물리정보기반 신경망
(Physics-Informed Neural Network and Deep Ritz Method)

Jaeyong Lee

Center for AI and Natural Sciences

Korea Institute for advanced Study (KIAS)

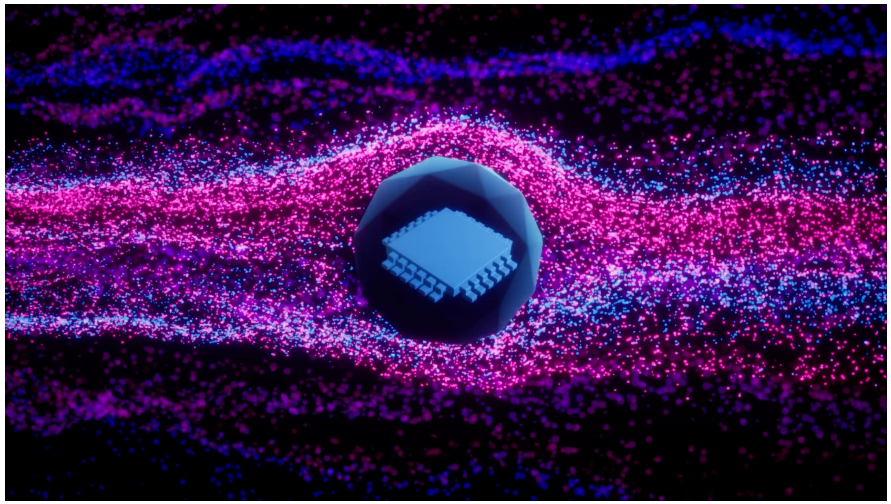
Personal Homepage - https://sites.google.com/view/jaeyonglee

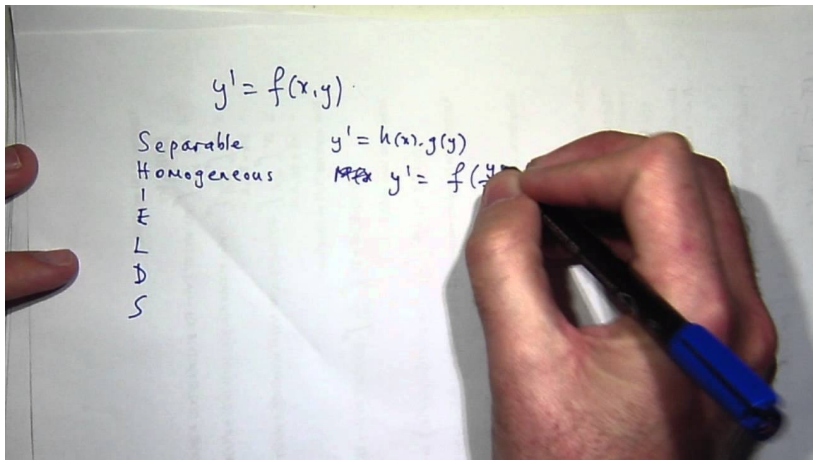
CAC 2023, June 26~27, 2023

Contents

Differential equations

- **In mathematics, a differential equation is an equation that relates one or more functions and their derivatives.**
  - The functions generally represent **physical quantities.**
  - The derivatives represent **their rates of change.**
  - The differential equation defines a **relationship between the above two.**

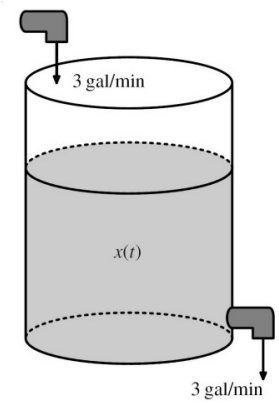




https://www.quantamagazine.org/latest-neural-nets-solve-worlds-hardest-equations-faster-than-ever-before-20210419/
https://www.youtube.com/watch?v=jb9U-ZOlvDs

Ordinary Differential equations (ODEs) and Partial Differential equations (PDEs)

ODE example)



Let $x(t)$ denote the amount of salt at time t.

Balance Law: (rate in)-(rate out)

$$\frac{dx(t)}{dt} = 3 - 3x(t).$$

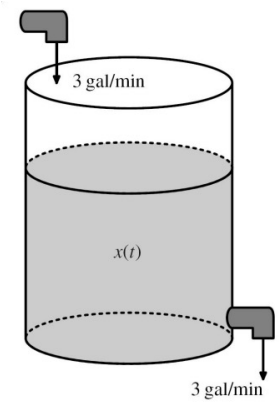
Initial condition (IC) :

$$x(t=0) = 100.$$

https://www.math.uh.edu/~jiwenhe/math3331/lectures/sec2_5.pdf

Ordinary Differential equations (ODEs) and Partial Differential equations (PDEs)

ODE example)



Let $x(t)$ denote the amount of salt at time t.

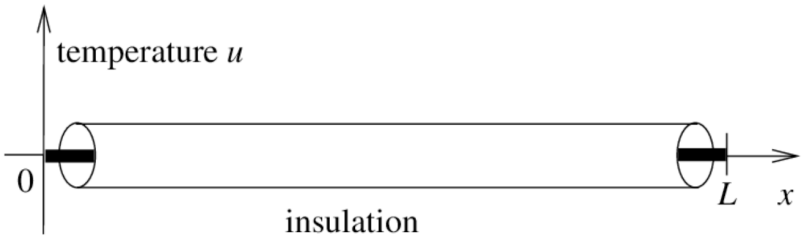
Balance Law: (rate in)-(rate out)

$$\frac{dx(t)}{dt} = 3 - 3x(t).$$

Initial condition (IC) :

$$x(t=0) = 100.$$

PDE example)



Let $u(t,x)$ denote the temperature at point x at time t.

Governing equation (heat equation) :

$$\frac{\partial u(t,x)}{\partial t} = k\frac{\partial^2 u(t,x)}{\partial x^2}.$$

Initial condition (IC) :

$$u(t=0,x) = f(x).$$

Boundary condition (BC) :

$$u(t,x=0) = h(t), \qquad u(t,x=L) = g(t).$$

https://math.libretexts.org/Bookshelves/Differential_Equations/Book%3A_Differential_Equations_for_Engineers_(Lebl)/4%3A_Fourier_series_and_PDEs/4.06%3A_PDEs_separation_of_variables_and_the_heat_equation

It is Hard to solve the equation exactly!!

Let $x(t)$ denote the amount of salt at time t.

Balance Law: (rate in)-(rate out)

$$\frac{dx(t)}{dt} = 3 - 3x(t).$$

Initial condition (IC) :

$$x(t=0) = 100.$$

Let $u(t,x)$ denote the temperature at point x at time t.

Governing equation (heat equation) :

$$\frac{\partial u(t,x)}{\partial t} = k\frac{\partial^2 u(t,x)}{\partial x^2}.$$

Initial condition (IC) :

$$u(t=0,x) = f(x).$$

Boundary condition (BC) :

$$u(t,x=0) = h(t), \qquad u(t,x=L) = g(t).$$

It is Hard to solve the equation exactly!!

Let $x(t)$ denote the amount of salt at time t.

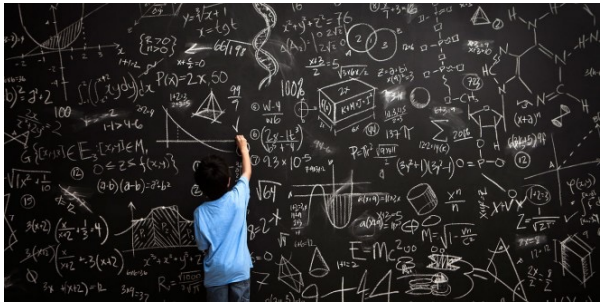
Balance Law: (rate in)-(rate out)

$$\frac{dx(t)}{dt} = 3 - 3x(t).$$

Initial condition (IC) :

$$x(t=0) = 100.$$

Numerical analysis



Let $u(t,x)$ denote the temperature at point x at time t.

