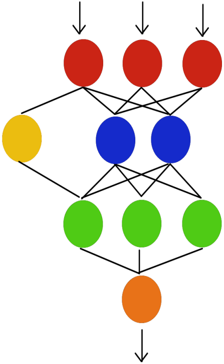
RICE ENGINEERING
Computational and Applied Mathematics



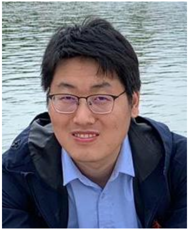
Deep learning enhancements of numerical methods

Deep Ray
Email: deep.ray@rice.edu
Website: deepray@github.io

CAAM Colloquium
9 September 2019

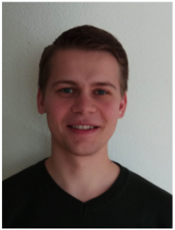
In collaboration with ...

Jan S. Hesthaven
Director MCSS
EPFL



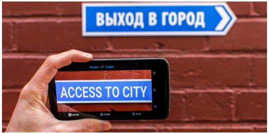
Qian Wang
Postdoc
EPFL

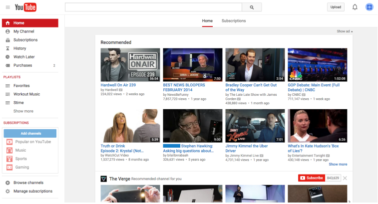
Niccolò Discacciati
Doctoral student
EPFL

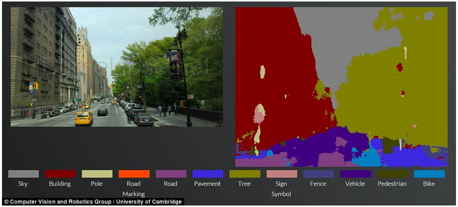


Lukas Schwander
Master student
ETH Zürich

Deep learning in action – conventional applications



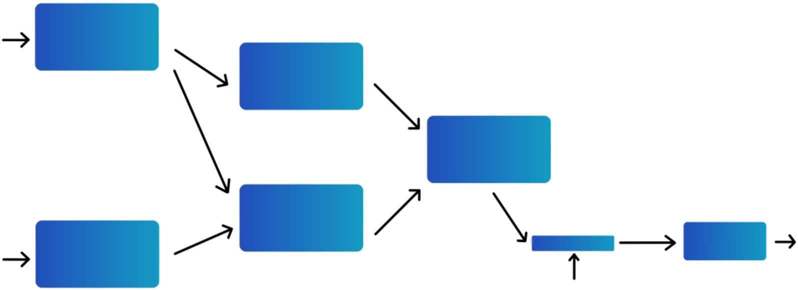




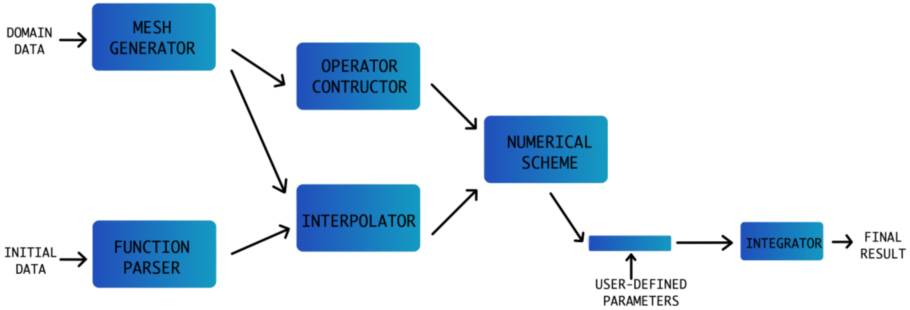
Blending deep learning and numerics

- Solving differential equations [Lagaris et al., 1998; Golak et al., 2007; Rudd et al., 2015; Tompson et al., 2017; Long et al. 2017; Raissi et al., 2018, Magiera et al., 2019]
- Subgrid scale modelling/LES/RANS [Tracey et al., 2013; Zhang et al., 2015; Ling et al., 2016; Kurian et al. 2018; Beck et al.; Duraisamy et al., 2019; Maulik et al., 2019]
- Uncertainty quantification [Tripathy et al., 2018, Kwon et al. 2018; Schwab et al, 2018; Mishra et al., 2019; Wang et al., 2019]
- Reduced order modelling [Kutz et al, 2016; Hartman et al., 2017; Carlberg et al. 2018; Willcox et al., 2019; Wang et al., 2019]
- Inverse problems [Schönlieb et al. 2017, Lunz et al., 2018; Chang et al., 2018; Raissi et al. 2019]
- Shape optimization [Timnaka et al., 2017; Baque at al., 2018; Duraisamy et al., 2019, Sasaki et al. 2019]
- ...

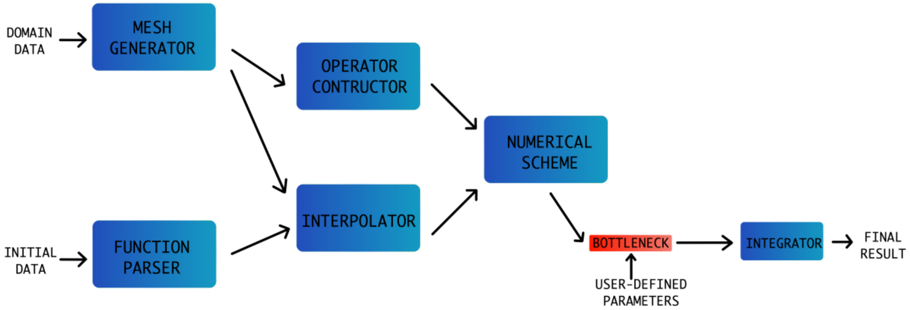
Motivation



Motivation

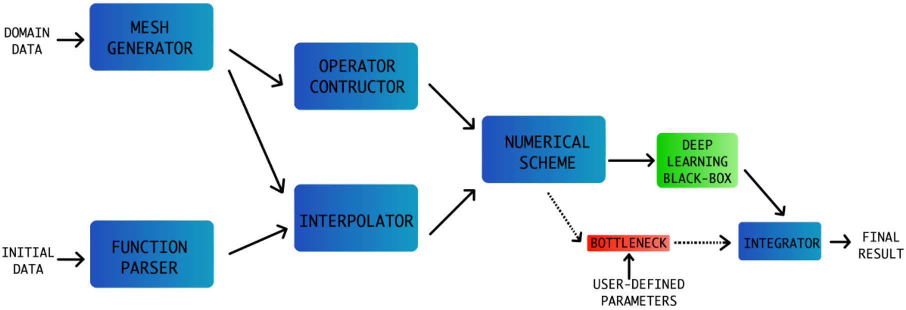

DOMAIN DATA
MESH GENERATOR
INITIAL DATA
FUNCTION PARSER
OPERATOR CONTRUCTOR
INTERPOLATOR
NUMERICAL SCHEME
USER-DEFINED PARAMETERS
INTEGRATOR
FINAL RESULT

Motivation

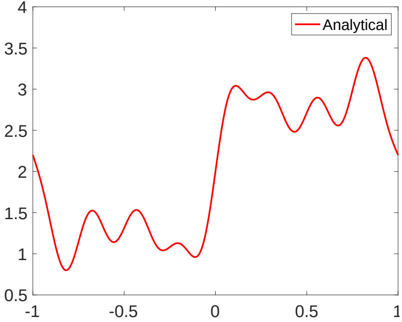


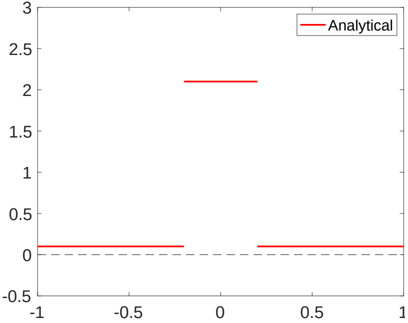
DOMAIN
DATA
MESH
GENERATOR
INITIAL
DATA
FUNCTION
PARSER
OPERATOR
CONTRUCTOR
INTERPOLATOR
NUMERICAL
SCHEME
BOTTLENECK
USER-DEFINED
PARAMETERS
INTEGRATOR
FINAL
RESULT

Motivation


DOMAIN
DATA
MESH
GENERATOR
OPERATOR
CONTRUCTOR
NUMERICAL
SCHEME
DEEP
LEARNING
BLACK-BOX
INTERPOLATOR
INITIAL
DATA
FUNCTION
PARSER
BOTTLENECK
INTEGRATOR
FINAL
RESULT
USER-DEFINED
PARAMETERS

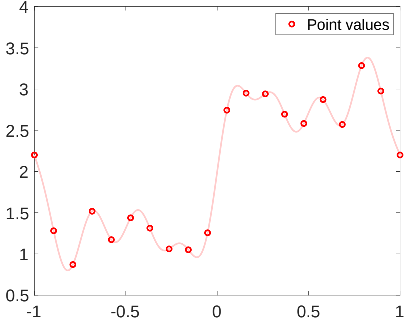
Today's bottleneck – controlling spurious oscillations

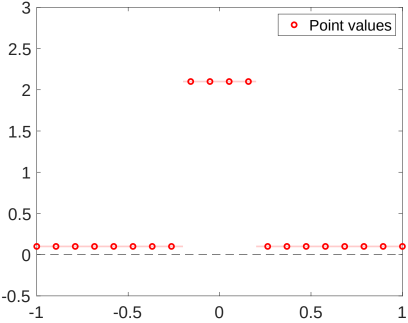




Today’s bottleneck – controlling spurious oscillations

Approximate the function using discrete data e.g. point values





Today's bottleneck – controlling spurious oscillations

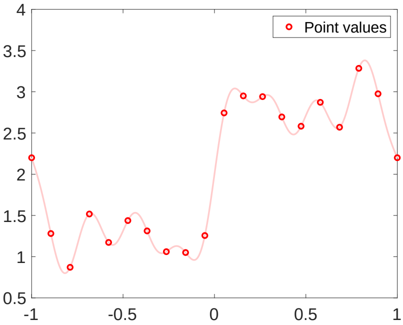
Approximate the function using discrete data e.g. point values

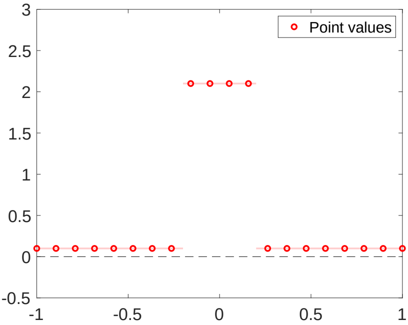
Use a smooth basis

$$u(x) \approx \sum_{p=1}^{K} u_p \phi_p(x)$$

$1, x, x^2, x^3, ...$

$\sin\left(\frac{\pi p}{K}\right), \cos\left(\frac{\pi p}{K}\right)$





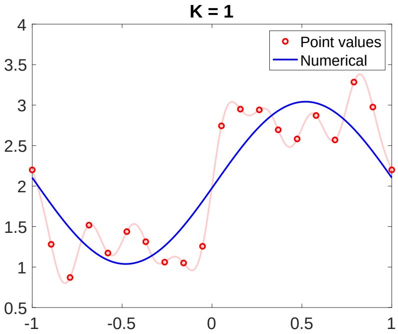
Approximate the function using discrete data e.g. point values

