Physics-informed neural networks

Raissi et al (2019)

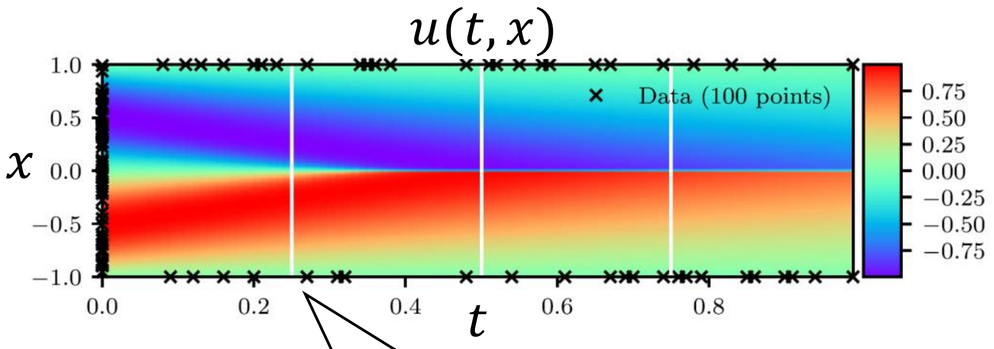
E.g., Burgers’ equation (inference)

Problem statement

$$u_t + u u_x - (0.01/\pi) u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$

$$IC: \; u(0, x) = -\sin(\pi x),$$

$$BC: \; u(t, -1) = u(t, 1) = 0.$$



Why are data of u sampled at different t between the upper and lower boundaries?

Training data (from ground truth):

$\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ $\qquad N_u = 100$

Collocation points:

$\{t_f^i, x_f^i\}_{i=1}^{N_f}$ $\qquad N_f = 10{,}000$

Physics equations:

$$f := u_t + u u_x - (0.01/\pi) u_{xx}$$

Loss function:

Data loss

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

Data points

Equation loss

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Collocation points

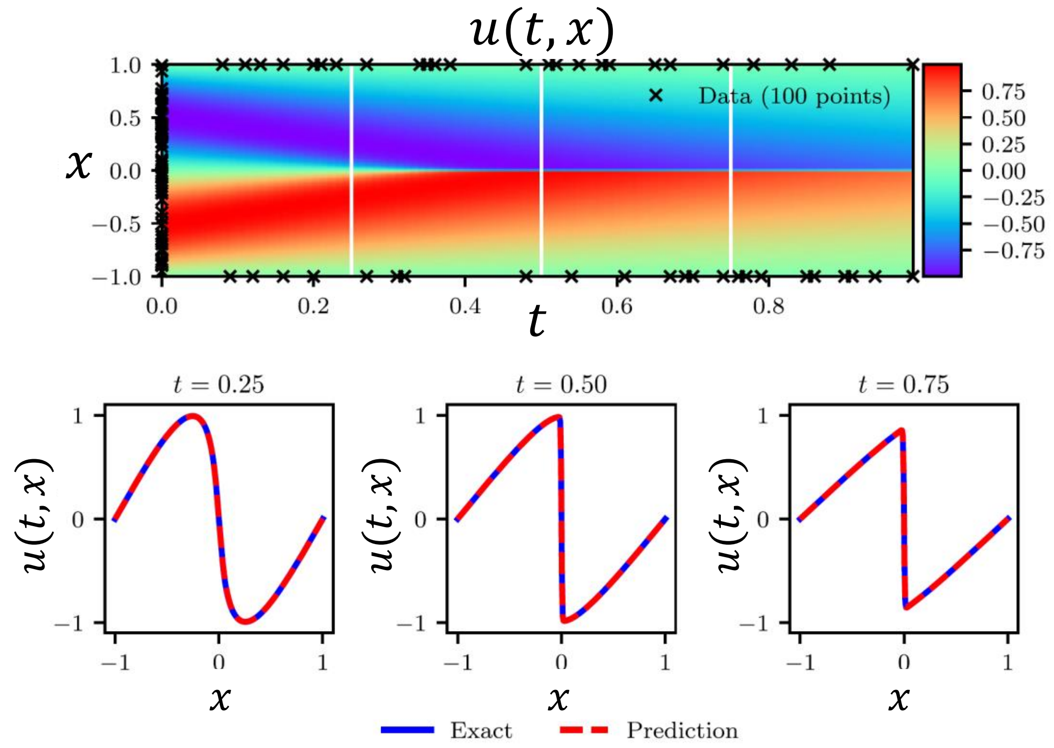
E.g., Burgers' equation (inference)

Problem statement

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$

$$IC: \; u(0, x) = -\sin(\pi x),$$

$$BC: \; u(t, -1) = u(t, 1) = 0.$$



Training data (from ground truth):

$$\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u} \qquad N_u = 100$$

Collocation points:

$$\{t_f^i, x_f^i\}_{i=1}^{N_f} \qquad N_f = 10{,}000$$

Physics equations:

$$f := u_t + uu_x - (0.01/\pi)u_{xx}$$

Loss function:

Data loss

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

Data points

Equation loss

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Collocation points

E.g., Burgers' equation (inference)

