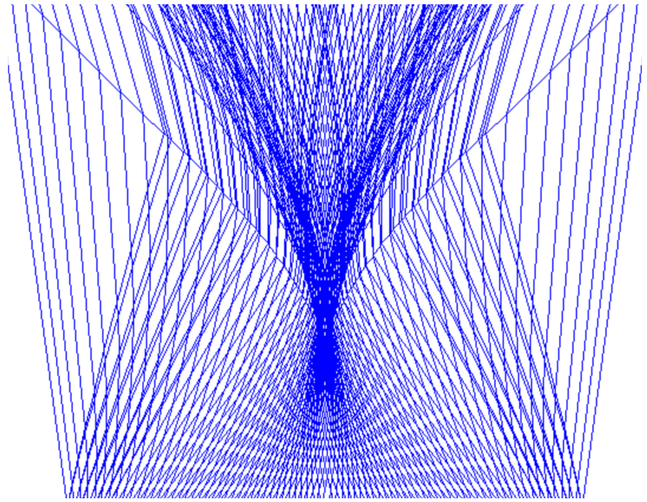
Physics-Informed Neural Networks for Hyperbolic Conservation Laws



Presentated by

Sagar Prakash Barad, 2011137

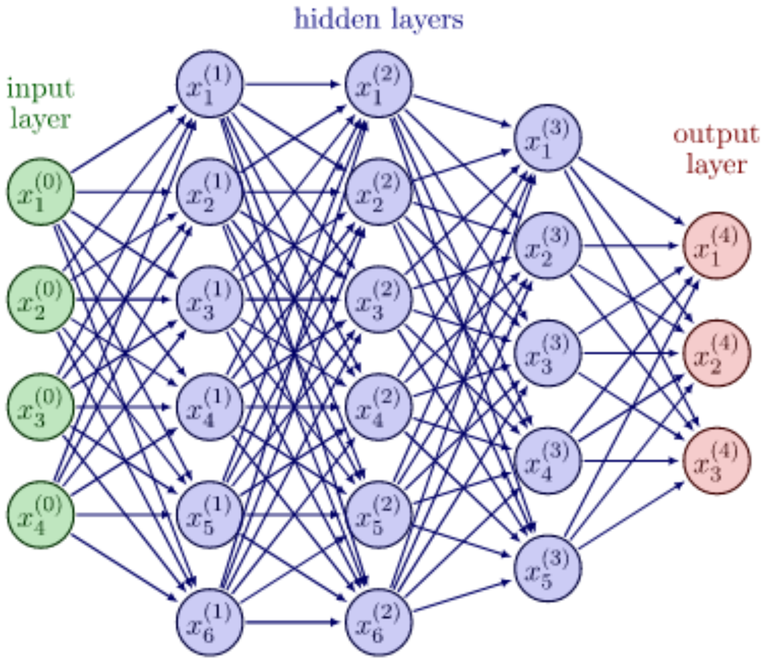
Under the Supervision of

Prof. Victor Roy & Prof. Subhasish Basak

The Universal Approximation Theorem

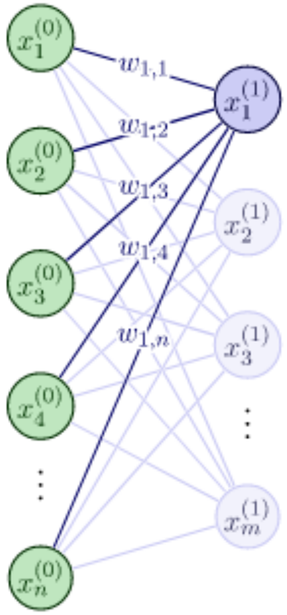
What is a Neural Network?

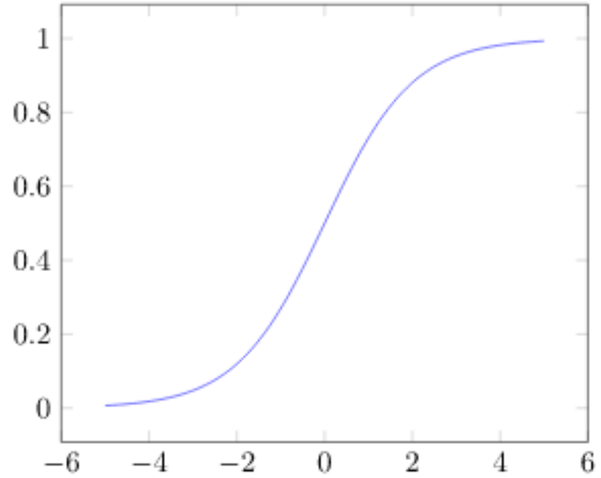
Ans: function approximator

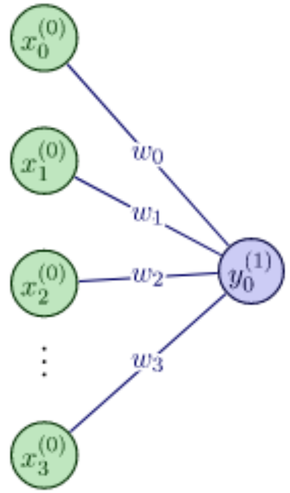


$$W^{(l)} = \begin{pmatrix} w_{1,1}^{(l)} & w_{1,2}^{(l)} & \dots & w_{1,n}^{(l)} \\ w_{2,1}^{(l)} & w_{2,2}^{(l)} & \dots & w_{2,n}^{(l)} \\ \dots & \dots & \ddots & \\ w_{m,1}^{(l)} & w_2^{(l)} & \dots & w_{m,n}^{(l)} \end{pmatrix}$$

$$\psi(x) := \sigma\left(\sum_{i=1}^{n} x_i w_i - b\right) = \sigma(\underbrace{w^\top \bullet x - b}_{\in \mathbb{R}}),$$







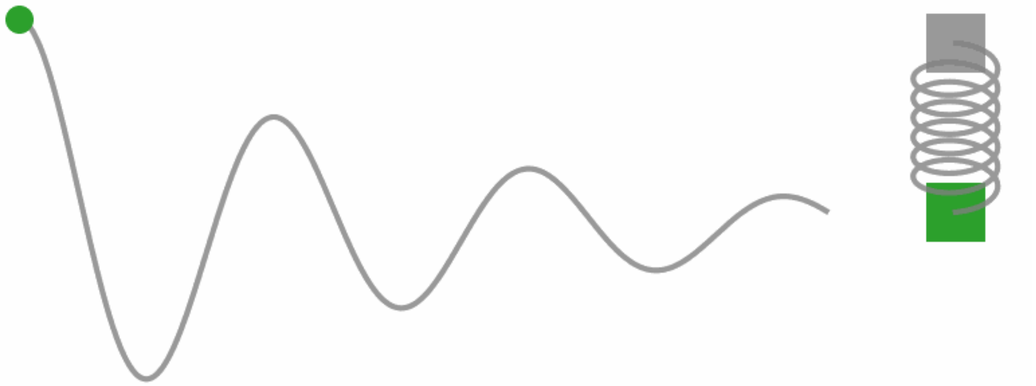
$$\int_{x \in \mathbb{R}^n} \sigma(w^\top x - b)\, d\mu(x) = 0 \quad \forall w \in \mathbb{R}^n, b \in \mathbb{R}$$

