An artificial neural network for detecting discontinuities

Deep Ray
MCSS, School of Basic Sciences
EPFL, Switzerland
deep.ray@epfl.ch
http://deepray.github.io

joint work with Jan S. Hesthaven

TIFR-CAM, Bangalore
4 January 2018

Outline

- Conservation laws and RKDG formulation
- Troubled-cell indicators and limited reconstruction
- Artificial Neural Networks and MLPs
- An MLP-based detector
- Numerical results

Conservation laws

Cauchy problem for a scalar conservation law

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0 \qquad \forall \quad (x,t) \in [a,b] \times [0,T]$$
$$u(x,0) = u_0(x) \qquad \forall \quad x \in [a,b]$$

Solutions approximated using

- Finite difference/volume schemes
- Discontinuous Galerkin schemes
- Spectral methods
- ....

Conservation laws

Cauchy problem for a scalar conservation law

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0 \qquad \forall \quad (x,t) \in [a,b] \times [0,T]$$
$$u(x,0) = u_0(x) \qquad \forall \quad x \in [a,b]$$

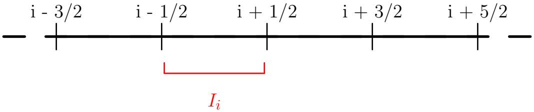
Solutions approximated using

- Finite difference/volume schemes
- Discontinuous Galerkin schemes
- Spectral methods
- ....

Non-linearity $\implies$ Discontinuities in finite time

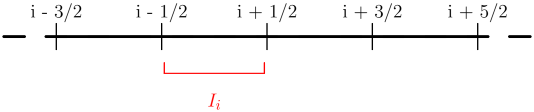
$\implies$ High-order methods suffer from Gibbs oscillations

Runge-Kutta discontinuous Galerkin schemes



Discretize domain into N cells $\bigcup_{i=1}^{N} I_i, \;\; I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$.

Runge-Kutta discontinuous Galerkin schemes

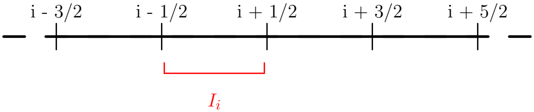


Discretize domain into N cells $\bigcup_{i=1}^{N} I_i, \;\; I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$.

Define space of broken polynomials

$$V_h^r = \{v \in L^2([a,b]) : v\big|_{I_i} \in \mathbb{P}_r(I_i), \quad 1 \le i \le N\}$$

Runge-Kutta discontinuous Galerkin schemes



Discretize domain into N cells $\bigcup_{i=1}^{N} I_i$, $I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$.

Define space of broken polynomials

$$V_h^r = \{v \in L^2([a,b]) : v\big|_{I_i} \in \mathbb{P}_r(I_i), \quad 1 \le i \le N\}$$

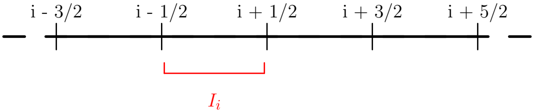
Semi-discrete scheme

Find $u_h(.,t) \in V_h^r$ s.t. $\forall\ v_h \in V_h^r$

$$\int_{I_i} \left[\frac{\partial u_h}{\partial t} v_h - f(u_h)\frac{\partial v_h}{\partial x}\right] \mathrm{d}x + \hat{f}_{i+\frac{1}{2}}(t) v_h(x^-_{i+\frac{1}{2}}) - \hat{f}_{i-\frac{1}{2}}(t) v_h(x^+_{i-\frac{1}{2}}) = 0$$

where $v_h(x^{\pm}) = \lim_{\epsilon \downarrow 0} v_h(x \pm \epsilon)$

Runge-Kutta discontinuous Galerkin schemes



Discretize domain into N cells $\bigcup_{i=1}^{N} I_i$, $I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$.

Define space of broken polynomials

$$V_h^r = \{v \in L^2([a,b]) : v\big|_{I_i} \in \mathbb{P}_r(I_i), \quad 1 \le i \le N\}$$

Semi-discrete scheme

Find $u_h(.,t) \in V_h^r$ s.t. $\forall\ v_h \in V_h^r$ *numerical flux*

$$\int_{I_i} \left[\frac{\partial u_h}{\partial t} v_h - f(u_h) \frac{\partial v_h}{\partial x} \right] \mathrm{d}x + \hat{f}_{i+\frac{1}{2}}(t) v_h(x_{i+\frac{1}{2}}^-) - \hat{f}_{i-\frac{1}{2}}(t) v_h(x_{i-\frac{1}{2}}^+) = 0$$

where $v_h(x^\pm) = \lim_{\epsilon \downarrow 0} v_h(x \pm \epsilon)$

Runge-Kutta discontinuous Galerkin schemes

In each cell I_i:

$$u_h(x,t) = \sum_{j=0}^{r} u_{ij}(t)\phi_{ij}(x), \quad x \in I_i$$

Runge-Kutta discontinuous Galerkin schemes

In each cell I_i:

$$u_h(x,t) = \sum_{j=0}^{r} u_{ij}(t)\phi_{ij}(x), \quad x \in I_i$$

Need to solve for $U^{(i)}(t) = [u_{i0}, ..., u_{ir}]$.

- Choose basis $\{\phi_{ij}\}$ (Legendre Polynomials, etc)
- High-order quadrature (Gauss-Legendre, etc)

Runge-Kutta discontinuous Galerkin schemes

In each cell I_i:

$$u_h(x,t) = \sum_{j=0}^{r} u_{ij}(t)\phi_{ij}(x), \quad x \in I_i$$

Need to solve for $U^{(i)}(t) = [u_{i0}, ..., u_{ir}]$.

- Choose basis $\{\phi_{ij}\}$ (Legendre Polynomials, etc)
- High-order quadrature (Gauss-Legendre, etc)

$$M^{(i)} \frac{\mathrm{d}U^{(i)}}{\mathrm{d}t} = R^{(i)}(U(t))$$

Runge-Kutta discontinuous Galerkin schemes

In each cell I_i:

$$u_h(x,t) = \sum_{j=0}^{r} u_{ij}(t)\phi_{ij}(x), \quad x \in I_i$$

Need to solve for $U^{(i)}(t) = [u_{i0}, ..., u_{ir}]$.

- Choose basis $\{\phi_{ij}\}$ (Legendre Polynomials, etc)
- High-order quadrature (Gauss-Legendre, etc)

$$\frac{\mathrm{d}U^{(i)}}{\mathrm{d}t} = \left(M^{(i)}\right)^{-1} R^{(i)}(U(t)) = L^{(i)}(U(t))$$

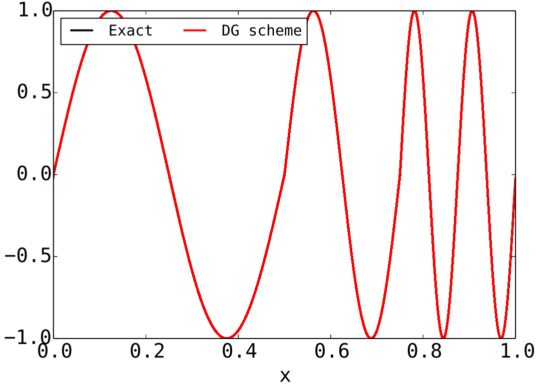
Solve ODEs using time-marching scheme (Runge-Kutta)

Is the scheme good enough?

