Physics-informed Neural Networks

Fan Xiao （樊箫）

Yau Mathematical Sciences Center,
Tsinghua University

June 20, 2023

Problem Setup

- In this work, we consider parametrized and nonlinear partial differential equations:

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

  where $u(t, x)$ denotes the latent solution, $\mathcal{N}[u; \lambda]$ is a nonlinear operator parametrized by λ and Ω is a subset of $\mathcal{R}^d$.
- We discuss the two problems:
  - Given fixed model parameters λ, what can be said about the unknown hidden state u(t, x) of the system?
  - What are the parameters λ that best describe the observed data?
- Propose two models: continuous time models and discrete time models

Continuous time models

- We parametrize $u(t,x)$ by a deep nerual network and define:

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

- Then we minimize the mean squared error loss

$$MSE = MSE_u + MSE_f$$

where

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

and

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

Here $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ denote the initial and boundary training data on $u(t,x)$ and $\left\{t_f^i, x_f^i\right\}_{i=1}^{N_f}$ specify the collocations points for $f(t,x)$.

Example(Schrodinger Equation)

- The nonlinear Schrodinger equation along with periodic boundary conditions is given by:

$$
\begin{aligned}
& ih_t + 0.5h_{xx} + |h|^2 h = 0, \quad x \in [-5, 5], \quad t \in [0, \pi/2], \\
& h(0, x) = 2\,\mathrm{sech}(x), \\
& h(t, -5) = h(t, 5), \\
& h_x(t, -5) = h_x(t, 5),
\end{aligned}
$$

- We parametrize $h(t, x) = [u(t, x), v(t, x)]$ by a 5-layer deep neural network with 100 neurons per layer and a hyperbolic tangent activation function.

Example(Schrodinger Equation)

- The parameters of the neural networks $h(t,x)$ can be learned by minimizing the mean squared error loss:

$$MSE = MSE_0 + MSE_b + MSE_f,$$

where

$$MSE_0 = \frac{1}{N_0}\sum_{i=1}^{N_0} |h(0, x_0^i) - h_0^i|^2,$$

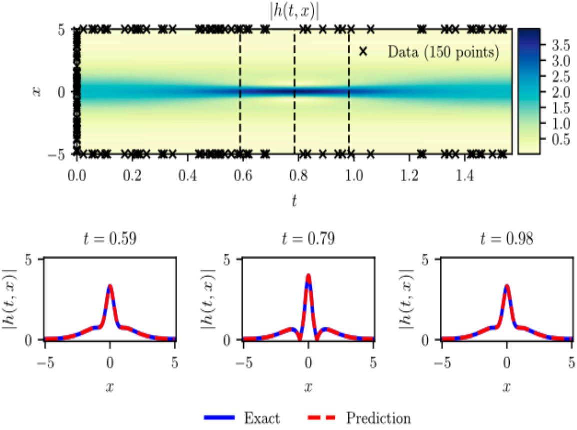
$$\begin{aligned} MSE_b = \frac{1}{N_b}\sum_{i=1}^{N_b} (|h^i(t_b^i, -5) - h^i(t_b^i, 5)|^2 \\ + |h_x^i(t_b^i, -5) - h_x^i(t_b^i, 5)|^2), \end{aligned}$$

and

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

Example(Schrodinger Equation)

- Here $\{x_0^i, h_0^i\}_{i=1}^{N_0}$ denotes the initial data, $\{t_b^i\}_{i=1}^{N_b}$ corresponds to the collocation points on the boundary, and $\left\{t_f^i, x_f^i\right\}_{i=1}^{N_f}$ represents the collocation points.
- We choose $N_0 = 50, N_b = 50$ and $N_f = 20000$.