Problem statement

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$

$$IC: \; u(0, x) = -\sin(\pi x),$$

$$BC: \; u(t, -1) = u(t, 1) = 0.$$

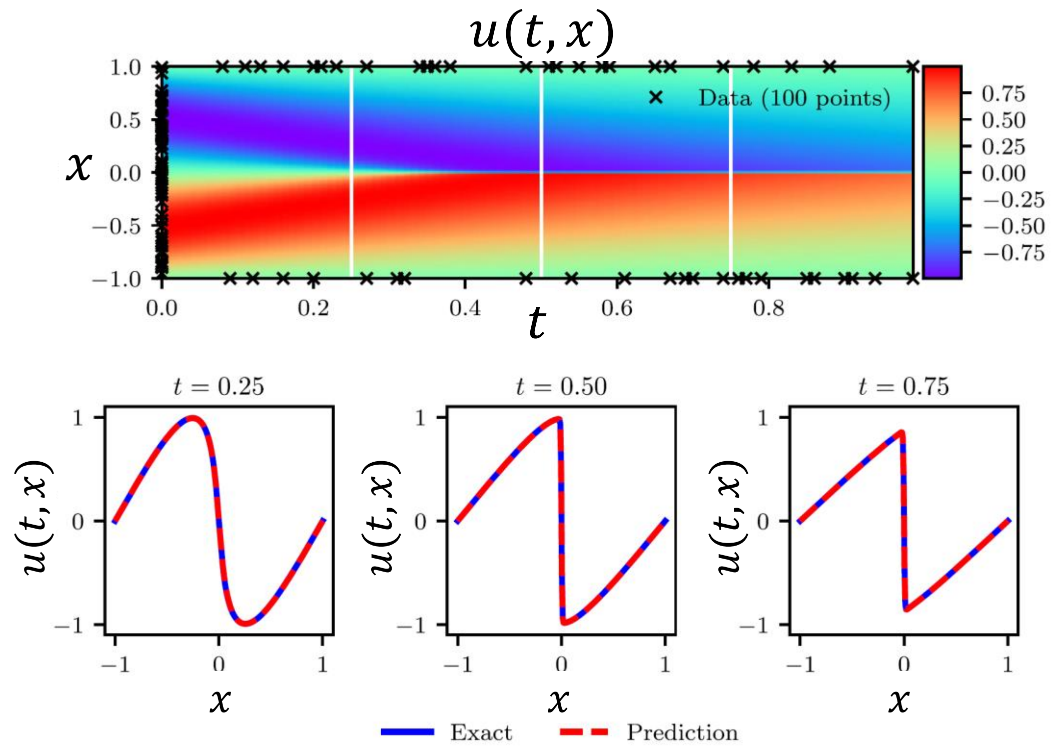


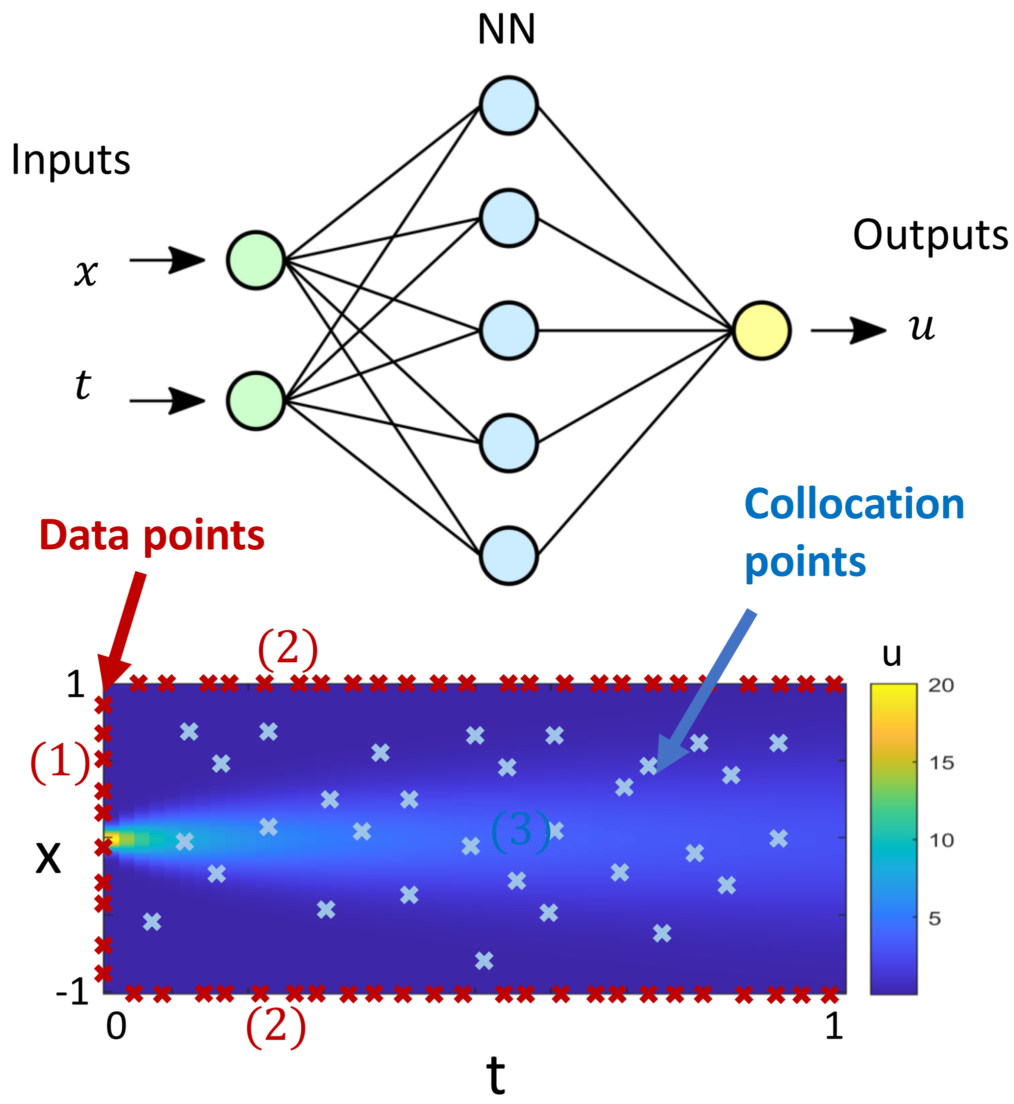
Table A.1
Burgers' equation: Relative $\mathbb{L}_2$ error between the predicted and the exact solution $u(t, x)$ for different number of initial and boundary training data N_u, and different number of collocation points N_f. Here, the network architecture is fixed to 9 layers with 20 neurons per hidden layer.