Governing equation (heat equation) :

$$\frac{\partial u(t,x)}{\partial t} = k\frac{\partial^2 u(t,x)}{\partial x^2}.$$

Initial condition (IC) :

$$u(t=0,x) = f(x).$$

Boundary condition (BC) :

$$u(t,x=0) = h(t), \qquad u(t,x=L) = g(t).$$

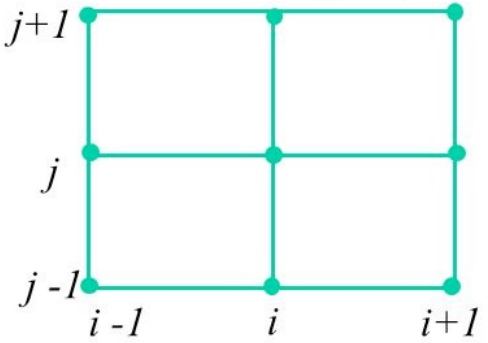
Traditional numerical methods

- **Finite difference method**

Finite Difference Method – Example

- Consider Poisson's equation.

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + g = 0$$



- Discretise

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} + g = 0$$

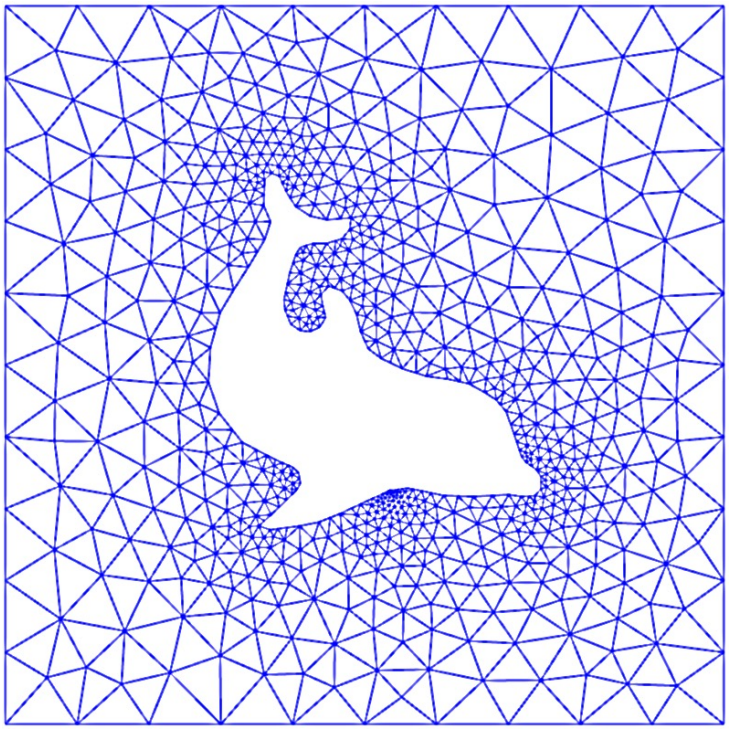
- Difference formula for each node:

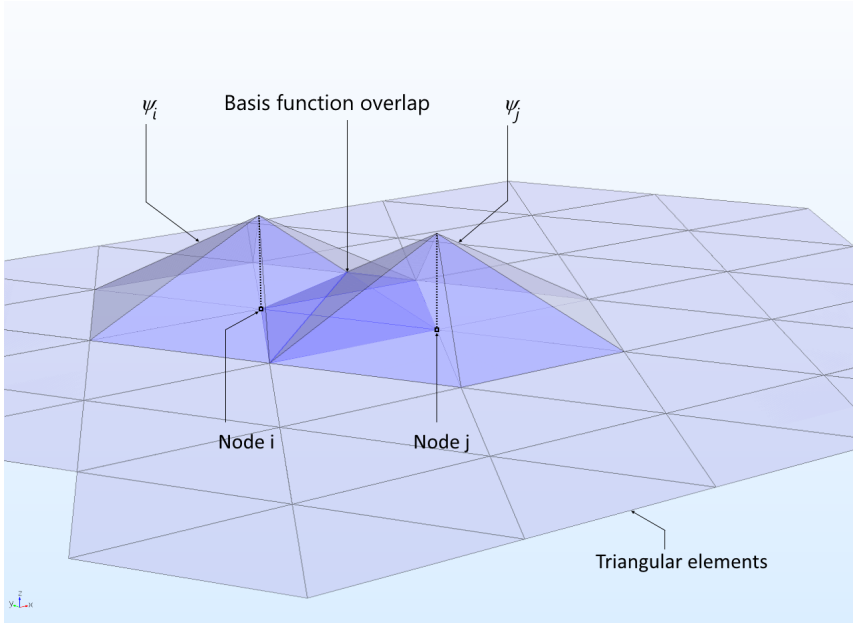
$$2\left(\Delta x^2 + \Delta y^2\right)u_{i,j} = \Delta y^2\left(u_{i+1,j} + u_{i-1,j}\right) + \Delta x^2\left(u_{i,j+1} + u_{i,j-1}\right) + \Delta x^2 \Delta y^2 g$$

15

Traditional numerical methods

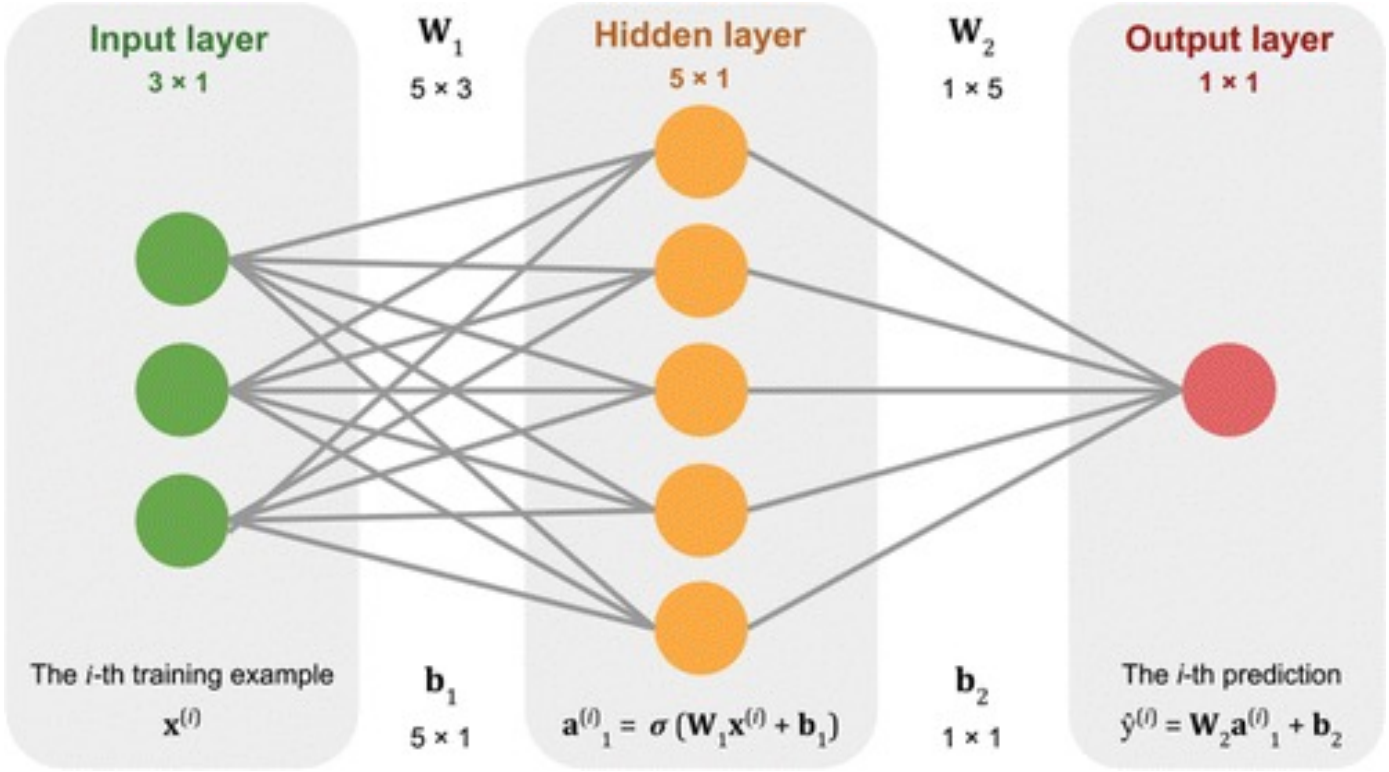
- **Finite difference method**
- **Finite element method (FEM), Finite volume method (FVM), …**