Runge-Kutta discontinuous Galerkin schemes

Scheme works well for smooth solutions

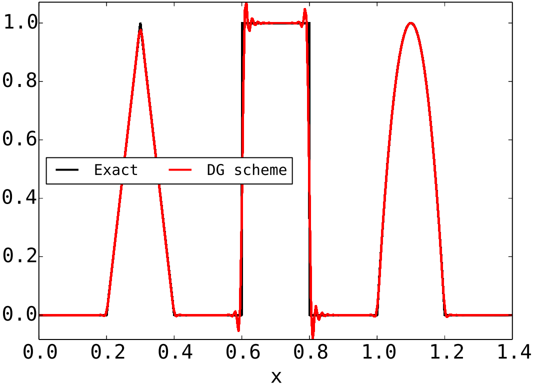
$$u_t + u_x = 0, \quad \Omega = [0,1], \quad T_f = 1, \quad r = 6, \quad N = 100$$



Runge-Kutta discontinuous Galerkin schemes

However, Gibbs oscillations near discontinuities

$$u_t + u_x = 0, \quad \Omega = [0, 1.4], \quad T_f = 1.4, \quad r = 4, \quad N = 100$$



Is the scheme good enough?
Answer: No!

What do we do about the oscillations?

Is the scheme good enough?
Answer: No!

What do we do about the oscillations?
Answer: Correct the solution near discontinuities

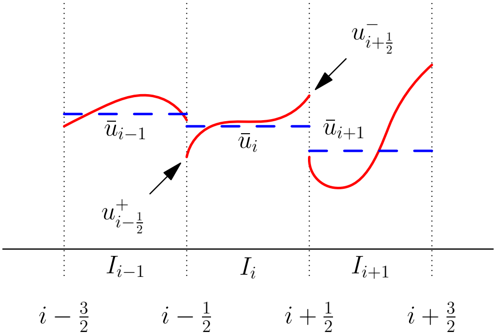
Limiting the solution (Qiu and Shu, 2005)

After each RK step:

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

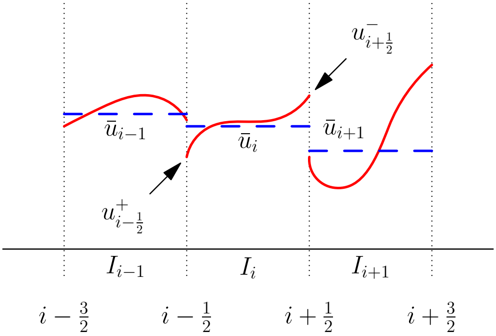
Limiting the solution (Qiu and Shu, 2005)

Identification: For each cell I_i, get $[\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}]$



Limiting the solution (Qiu and Shu, 2005)

Identification: For each cell I_i, get $[\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}]$

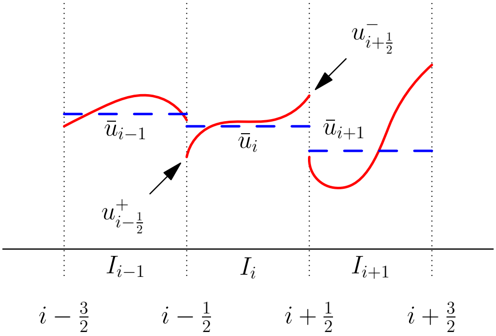


Evaluate 4 differences

$$\Delta^- u_i = \overline{u}_i - \overline{u}_{i-1}, \qquad \Delta^+ u_i = \overline{u}_{i+1} - \overline{u}_i,$$
$$\check{u}_i = \overline{u}_i - u^+_{i-\frac{1}{2}}, \qquad \hat{u}_i = u^-_{i+\frac{1}{2}} - \overline{u}_i$$

Limiting the solution (Qiu and Shu, 2005)

Identification: For each cell I_i, get $[\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}]$

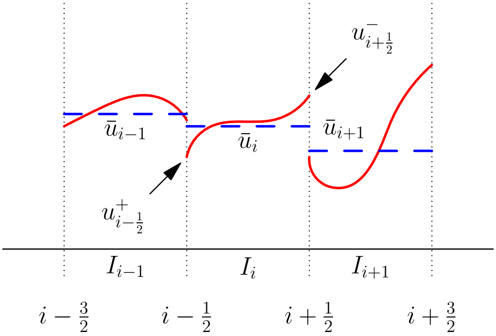


Modify interface values

$$\widetilde{u}^+_{i-\frac{1}{2}} = \overline{u}_i + \mathcal{F}\left(\check{u}_i, \Delta^- u_i, \Delta^+ u_i\right)$$
$$\widetilde{u}^-_{i+\frac{1}{2}} = \overline{u}_i - \mathcal{F}\left(\hat{u}_i, \Delta^- u_i, \Delta^+ u_i\right)$$

Limiting the solution (Qiu and Shu, 2005)

Identification: For each cell I_i, get $[\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}]$



Flag I_i as troubled-cell if

$$\widetilde{u}^+_{i-\frac{1}{2}} \neq u^+_{i-\frac{1}{2}} \quad \text{or} \quad \widetilde{u}^-_{i+\frac{1}{2}} \neq u^-_{i+\frac{1}{2}}$$

Search for the elusive M

We consider the following limiter-based indicators $\mathcal{F}$:

- Minmod limiter:

$$\mathcal{F}^{\mathsf{mm}}(a,b,c) = \begin{cases} s.\min(|a|,|b|,|c|), & \text{if } s = \mathrm{sign}(a) = \mathrm{sign}(b) = \mathrm{sign}(c) \\ 0, & \text{otherwise} \end{cases}$$

Disadvantage: Flags cell with smooth extrema

Search for the elusive M

We consider the following limiter-based indicators $\mathcal{F}$:

- Minmod limiter:

$$\mathcal{F}^{\text{mm}}(a,b,c) = \begin{cases} s.\min(|a|,|b|,|c|), & \text{if } s = \text{sign}(a) = \text{sign}(b) = \text{sign}(c) \\ 0, & \text{otherwise} \end{cases}$$

Disadvantage: Flags cell with smooth extrema

- TVB limiter: Depends on h and tunable parameter M

$$\mathcal{F}^{\text{tvb}}(a,b,c,h,M) = \begin{cases} a, & \text{if } |a| \le Mh^2 \\ \mathcal{F}^{\text{mm}}(a,b,c), & \text{otherwise} \end{cases}$$

M is proportional to second derivative at smooth extreme

Disadvantage: M is problem dependent

Limiting the solution (Qiu and Shu, 2005)

Limited reconstruction: In troubled cells:

- Project u_h to $\mathbb{P}_1$

$$u_h = \overline{u}_i + \left(\frac{x - x_i}{\frac{1}{2}\Delta x_i}\right) s_i + \text{ H.O.T.}$$

Limiting the solution (Qiu and Shu, 2005)

Limited reconstruction: In troubled cells:

- Project u_h to $\mathbb{P}_1$

$$\widetilde{u}_h = \Pi^1 u_h = \overline{u}_i + \left(\frac{x - x_i}{\frac{1}{2}\Delta x_i} \right) s_i$$

Limiting the solution (Qiu and Shu, 2005)

Limited reconstruction: In troubled cells:

- Project u_h to $\mathbb{P}_1$

$$\widetilde{u}_h = \Pi^1 u_h = \overline{u}_i + \left(\frac{x - x_i}{\frac{1}{2}\Delta x_i} \right) s_i$$