N_u \ N_f	2000	4000	6000	7000	8000	10000
20	2.9e−01	4.4e−01	8.9e−01	1.2e+00	9.9e−02	4.2e−02
40	6.5e−02	1.1e−02	5.0e−01	9.6e−03	4.6e−01	7.5e−02
60	3.6e−01	1.2e−02	1.7e−01	5.9e−03	1.9e−03	8.2e−03
80	5.5e−03	1.0e−03	3.2e−03	7.8e−03	4.9e−02	4.5e−03
100	6.6e−02	2.7e−01	7.2e−03	6.8e−04	2.2e−03	6.7e−04
200	1.5e−01	2.3e−03	8.2e−04	8.9e−04	6.1e−04	4.9e−04

Table A.2
Burgers' equation: Relative $\mathbb{L}_2$ error between the predicted and the exact solution $u(t, x)$ for different number of hidden layers and different number of neurons per layer. Here, the total number of training and collocation points is fixed to $N_u = 100$ and $N_f = 10{,}000$, respectively.

Layers \ Neurons	10	20	40
2	7.4e−02	5.3e−02	1.0e−01
4	3.0e−03	9.4e−04	6.4e−04
6	9.6e−03	1.3e−03	6.1e−04
8	2.5e−03	9.6e−04	5.6e−04

Physics-informed NN



Given a partial differential equation of a general form:

$$u_t + \mathcal{N}[u] = 0,\ x \in \Omega,\ t \in [0, T]$$

where $\mathcal{N}[\cdot]$ is a nonlinear differential operator.
Define equation residue f as

$$f := u_t + \mathcal{N}[u]$$

Cost function: (MSE: mean squared error)

$$MSE = \underbrace{MSE_u}_{\text{Data loss}} + \underbrace{MSE_f}_{\text{Equation loss}},$$

Data loss

$$\frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2,$$

Data points

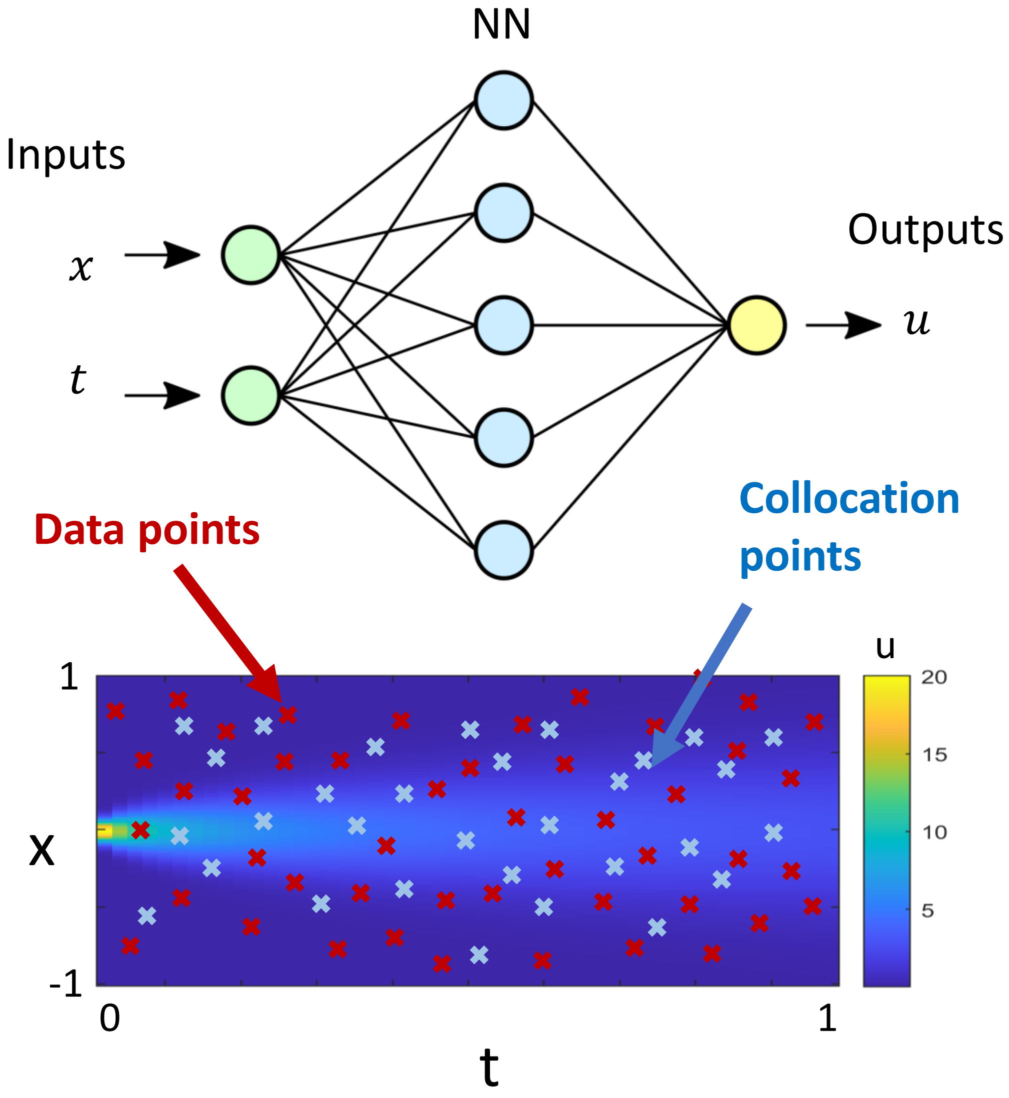
Equation loss

$$\frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Collocation points

Raissi et al (2019)