More Numerical Results

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

Discrete Time Models

- Let us apply the general form of Runge-Kutta methods with q stages and obtain
  $u^{n+c_i} = u^n - \Delta t \sum_{j=1}^{q} a_{ij} \mathcal{N}\left[u^{n+c_j}\right], i = 1, \ldots, q,$
  $u^{n+1} = u^n - \Delta t \sum_{j=1}^{q} b_j \mathcal{N}\left[u^{n+c_j}\right].$
- Here, $u^{n+c_j}(x) = u\left(t^n + c_j \Delta t, x\right)$ for $j = 1, \ldots, q$.This equation can be equivalently expressed as

$$\begin{aligned} u^n &= u_i^n, \quad i = 1, \ldots, q, \\ u^n &= u_{q+1}^n. \end{aligned}$$

  where

$$\begin{aligned} u_i^n &:= u^{n+c_i} + \Delta t \textstyle\sum_{j=1}^{q} a_{ij} \mathcal{N}\left[u^{n+c_j}\right], \quad i = 1, \ldots, q, \\ u_{q+1}^n &:= u^{n+1} + \Delta t \textstyle\sum_{j=1}^{q} b_j \mathcal{N}\left[u^{n+c_j}\right]. \end{aligned}$$

Discrete Time Models

- We place a multi-output neural network prior on

$$\left[u^{n+c_1}(x), \ldots, u^{n+c_q}(x), u^{n+1}(x)\right]$$

- This prior assumption results in a physics-informed neural network that takes x as an input and outputs

$$\left[u_1^n(x), \ldots, u_q^n(x), u_{q+1}^n(x)\right]$$

- Let us consider the Allen-Cahn equation along with periodic boundary conditions

$$\begin{aligned}
&u_t - 0.0001 u_{xx} + 5u^3 - 5u = 0, \quad x \in [-1, 1], \quad t \in [0, 1], \\
&u(0, x) = x^2 \cos(\pi x), \\
&u(t, -1) = u(t, 1), \\
&u_x(t, -1) = u_x(t, 1).
\end{aligned}$$

Allen-Cahn Equation

- The nonlinear operator is given by

$$\mathcal{N}\left[u^{n+c_j}\right] = -0.0001 u_{xx}^{n+c_j} + 5\left(u^{n+c_j}\right)^3 - 5u^{n+c_j},$$

  and the parameters of the neural networks are learned by minimizing

$$SSE = SSE_n + SSE_b,$$

  where

$$SSE_n = \sum_{j=1}^{q+1} \sum_{i=1}^{N_n} \left| u_j^n\left(x^{n,i}\right) - u^{n,i} \right|^2$$

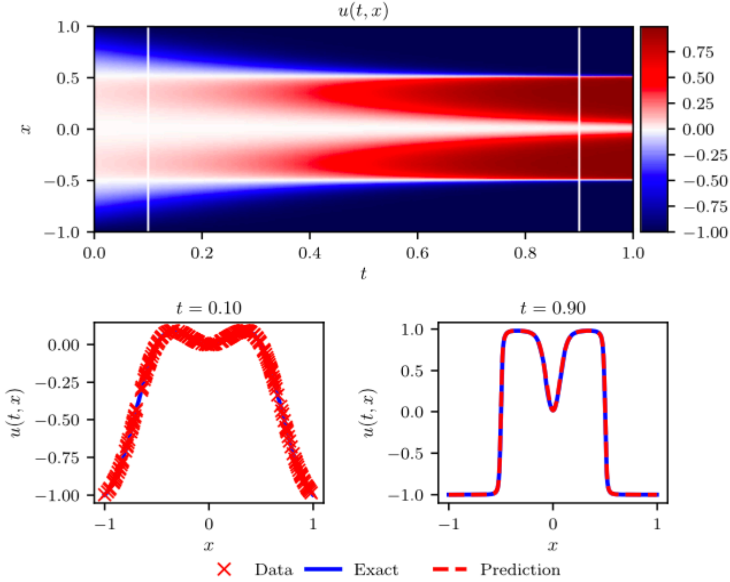
  and

$$\begin{aligned} SSE_b = {} & \sum_{i=1}^{q} \left| u^{n+c_i}(-1) - u^{n+c_i}(1) \right|^2 + \left| u^{n+1}(-1) - u^{n+1}(1) \right|^2 \\ & + \sum_{i=1}^{q} \left| u_x^{n+c_i}(-1) - u_x^{n+c_i}(1) \right|^2 + \left| u_x^{n+1}(-1) - u_x^{n+1}(1) \right|^2 \end{aligned}$$

Allen-Cahn Equation

- Here, $\{x^{n,i}, u^{n,i}\}_{i=1}^{N_n}$ cooresonds to the data at time-step t^n.
- Our training dataset consists of $N_n = 200$ initial data points that are sampled at time $t = 0.1$, and our goal is to predict the solution at time $t = 0.9$ using a singe time-step with size $\Delta t = 0.8$.
- We employ a discrete time physics-informed neural network with 4 hidden layers and 200 neurons per layer, the outpur layer predicts 101 quantities of interest corresponding to the $q = 100$ Runge-Kutta stages $u^{n+c_i}(x)$.
- Predict the nearly discontinuous solution at $t = 0.9$ in a single time-step with a relative L2 error of $6.99 \cdot 10^{-3}$.

