An artificial neural network for detecting discontinuities

Deep Ray
MATH-MCSS,
EPFL, Switzerland
deep.ray@epfl.ch
http://deepray.github.io

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

joint work with Jan S. Hesthaven

7th International Conference on High Performance Scientific Computing
Hanoi, 20 March 2018

Runge-Kutta discontinuous Galerkin schemes

Consider

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

$$u(x,0) = u_0(x) \qquad \forall \quad x \in [a,b]$$

Runge-Kutta discontinuous Galerkin schemes

Consider

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

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

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

Runge-Kutta discontinuous Galerkin schemes

Consider

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

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

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

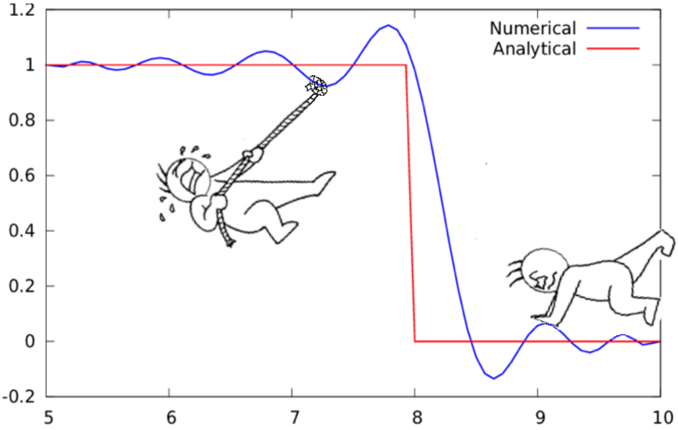
- basis $\{\phi_{ij}\}$ (Legendre Polynomials, etc)
- high-order quadrature (Gauss-Legendre, etc)
- numerical flux (Lax-Friedrich, etc)

$$\frac{\mathrm{d}U^{(i)}}{\mathrm{d}t} = R^{(i)}(U(t)) \quad \longrightarrow$$ Solve using SSP-RK3

Handling discontinuities

Non-linearity $\implies$ Discontinuities in finite time

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



Handling discontinuities

$$\text{Non-linearity} \implies \text{Discontinuities in finite time}$$
$$\implies \text{High-order methods suffer from Gibbs oscillations}$$

- Limiting
  - Cockburn et al. [JCP '89, Math. Comp. '89, Math. Comp. '90]
  - Qui and Shu [JCP '03, JCP '05]
- Streamline diffusion
  - Hughes et al. [Comp. Meth. Appl. Mech. Eng. '86]
  - Jaffre et al. [Math. Model. Meth. Appl. Sci. '95]
  - Hiltebrand et al. [Num. Math. '14]
- Shock capturing
  - Johnson et al. [Math. Comp. '90]
  - Persson et al. [AIAA '06]
  - Zingan et al. [Comp. Meth. Appl. Mech. Eng. '13]

Handling discontinuities

Non-linearity $\implies$ Discontinuities in finite time

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

- **Limiting**
  - Cockburn et al. [JCP '89, Math. Comp. '89, Math. Comp. '90]
  - Qui and Shu [JCP '03, JCP '05]
- Streamline diffusion
  - Hughes et al. [Comp. Meth. Appl. Mech. Eng. '86]
  - Jaffre et al. [Math. Model. Meth. Appl. Sci. '95]
  - Hiltebrand et al. [Num. Math. '14]
- Shock capturing
  - Johnson et al. [Math. Comp. '90]
  - Persson et al. [AIAA '06]
  - Zingan et al. [Comp. Meth. Appl. Mech. Eng. '13]

RKDG and limiting

RKDG solver

```
Initialize U[0]
n = 1
while t .lte. Tf do
    U[n] = U[n-1]
    for r = 1 to 3 do
        L = FindRHS(U[n])
        U[n] = RK_update(U[n-1], U[n], L, r)
        U[n] = Limit(U[n])
    end for
    n++, t+=dt
end while
```

RKDG and limiting

RKDG solver

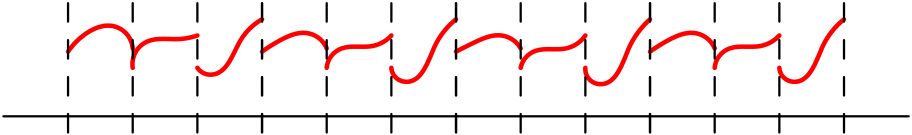
```
Initialize U[0]
n = 1
while t .lte. Tf do
    U[n] = U[n-1]
    for r = 1 to 3 do
        L = FindRHS(U[n])
        U[n] = RK_update(U[n-1], U[n], L, r)
        U[n] = Limit(U[n])
    end for
    n++, t+=dt
end while
```

Bottleneck step!!

RKDG and limiting

Strategy for limiting

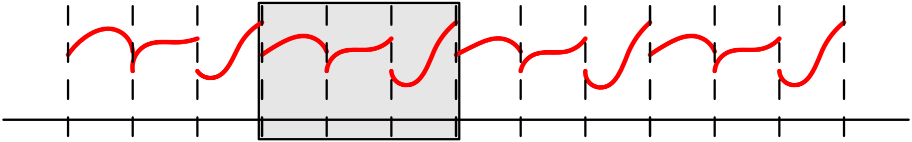
1. Identify troubled-cells
2. Limit solution in flagged cells



RKDG and limiting

Strategy for limiting

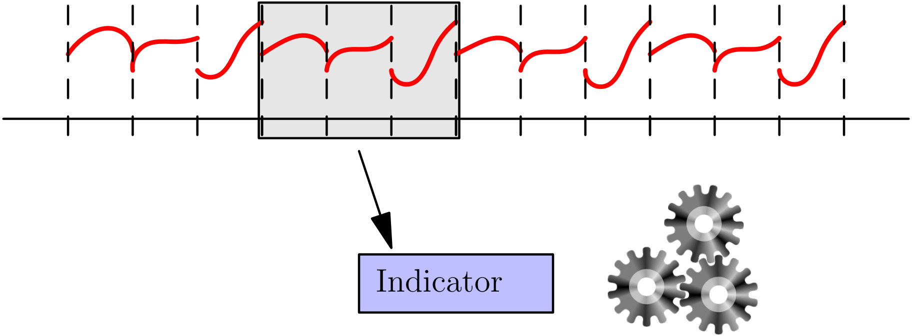
1. Identify troubled-cells
2. Limit solution in flagged cells



