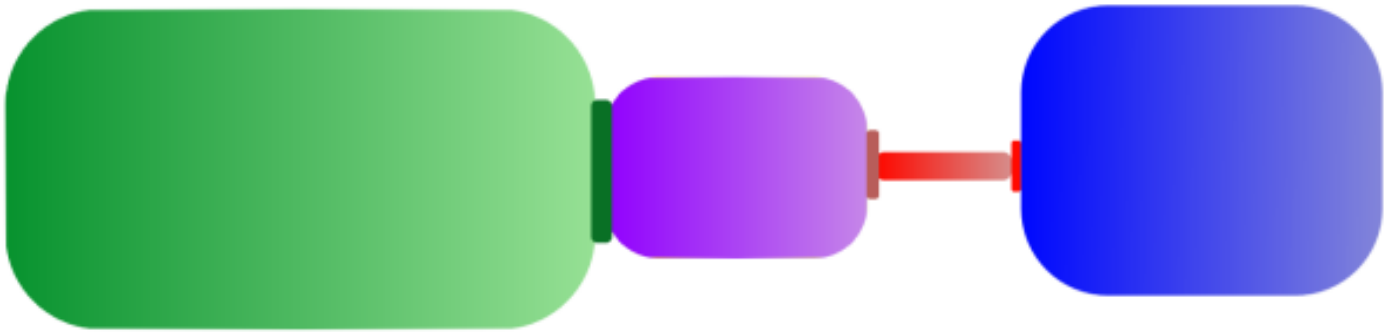
Detecting discontinuities using deep learning

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

12 April, 2019

Motivation

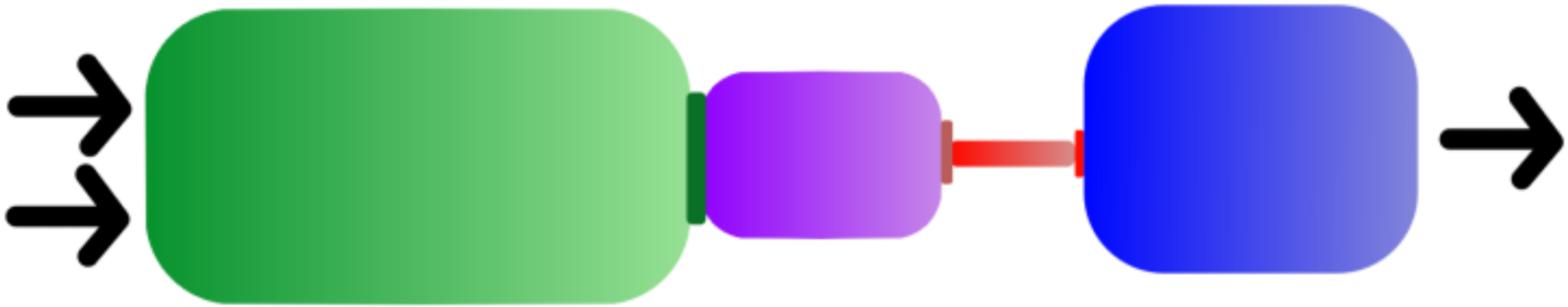
Good numerical methods developed over the past several decades



Motivation

Good numerical methods developed over the past several decades

Gives satisfactory results, but ...

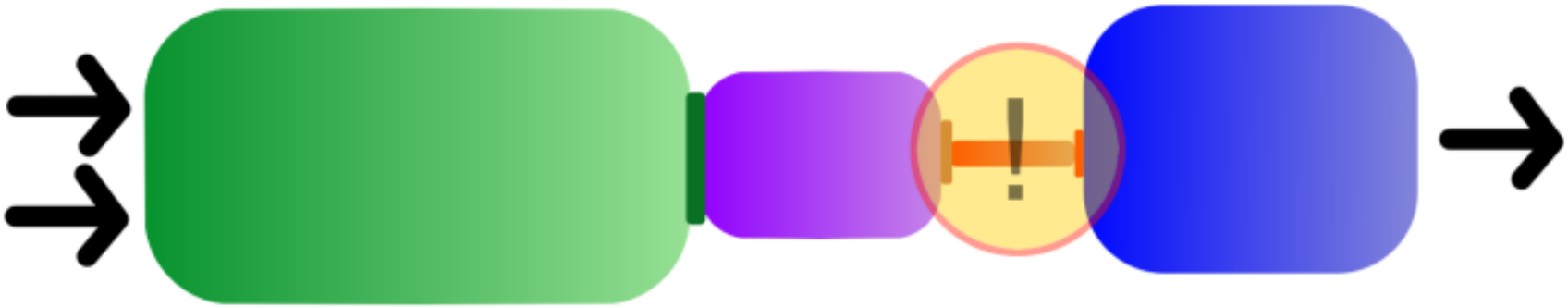


Motivation

Good numerical methods developed over the past several decades

Gives satisfactory results, but ...

Bottlenecks exist – computationally expensive subcomponents, problem dependent parameters, etc

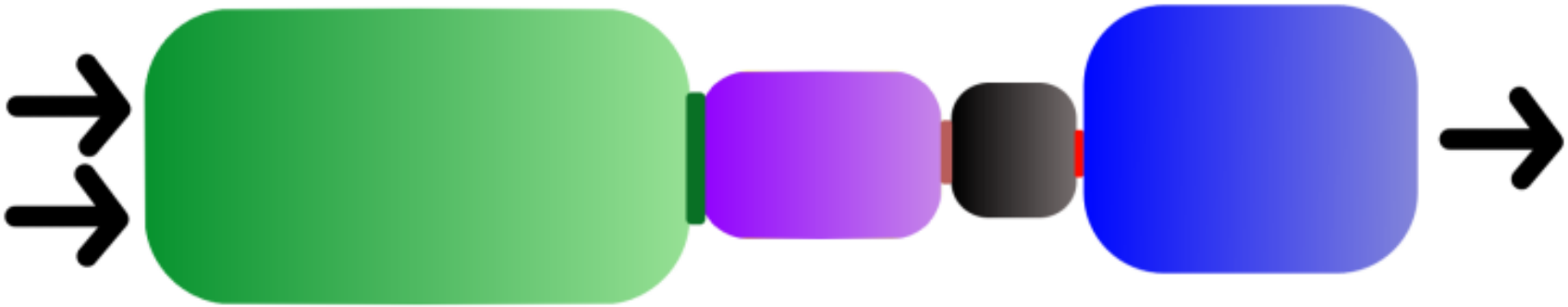


Motivation

Good numerical methods developed over the past several decades

Gives satisfactory results, but ...

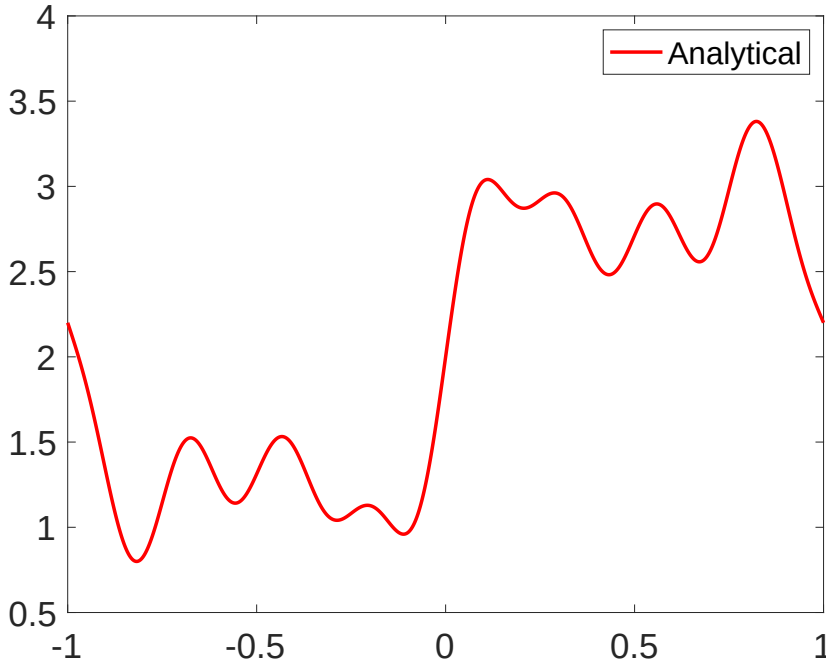
Bottlenecks exist – computationally expensive subcomponents, problem dependent parameters, etc

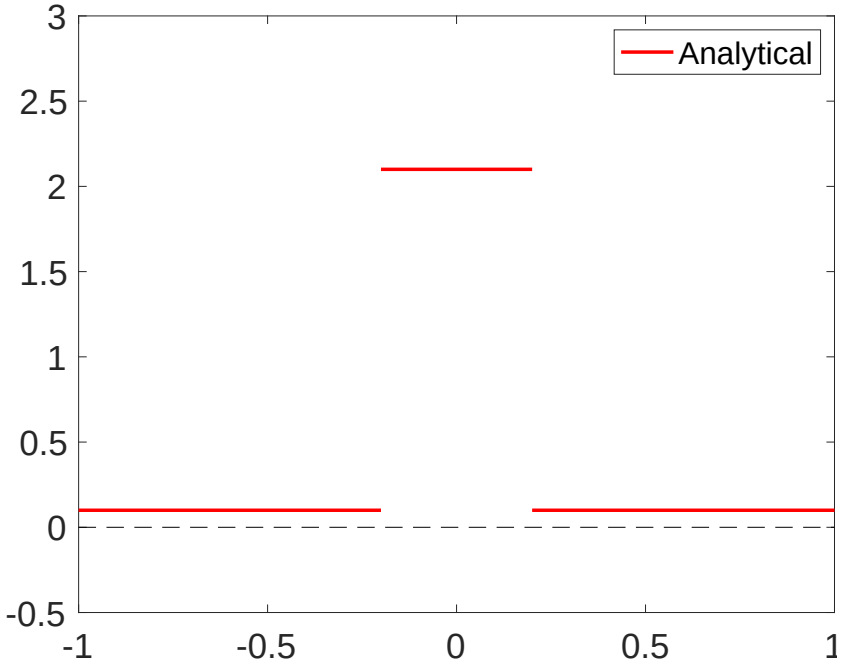


Startegy: Replace (only) the problematic part with deep learning black box

Controlling spurious oscillations

Smooth vs discontinuous





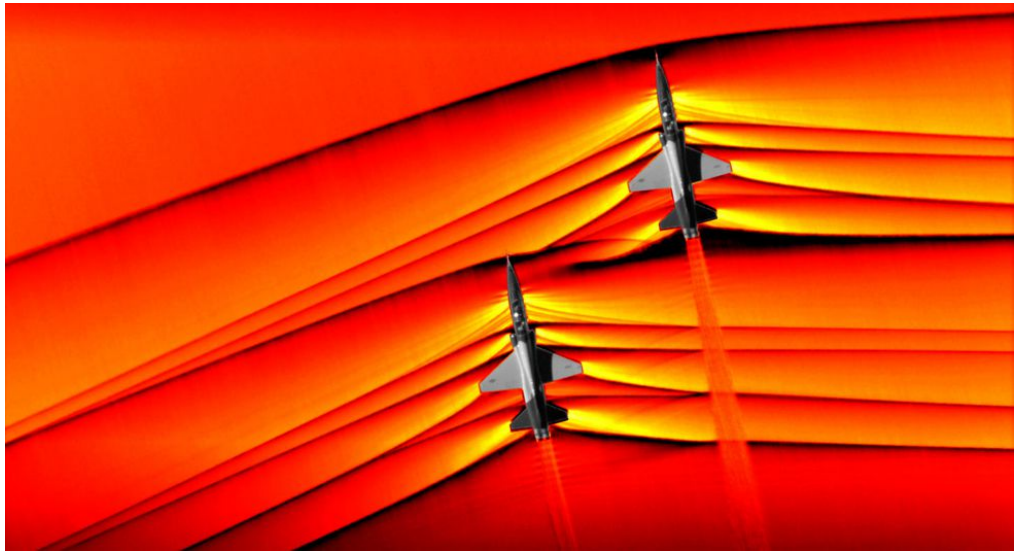
Discontinuities in real life

Discontinuities in real life

(courtesy Nicole Honeywill)

Discontinuities in real life

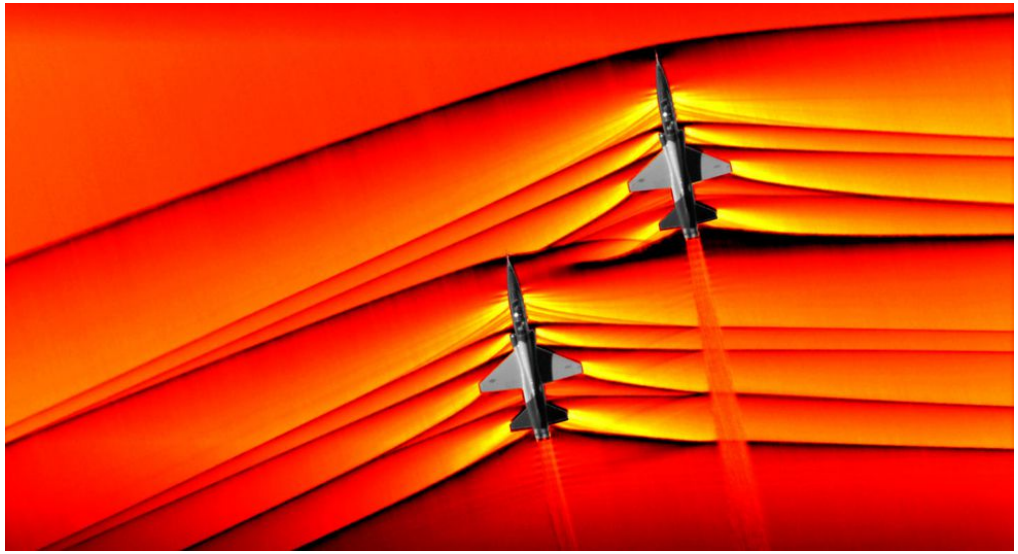
(courtesy Nicole Honeywill)



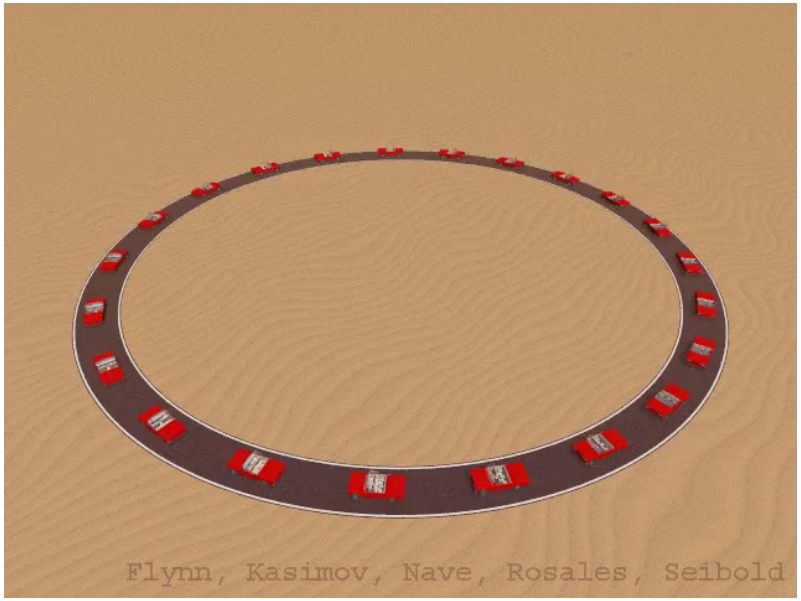
(courtesy NASA)

Discontinuities in real life

(courtesy Nicole Honeywill)

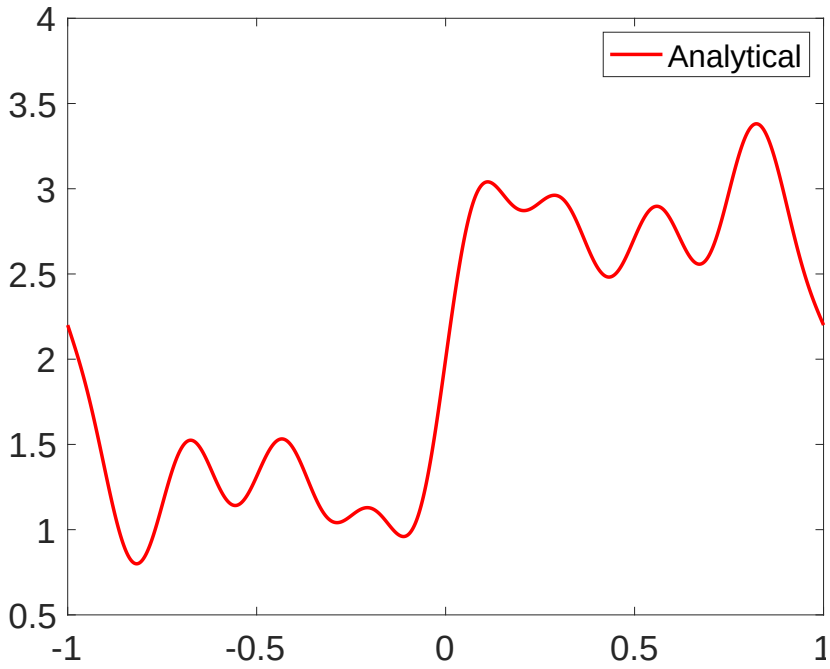


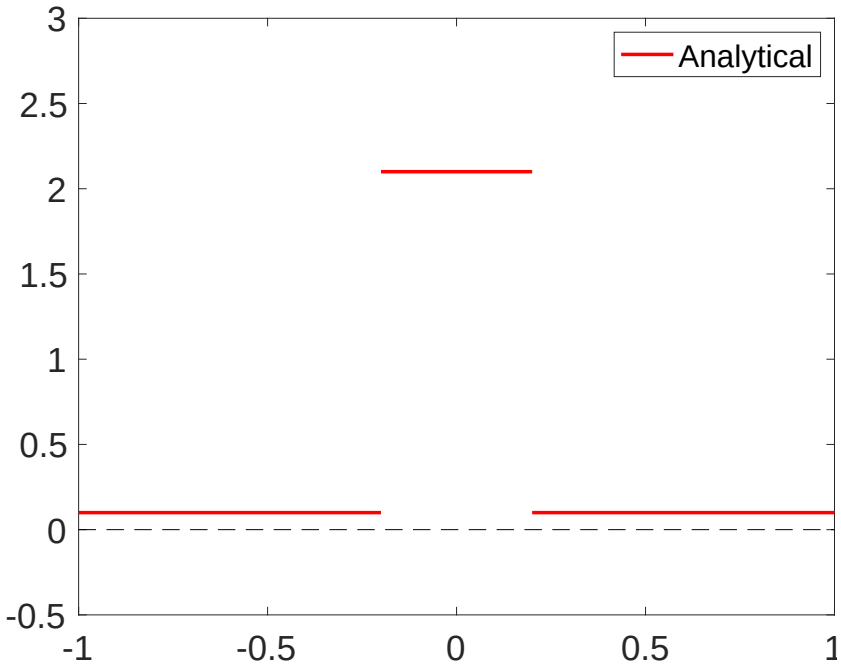
(courtesy NASA)



(Flynn et al.)