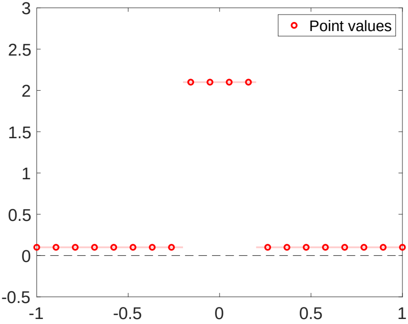
Use a smooth basis

$$u(x) \approx \sum_{p=1}^{K} u_p \phi_p(x)$$

$$1, x, x^2, x^3, \ldots$$

$$\sin\left(\frac{\pi p}{K}\right), \cos\left(\frac{\pi p}{K}\right)$$





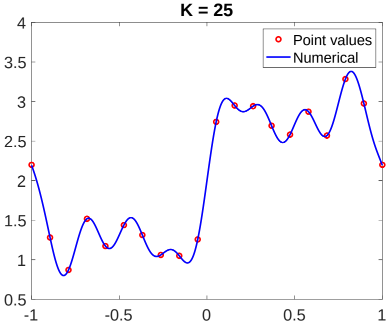
Approximate the function using discrete data e.g. point values

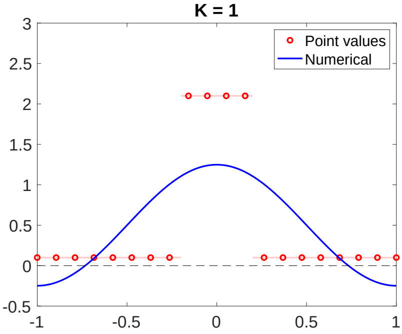
Use a smooth basis

$$u(x) \approx \sum_{p=1}^{K} u_p \phi_p(x)$$

$1, x, x^2, x^3, \ldots$

$\sin\left(\frac{\pi p}{K}\right), \cos\left(\frac{\pi p}{K}\right)$





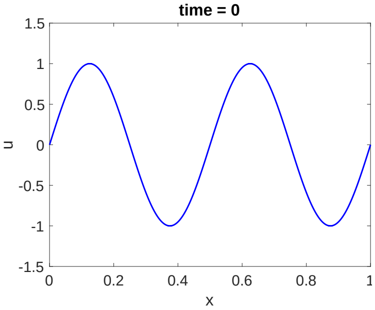
Consider

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}(\mathbf{u})}{\partial x} = 0$$
$$\mathbf{u}(x, 0) = \mathbf{u}_0(x)$$

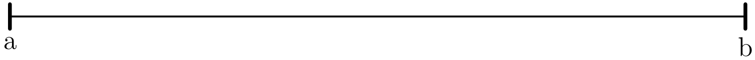
Non-linearity

$\Longrightarrow$

Discontinuities in finite time

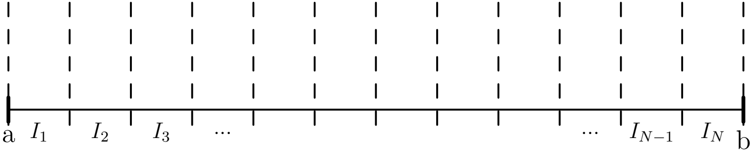


$$0 \leq t \leq T_f \qquad\qquad a \leq x \leq b$$



$$0 \leq t \leq T_f \qquad\qquad a \leq x \leq b$$

- Discretize spatial domain $[a, b]$ into N cells

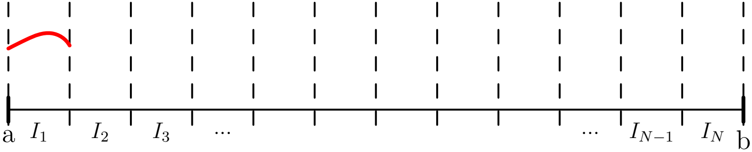


Solving conservation laws numerically

$$0 \leq t \leq T_f \qquad\qquad a \leq x \leq b$$

- Discretize spatial domain $[a, b]$ into N cells
- At time t approximate solution in each cell

$$u_i(x) = \sum_{p=1}^{K} u_p^i \phi_p^i(x), \quad x \in I_i, \quad 1 \leq i \leq N$$



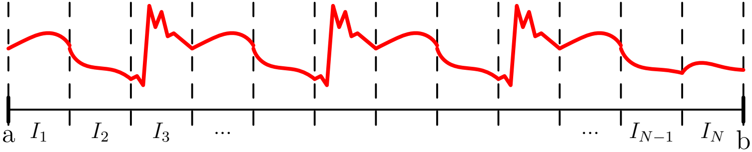
Solving conservation laws numerically

$$0 \leq t \leq T_f \qquad\qquad a \leq x \leq b$$

- Discretize spatial domain $[a, b]$ into N cells
- At time t approximate solution in each cell

$$u_i(x) = \sum_{p=1}^{K} u_p^i \phi_p^i(x), \quad x \in I_i, \quad 1 \leq i \leq N$$

- Evolve solution from time $t \longrightarrow t + \Delta t$

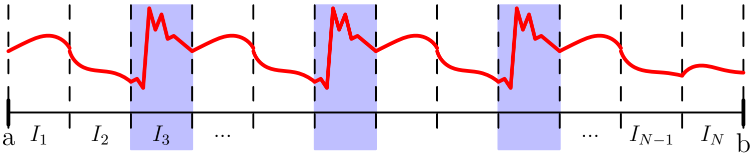


$$0 \leq t \leq T_f \qquad\qquad a \leq x \leq b$$

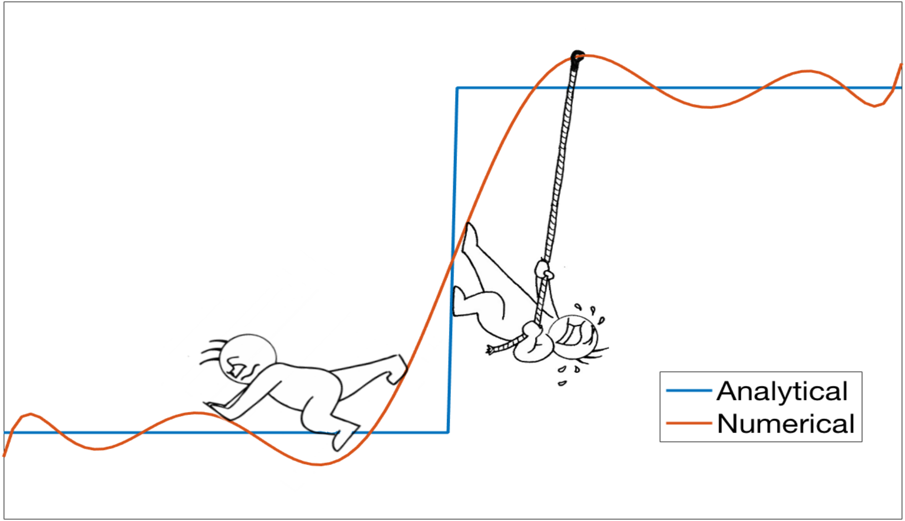
- Discretize spatial domain $[a, b]$ into N cells
- At time t approximate solution in each cell

$$u_i(x) = \sum_{p=1}^{K} u_p^i \phi_p^i(x), \quad x \in I_i, \quad 1 \leq i \leq N$$

- Evolve solution from time $t \longrightarrow t + \Delta t$

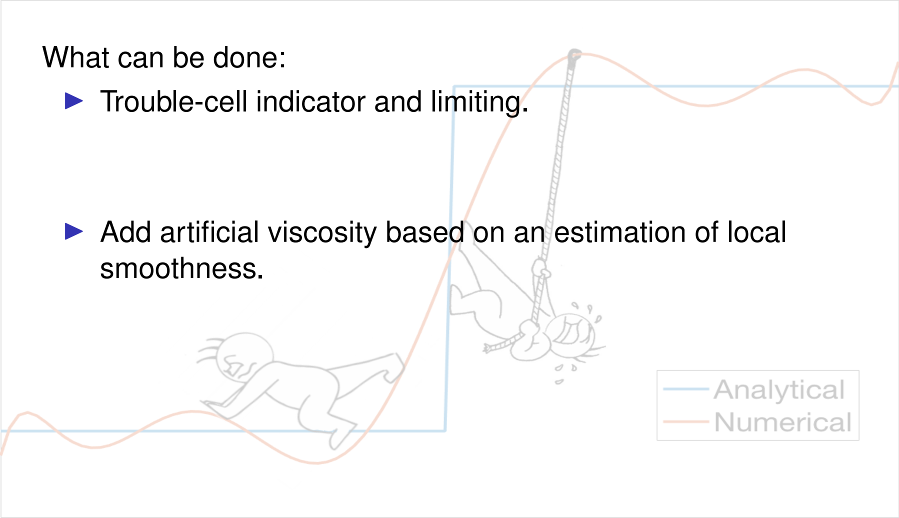


The menace of Gibbs oscillations



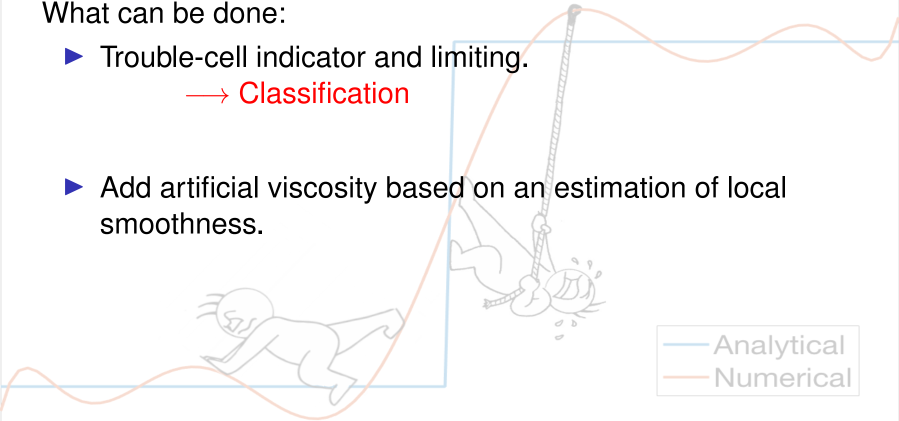
What can be done:

- Trouble-cell indicator and limiting.
- Add artificial viscosity based on an estimation of local smoothness.



What can be done:

- Trouble-cell indicator and limiting.
  $\longrightarrow$ Classification
- Add artificial viscosity based on an estimation of local smoothness.