Source: https://www.deep-mind.org/2023/03/26/the-universal-approximation-theorem/

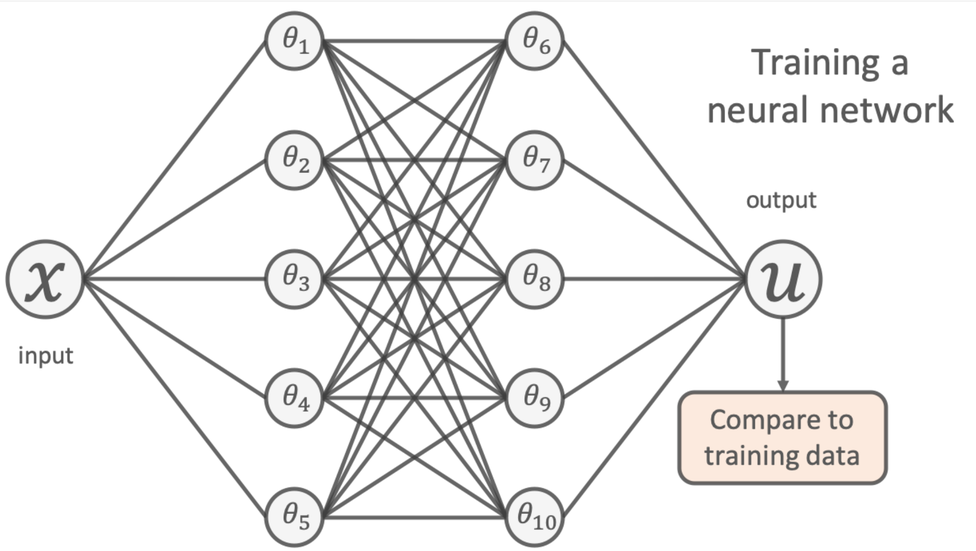
Can AI do Physics?

Specifically, can it approximate the solution to PDE or ODE, given sufficient data?

1D damped harmonic oscillator



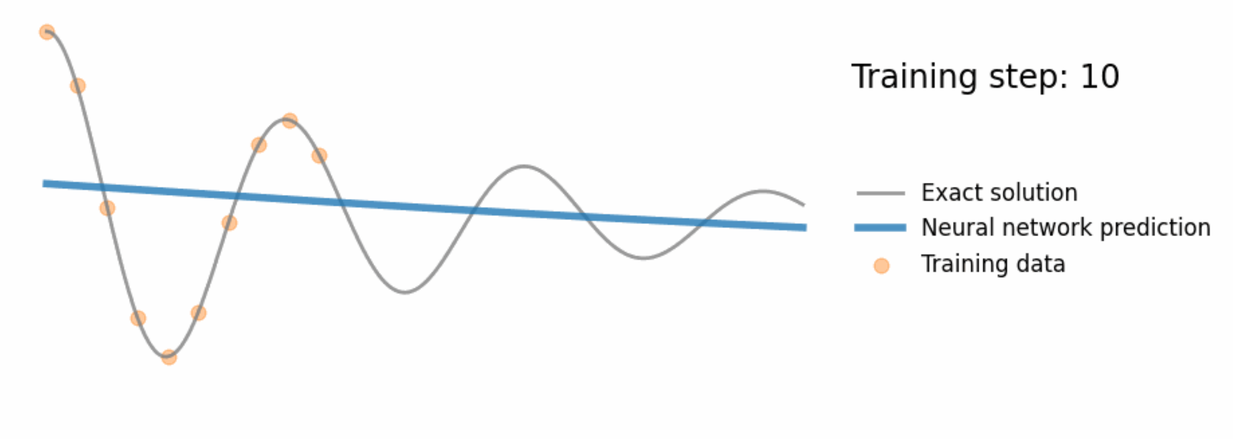
$$m\frac{d^2u}{dx^2} + \mu\frac{du}{dx} + ku = 0$$



$$u_{NN} = u_{solution}\,??$$

Source: https://benmoseley.blog/my-research/so-what-is-a-physics-informed-neural-network/

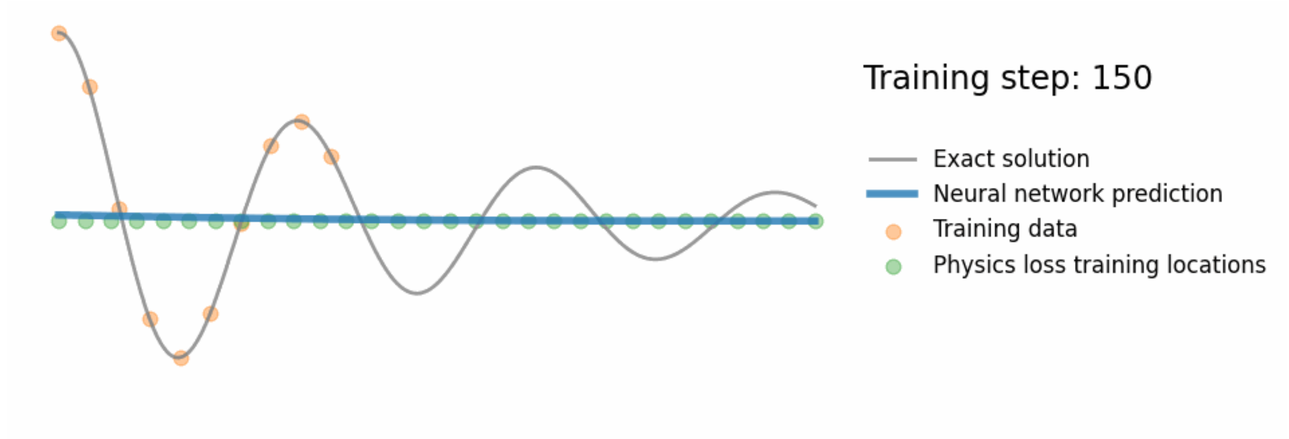
‘Naive’ Neural Network



$$\min \frac{1}{N}\sum_{i}^{N}(u_{\mathrm{NN}}(x_i;\theta)-u_{\mathrm{true}}(x_i))^2$$