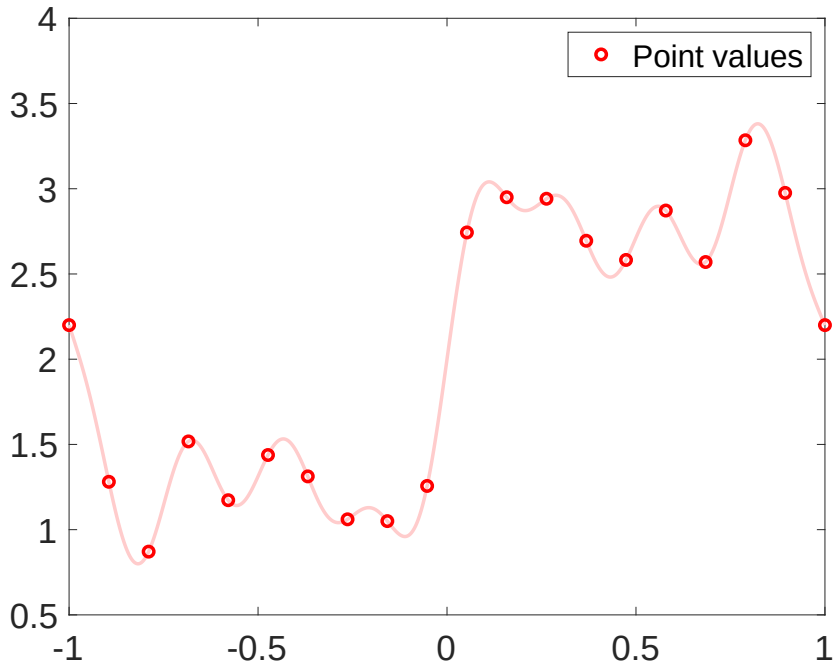
Approximating functions

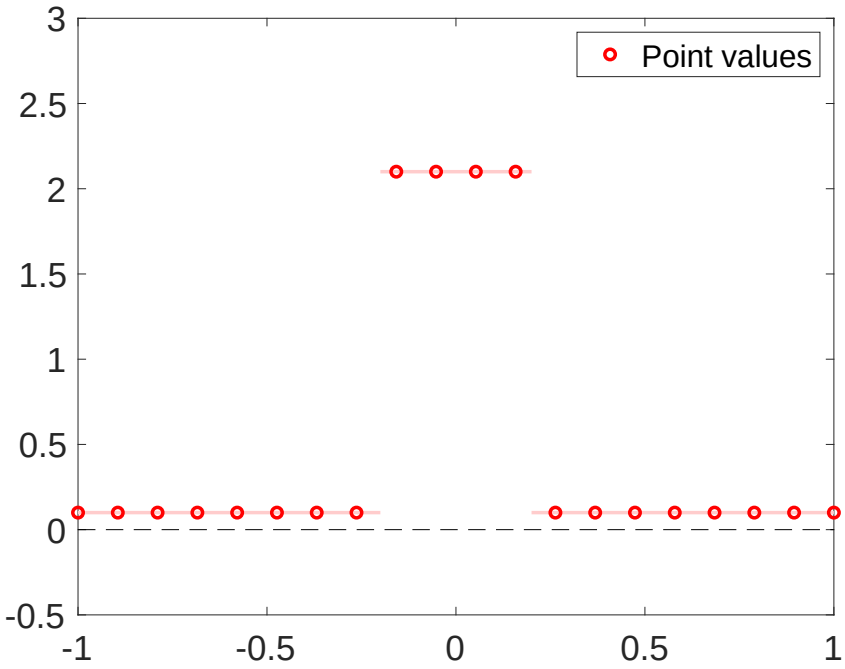




Approximating functions

In practice, function known only at select points

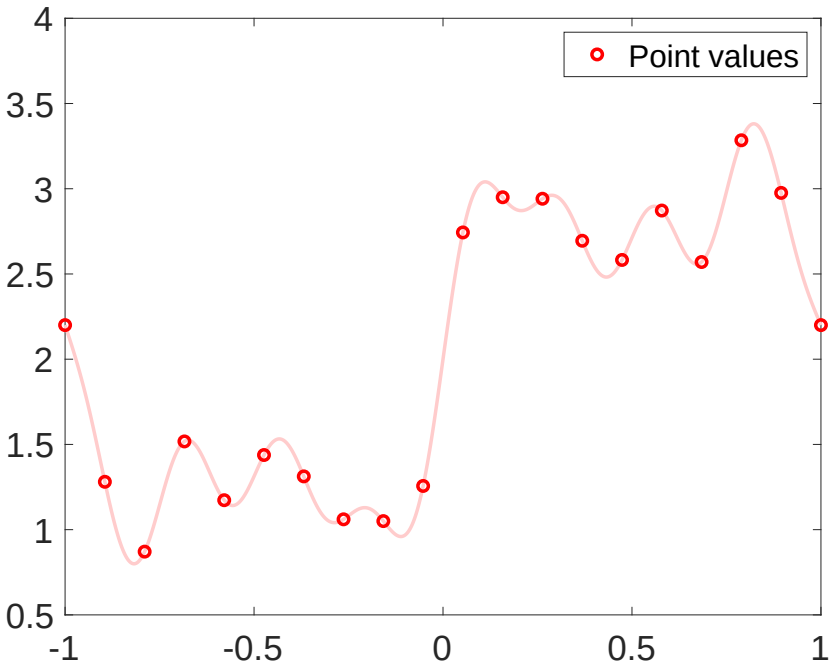


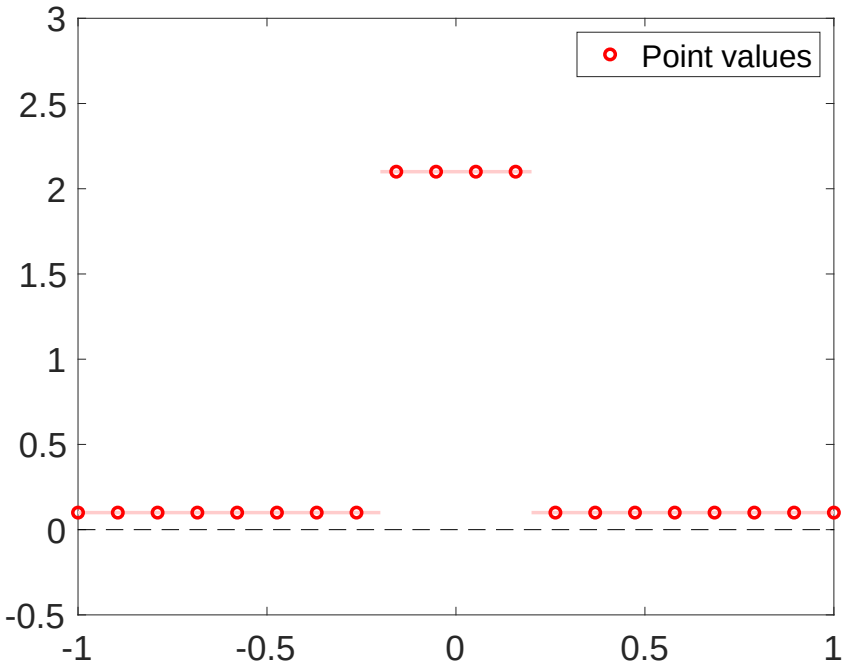


Approximating functions

In practice, function known only at select points
Approximating the function using a smooth basis

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



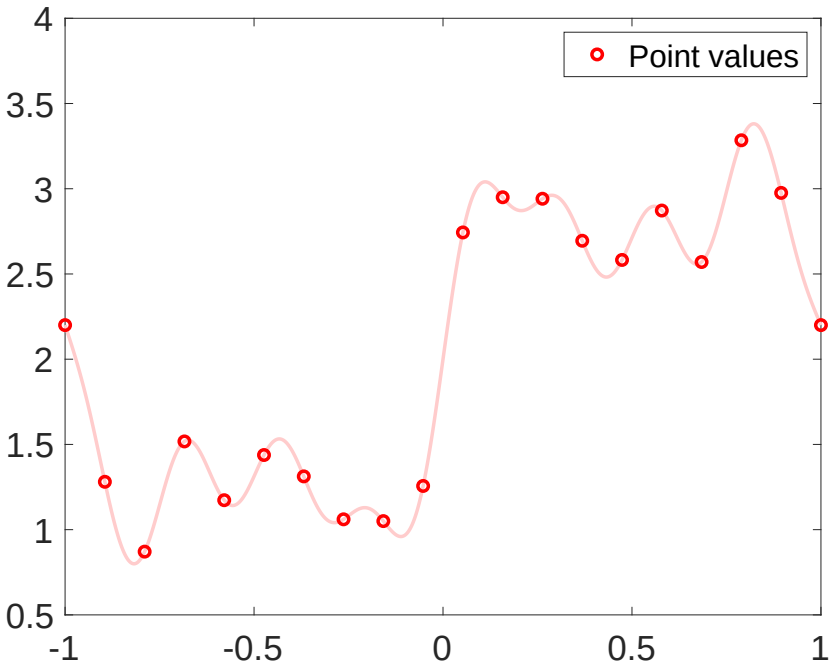


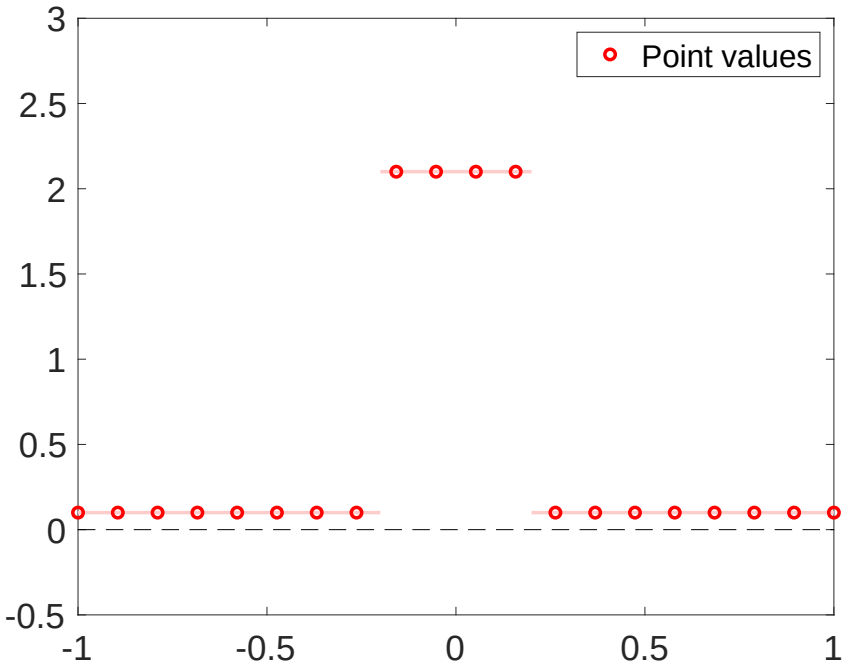
Approximating functions

In practice, function known only at select points
Approximating the function using a smooth basis

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

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





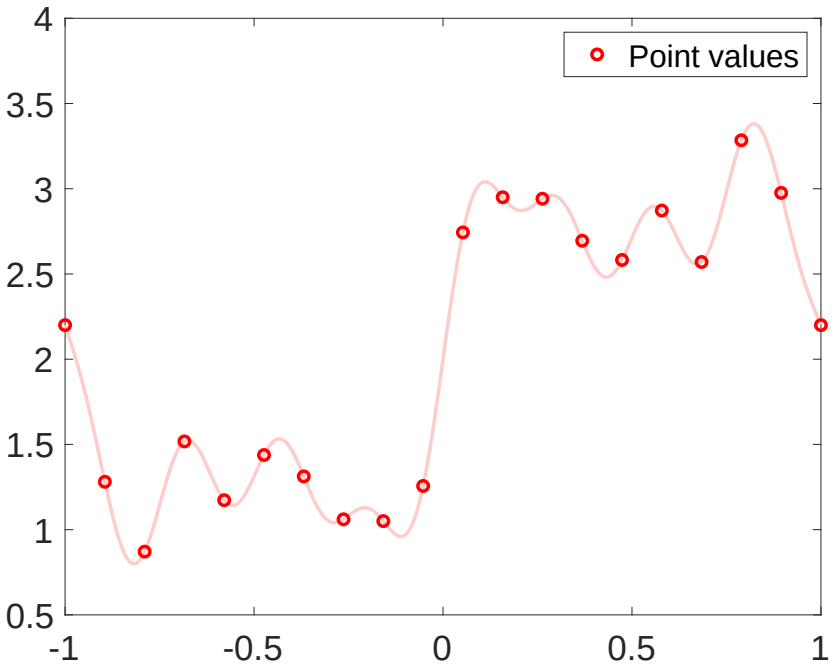
Approximating functions

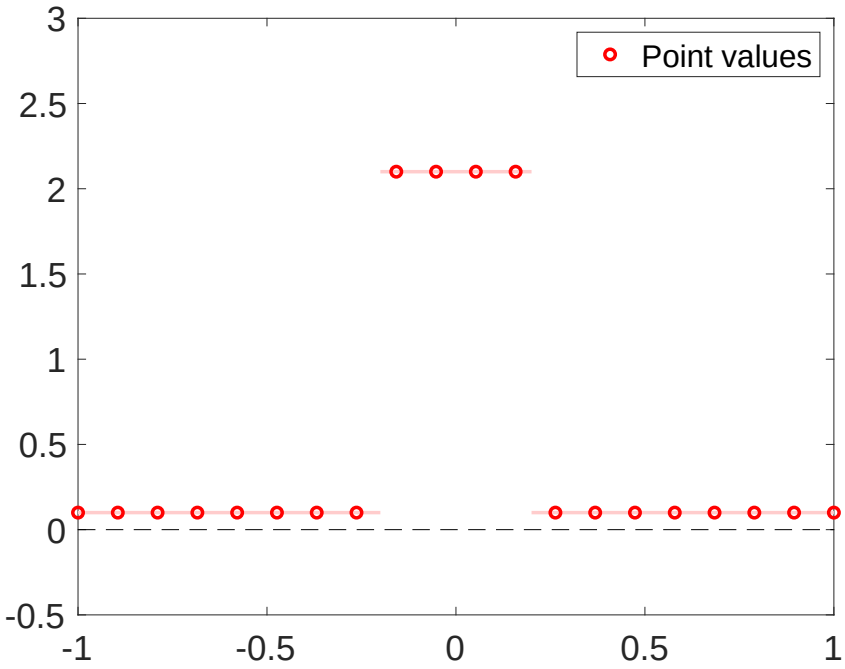
In practice, function known only at select points
Approximating the function using a smooth basis