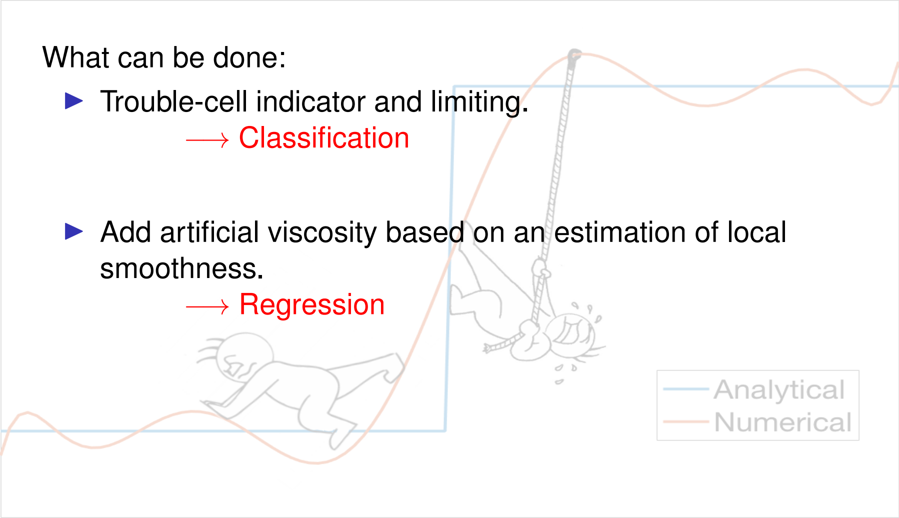
What can be done:

- Trouble-cell indicator and limiting.

  $\longrightarrow$ Classification

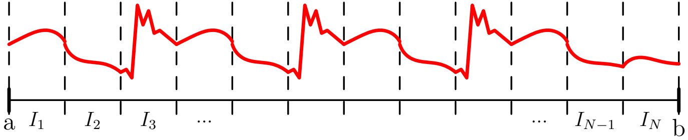
- Add artificial viscosity based on an estimation of local smoothness.

  $\longrightarrow$ Regression



Detecting and limiting

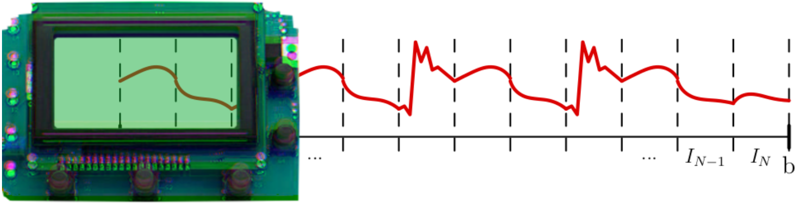
Strategy:



Detecting and limiting

Strategy:

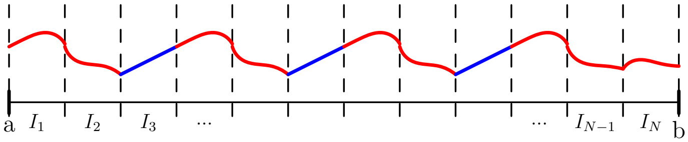
1. Find troubled-cells.



Detecting and limiting

Strategy:

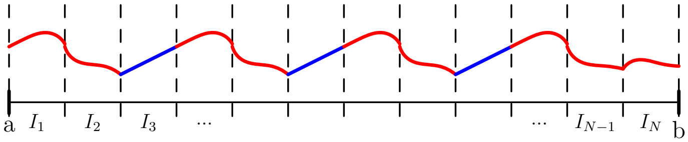
1. Find troubled-cells.
2. Limit solution in flagged cells.



Detecting and limiting

Strategy:

1. Find troubled-cells.
2. Limit solution in flagged cells.



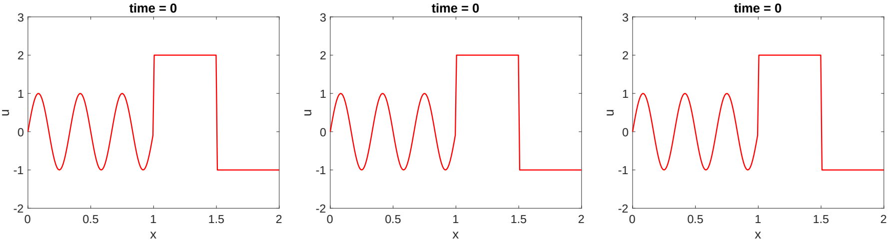
Issues:

- Problem-dependent parameters.
- If insufficient cells marked $\longrightarrow$ re-appearance of Gibbs oscillations.
- If excessive cells marked
  - Unnecessary computational cost.
  - Loss of accuracy for strong limiters.

Adding artificial viscosity

Consider the following modified PDE

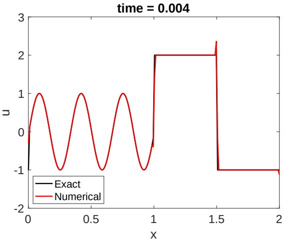
$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = \frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right)$$



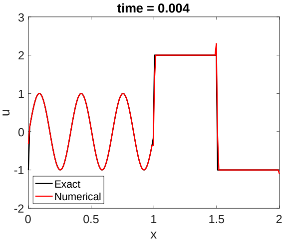
$\mu = 0$ $\qquad$ $\mu = 5.0e-4$ $\qquad$ $\mu = 1.0e-3$

Consider the following modified PDE

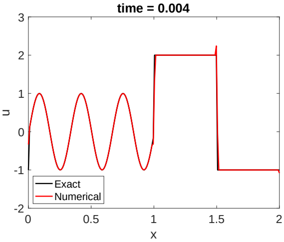
$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = \frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right)$$



$\mu = 0$



$\mu = 5.0e-4$

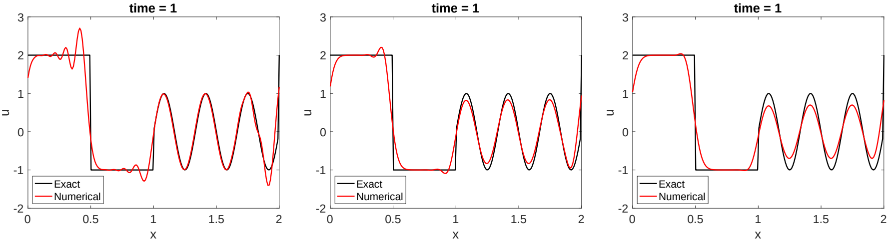


$\mu = 1.0e-3$

Adding artificial viscosity

Consider the following modified PDE

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = \frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right)$$



$\mu = 0$ $\mu = 5.0e-4$ $\mu = 1.0e-3$

Issues:

- Where to introduce viscosity?
- How much viscosity?

AIM: To develop a method that

1. Detects troubled-cells/predicts the viscosity μ.
2. Is free of problem-dependent parameters.
3. Is computationally efficient.

AIM: To develop a method that

1. Detects troubled-cells/predicts the viscosity μ.
2. Is free of problem-dependent parameters.
3. Is computationally efficient.

Accomplish this using neural networks.

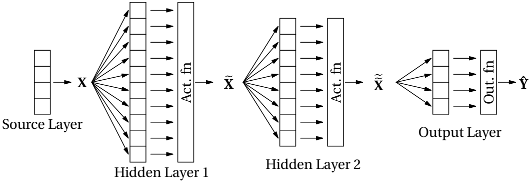
We wish to approximate the function

$$\mathbf{F} : \mathbf{X} \mapsto \mathbf{Y}, \quad \mathbf{X} \in \mathbb{R}^n,\ \mathbf{Y} \in \mathbb{R}^m$$

We wish to approximate the function

$$\mathbf{F} : \mathbf{X} \mapsto \mathbf{Y}, \quad \mathbf{X} \in \mathbb{R}^n,\ \mathbf{Y} \in \mathbb{R}^m$$

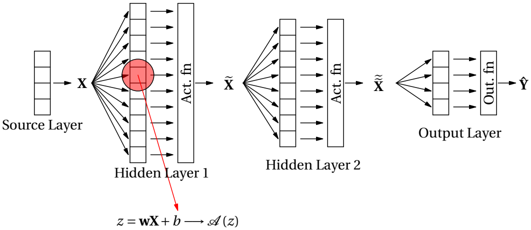
We train a suitable multilayer perceptron (MLP)



We wish to approximate the function

$$\mathbf{F} : \mathbf{X} \mapsto \mathbf{Y}, \quad \mathbf{X} \in \mathbb{R}^n,\ \mathbf{Y} \in \mathbb{R}^m$$

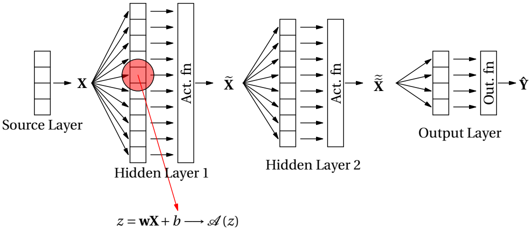
We train a suitable multilayer perceptron (MLP)

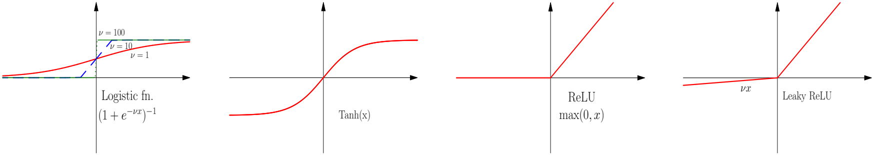


We wish to approximate the function

$$\mathbf{F} : \mathbf{X} \mapsto \mathbf{Y}, \quad \mathbf{X} \in \mathbb{R}^n, \ \mathbf{Y} \in \mathbb{R}^m$$

We train a suitable multilayer perceptron (MLP)



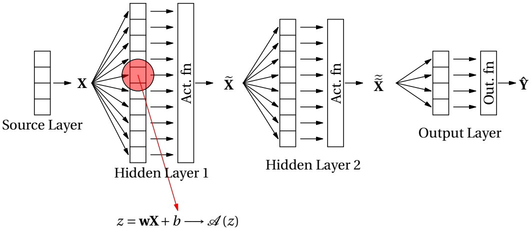


Neural networks

We wish to approximate the function

$$\mathbf{F} : \mathbf{X} \mapsto \mathbf{Y}, \quad \mathbf{X} \in \mathbb{R}^n,\ \mathbf{Y} \in \mathbb{R}^m$$

We train a suitable multilayer perceptron (MLP)



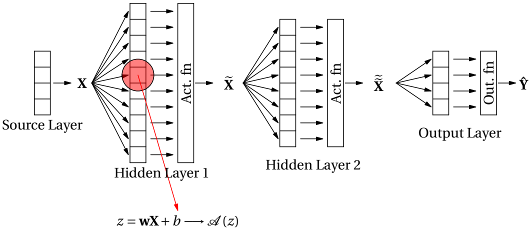
$$\hat{\mathbf{Y}} = \mathcal{O} \circ \mathcal{A} \circ H^L \circ \mathcal{A} \circ H^{L-1} \circ \ldots \circ \mathcal{A} \circ H^1(\mathbf{X}), \quad H^l(\tilde{\mathbf{X}}) = \mathbf{W}^l\tilde{\mathbf{X}} + \mathbf{b}^l$$

Neural networks

We wish to approximate the function

$$\mathbf{F} : \mathbf{X} \mapsto \mathbf{Y}, \quad \mathbf{X} \in \mathbb{R}^n,\ \mathbf{Y} \in \mathbb{R}^m$$

We train a suitable multilayer perceptron (MLP)



Find parameters $\theta = \{\mathbf{W}^l, \mathbf{b}^l\}$ that minimizes the loss function

$$\mathcal{L}(\mathbf{Y}_i, \hat{\mathbf{Y}}_i), \qquad (\mathbf{X}_i, \mathbf{Y}_i) \in \mathbb{T}.$$

Then $\hat{\mathbf{F}}(\theta) \approx \mathbf{F}$.