- Limit slope

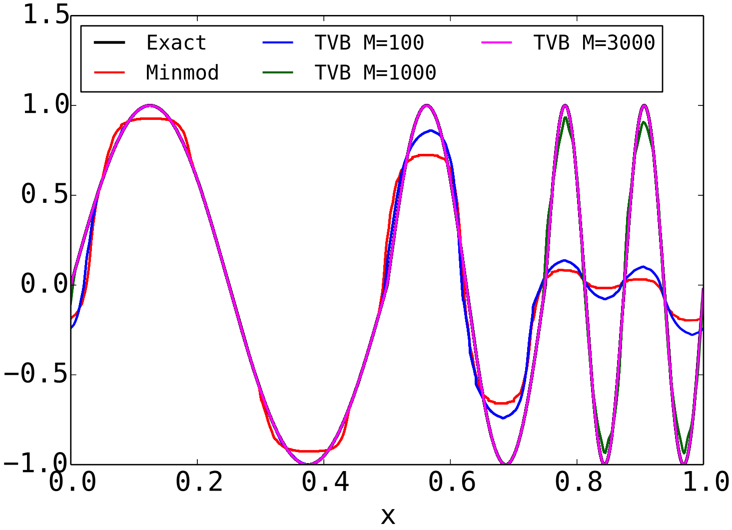
$$\widetilde{u}_h^{(m)} = \overline{u}_i + \left(\frac{x - x_i}{\frac{1}{2}\Delta x_i} \right) \widetilde{s}_i$$

where

$$\widetilde{s}_i = \mathcal{Q}(s_i, \overline{u}_i - \overline{u}_{i-1}, \overline{u}_{i+1} - \overline{u}_i)$$

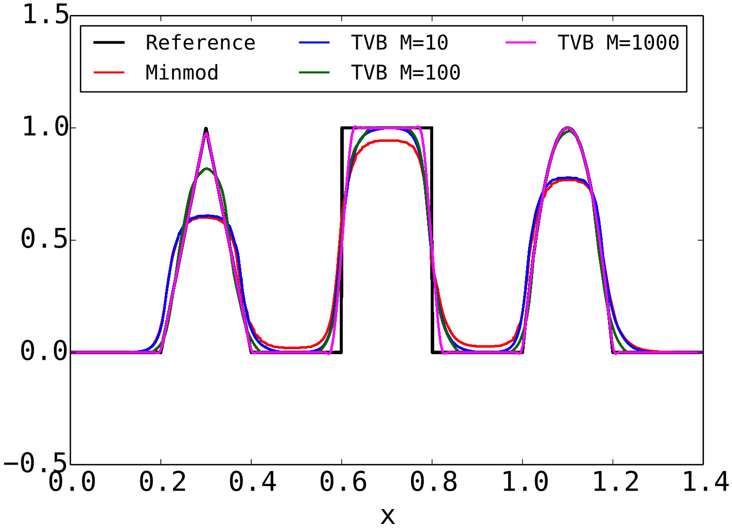
Varying M

Clipping near smooth extrema



Varying M

Good discontinuous solution only for suitable M



Issue: A single M doesn't fit all scenarios

Issue: A single M doesn't fit all scenarios

Objective: Find a parameter-free troubled-cell indicator which doesn't flag smooth extrema

Approximating functions

Consider the unknown function

$$G : \mathbb{R}^n \mapsto \mathbb{R}^m$$

whose value is know only on a set,

$$\{(\mathbf{X}_p, \mathbf{Y}_p\}\}_{p \in \Lambda}, \quad s.t. \quad G(\mathbf{X}_p) = \mathbf{Y}_p$$

Linear regression is not suitable for highly-nonlinear G.

Approximating functions

Consider the unknown function

$$G : \mathbb{R}^n \mapsto \mathbb{R}^m$$

whose value is know only on a set,

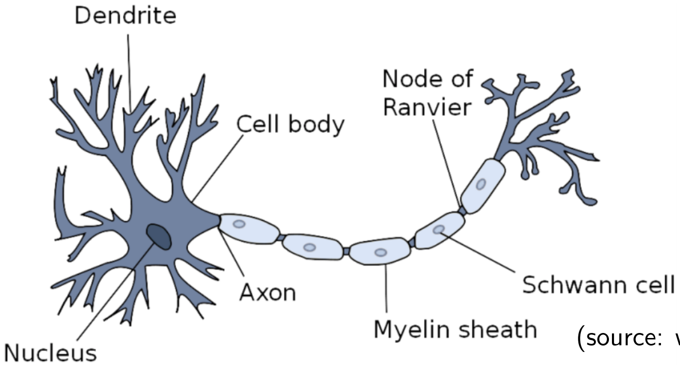
$$\{(\mathbf{X}_p, \mathbf{Y}_p)\}_{p\in\Lambda}, \quad s.t. \quad G(\mathbf{X}_p) = \mathbf{Y}_p$$

Linear regression is not suitable for highly-nonlinear G.

Learn like the human brain!!

Artificial Neural Network (ANN)

A biological neuron



(source: wikipedia)

Network gains knowledge by creating and modifying connections

Artificial Neural Network (ANN)

An ANN is given by $(\mathcal{N}, \mathcal{V}, w)$ where

$$\begin{aligned}
\mathcal{N} &\longrightarrow \text{set of neurons} \\
\mathcal{V} &\longrightarrow \text{set of connections } \{(i,j) : 1 \leq i,j \leq |\mathcal{N}|\} \\
w : \mathcal{V} \mapsto \mathbb{R} &\longrightarrow \text{connection weight } \{w_{i,j} : 1 \leq i,j \leq |\mathcal{N}|\}
\end{aligned}$$

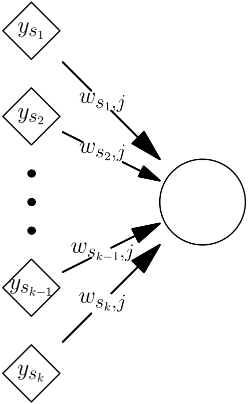
Artificial Neural Network (ANN)

An ANN is given by $(\mathcal{N}, \mathcal{V}, w)$ where

$$\mathcal{N} \longrightarrow \text{set of neurons}$$

$$\mathcal{V} \longrightarrow \text{set of connections} \quad \{(i,j) : 1 \leq i,j \leq |\mathcal{N}|\}$$

$$w : \mathcal{V} \mapsto \mathbb{R} \longrightarrow \text{connection weight} \quad \{w_{i,j} : 1 \leq i,j \leq |\mathcal{N}|\}$$



Artificial Neural Network (ANN)

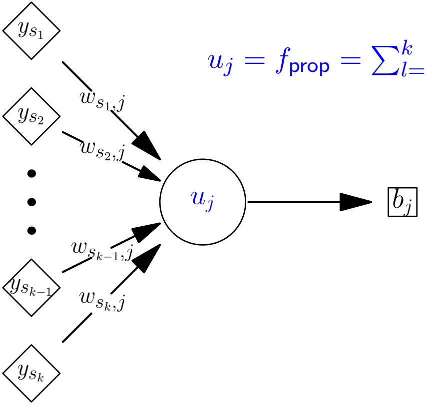
An ANN is given by $(\mathcal{N}, \mathcal{V}, w)$ where

$$\mathcal{N} \longrightarrow \text{set of neurons}$$

$$\mathcal{V} \longrightarrow \text{set of connections} \quad \{(i,j) : 1 \leq i,j \leq |\mathcal{N}|\}$$

$$w : \mathcal{V} \mapsto \mathbb{R} \longrightarrow \text{connection weight} \quad \{w_{i,j} : 1 \leq i,j \leq |\mathcal{N}|\}$$