“Data loss”

‘Physics Informed’ Neural Network



$$\min \frac{1}{N}\sum_{i}^{N}(u_{\mathrm{NN}}(x_i;\theta)-u_{\mathrm{true}}(x_i))^2 + \frac{1}{M}\sum_{j}^{M}\left(\left[m\frac{d^2}{dx^2}+\mu\frac{d}{dx}+k\right]u_{\mathrm{NN}}(x_j;\theta)\right)^2$$

“physics loss”

Source: https://benmoseley.blog/my-research/so-what-is-a-physics-informed-neural-network/

Physics Informed Neural Network (PINNs) can solve PDEs

Forward Problem

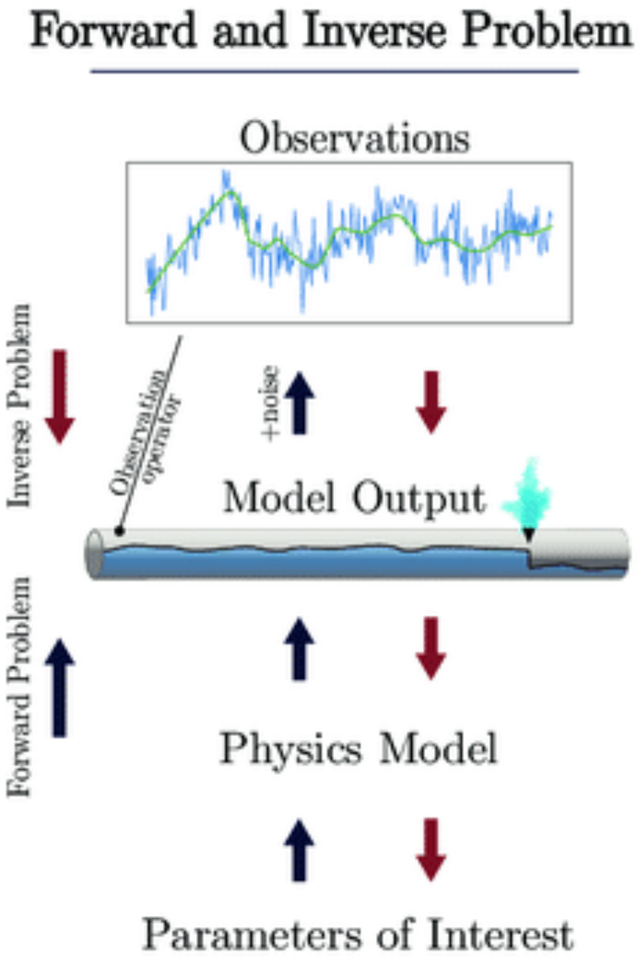
Given λ what is u(t; x) (Inference, filtering and smoothing or simply

data-driven solutions of PDEs)

Forward Problem

Find λ that best describes observations u (ti; xj) (Learning, system

identification, or data-driven discovery of PDEs)



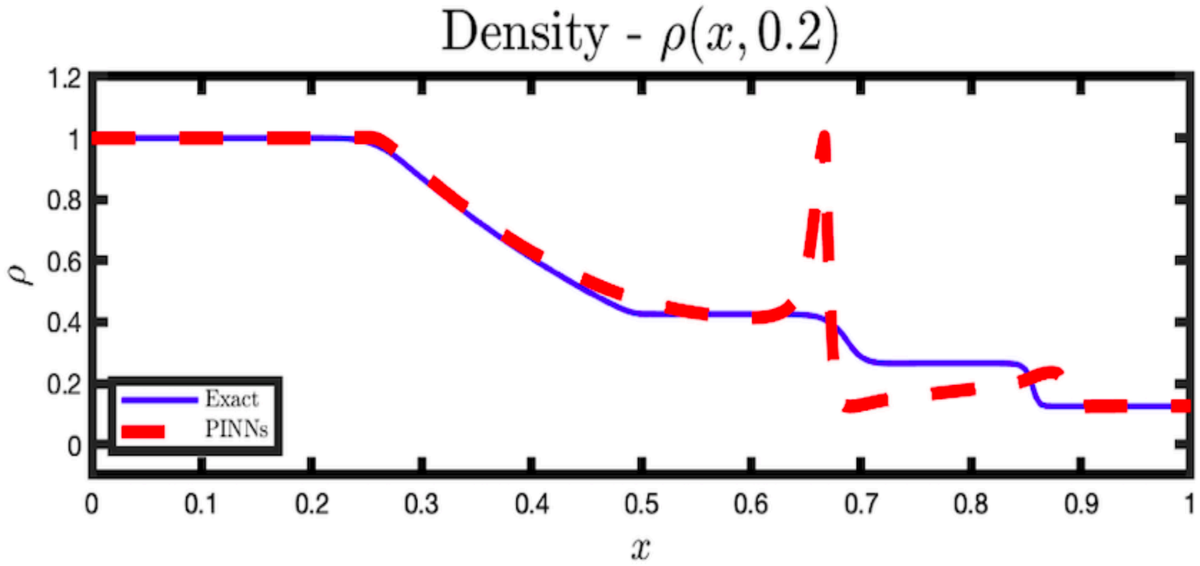
Source: Mücke, Nikolaj & Sanderse, Benjamin & Bohté, Sander & Oosterlee, Cornelis. (2021). Markov Chain Generative Adversarial Neural Networks for Solving Bayesian Inverse Problems in Physics Applications.

Want to

solve **hyperbolic PDEs** or hyperbolic conservation laws

Whose

Solutions are often **discontinuous** & **non-differentiable**



Solution to Sod Shock-Tube Problem

But

PINNs struggle to approximate discontinuous solutions of PDEs.

