Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations

Raissi, M., Perdikaris, P., & Karniadakis, G. E.

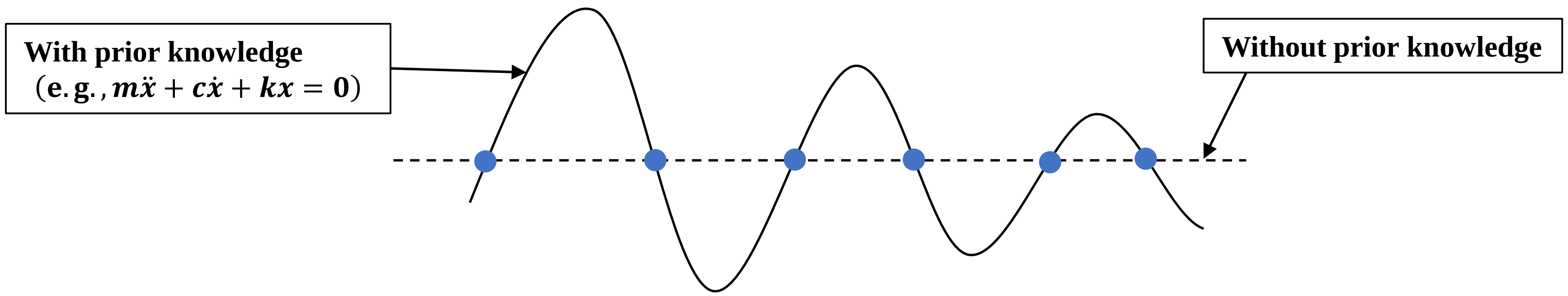
Journal of Computational Physics (JCP), 2019

2023. 01. 10

Seonghak KIM

Introduction

- **Expensive data acquisition in complex physical engineering systems (i.e., small data)**
  - (−) under partial information, making decisions
  - (−) lack of robustness and fail to convergence (albeit using state-of-the-art ML techniques)
- **Utilizing of prior knowledge**
  - Physical law (e.g., Newton's laws)
  - Constraints the space of admissible solutions
    - e.g., Abrogation of non-realistic solutions that violate the conservation law



[Constraint of solutions space]

Introduction

- **Previous works**
  - Gaussian process regression tailored to linear operator
    - $(-)$ Local linearization of nonlinear terms → limited applications
    - $(-)$ Inaccurate predictability in highly nonlinear regimes
- **Physics-informed neural networks**
  - Neural networks as universal function approximators[†]
    - Automatic differentiation → 'auto_grad'
    - $(+)$ It can address the nonlinear problems.