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

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

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





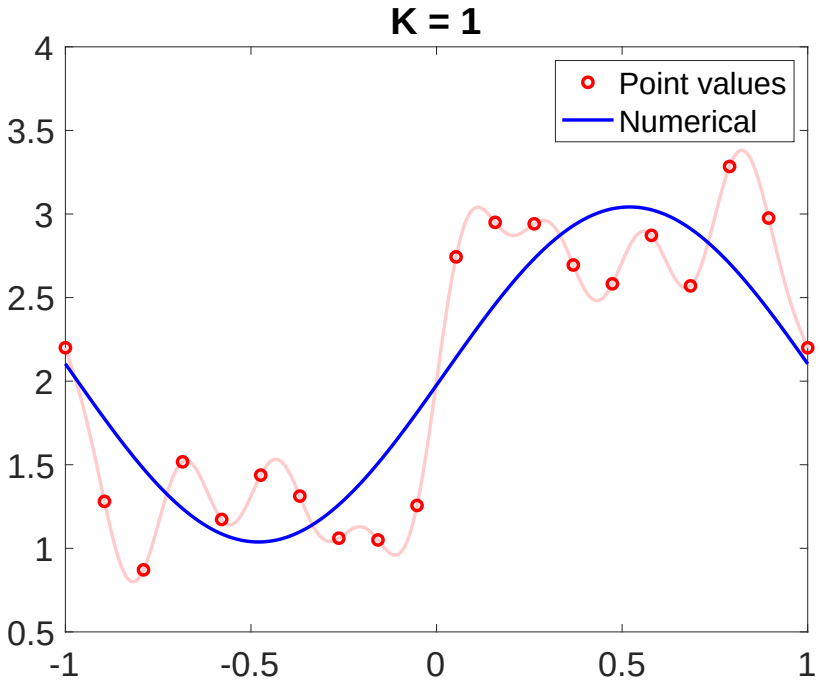
Approximating functions

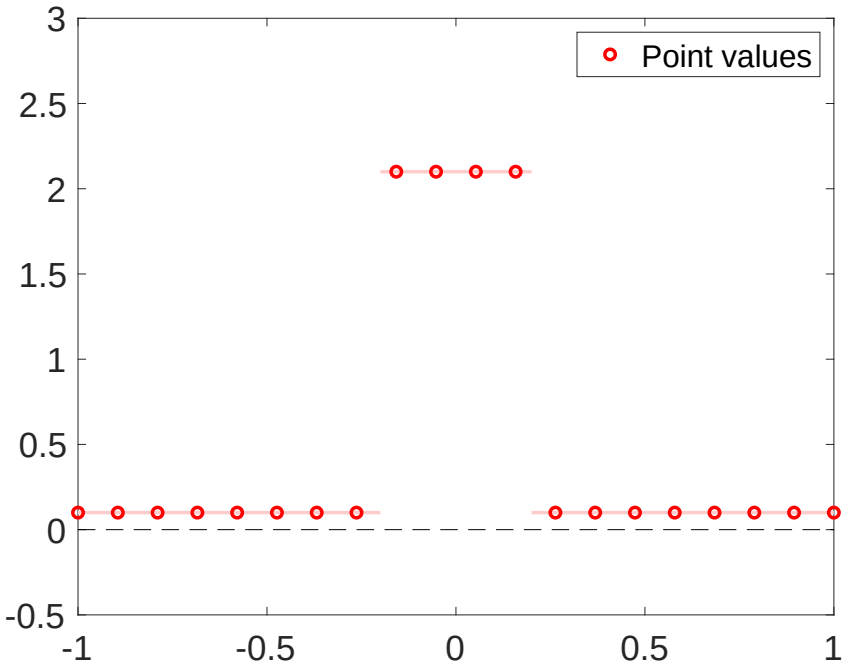
In practice, function known only at select points
Approximating the function using a smooth basis

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

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

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





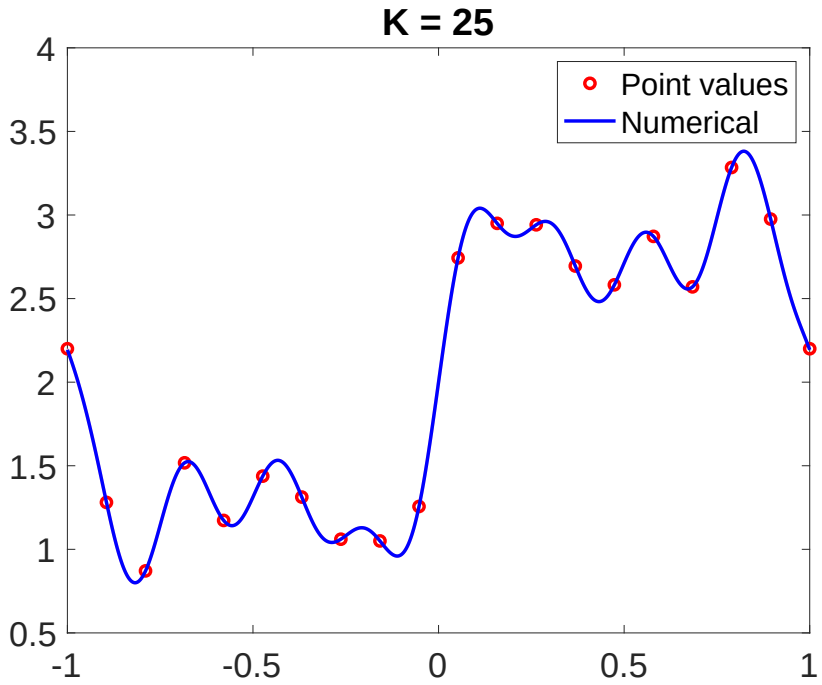
Approximating functions

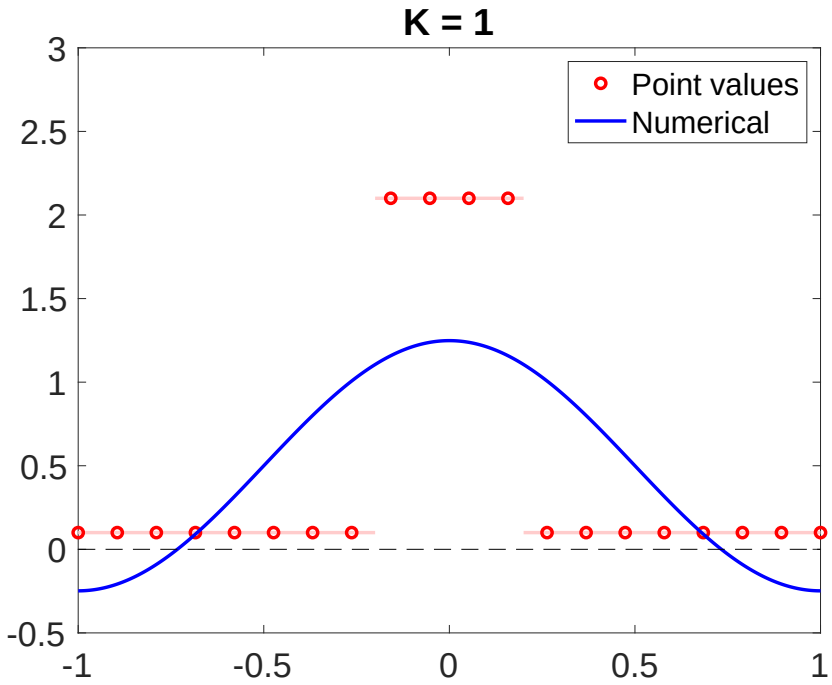
In practice, function known only at select points
Approximating the function using a smooth basis

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

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

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





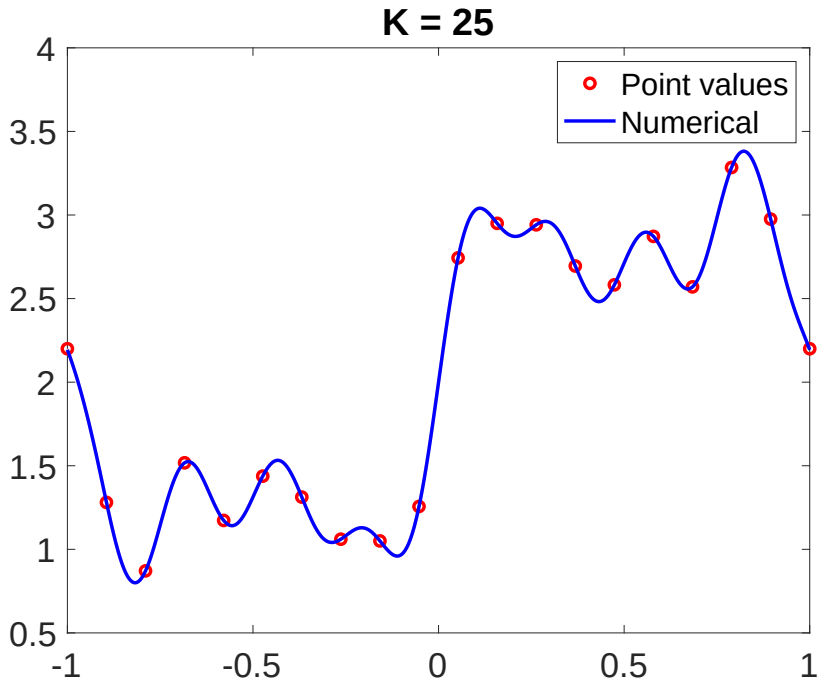
Approximating functions

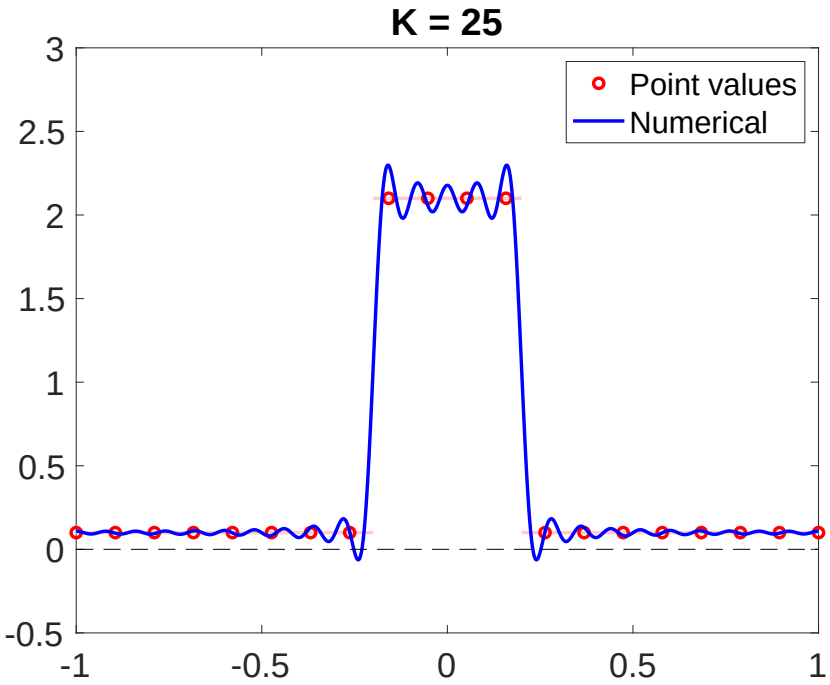
In practice, function known only at select points
Approximating the function using a smooth basis

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

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

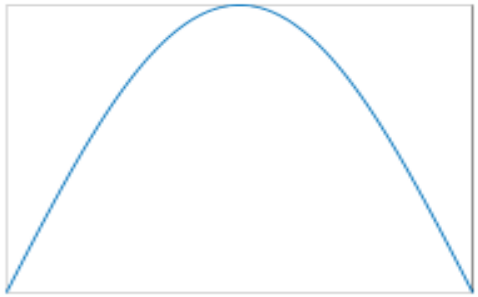
$\sin\left(\frac{\pi p}{K}\right), \cos\left(\frac{\pi p}{K}\right)$

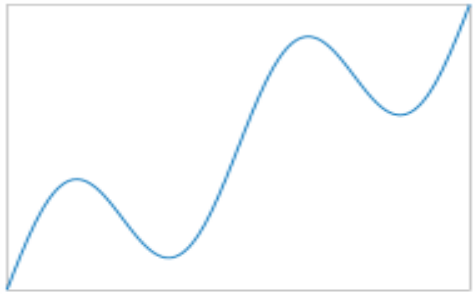


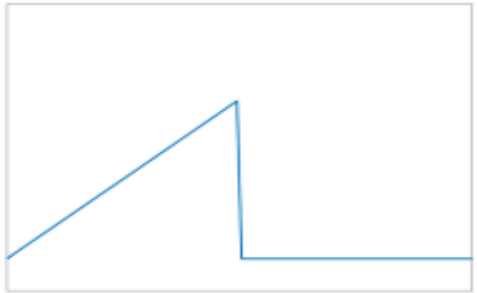


Unphysical/spurious oscillations $\Longrightarrow$ loss of positivity!

Approximating functions







Conservation laws

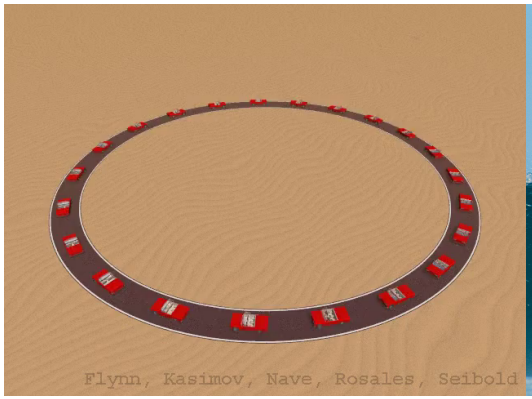
Consider the partial differential equation

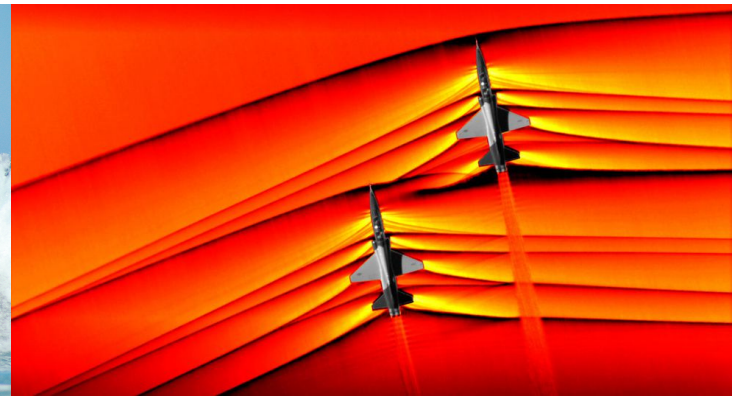
$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}(\mathbf{u})}{\partial x} = 0$$
$$\mathbf{u}(x, 0) = \mathbf{u}_0(x)$$

Conservation laws

Consider the partial differential equation

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

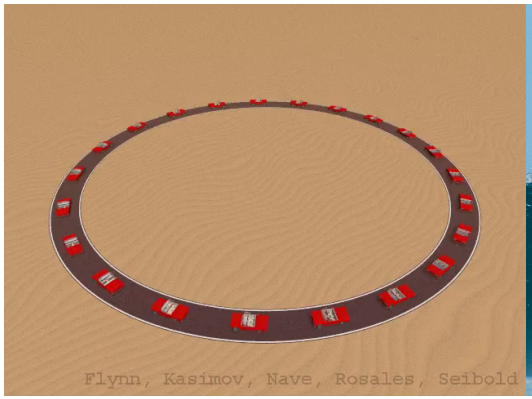


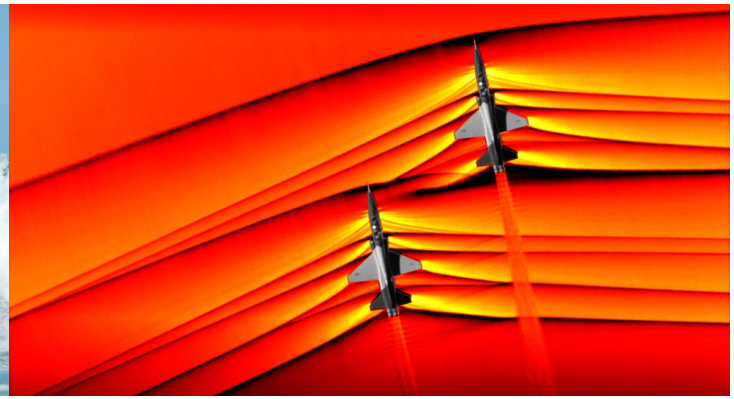



Conservation laws

Consider the partial differential equation

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



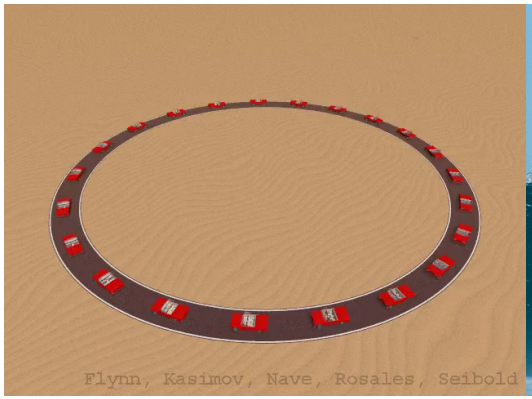


Lighthill-Whitham-Richards model

Conservation laws

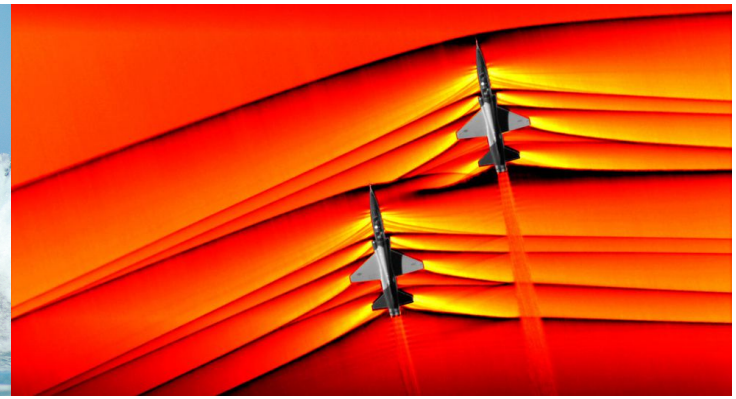
Consider the partial differential equation

