{u}_L| = 8.25, \; \arctan \underline{u} = \frac{\pi}{6}, \; p_L = 116.5$
- $\rho_R = 1.4, \; u_R = 0, v_R = 0, \; p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

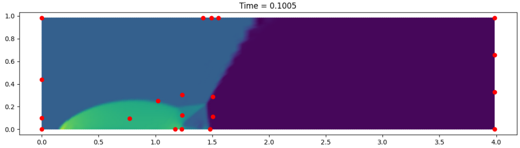
FOM on Ω



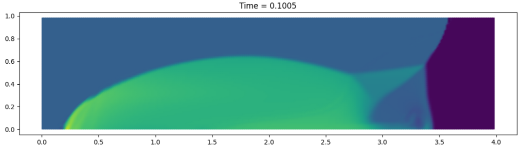
FOM on $\mathcal{R}$



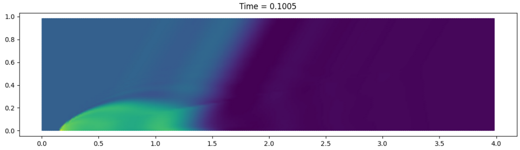
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

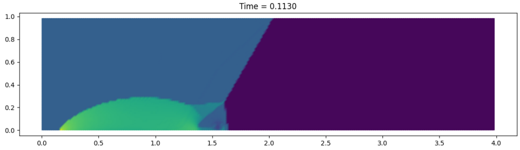


Problem data

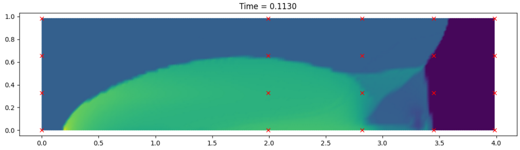
- Oblique shock hitting the bottom wall
- $\rho_L = 8,\ |\underline{u}_L| = 8.25,\ \arctan \underline{u} = \frac{\pi}{6},\ p_L = 116.5$
- $\rho_R = 1.4,\ u_R = 0, v_R = 0,\ p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

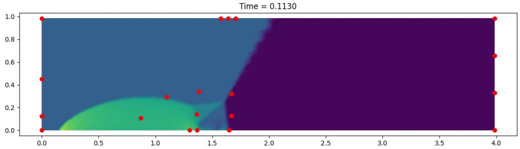
FOM on Ω



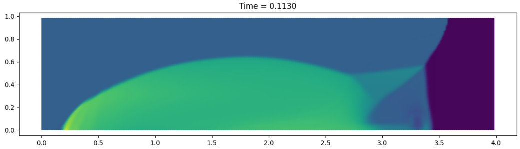
FOM on $\mathcal{R}$



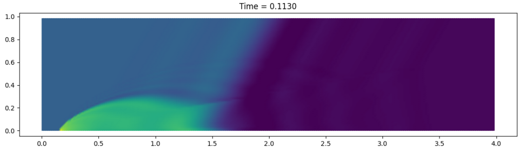
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

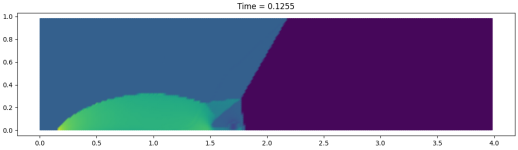


Problem data

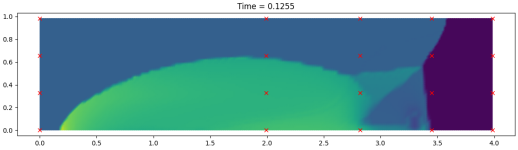
- Oblique shock hitting the bottom wall
- $\rho_L = 8, \ |\underline{u}_L| = 8.25, \ \arctan \underline{u} = \frac{\pi}{6}, \ p_L = 116.5$
- $\rho_R = 1.4, \ u_R = 0, v_R = 0, \ p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

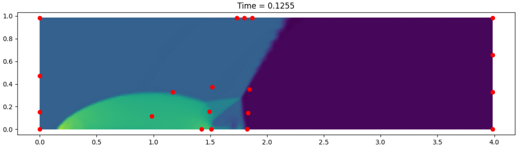
FOM on Ω



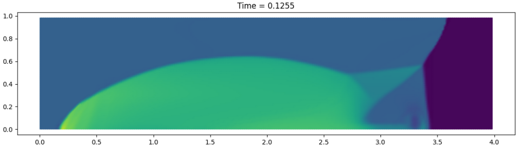
FOM on $\mathcal{R}$



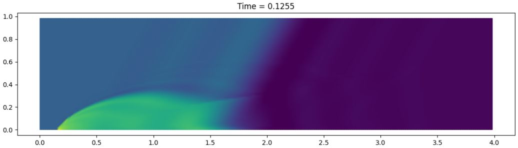
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

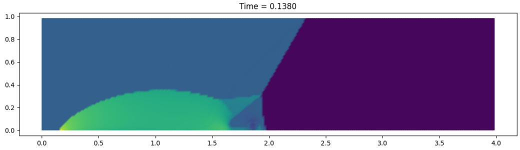


Problem data

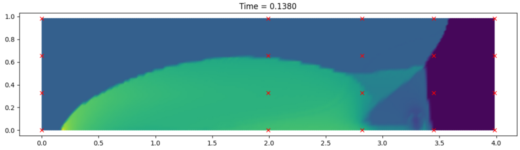
- Oblique shock hitting the bottom wall
- $\rho_L = 8,\ |\underline{u}_L| = 8.25,\ \arctan \underline{u} = \frac{\pi}{6},\ p_L = 116.5$
- $\rho_R = 1.4,\ u_R = 0,\ v_R = 0,\ p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

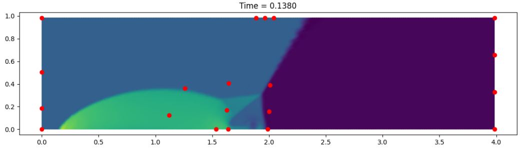
FOM on Ω



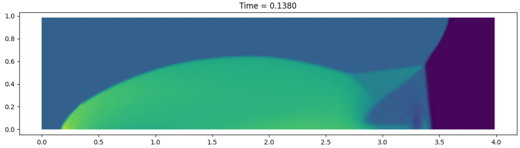
FOM on $\mathcal{R}$



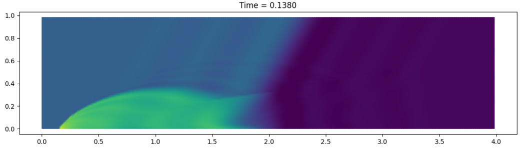
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

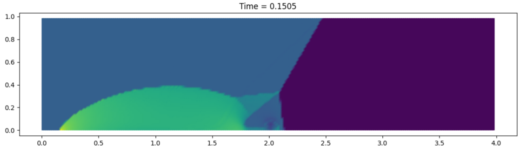


Problem data

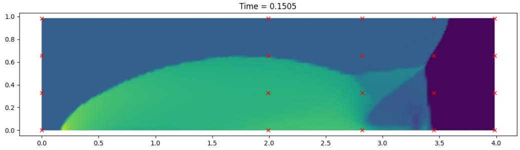
- Oblique shock hitting the bottom wall
- $\rho_L = 8$, $|\underline{u}_L| = 8.25$, $\arctan \underline{u} = \frac{\pi}{6}$, $p_L = 116.5$
- $\rho_R = 1.4$, $u_R = 0$, $v_R = 0$, $p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

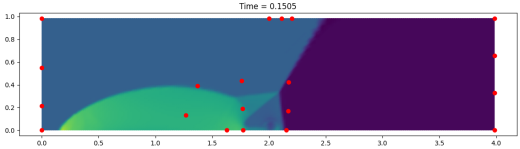
FOM on Ω



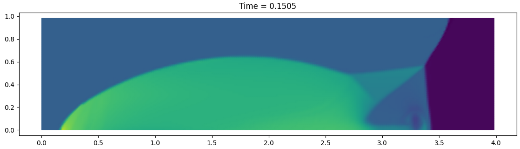
FOM on $\mathcal{R}$



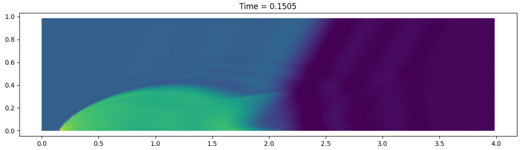
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

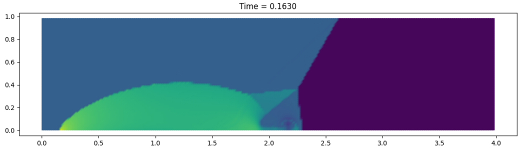


Problem data

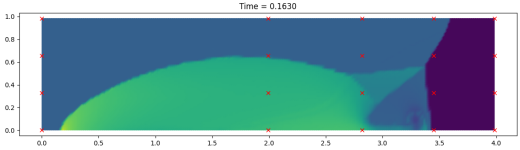
- Oblique shock hitting the bottom wall
- $\rho_L = 8$, $|\underline{u}_L| = 8.25$, $\arctan \underline{u} = \frac{\pi}{6}$, $p_L = 116.5$
- $\rho_R = 1.4$, $u_R = 0$, $v_R = 0$, $p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