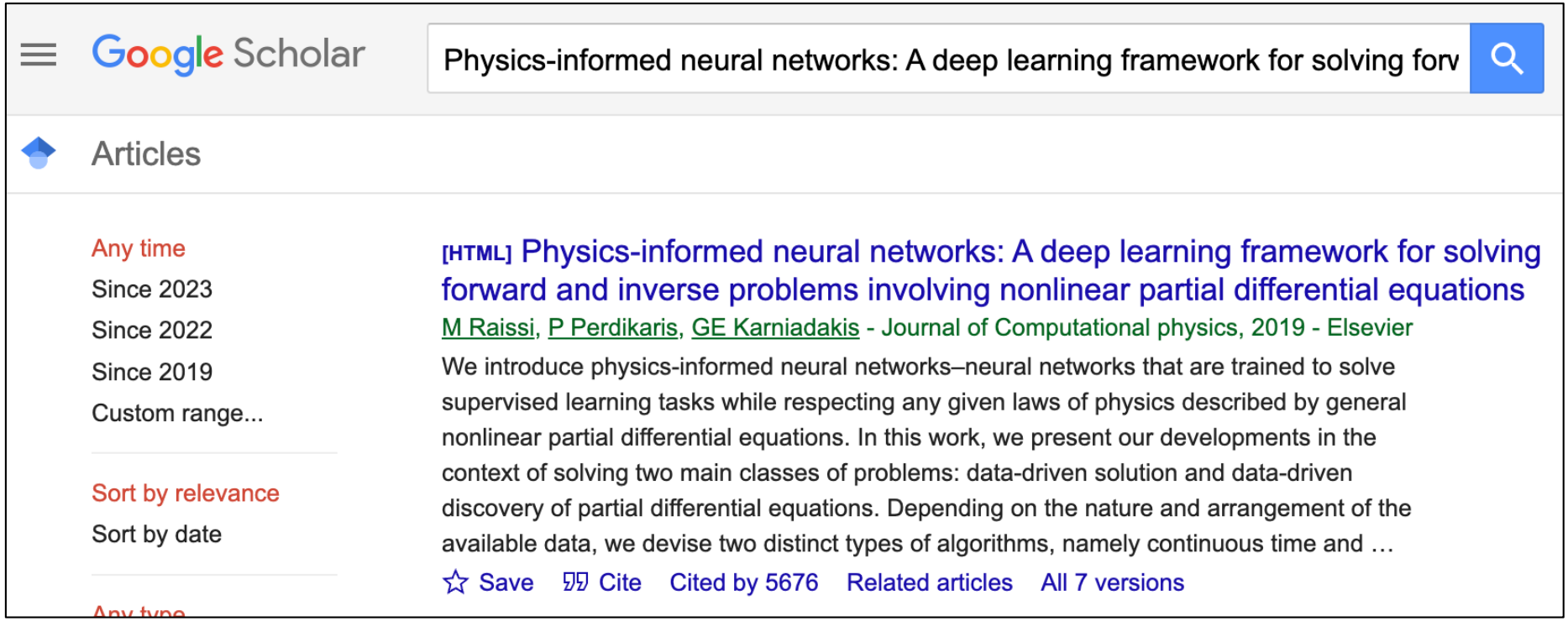
https://www.comsol.com/multiphysics/finite-element-method

Deep neural network

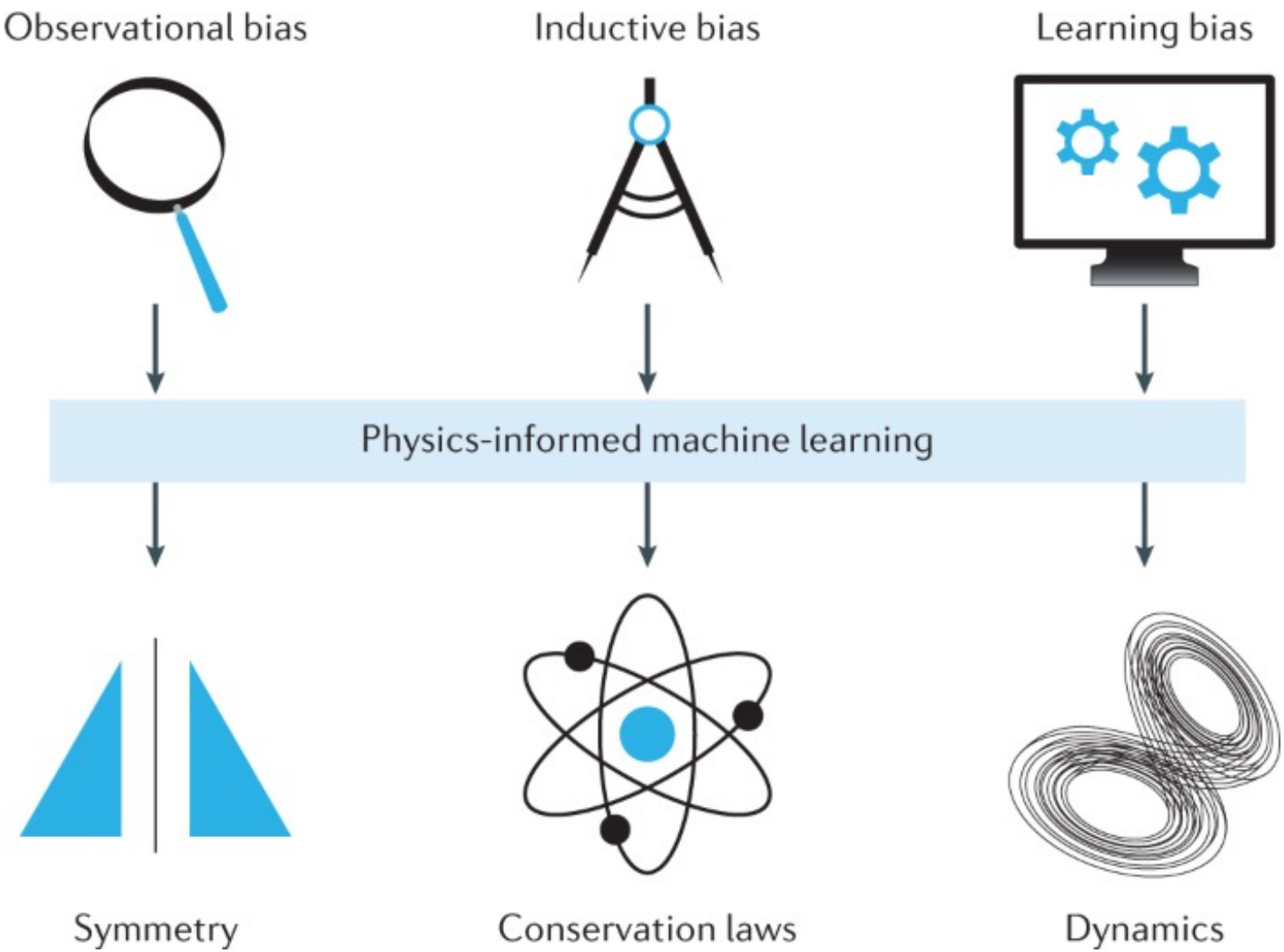


https://wiki.seg.org/wiki/Neural_networks

Physics-informed machine learning???



Physics-informed machine learning???



https://www.nature.com/articles/s42254-021-00314-5

Deep learning approach to PDEs

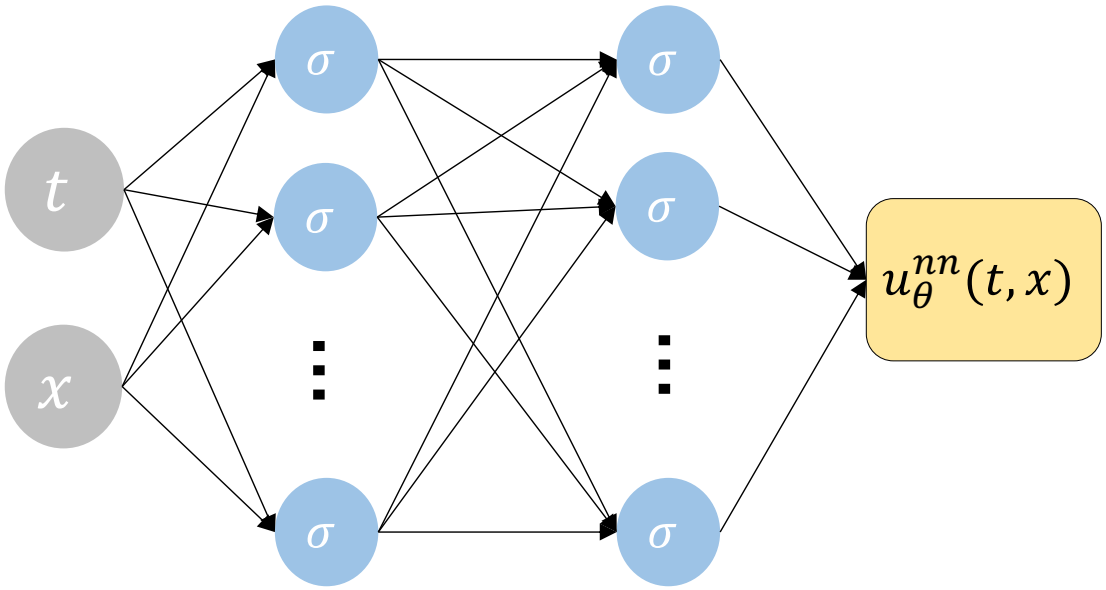
- **Using neural networks directly to parametrize the solution to PDEs.**
- **Solve one instance of PDE at a time.**
- **Models**
  - Physics Informed Neural Network (PINN)
  - Deep Ritz Method (DRM)