Data-driven prediction of solution



Given a partial differential equation of a general form:

$$u_t + \mathcal{N}[u] = 0,\ x \in \Omega,\ t \in [0, T]$$

where $\mathcal{N}[\cdot]$ is a nonlinear differential operator.
Define equation residue f as

$$f := u_t + \mathcal{N}[u]$$

Cost function: (MSE: mean squared error)

$$MSE = \underbrace{MSE_u}_{\text{Data loss}} + \underbrace{MSE_f}_{\text{Equation loss}},$$

Data loss

$$\frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2,$$

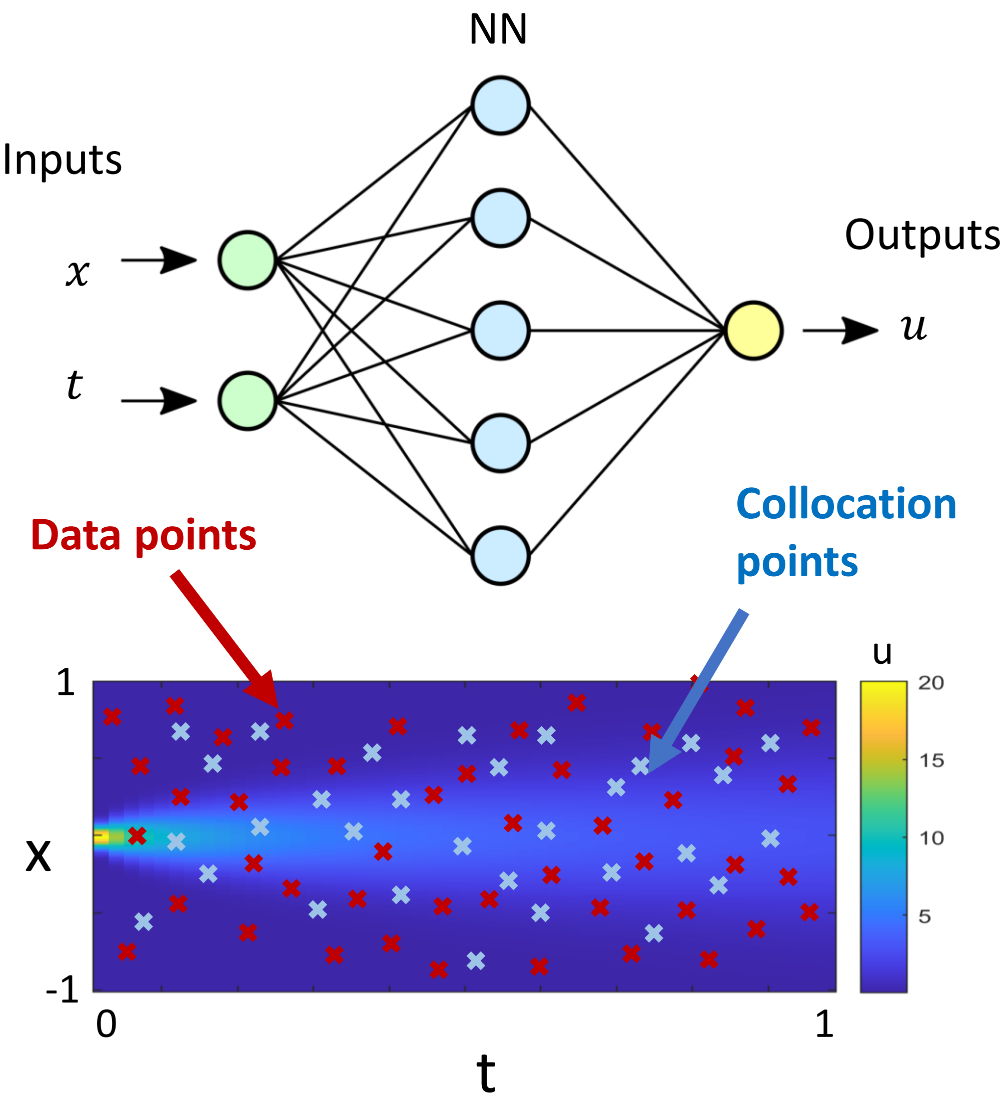
Data points

Equation loss

$$\frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Collocation points

Data-driven discovery of unknown parameters



Given a partial differential equation of a general form:

$$u_t + \mathcal{N}[u;\lambda] = 0,\ x \in \Omega,\ t \in [0, T]$$

where $\mathcal{N}[\cdot]$ is a nonlinear differential operator.
Define equation residue f as

$$f := u_t + \mathcal{N}[u;\lambda]$$

What are the parameters λ that best describe the data?

Cost function: (MSE: mean squared error)

$$MSE = \underbrace{MSE_u}_{\text{Data loss}} + \underbrace{MSE_f}_{\text{Equation loss}},$$

Data loss

$$\frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

Data points

Equation loss

$$\frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Collocation points

PINN

$$\frac{\partial u}{\partial t} + N(u, \lambda) = 0, \qquad x \in [-L, L], \qquad t \in [0\ T]$$

- Application 1: Prediction of solution for a **well-posed problem** (this is what a traditional numerical solver can do)

  Given an eqn + BC + IC and parameters λ, what's the model prediction?

- Application 2: Prediction of solution when data is available within the domain but not at the IC, BC
  (difficult for a traditional numerical solver!)