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} + \frac{\partial \boldsymbol{G}}{\partial y} = \mathbf{0}, \quad (x, y, t) \in \Omega \times (0, T]$$

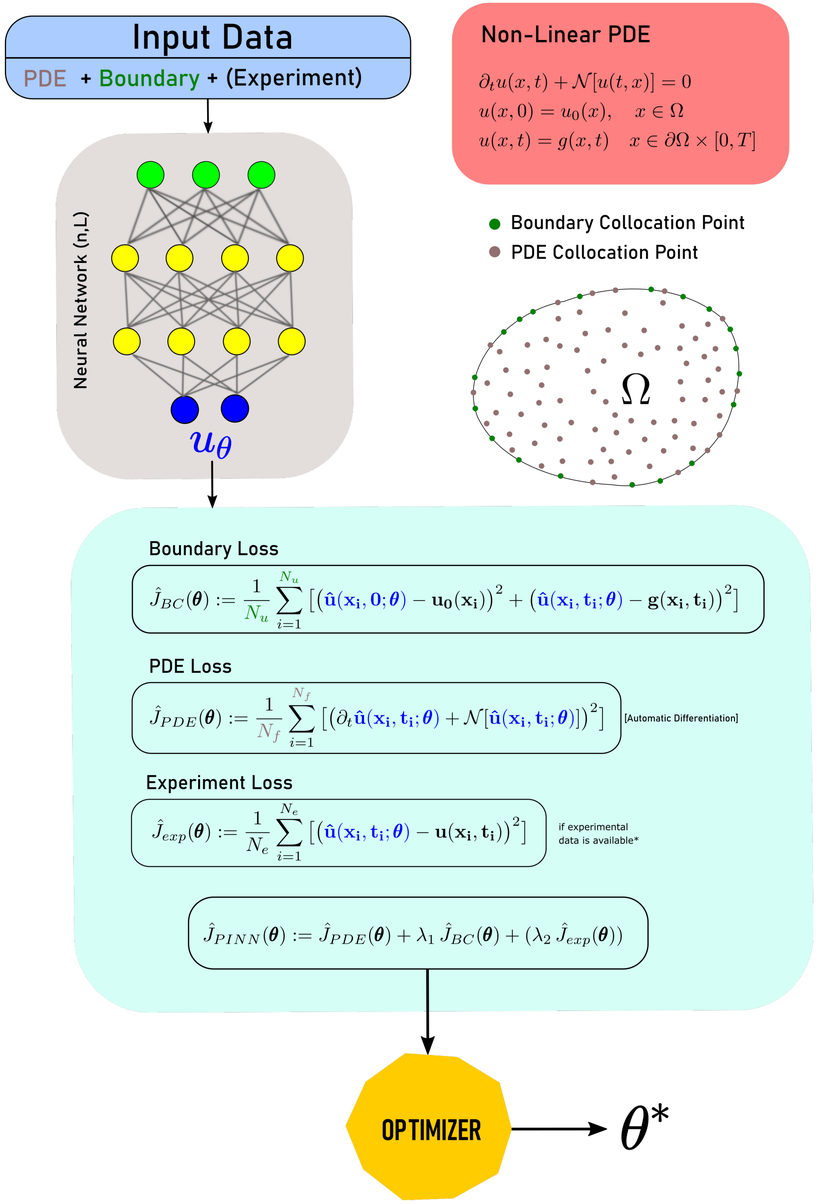
Hyperbolic PDEs

Because

Deep Learning frameworks use **Automatic Differentiation**

Which

differentiate the neural network with respect to the governing PDE, may not consistently resolve discontinuities that arise in solutions to hyperbolic conservation laws.



PDE Loss is problematic since it uses automatic differentiation

Solution?

Use central schemes (Finite Volume Methods) with Flux limiting

To approximate the PDE and then use this **DIFF Loss**

$$\frac{d\bar{\boldsymbol{U}}}{dt} + \frac{1}{v_i}\oint_{S_i}\boldsymbol{f}(\boldsymbol{U})\cdot d\boldsymbol{n}\, dS = \mathbf{0}$$

$$\frac{\partial\bar{\boldsymbol{U}}}{\partial t} + \frac{1}{\Delta x}\left(\boldsymbol{F}_{j+1/2} - \boldsymbol{F}_{j-1/2}\right) = \mathbf{0}$$

Finite Volume Methods with Flux Limiting

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} + \frac{1}{\Delta x_i}\left[F^*_{i+1/2} - F^*_{i-1/2}\right] = 0.$$

The numerical fluxes $F^*_{i\pm 1/2}$ correspond to a nonlinear combination of first and second-order approximations to the continuous flux function.

The symbols $u^*_{i+1/2}$ and $u^*_{i-1/2}$ represent scheme dependent functions (of the limited extrapolated cell edge variables), *i.e.*,

$$u^*_{i+1/2} = u^*_{i+1/2}\left(u^L_{i+1/2}, u^R_{i+1/2}\right), u^*_{i-1/2} = u^*_{i-1/2}\left(u^L_{i-1/2}, u^R_{i-1/2}\right),$$

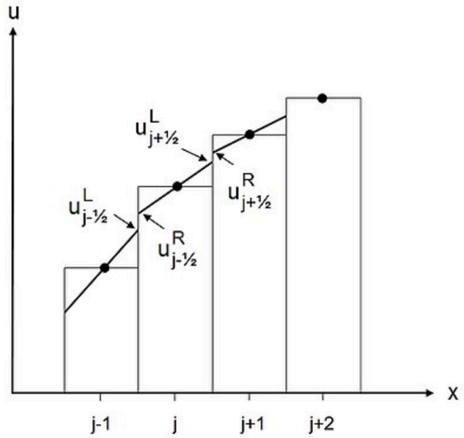
where, using downwind slopes:

$$u^L_{i+1/2} = u_i + 0.5\phi\left(r_i\right)\left(u_{i+1} - u_i\right), u^R_{i+1/2} = u_{i+1} - 0.5\phi\left(r_{i+1}\right)\left(u_{i+2} - u_{i+1}\right),$$

$$u^L_{i-1/2} = u_{i-1} + 0.5\phi\left(r_{i-1}\right)\left(u_i - u_{i-1}\right), u^R_{i-1/2} = u_i - 0.5\phi\left(r_i\right)\left(u_{i+1} - u_i\right),$$

and

$$r_i = \frac{u_i - u_{i-1}}{u_{i+1} - u_i}.$$



Piecewise linear extrapolation

An example of MUSCL type left and right state linear-extrapolation.

Source: https://en.m.wikipedia.org/wiki/MUSCL_scheme

Kurganov and Tadmor central scheme

$$F^*_{i-\frac{1}{2}} = \frac{1}{2}\left\{\left[F\left(u^R_{i-\frac{1}{2}}\right) + F\left(u^L_{i-\frac{1}{2}}\right)\right] - a_{i-\frac{1}{2}}\left[u^R_{i-\frac{1}{2}} - u^L_{i-\frac{1}{2}}\right]\right\}.$$

$$F^*_{i+\frac{1}{2}} = \frac{1}{2}\left\{\left[F\left(u^R_{i+\frac{1}{2}}\right) + F\left(u^L_{i+\frac{1}{2}}\right)\right] - a_{i+\frac{1}{2}}\left[u^R_{i+\frac{1}{2}} - u^L_{i+\frac{1}{2}}\right]\right\}.$$

Where the *local propagation speed*, $a_{i\pm\frac{1}{2}}$, is the maximum absolute value of the eigenvalue of the Jacobian of $F\left(u\left(x,t\right)\right)$ over cells $i, i\pm 1$ given by