Find $\boldsymbol{u(t,x)}$ satisfying

$$\mathcal{L}_{PDE} = f(u, u_t, u_x, u_{xx}, \dots) = 0$$

$$\mathcal{L}_{IC} = u(0,x) - g(x) = 0$$

$$\mathcal{L}_{BC} = u|_{\partial\Omega} - h(t,x)|_{\partial\Omega} = 0$$



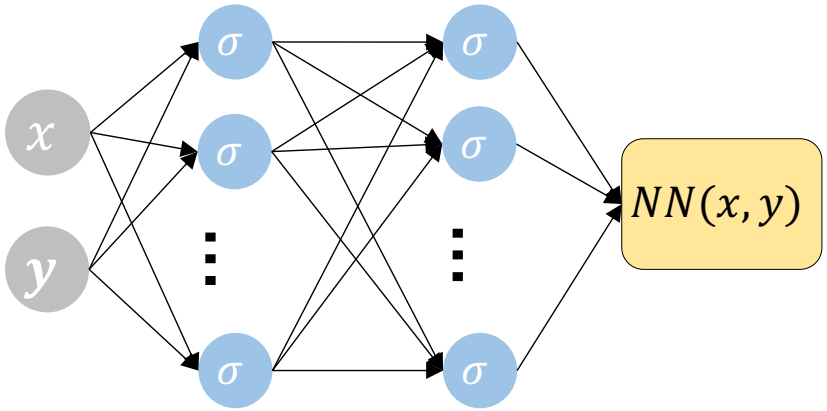
History of PDE solver based on neural network

"Artificial neural networks for solving ordinary and partial differential equations", IEEE Trans. Neural Netw. (1997)

Ex) Poisson equation

$$\begin{cases} \Psi_{xx} + \Psi_{yy} = f(x,y), & (x,y) \in [0,1]^2 \\ \Psi(0,y) = f_0(y), & \Psi(1,y) = f_1(y) \\ \Psi(x,0) = g_0(x), & \Psi(x,1) = g_1(x) \end{cases}$$

Network Parameters : $\vec{p}$

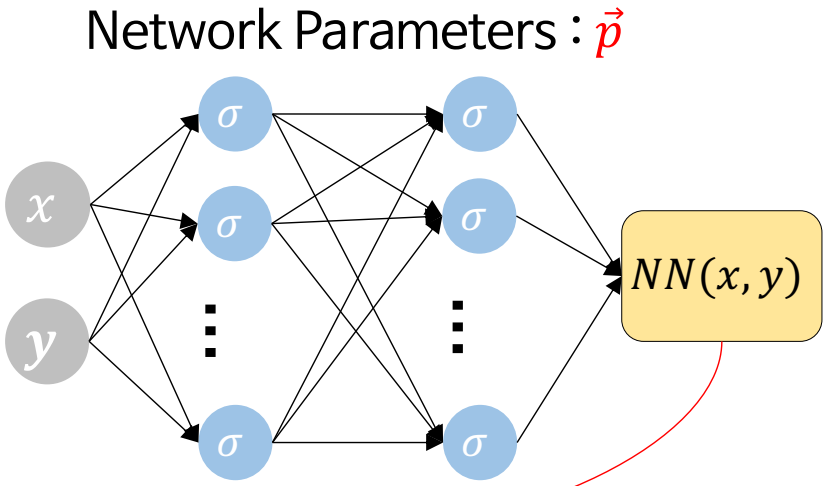


History of PDE solver based on neural network

"Artificial neural networks for solving ordinary and partial differential equations", IEEE Trans. Neural Netw. (1997)

Ex) Poisson equation

$$\begin{cases} \Psi_{xx} + \Psi_{yy} = f(x,y), & (x,y) \in [0,1]^2 \\ \Psi(0,y) = f_0(y), & \Psi(1,y) = f_1(y) \\ \Psi(x,0) = g_0(x), & \Psi(x,1) = g_1(x) \end{cases}$$



They use a trial solution Ψ_t as

$$\Psi_t = A(x,y) + x(1-x)y(1-y)NN(x,y,\vec{p}),$$

where $A(x,y)$ is chosen to satisfy the boundary conditions.

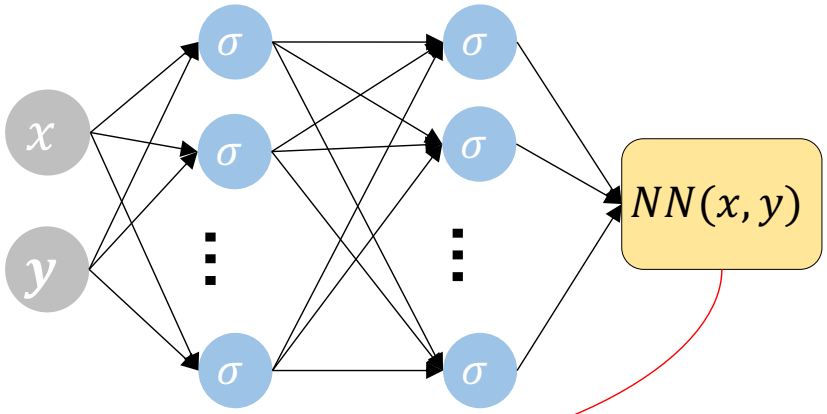
History of PDE solver based on neural network

"Artificial neural networks for solving ordinary and partial differential equations", IEEE Trans. Neural Netw. (1997)

Ex) Poisson equation

$$\begin{cases} \Psi_{xx} + \Psi_{yy} = f(x,y), & (x,y) \in [0,1]^2 \\ \Psi(0,y) = f_0(y), & \Psi(1,y) = f_1(y) \\ \Psi(x,0) = g_0(x), & \Psi(x,1) = g_1(x) \end{cases}$$

Network Parameters : $\vec{p}$



They use a trial solution Ψ_t as

$$\Psi_t = A(x,y) + x(1-x)y(1-y)NN(x,y,\vec{p}),$$

where $A(x,y)$ is chosen to satisfy the boundary conditions.