  Given an eqn and parameters λ, what's the model prediction best describes the data?

- Application 3: Data-driven discovery of **unknown parameters** (difficult for a traditional numerical solver!)

  What are the parameters λ that best describe the data and the eqn?

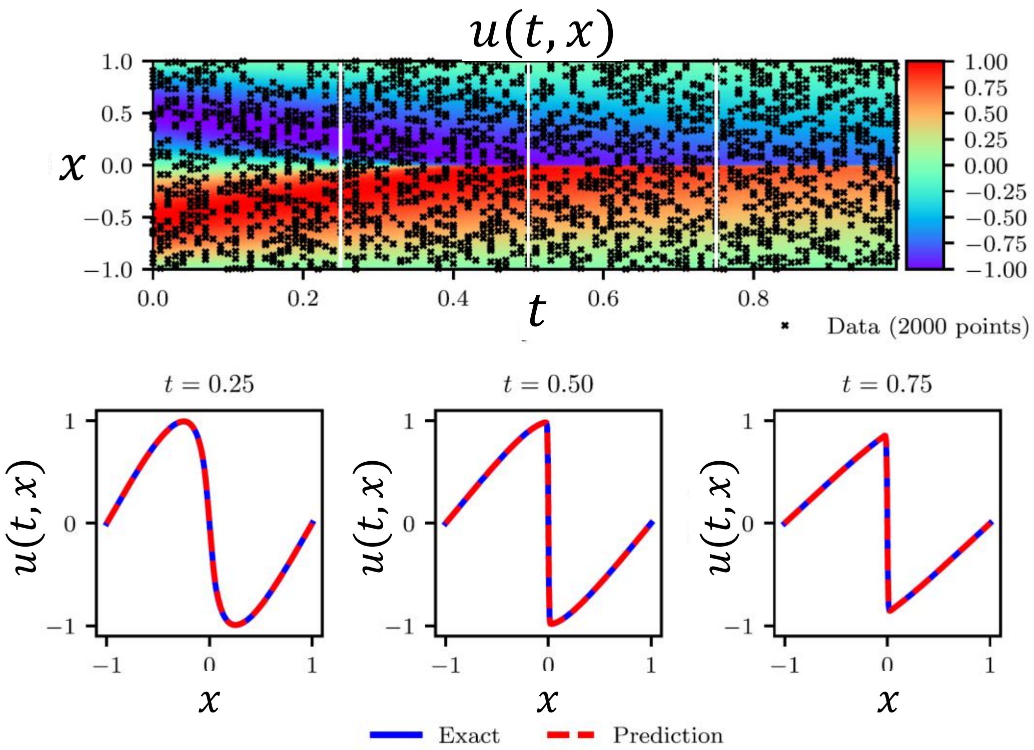
Raissi et al (2019)

E.g., Burgers' equation (identification)

Given training data of u, t, x find λ_1, λ_2

$$u_t + \lambda_1 u u_x - \lambda_2 u_{xx} = 0 \qquad x \in [-1, 1], \quad t \in [0, 1],$$

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)	$u_t + 1.00042uu_x - 0.0032098u_{xx} = 0$



Training data (from ground truth):

$\{t_u^i, x_u^i, u^i\}_{i=1}^{N}$ $\qquad N = 2{,}000$

Collocation points:

$\{t_u^i, x_u^i\}_{i=1}^{N}$ $\qquad N = 2{,}000$

Physics equations:

$$f := u_t + \lambda_1 u u_x - \lambda_2 u_{xx}$$

Loss function:

Data loss

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

Data points

Equation loss

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Collocation points

Raissi et al (2019)

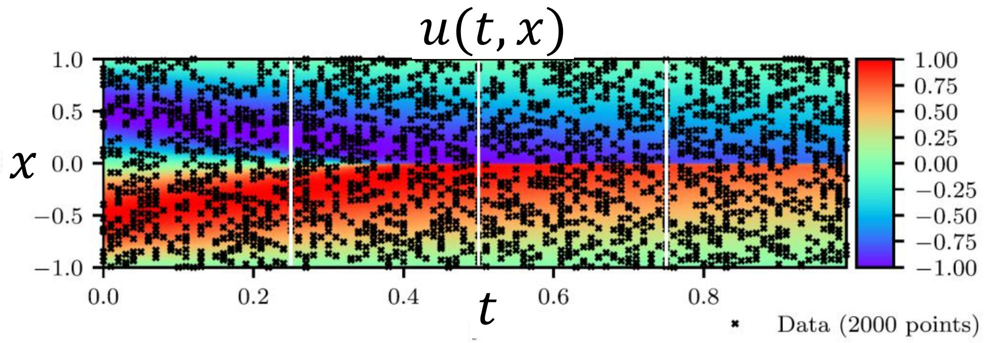
E.g., Burgers' equation (identification)

Given training data of u, t, x find λ_1, λ_2

What about noisy data?

$$u_t + \lambda_1 u u_x - \lambda_2 u_{xx} = 0 \qquad x \in [-1, 1], \quad t \in [0, 1],$$

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)	$u_t + 1.00042uu_x - 0.0032098u_{xx} = 0$