† K. Hornik et al., *Neural Networks*, 1989

Models

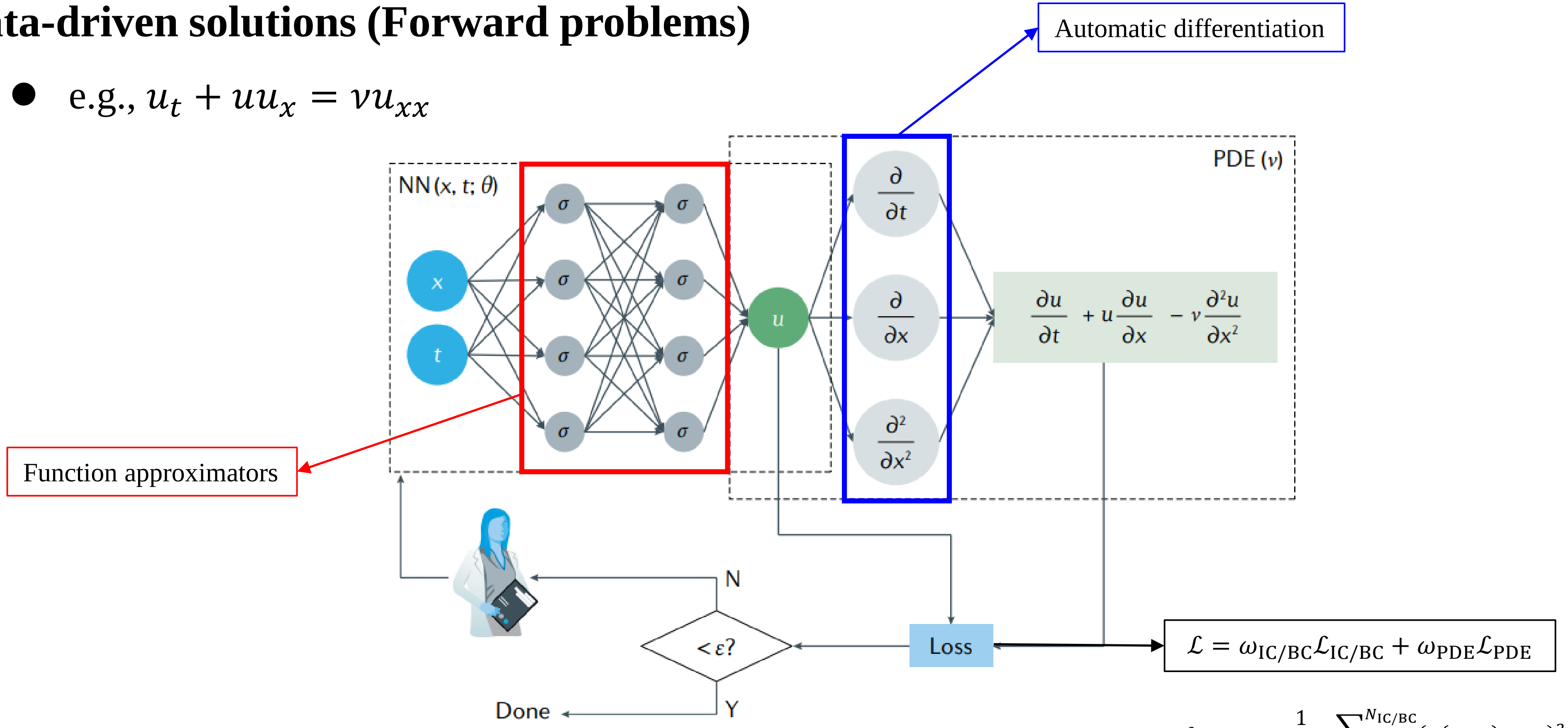
- **Parametrized and nonlinear PDE of general form**
  - It encapsulates a wide range of problems in math, physics including conservations laws, diffusion, and so on.

$$\boxed{u_t + \mathcal{N}[u;\lambda] = 0, x \in \Omega \subset \mathbb{R}^D, t \in [0,T]}$$

    - $u(t,x)$: latent (hidden) solution
    - $\mathcal{N}(\cdot;\lambda)$: nonlinear operator parametrized by λ
  - e.g., 1-D Burgers eq.
    - $u_t + \lambda_1 u_x - \lambda_2 u_{xx} = 0 \rightarrow \mathcal{N}(u;\lambda) = \lambda_1 u u_x - \lambda_2 u u_{xx}$ and $\lambda = (\lambda_1, \lambda_2)$
- **Algorithms**
  - Data-driven solutions of PDE: model → data; Forward problem
  - Data-driven discovery of PDE: data → model; ***Inverse problem***

Models

- **Data-driven solutions (Forward problems)**
  - e.g., $u_t + uu_x = \nu u_{xx}$



$$\mathcal{L}_{\text{IC/BC}} = \frac{1}{N_{\text{IC/BC}}}\sum_{i=1}^{N_{\text{IC/BC}}}(u(x_i, t_i) - u_i)^2$$

$$\mathcal{L}_{PDE} = \frac{1}{N_{\text{PDE}}}\sum_{j=1}^{N_{\text{PDE}}}(u_t + uu_x - \nu u_{xx})^2\Big|_{x_j, t_j}$$