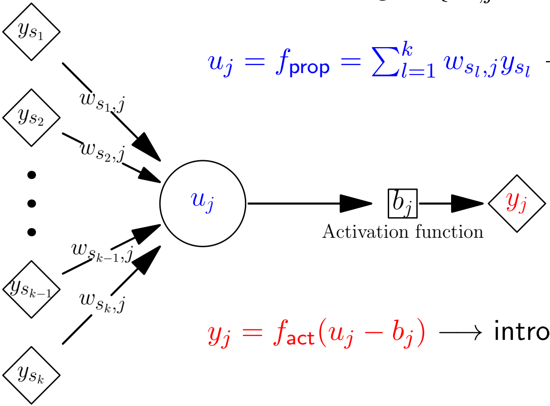
$u_j = f_{\text{prop}} = \sum_{l=1}^{k} w_{s_l,j} y_{s_l}$ $\longrightarrow$ linear



Artificial Neural Network (ANN)

An ANN is given by $(\mathcal{N}, \mathcal{V}, w)$ where

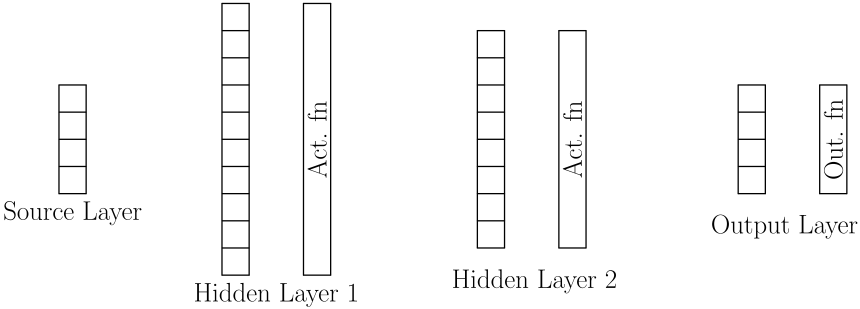
$$\begin{aligned} \mathcal{N} &\longrightarrow \text{set of neurons} \\ \mathcal{V} &\longrightarrow \text{set of connections } \{(i,j) : 1 \le i,j \le |\mathcal{N}|\} \\ w : \mathcal{V} \mapsto \mathbb{R} &\longrightarrow \text{connection weight } \{w_{i,j} : 1 \le i,j \le |\mathcal{N}|\} \end{aligned}$$



$u_j = f_{\text{prop}} = \sum_{l=1}^{k} w_{s_l,j} y_{s_l} \longrightarrow$ linear

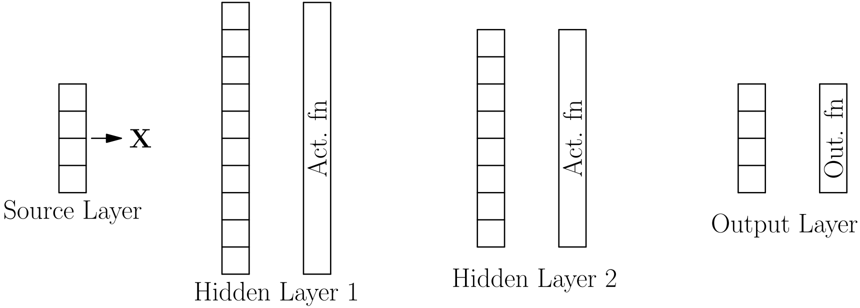
$y_j = f_{\text{act}}(u_j - b_j) \longrightarrow$ introduces non-linearity

Multilayer perceptron (MLP)



n_1 neurons in Hidden layer 1, n_2 neurons in Hidden layer 2

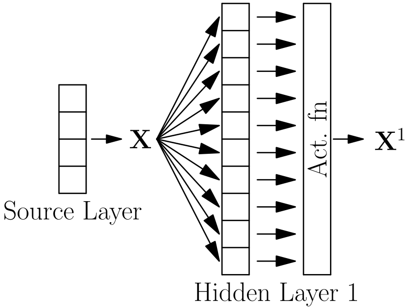
Multilayer perceptron (MLP)



n_1 neurons in Hidden layer 1, n_2 neurons in Hidden layer 2

$\mathbf{X} \in \mathbb{R}^{N_I}$

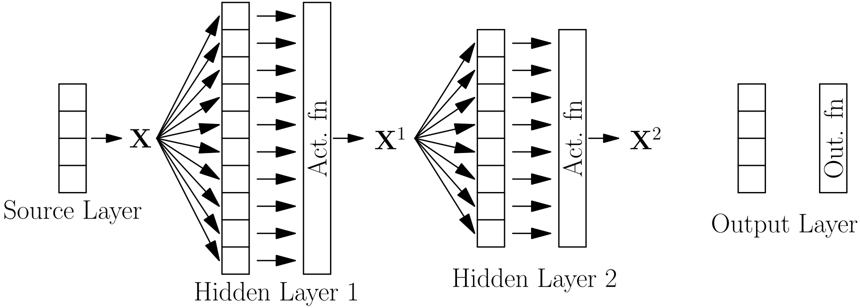
Multilayer perceptron (MLP)



n_1 neurons in Hidden layer 1, n_2 neurons in Hidden layer 2

$$\mathbf{X} \in \mathbb{R}^{N_I} \longrightarrow \underbrace{W^1}_{\mathbb{R}^{n_1 \times N_I}} \mathbf{X} + \underbrace{b^1}_{\mathbb{R}^{n_1}} \longrightarrow \text{Act. fn.} \longrightarrow \mathbf{X}^1 \in \mathbb{R}^{n_1}$$

Multilayer perceptron (MLP)

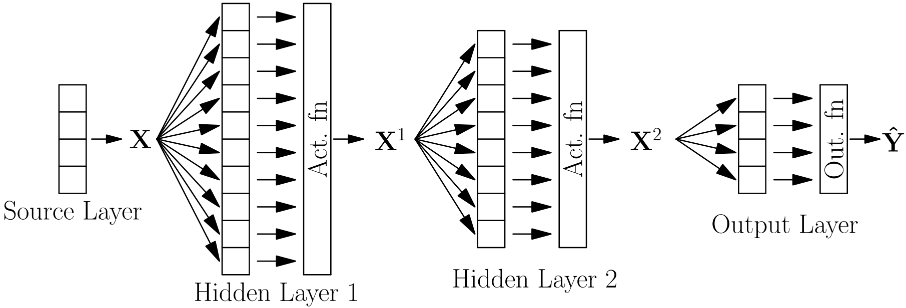


n_1 neurons in Hidden layer 1, n_2 neurons in Hidden layer 2

$$\mathbf{X} \in \mathbb{R}^{N_I} \longrightarrow \underbrace{W^1}_{\mathbb{R}^{n_1 \times N_I}} \mathbf{X} + \underbrace{b^1}_{\mathbb{R}^{n_1}} \longrightarrow \text{Act. fn.} \longrightarrow \mathbf{X}^1 \in \mathbb{R}^{n_1}$$

$$\mathbf{X}^1 \longrightarrow W^2\mathbf{X}^1 + b^2 \longrightarrow \text{Act. fn.} \longrightarrow \mathbf{X}^2 \in \mathbb{R}^{n_2}$$