The hyperparameters

Based on some theory and prior experience, choose

- Network size – depth and width
- Activation function
- Loss function
- Regularization technique – to avoid overfitting
- Training and validation datasets
- Stopping criteria for training
- Optimizer

Troubled-cell detector

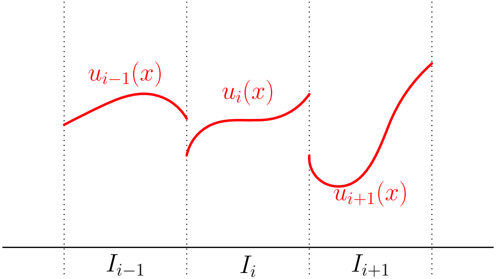
Available troubled-cell indicators

- Minmod-based TVB limiter [Cockburn and Shu; Math. Comp. '98]
- Moment limiter [Biswas et al.; Appl. Numer. Math. '94]
- Modified moment limiter [Burbeau; JCP '01]
- Monotonicity preserving limiter [Suresh and Huynh; JCP '97]
- Modified MP limiter [Rider and Margolin; JCP '01]
- KXRCF indicator [Krivodonova et al.; App. Numer. Math. '04]
- Polynomial degree based limiter [Fu and Shu; JCP '17]
- Outlier detection using Tukey's boxplot method [Vuik and Ryan; J. Sci. Comp. '16]
- ...

Available troubled-cell indicators

- Minmod-based TVB limiter [Cockburn and Shu; Math. Comp. ‘98]
- Moment limiter [Biswas et al.; Appl. Numer. Math. ‘94]
- Modified moment limiter [Burbeau; JCP ‘01]
- Monotonicity preserving limiter [Suresh and Huynh; JCP ‘97]
- Modified MP limiter [Rider and Margolin; JCP ‘01]
- KXRCF indicator [Krivodonova et al.; App. Numer. Math. ‘04]
- Polynomial degree based limiter [Fu and Shu; JCP ‘17]
- Outlier detection using Tukey’s boxplot method [Vuik and Ryan; J. Sci. Comp. ‘16]
- ...

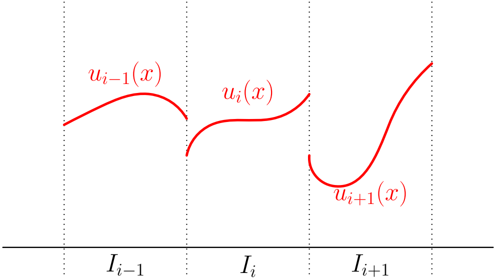
TVB Indicator: Search for the elusive M



- For each cell I_i, get

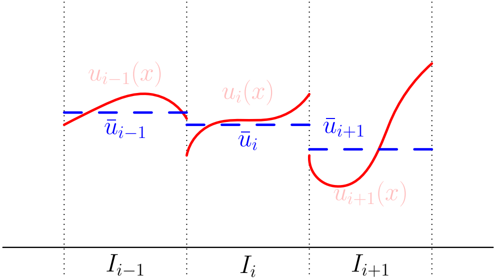
TVB Indicator: Search for the elusive M

$$\overline{u}_i = \frac{1}{|I_i|} \int_{I_i} u_i(x) \mathrm{d}x$$

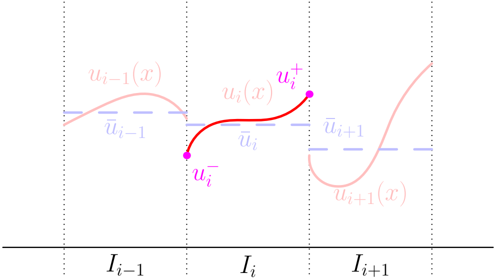


- For each cell I_i, get

TVB Indicator: Search for the elusive M

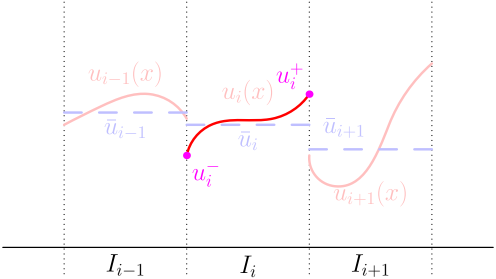


- For each cell I_i, get $[\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1} \quad\quad\quad]$



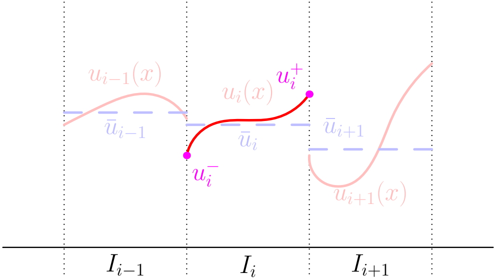
- For each cell I_i, get $[\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u_i^-, u_i^+] \in \mathbb{R}^5$

TVB Indicator: Search for the elusive M



- For each cell I_i, get $[\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u_i^-, u_i^+] \in \mathbb{R}^5$
- Evaluate divided difference $\longrightarrow$ estimate the local gradient.

TVB Indicator: Search for the elusive *M*



- For each cell I_i, get $[\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u_i^-, u_i^+] \in \mathbb{R}^5$
- Evaluate divided difference $\longrightarrow$ estimate the local gradient.
- Choose M $\longrightarrow$ problem dependent!!

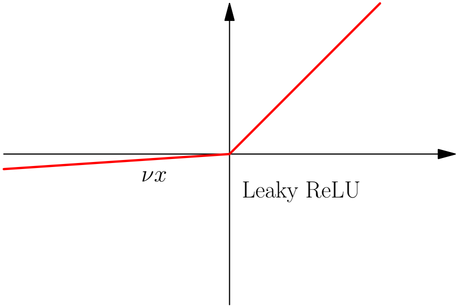
An MLP-based detector

- Input $\mathbf{X} = [\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u_i^-, u_i^+] \in \mathbb{R}^5$ (Scaled)

An MLP-based detector

- Input $\mathbf{X} = [\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u_i^-, u_i^+] \in \mathbb{R}^5$ (Scaled)
- 5 Hidden Layers with width 256, 128, 64, 32, 16

- Input $\mathbf{X} = [\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u_i^-, u_i^+] \in \mathbb{R}^5$ (Scaled)
- 5 Hidden Layers with width 256, 128, 64, 32, 16
- Leaky ReLU activation function with $\nu = 10^{-3}$



- Input $\mathbf{X} = [\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u_i^-, u_i^+] \in \mathbb{R}^5$ (Scaled)
- 5 Hidden Layers with width 256, 128, 64, 32, 16
- Leaky ReLU activation function with $\nu = 10^{-3}$
- Softmax output function

$$\hat{Y}^{(k)} \leftarrow \frac{e^{\hat{Y}^{(k)}}}{\sum_j e^{\hat{Y}^{(j)}}} \quad \in \quad [0,1] \quad \longrightarrow \quad \text{probabilities}$$

Output $\hat{\mathbf{Y}} = [\hat{Y}^{(0)}, \hat{Y}^{(1)}] \in [0,1]^2$

Troubled-cell if $\hat{Y}^{(0)} > 0.5$

- Input $\mathbf{X} = [\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u_i^-, u_i^+] \in \mathbb{R}^5$ (Scaled)
- 5 Hidden Layers with width 256, 128, 64, 32, 16
- Leaky ReLU activation function with $\nu = 10^{-3}$
- Softmax output function

$$\hat{Y}^{(k)} \leftarrow \frac{e^{\hat{Y}^{(k)}}}{\sum_j e^{\hat{Y}^{(j)}}} \quad \in \quad [0,1] \quad \longrightarrow \quad \text{probabilities}$$

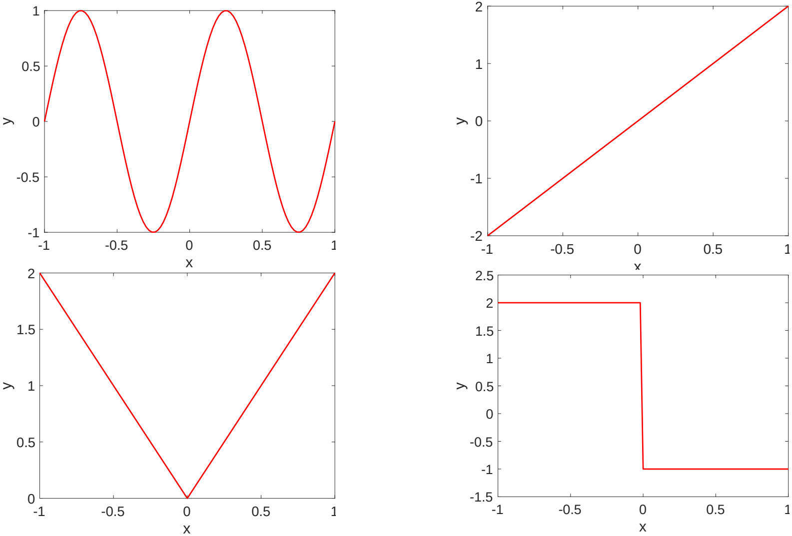
Output $\hat{\mathbf{Y}} = [\hat{Y}^{(0)}, \hat{Y}^{(1)}] \in [0,1]^2$

Troubled-cell if $\hat{Y}^{(0)} > 0.5$

- Cost functional: L2 regularized cross-entropy

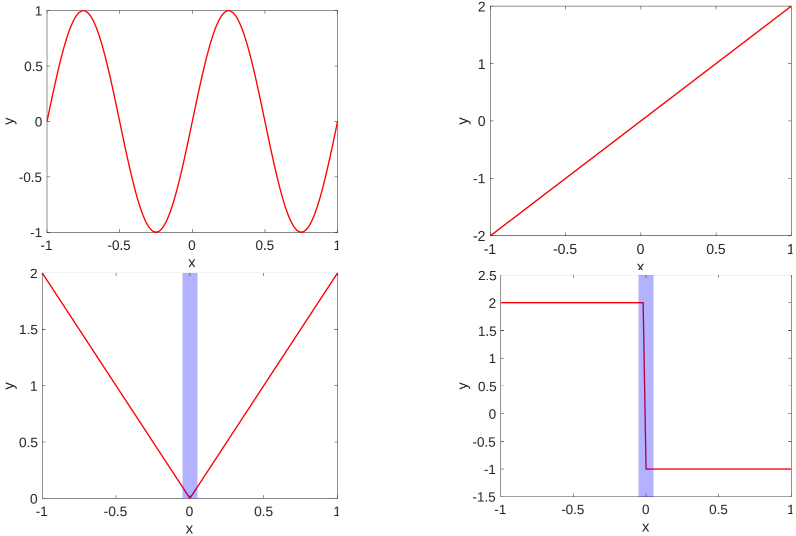
$$\mathcal{L} = -\sum_{i=1}^{K} \sum_{j=0}^{1} Y_i^{(j)} \log(\hat{Y}_i^{(j)}) \; + \; \lambda \|\mathbf{W}\|_2^2$$

Types of functions used:



Generating training data

Types of functions used:



Numerical setup

- Discontinuous Galerkin scheme.
- In flagged cells, perform limited linear reconstruction with MUSCL limiter.
- Legendre basis (Jacobi polynomials in 2D) with degree r.
- Local Lax-Friedrich numerical flux.
- Time integration with SSP-RK3.
- Comparison with TVB indicator by setting parameter M.