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



Lighthill-Whitham-Richards model

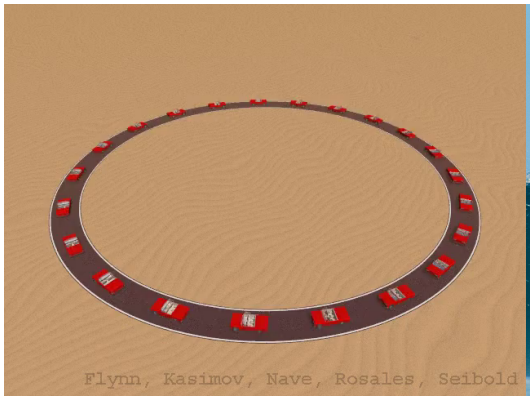
Shallow water equations



Conservation laws

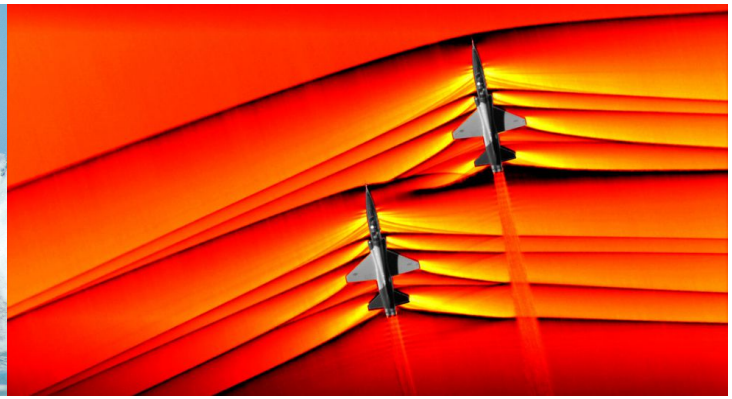
Consider the partial differential equation

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



Lighthill-Whitham-Richards model

Shallow water equations



Euler equations

Conservation laws

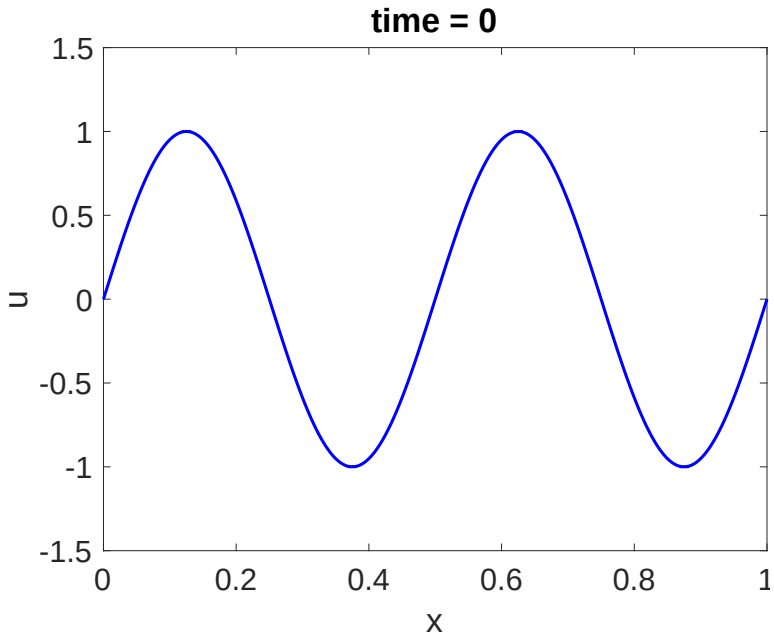
Consider the partial differential equation

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

Non-linearity

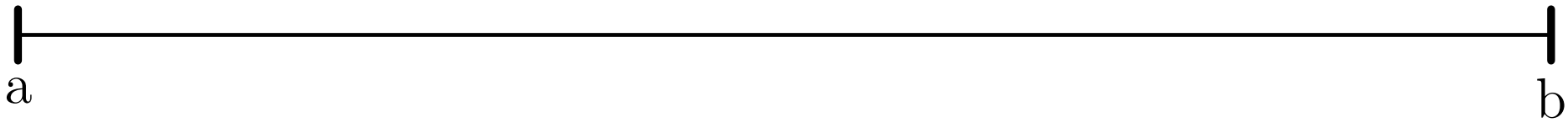
$\Longrightarrow$

Discontinuities in finite time



Solving conservation laws numerically

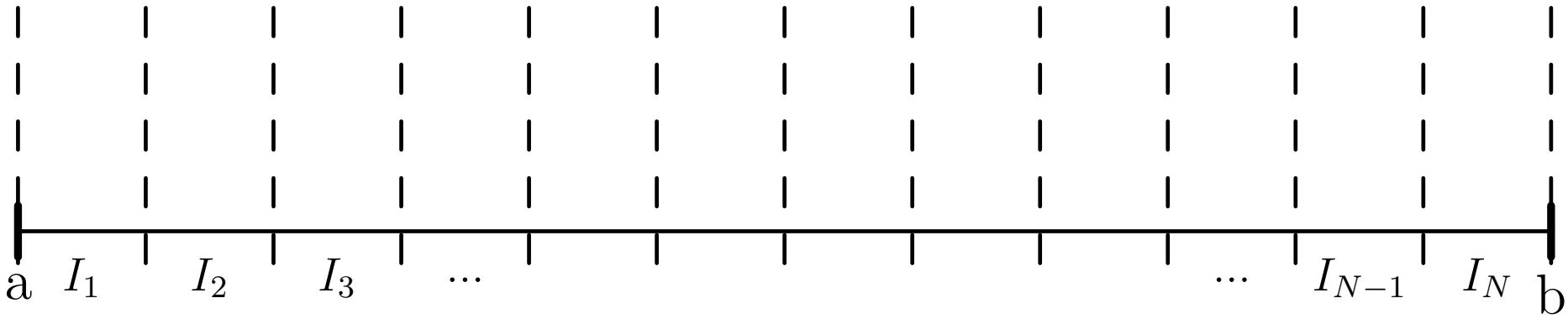
$$0 \leq t \leq T_f \qquad\qquad a \leq x \leq b$$



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

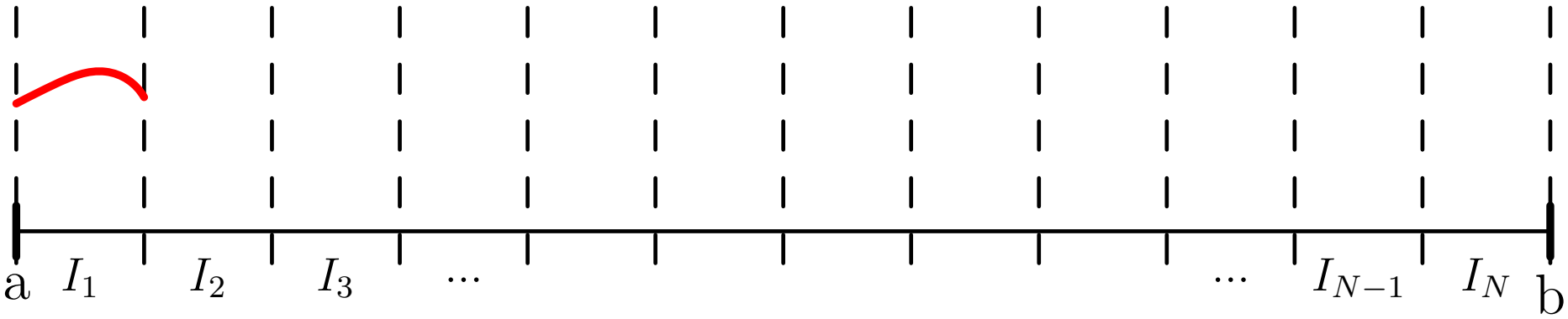


Solving conservation laws numerically

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

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

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



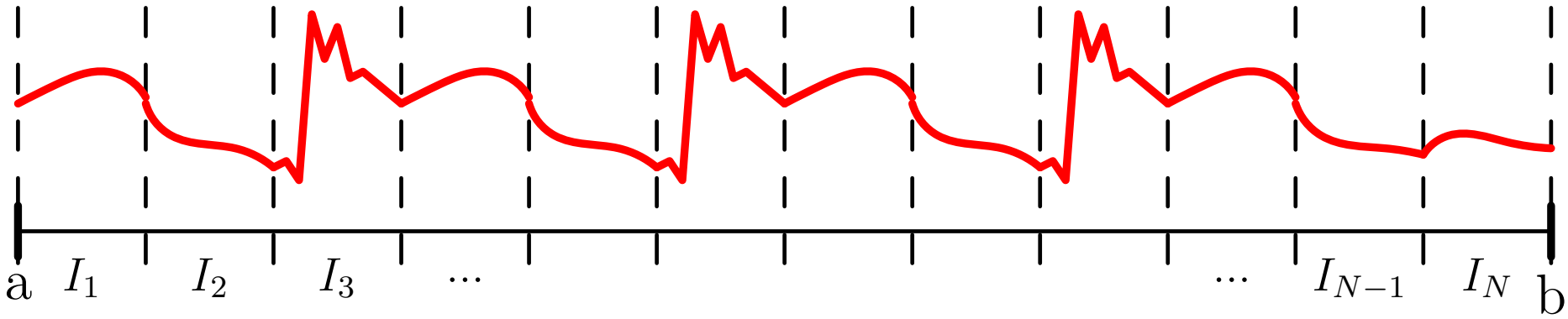
Solving conservation laws numerically

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

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

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

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



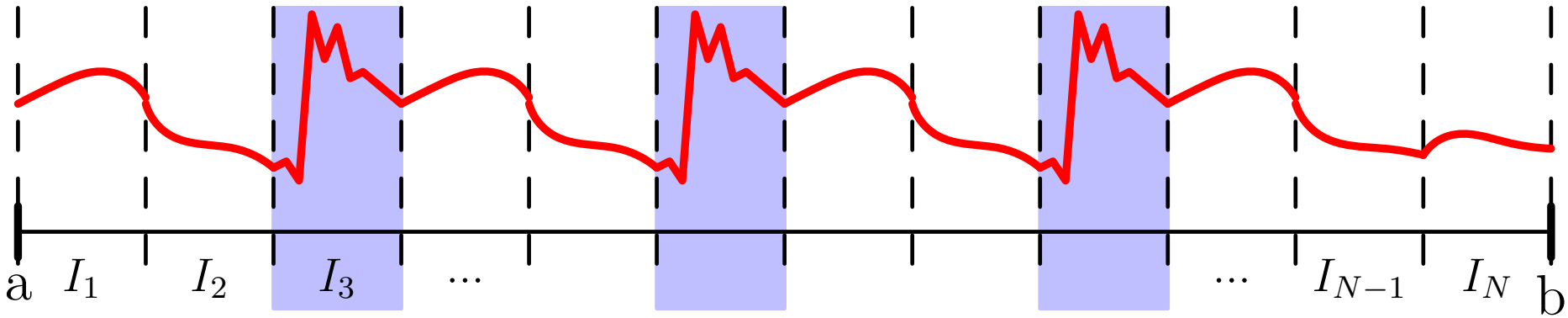
Solving conservation laws numerically

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

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

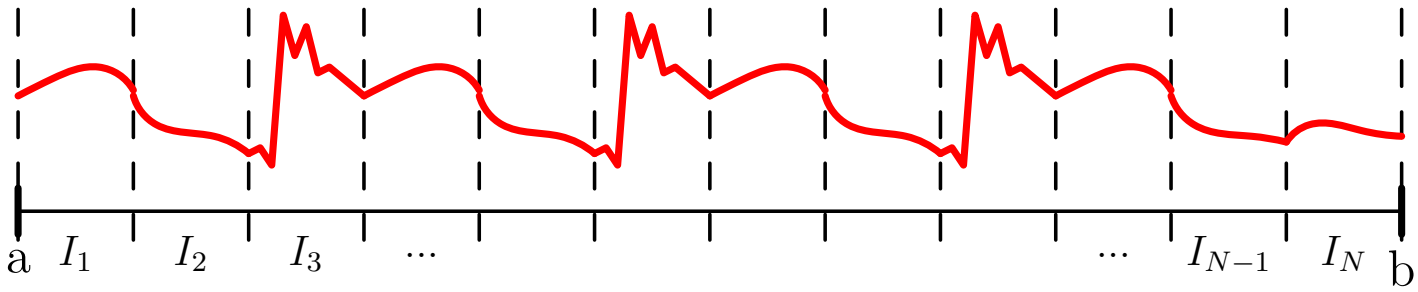
$$u_i(x) = \sum_{p=1}^{K} a_p^i \phi_p^i(x), \quad x \in I_i, \quad 1 \leq i \leq N$$

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



Detecting and limiting

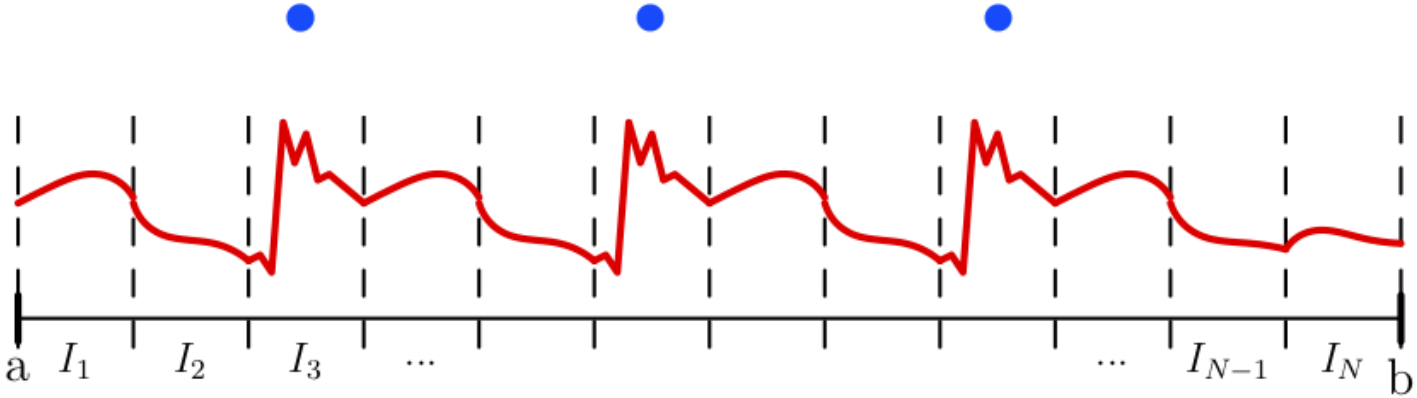
Strategy:



Detecting and limiting

Strategy:

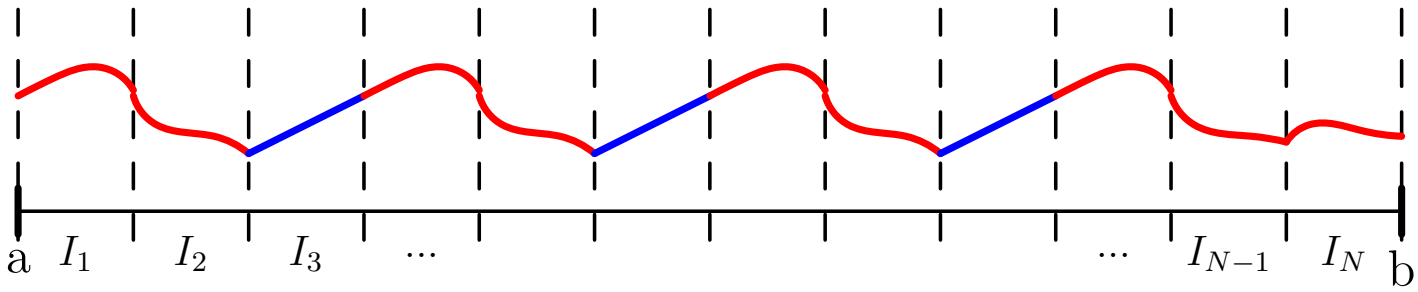
1. Find troubled-cells



Detecting and limiting

Strategy:

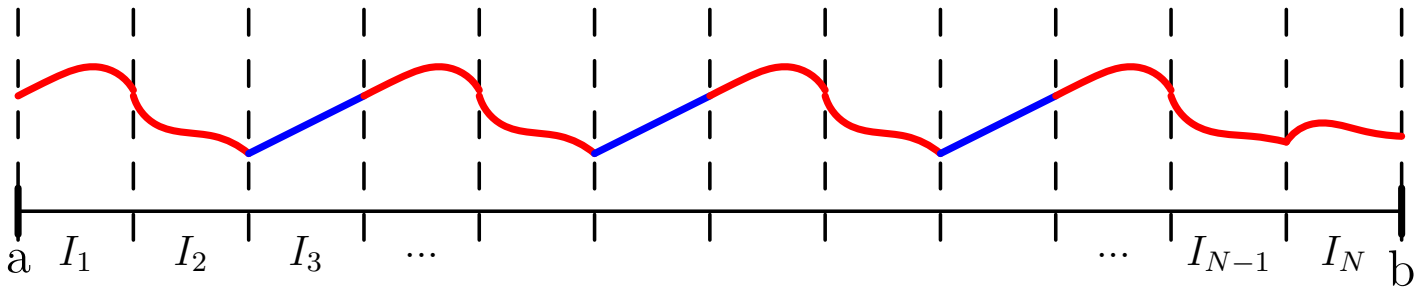
1. Find troubled-cells
2. Limit solution in flagged cells