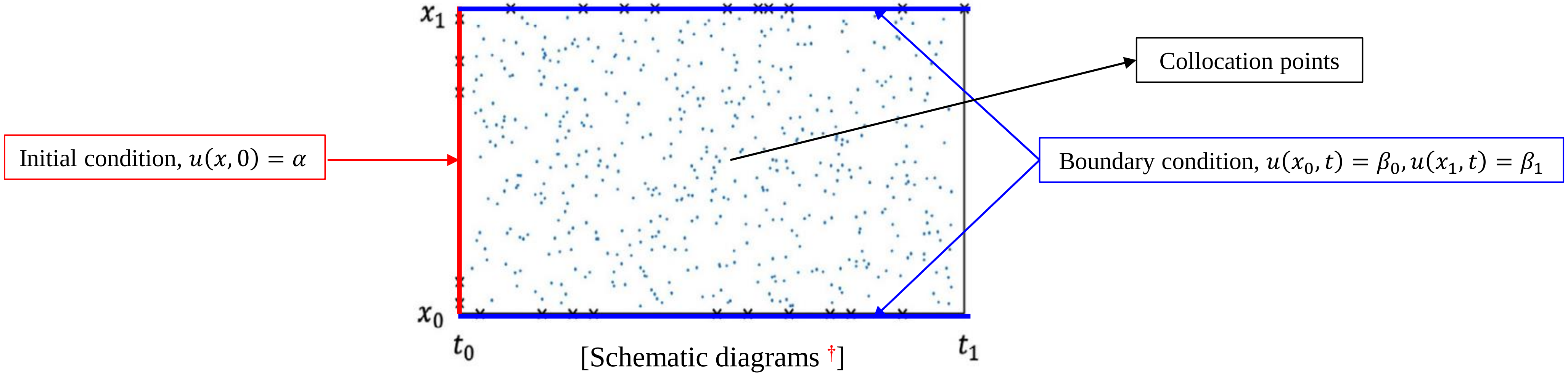
[PINN algorithm †]

† G. E. Karniadakis et al., *Nature Reviews Physics*, 2021

Models

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

- **Data-driven solutions**
  - $f \coloneqq u_t + \mathcal{N}[u] \to f = 0$
  - Loss functions $\mathcal{L} = \mathcal{L}_u + \mathcal{L}_f$
    - $\mathcal{L}_u = \frac{1}{N_u}\sum_{i=1}^{N_u}(u(x_i,t_i) - u_i)^2$ where (x_i, t_i) is sampled points at the initial/boundary locations
    - $\mathcal{L}_f = \frac{1}{N_f}\sum_{j=1}^{N_f} f\left(x_j, t_j\right)^2$ where $\left(x_j, t_j\right)$ is sampled points in the entire domain ($\coloneqq$ collocation points)

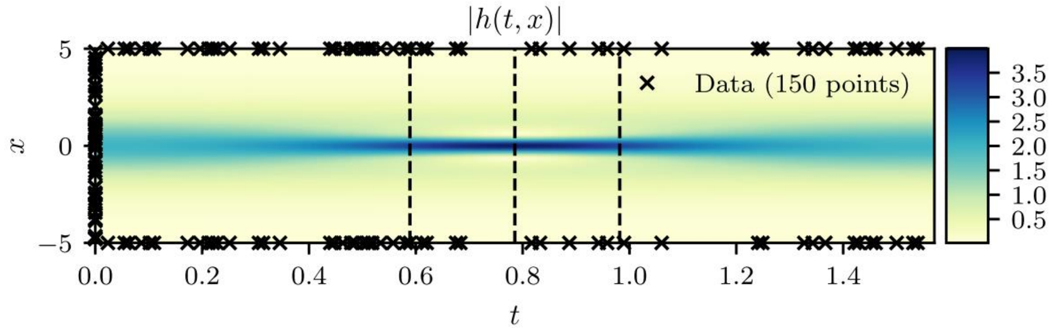


[Schematic diagrams †]

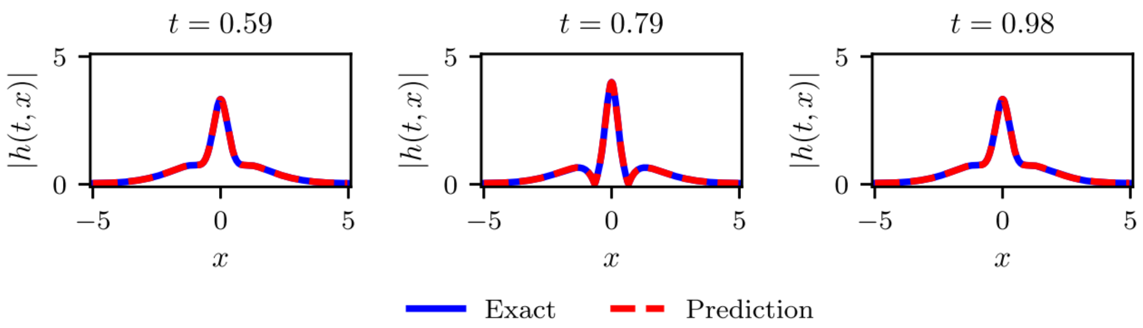
† S. Lin & Y. Chen, *Journal of Computational Physics*, 2022