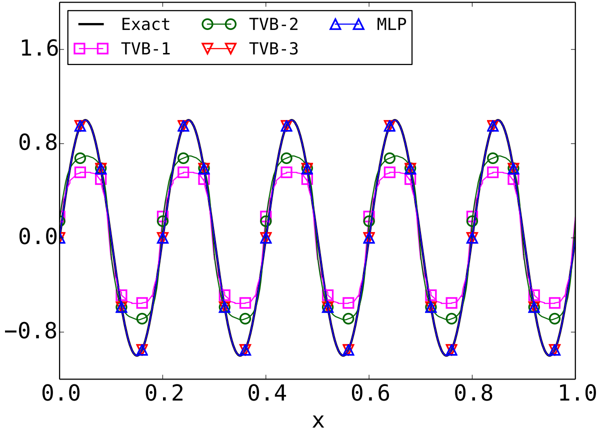
An artificial neural network as a troubled-cell indicator, by R. and Hesthaven; JCP vol. 367, 2018.

Detecting troubled-cells on two-dimensional unstructured grids using a neural network, by R. and Hesthaven, JCP vol. 397, 2019.

Linear advection: $u_t + u_x = 0$

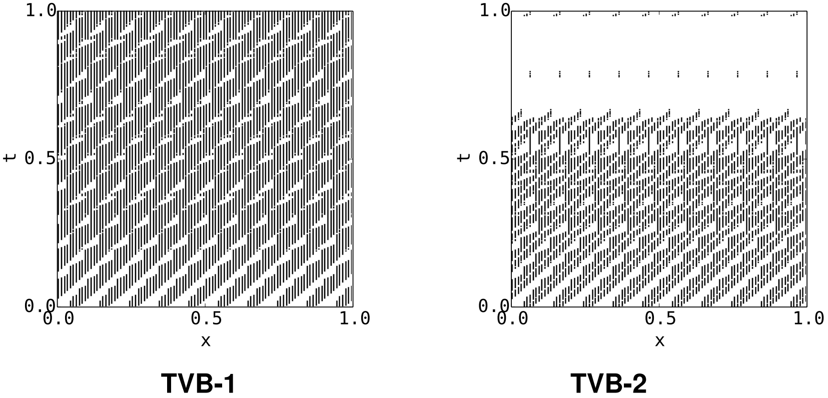
$$u_0(x) = \sin(10\pi x), \quad x \in [0,1], \quad T_f = 1, \quad N = 100, \quad r = 4$$

TVB-1 $\longrightarrow$ M=10
TVB-2 $\longrightarrow$ M=100
TVB-3 $\longrightarrow$ M=1000



Linear advection: $u_t + u_x = 0$

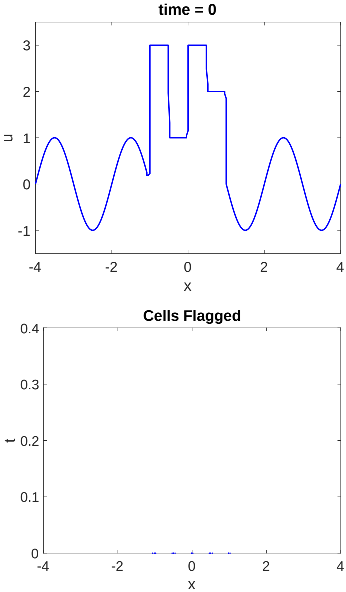
$$u_0(x) = \sin(10\pi x), \quad x \in [0,1], \quad T_f = 1, \quad N = 100, \quad r = 4$$



MLP and TVB-3 do not flag any cell

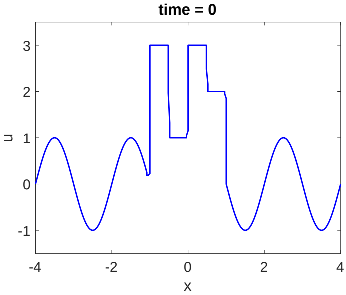
Burgers equation: $u_t + (u^2/2)_x = 0$

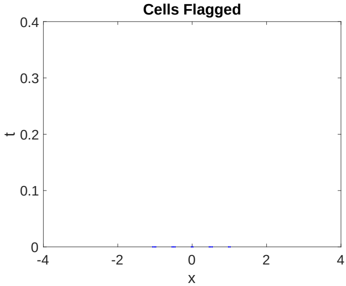
$N = 200$,
$r = 4$,
$T_f = 0.4$



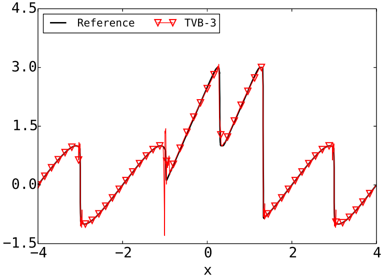
Burgers equation: $u_t + (u^2/2)_x = 0$

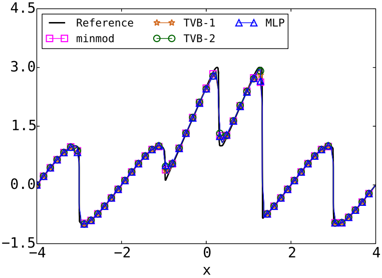
$N = 200,$
$r = 4,$
$T_f = 0.4$





Burgers equation: $u_t + (u^2/2)_x = 0$





Euler equations

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho \\ \rho v \\ E\end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix}\rho v \\ p + \rho v^2 \\ (E+p)v\end{bmatrix} = 0$$

$$E = \rho\left(\frac{v^2}{2} + e\right), \qquad e = \frac{p}{(\gamma - 1)\rho}, \qquad \gamma = 1.4$$