RKDG and limiting

Strategy for limiting

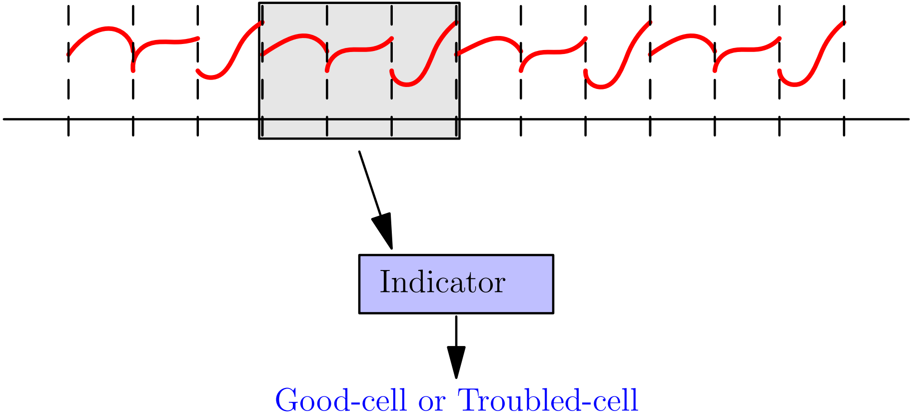
1. Identify troubled-cells
2. Limit solution in flagged cells



RKDG and limiting

Strategy for limiting

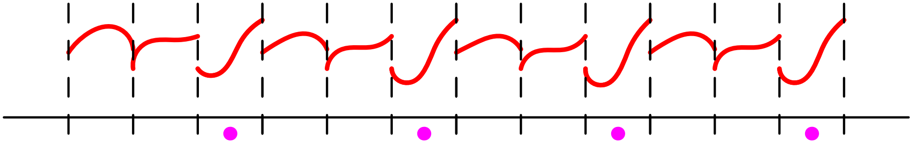
1. Identify troubled-cells
2. Limit solution in flagged cells



RKDG and limiting

Strategy for limiting

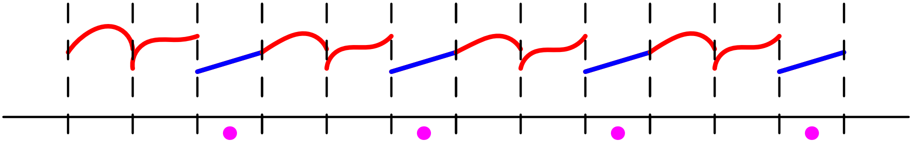
1. Identify troubled-cells
2. Limit solution in flagged cells



RKDG and limiting

Strategy for limiting

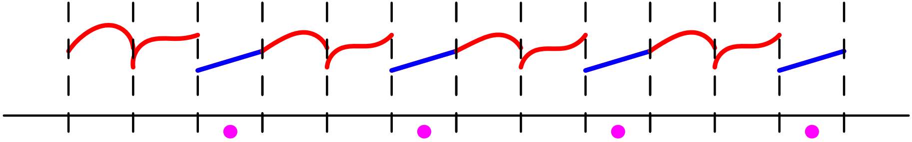
1. Identify troubled-cells
2. Limit solution in flagged cells



RKDG and limiting

Strategy for limiting

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



Some 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

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)

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)

Tested and compared by Qui and Shu (J. Sci. Comp. '05)

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)
- Outlier detection using Tukey’s boxplot method (Vuik and Ryan; J. Sci. Comp. ‘16)
- Polynomial degree based limiter (Fu and Shu; JCP ‘17)
- ...

Objective: Find a troubled-cell indicator which is:

- independent of problem-dependent parameters
- flags the necessary cells
- relatively inexpensive

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

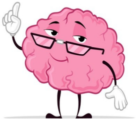
$$G : \mathbb{R}^n \mapsto \mathbb{R}^m$$

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)

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

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

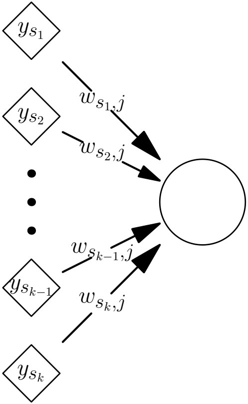
Artificial Neural Network (ANN)

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

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

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

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

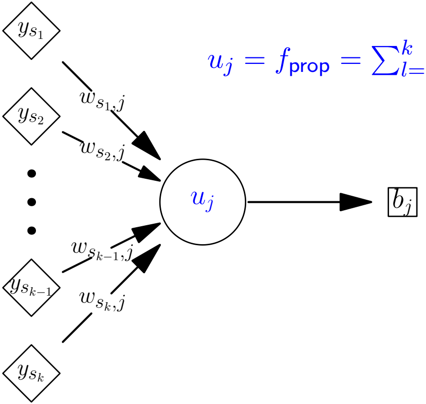


Artificial Neural Network (ANN)

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

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

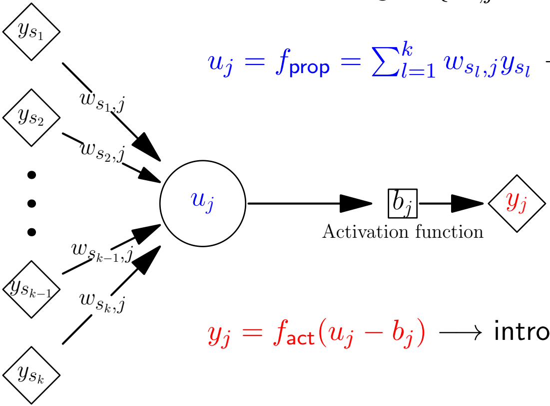
$u_j = f_{\text{prop}} = \sum_{l=1}^{k} w_{s_l,j} y_{s_l} \longrightarrow$ linear