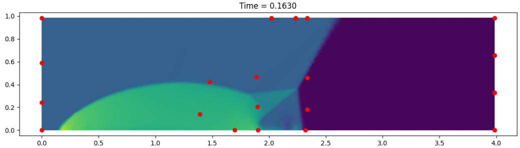
FOM on Ω



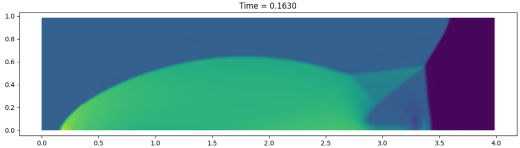
FOM on $\mathcal{R}$



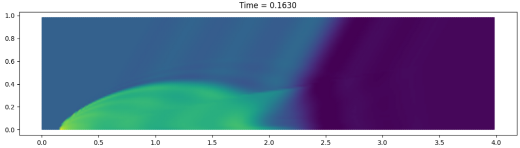
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

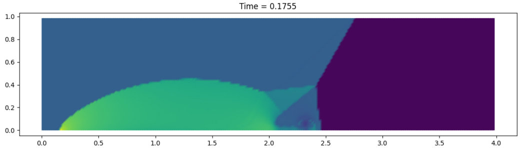


Problem data

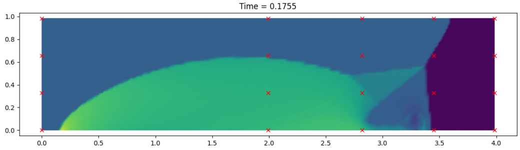
- Oblique shock hitting the bottom wall
- $\rho_L = 8, \; |\underline{u}_L| = 8.25, \; \arctan \underline{u} = \frac{\pi}{6}, \; p_L = 116.5$
- $\rho_R = 1.4, \; u_R = 0, \; v_R = 0, \; p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

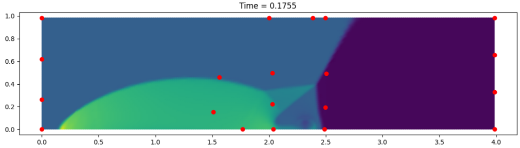
FOM on Ω



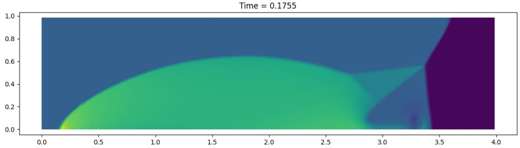
FOM on $\mathcal{R}$



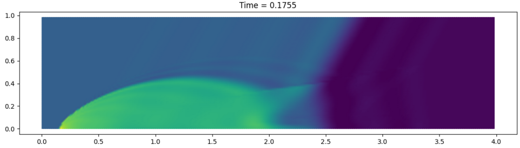
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

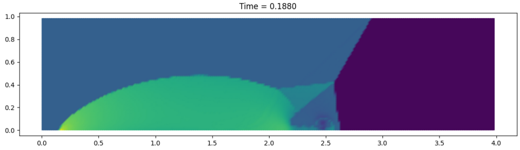


Problem data

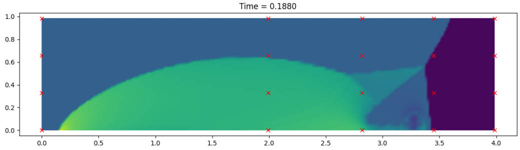
- Oblique shock hitting the bottom wall
- $\rho_L = 8,\ |\underline{u}_L| = 8.25,\ \arctan \underline{u} = \frac{\pi}{6},\ p_L = 116.5$
- $\rho_R = 1.4,\ u_R = 0, v_R = 0,\ p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

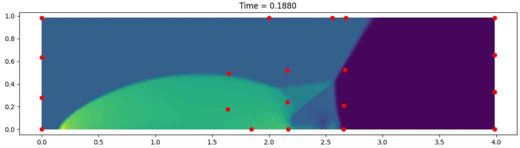
FOM on Ω



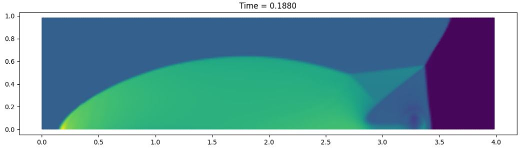
FOM on $\mathcal{R}$



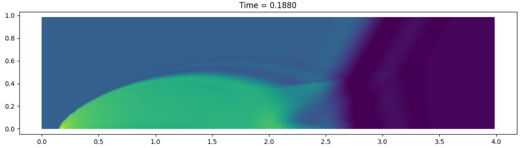
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

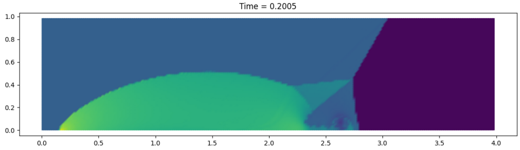


Problem data

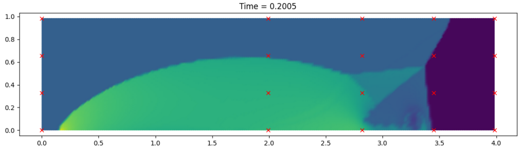
- Oblique shock hitting the bottom wall
- $\rho_L = 8$, $|\underline{u}_L| = 8.25$, $\arctan \underline{u} = \frac{\pi}{6}$, $p_L = 116.5$
- $\rho_R = 1.4$, $u_R = 0$, $v_R = 0$, $p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

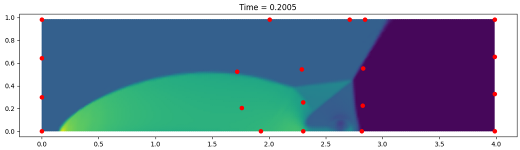
FOM on Ω



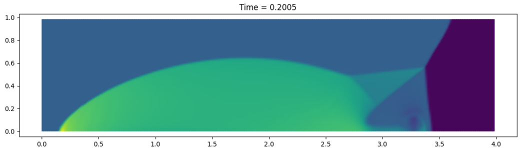
FOM on $\mathcal{R}$



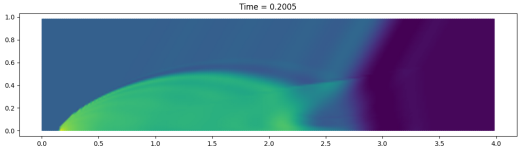
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

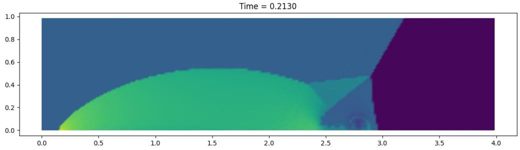


Problem data

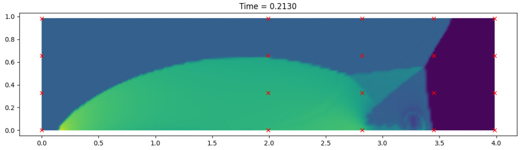
- Oblique shock hitting the bottom wall
- $\rho_L = 8$, $|\underline{u}_L| = 8.25$, $\arctan \underline{u} = \frac{\pi}{6}$, $p_L = 116.5$
- $\rho_R = 1.4$, $u_R = 0$, $v_R = 0$, $p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

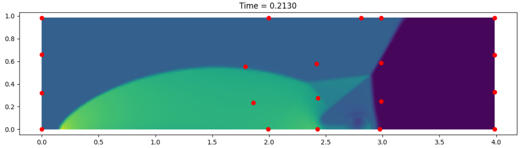
FOM on Ω



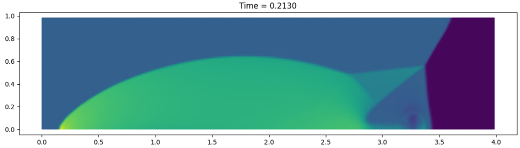
FOM on $\mathcal{R}$



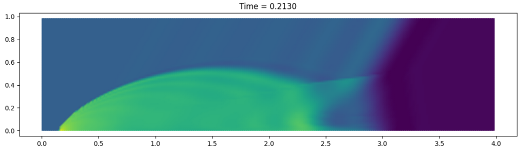
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

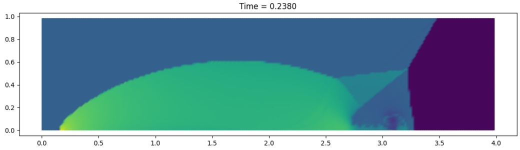


Problem data

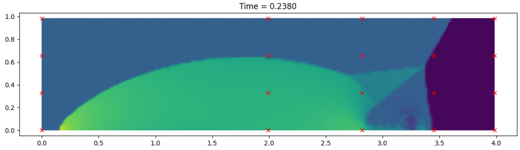
- Oblique shock hitting the bottom wall
- $\rho_L = 8, \; |\underline{u}_L| = 8.25, \; \arctan \underline{u} = \frac{\pi}{6}, \; p_L = 116.5$
- $\rho_R = 1.4, \; u_R = 0, \; v_R = 0, \; p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

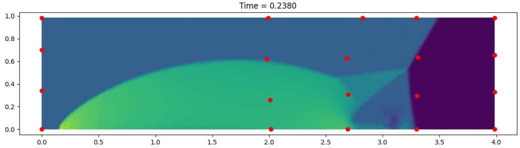
FOM on Ω



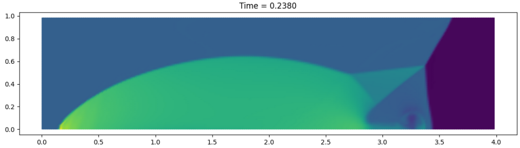
FOM on $\mathcal{R}$



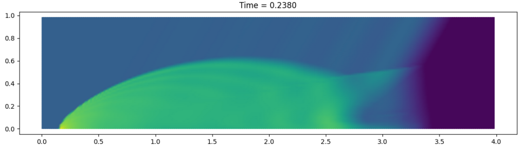
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω

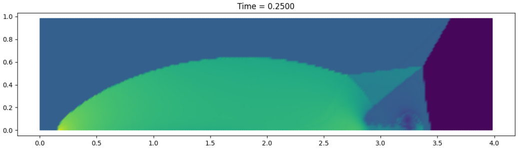


Problem data

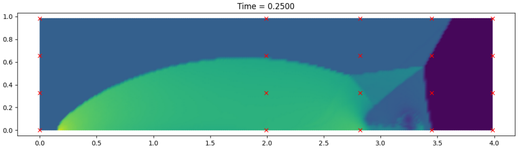
- Oblique shock hitting the bottom wall
- $\rho_L = 8,\ |\underline{u}_L| = 8.25,\ \arctan \underline{u} = \frac{\pi}{6},\ p_L = 116.5$
- $\rho_R = 1.4,\ u_R = 0,\ v_R = 0,\ p_R = 1$
- $N_{RB} = 12$

Double Mach Reflection 2D

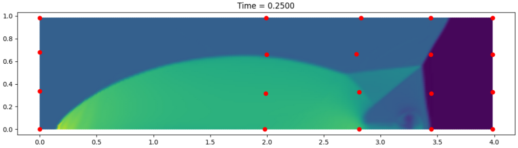
FOM on Ω



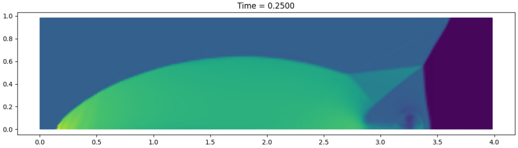
FOM on $\mathcal{R}$



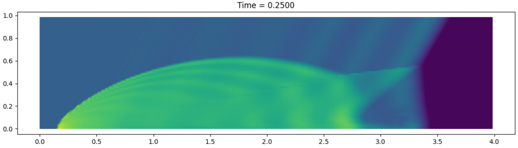
ROM ALE on Ω



ROM ALE on $\mathcal{R}$



ROM Eulerian on Ω



Problem data

- Oblique shock hitting the bottom wall
- $\rho_L = 8, \; |\underline{u}_L| = 8.25, \; \arctan \underline{u} = \frac{\pi}{6}, \; p_L = 116.5$
- $\rho_R = 1.4, \; u_R = 0, \; v_R = 0, \; p_R = 1$
- $N_{RB} = 12$

Table of contents

Graph Neural Network on vanishing viscosity solutions arxiv:2308.03378

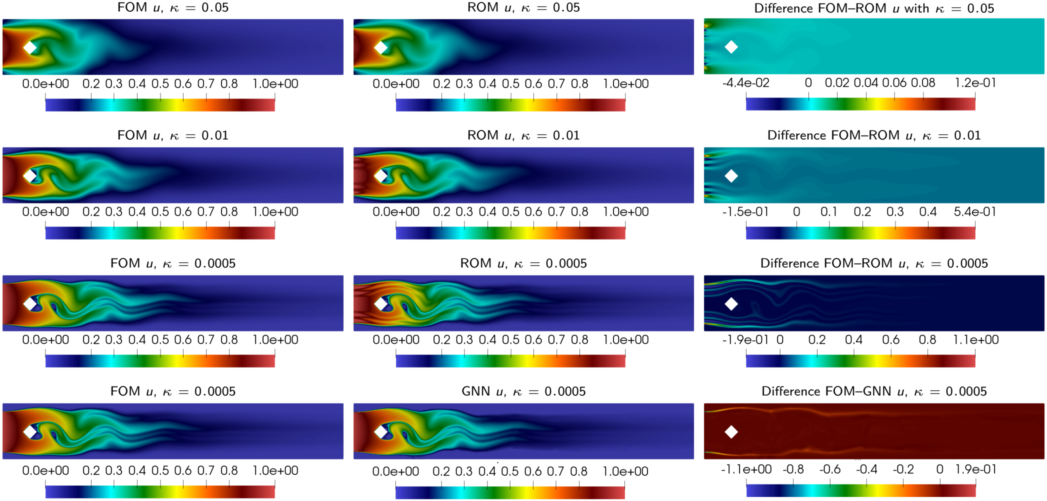


Figure: **VV.** Scalar concentration advected by incompressible flow for $i = 99$. Comparison of ROM approach at different viscosity levels $\kappa \in \{0.05, 0.01, 0.0005\}$ and GNN for $\kappa = 0.0005$.

Table of contents

Extensions and limitations

Limitations

- Beginning of the simulation still tricky (singularity not handled)
- Chaotic simulations
- Extrapolatory regime
- Delicate optimization

Extensions

- ALE online solver for new approach
- Improve optimization process in presence of multiple parameters
- Local ROM for different regimes
- Extend to other contexts the GNN approach

Extensions and limitations

Limitations

- Beginning of the simulation still tricky (singularity not handled)
- Chaotic simulations
- Extrapolatory regime
- Delicate optimization

Extensions

- ALE online solver for new approach
- Improve optimization process in presence of multiple parameters
- Local ROM for different regimes
- Extend to other contexts the GNN approach

Thank you for the attention!

Residual Distribution

- High order
- FE based
- Compact stencil
- Explicit
- Can recast some other FV, FE, FD, DG schemes [1]

[1] R. Abgrall. Computational Methods in Applied Mathematics, 2018

Residual Distribution

- High order
- FE based
- Compact stencil
- Explicit
- Can recast some other FV, FE, FD, DG schemes [1]

$$\partial_t U + \nabla \cdot F(U) = 0 \tag{2}$$

$$V_h = \{U \in L^2(\Omega_h, \mathbb{R}^D) \cap \mathcal{C}^0(\Omega_h),\ U|_K \in \mathbb{P}^k,\ \forall K \in \Omega_h\}. \tag{3}$$

$$U_h = \sum_{\sigma \in D_\mathcal{N}} U_\sigma \varphi_\sigma = \sum_{K \in \Omega_h} \sum_{\sigma \in K} U_\sigma \varphi_\sigma|_K \tag{4}$$

[1] R. Abgrall. Computational Methods in Applied Mathematics, 2018

Residual Distribution - Spatial Discretization

1. Define $\forall K \in \Omega_h$ a fluctuation term (total residual) $\phi^K = \int_K \nabla \cdot F(U) dx$
2. Define a nodal residual ϕ^K_σ $\forall \sigma \in K$:

$$\phi^K = \sum_{\sigma \in K} \phi^K_\sigma, \quad \forall K \in \Omega_h. \tag{5}$$

3. The resulting scheme is

$$U^{n+1}_\sigma - U^n_\sigma + \Delta t \sum_{K|\sigma \in K} \phi^K_\sigma = 0, \quad \forall \sigma \in D_{\mathcal{N}}. \tag{6}$$

Residual Distribution

- High order
- Easy to code
- FE based
- Compact stencil
- No need of Riemann solver
- No need of conservative variables
- Can recast some other FV, FE schemes

Residual Distribution

- High order
- Easy to code
- FE based
- Compact stencil
- No need of Riemann solver
- No need of conservative variables
- Can recast some other FV, FE schemes

$$\partial_t U + \nabla \cdot A(U) = S(U) \tag{7}$$

$$V_h = \{U \in L^2(\Omega_h, \mathbb{R}^D) \cap \mathcal{C}^0(\Omega_h),\ U|_K \in \mathbb{P}^k,\ \forall K \in \Omega_h\}. \tag{8}$$

$$U_h = \sum_{\sigma \in D_{\mathcal{N}}} U_\sigma \varphi_\sigma = \sum_{K \in \Omega_h} \sum_{\sigma \in K} U_\sigma \varphi_\sigma|_K \tag{9}$$

Residual Distribution - Spatial Discretization

Focus on steady case.

1. Define $\forall K \in \Omega_h$ a fluctuation term (total residual) $\phi^K = \int_K \nabla \cdot A(U) - S(U) dx$
2. Define a nodal residual $\phi^K_\sigma \; \forall \sigma \in K$:

$$\phi^K = \sum_{\sigma \in K} \phi^K_\sigma, \quad \forall K \in \Omega_h. \tag{10}$$

   Often done assigning $\phi^K_\sigma = \beta^K_\sigma \phi^K$, where must hold that

$$\sum_{\sigma \in K} \beta^K_\sigma = \mathrm{Id}. \tag{11}$$

3. The resulting scheme is

$$\sum_{K|\sigma \in K} \phi^K_\sigma = 0, \quad \forall \sigma \in D_{\mathcal{N}}. \tag{12}$$

   This will be called residual distribution scheme.