Multilayer perceptron (MLP)



n_1 neurons in Hidden layer 1, n_2 neurons in Hidden layer 2

$$\mathbf{X} \in \mathbb{R}^{N_I} \longrightarrow \underbrace{W^1}_{\mathbb{R}^{n_1 \times N_I}} \mathbf{X} + \underbrace{b^1}_{\mathbb{R}^{n_1}} \longrightarrow \text{Act. fn.} \longrightarrow \mathbf{X}^1 \in \mathbb{R}^{n_1}$$

$$\mathbf{X}^1 \longrightarrow W^2\mathbf{X}^1 + b^2 \longrightarrow \text{Act. fn.} \longrightarrow \mathbf{X}^2 \in \mathbb{R}^{n_2}$$

$$\mathbf{X}^2 \longrightarrow W^O\mathbf{X}^2 + b^2 \longrightarrow \text{Out. fn.} \longrightarrow \hat{\mathbf{Y}} \in \mathbb{R}^{N_O}$$

Multilayer perceptron (MLP)

Problem statement (Supervised learning)

Given the data $\{(\mathbf{X}_p, \mathbf{Y}_p)\}_p$ and the predictions

$$\hat{\mathbf{Y}}_p = G_{MLP}(\mathbf{X}_p)$$

and a cost functional $C(\mathbf{Y}, \hat{\mathbf{Y}})$. Find the weights W and biases b of the MLP which minimize C.

Multilayer perceptron (MLP)

Problem statement (Supervised learning)

Given the data $\{(\mathbf{X}_p, \mathbf{Y}_p)\}_p$ and the predictions

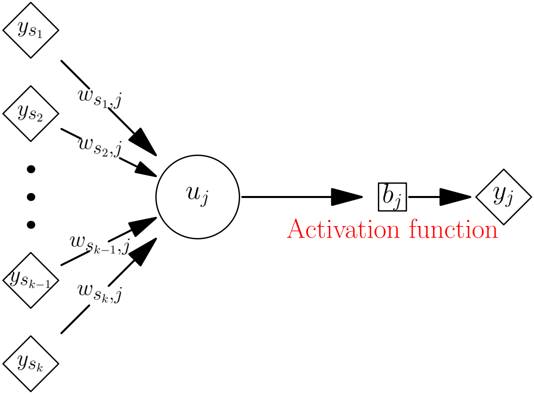
$$\hat{\mathbf{Y}}_p = G_{MLP}(\mathbf{X}_p)$$

and a cost functional $C(\mathbf{Y}, \hat{\mathbf{Y}})$. Find the weights W and biases b of the MLP which minimize C.

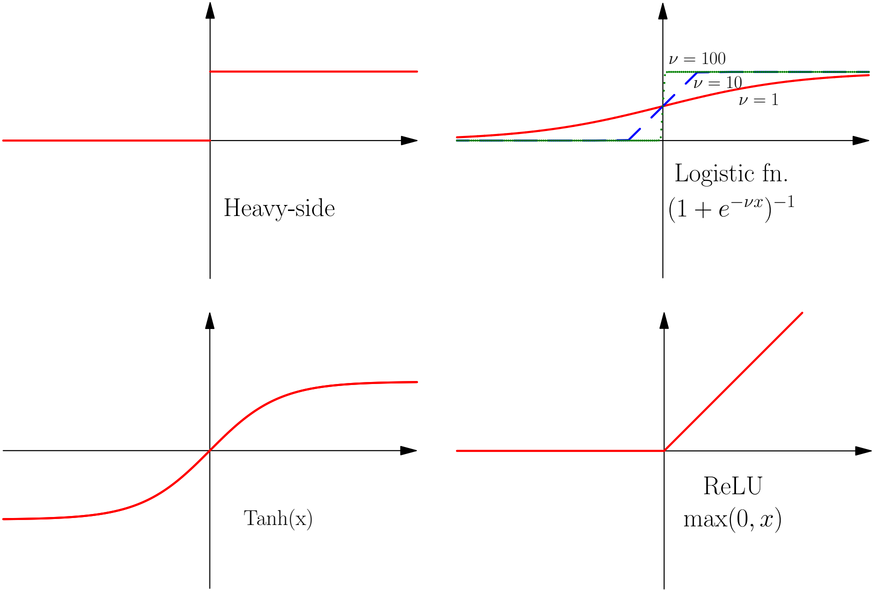
Remarks:

- The network is trained offline using given data
- Optimize using gradient descent, Adams, etc.
- C convex wrt $\mathbf{Y}, \hat{\mathbf{Y}}$ does not imply C convex wrt W, b
- Capacity to generalize

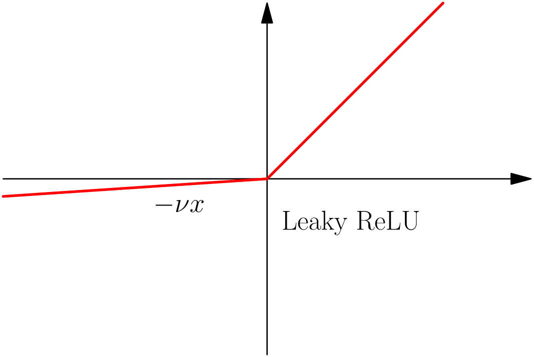
Activation functions



Activation functions



Activation functions



An MLP-based indicator

- Input $\mathbf{X} = [\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}] \in \mathbb{R}^5$

An MLP-based indicator

- Input $\mathbf{X} = [\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}] \in \mathbb{R}^5$
- 5 Hidden Layers with width 256, 128, 64, 32, 16

An MLP-based indicator

- Input $\mathbf{X} = [\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}] \in \mathbb{R}^5$
- 5 Hidden Layers with width 256, 128, 64, 32, 16
- ReLU activation function with $\nu = 10^{-3}$

An MLP-based indicator

- Input $\mathbf{X} = [\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}] \in \mathbb{R}^5$
- 5 Hidden Layers with width 256, 128, 64, 32, 16
- ReLU activation function with $\nu = 10^{-3}$
- Softmax output function

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

An MLP-based indicator

- Input $\mathbf{X} = [\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}] \in \mathbb{R}^5$
- 5 Hidden Layers with width 256, 128, 64, 32, 16
- ReLU activation function with $\nu = 10^{-3}$
- Softmax output function

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

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

An MLP-based indicator

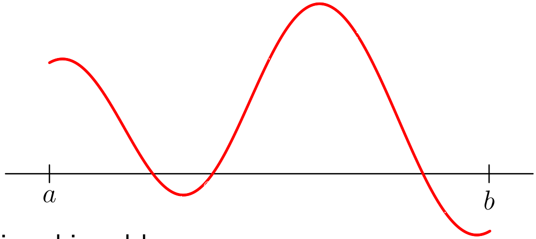
- Input $\mathbf{X} = [\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}] \in \mathbb{R}^5$
- 5 Hidden Layers with width 256, 128, 64, 32, 16
- ReLU activation function with $\nu = 10^{-3}$
- Softmax output function

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

- Output $\hat{Y} = [\hat{Y}^{(0)}, \hat{Y}^{(1)}] \in [0,1]^2$
- Cost functional: regularized cross-entropy

$$C = -\sum_{i=1}^{N} \left[Y_i^{(0)} \log\left(\hat{Y}_i^{(0)}\right) + Y_i^{(1)} \log\left(\hat{Y}_i^{(1)}\right) \right] + \underbrace{\frac{\lambda}{2} ||W||_2^2}_{\text{regularization}}$$