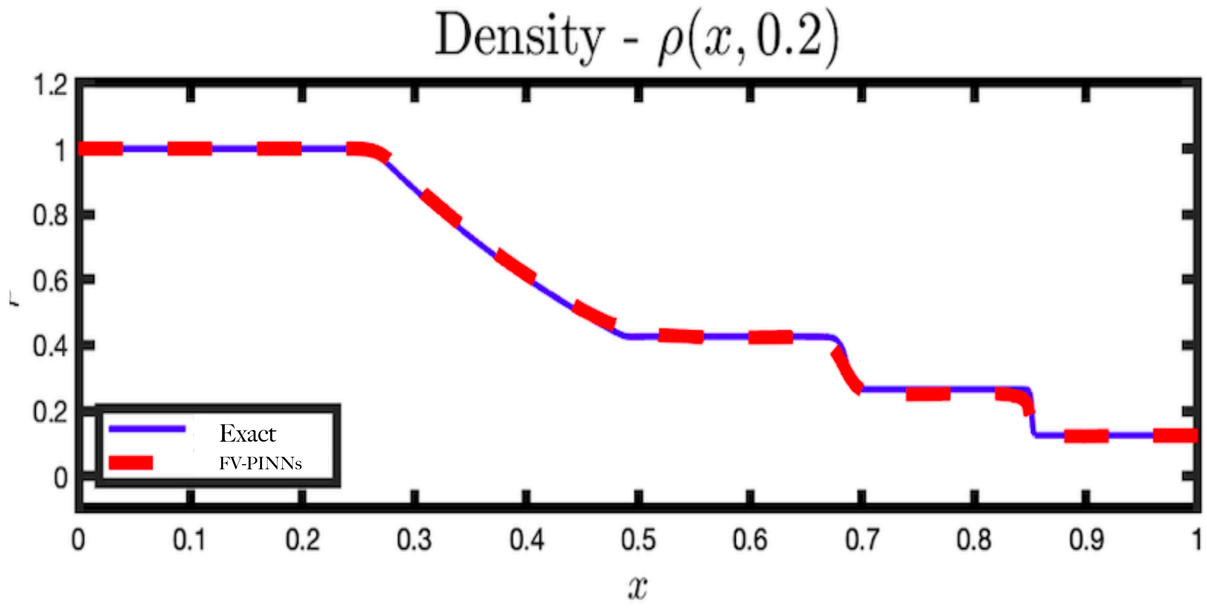
$$a_{i+\frac{1}{2}}\left(t\right) = \max\left[\rho\left(\frac{\partial F\left(u^L_{i+1/2}\left(t\right)\right)}{\partial u}\right), \rho\left(\frac{\partial F\left(u^R_{i+1/2}\left(t\right)\right)}{\partial u}\right),\right]$$

and $\rho\left(\frac{\partial F\left(u\left(t\right)\right)}{\partial u}\right)$ represents the spectral radius of $\frac{\partial F\left(u\left(t\right)\right)}{\partial u}$.

Finite Volume Physics Informed Neural Networks

$$DIFF\_loss = \sum_{i,j} G(\theta)^2, \quad G(\theta) = U_{i,j}^{n+1}(\theta) - \left(U_{i,j}^{n}(\theta) + \frac{\Delta t}{\Delta x} R\left(F_{i+1/2}^{n}, F_{i-1/2}^{n} \right) \right)$$

“DIFF loss”



Solution to Sod Shock-Tube Problem with FV-PINNs

Algorithm

Algorithm 1 FV-PINNS Algorithm with Rusanov Flux Solver

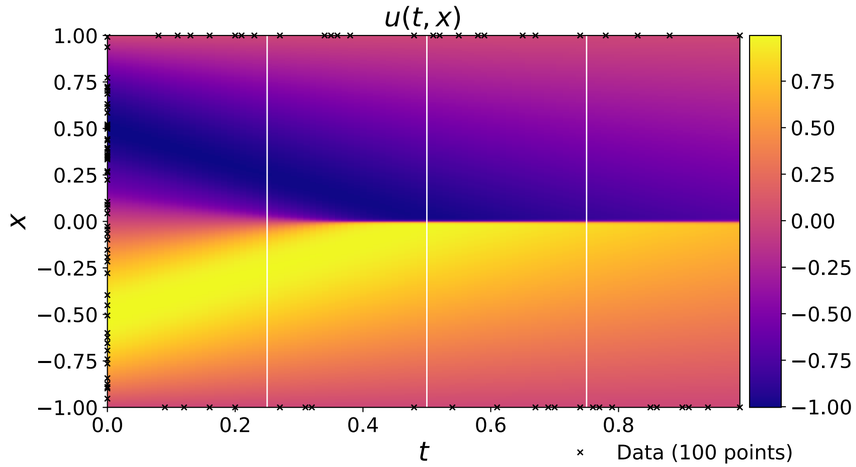
1: Generate weights $\theta \in \mathbb{R}^k$ and a deep neural network (DNN), $U(\boldsymbol{x}_i, \boldsymbol{y}_j, t_n, \theta) = U_{i,j}^n$, where $(\boldsymbol{x}_i, \boldsymbol{y}_j, t_n)$ are inputs to the network, and $U_{i,j}^n = [\rho_{i,j}^n, u_{i,j}^n, v_{i,j}^n, p_{i,j}^n]$ are the outputs.
2: Minimize $\|U^0 - \hat{U}^0\|^2$ via stochastic gradient descent.
3: **while** $t < T$ **do**
4: $\quad G(\theta) = U_{i,j}^{n+1} - \left(U_{i,j}^{n} + \frac{\Delta t}{\Delta x} R\left(F_{i+1/2}^{n}, F_{i-1/2}^{n} \right) \right)$
5: $\quad$ Update θ by performing stochastic gradient descent: $\theta = \theta - \eta \nabla_\theta G(\theta)$, where η is the learning rate.
6: $\quad$ Employ boundary conditions.
7: $\quad t_{n+1} = t_n + \Delta t$
8: **end while**