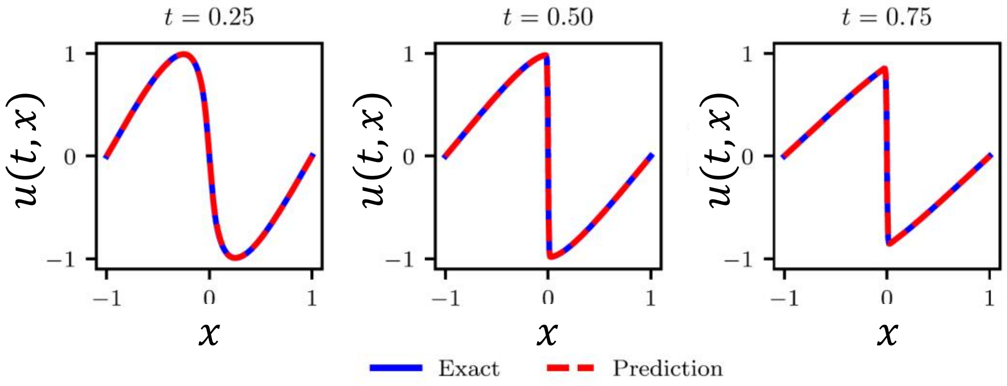


Table B.7
Burgers' equation: Percentage error in the identified parameters λ_1 and λ_2 for different number of hidden layers and neurons per layer. Here, the training data is considered to be noise-free and fixed to $N = 2{,}000$.

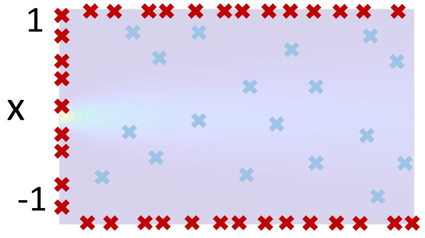
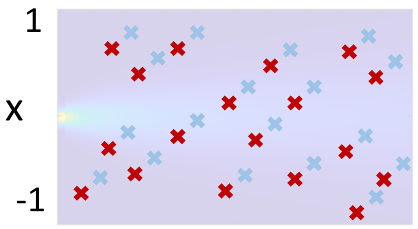
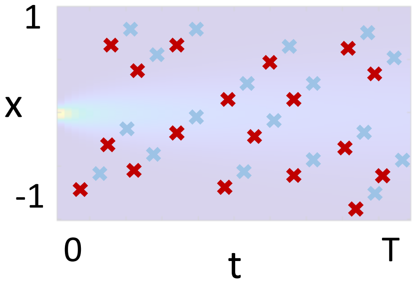
Layers \ Neurons	% error in λ_1: 10	20	40	% error in λ_2: 10	20	40
2	11.696	2.837	1.679	103.919	67.055	49.186
4	0.332	0.109	0.428	4.721	1.234	6.170
6	0.668	0.629	0.118	3.144	3.123	1.158
8	0.414	0.141	0.266	8.459	1.902	1.552

Table B.6
Burgers' equation: Percentage error in the identified parameters λ_1 and λ_2 for different number of training data N corrupted by different noise levels. Here, the neural network architecture is kept fixed to 9 layers and 20 neurons per layer.

N_u \ Noise	% error in λ_1: 0%	1%	5%	10%	% error in λ_2: 0%	1%	5%	10%
500	0.131	0.518	0.118	1.319	13.885	0.483	1.708	4.058
1000	0.186	0.533	0.157	1.869	3.719	8.262	3.481	14.544
1500	0.432	0.033	0.706	0.725	3.093	1.423	0.502	3.156
2000	0.096	0.039	0.190	0.101	0.469	0.008	6.216	6.391

PINN

$$\frac{\partial u}{\partial t} + N(u, \lambda) = 0, \qquad x \in [-1, 1], \qquad t \in [0\ T]$$

Applications	Training data	NN prediction	
1. Prediction of solution of a **well-posed problem** (this is what a traditional numerical solver can do)	$IC, (BC)$	$u(x,t)$	
2. Prediction of solution when data is available within the domain but not at the IC, BC (difficult for a traditional numerical solver)	t_i, x_i, u_i from $i = 0$ to m $x_i \in [-1, 1], t_i \in [0\ T]$	$u(x,t)$	
3. Data-driven discovery of **unknown parameters** (difficult for a traditional numerical solver)	t_i, x_i, u_i from $i = 0$ to m $x_i \in [-1, 1], t_i \in [0\ T]$	$u(x,t)$ λ	

Burgers' equation coding exercise

- TF2/TF1.14
- Activation function: tanh or sin, cos?
- In the inference problem, change the initial and boundary training data (N=100) to data randomly selected within the {t,x} domain, keep the collocation points (Nf=10,000). How does that affect the performance of the prediction?