Residual distribution - Choice of the scheme

How to split total residuals into nodal residuals $\Rightarrow$ choice of the scheme.

$$
\begin{aligned}
&\phi_\sigma^{K,LxF}(U_h) = \int_K \varphi_\sigma \left(\nabla \cdot A(U_h) - S(U_h)\right) dx + \alpha_K (U_\sigma - \overline{U}_h^K),\\
&\overline{U}_h^K = \int_K U_h, \quad \alpha_K = \max_{e \text{ edge } \in K} \left(\rho_S \left(\nabla A(U_h) \cdot \mathbf{n}_e\right)\right),\\
&\beta_\sigma^K(U_h) = \max\left(\frac{\Phi_\sigma^{K,LxF}}{\Phi^K}, 0\right) \left(\sum_{j \in K} \max\left(\frac{\Phi_j^{K,LxF}}{\Phi^K}, 0\right)\right)^{-1},\\
&\phi_\sigma^{*,K} = (1-\Theta)\beta_\sigma^K \phi_\sigma^K + \Theta \Phi_\sigma^{K,LxF}, \quad \Theta = \frac{|\Phi^K|}{\sum_{j\in K} |\Phi_j^{K,LxF}|},\\
&\phi_\sigma^K = \beta_\sigma^K \phi_\sigma^{*,K} + \sum_{e|\text{edge of } K} \theta h_e^2 \int_e [\nabla U_h] \cdot [\nabla \varphi_\sigma] d\Gamma.
\end{aligned}
\tag{13}
$$

Error estimator

Additional hypothesis:

- $Id + \Delta t\mathcal{L}$ is Liptschitz continuous with constant $C > 0$,
- There are N'_{EIM} extra functions and functionals that capture the evolution of the solutions. (experimentally not so strict),
- Initial conditions are exactly represented in the reduced basis RB.

Total error estimator:

- EIM error, estimated by other N'_{EIM} basis functions f and functional τ iterating the EIM procedure after the stop, cost $\mathcal{O}(N'_{EIM})$,
- RB error given by the Lipschitz constant times residual of the small system,
- additionally one can add the projection error of the initial condition when not in RB.

Empirical interpolation method (EIM)

INPUT: $\mathcal{L}^n(U^n, \boldsymbol{\mu}, t^n)$, for $\boldsymbol{\mu} \in \mathcal{P}_h$, $n \leq N_t$

OUTPUT: $EIM = (\tau_k, f_k)_{k=1}^{N_{EIM}}$ where functions $f_k \in \mathbb{R}^{\mathcal{N}}$ and $\tau_k \in (\mathbb{R}^{\mathcal{N}})'$ (Examples of τ_k are point evaluations)

- Greedy iterative procedure
- At each step chooses the worst approximated function via an error estimator $\mathcal{L}^{worst} = \arg\max_{\mathcal{L}} ||\mathcal{L} - \sum_{k=1}^{N_{EIM}} \tau_k(\mathcal{L}) f_k||$
- Maximise the functional τ on the function $\mathcal{L}^{worst}$ $\tau^{chosen} = \arg\max_{\tau} |\tau(\mathcal{L}^{worst})|$
- $EIM = EIM \cup (\tau^{chosen}, \mathcal{L}^{worst})$
- Stop when error is smaller than a tolerance

Proper orthogonal decomposition (POD)

INPUT: Collection of functions $\{f_j\}_{j=1}^N$

OUTPUT: Reduced basis spaces $RB = \underset{U|dim(U)=N_{POD}}{\arg\min} \sum_{j=1}^N ||f_j - \mathcal{P}_U(f_j)||_2$

- Based on SVD
- Prescribed tolerance to stop the algorithm
- Global optimizer of the problem

Greedy algorithm

INPUT: Collection of functions $\{f_j\}_{j=1}^N$

OUTPUT: Reduced basis space RB

- There is an error estimator (normally cheap) $\varepsilon_{RB}(f) \sim ||f - \mathcal{P}_{RB}(f)||$
- Iteratively choose the worst represented function $f^{worst} = \arg\max_f \varepsilon_{RB}(f)$
- Add f^{worst} to the RB space
- Stop up to a certain tolerance

MOR: Ingredients

- Discretized solution $u_{\mathcal{N}}(\cdot, t, \boldsymbol{\mu}) \in \mathbb{V}_{\mathcal{N}}$ for $t \in \mathbb{R}^+$, $\boldsymbol{\mu} \in \mathcal{P}$
- Solution manifold: $\mathcal{S} := \{u_{\mathcal{N}}(\cdot, t, \boldsymbol{\mu}) \in \mathbb{V}_{\mathcal{N}} : t \in \mathbb{R}^+, \boldsymbol{\mu} \in \mathcal{P}\}$
- Ansatz:

$$\mathcal{S} \approx \mathbb{V}_{N_{RB}} \subset \mathbb{V}_{\mathcal{N}}, \qquad N_{RB} \ll \mathcal{N} \tag{14}$$

- Example: diffusion equation $u_t + \mu u_{xx} = 0$ with $u_0 = \sin(x\pi)$

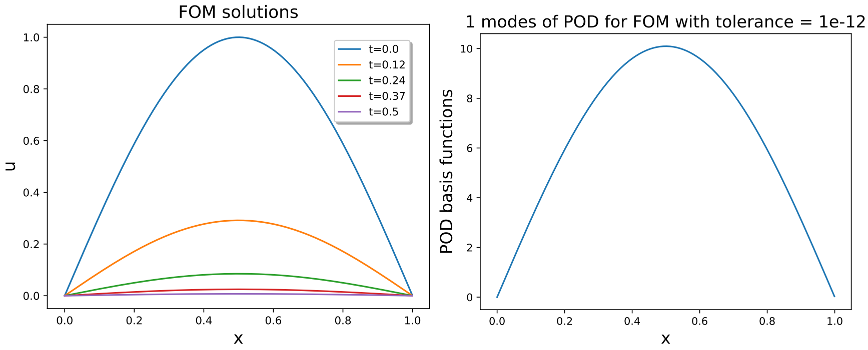


Figure: POD on a diffusion problem

MOR: Ingredients

Problem:

$$U^{n+1}(\boldsymbol{\mu}) - U^n(\boldsymbol{\mu}) + \mathcal{L}^n(U^n, \boldsymbol{\mu}) = 0, \quad U^n, U^{n+1} \in \mathbb{V}_{\mathcal{N}} \tag{15}$$

Objective:

$$\sum_{i=1}^{N_{RB}} \mathrm{u}_i^{n+1}(\boldsymbol{\mu})\psi_{RB}^i - \mathrm{u}_i^n(\boldsymbol{\mu})\psi_{RB}^i + \sum_{i=1}^{N_{RB}} \mathrm{L}^i(\mathrm{u}^n, \boldsymbol{\mu})\psi_{RB}^i = 0, \qquad \psi_{RB}^i \in \mathbb{V}_{\mathcal{N}}, \mathrm{u}^n, \mathrm{u}^{n+1} \in \mathbb{V}_{N_{RB}} \tag{16}$$

- EIM $\Rightarrow$ non–linear fluxes and scheme $\mathrm{L}^i(\mathrm{u}^n, \boldsymbol{\mu})$
- POD $\Rightarrow$ create the RB space and span the time evolution
- Greedy $\Rightarrow$ span the parameter space

Proper orthogonal decomposition (POD)

POD

INPUT:

- Collection of functions $\{f_j\}_{j=1}^N$

OUTPUT:

- Reduced basis spaces $\mathbb{V}_{N_{RB}} = \underset{U|dim(U)=N_{POD}}{\arg\min} \sum_{j=1}^N ||f_j - \mathcal{P}_U(f_j)||_2$

ALGORITHM:

- Based on SVD of matrix $\{f_j\}_{j=1}^N$
- We obtain (ordered) singular values and vectors
- Retain most energetic (largest singular value)
- Prescribed tolerance to stop the algorithm or maximum number of basis
- Related singular vectors gives $\mathbb{V}_{N_{RB}}$
- Global optimizer of the problem

Greedy algorithm

Greedy algorithm

INPUT:

- Collection of functions $\{f_j\}_{j=1}^N$
- Cheap error estimator $\varepsilon_{RB}(f) \sim ||f - \mathcal{P}_{RB}(f)||$

OUTPUT:

- Reduced basis space $\mathbb{V}_{N_{RB}}$

ALGORITHM:

- Iteratively choose the worst represented function $f^{worst} = \arg\max_f \varepsilon_{RB}(f)$
- Add f^{worst} to the $\mathbb{V}_{N_{RB}}$ space
- Stop up to a certain tolerance
- Not globally optimal, but locally optimal at each iteration (Greedy)

Empirical interpolation method (EIM)

Empirical interpolation method (EIM)

INPUT:

- $\mathcal{L}^n(U^n, \boldsymbol{\mu}, t^n)$, for $\boldsymbol{\mu} \in \mathcal{P}_h$, $n \leq N_t$

OUTPUT:

- $EIM = (\tau_k, f_k)_{k=1}^{N_{EIM}}$ where functions $f_k \in \mathbb{R}^{\mathcal{N}}$ and $\tau_k \in (\mathbb{R}^{\mathcal{N}})'$
  (τ_k, f_k) are often called magic points and functions, where τ_k are function evaluations

ALGORITHM:

- Greedy iterative procedure
- At each step chooses the worst approximated function via an error (estimator)
  $\mathcal{L}^{worst} = \arg\max_{\mathcal{L}} ||\mathcal{L} - \sum_{k=1}^{N_{EIM}} \tau_k(\mathcal{L}) f_k||$
- Maximise the functional τ on the function $\mathcal{L}^{worst}$ $\tau^{chosen} = \arg\max_{\tau} |\tau(\mathcal{L}^{worst})|$
- $EIM = EIM \cup (\tau^{chosen}, \mathcal{L}^{worst})$
- Stop when error is smaller than a tolerance

Offline Algorithm: PODEIM–Greedy [2]

PODEIM–Greedy

INITIALIZATION:

- EIM on $\mathcal{L}(U^n, \boldsymbol{\mu}_0, t^n)$ for $n \leq N_t$
- $\mathbb{V}_{N_{RB}} = POD(\{U^n(\boldsymbol{\mu}_0)\}_{n=0}^{N_t})$

ITERATION:

- Greedy algorithm spanning over the parameter space $\mathcal{P}_h$, with an error indicator $\varepsilon(\mathbf{U}(\boldsymbol{\mu}))$ where $\mathbf{U}(\boldsymbol{\mu}) \in \mathbb{R}^N \times \mathbb{R}^+$
- Choose worst parameter as $\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\mu} \in \mathcal{P}_h} \varepsilon(\mathbf{U}(\boldsymbol{\mu}))$
- Apply POD on time evolution of selected solution $POD_{add} = POD\left(\{U^n(\boldsymbol{\mu}^*)\}_{n=1}^{N_t}\right)$
- Update the $\mathbb{V}_{N_{RB}}$ with $\mathbb{V}_{N_{RB}} = POD\left(\mathbb{V}_{N_{RB}} \cup POD_{add}\right)$
- Update EIM basis function with $EIM_{space} = EIM_{space} \cup EIM(\{\mathcal{L}(U^n, \boldsymbol{\mu}^*, t^n)\}_{n=0}^{N_t})$

[2] B. Haasdonk and M. Ohlberger, in Hyperbolic problems: theory, numerics and applications, vol. 67, Amer. Math. Soc., 2009.

Online algorithm: PODEIM–Greedy

Reduced Order Model system

Solve the smaller system:

$$\sum_{i=1}^{N_{RB}} (\mathrm{u}_i^{n+1}(\boldsymbol{\mu}) - \mathrm{u}_i^{n}(\boldsymbol{\mu}))\psi_{RB}^i + \sum_{i=1}^{N_{RB}} \sum_{j=1}^{N_{EIM}} \tau_j(\mathcal{L}(U^n, \boldsymbol{\mu}))\Pi_{RB,i}(f_j)\psi_{RB}^i = 0$$

- $\Pi_{RB,i}(f_j)$ are the projection on $\mathbb{V}_{N_{RB}}$ of the EIM functions: offline
- $\tau_j(\mathcal{L}(U^n, \boldsymbol{\mu}))$ are inexpensive to compute, but depend on the method (for RD $\approx \mathcal{O}(d)$)
- MOR cost $\mathcal{O}(N_t N_{RB} N_{EIM})$ vs FOM cost $\mathcal{O}(N_t \mathcal{N})$
- Gain if $N_{RB}, N_{EIM} \ll \mathcal{N}$
- Error estimator

Learning of θ

Calibration map

- $\theta(\mu)$ tells us where a feature is (maximum point, steepest gradient)
- How to choose it? (second part)
- How to learn the map? (Offline we want to know the map in advance)
- Offline: optimization of θs on a training sample
- Generation of a regression map $\widehat{\theta}$

Piecewise linear regression for every timestep t^n

- If parameter domain is a grid $\Rightarrow$ Easy, fast
- Non–structured parameter domain $\Rightarrow$ Different algorithms, may be costly
- Precise if $|\mathcal{P}_h| \sim s^P$ with s big enough
- May not catch the nonlinear behavior and produce unreasonable results

Learning of θ

Calibration map

- $\theta(\mu)$ tells us where a feature is (maximum point, steepest gradient)
- How to choose it? (second part)
- How to learn the map? (Offline we want to know the map in advance)
- Offline: optimization of θs on a training sample
- Generation of a regression map $\widehat{\theta}$

Polynomial regression

- Hyperparameter p
- Risk of overfitting
- Can easily catch the nonlinear (polynomial) behavior
- Number of coefficients grows exponentially with p

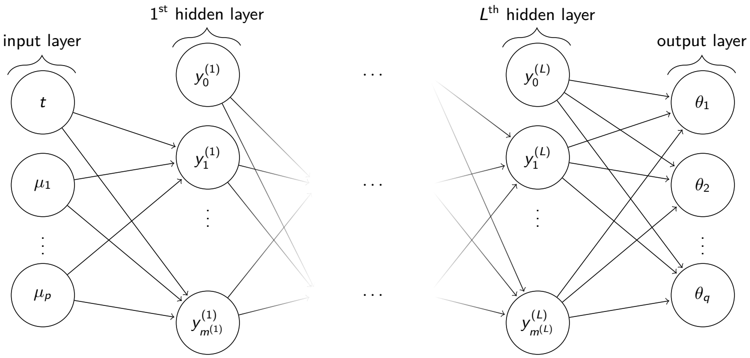
$$\theta(\boldsymbol{\mu}, t) \approx \sum_{|\alpha|\leq p} \beta_\alpha t^{\gamma_0} \prod_{i=1}^{p} \mu_i^{\gamma_i}$$

Learning of θ

Neural networks

- Why? Naturally nonlinear, we may not have a structured dictionary
- Which one? Multi-layer-perceptron, N layers ($[4, 10]$), M_n nodes ($[6, 20]$)

Multilayer perceptron



Travelling wave, time evolution solution

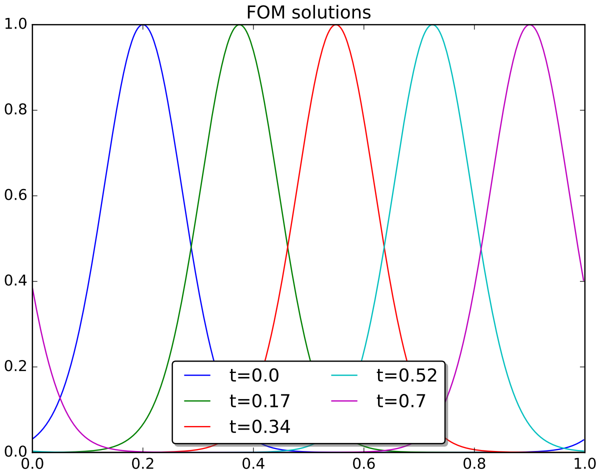


Figure: Solution of advection equation $\partial_t u + \partial_x u = 0$ with gaussian IC