Detecting and limiting

Strategy:

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



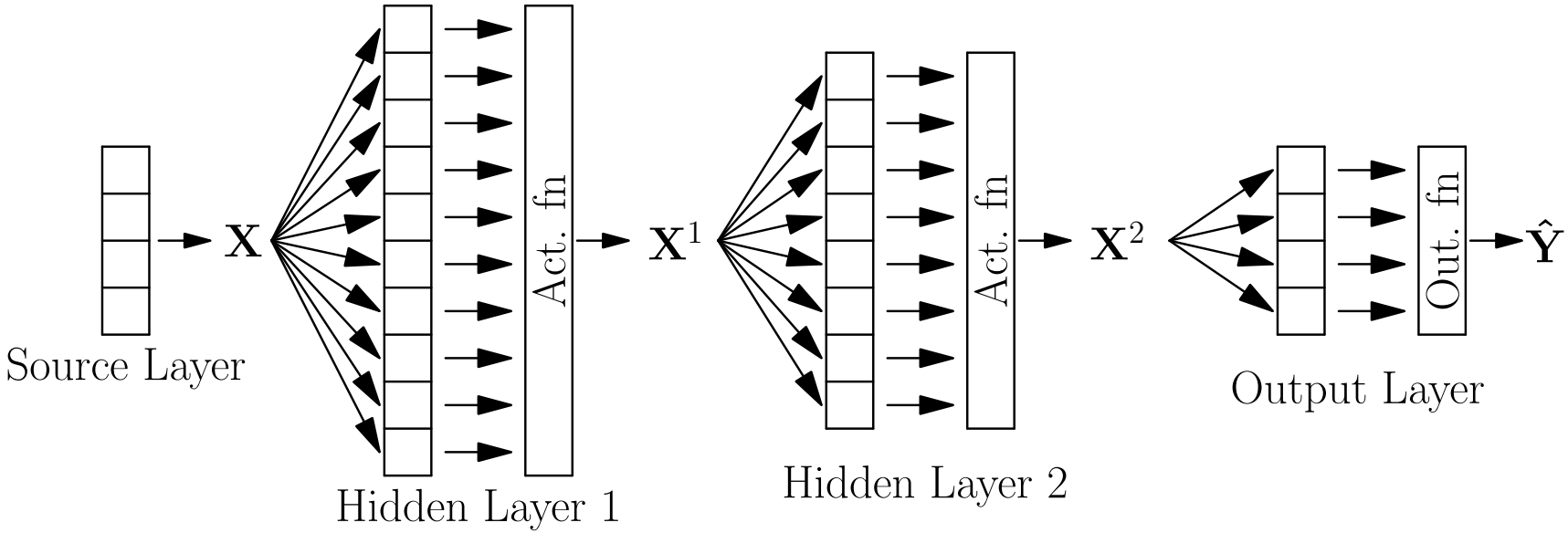
Some issues with existing detectors:

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

AIM: Train a neural network to detect discontinuities

- Free from problem-dependent parameters
- Does not flag smooth regions

An MLP-based detector

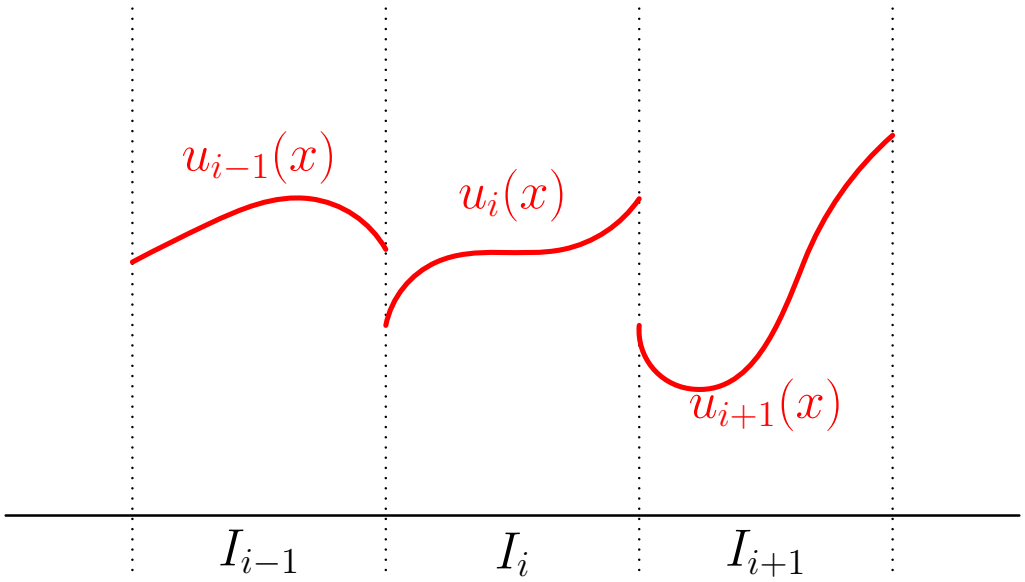


An MLP-based detector

- Input $\mathbf{X} = ?$

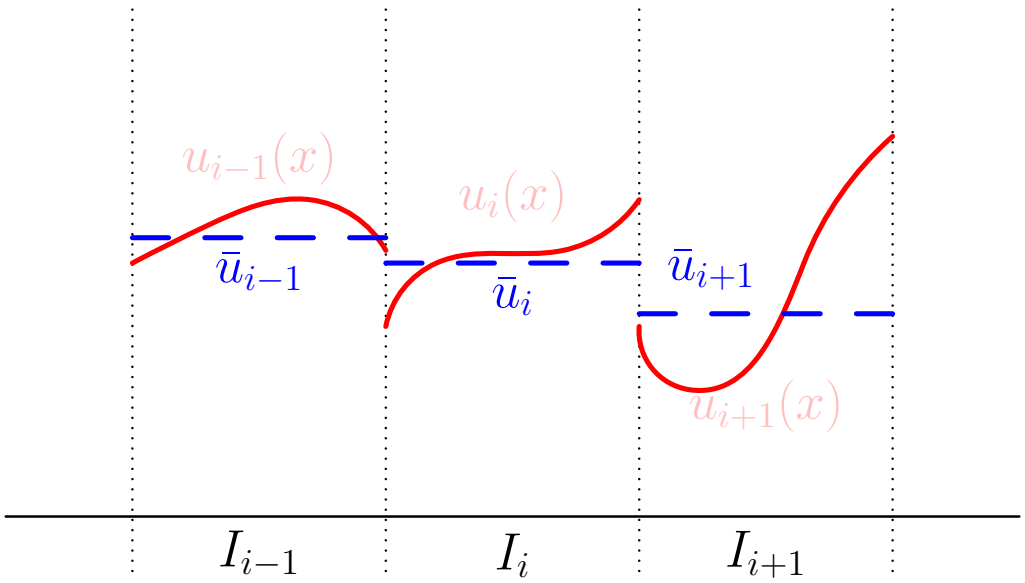
An MLP-based detector

- Input $\mathbf{X} = ?$



An MLP-based detector

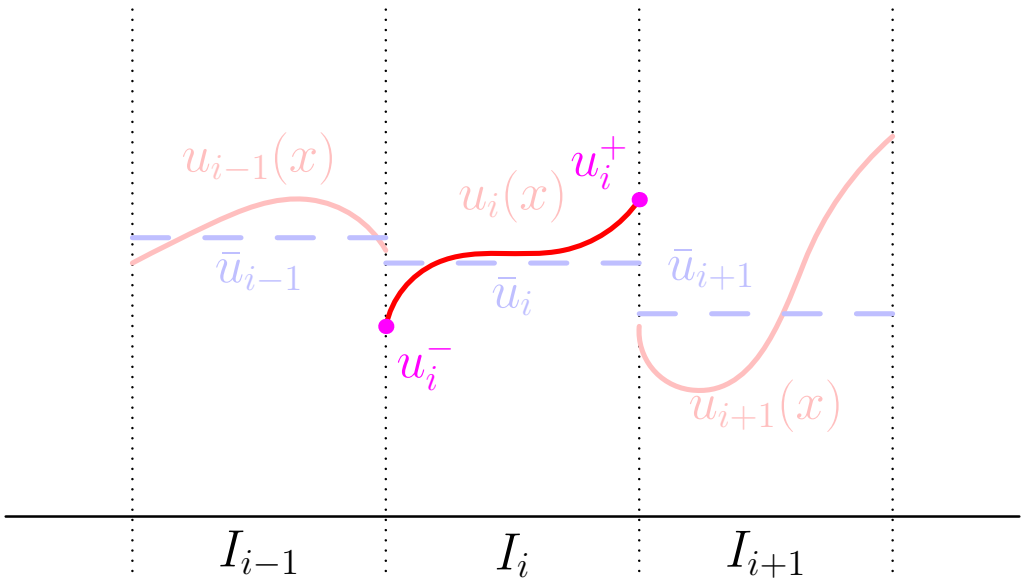
- Input $\mathbf{X} = [\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1} \quad]$



$$\overline{u}_j = \frac{1}{|I_j|} \int_{I_j} u_j(x) \mathrm{d}x$$

An MLP-based detector

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

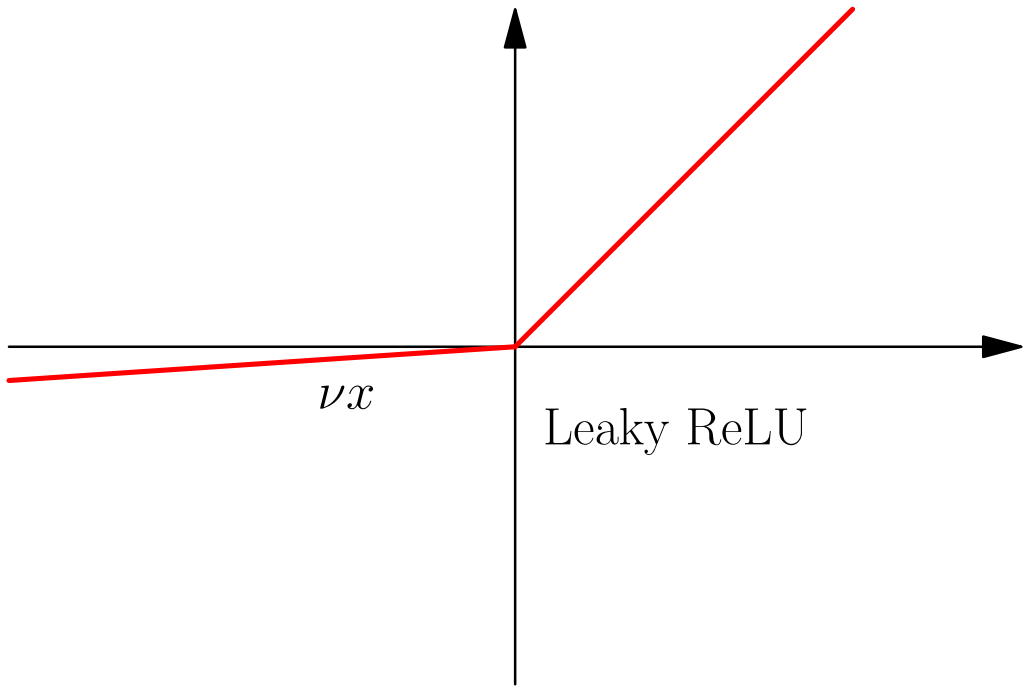


An MLP-based detector

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

An MLP-based detector

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



An MLP-based detector

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

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

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

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

An MLP-based detector

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

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

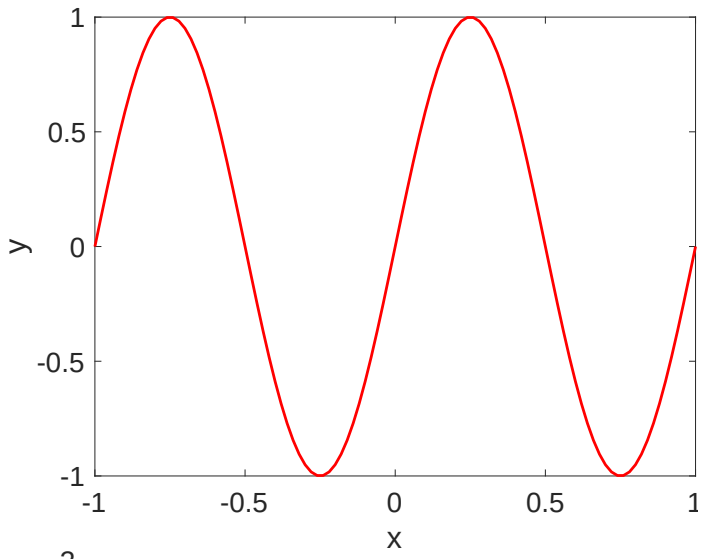
Output $\hat{Y} = [\hat{Y}^{(0)}, \hat{Y}^{(1)}] \in [0,1]^2$

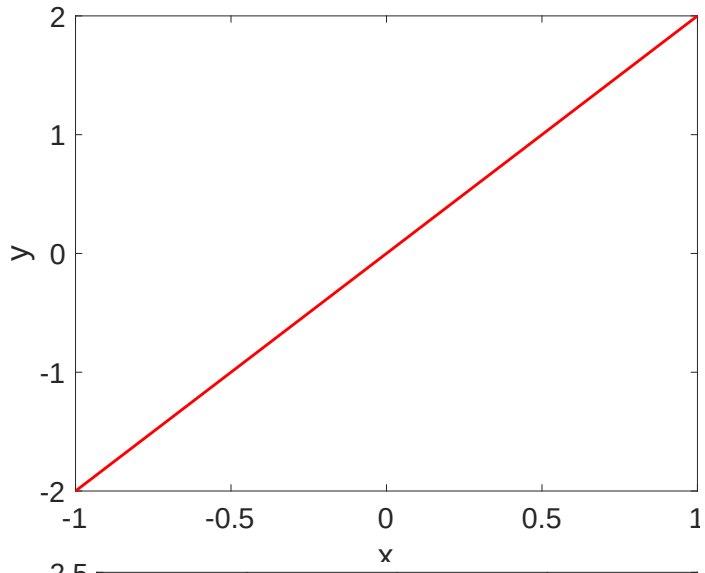
Troubled-cell if $\hat{Y}^{(0)} > 0.5$

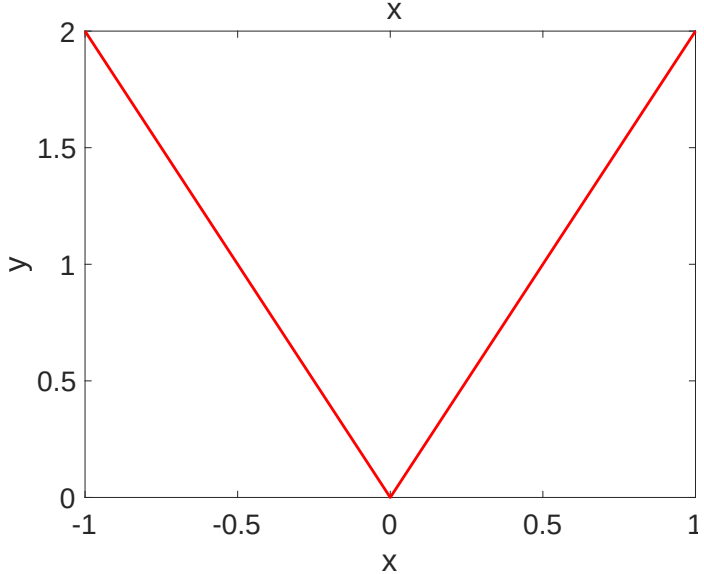
- Cost functional: L2 regularized cross-entropy

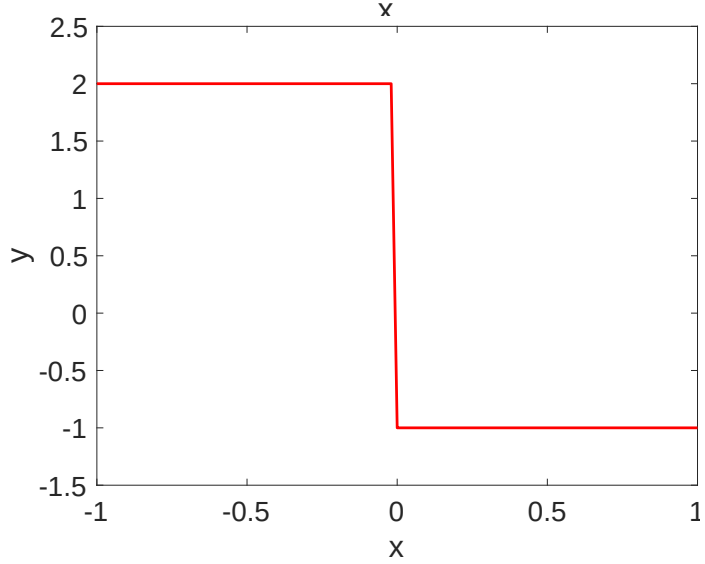
Generating the training data for indicator