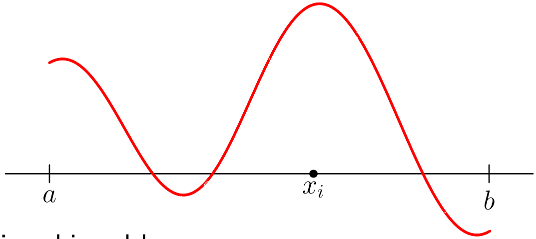
Generating the training data



Data sampling is achieved by

- Choose a known function $u(x)$

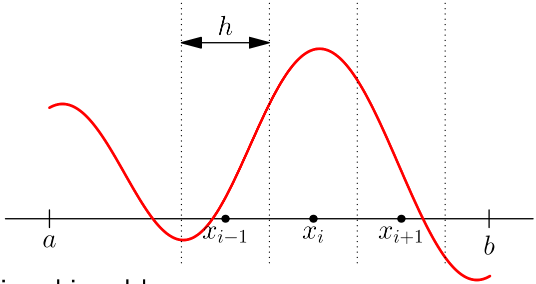
Generating the training data



Data sampling is achieved by

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

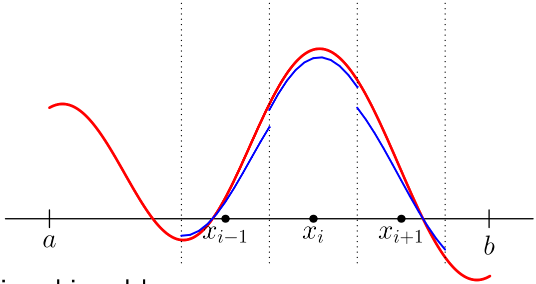
Generating the training data



Data sampling is achieved by

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

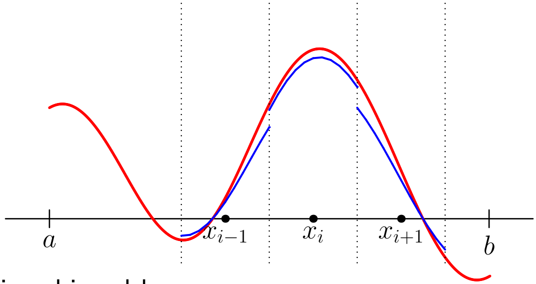
Generating the training data



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

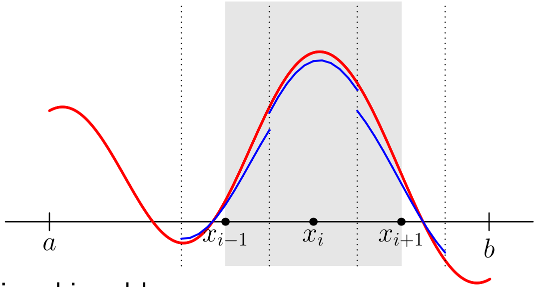
Generating the training data



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 the 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 $[x_{i-\frac{1}{2}} - h/2, x_{i+\frac{1}{2}} + h/2]$

Generating the training data

$\mathbf{u}(\mathbf{x})$	**Domain**	**Additional parameters**
$\sin(4\pi x)$	$[0, 1]$	–
ax	$[-1, 1]$	$a \in \mathbb{R}$
$a\|x\|$	$[-1, 1]$	$a \in \mathbb{R}$
$ul.(x < x_0) + ur.(x > x_0)$ (only troubled-cells selected)	$[-1, 1]$	$(u_l, u_r) \in [-1, 1]^2$ $x_0 \in [-0.76, 0.76]$

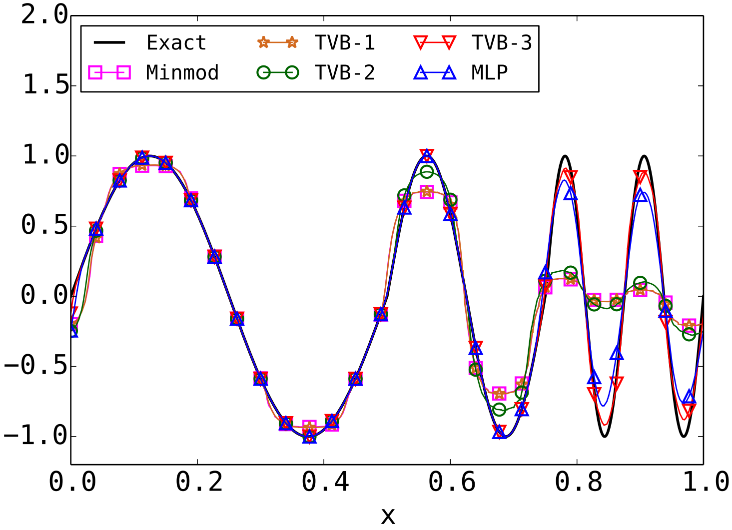
Parameters varied:

- Mesh size h
- Approximating polynomial degree r
- Additional parameters (if available)
- good-cells = 11220, troubled-cells = 13060

How well does the trained network really work?

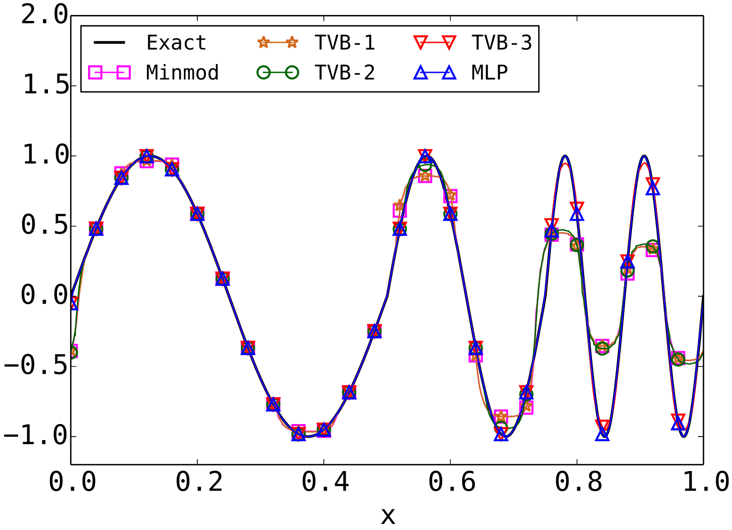
Linear advection (r=3)

TVB-1 (M=10), TVB-2 (M=100), TVB-3 (M=1000), N=100



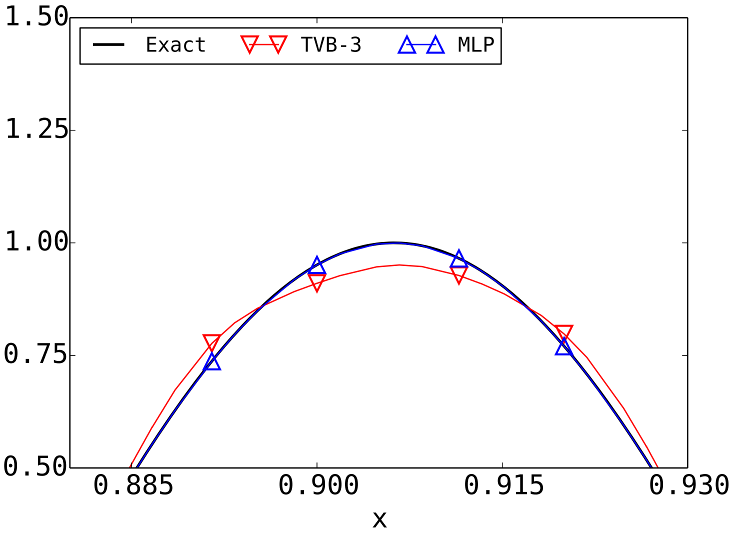
Linear advection (r=3)