$\rho \longrightarrow$ fluid density

$v \longrightarrow$ velocity

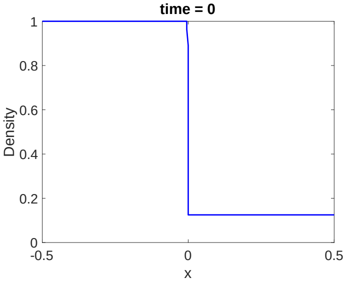
$p \longrightarrow$ pressure

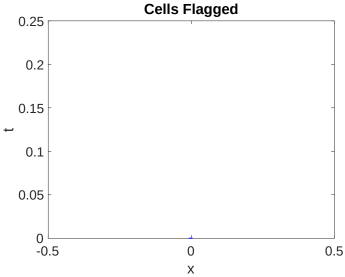
$E \longrightarrow$ total energy

$e \longrightarrow$ internal energy

Euler equations: Sod shock tube

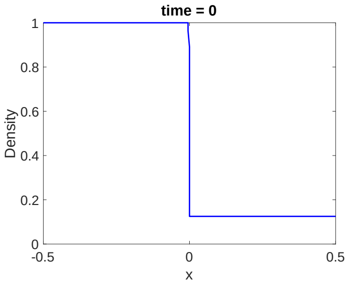
$N = 100$,
$r = 4$,
$T_f = 2$

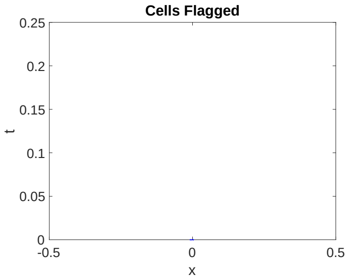


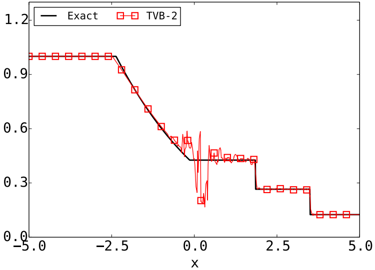


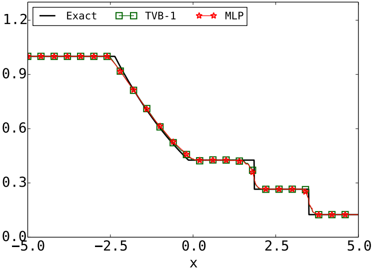
Euler equations: Sod shock tube

$N = 100,$
$r = 4,$
$T_f = 2$





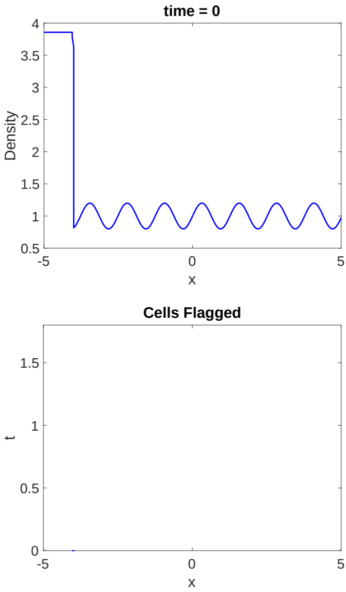




Loss of positivity with TVB-3

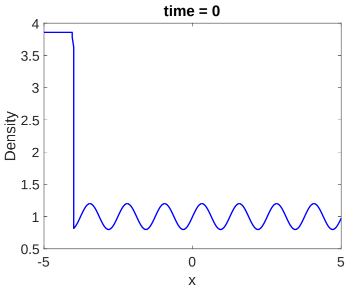
1D Euler equations: Shu-Osher problem

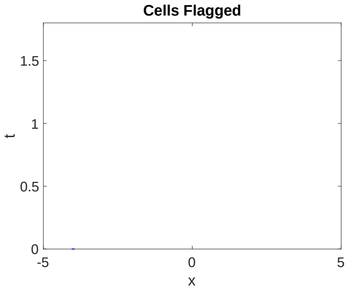
$N = 256,$
$r = 4,$
$T_f = 1.8$



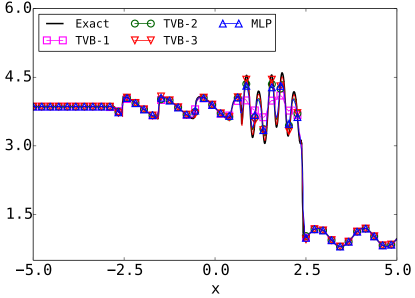
1D Euler equations: Shu-Osher problem

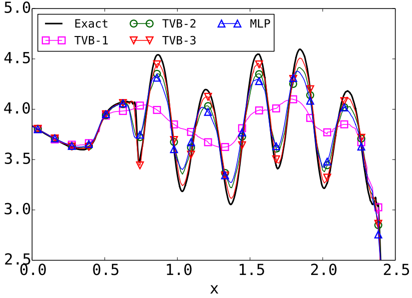
$N = 256$,
$r = 4$,
$T_f = 1.8$





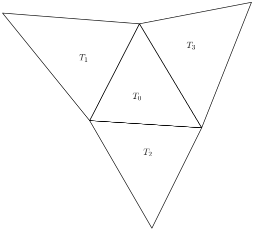
1D Euler equations: Shu-Osher problem





Extension to 2D

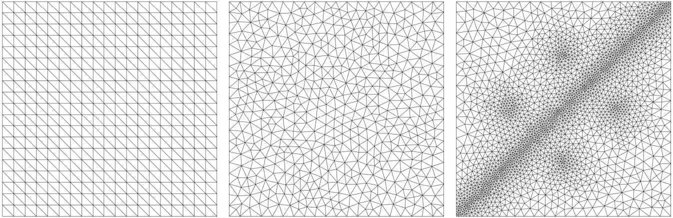
We consider unstructured triangular grids



MLP network with 5 hidden layers of width 20 each

Extension to 2D

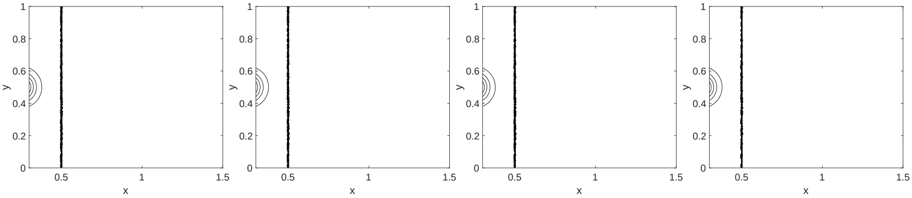
We consider unstructured triangular grids



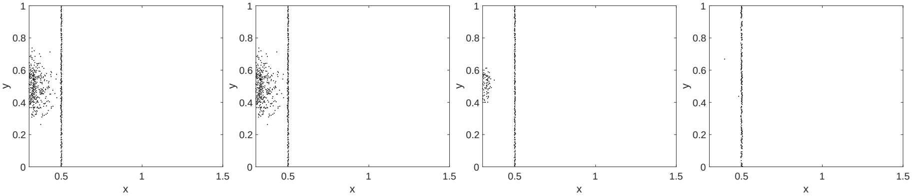
MLP network with 5 hidden layers of width 20 each
Training data constructed by:

- Interpolating functions to patch of 4 triangles.
- Extracting linear components, with $\mathbf{X} \in \mathbb{R}^{12}$.
- Label cell as troubled-cell if discontinuity present within circumscribed circle.

$$T_f = 0.8, \quad h = 0.01, \quad r = 3, \quad \text{Barth-Jesperson limiter}$$

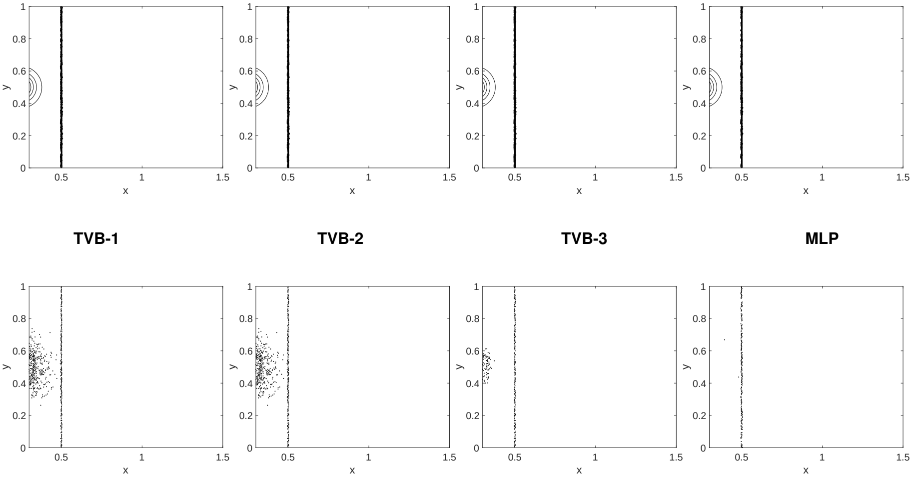


TVB-1 **TVB-2** **TVB-3** **MLP**



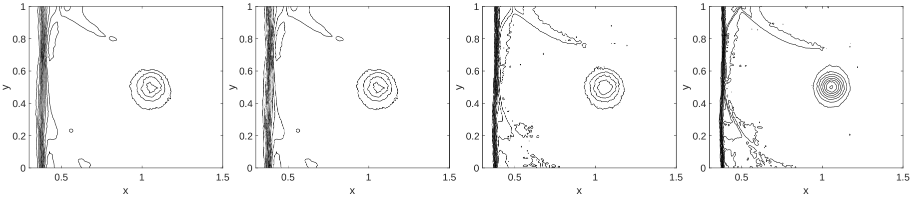
2D Euler equations: Shock-vortex

$T_f = 0.8, \quad h = 0.01, \quad r = 3,$ Barth-Jesperson limiter

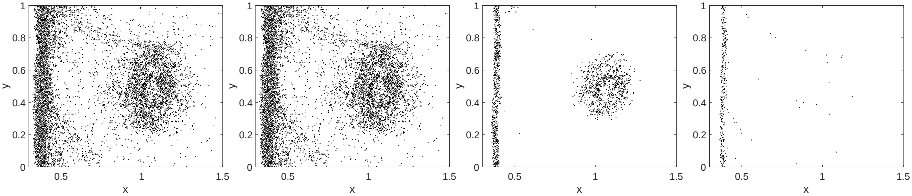


2D Euler equations: Shock-vortex

$T_f = 0.8, \quad h = 0.01, \quad r = 3,$ Barth-Jesperson limiter



TVB-1 **TVB-2** **TVB-3** **MLP**



Predicting artificial viscosity

Adding artificial viscosity

Now consider the problem

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{f}(\mathbf{u}) = \nabla \cdot \mathbf{g}, \quad \mathbf{g} = \mu \mathbf{w}, \quad \mathbf{w} = \nabla \mathbf{u}$$

Viscosity depends on **u** and locally controls oscillations.

$$\mu \sim h|\mathbf{f}'(\mathbf{u})|$$

Several artificial viscosity models exist

- MDH: Highest Modal Decay [Persson and Peraire; 14th AIAA meet, 2016]
- MDA: Averaged Modal Decay [Klöckner et al.; Math. Mod. Nat. Phen., 2011]
- EV: Entropy Viscosity [Guermond et al.; JCP, 2011]

Dependent on problem specific parameters

Training an MLP

Network architecture:

- Input: Full data from each element (no neighbours).
- 5 hidden layers of width 10 each with Leaky ReLU.
- Softplus output function, i.e., $f(x) = \log(1 + e^x)$.
- Mean Squared Error cost function.

Training an MLP

Network architecture:

- Input: Full data from each element (no neighbours).
- 5 hidden layers of width 10 each with Leaky ReLU.
- Softplus output function, i.e., $f(x) = \log(1 + e^x)$.
- Mean Squared Error cost function.

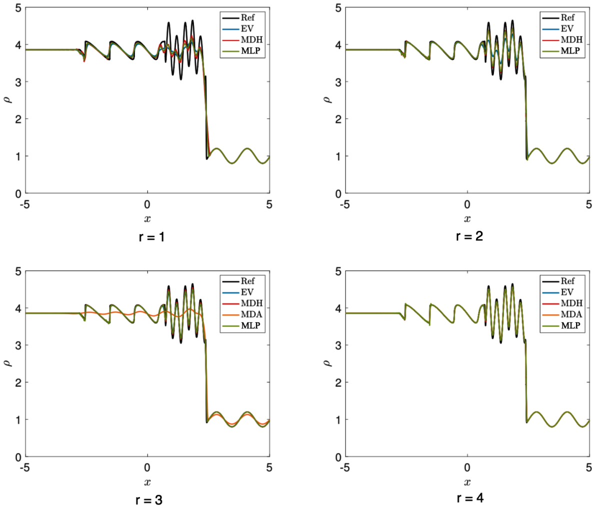
Training/validation sets:

- Using numerical solution of conservation laws (only linear adv. and Burgers).
- Target viscosity: viscosity corresponding to "best" model.
- Different network for each degree r.

Controlling oscillations in high-order Discontinuous Galerkin schemes using artificial viscosity tuned by neural networks, by Discacciti, Hesthaven and R.; 2019 (submitted).