Types of functions used:



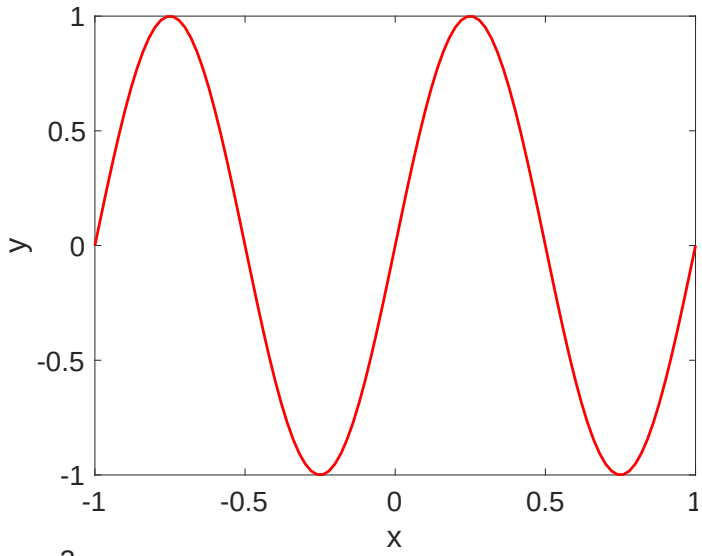


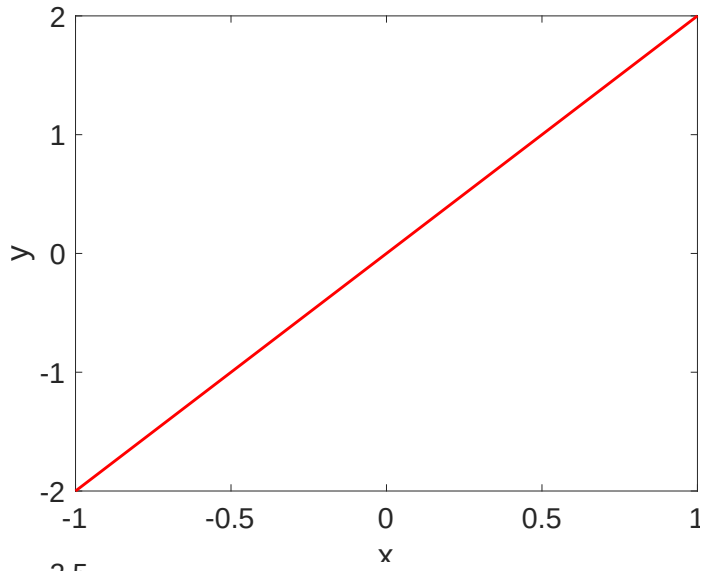


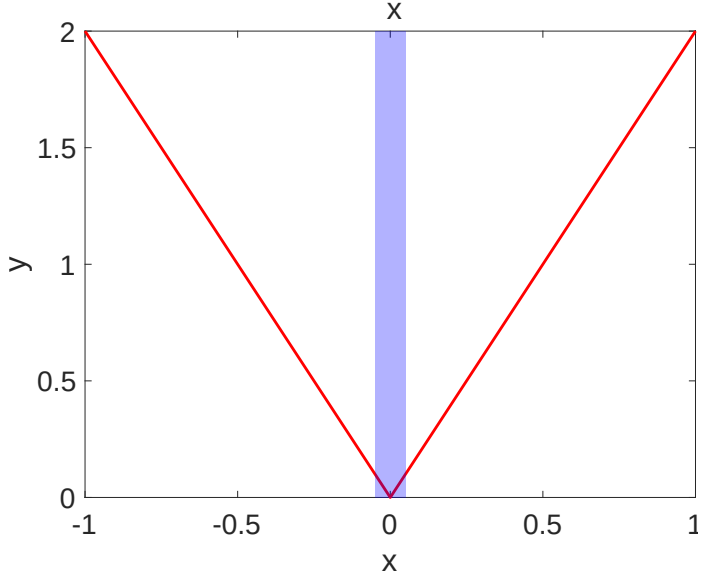


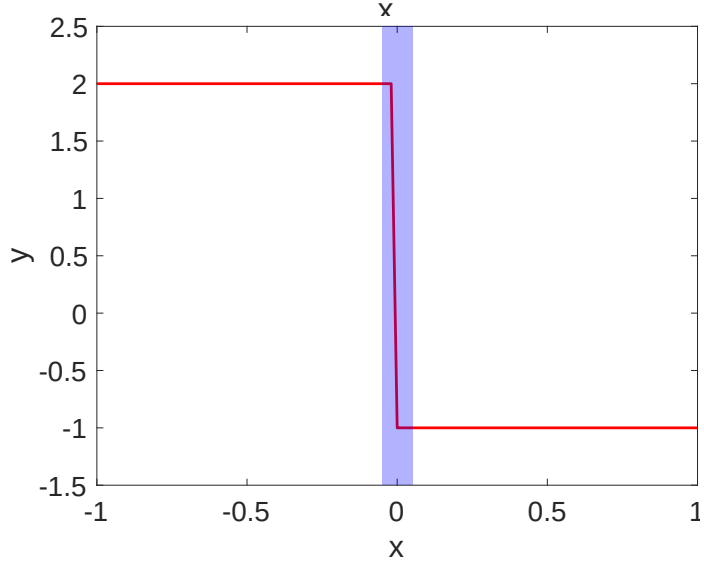
Generating the training data for indicator

Types of functions used:



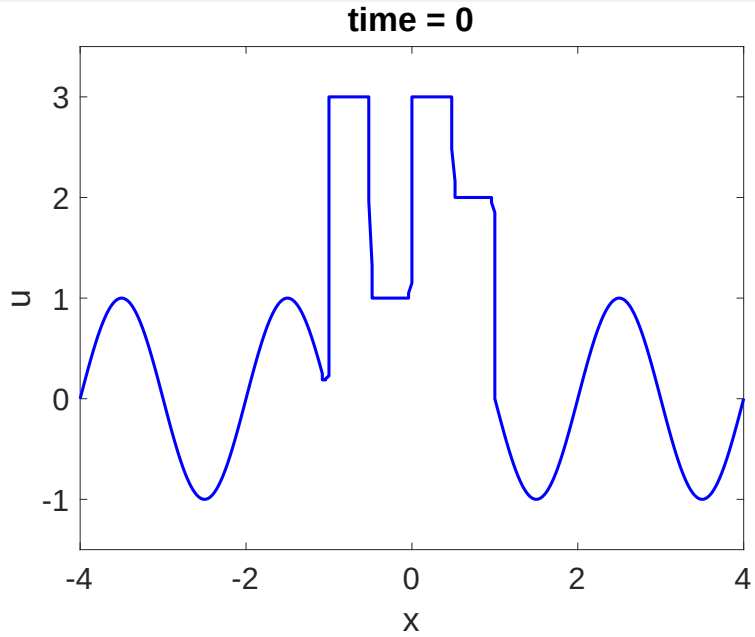


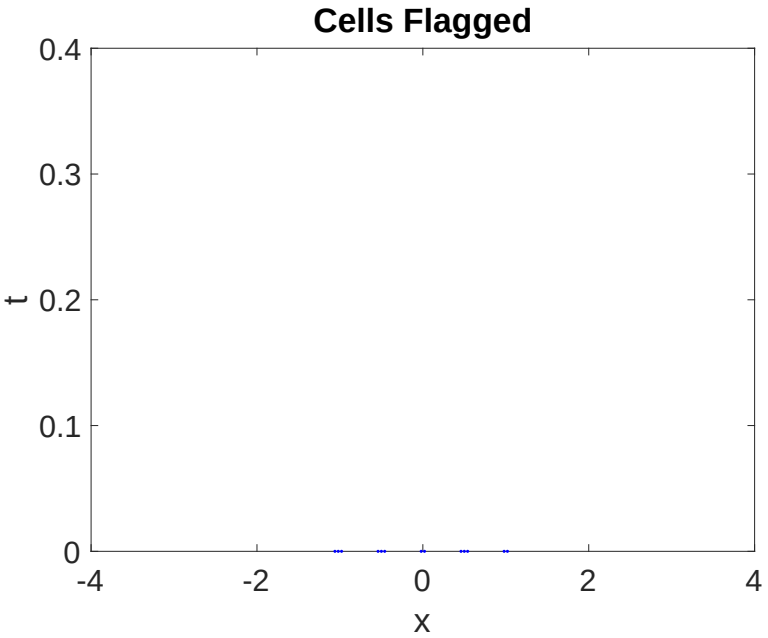




Burgers' equation

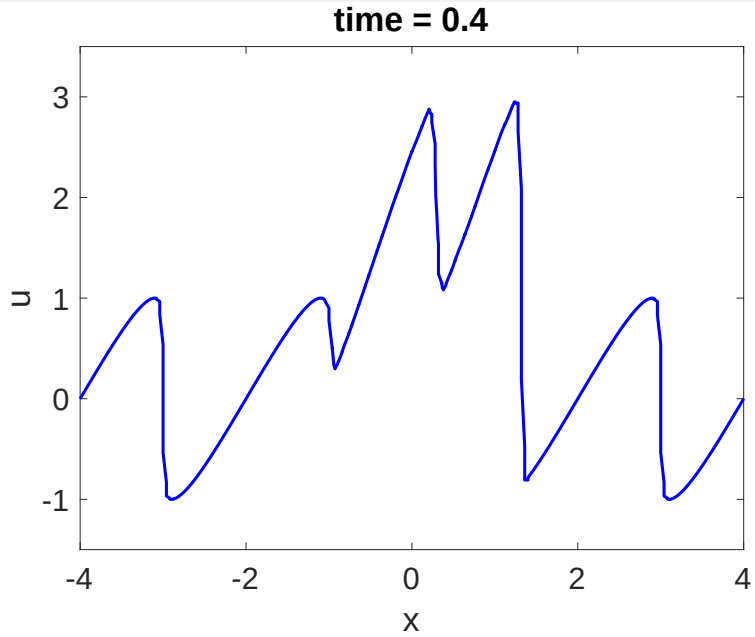
$$\partial_t u + \partial_x \left(\frac{u^2}{2} \right) = 0$$

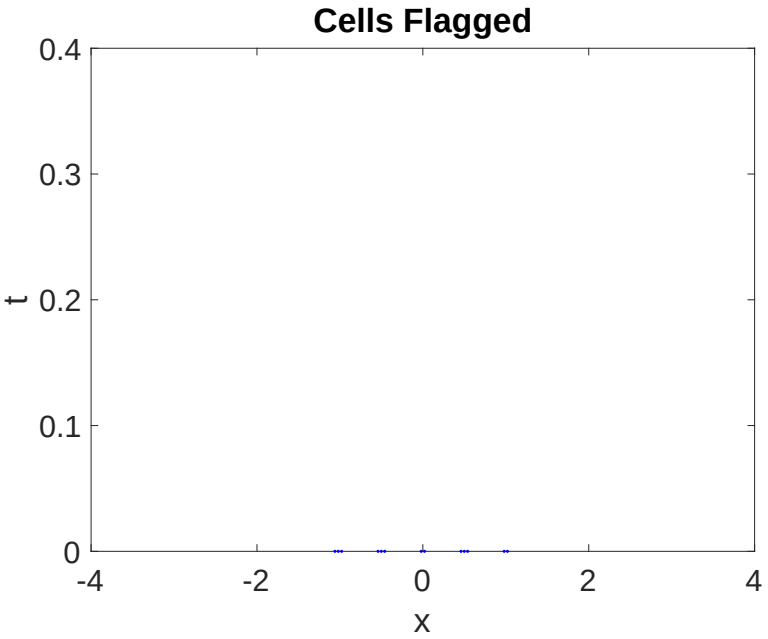




Burgers' equation

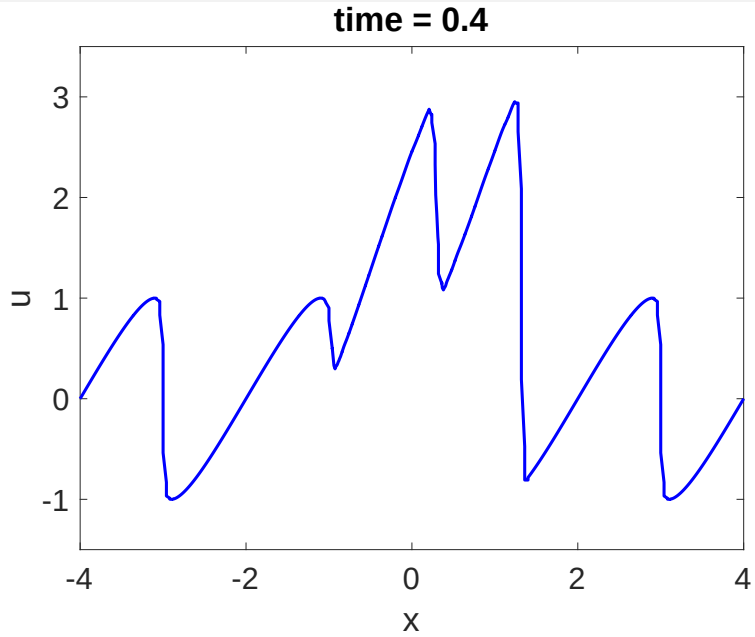
$$\partial_t u + \partial_x \left(\frac{u^2}{2} \right) = 0$$