Models

- **Data-driven solutions**
  - Small number of training dataset, N_u
    - e.g., initial / boundary condition
  - Loss function is optimized using L-BFGS, full-batch.
  - No theoretical guarantee that it converges to a global minimum, but if the PDE has unique solution
    → Accurate prediction with sufficient number of collocation points, N_f



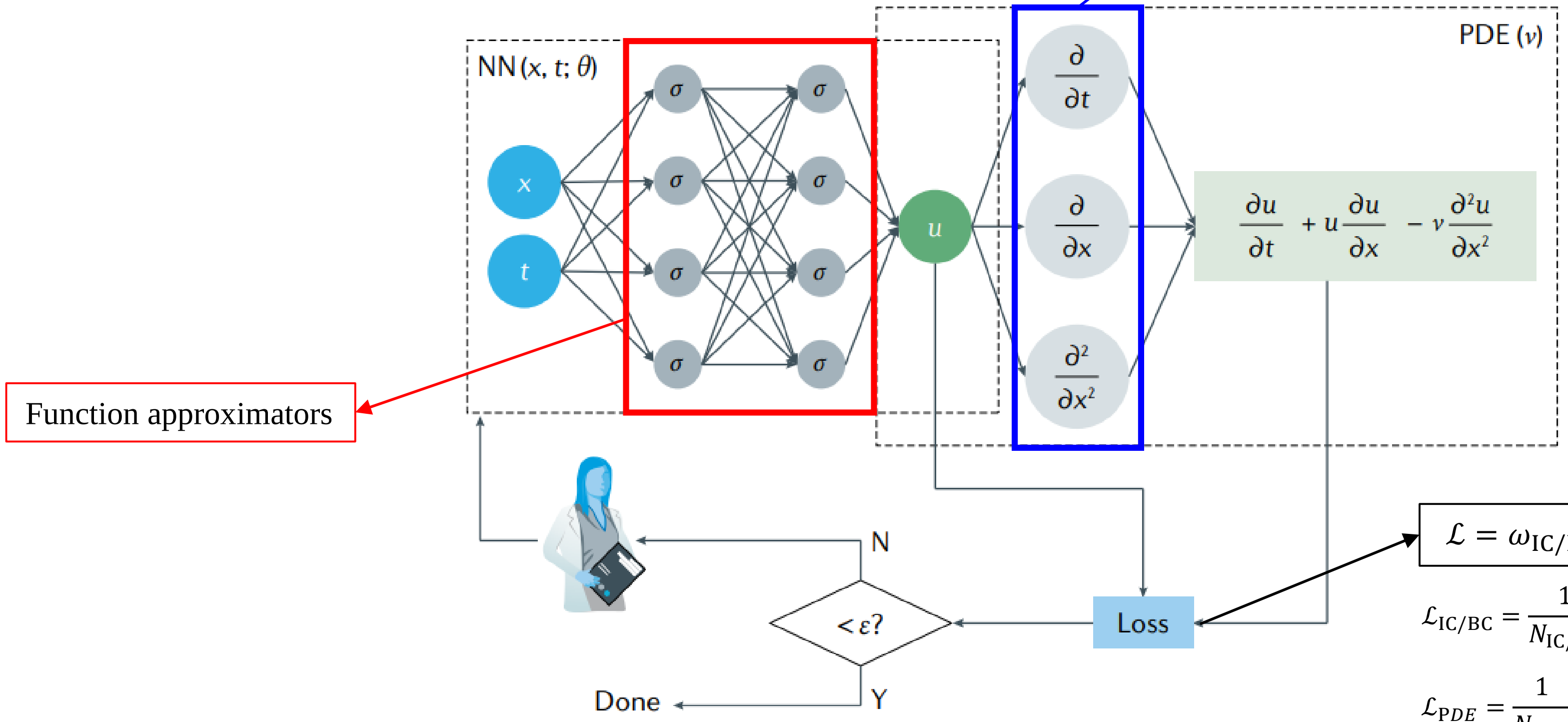
[Predicted solution $|h(t,x)|$]