Artificial Neural Network (ANN)

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

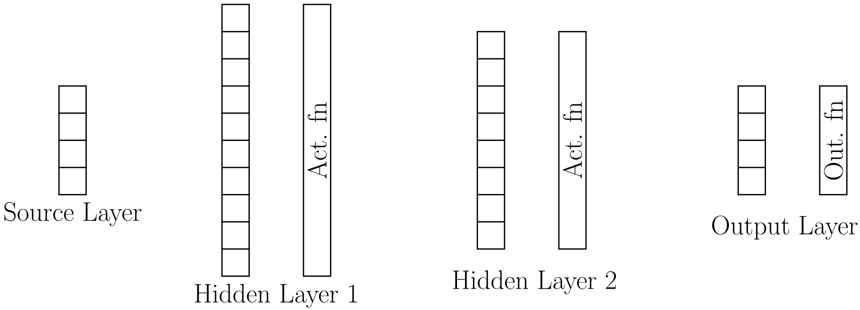
$$\mathcal{N} \longrightarrow \text{set of neurons}$$
$$\mathcal{V} \longrightarrow \text{set of connections} \quad \{(i,j) : 1 \leq i, j \leq |\mathcal{N}|\}$$
$$w : \mathcal{V} \mapsto \mathbb{R} \longrightarrow \text{connection weight} \quad \{w_{i,j} : 1 \leq i, j \leq |\mathcal{N}|\}$$



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

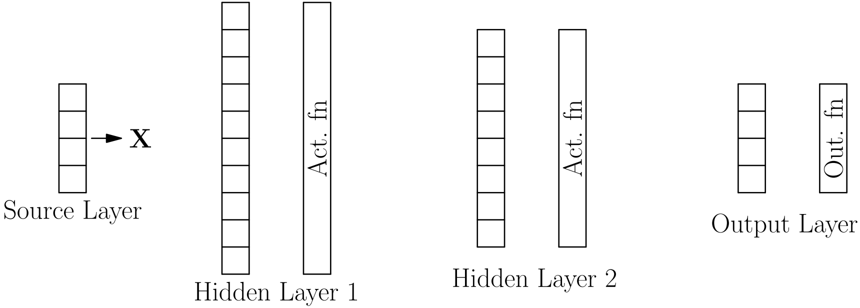
$y_j = f_{\text{act}}(u_j - b_j) \longrightarrow$ introduces non-linearity

Multilayer perceptron (MLP)



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

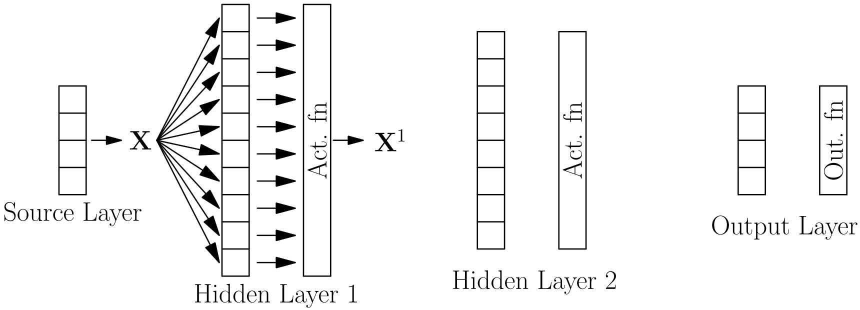
Multilayer perceptron (MLP)



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

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

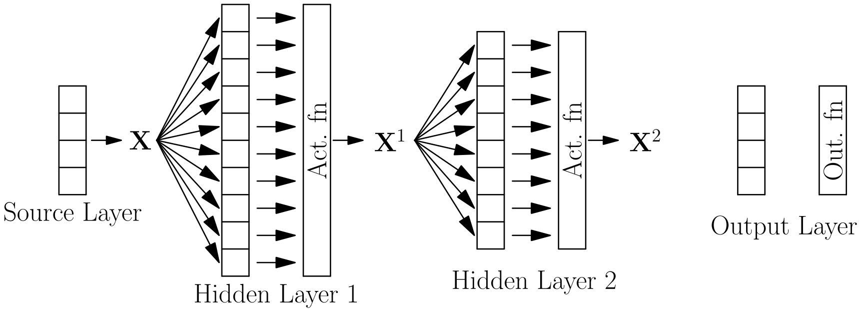
Multilayer perceptron (MLP)



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)

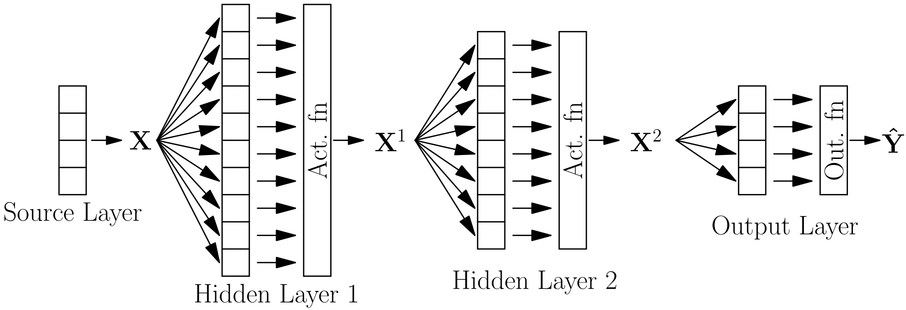


n_1 neurons in Hidden layer 1, n_2 neurons in Hidden layer 2

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

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

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

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

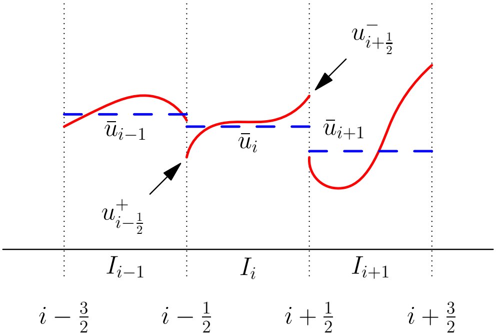
and a cost functional $C(\mathbf{Y}, \hat{\mathbf{Y}})$. Find the weights W and biases b of the MLP which minimize C.