Travelling wave, POD

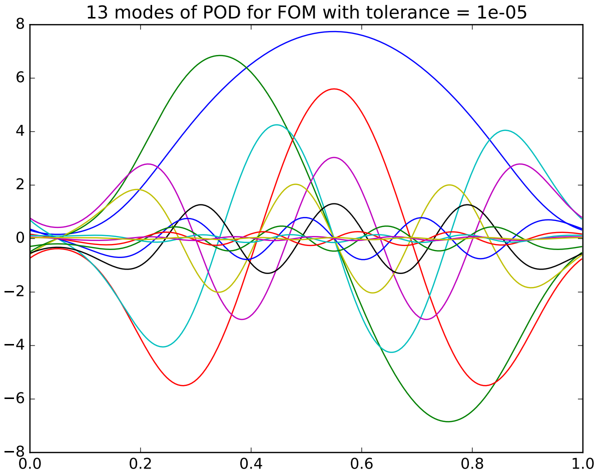


Figure: Solution of advection equation with wave IC

Travelling shock, time evolution solution, little diffusion

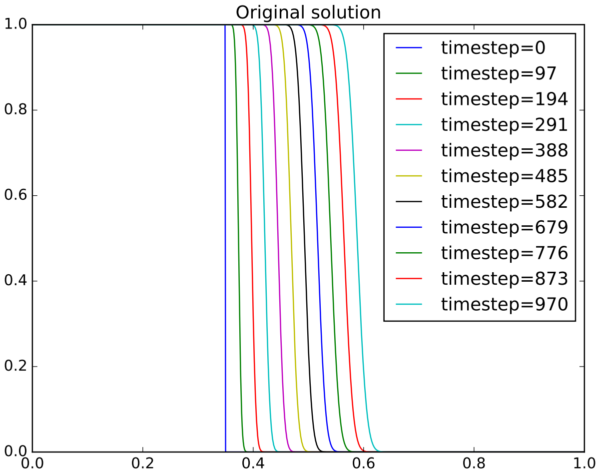


Figure: Solution of advection equation with shock IC

Travelling shock, POD, little diffusion

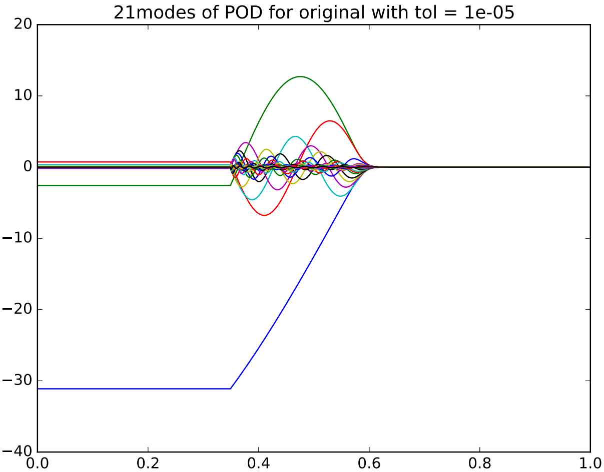


Figure: POD of time evolution of advection equation with shock IC

Travelling shock, time evolution solution, no diffusion

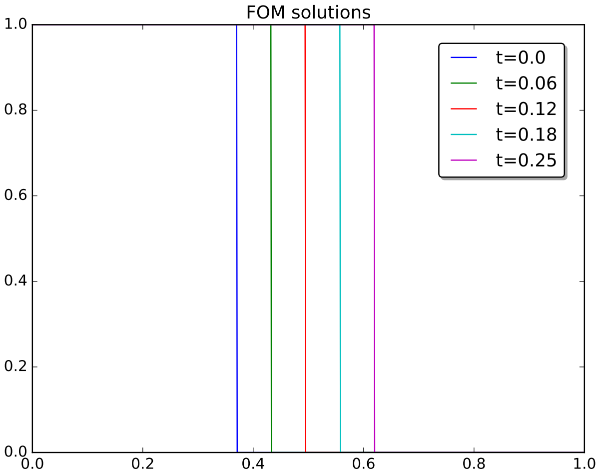


Figure: Solution of advection equation with shock IC

Travelling shock, POD, no diffusion

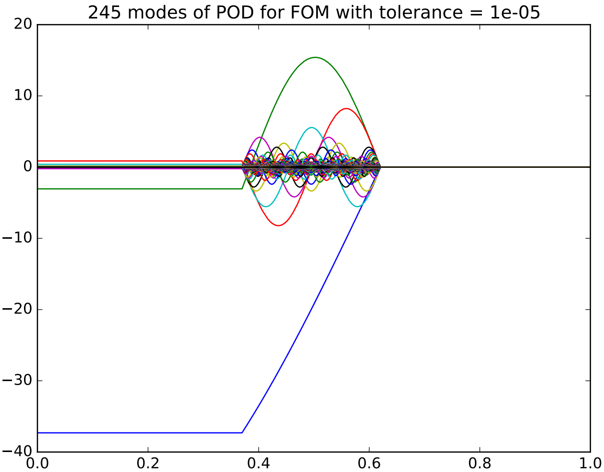


Figure: POD of time evolution of advection equation with shock IC

Common problems and properties

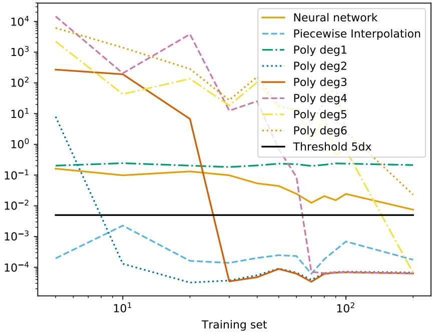
- As many basis functions as positions of the shock
- Slow decay of Kolmogorov N–width

$$d_N(\mathcal{S},\mathbb{V}) := \inf_{\mathbb{V}_N\subset\mathbb{V}} \sup_{f\in\mathcal{S}} \inf_{g\in\mathbb{V}_N} ||f-g||$$

- Non linear dependency leads to big EIM and RB space
- $1/2$ parameters problem (highly non linear dependence on parameters)

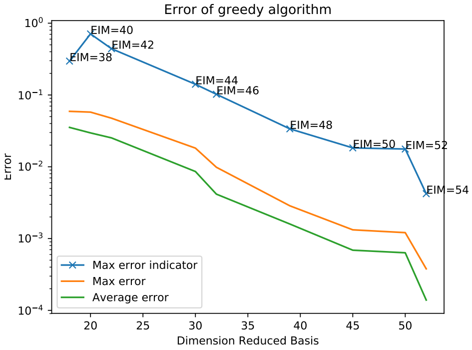
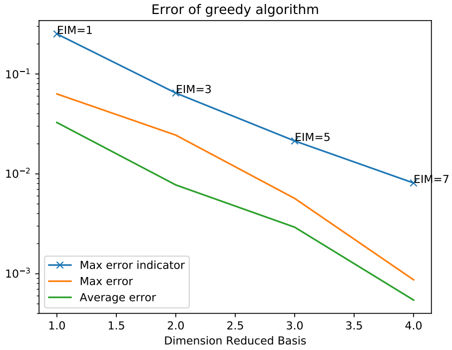
Advection: traveling wave

$$\begin{cases} u_t + \mu_0 u_x = 0, \; D = [0,1], \; T_{max} = 0.6, \text{ periodic BC} \\ u_0(x, \boldsymbol{\mu}) = e^{-\mu_1 (x-\mu_2)^2} \\ \mu_0 \sim \mathcal{U}([0,2]), \mu_1 \sim \mathcal{U}([500,1500]), \mu_2 \sim \mathcal{U}([0.1,0.3]) \end{cases}$$

Without calibration	With calibration: Regressions
	

Advection: traveling wave

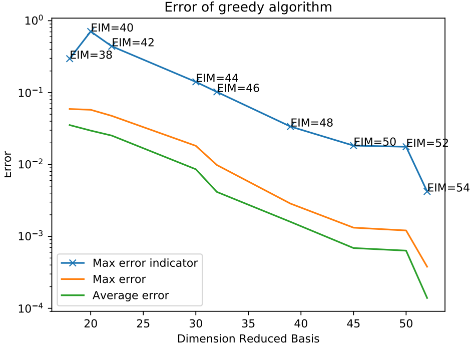
$$\begin{cases} u_t + \mu_0 u_x = 0,\ D = [0,1],\ T_{max} = 0.6, \text{ periodic BC} \\ u_0(x, \boldsymbol{\mu}) = e^{-\mu_1 (x-\mu_2)^2} \\ \mu_0 \sim \mathcal{U}([0,2]), \mu_1 \sim \mathcal{U}([500,1500]), \mu_2 \sim \mathcal{U}([0.1,0.3]) \end{cases}$$

Without calibration	With calibration: Poly2
	

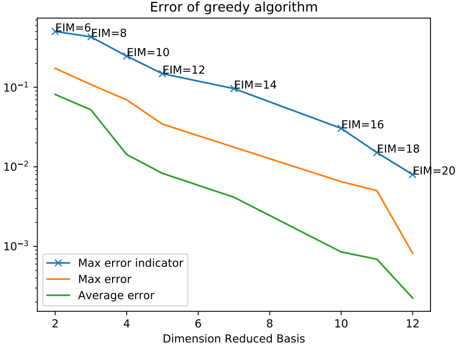
Advection: traveling wave

$$\begin{cases} u_t + \mu_0 u_x = 0,\ D = [0,1],\ T_{max} = 0.6, \text{ periodic BC} \\ u_0(x, \boldsymbol{\mu}) = e^{-\mu_1 (x-\mu_2)^2} \\ \mu_0 \sim \mathcal{U}([0,2]), \mu_1 \sim \mathcal{U}([500,1500]), \mu_2 \sim \mathcal{U}([0.1,0.3]) \end{cases}$$

Without calibration



With calibration: ANN



Advection: traveling wave

$$\begin{cases} u_t + \mu_0 u_x = 0, \; D = [0,1], \; T_{max} = 0.6, \text{ periodic BC} \\ u_0(x, \boldsymbol{\mu}) = e^{-\mu_1 (x-\mu_2)^2} \\ \mu_0 \sim \mathcal{U}([0,2]), \mu_1 \sim \mathcal{U}([500,1500]), \mu_2 \sim \mathcal{U}([0.1, 0.3]) \end{cases}$$

Without calibration		With calibration: Poly2	
RB dim	52	RB dim	4
EIM dim	54	EIM dim	7
FOM time	191 s	FOM time	516 s
RB time	24 s	RB time	18 s
RB/FOM time	12%	RB/FOM time	3%

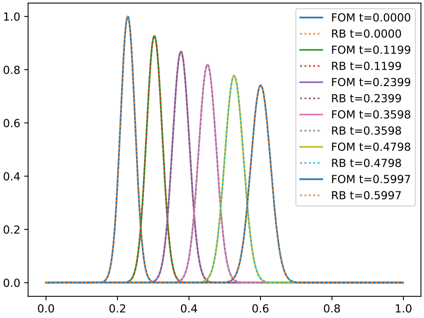
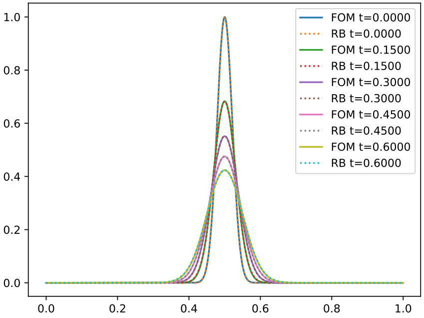
Advection: traveling wave

$$\begin{cases} u_t + \mu_0 u_x = 0, \; D = [0,1], \; T_{max} = 0.6, \text{ periodic BC} \\ u_0(x, \boldsymbol{\mu}) = e^{-\mu_1 (x-\mu_2)^2} \\ \mu_0 \sim \mathcal{U}([0,2]), \mu_1 \sim \mathcal{U}([500,1500]), \mu_2 \sim \mathcal{U}([0.1,0.3]) \end{cases}$$

Without calibration		With calibration: ANN	
RB dim	52	RB dim	12
EIM dim	54	EIM dim	20
FOM time	191 s	FOM time	516 s
RB time	24 s	RB time	38 s
RB/FOM time	12%	RB/FOM time	7%

Advection: traveling wave

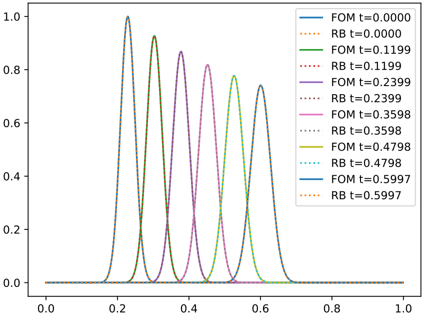
$$\begin{cases} u_t + \mu_0 u_x = 0, \; D = [0,1], \; T_{max} = 0.6, \text{ periodic BC} \\ u_0(x, \boldsymbol{\mu}) = e^{-\mu_1 (x-\mu_2)^2} \\ \mu_0 \sim \mathcal{U}([0,2]), \mu_1 \sim \mathcal{U}([500,1500]), \mu_2 \sim \mathcal{U}([0.1,0.3]) \end{cases}$$