The error to be minimized is given by

$$E(\vec{p}) = \sum_i \left\{ \frac{\partial^2 \Psi(x_i,y_i)}{\partial x^2} + \frac{\partial^2 \Psi(x_i,y_i)}{\partial y^2} - f(x_i,y_i) \right\}^2$$

How to calculate the derivatives of the output with respect to any of its inputs??
=> Chain rule

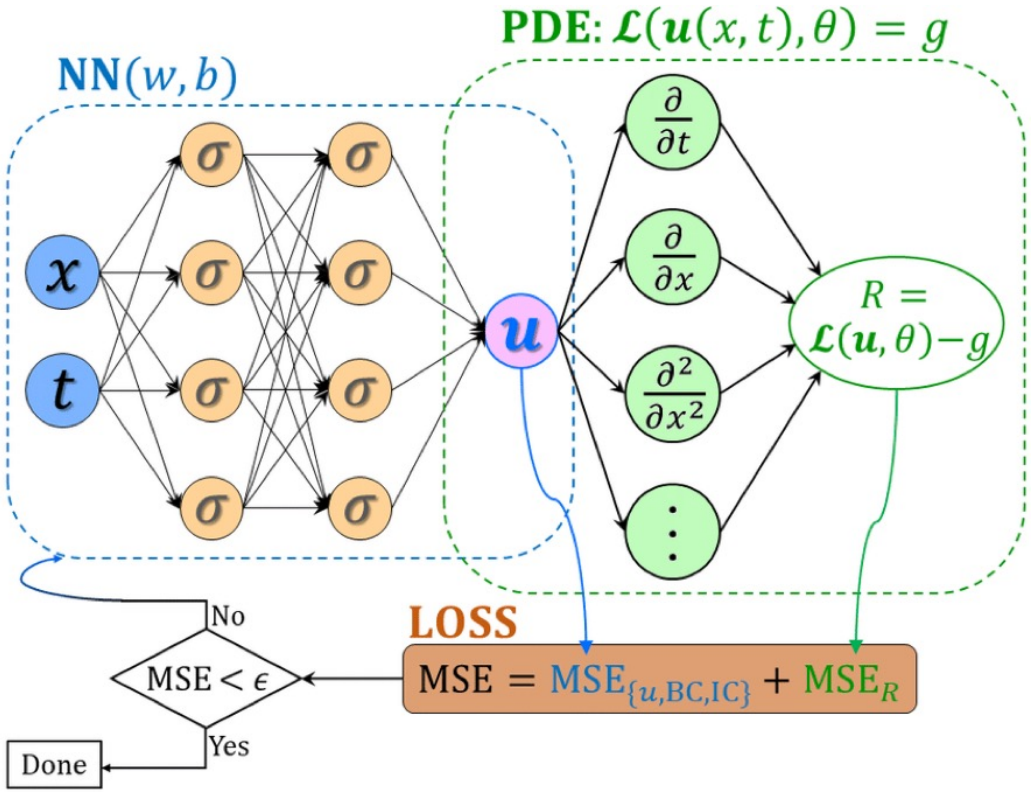
Physics-informed neural network (PINN)

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computation al physics*, *378*, 686-707.

Automatic differentiation and the back-propagation algorithm

"Here we revisit them using modern computational tools, and apply them to more challenging dynamic problems described by time-dependent nonlinear partial differential equations."

[Schematic of a PINN]



Lu, L., Meng, X., Mao, Z., & Karniadakis, G. E. (2021). DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, *63*(1), 208-228.

Physics-informed neural network (PINN) - Forward problem

(e.g.) Burgers' equation

$$\text{Eq}: u_t + uu_x - (0.01/\pi)u_{xx} = 0$$
$$\text{IC}: u(0, x) = -\sin(\pi x)$$
$$\text{BC}: u(t, -1) = u(t, 1) = 0$$

$$x \in [-1,1], t \in [0,1]$$

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial different ial equations. *Journal of Computational physics*, *378*, 686-707.

Physics-informed neural network (PINN) - Forward problem

(e.g.) Burgers' equation

$$\text{Eq}: u_t + uu_x - (0.01/\pi)u_{xx} = 0$$
$$\text{IC}: u(0,x) = -\sin(\pi x)$$
$$\text{BC}: u(t,-1) = u(t,1) = 0$$
$$x \in [-1,1], t \in [0,1]$$

Loss function

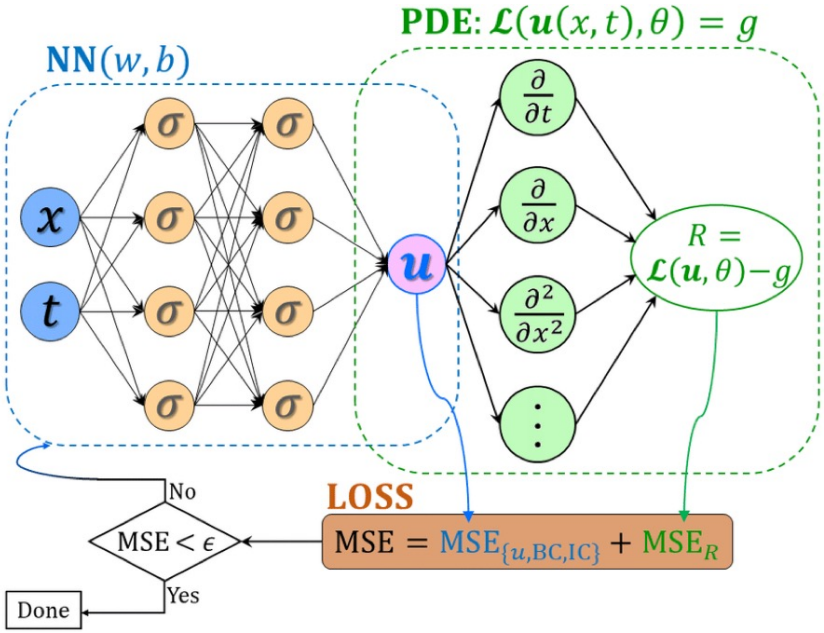
$$\boldsymbol{L(\theta)} = \frac{1}{N_{Eq}}\sum_{i=1}^{N_{Eq}} \left|u_t(t^i,x^i) + u(t^i,x^i)u_x(t^i,x^i) - (0.01/\pi)u_{xx}(t^i,x^i)\right|^2 + \frac{1}{N_{IC}}\sum_{j=1}^{N_{IC}} \left|u(t^j,x^j) + \sin(\pi x^j)\right|^2 + \frac{1}{N_{BC}}\sum_{k=1}^{N_{BC}} \left|u(t^k,x^k)\right|^2$$

Data:

$(t_i, x_i) \in [0,1]\times[-1,1]$

$(t_j, x_j) \in \{0\}\times[-1,1]$

$(t_k, x_k) \in [0,1]\times\{-1,1\}$



Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, *378*, 686-707.

Physics-informed neural network (PINN) - Forward problem

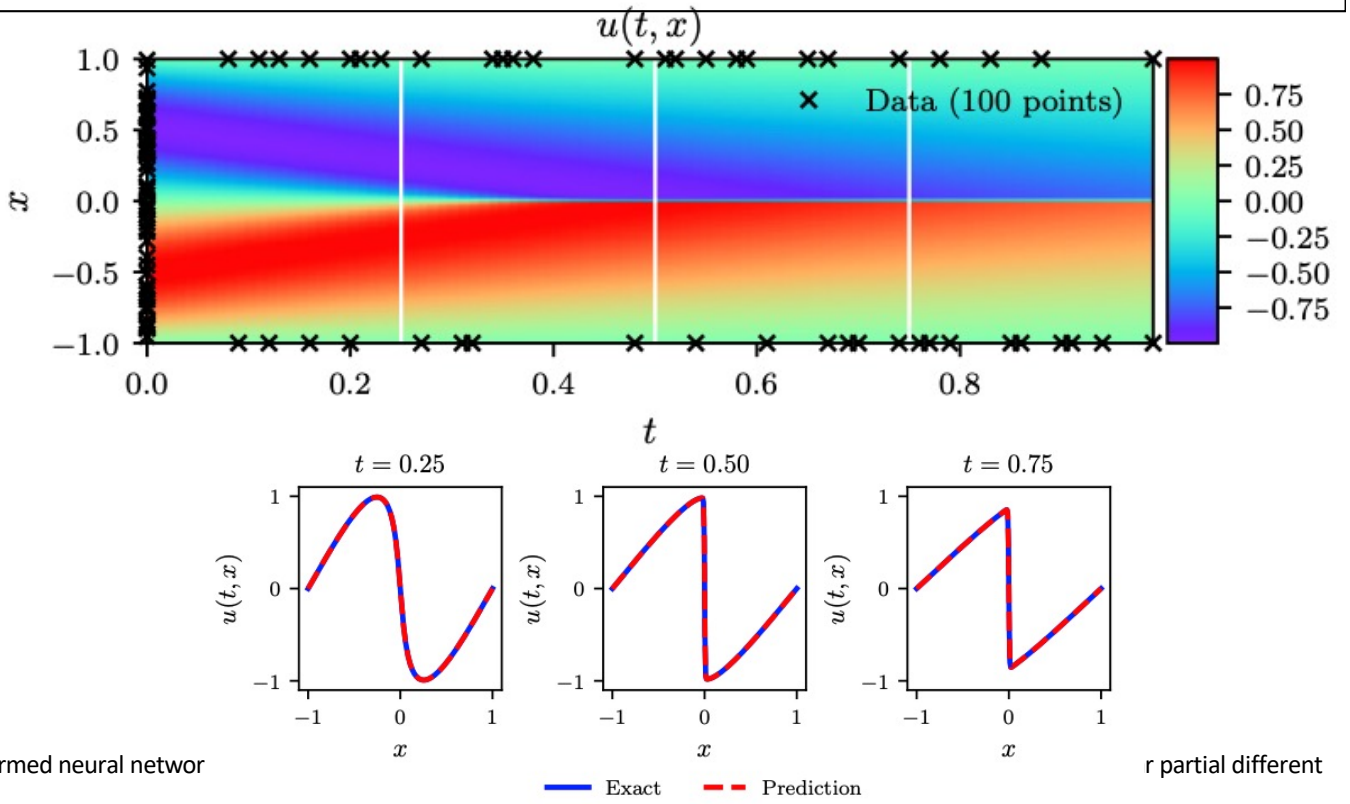
(e.g.) Burgers' equation

$$\text{Eq}: u_t + uu_x - (0.01/\pi)u_{xx} = 0$$
$$\text{IC}: u(0,x) = -\sin(\pi x)$$
$$\text{BC}: u(t,-1) = u(t,1) = 0$$
$$x \in [-1,1], t \in [0,1]$$