Remarks:

- The network is trained offline using given data
- Optimize using gradient descent, Adams, etc.
- Capacity to generalize

An MLP-based indicator

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



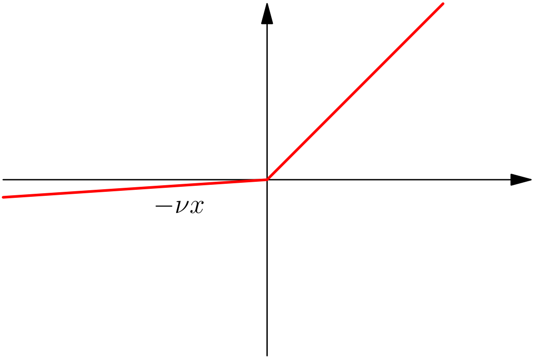
This data is also used by the TVB indicator!!

An MLP-based indicator

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

An MLP-based indicator

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



An MLP-based indicator

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

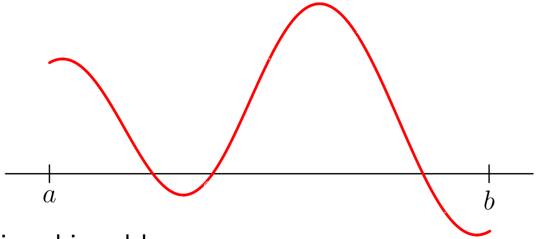
- Input $\mathbf{X} = [\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u^+_{i-\frac{1}{2}}, u^-_{i+\frac{1}{2}}] \in \mathbb{R}^5$ (with scaling)
- 5 Hidden Layers with width 256, 128, 64, 32, 16
- Leaky ReLU activation function with $\nu = 10^{-3}$
- Softmax output function

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

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

$$C = -\sum_{i=1}^{N} \left[Y_i^{(0)} \log\left(\hat{Y}_i^{(0)}\right) + Y_i^{(1)} \log\left(\hat{Y}_i^{(1)}\right) \right]$$

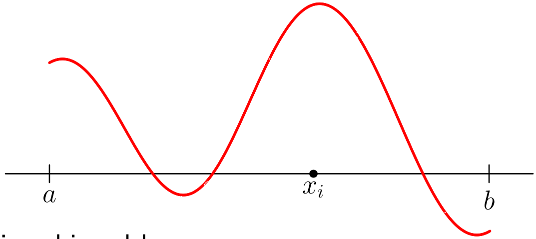
Generating the training data



Data sampling is achieved by

- Choose a known function $u(x)$

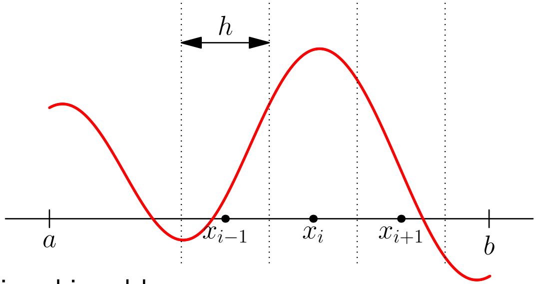
Generating the training data



Data sampling is achieved by

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

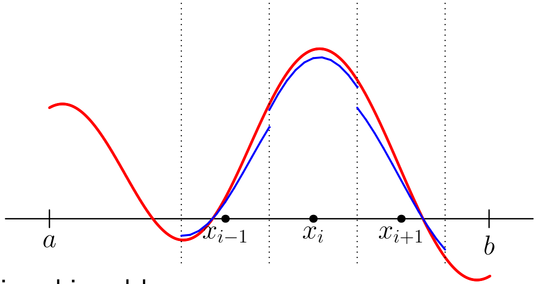
Generating the training data



Data sampling is achieved by

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

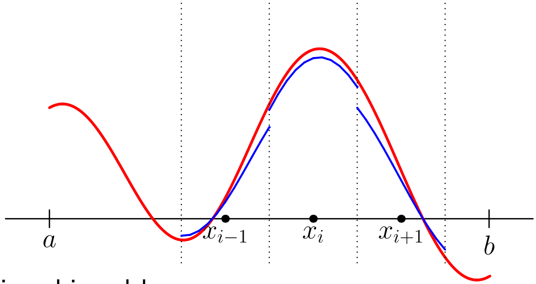
Generating the training data



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

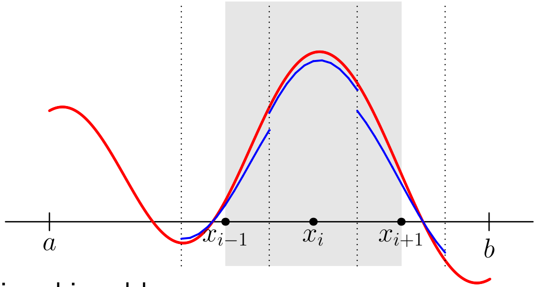
Generating the training data



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)

So how well does the trained MLP really work?