Burgers’ equation coding exercise

Ex1: Sin activation, Iter: 4800

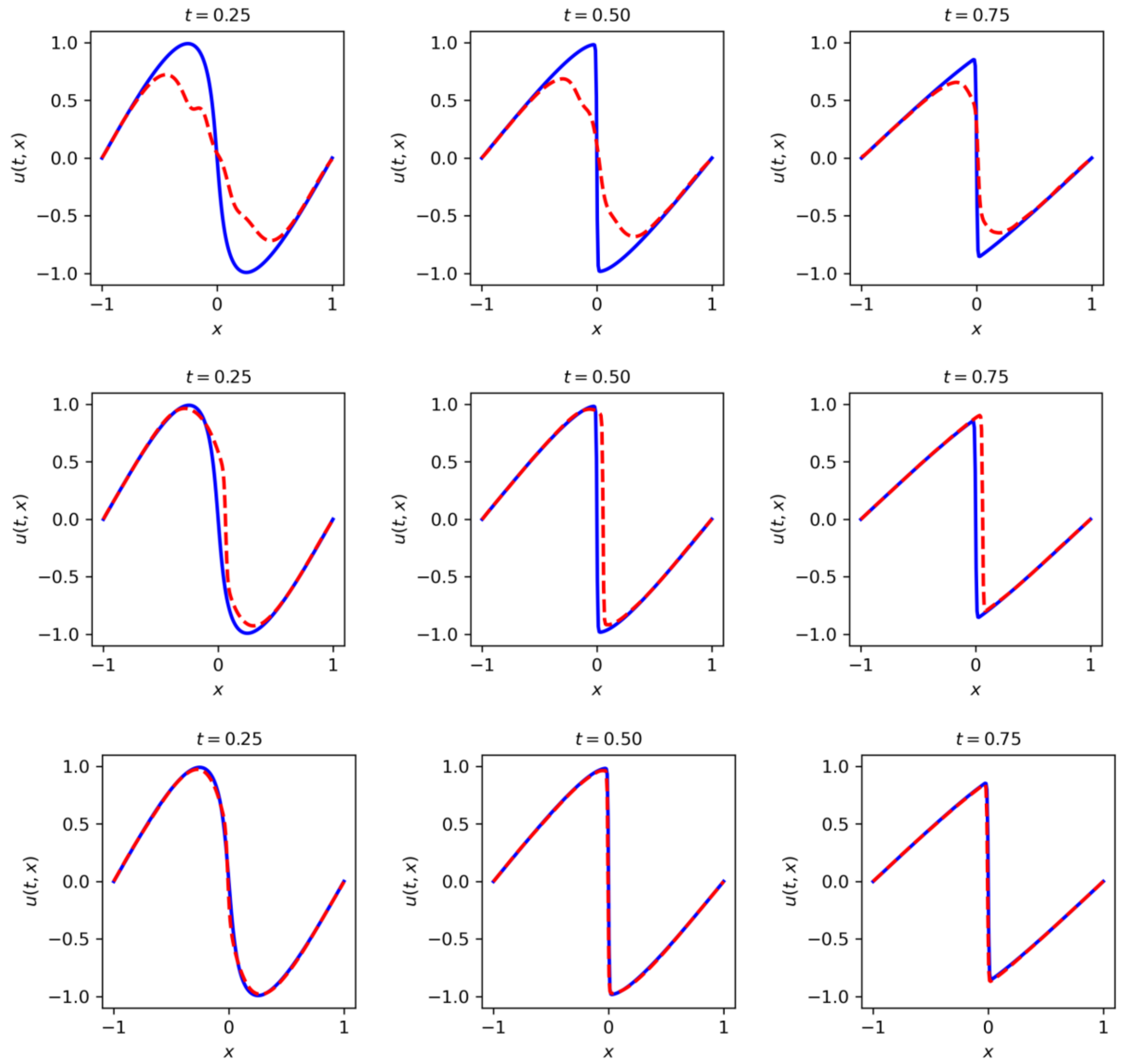
loss: 0.0319006
Training time: 173.6166 Error u: 3.554990e-01

Ex2: Sin activation, Iter: 24680

loss: 0.0029452811
Training time: 1126.4058 Error u: 3.946169e-01

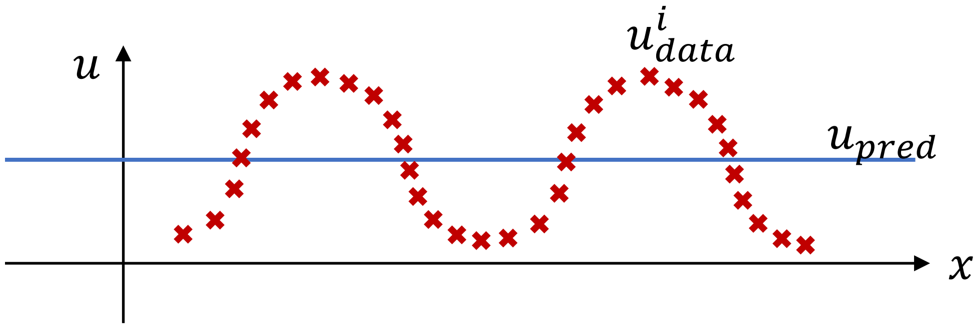
Ex3: Tanh activation, Iter: 4800

loss: 0.000854328566
Training time: 163.6977 Error u: 6.638371e-02



Pause and Ponder

- Can we replace $\sum_i^N \left| u_{data}^i - u_{pred}^i \right|^2$ with $\sum_i^N \left(u_{data}^i - u_{pred}^i \right)$ in the cost function?



$\sum_i^N \left(u_{data}^i - u_{pred}^i \right) \approx 0$

$\sum_i^N \left| u_{data}^i - u_{pred}^i \right|^2$ can be large!

- Is PINN using supervised or unsupervised learning?

  The training at collocation points is unsupervised learning!

  The training at data points is supervised learning

Raissi et al (2019)