Loss function

$$\boldsymbol{L(\theta)} = \frac{1}{N_{Eq}}\sum_{i=1}^{N_{Eq}} \left|u_t(t^i,x^i) + u(t^i,x^i)u_x(t^i,x^i) - (0.01/\pi)u_{xx}(t^i,x^i)\right|^2 + \frac{1}{N_{IC}}\sum_{j=1}^{N_{IC}} \left|u(t^j,x^j) + \sin(\pi x^j)\right|^2 + \frac{1}{N_{BC}}\sum_{k=1}^{N_{BC}} \left|u(t^k,x^k)\right|^2$$

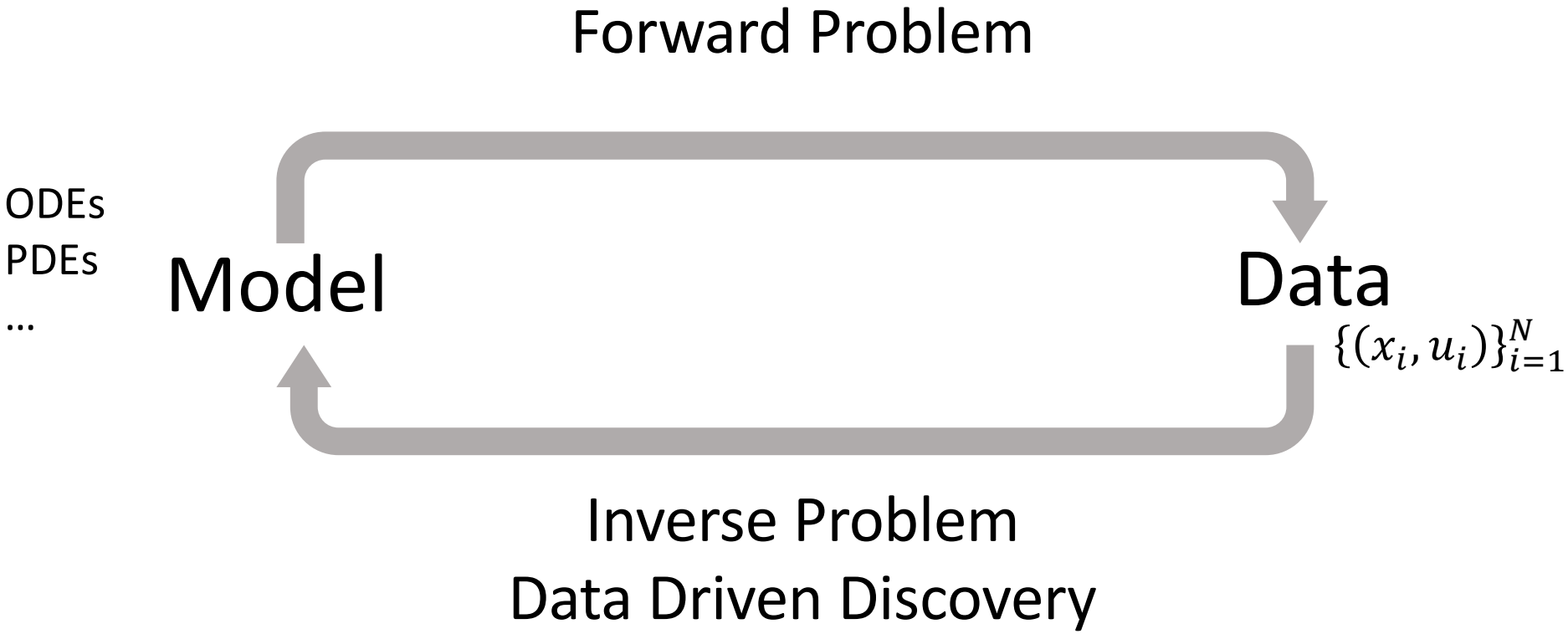
Data:
$(t_i, x_i) \in [0,1]\times[-1,1]$
$(t_j, x_j) \in \{0\}\times[-1,1]$
$(t_k, x_k) \in [0,1]\times\{-1,1\}$



Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networ
ial equations. *Journal of Computational physics*, *378*, 686-707.

r partial different

Physics-informed neural network (PINN) - Inverse problem???



Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, *378*, 686-707.

Physics-informed neural network (PINN) - Inverse problem

(e.g.) Burgers' equation

$x \in [-1,1], t \in [0,1]$

Forward problem

Eq : $u_t + uu_x - (0.01/\pi)u_{xx} = 0$
IC : $u(0, x) = -\sin(\pi x)$
BC : $u(t, -1) = u(t, 1) = 0$

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial different ial equations. *Journal of Computational physics*, *378*, 686-707.

Physics-informed neural network (PINN) - Inverse problem

(e.g.) Burgers' equation

$$\text{Eq}: u_t + \lambda_1 u u_x - \lambda_2 u_{xx} = 0$$
$$\text{IC}: u(0, x) = -\sin(\pi x)$$
$$\text{BC}: u(t, -1) = u(t, 1) = 0$$

$x \in [-1,1], t \in [0,1]$

Forward problem

$$\text{Eq}: u_t + u u_x - (0.01/\pi) u_{xx} = 0$$
$$\text{IC}: u(0, x) = -\sin(\pi x)$$
$$\text{BC}: u(t, -1) = u(t, 1) = 0$$

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial different ial equations. *Journal of Computational physics*, *378*, 686-707.

Physics-informed neural network (PINN) - Inverse problem

(e.g.) Burgers' equation

$$\text{Eq}: u_t + \lambda_1 u u_x - \lambda_2 u_{xx} = 0$$
$$\text{IC}: u(0, x) = -\sin(\pi x)$$
$$\text{BC}: u(t, -1) = u(t, 1) = 0$$

$$x \in [-1,1], t \in [0,1]$$

Data-driven discovery of PDE (Solving inverse problem for unknown parameter λ_1 and λ_2)

- We want to identify unknown using an additional learning parameter.
- Assume a small amount of additional data $\{t_l, x_l, u_l\}_{l=1}^{N_{data}}$ is given.
- Update not only nn parameter θ but also the PDE parameters λ_1 and λ_2.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial different ial equations. *Journal of Computational physics*, *378*, 686-707.

Physics-informed neural network (PINN) - Inverse problem

(e.g.) Burgers' equation

$$\text{Eq}: u_t + \lambda_1 u u_x - \lambda_2 u_{xx} = 0$$
$$\text{IC}: u(0,x) = -\sin(\pi x)$$
$$\text{BC}: u(t,-1) = u(t,1) = 0$$

$$x \in [-1,1], t \in [0,1]$$

Loss function

$$\boldsymbol{L(\theta, \lambda_1, \lambda_2)}$$
$$= \frac{1}{N_{Eq}} \sum_{i=1}^{N_{Eq}} \left| u_t(t^i,x^i) + \lambda_1 u(t^i,x^i) u_x(t^i,x^i) - \lambda_2 u_{xx}(t^i,x^i) \right|^2 + \frac{1}{N_{IC}} \sum_{j=1}^{N_{IC}} \left| u(t^j,x^j) + \sin(\pi x^j) \right|^2 + \frac{1}{N_{BC}} \sum_{k=1}^{N_{BC}} \left| u(t^k,x^k) \right|^2$$
$$+ \frac{1}{N_{data}} \sum_{l=1}^{N_{data}} \left| u(t^l,x^l) - u_l \right|^2$$

Data:

$(t_i, x_i) \in [0,1] \times [-1,1]$

$(t_j, x_j) \in \{0\} \times [-1,1]$

$(t_k, x_k) \in [0,1] \times \{-1,1\}$

$(t_l, x_l, u_l) \in [0,1] \times [-1,1] \times \mathbb{R}$

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial different ial equations. *Journal of Computational physics*, *378*, 686-707.