Numerical setup

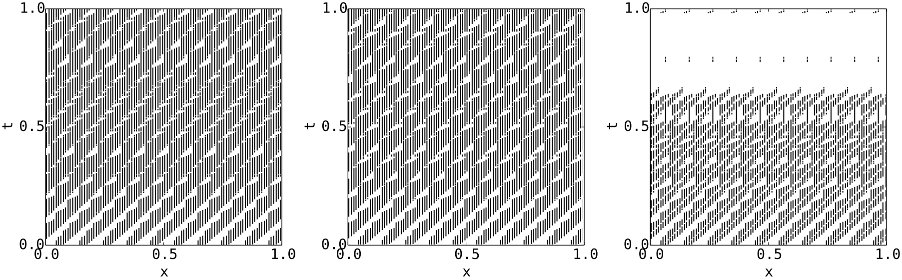
- Comparison with TVB limiter by setting parameter M

$$\begin{aligned} \text{minmod} &\longrightarrow M = 0 \quad \text{(flags smooth extrema)} \\ \text{TVB-1} &\longrightarrow M = 10 \\ \text{TVB-2} &\longrightarrow M = 100 \\ \text{TVB-3} &\longrightarrow M = 1000 \end{aligned}$$

- In flagged cells, perform limited linear reconstruction with MUSCL limiter
- Legendre basis with degree $r = 4$
- Local Lax-Friedrich numerical flux
- Time integration with SSP-RK3

Linear advection: $u_t + u_x = 0$

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

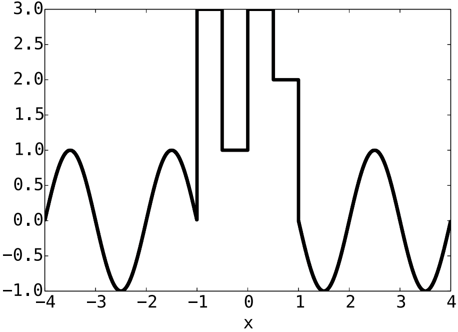


minmod **TVB-1** **TVB-2**

MLP and TVB-3 do not flag any cell!!

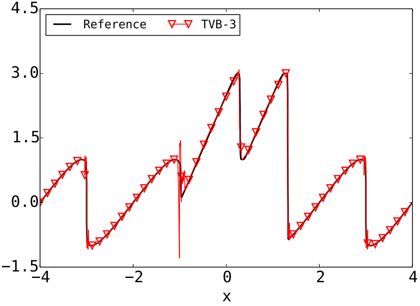
Burgers equation: $u_t + (u^2/2)_x = 0$

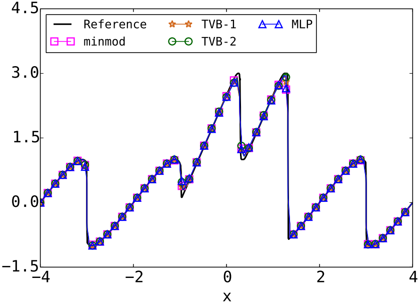
$$x \in [-4, 4], \quad T_f = 0.4, \quad N = 200$$



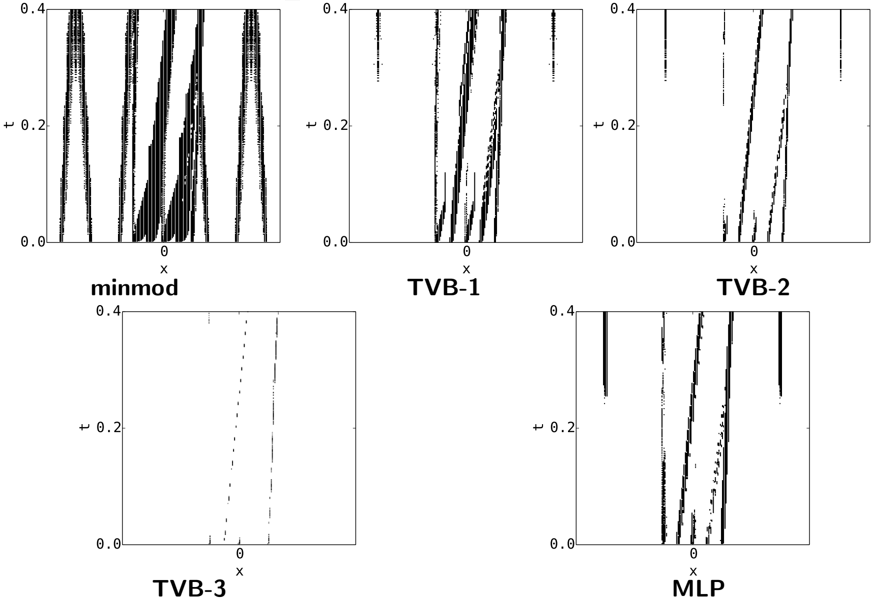
Burgers equation: $u_t + (u^2/2)_x = 0$

$$x \in [-4, 4], \quad T_f = 0.4, \quad N = 200$$





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



Shallow water equations: Dam break

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

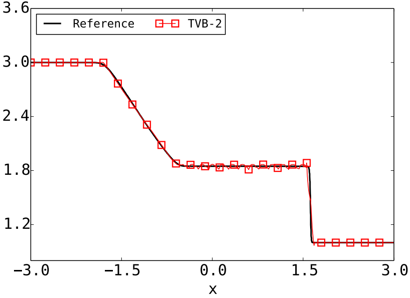
$$D_0(x) = \begin{cases} 3 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \end{cases}, \qquad u_0(x) = 0, \qquad g = 1,$$