[Comparison of predicted and exact solution]

Models

- **Data-driven discovery (Inverse problems)**
  - e.g., $u_t + \lambda_1 u_x = \lambda_2 u_{xx}$



$$\mathcal{L} = \omega_{\mathrm{IC/BC}}\mathcal{L}_{\mathrm{IC/BC}} + \omega_{\mathrm{PDE}}\mathcal{L}_{\mathrm{PDE}} + \boldsymbol{\omega_{\mathrm{DATA}}\mathcal{L}_{\mathrm{DATA}}}$$

$$\mathcal{L}_{\mathrm{IC/BC}} = \frac{1}{N_{\mathrm{IC/BC}}}\sum_{i=1}^{N_{\mathrm{IC/BC}}}(u(x_i, t_i) - u_i)^2$$

$$\mathcal{L}_{PDE} = \frac{1}{N_{\mathrm{PDE}}}\sum_{j=1}^{N_{\mathrm{PDE}}}(u_t + uu_x - \nu u_{xx})^2\Big|_{x_j, t_j}$$

$$\boldsymbol{\mathcal{L}_{\mathrm{DATA}} = \frac{1}{N_{\mathrm{DATA}}}\sum_{k=1}^{N_{\mathrm{DATA}}}(u(x_k, t_k) - u_k)^2}$$

[PINN algorithm †]