Physics-informed neural network (PINN) - Inverse problem

(e.g.) Burgers' equation

$$\text{Eq}: u_t + \lambda_1 u u_x - \lambda_2 u_{xx} = 0$$
$$\text{IC}: u(0,x) = -\sin(\pi x)$$
$$\text{BC}: u(t,-1) = u(t,1) = 0$$
$$x \in [-1,1], t \in [0,1]$$

Loss function

$$L(\theta, \lambda_1, \lambda_2) = \frac{1}{N_{Eq}} \sum_{i=1}^{N_{Eq}} \left| u_t(t^i,x^i) + \lambda_1 u(t^i,x^i) u_x(t^i,x^i) - \lambda_2 u_{xx}(t^i,x^i) \right|^2 + \frac{1}{N_{IC}} \sum_{j=1}^{N_{IC}} \left| u(t^j,x^j) + \sin(\pi x^j) \right|^2 + \frac{1}{N_{BC}} \sum_{k=1}^{N_{BC}} \left| u(t^k,x^k) \right|^2$$

$$+ \frac{1}{N_{data}} \sum_{l=1}^{N_{data}} \left| u(t^l,x^l) - u_l \right|^2$$

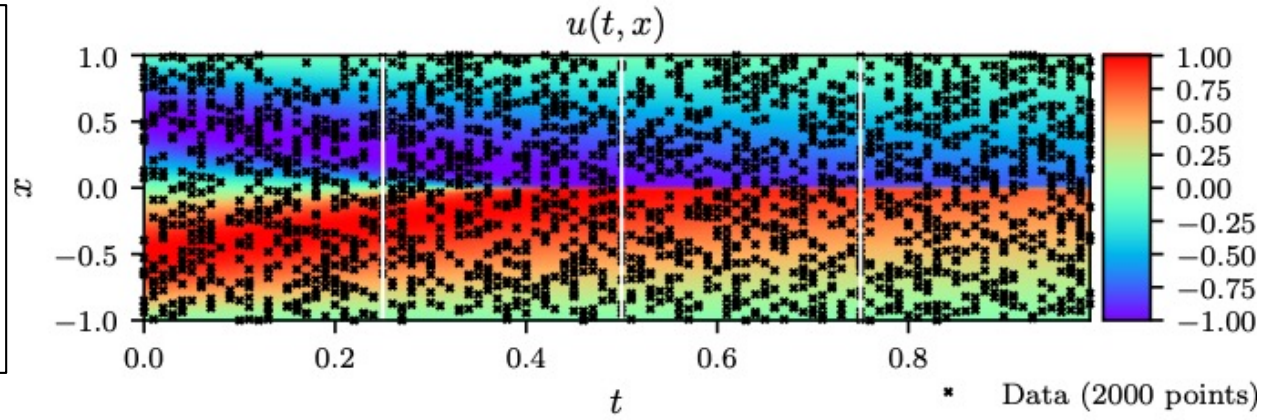
Data:

$(t_i, x_i) \in [0,1] \times [-1,1]$

$(t_j, x_j) \in \{0\} \times [-1,1]$

$(t_k, x_k) \in [0,1] \times \{-1,1\}$

$(t_l, x_l, u_l) \in [0,1] \times [-1,1] \times \mathbb{R}$



Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial different ial equations. *Journal of Computational physics*, *378*, 686-707.

Physics-informed neural network (PINN) - Inverse problem

(e.g.) Burgers' equation

$$\text{Eq}: u_t + \lambda_1 u u_x - \lambda_2 u_{xx} = 0$$
$$\text{IC}: u(0,x) = -\sin(\pi x)$$
$$\text{BC}: u(t,-1) = u(t,1) = 0$$
$$x \in [-1,1], t \in [0,1]$$

Loss function

$$L(\theta, \lambda_1, \lambda_2) = \frac{1}{N_{Eq}} \sum_{i=1}^{N_{Eq}} \left| u_t(t^i,x^i) + \lambda_1 u(t^i,x^i) u_x(t^i,x^i) - \lambda_2 u_{xx}(t^i,x^i) \right|^2 + \frac{1}{N_{IC}} \sum_{j=1}^{N_{IC}} \left| u(t^j,x^j) + \sin(\pi x^j) \right|^2 + \frac{1}{N_{BC}} \sum_{k=1}^{N_{BC}} \left| u(t^k,x^k) \right|^2$$

$$+ \frac{1}{N_{data}} \sum_{l=1}^{N_{data}} \left| u(t^l,x^l) - u_l \right|^2$$

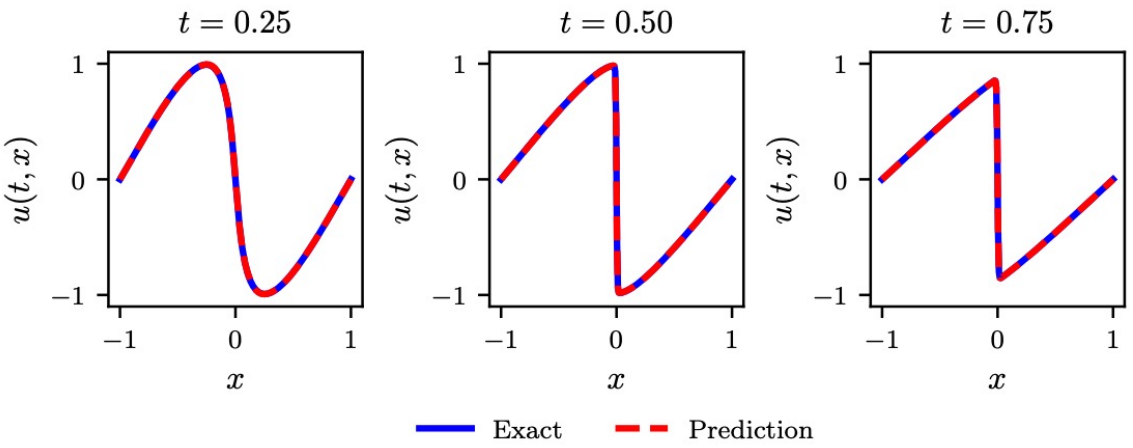
Data:

$(t_i, x_i) \in [0,1] \times [-1,1]$

$(t_j, x_j) \in \{0\} \times [-1,1]$

$(t_k, x_k) \in [0,1] \times \{-1,1\}$

$(t_l, x_l, u_l) \in [0,1] \times [-1,1] \times \mathbb{R}$



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

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural
ial equations. *Journal of Computational physics*, *378*, 686-707.

ferent

Deep Ritz Method (DRM)

Yu, B. (2018). The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, *6*(1), 1-12.

Consider the following PDE:

$$\begin{cases} -\Delta u = f, \ in \ \Omega \\ u = 0, \ on \ \partial\Omega \end{cases}$$

Deep Ritz Method (DRM)

Yu, B. (2018). The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, *6*(1), 1-12.

Consider the following PDE:

$$\begin{cases} -\Delta u = f, \ in\ \Omega \\ u = 0, \ on\ \partial\Omega \end{cases}$$

Main idea!!

Loss = $\min_{u|_{\partial\Omega}=0} \int_\Omega \left(\frac{1}{2}|\nabla u|^2 - fu\right) dx$

Deep Ritz Method (DRM)

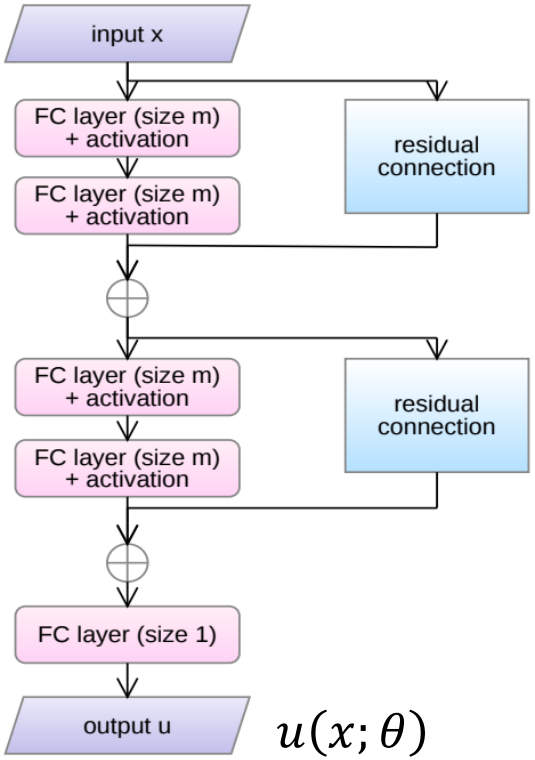
Yu, B. (2018). The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, *6*(1), 1-12.