E.g., Navier-Stokes equation

Given training data of u, v, find $\boldsymbol{\lambda_1, \lambda_2}$

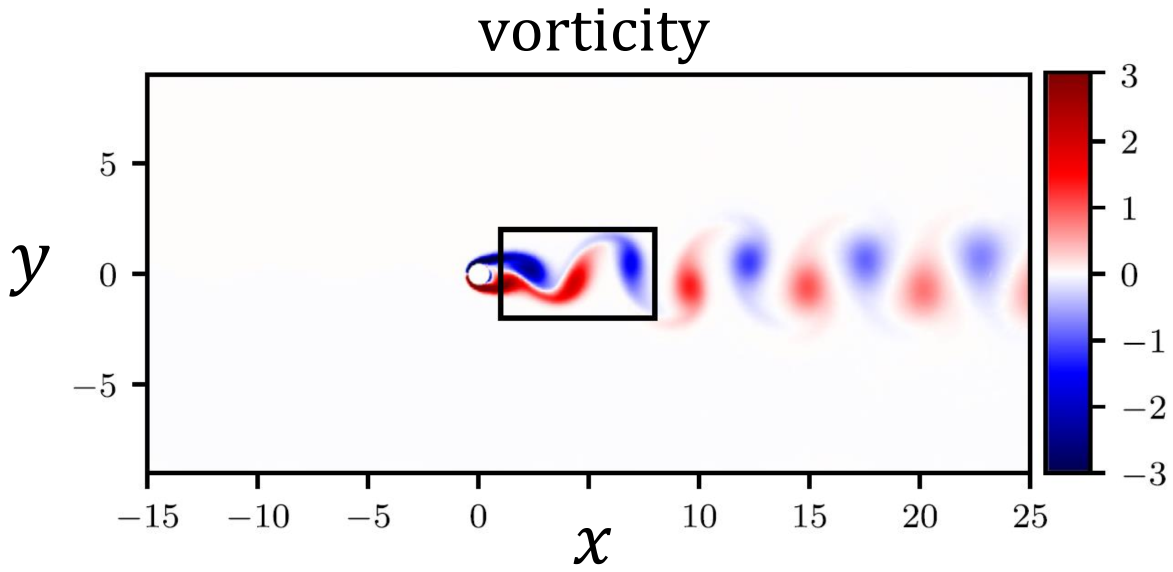
Problem statement

$$u_t + \lambda_1 (u u_x + v u_y) = -p_x + \lambda_2 (u_{xx} + u_{yy}),$$
$$v_t + \lambda_1 (u v_x + v v_y) = -p_y + \lambda_2 (v_{xx} + v_{yy}),$$

$$u_x + v_y = 0 \longrightarrow u = \psi_y, \quad v = -\psi_x$$

λ_1, λ_2: unknown parameters to be identified.
ψ: the stream function.
u, v: velocities
p: pressure

Ground truth from numerical simulation:



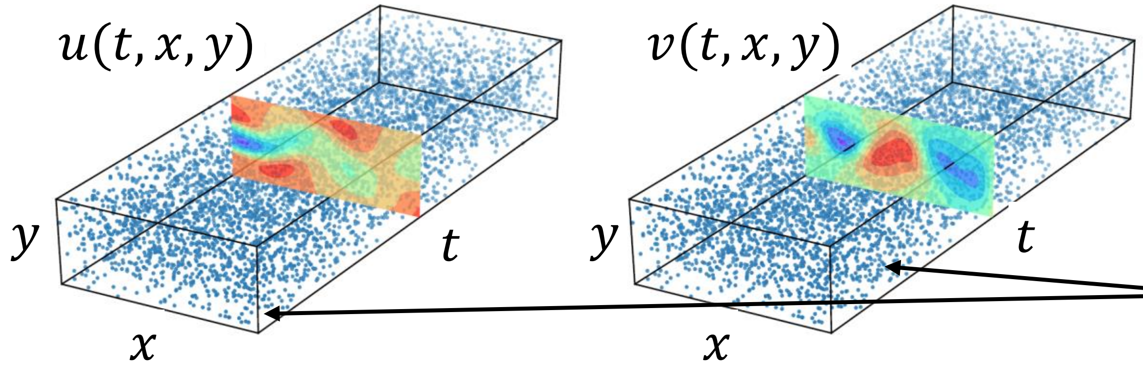
NN input: x, y, t
NN output: ψ, p
NN architecture: 9 layers 20 neurons per layer

Q: Why do we choose NN input/output this way?

Raissi et al (2019)

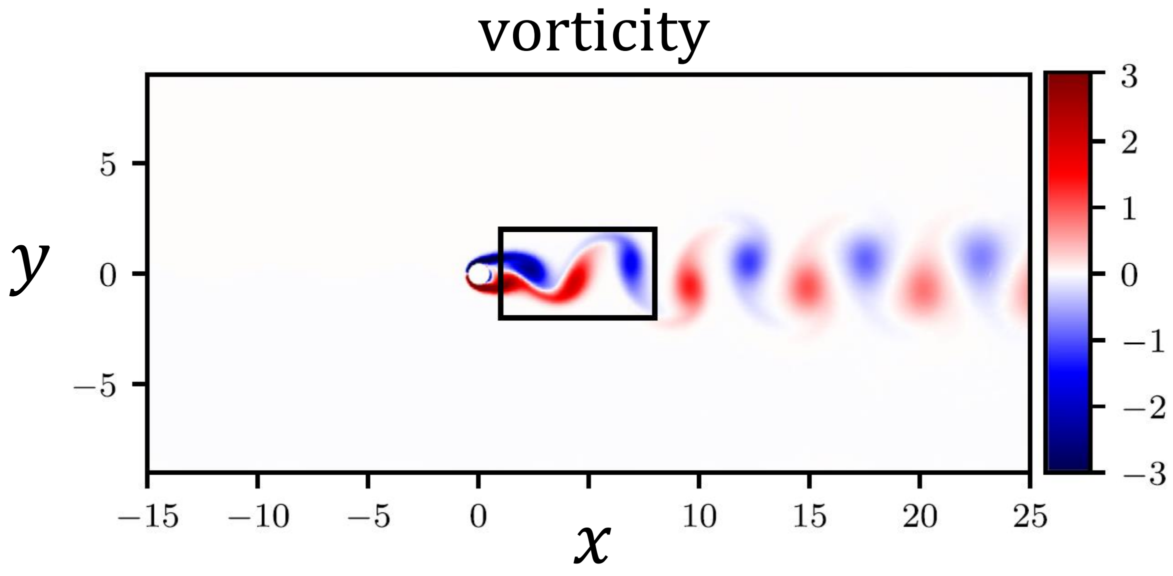
E.g., Navier-Stokes equation

Given training data of u, v, find $\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}$



Training data: 5000 pts of u, v within the domain
Collocation points: 5000 pts within the domain

Ground truth from numerical simulation:



NN input: x, y, t
NN output: ψ, p
NN architecture: 9 layers 20 neurons per layer