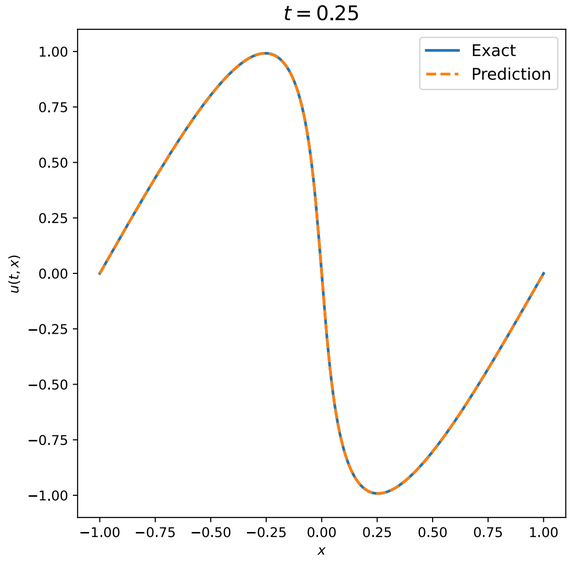
Results

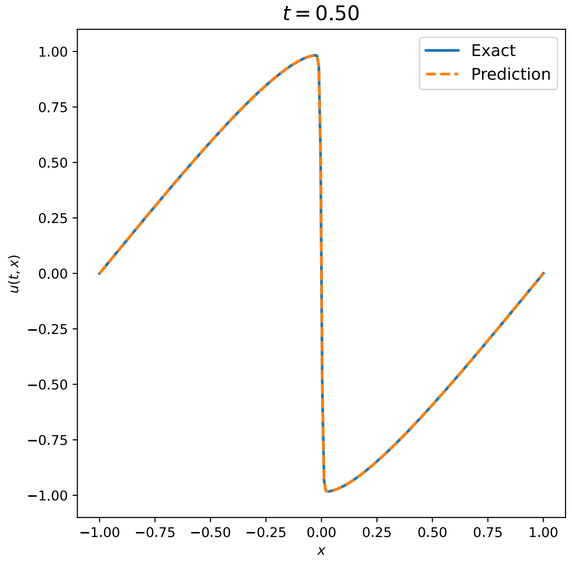
Burger's Equation

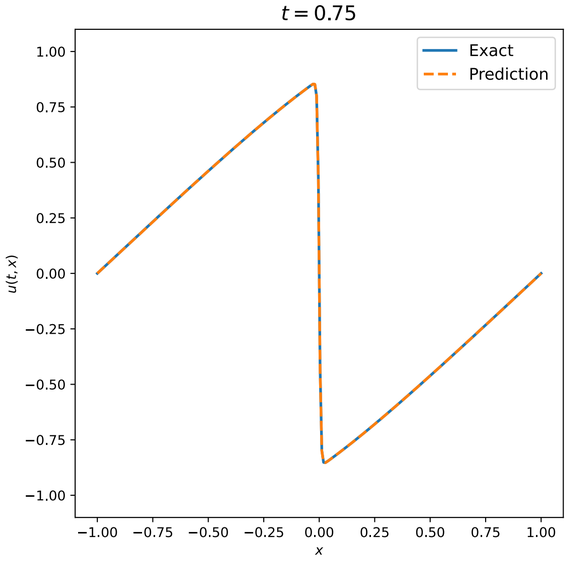
$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$

Forward Problem





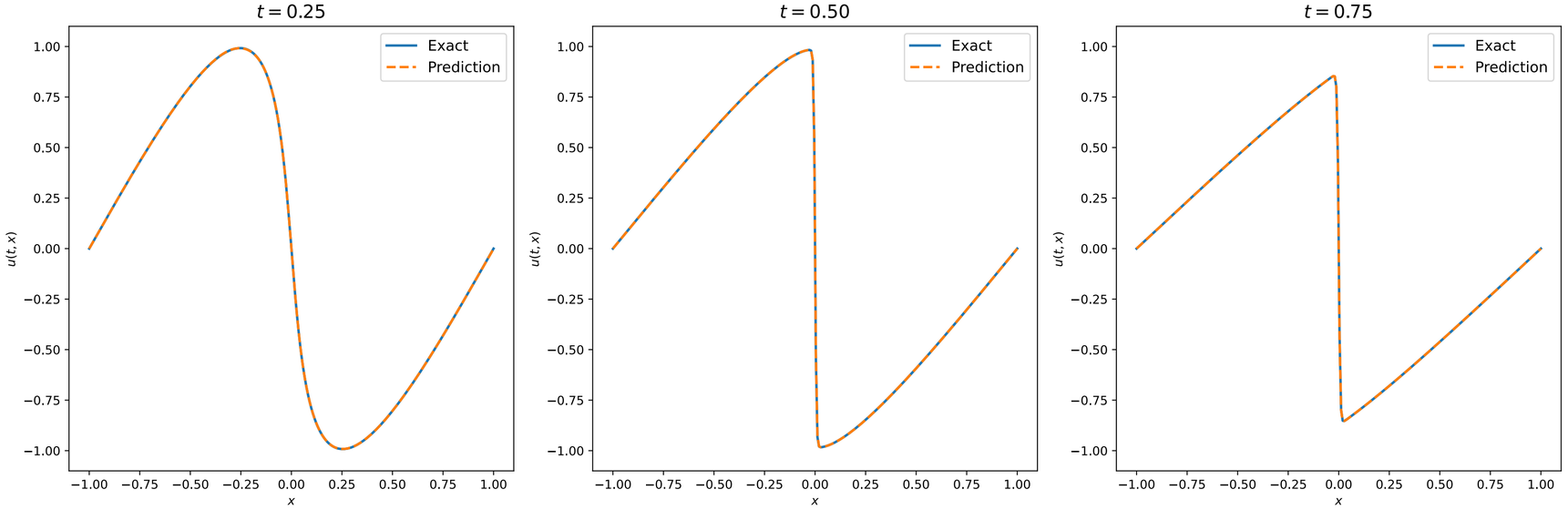


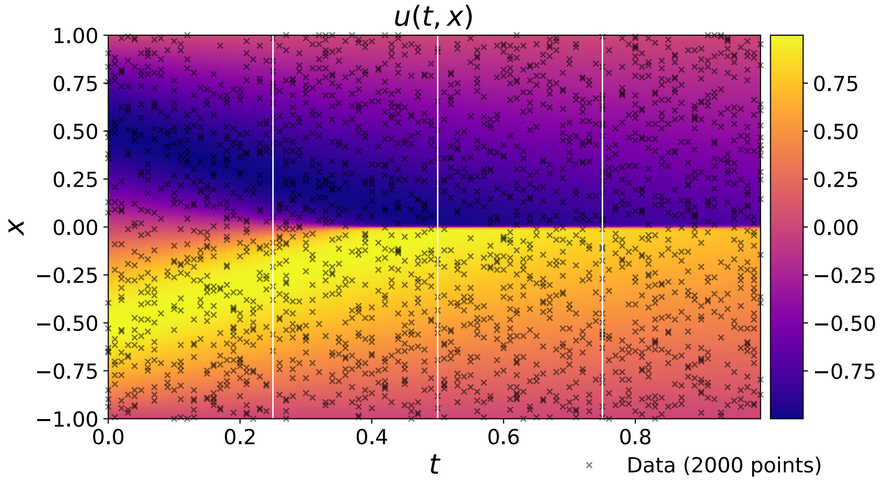


Burger's Equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$

Inverse Problem





Correct PDE	$u_t + uu_x - 0.0031831u_{xx} = 0$
Identified PDE (clean data)	$u_t + 0.99915uu_x - 0.0031794u_{xx} = 0$
Identified PDE (1% noise)	$u_t + 1.00042uu_x - 0.0032098u_{xx} = 0$