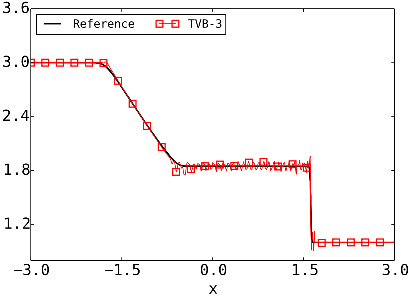
$$T_f = 1, \quad N = 100$$

Indicator variables $\longrightarrow$ $(D,\ u)$

Limiting variables $\longrightarrow$ local characteristic variables

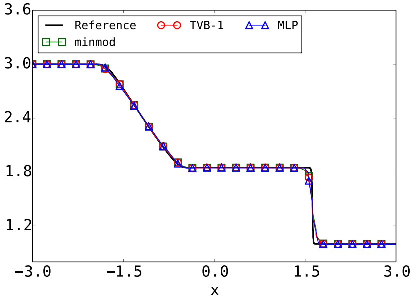
Shallow water equations: Dam break



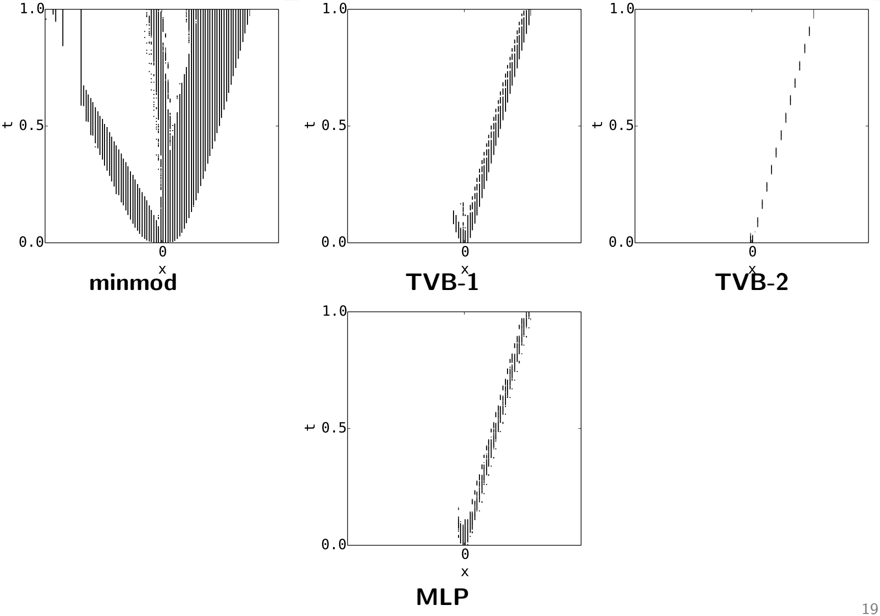


No cells flagged by TVB-3!!

Shallow water equations: Dam break



Shallow water equations: Dam break



Euler equations

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

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

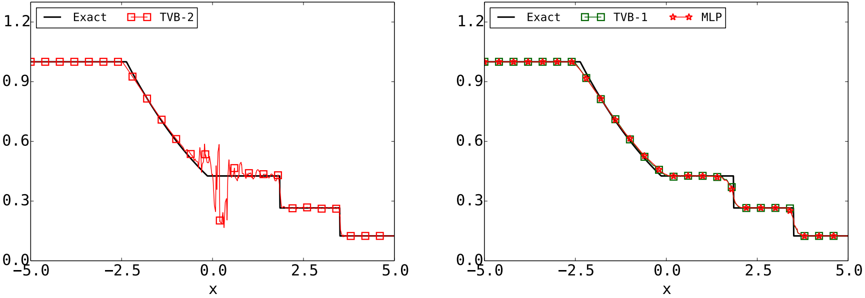
Indicator variables $\longrightarrow$ $(\rho,\ u,\ p)$

Limiting variables $\longrightarrow$ local characteristic variables

Euler equations: Sod shock tube

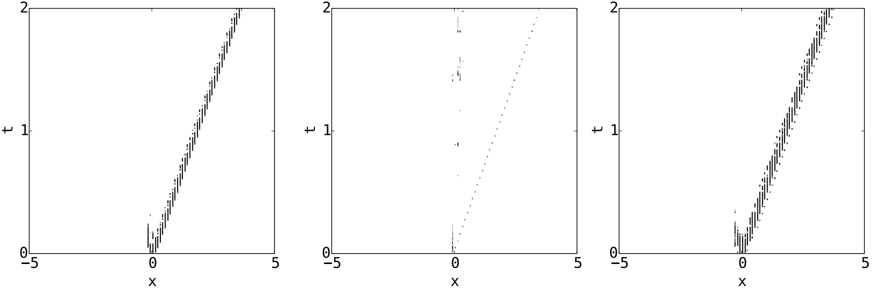
$$(\rho,\ u,\ p) = \begin{cases} (1,\ 0,\ 1) & \text{if } x < 0 \\ (0.125,\ 0,\ 0.1) & \text{if } x > 0 \end{cases}, \qquad x \in [-1, 1]$$

$$T_f = 2, \quad N = 100$$



Loss of positivity with TVB-3!!