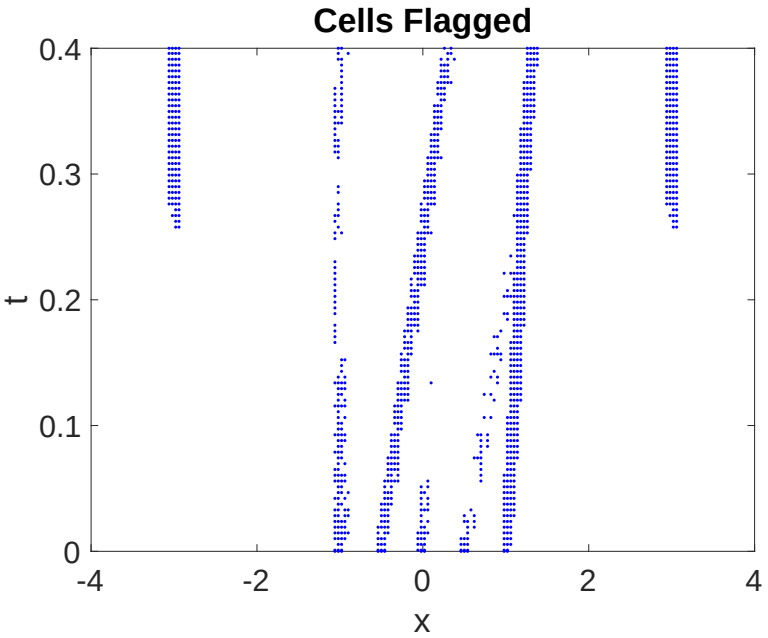




Burgers' equation

$$\partial_t u + \partial_x \left(\frac{u^2}{2} \right) = 0$$





Euler equations

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

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

ρ : density

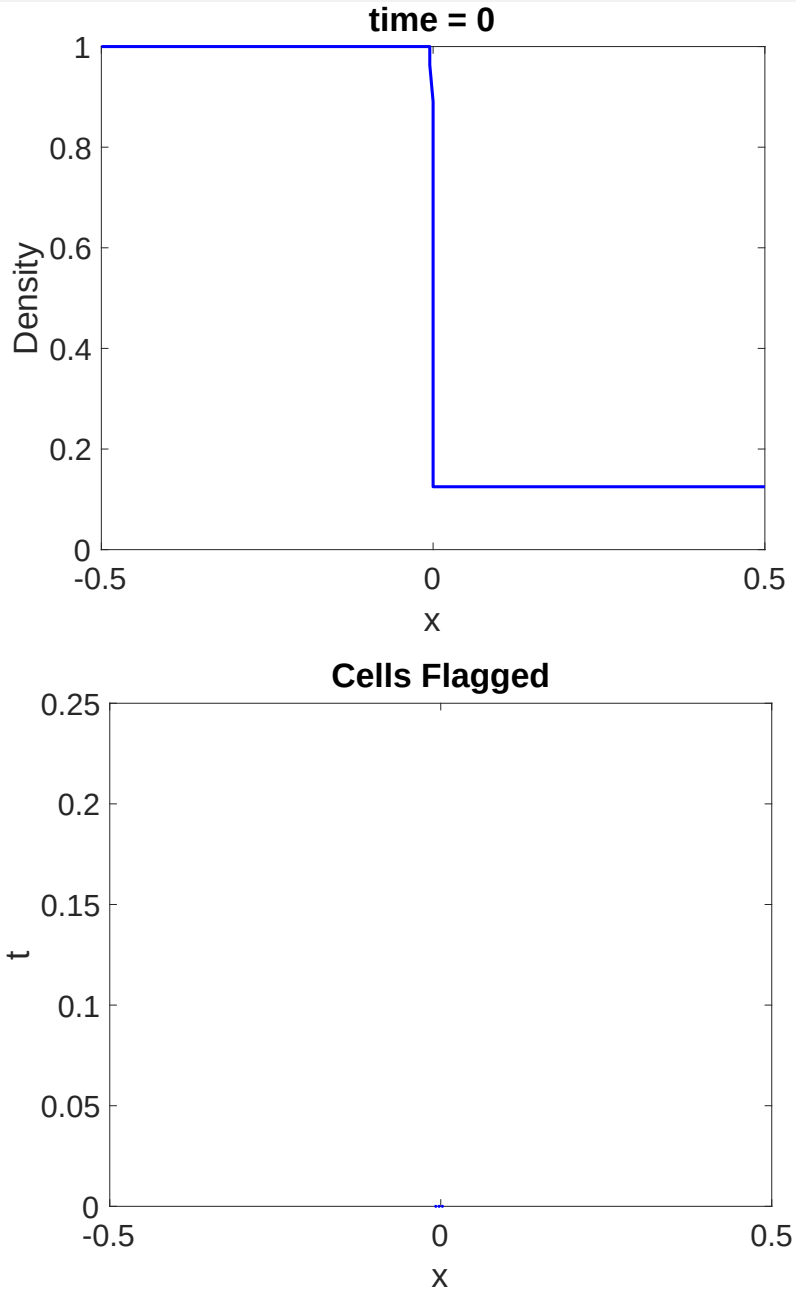
u : velocity

p : pressure

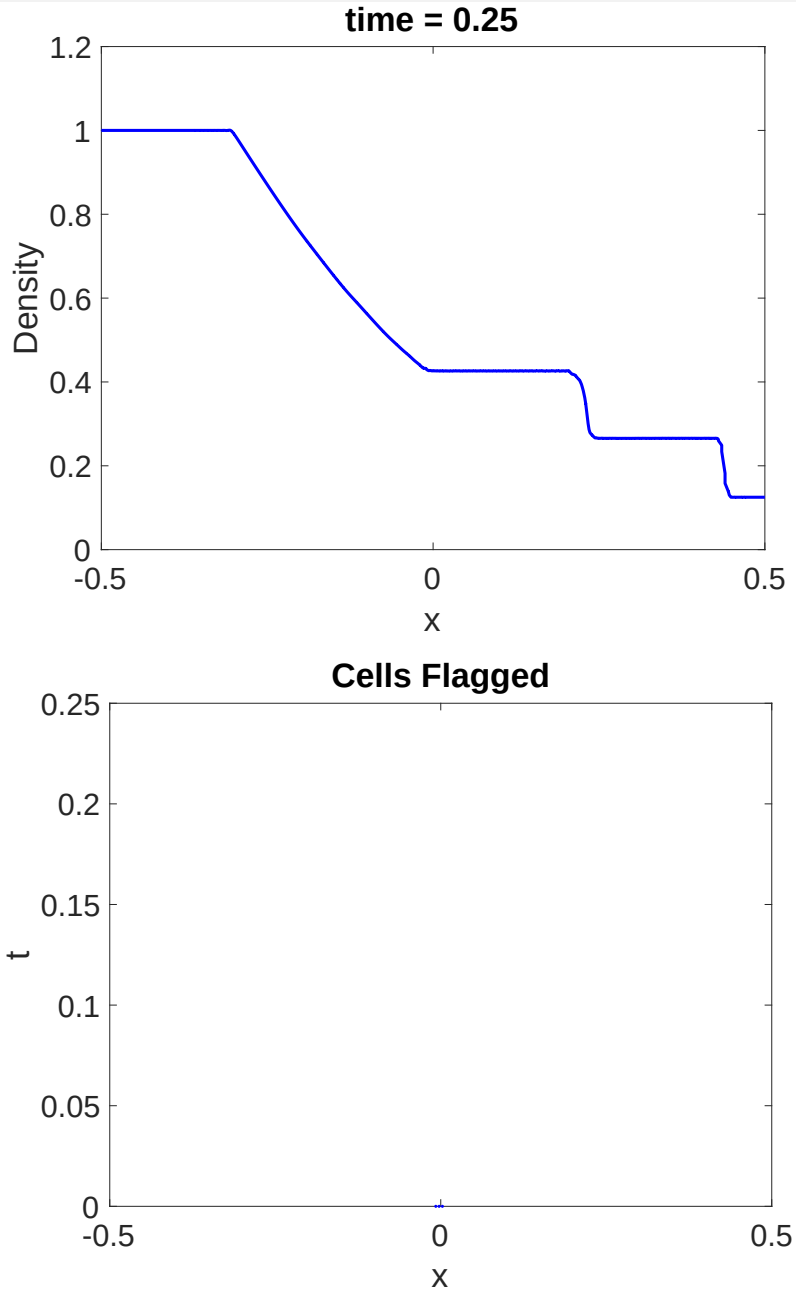
E : total energy

e : internal energy

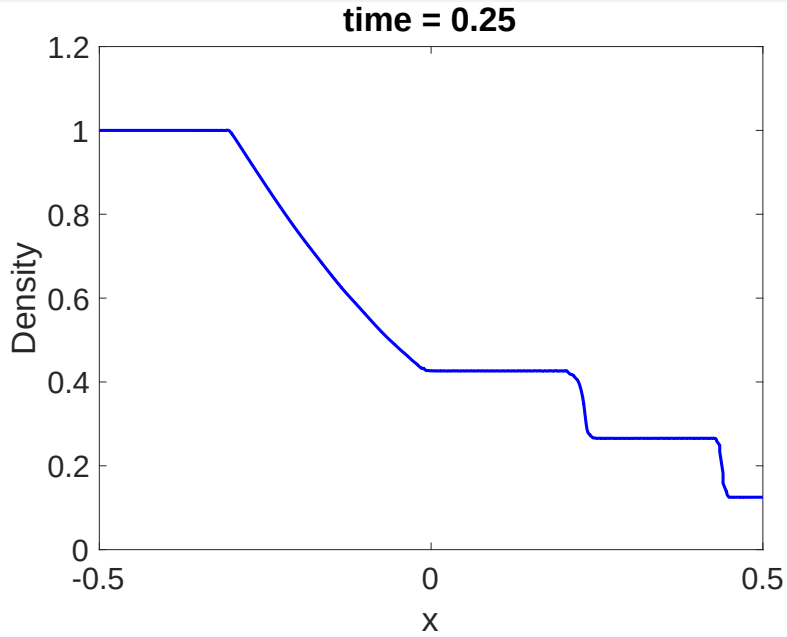
Euler equations: Shock-tube problem

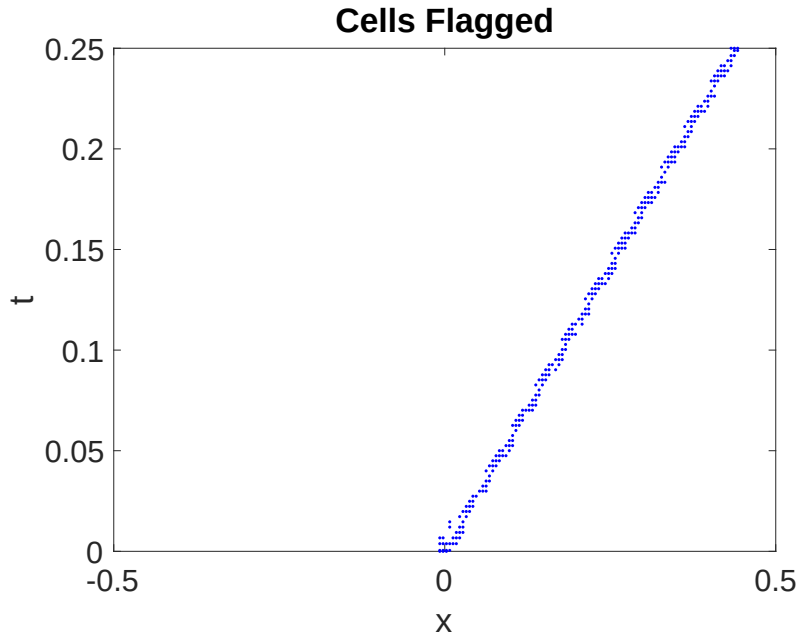


Euler equations: Shock-tube problem

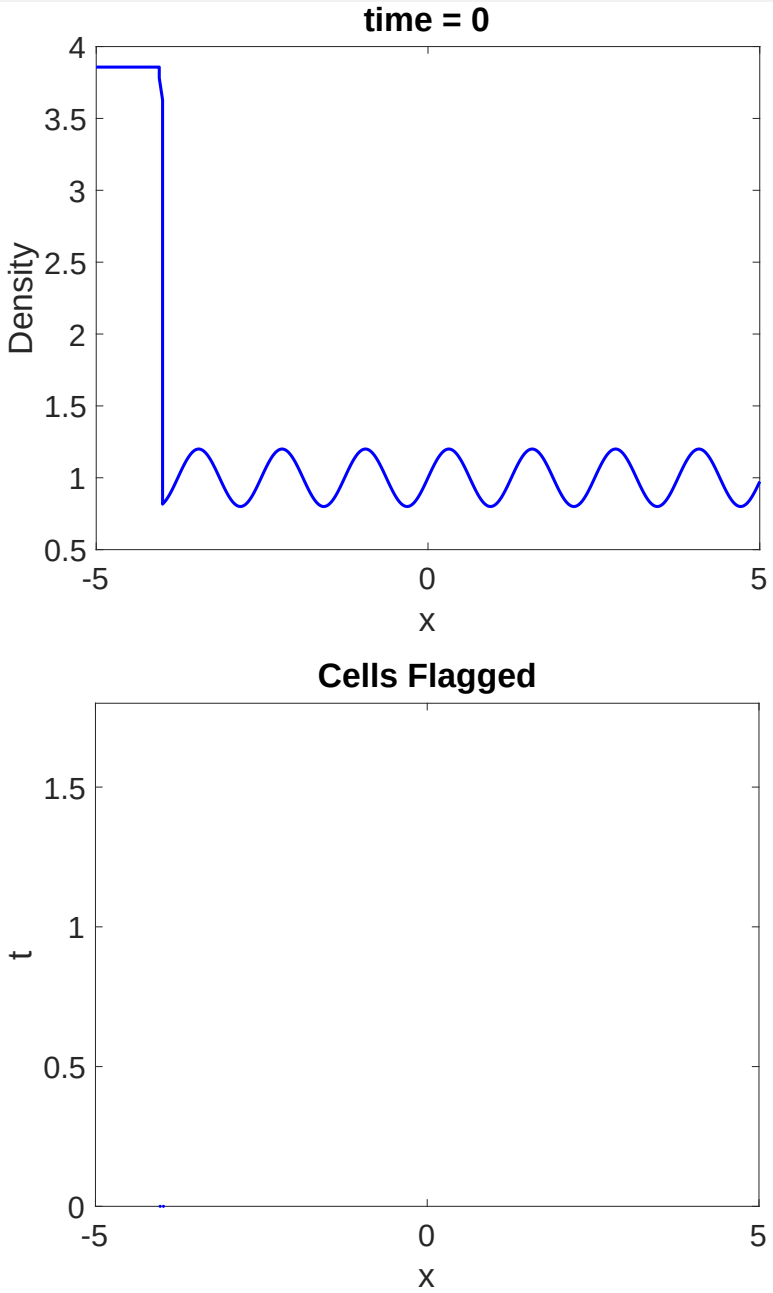


Euler equations: Shock-tube problem

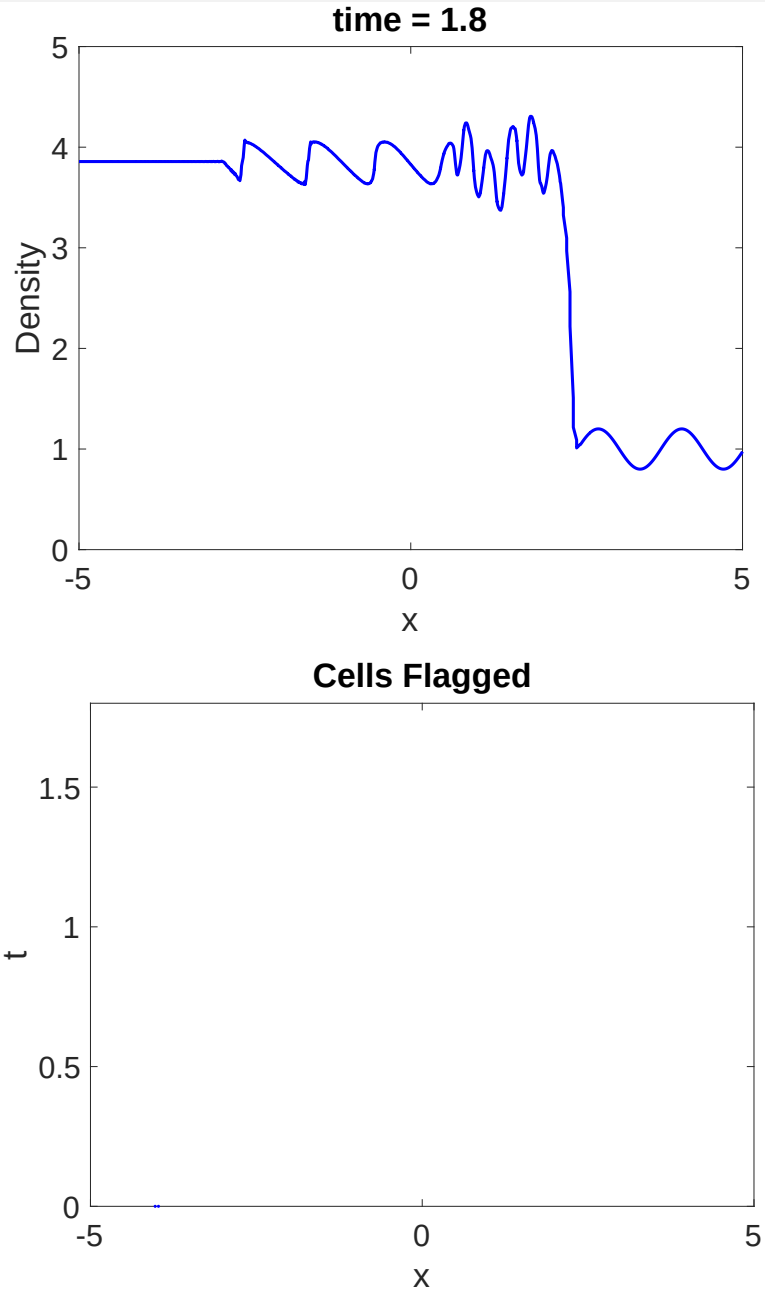




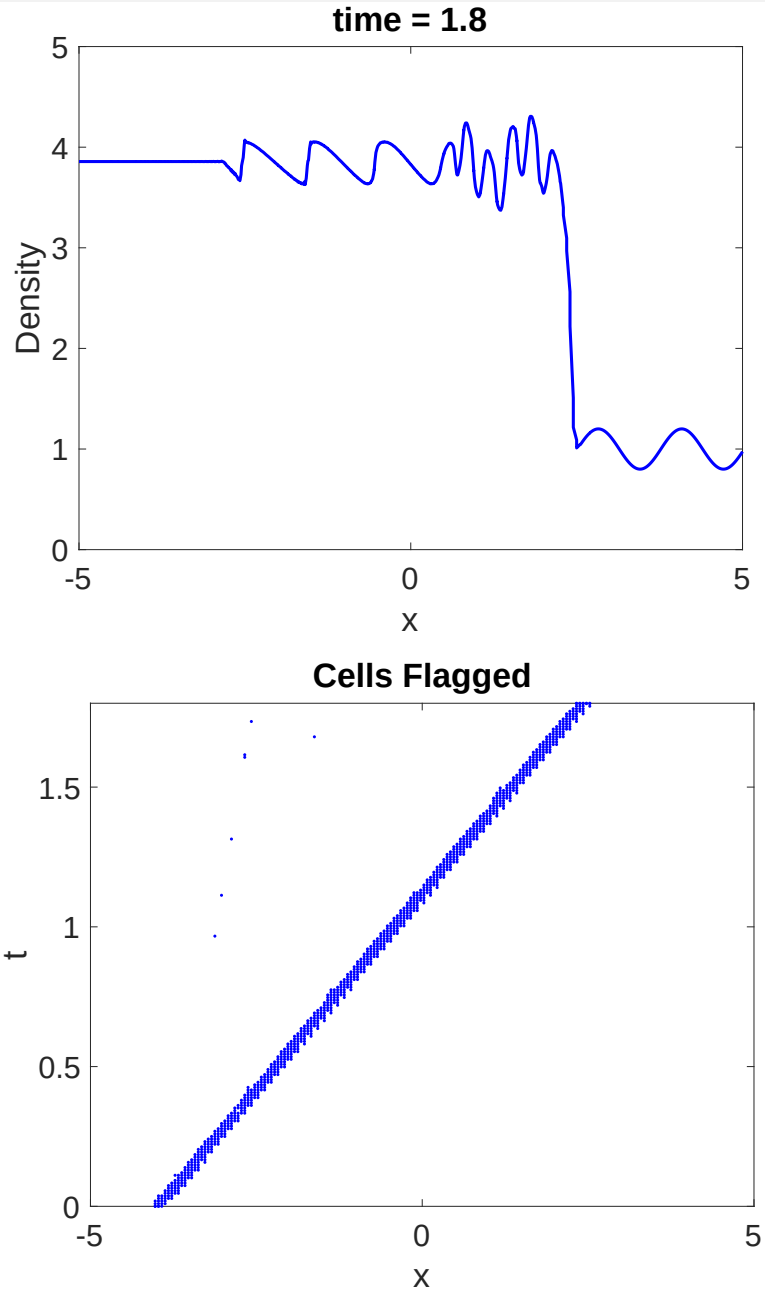
Euler equations: Shock-entropy problem



Euler equations: Shock-entropy problem



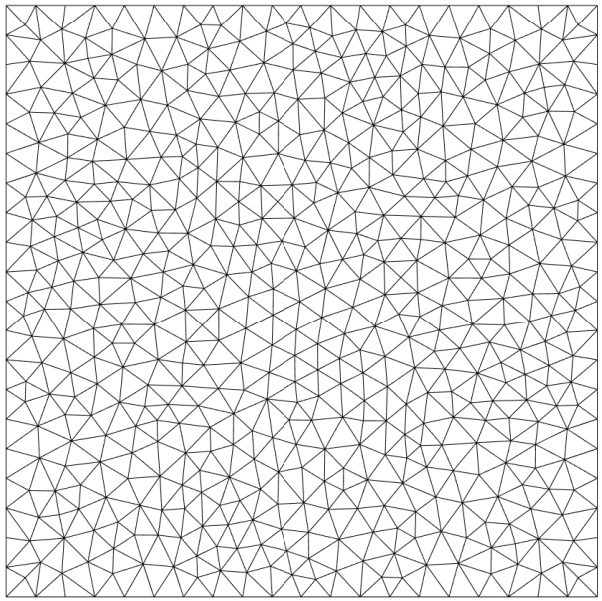
Euler equations: Shock-entropy problem

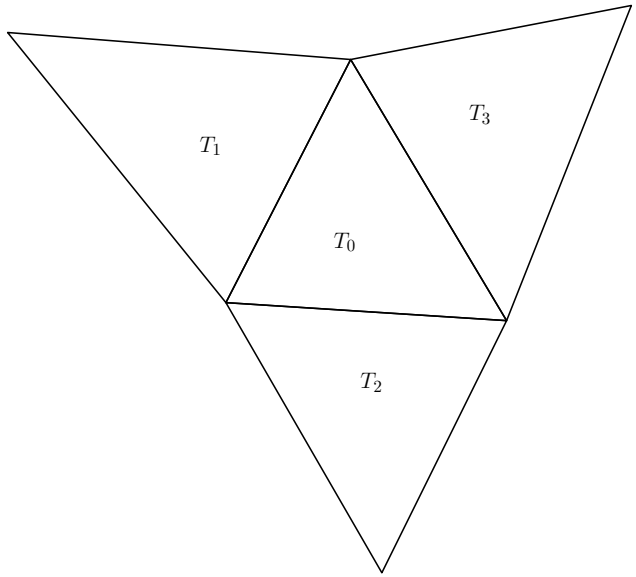


What else?

What else?

- We can also do this in 2D





What else?

- We can also do this in 2D
- Add artificial viscosity instead of limiting

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}(\mathbf{u})}{\partial x} = \mu \frac{\partial^2 \mathbf{u}}{\partial x^2}$$

μ based on local smoothness $\longrightarrow$ predicted by a network

What else?

- We can also do this in 2D
- Add artificial viscosity instead of limiting

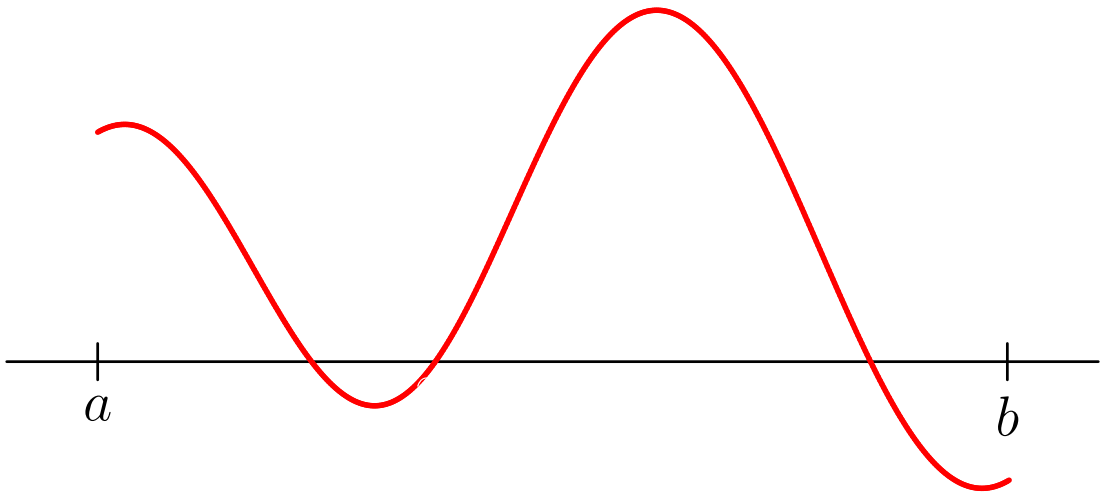
$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}(\mathbf{u})}{\partial x} = \mu \frac{\partial^2 \mathbf{u}}{\partial x^2}$$

  μ based on local smoothness $\longrightarrow$ predicted by a network

- Similar applications (see deepray.github.io/publications)