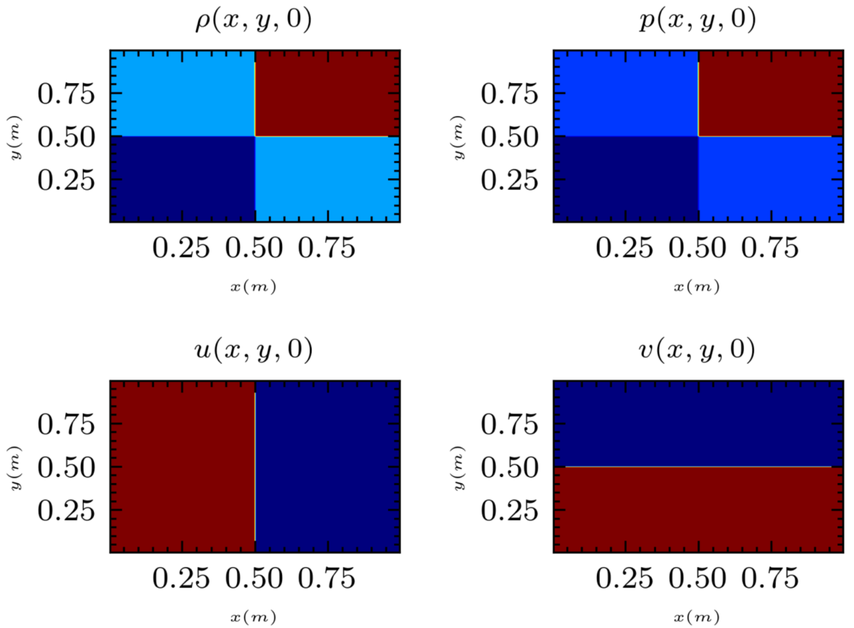
Euler Gas Equation (1D & 2D)

1D Sod Shock-Tube

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} = \boldsymbol{0}, \quad (x,t) \in \Omega \times (0,T]$$

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ (E+p)\,u \end{bmatrix},$$

2D Liksa Shock-Tube

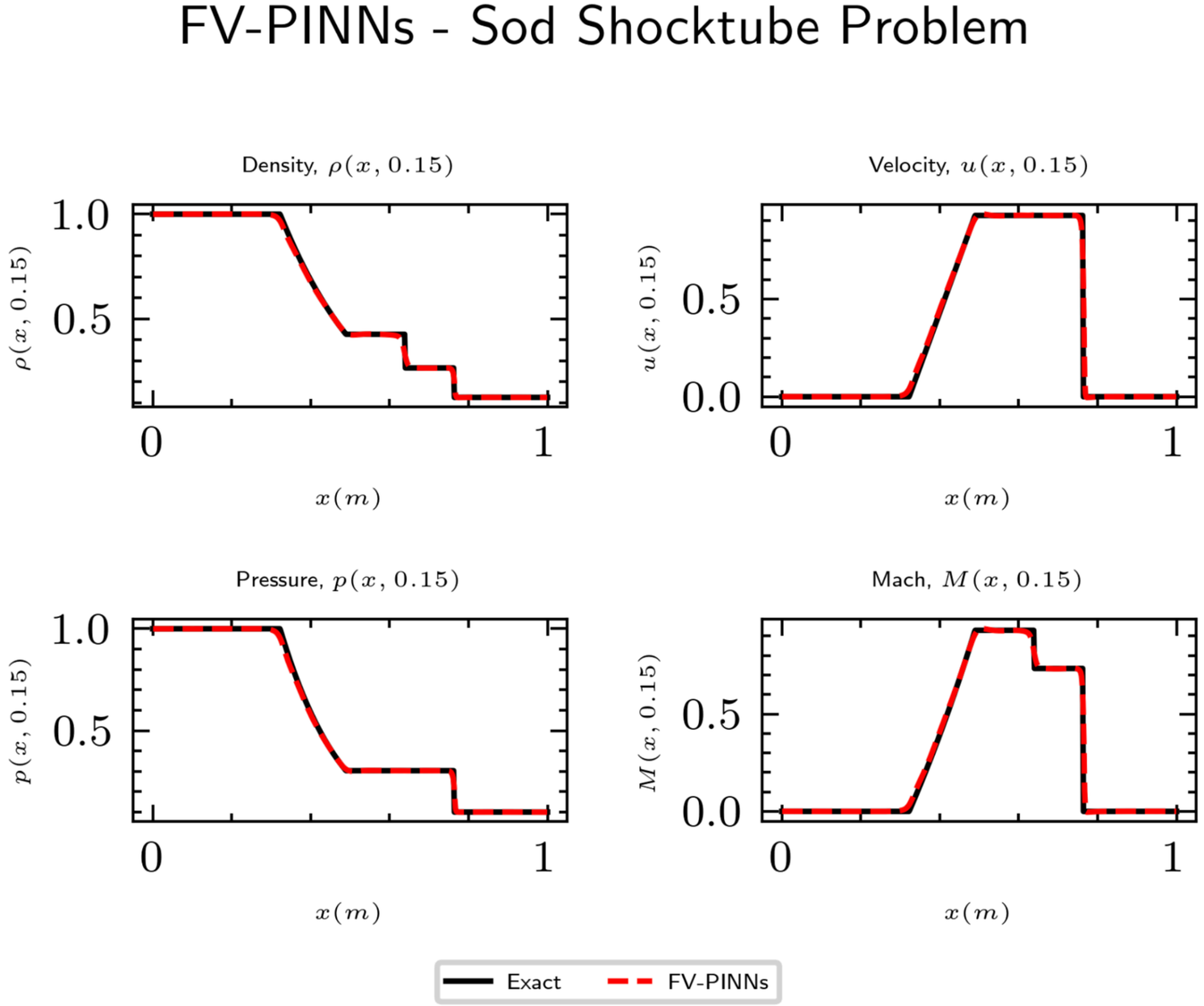


$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} + \frac{\partial \boldsymbol{G}}{\partial y} = \boldsymbol{0}, \quad (x,y,t) \in \Omega \times (0,T]$$

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (E+p)u \end{bmatrix}, \quad \boldsymbol{G} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (E+p)v \end{bmatrix}$$

Euler Gas Equation (1D)