Without calibration	With calibration: Poly2
	

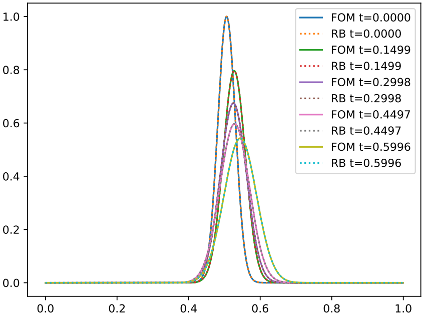
Advection: traveling wave

$$\begin{cases} u_t + \mu_0 u_x = 0,\ D=[0,1],\ T_{max}=0.6, \text{ periodic BC} \\ u_0(x,\boldsymbol{\mu}) = e^{-\mu_1(x-\mu_2)^2} \\ \mu_0 \sim \mathcal{U}([0,2]), \mu_1 \sim \mathcal{U}([500,1500]), \mu_2 \sim \mathcal{U}([0.1,0.3]) \end{cases}$$

Without calibration

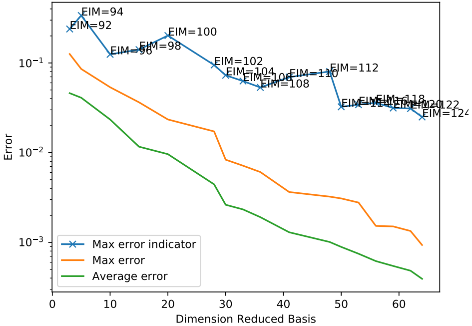
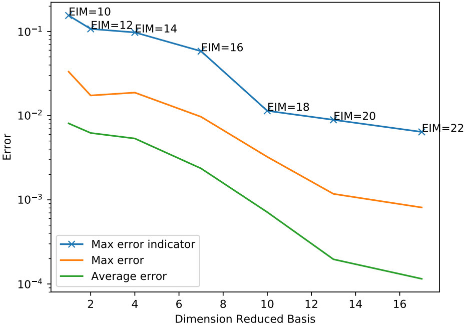


With calibration: ANN



Advection: traveling shock

$$\begin{cases} u_t + \mu_0 u_x = 0, \; D = [0,1], \; T_{max} = 1.5, \text{ Dirichlet BC} \\ u_0(x, \boldsymbol{\mu}) = \begin{cases} \mu_1 & \text{if } x < 0.35 + 0.05\mu_2 \\ 0 & \text{else} \end{cases} \\ \mu_0 \sim \mathcal{U}([0,2]), \; \mu_1, \mu_2 \sim \mathcal{U}([-1,1]) \end{cases}$$

Without calibration	With calibration: Poly2
	

Advection: traveling shock

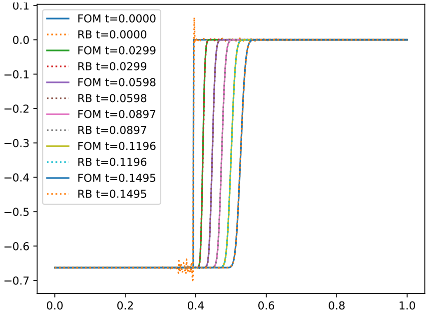
$$\begin{cases} u_t + \mu_0 u_x = 0, \; D = [0,1], \; T_{max} = 1.5, \text{ Dirichlet BC} \\ u_0(x, \boldsymbol{\mu}) = \begin{cases} \mu_1 & \text{if } x < 0.35 + 0.05\mu_2 \\ 0 & \text{else} \end{cases} \\ \mu_0 \sim \mathcal{U}([0,2]), \; \mu_1, \mu_2 \sim \mathcal{U}([-1,1]) \end{cases}$$

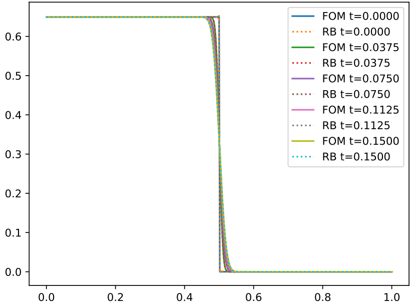
Without calibration		With calibration: Poly2	
RB dim	64	RB dim	17
EIM dim	124	EIM dim	22
FOM time	49 s	FOM time	125 s
RB time	9 s	RB time	6 s
RB/FOM time	18%	RB/FOM time	5%

Advection: traveling shock

$$\begin{cases} u_t + \mu_0 u_x = 0, \; D = [0,1], \; T_{max} = 1.5, \text{ Dirichlet BC} \\ u_0(x, \boldsymbol{\mu}) = \begin{cases} \mu_1 & \text{if } x < 0.35 + 0.05\mu_2 \\ 0 & \text{else} \end{cases} \\ \mu_0 \sim \mathcal{U}([0,2]), \; \mu_1, \mu_2 \sim \mathcal{U}([-1,1]) \end{cases}$$

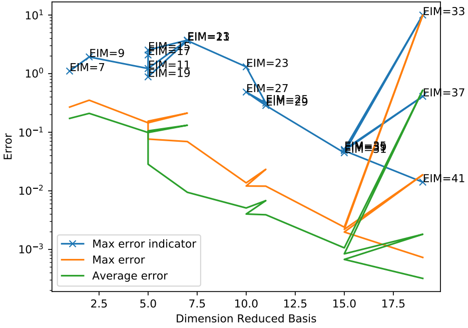
Without calibration	With calibration: Poly2





Burgers sine

$$\begin{cases} u_t + \mu_0 (u^2/2)_x = 0, \; D = [0, \pi], \; T_{max} = 0.15, \text{ periodic BC} \\ u_0(x, \boldsymbol{\mu}) = |\sin(x + \mu_1)| + 0.1 \\ \mu_0 \sim \mathcal{U}([0, 2]), \; \mu_1 \sim \mathcal{U}([0, \pi]) \end{cases}$$

Without calibration	With calibration: Poly3
	

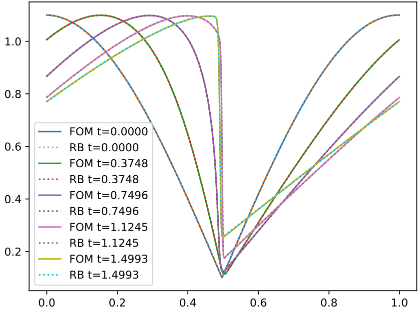
Burgers sine

$$\begin{cases} u_t + \mu_0 (u^2/2)_x = 0, \; D = [0,\pi], \; T_{max} = 0.15, \text{ periodic BC} \\ u_0(x, \boldsymbol{\mu}) = |\sin(x+\mu_1)| + 0.1 \\ \mu_0 \sim \mathcal{U}([0,2]), \; \mu_1 \sim \mathcal{U}([0,\pi]) \end{cases}$$

Without calibration		With calibration: Poly3	
RB dim	failed	RB dim	19
EIM dim	>600	EIM dim	41
FOM time	167 s	FOM time	444 s
RB time	∞	RB time	53 s
RB/FOM time	∞	RB/FOM time	11%

Burgers sine

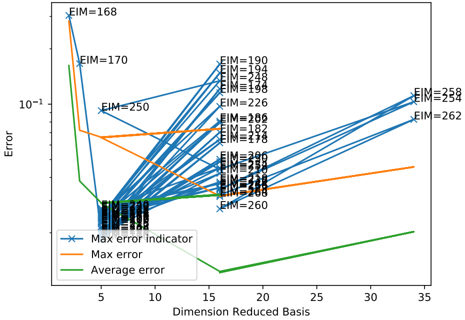
$$\begin{cases} u_t + \mu_0(u^2/2)_x = 0, \; D = [0,\pi], \; T_{max} = 0.15, \text{ periodic BC} \\ u_0(x,\boldsymbol{\mu}) = |\sin(x+\mu_1)| + 0.1 \\ \mu_0 \sim \mathcal{U}([0,2]), \; \mu_1 \sim \mathcal{U}([0,\pi]) \end{cases}$$

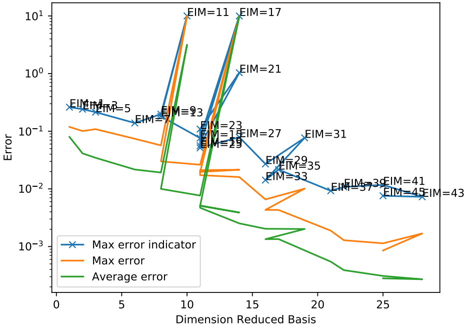
Without calibration	With calibration: Poly3
	

Buckley-Leverett equation

$$\begin{cases} \partial_t u + \partial_x \dfrac{u^2}{u^2 + \mu_0(1-u^2)} = 0, \; D = [0,1], \; T_{max} = 0.25, \text{ periodic BC} \\ u_0(x, \boldsymbol{\mu}) = 0.5 + 0.2\mu_1 + 0.3\mu_1 \sin(2\pi(x - \mu_1 - 0.5)) \\ \mu_0 \sim \mathcal{U}([0.001, 2]), \; \mu_1 \sim \mathcal{U}([0.1, 1]) \end{cases}$$