Allen-Cahn Equation



More Numerical Results

Table A.4

Burgers' equation: Relative final prediction error measured in the $\mathbb{L}_2$ norm for different number of Runge–Kutta stages q and time-step sizes Δt. Here, the network architecture is fixed to 4 hidden layers with 50 neurons in each layer.

q \ Δt	0.2	0.4	0.6	0.8
1	3.5e−02	1.1e−01	2.3e−01	3.8e−01
2	5.4e−03	5.1e−02	9.3e−02	2.2e−01
4	1.2e−03	1.5e−02	3.6e−02	5.4e−02
8	6.7e−04	1.8e−03	8.7e−03	5.8e−02
16	5.1e−04	7.6e−02	8.4e−04	1.1e−03
32	7.4e−04	5.2e−04	4.2e−04	7.0e−04
64	4.5e−04	4.8e−04	1.2e−03	7.8e−04
100	5.1e−04	5.7e−04	1.8e−02	1.2e−03
500	4.1e−04	3.8e−04	4.2e−04	8.2e−04

Table A.5

Burgers equation: Relative $\mathcal{L}_2$ error between the predicted and the exact solution $u(t,x)$ for different number of training data N_n. Here, we have fixed $q = 500$, and used a neural network architecture with 3 hidden layers and 50 neurons per hidden layer.

N	250	200	150	100	50	10
Error	4.02e−4	2.93e−3	9.39e−3	5.54e−2	1.77e−2	7.58e−1

Data-driven discovery of partial differential equations

- Consider the Navier-Stokes equations in two dimensions given explicitly by

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

  where $u(t, x, y)$ denotes the x-component of the velocity field, $v(t, x, y)$ the y-component and $p(t, x, y)$.
- We assume that $u = \Phi_y, v = -\Phi_x$ for some latent function $\Phi(t, x, y)$.
- Given noisy measurements $\{t^i, x^i, y^i, u^i, v^i\}_{i=1}^N$ of the velocity field, we are interested in learning the parameters λ as well as the pressure $p(t, x, y)$. We define f and g as:

$$
\begin{aligned}
f &:= u_t + \lambda_1 (u u_x + v u_y) + p_x - \lambda_2 (u_{xx} + u_{yy}) \\
g &:= v_t + \lambda_1 (u v_x + v v_y) + p_y - \lambda_2 (v_{xx} + v_{yy})
\end{aligned}
$$

Data-driven discovery of partial differential equations

- We parametrize $[\Phi(t,x,y), p(t,x,y)]$ using a single neural network with two outputs and optimize

$$MSE := \frac{1}{N}\sum_{i=1}^{N}\left(\left|u\left(t^i,x^i,y^i\right)-u^i\right|^2+\left|v\left(t^i,x^i,y^i\right)-v^i\right|^2\right) + \frac{1}{N}\sum_{i=1}^{N}\left(\left|f\left(t^i,x^i,y^i\right)\right|^2+\left|g\left(t^i,x^i,y^i\right)\right|^2\right)$$

- We assume that $u=\Phi_y, v=-\Phi_x$ for some latent function $\Phi(t,x,y)$.
- Given noisy measurements $\{t^i,x^i,y^i,u^i,v^i\}_{i=1}^N$ of the velocity field, we are interested in learning the parameters λ as well as the pressure $p(t,x,y)$. We define f and g as:

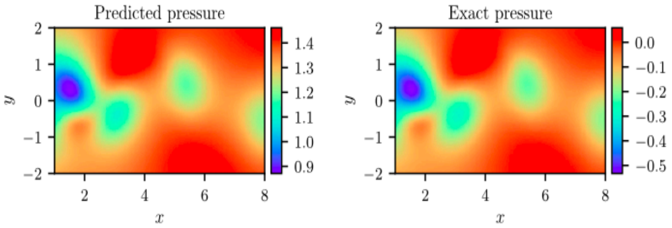
$$f := u_t + \lambda_1\left(uu_x+vu_y\right)+p_x-\lambda_2\left(u_{xx}+u_{yy}\right)$$
$$g := v_t + \lambda_1\left(uv_x+vv_y\right)+p_y-\lambda_2\left(v_{xx}+v_{yy}\right)$$

Data-driven discovery of partial differential equations

- We have assumed a uniform free stream velocity profile imposed at the left boundary, a zero pressure outflow condition imposed at the right boundary located 25 diameters downstream of the cylinder, and periodicity for the top and bottom boundaries of the $[-15, 25] \times [-8, 8]$ domain.
- We have chosen $N = 5000$.
- The neural network architecture used here consists of 9 layers with 20 neurons in each layer.
- For the case of noise-free training data, the error in estimating λ_1 and λ_2 is 0.078% and 4.67%. The predictions remain robust even when the training data are corrupted with 1% uncorrelated Gaussian noise, returning an error of 0.17%, and 5.70%, for λ_1 and λ_2, respectively.

Data-driven discovery of partial differential equations

- Provide a qualitatively accurate prediction of the entire pressure field $p(t, x, y)$ in the absence of training data of the pressure.



Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 0.999(uu_x + vu_y) = -p_x + 0.01047(u_{xx} + u_{yy})$ $v_t + 0.999(uv_x + vv_y) = -p_y + 0.01047(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t + 0.998(uu_x + vu_y) = -p_x + 0.01057(u_{xx} + u_{yy})$ $v_t + 0.998(uv_x + vv_y) = -p_y + 0.01057(v_{xx} + v_{yy})$

More Numerical Results

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.

	% error in λ_1				% error in λ_2			
N_u \ Noise	0%	1%	5%	10%	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

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 $N = 2{,}000$.

	% error in λ_1			% error in λ_2		
Layers \ Neurons	10	20	40	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

Data-driven discovery of partial differential equations

- The Korteweg-de Vries equation is given by:

$$u_t + \lambda_1 u u_x + \lambda_2 u_{xxx} = 0$$

with (λ_1, λ_2) being the unknown parameters. For the KdV equation, the nonlinear operator is given by

$$\mathcal{N}\left[u^{n+c_j}\right] = \lambda_1 u^{n+c_j} u_x^{n+c_j} - \lambda_2 u_{xxx}^{n+c_j}.$$

- We place a muti-output nerual network prior on

$$\left[u^{n+c_1}(x), \ldots, u^{n+c_q}(x)\right]$$

Data-driven discovery of partial differential equations

- Recall the Runge-kutta methods

$$
\begin{aligned}
u^{n+c_i} &= u^n - \Delta t \textstyle\sum_{j=1}^{q} a_{ij} \mathcal{N}\left[u^{n+c_j};\lambda\right], \quad i = 1,\ldots,q \\
u^{n+1} &= u^n - \Delta t \textstyle\sum_{j=1}^{q} b_j \mathcal{N}\left[u^{n+c_j};\lambda\right].
\end{aligned}
$$

- The above equation is equivalent to

$$
\begin{aligned}
u^n &= u_i^n, \quad i = 1,\ldots,q, \\
u^{n+1} &= u_i^{n+1}, \quad i = 1,\ldots,q,
\end{aligned}
$$

where

$$
\begin{aligned}
u_i^n &:= u^{n+c_i} + \Delta t \textstyle\sum_{j=1}^{q} a_{ij} \mathcal{N}\left[u^{n+c_j};\lambda\right], \quad i = 1,\ldots,q \\
u_i^{n+1} &:= u^{n+c_i} + \Delta t \textstyle\sum_{j=1}^{q} (a_{ij} - b_j) \mathcal{N}\left[u^{n+c_j};\lambda\right], \quad i = 1,\ldots,q.
\end{aligned}
$$