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

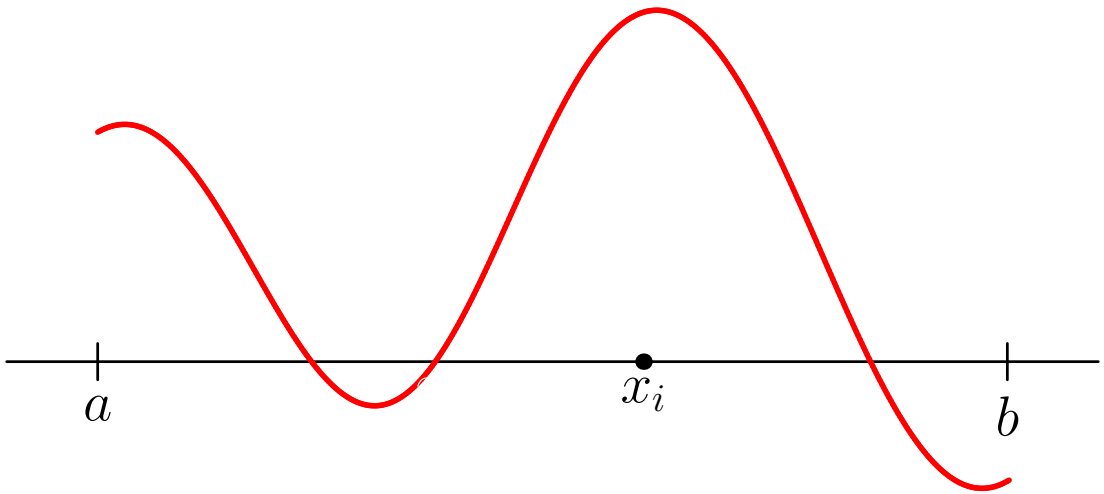
Generating the training data for indicator



Data sampling is achieved by

- Choose a known function $u(x)$ – for eg $\sin(x)$, x^p, etc

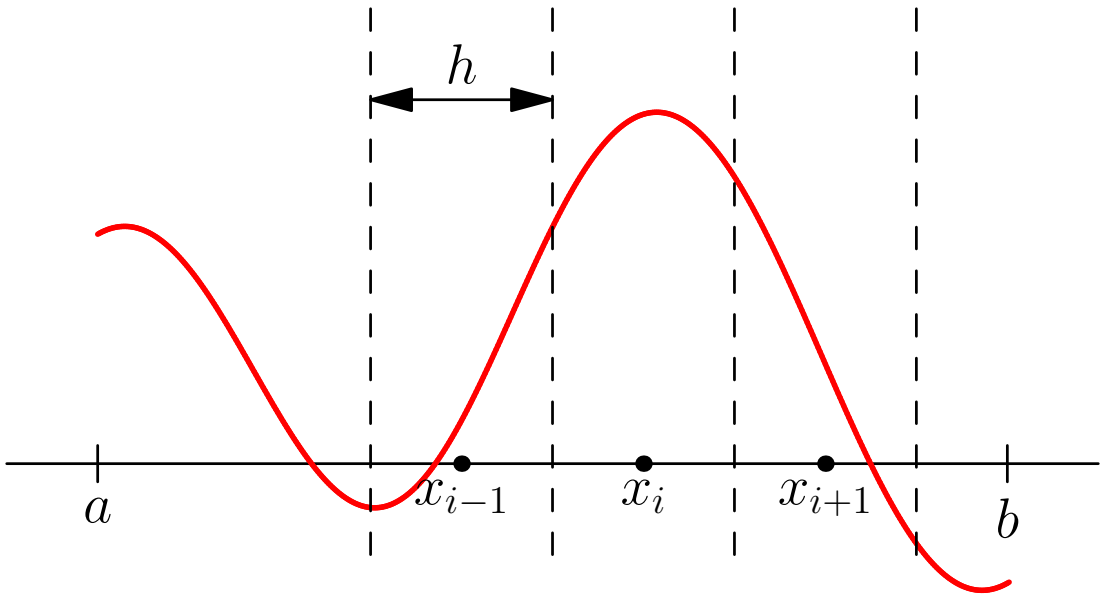
Generating the training data for indicator



Data sampling is achieved by

- Choose a known function $u(x)$ – for eg $\sin(x)$, x^p, etc
- Pick a point x_i

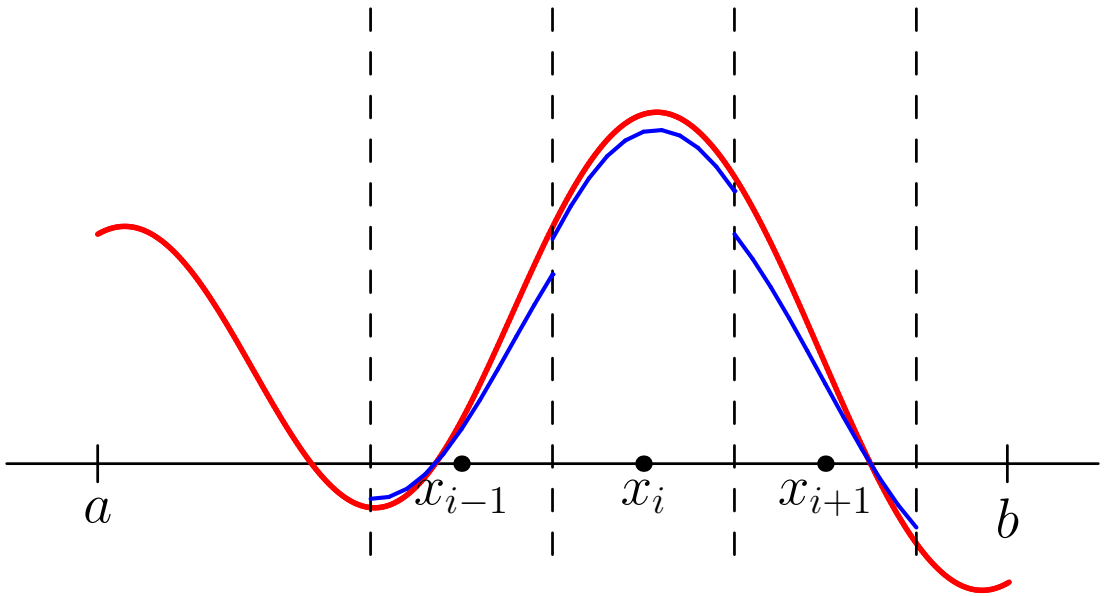
Generating the training data for indicator



Data sampling is achieved by

- Choose a known function $u(x)$ – for eg $\sin(x)$, x^p, etc
- Pick a point x_i
- Pick a cell size h and make stencil

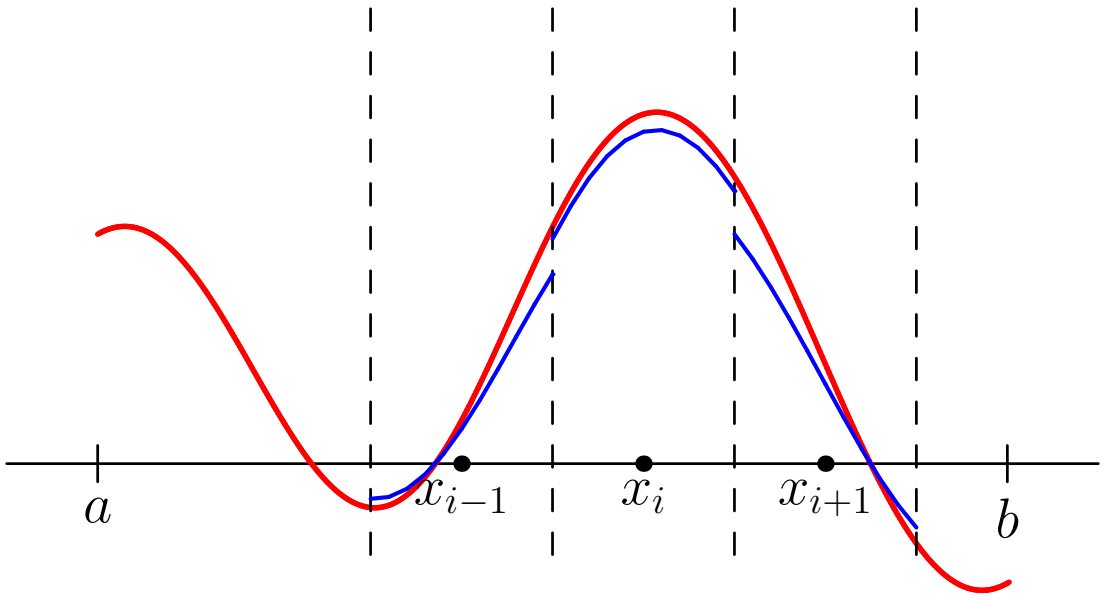
Generating the training data for indicator



Data sampling is achieved by

- Choose a known function $u(x)$ – for eg $\sin(x)$, x^p, etc
- Pick a point x_i
- Pick a cell size h and make stencil
- Project on space generated by basis

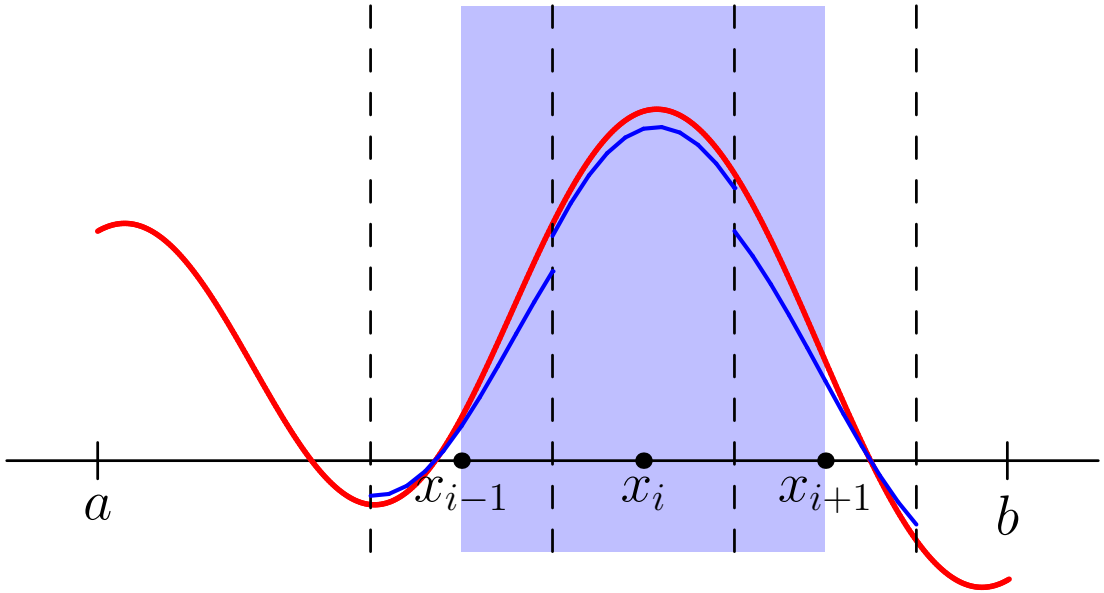
Generating the training data for indicator



Data sampling is achieved by

- Choose a known function $u(x)$ – for eg $\sin(x)$, x^p, etc
- Pick a point x_i
- Pick a cell size h and make stencil
- Project on space generated by basis
- Extract needed data $[\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u_i^-, u_i^+]$

Generating the training data for indicator



Data sampling is achieved by

- Choose a known function $u(x)$ – for eg $\sin(x)$, x^p, etc
- Pick a point x_i
- Pick a cell size h and make stencil
- Project on space generated by basis
- Extract needed data $[\overline{u}_{i-1}, \overline{u}_i, \overline{u}_{i+1}, u_i^-, u_i^+]$
- Flag cell if discontinuity in $[x_{i-\frac{1}{2}} - h/2, x_{i+\frac{1}{2}} + h/2]$