Euler equations: Sod shock tube

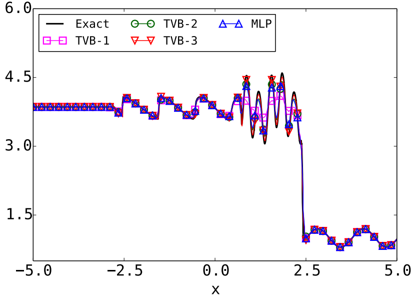


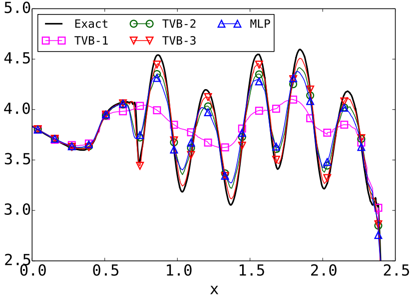
TVB-1 TVB-2 MLP

Euler equations: Shock-entropy problem

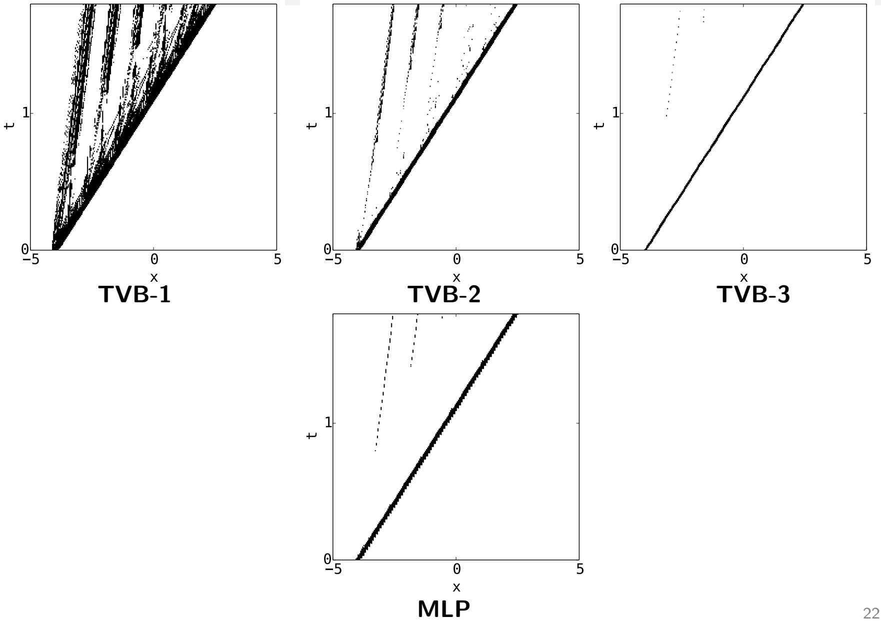
$$(\rho,\ u,\ p) = \begin{cases} (3.857143,\ 2.629369,\ 10.33333) & \text{if } x < -4 \\ (1 + 0.2\sin(5x),\ 0,\ 1) & \text{if } x > -4 \end{cases}$$

$$T_f = 1.8, \quad N = 256$$





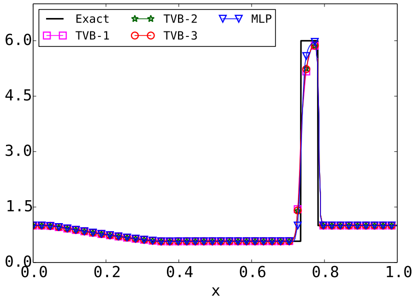
Euler equations: Shock-entropy problem

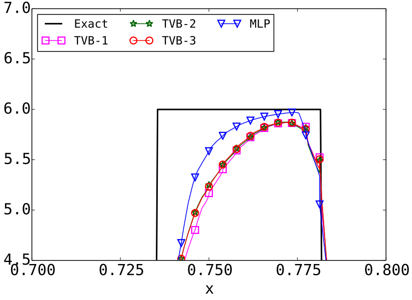


Euler equations: Left half of blast-wave

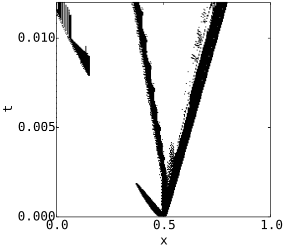
$$(\rho,\ u,\ p) = \begin{cases} (1,\ 0,\ 1000) & \text{if } x < 0.5 \\ (1,\ 0,\ 0.01) & \text{if } x > 0.5 \end{cases}, \qquad x \in [0,1],$$

$$T_f = 0.012, \quad N = 256$$

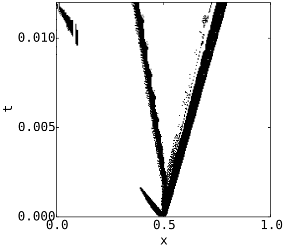




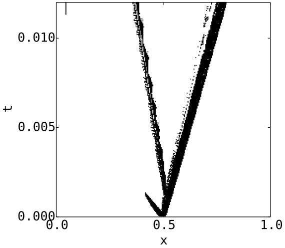
Euler equations: Left half of blast-wave



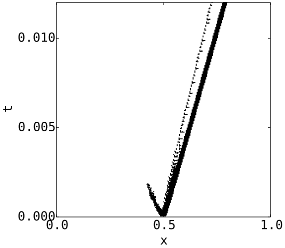
TVB-1



TVB-2



TVB-3



MLP

Conclusion

- Constructed a MLP-based indicator independent of problem-dependent parameters
- Works for 1D scalar and systems of conservation laws
- Flags the necessary number of cells

Conclusion

- Constructed a MLP-based indicator independent of problem-dependent parameters
- Works for 1D scalar and systems of conservation laws
- Flags the necessary number of cells

What next?

- Experiments with depth and width of the network
- Other forms of inputs for training
- Unstructured 2D and 3D grids

Conclusion

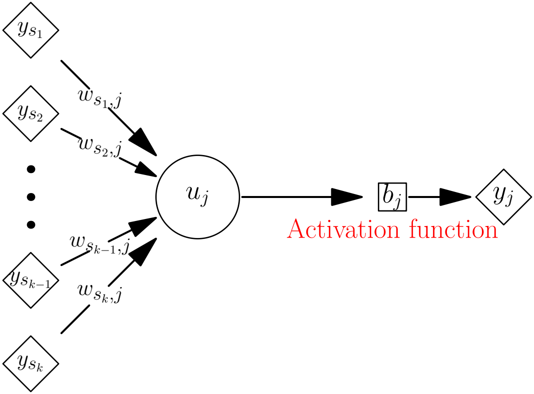
- Constructed a MLP-based indicator independent of problem-dependent parameters
- Works for 1D scalar and systems of conservation laws
- Flags the necessary number of cells

What next?