TVB-1 (M=10), TVB-2 (M=100), TVB-3 (M=1000), N=150



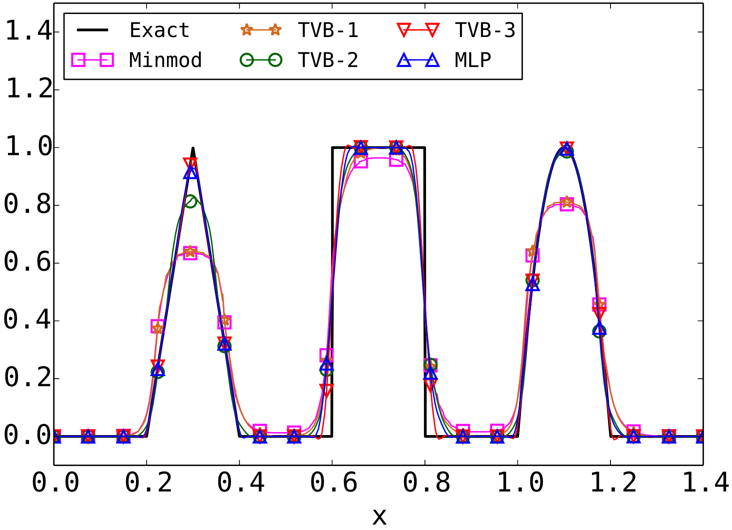
Linear advection (r=3)

TVB-1 (M=10), TVB-2 (M=100), TVB-3 (M=1000), N=150



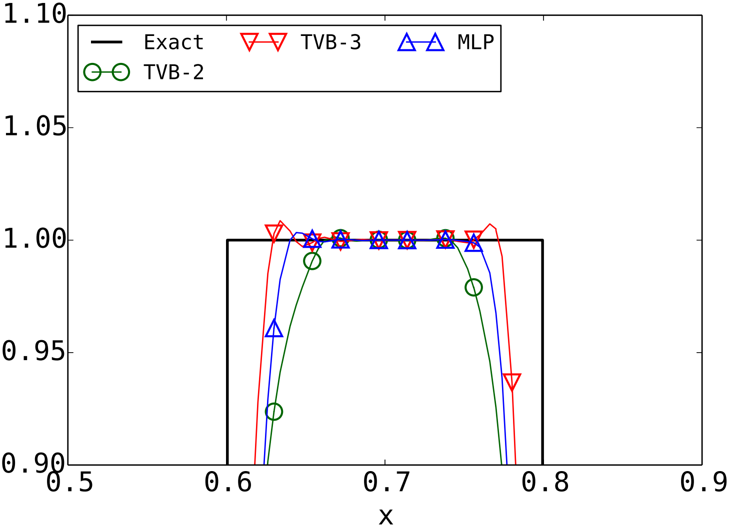
Linear advection (r=3)

TVB-1 (M=10), TVB-2 (M=100), TVB-3 (M=1000), N=100



Linear advection (r=3)

TVB-1 (M=10), TVB-2 (M=100), TVB-3 (M=1000), N=100



Burgers equation: Collision of shocks

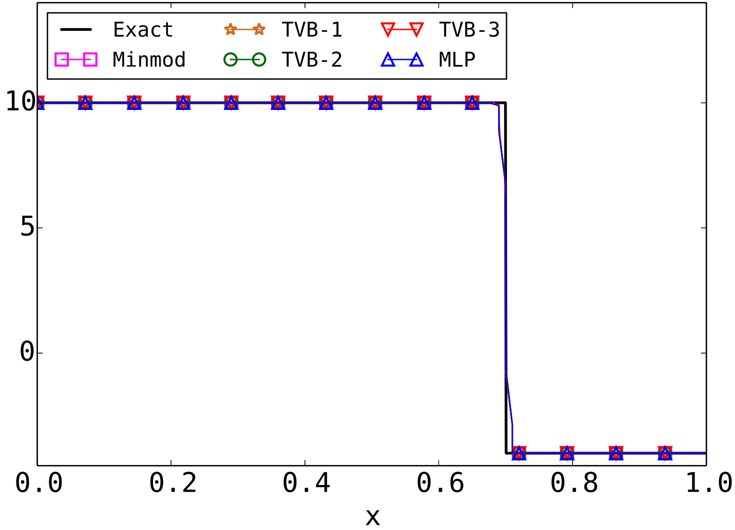
Initial condition has three shocks

$$u_0(x) = \begin{cases} 10 & \text{if } x \leq 0.2 \ , \\ 6 & \text{if } 0.2 < x \leq 0.4 \ , \\ 0 & \text{if } 0.4 < x \leq 0.6 \ , \\ -4 & \text{if } 0.6 < x \ , \end{cases}$$

At t=0.1, only a single shock survives

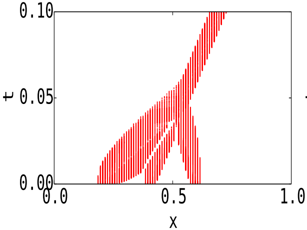
Burgers equation: Collision of shocks

$N = 100$, $r = 4$

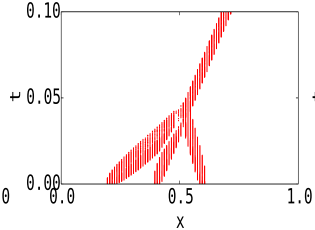


Burgers equation: Collision of shocks

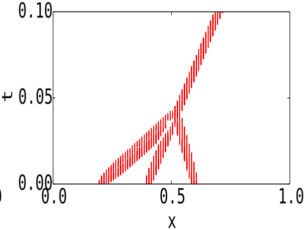
Evolution of flagged cells



minmod

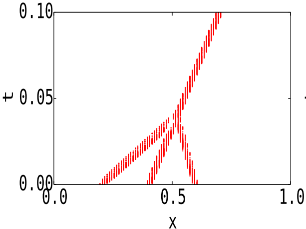


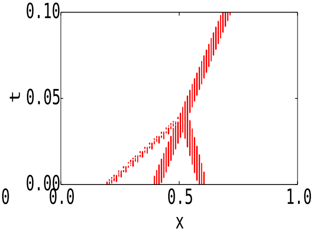
TVB-1



TVB-2

TVB-3





MLP

Buckley-Leverett equations

Non-convex, non-linear flux

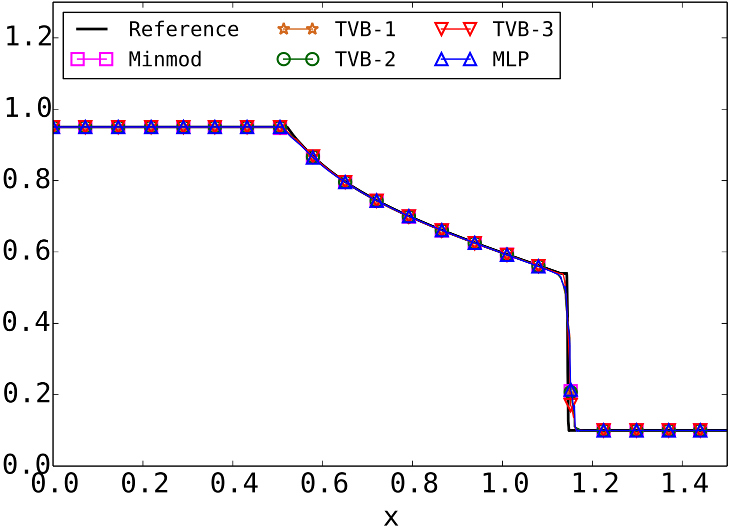
$$f(u) = \frac{u^2}{u^2 + 0.5(1-u)^2}$$

Initial condition given by

$$u_0(x) = \begin{cases} 0.95 & \text{if } x < 0.5 \\ 0.1 & \text{if } x > 0.5 \end{cases},$$