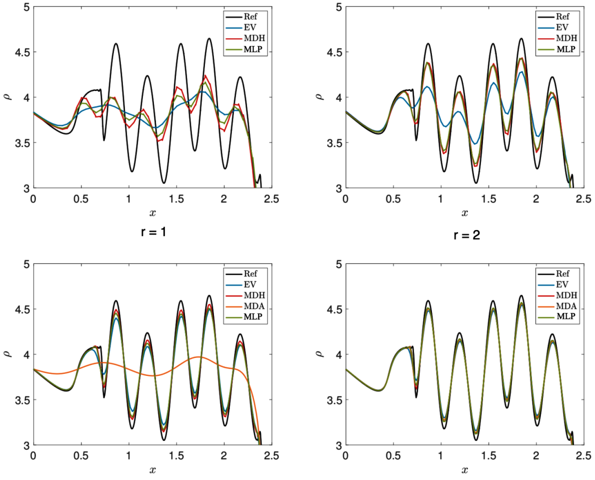
1D Euler equations: Shu-Osher

$T_f = 1.8, \quad h = 10/200$



1D Euler equations: Shu-Osher

$$T_f = 1.8, \quad h = 10/200$$

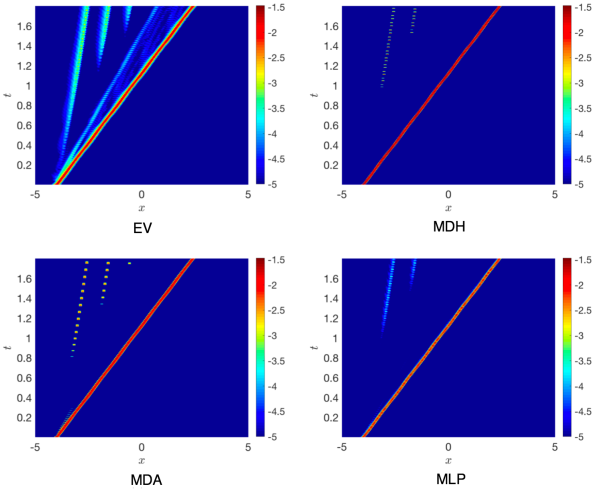


r = 1 r = 2

r = 3 r = 4

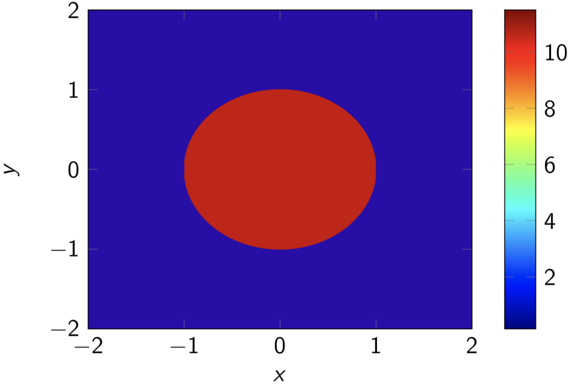
1D Euler equations: Shu-Osher

$$T_f = 1.8, \quad h = 10/200, \quad r = 4$$



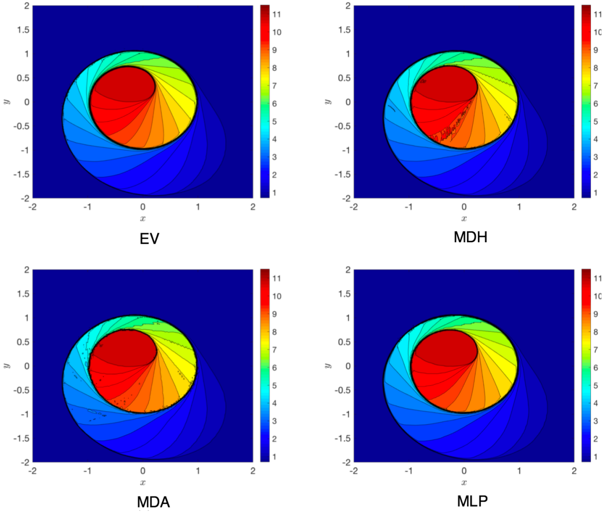
2D KPP equations: Rotating wave

$$\mathbf{f}(u) = (\sin u, \cos u), \quad T_f = 1, \quad h = 4\sqrt{2}/120, \quad r = 4$$

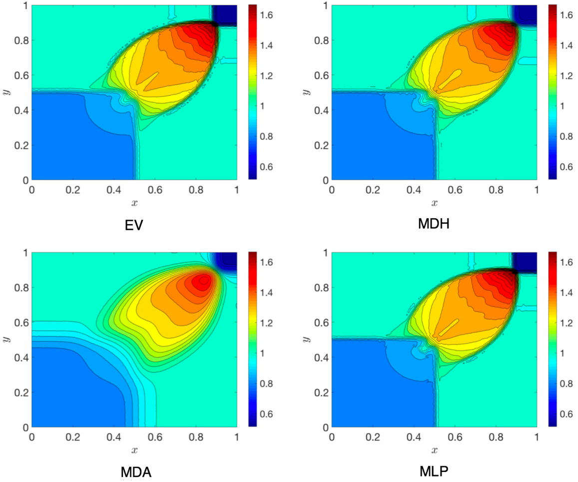


2D KPP equations: Rotating wave

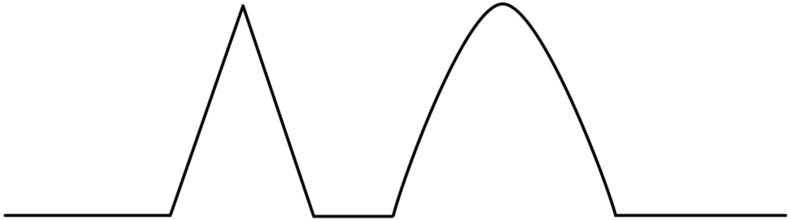
$$\mathbf{f}(u) = (\sin u, \cos u), \quad T_f = 1, \quad h = 4\sqrt{2}/120, \quad r = 4$$



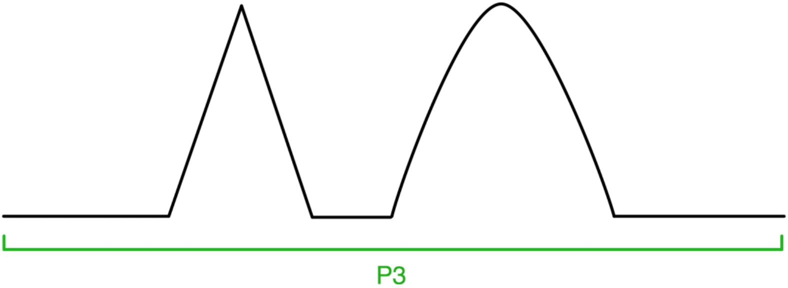
2D Euler equations: Riemann Problem config. 12 ($r = 3$)



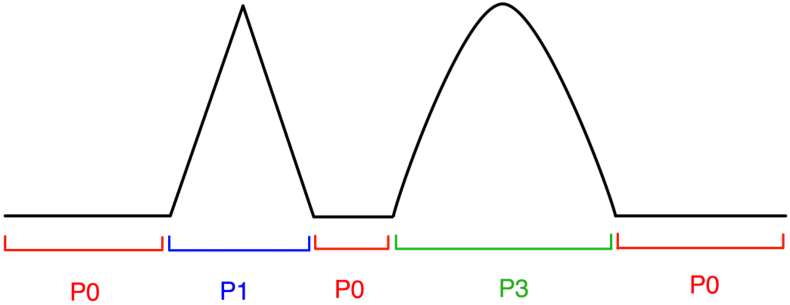
What else can we do?



p-adaption



In each element: $u_i^h(x) = \sum_{l=0}^{p_{max}} \hat{u}_l^i \phi_l(x)$

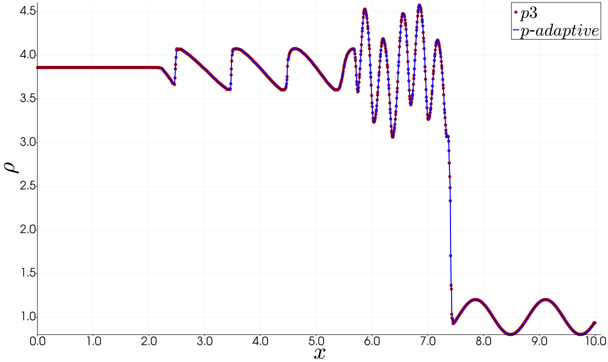


In each element: $u_i^h(x) = \sum_{l=0}^{p_{max}} \hat{u}_l^i \phi_l(x)$

Adapt P**:** $u_i^h(x) = \sum_{l=0}^{p_i} \hat{u}_l^i \phi_l(x), \;\; 0 \leq p_i \leq p_{max}$

Euler equations: Shu-Osher

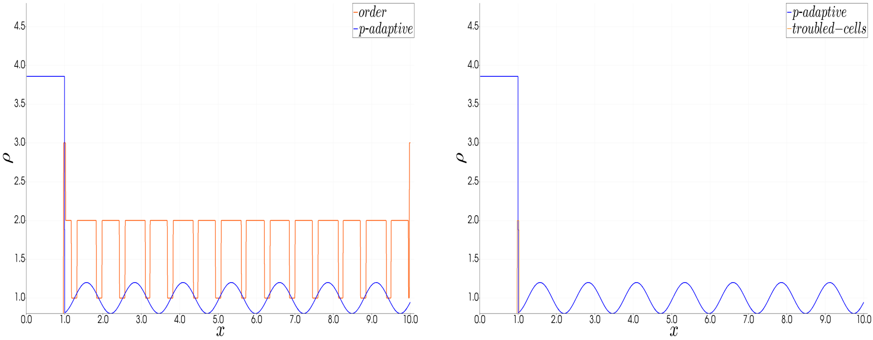
$T_f = 1.8, \quad N = 400$



Multi-dimensional WBAP limiter [Li et al., JCP, 2011] used near discontinuities.

Euler equations: Shu-Osher

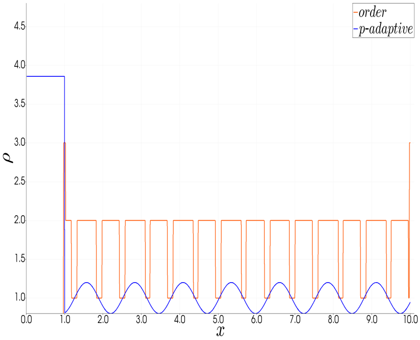
$$T_f = 1.8, \quad N = 400$$

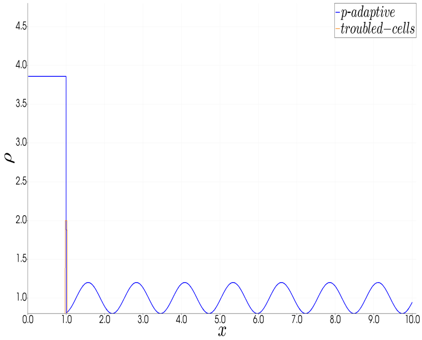


Multi-dimensional WBAP limiter [Li et al., JCP, 2011] used near discontinuities.

Euler equations: Shu-Osher

$$T_f = 1.8, \quad N = 400$$

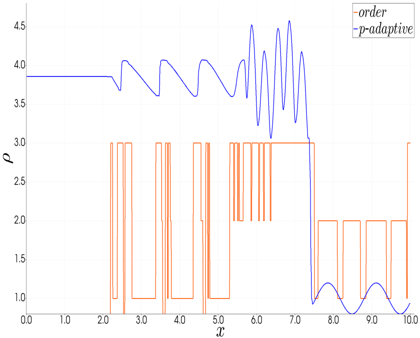




Multi-dimensional WBAP limiter [Li et al., JCP, 2011] used near discontinuities.

Euler equations: Shu-Osher

$$T_f = 1.8, \quad N = 400$$



Multi-dimensional WBAP limiter [Li et al., JCP, 2011] used near discontinuities.