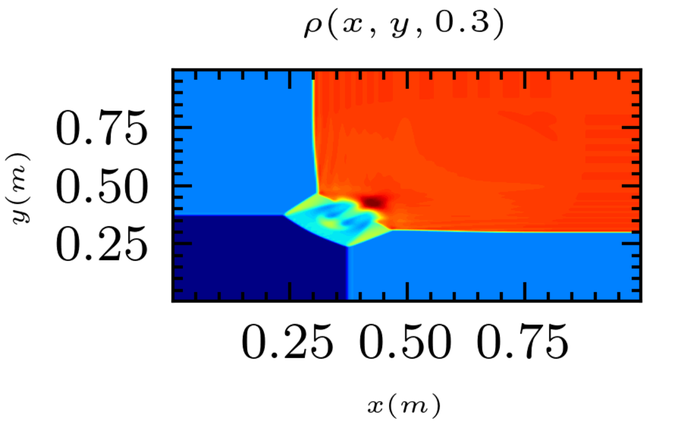
Results

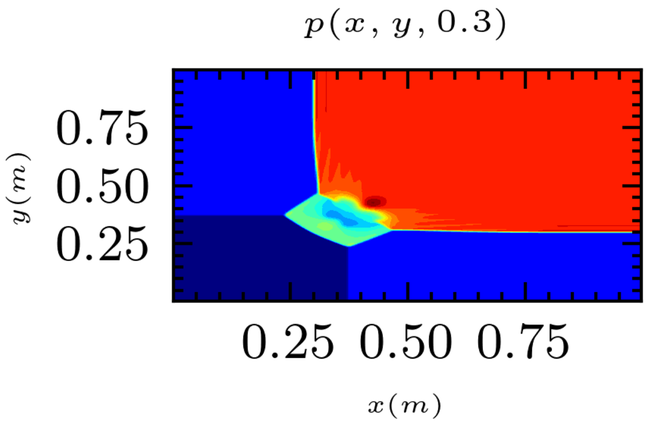


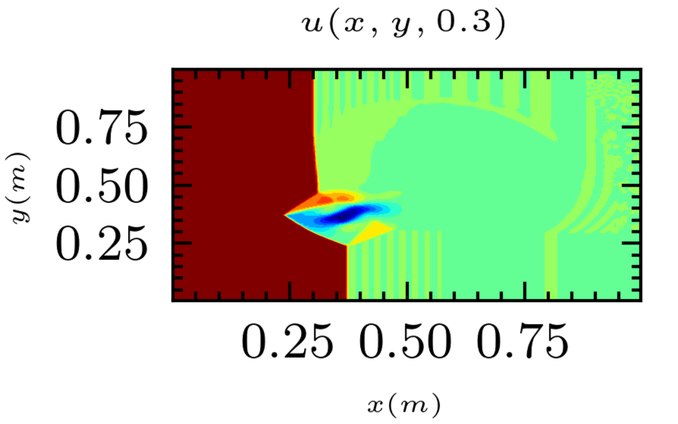
Euler Gas Equation (2D)

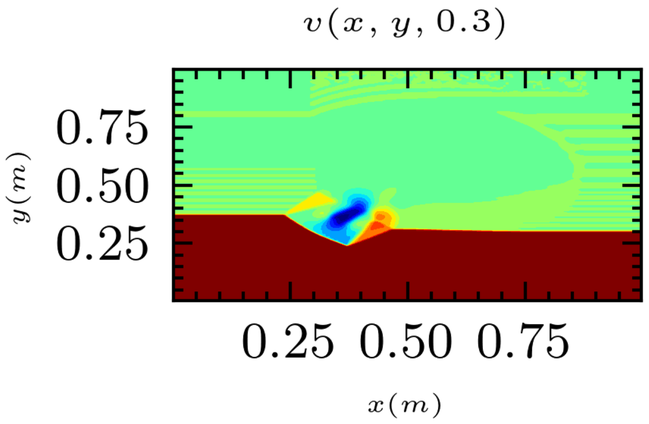
Results

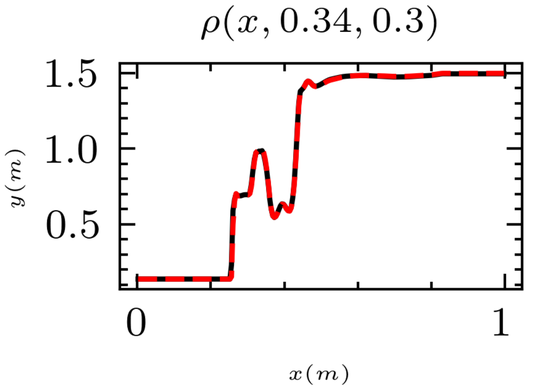
FV-PINNs - 2D Shocktube (Liska, 2003)

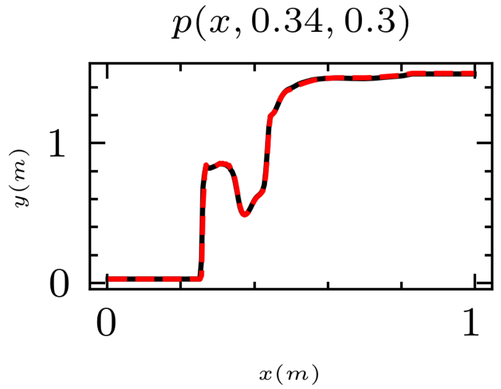


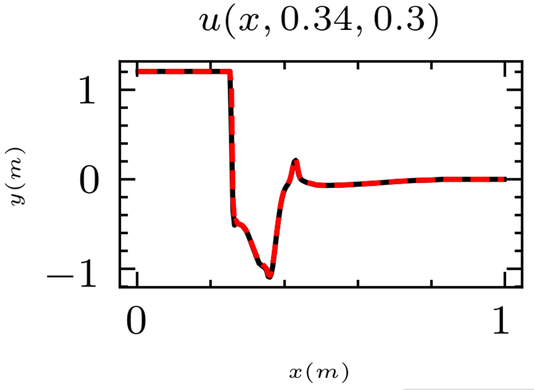


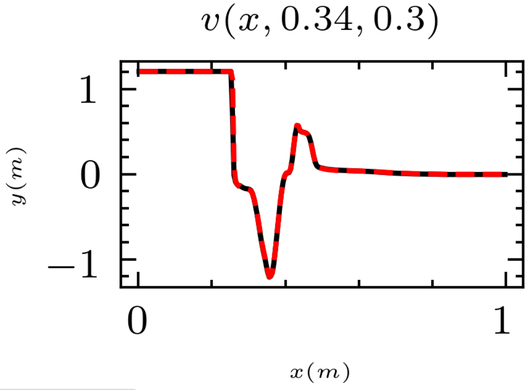












Thank you for Listening!!!