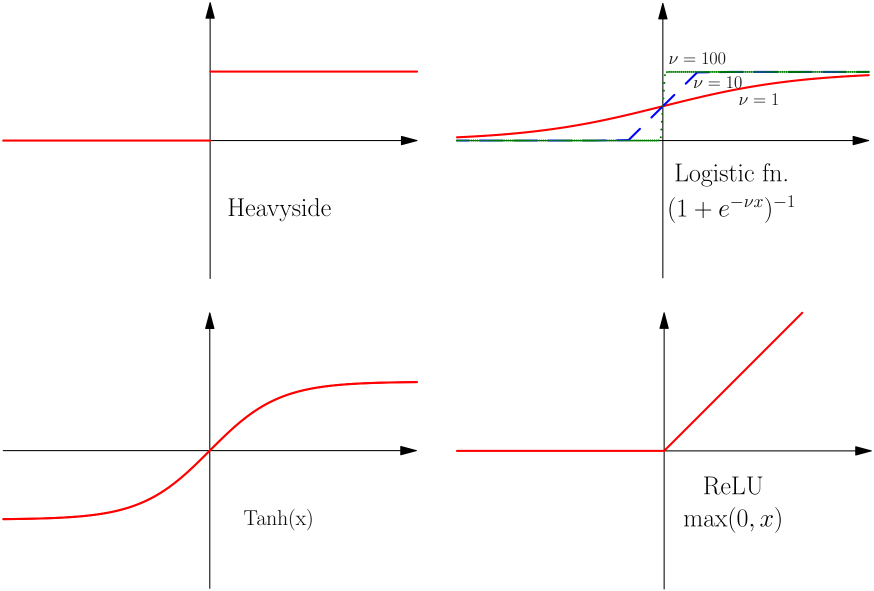
- Experiments with depth and width of the network
- Other forms of inputs for training
- Unstructured 2D and 3D grids

Questions?

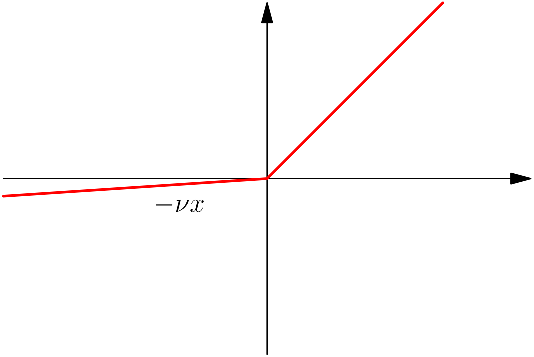
Activation functions



Activation functions

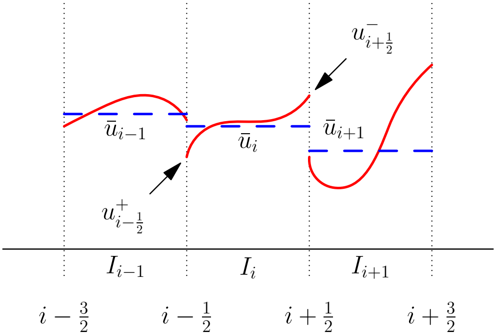


Activation functions



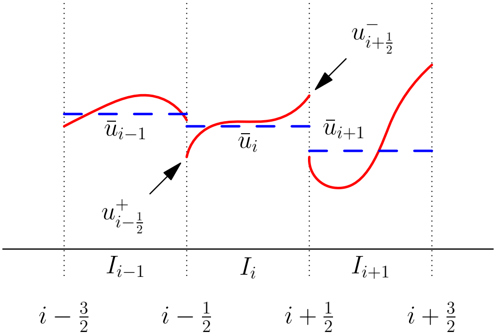
TVB Limiter (Qiu and Shu, 2005)

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



TVB Limiter (Qiu and Shu, 2005)

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



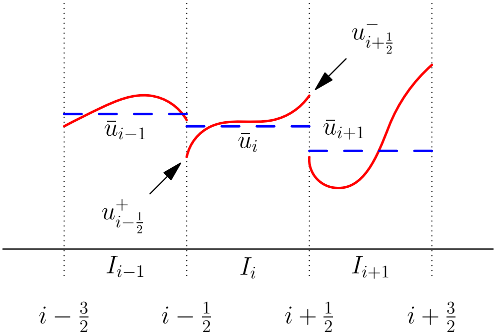
Evaluate 4 differences

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

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

TVB Limiter (Qiu and Shu, 2005)

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

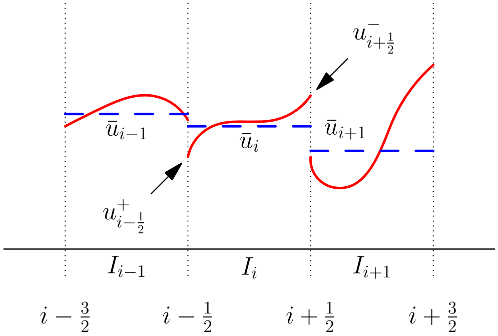


Modify interface values

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

TVB Limiter (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:

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

Disadvantage: Flags cell with smooth extrema

Search for the elusive M

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

- Minmod limiter:

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

  Disadvantage: Flags cell with smooth extrema

- TVB limiter: Depends on h and tunable parameter M

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

  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

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

Application of ANNs

- Computer vision
- Speech recognition

Application of ANNs

- Computer vision
- Speech recognition
- Solving ODEs and PDEs (Lagaris et al. '98, Golak '10)
- Poisson solver (Yang et al. '16, Tompson et al. '17)
- Physics Informed Deep Learning (Karniadakis '17)

Application of ANNs

- Computer vision
- Speech recognition
- Solving ODEs and PDEs (Lagaris et al. '98, Golak '10)
- Poisson solver (Yang et al. '16, Tompson et al. '17)
- Physics Informed Deep Learning (Karniadakis '17)

Theoretical results

- Can approximate any continuous function (Cybenko '89)
- Funahashi ('89)
- Chen et al. ('92)
- Costarelli et al. ('13)
- Guliyev et al. ('16)

Application of ANNs

- Computer vision
- Speech recognition
- Solving ODEs and PDEs (Lagaris et al. ‘98, Golak ‘10)
- Poisson solver (Yang et al. ‘16, Tompson et al. ‘17)
- Physics Informed Deep Learning (Karniadakis ‘17)

Theoretical results

- Can approximate any continuous function (Cybenko ‘89)
- Funahashi (‘89)
- Chen et al. (‘92)
- Costarelli et al. (‘13)
- Guliyev et al. (‘16)

Rigorous results for general networks not available!