Without calibration	With calibration: pwL





Buckley-Leverett equation

$$\begin{cases} \partial_t u + \partial_x \dfrac{u^2}{u^2 + \mu_0(1-u^2)} = 0, \; D = [0,1], \; T_{max} = 0.25, \text{ periodic BC} \\ u_0(x, \boldsymbol{\mu}) = 0.5 + 0.2\mu_1 + 0.3\mu_1 \sin(2\pi(x - \mu_1 - 0.5)) \\ \mu_0 \sim \mathcal{U}([0.001, 2]), \; \mu_1 \sim \mathcal{U}([0.1, 1]) \end{cases}$$

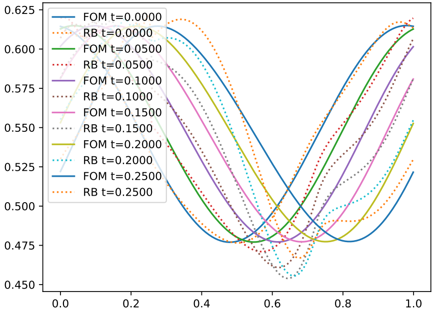
Without calibration[3]		With calibration: pwL	
RB dim	16	RB dim	25
EIM dim	270	EIM dim	45
FOM time	190 s	FOM time	462 s
RB time	69 s	RB time	79 s
RB/FOM time	36%	RB/FOM time	17%

[3] It does not reach the requested tolerance 10^{-3}

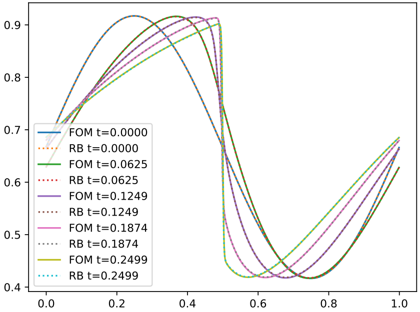
Buckley-Leverett equation

$$\begin{cases} \partial_t u + \partial_x \dfrac{u^2}{u^2 + \mu_0(1-u^2)} = 0, \; D = [0,1], \; T_{max} = 0.25, \text{ periodic BC} \\ u_0(x, \boldsymbol{\mu}) = 0.5 + 0.2\mu_1 + 0.3\mu_1 \sin(2\pi(x - \mu_1 - 0.5)) \\ \mu_0 \sim \mathcal{U}([0.001, 2]), \; \mu_1 \sim \mathcal{U}([0.1, 1]) \end{cases}$$

Without calibration



With calibration: pwL



Augmenting diffusion and ROMs

Problem: Advection diffusion.
Parameter: inlet channel width

Advection field data: given by Navier-Stokes simulation.
Discretization: DG

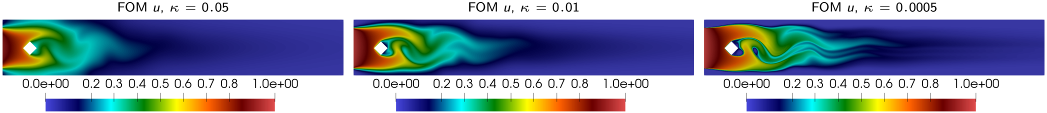


Figure: **VV.** Scalar concentration advected by incompressible flow for $i = 99$ at different viscosity levels $\kappa \in \{0.05, 0.01, 0.0005\}$

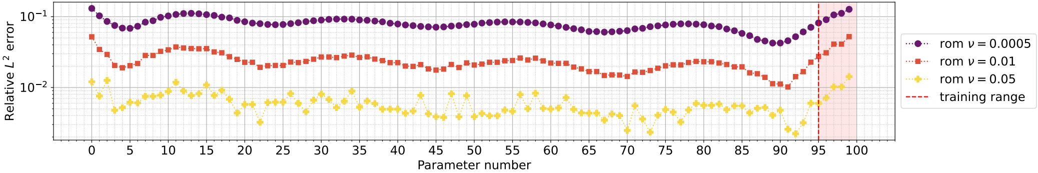


Figure: Relative errors of ROMs for different viscosities. Training correspond to the abscissae $0, 5, 10, \dots, 95$, the rest are test parameters. The dashed red background highlights the extrapolation range. The reduced dimensions of the ROMs are $\{N_{RB\Omega_i}\}_{i=1}^{K} = [5, 5, 5, 5]$ with $K = 4$ partitions.

Graph neural network architecture

Viscosity

- **Vanishing viscosity** guarantees the convergence towards physically relevant solution
- Viscosity can be **artificial** or physically modeled
- Higher viscosity can come from **coarser** grids
- **High viscosity** solutions do not suffer from slow decay of Kolomogorov n-width

Main idea

- Use classical ROM for high viscosity
- **Learn** with NN the vanishing viscosity limit

Architecture

Inputs

- **Local** data of reduced solutions from higher viscosity levels (cheap to compute) (solution, its gradient)
- Mesh connectivity (all)

Output

- "Vanishing" viscosity solution

Supervised learning

- Training data: ROMs for high viscosities, FOM for vanishing viscosity

Layers: arxiv:2308.03378
Training time: 1h

Results GNN

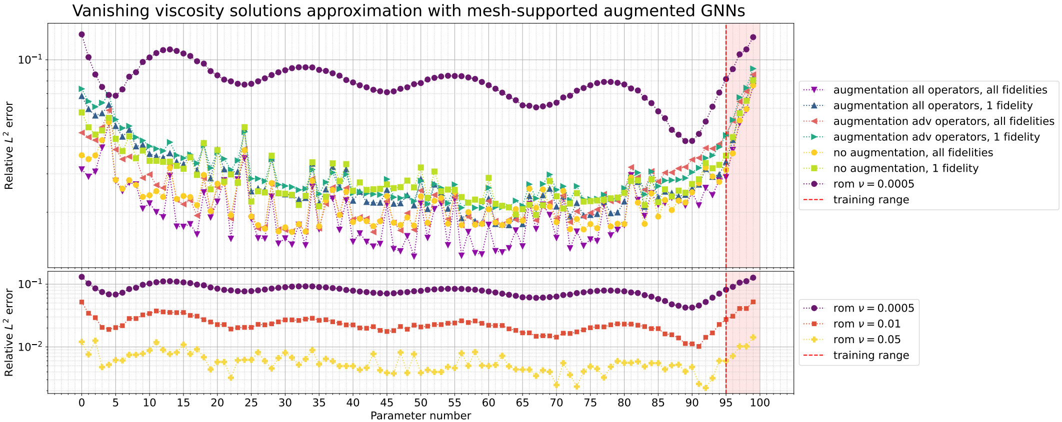


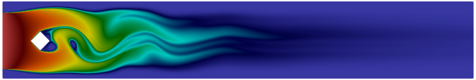
Figure: Relative errors for the scalar conservation advected by incompressible flow problem. **Top:** errors with different GNN approaches given by the three augmentation $\mathcal{O}_1$, $\mathcal{O}_2$ and $\mathcal{O}_3$ and by using either 1 viscosity level (1 fidelity) or 2 (all fidelities) and errors for DD-ROM with the same viscosity level $\nu = 0.0005$. **Bottom:** errors for ROM approaches at different viscosity levels. The reduced dimensions of the ROMs are $\{N_{RB\,\Omega_i}\}_{i=1}^{K} = [5, 5, 5, 5]$ with $K = 4$ partitions.

Results GNN

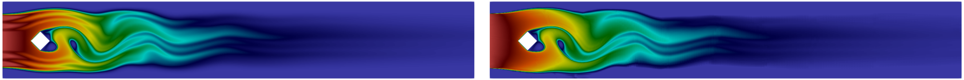
	FOM		ROM			
κ	N_h	time	$N_{RB\,i}$	time	speedup	mean L^2 error
0.05	43776	3.243 [s]	[5, 5, 5, 5]	59.912 [μs]	54129	0.00595
0.01	43776	3.236 [s]	[5, 5, 5, 5]	79.798 [μs]	40552	0.0235
0.0005	175104	9.668 [s]	[5, 5, 5, 5]	95.844 [μs]	100872	0.0796

κ	GNN training time	Single forward GNN online time	Average online time	GNN speedup	mean L^2 error
0.0005	≤ 60 [min]	2.661 [s]	0.172 [s]	~ 56	0.0217

FOM u, $\kappa = 0.0005$



ROM u, $\kappa = 0.0005$ GNN u, $\kappa = 0.0005$



GNN architecture

Table: Mesh supported augmented GNN

Net	Weights $[f_{\text{inp}}, f_{\text{out}}]$	Aggregation	Activation
Input NNConv	$[3n_{\text{aug}}, 18]$	Avg_1	ReLU
SAGEconv	[18, 21]	Avg_2	ReLU
SAGEconv	[21, 24]	Avg_2	ReLU
SAGEconv	[24, 27]	Avg_2	ReLU
SAGEconv	[27, 30]	Avg_2	ReLU
Output NNConv	[30, 1]	Avg_1	-

NNConvFilters	First Layer $[2, l]$	Activation	Second Layer $[l, f_{\text{inp}} f_{\text{out}}]$
Input NNConv	[2, 12]	ReLU	$[12, 3n_{\text{aug}} \cdot 18]$
Output NNConv	[2, 8]	ReLU	[8, 30]