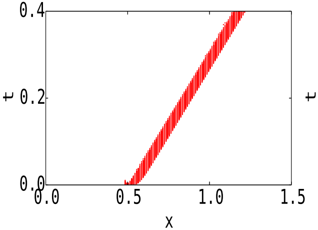
Buckley-Leverett equations

Solution simulated till t=0.4, with N=150, r=4

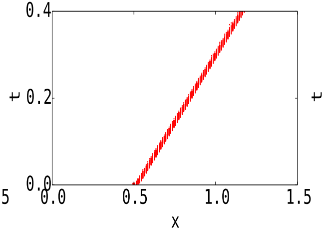


Buckley-Leverett equations

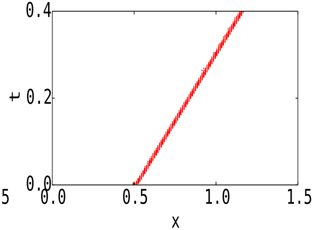
Evolution of flagged cells



minmod

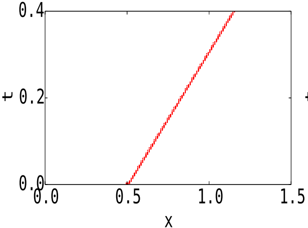


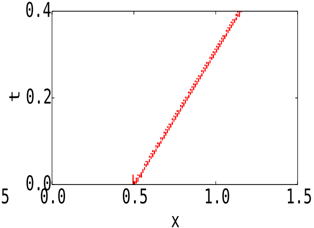
TVB-1



TVB-2

TVB-3





MLP

Systems: Shallow water equations

One-dimensional SWE

$$\frac{\partial}{\partial t}\begin{bmatrix} D \\ Du \end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix} Du \\ Du^2 + \frac{1}{2}gD \end{bmatrix}$$

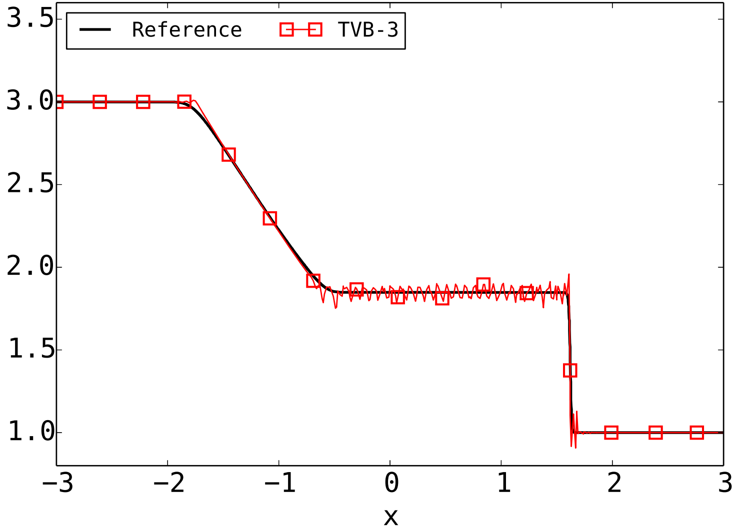
Initial condition given by

$$D_0(x) = \begin{cases} 3 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \end{cases}, \qquad u_0(x) = 0,$$

Solution simulated till t=1, with N=100, r=4

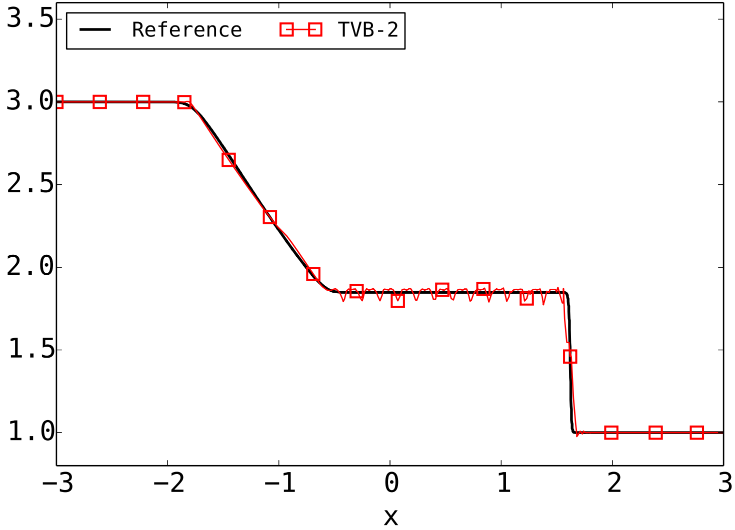
Systems: Shallow water equations

Depth profile:



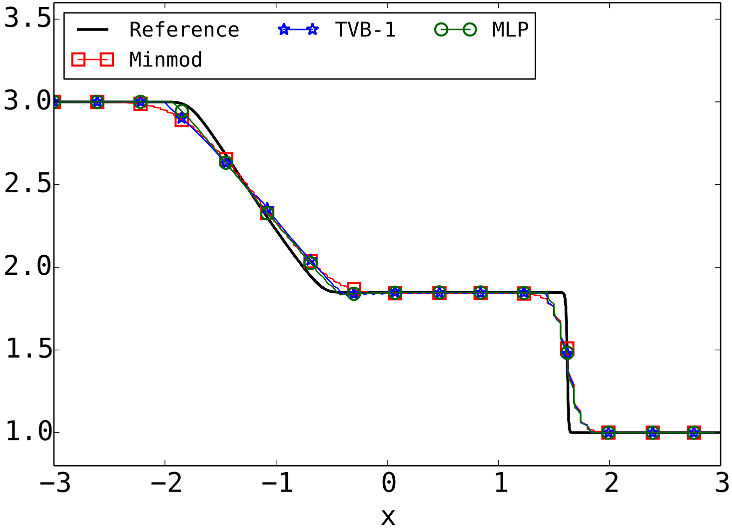
Systems: Shallow water equations

Depth profile:



Systems: Shallow water equations

Depth profile:



Conclusion

- Constructed a parameter-free MLP-based indicator
- Works for 1D scalar and systems of conservation laws
- Flags the necessary number of cells

Conclusion

- Constructed a parameter-free MLP-based indicator
- Works for 1D scalar and systems of conservation laws
- Flags the necessary number of cells

Further issues:

- Testing on Euler equations
- Non-uniform grids
- Unstructured 2D and 3D grids
- Other forms of inputs for training

Conclusion

- Constructed a parameter-free MLP-based indicator
- Works for 1D scalar and systems of conservation laws
- Flags the necessary number of cells

Further issues:

- Testing on Euler equations
- Non-uniform grids
- Unstructured 2D and 3D grids
- Other forms of inputs for training

Questions?