Assuming periodic boundary conditions on $[0, L]$

$$u(x,t) \approx u_h(x,t) = \sum_{n=-N}^{N} \tilde{u}_n(t) \exp\left(i\, n\, 2\pi \frac{x}{L}\right) \quad \longrightarrow \quad \text{global}$$

Define $2N+1$ collocation points (uniform mesh)

$$x_j = \frac{L}{2N+1} j, \quad j = 0, \ldots 2N$$

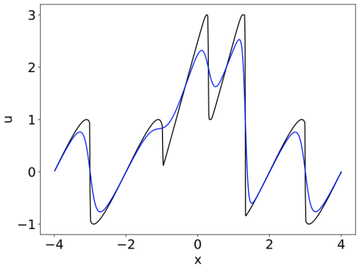
Solve for

$$\frac{\mathrm{d}\mathbf{u}(t)}{\mathrm{d}t} + \mathbf{D}\mathbf{u}(t) = 0$$

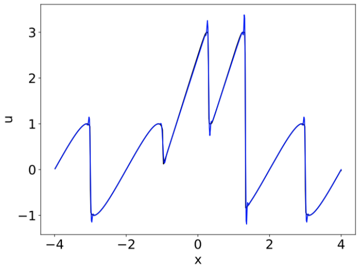
where $\mathbf{u}(t) = [u_h(x_0, t), \ldots, u_h(x_{2N}, t)]^\top$.

Handling Gibbs oscillations:

- Add hyper-viscosity
  - Second-order: cures oscillations but smears solution.
  - High-order: better accuracy but local oscillations remain.
- Post-processing exponential filter – same issue as above.
- Use Fourier-Padé reconstruction – oscillations still persist.



Filter order 2



Filter order 4

Handling Gibbs oscillations:

- Add hyper-viscosity
  - Second-order: cures oscillations but smears solution.
  - High-order: better accuracy but local oscillations remain.
- Post-processing exponential filter – same issue as above.
- Use Fourier-Padé reconstruction – oscillations still persist.

A simpler approach using MLPs

$$\frac{\mathrm{d}\mathbf{u}(t)}{\mathrm{d}t} + \mathbf{D}\mathbf{u}(t) = \mathbf{D}[\mu \otimes \mathbf{D}\mathbf{u}]$$

where μ varies locally based on local-regularity

Local viscosity with an MLP

Evaluate viscosity as a classification problem [Klöckner et al., 2011; Yu et al., 2018]

The network predicts the local regularity τ based on a seven-point stencil

τ	1	2	3	4
Reg.	Disc.	C0/C1	C1/C2	C2

Local viscosity with an MLP

Evaluate viscosity as a classification problem [Klöckner et al., 2011; Yu et al., 2018]

The network predicts the local regularity τ based on a seven-point stencil

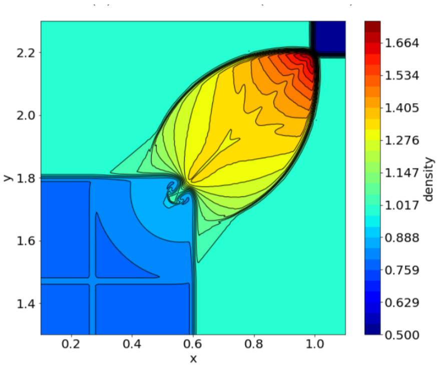
τ	1	2	3	4
Reg.	Disc.	C0/C1	C1/C2	C2

$$\mu = \mu_{max} \begin{cases} 1 - (\tau - 1)/2 & \text{if } 1 \leq \tau \leq 3 \\ 0 & \text{if } \tau = 4 \end{cases}$$

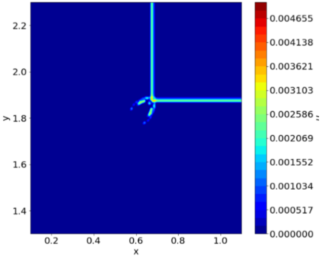
where $\mu_{max} = \frac{h}{2} \max(|f'(u)|)$.

2D Euler equations: Riemann problem config. 12

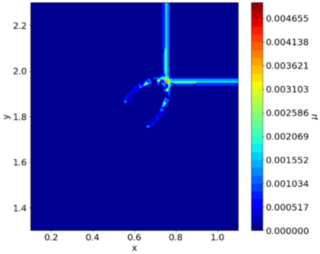
$$N_x = N_y = 250$$



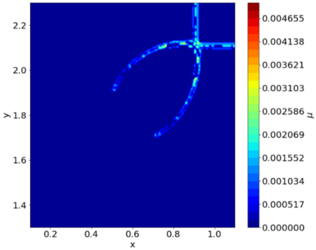
2D Euler equations: Riemann problem config. 12



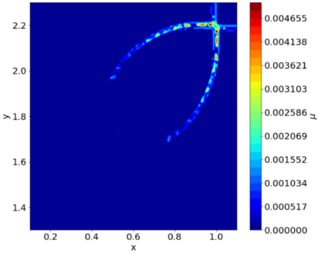
t=0.05



t = 0.1

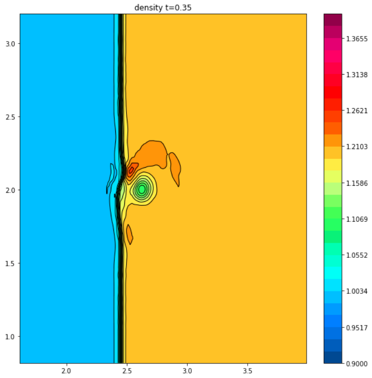


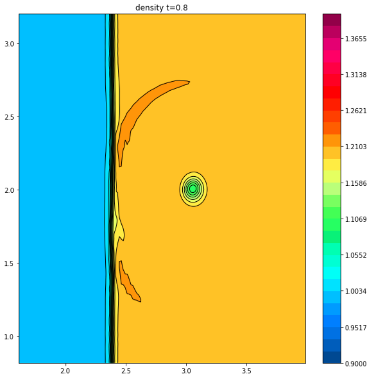
t=0.2



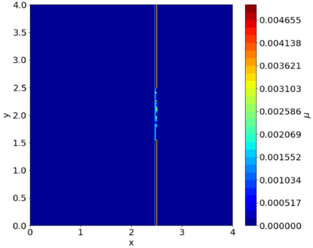
t = 0.25

$$N_x = N_y = 200$$

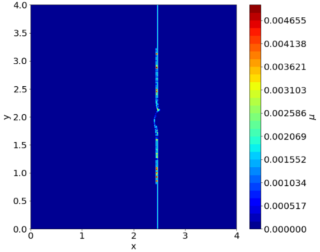

density t=0.35
3.0
2.5
2.0
1.5
1.0
2.0
2.5
3.0
3.5
1.3655
1.3138
1.2621
1.2103
1.1586
1.1069
1.0552
1.0034
0.9517
0.9000


density t=0.8
3.0
2.5
2.0
1.5
1.0
2.0
2.5
3.0
3.5
1.3655
1.3138
1.2621
1.2103
1.1586
1.1069
1.0552
1.0034
0.9517
0.9000

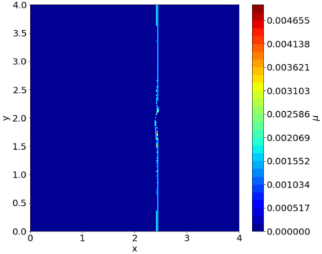
2D Euler equations: Shock-vortex



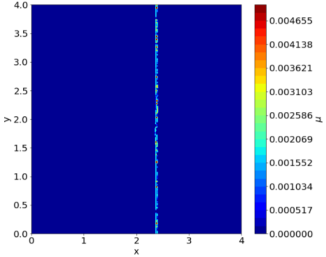
t=0.1



t = 0.35



t=0.5



t = 0.8

Conclusions

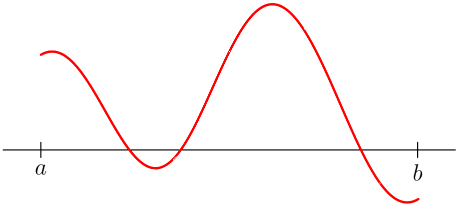
- Demonstrated that deep learning can be used as a surrogate – both for classification and regression.
- Networks trained (once) offline and then used for any model (conservation law).
- Useful in constructing methods free of problem-dependent parameters.
- Need to use domain-knowledge to construct training sets.
- Must have reasonable expectations from networks – cannot solve every problem!

Don't replace but enhance existing numerical frameworks with deep networks

* Don't replace but enhance existing numerical frameworks with deep networks

*if possible

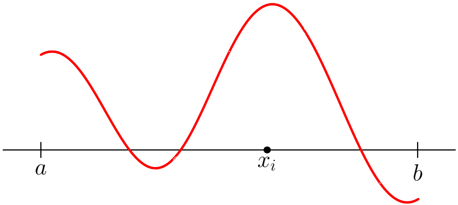
Generating training data



Data sampling is achieved by

- Choose a known function $u(x)$

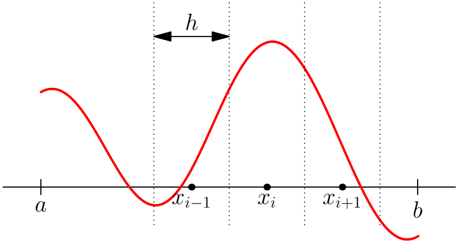
Generating training data



Data sampling is achieved by

- Choose a known function $u(x)$
- Pick a point x_i

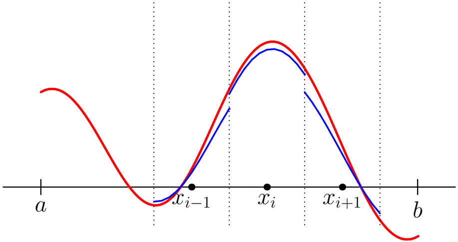
Generating training data



Data sampling is achieved by

- Choose a known function $u(x)$
- Pick a point x_i
- Pick a cell size h and make stencil

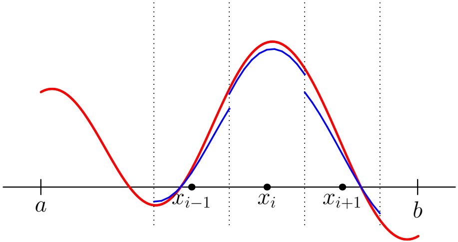
Generating training data



Data sampling is achieved by

- Choose a known function $u(x)$
- Pick a point x_i
- Pick a cell size h and make stencil
- Pick a degree r and approximate

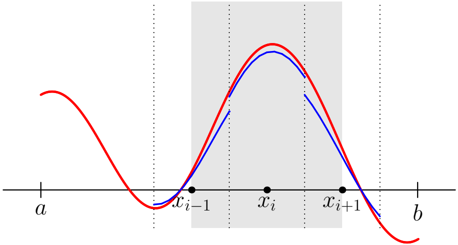
Generating training data



Data sampling is achieved by

- Choose a known function $u(x)$
- Pick a point x_i
- Pick a cell size h and make stencil
- Pick a degree r and approximate
- Extract needed data $[\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}]$

Generating training data



Data sampling is achieved by

- Choose a known function $u(x)$
- Pick a point x_i
- Pick a cell size h and make stencil
- Pick a degree r and approximate
- Extract needed data $[\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}]$
- Flag cell if discontinuity in $[x_{i-\frac{1}{2}} - h/2, x_{i+\frac{1}{2}} + h/2]$