Data-driven discovery of partial differential equations

- We minimize the sum of the squared errors

$$SSE = SSE_n + SSE_{n+1},$$

where

$$SSE_n := \sum_{j=1}^{q} \sum_{i=1}^{N_n} \left| u_j^n \left(x^{n,i} \right) - u^{n,i} \right|^2,$$

and

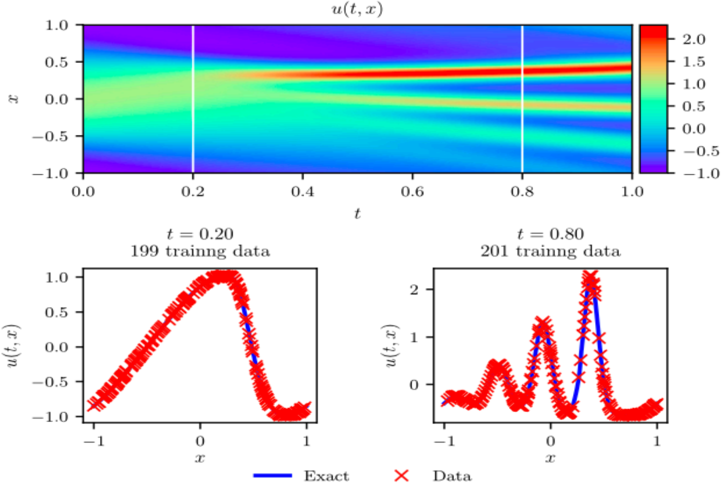
$$SSE_{n+1} := \sum_{j=1}^{q} \sum_{i=1}^{N_{n+1}} \left| u_j^{n+1} \left(x^{n+1,i} \right) - u^{n+1,i} \right|^2$$

- Here $\boldsymbol{x}^n = \left\{ x^{n,i} \right\}_{i=1}^{N_n}, \boldsymbol{u}^n = \left\{ u^{n,i} \right\}_{i=1}^{N_n}, \boldsymbol{x}^{n+1} = \left\{ x^{n+1,i} \right\}_{i=1}^{N_{n+1}}$, and $\boldsymbol{u}^{n+1} = \left\{ u^{n+1,i} \right\}_{i=1}^{N_{n+1}}$.

Data-driven discovery of partial differential equations

- We extract two solution snapshots at time $t^n = 0.2$ and $t^{n+1} = 0.8$, and randomly sub-sample them using $N_n = 199$ and $N_{n+1} = 201$ to generate a traning dataset. Here $\Delta t = 0.6$.
- The network architecture comprises of 4 hidden layers, 50 neurons per layer and an output layer predicting the solution at the q Runge-Kutta stages.
- Specifically, for the case of noise-free training data, the error in estimating λ_1 and λ_2 is 0.023%, and 0.006%, respectively, while the case with 1% noise in the training data returns errors of 0.057%, and 0.017%, respectively.

Data-driven discovery of partial differential equations



Correct PDE	$u_t + uu_x + 0.0025u_{xxx} = 0$
Identified PDE (clean data)	$u_t + 1.000uu_x + 0.0025002u_{xxx} = 0$
Identified PDE (1% noise)	$u_t + 0.999uu_x + 0.0024996u_{xxx} = 0$

More Numerical Results

Table B.8
Burgers' equation: Percentage error in the identified parameters λ_1 and λ_2 for different gap size Δt between two different snapshots and for different noise levels.

	% error in λ_1				% error in λ_2			
Δt \ Noise	0%	1%	5%	10%	0%	1%	5%	10%
0.2	0.002	0.435	6.073	3.273	0.151	4.982	59.314	83.969
0.4	0.001	0.119	1.679	2.985	0.088	2.816	8.396	8.377
0.6	0.002	0.064	2.096	1.383	0.090	0.068	3.493	24.321
0.8	0.010	0.221	0.097	1.233	1.918	3.215	13.479	1.621

Table B.9
Burgers' equation: Percentage error in the identified parameters λ_1 and λ_2 for different number of hidden layers and neurons in each layer.

	% error in λ_1			% error in λ_2		
Layers \ Neurons	10	25	50	10	25	50
1	1.868	4.868	1.960	180.373	237.463	123.539
2	0.443	0.037	0.015	29.474	2.676	1.561
3	0.123	0.012	0.004	7.991	1.906	0.586
4	0.012	0.020	0.011	1.125	4.448	2.014

Thank you!