Consider the following PDE:

$$\begin{cases} -\Delta u = f, \ in\ \Omega \\ u = 0, \ on\ \partial\Omega \end{cases}$$

Main idea!!

$$\text{Loss} = \min_{u|_{\partial\Omega}=0} \int_\Omega \left(\frac{1}{2}|\nabla u|^2 - fu\right) dx$$



$u(x;\theta)$

Deep Ritz method

$$\min_\theta \int_\Omega \left(\frac{1}{2}|\nabla u(x;\theta)|^2 - f(x)u(x;\theta)\right) dx + \beta \int_{\partial\Omega} (u(x;\theta))^2 ds$$

Penalty term for BC

Deep Ritz Method (DRM)

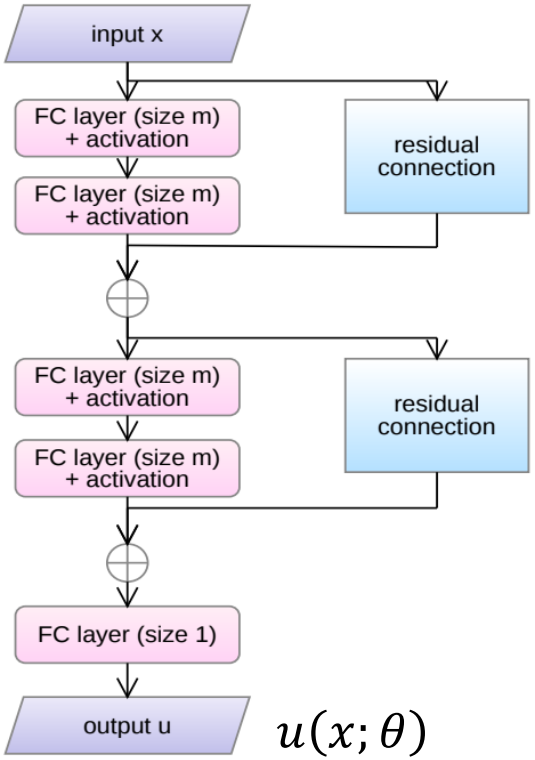
Yu, B. (2018). The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, *6*(1), 1-12.

Consider the following PDE:

$$\begin{cases} -\Delta u = f, \ in\ \Omega \\ u = 0, \ on\ \partial\Omega \end{cases}$$

Main idea!!

$$\text{Loss} = \min_{u|_{\partial\Omega}=0} \int_\Omega \left(\frac{1}{2}|\nabla u|^2 - fu\right) dx$$



$u(x;\theta)$

Deep Ritz method

$$\min_\theta \int_\Omega \left(\frac{1}{2}|\nabla u(x;\theta)|^2 - f(x)u(x;\theta)\right) dx + \beta \int_{\partial\Omega} (u(x;\theta))^2 ds$$

Penalty term for BC

Consider $(d = 10)$

$$\begin{aligned} -\Delta u &= 0, && x \in (0,1)^{10} \\ u(x) &= \sum_{k=1}^{5} x_{2k-1}x_{2k}, && x \in \partial(0,1)^{10}. \end{aligned} \tag{15}$$

The solution of this problem is simply $u(x) = \sum_{k=1}^{5} x_{2k-1}x_{2k}$, and we will use the exact solution to compute the error of our model later.

Benefits and drawbacks of deep learning approach

Dockhorn, T. (2019). A discussion on solving partial differential equations using neural networks. *arXiv preprint arXiv:1904.07200*.

Table 6: Benefits and drawbacks of solving partial differential equations with neural networks.

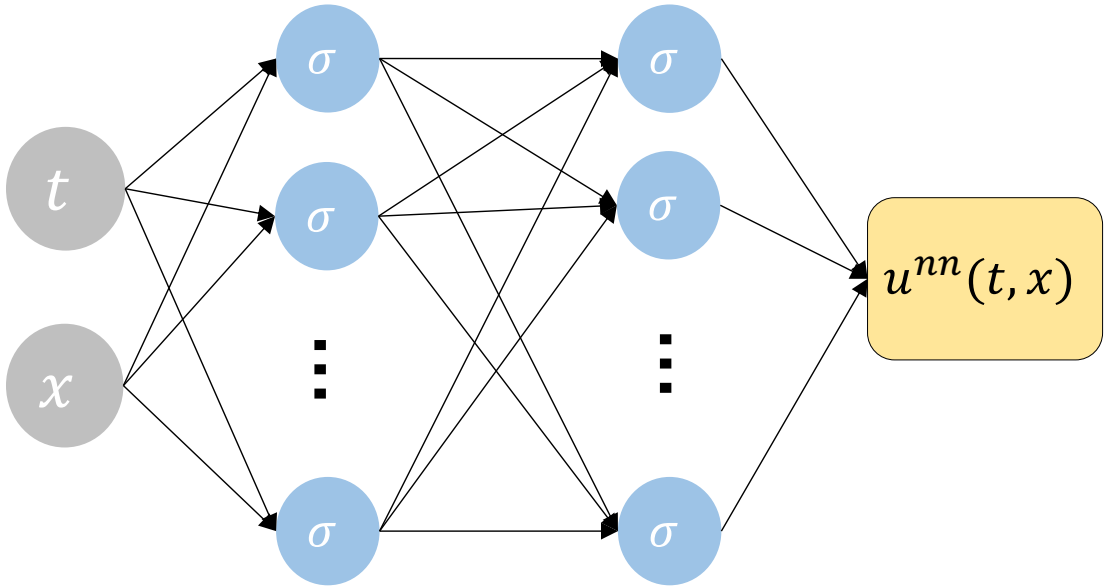
Benefits	**Drawbacks**
Expressive power: Already small neural networks are able to approximate the solution of partial differential equations well.	*Convergence:* Most probably due to optimization issues, we could not empirically show that errors decrease (with a certain convergence rate) with increasing neural network size; showing theoretical convergence seems to be even more difficult.
Ease of implementation: Our algorithm is straightforward and can be easily implemented using backpropagation.	*Run time:* At least when the BFGS optimizer is used, our simulations seem to be considerably slower than classical numerical methods. Using the proposed algorithm in combination with a first order optimization method, e.g. Adam [16], is subject to future work.
Arbitrary domains (in higher dimensions): The algorithm is based on drawing random points from a domain, which can be readily extended to arbitrary domains; no triangulation of the domain is needed.	*Scalability:* At the moment, it is unclear how well the algorithm will scale to more difficult problems, e.g., three-dimensional turbulent flows.
Freedom of approximation spaces: For some classical methods and systems of PDEs, we have certain restrictions on the function spaces of the solutions, e.g., the inf–sup condition [15] needs to be satisfied for the Navier–Stokes equations when using the finite element method.	*Randomness:* Different random initializations lead to different results of the algorithm whereas (most) classical numerical methods are deterministic. Further investigation is necessary to better exploit this nondeterminism and to eventually turn it into an advantage.
Sensor data: We can easily incorporate (noisy) information of sensors by adding a term to the loss function.	

Conclusion

- **PDE solver**

Find $\boldsymbol{u(t,x)}$ satisfying

$$\mathcal{L}_{PDE} = f(u, u_t, u_x, u_{xx}, \dots) = 0$$
$$\mathcal{L}_{IC} = u(0,x) - g(x) = 0$$
$$\mathcal{L}_{BC} = u|_{\partial\Omega} - h(t,x)|_{\partial\Omega} = 0$$



- **Models**
  - Physics-Informed Neural Network
  - Deep Ritz Method

(실습)

Let's implement

Physics-Informed Neural Network and

Deep Ritz Method!