† G. E. Karniadakis et al., *Nature Reviews Physics*, 2021

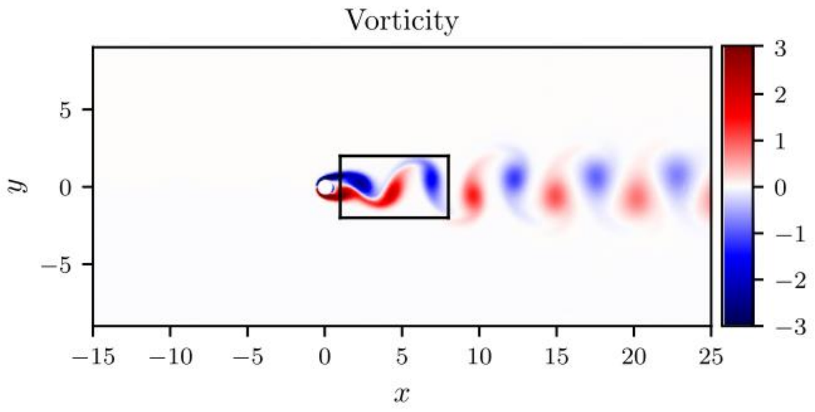
Models

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

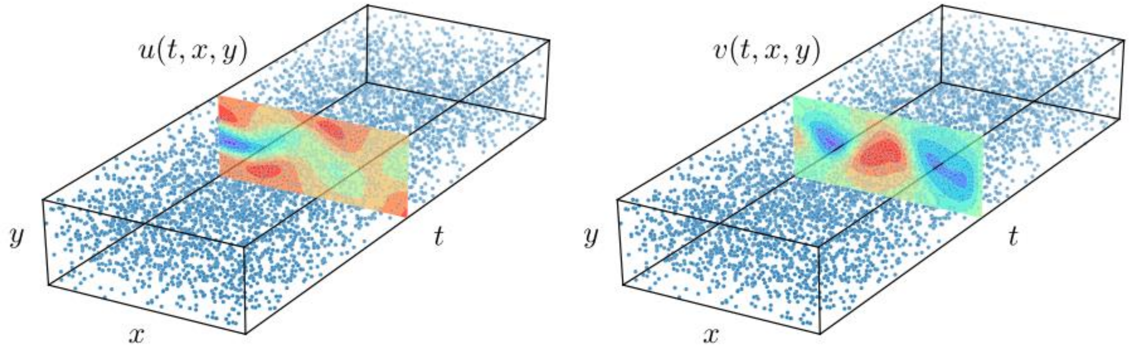
- **Data-driven discovery**
  - e.g., 2D Navier-Stokes Equations given datasets $\{x_i, y_i, u_i, v_i, t_i\}_{i=1}^N$
    - $u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}) \rightarrow f = u_t + \lambda_1(uu_x + vu_y) - p_x - \lambda_2(u_{xx} + u_{yy})$
    - $v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}) \rightarrow g = v_t + \lambda_1(uv_x + vv_y) - p_y - \lambda_2(v_{xx} + v_{yy})$
    - $u_x + v_y = 0$
    - Loss functions $\mathcal{L} = \mathcal{L}_u + \mathcal{L}_v + \mathcal{L}_f + \mathcal{L}_g$
      - $\mathcal{L}_u = \frac{1}{N}\sum_{i=1}^N (u(x_i, y_i, t_i) - u_i)^2$ and $\mathcal{L}_v = \frac{1}{n}\sum_{i=1}^N (v(x_i, y_i, t_i) - v_i)^2$
      - $\mathcal{L}_f = \frac{1}{N}\sum_{i=1}^N f(x_i, y_i, t_i)^2$ and $\mathcal{L}_g = \frac{1}{N}\sum_{i=1}^N g(x_i, y_i, t_i)^2$
    - Scattered and noisy data (u, v) → unknown parameters (λ_1, λ_2) and pressure filed $p(x, y, t)$

Models

- **Data-driven discovery**
  - Larger training dataset, N_u
    - e.g., CFD simulation results and experimental data
  - Loss function is optimized using mini-batch.



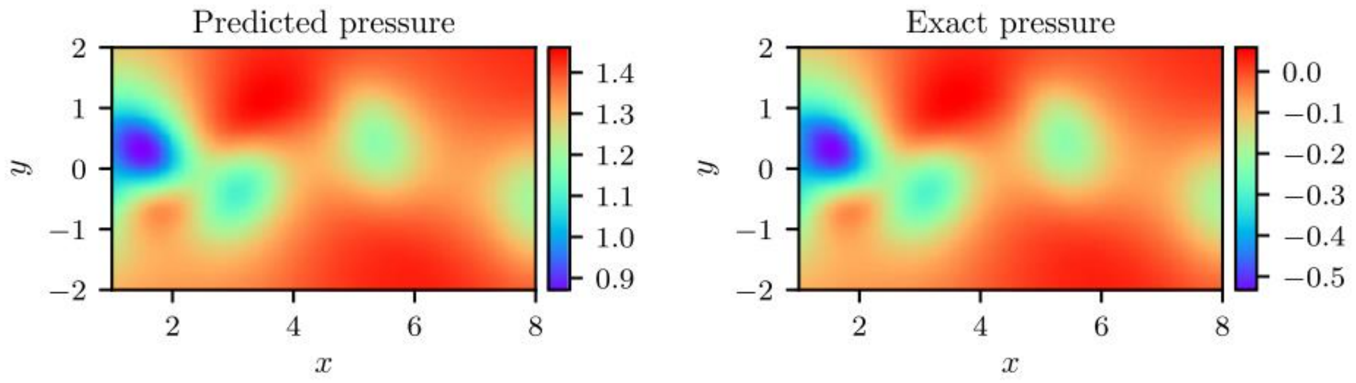
[Simulation results]



[Locations of training data-points for $u(x, y, t)$ and $v(x, y, u)$]

Models

- **Data-driven discovery**
  - Larger training dataset, N_u
    - e.g., CFD simulation results and experimental data
  - Loss function is optimized using mini-batch.



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)	$u_t + 0.998(uu_x + vu_y) = -p_x + 0.01057(u_{xx} + u_{yy})$ $v_t + 0.998(uv_x + vv_y) = -p_y + 0.01057(v_{xx} + v_{yy})$

[Top: comparison of predicted and exact pressure, Bottom: comparison of PDEs]

KAIST

Conclusions

- **Contribution**
  - Physics-informed NN, a new class of universal function approximators that can reflect underlying physical laws is introduced.
  - Two algorithms are suggested.
    - Solutions to general nonlinear PDEs are inferred. (Forward problem)
    - Efficient physics-informed surrogate model is constructed. (Inverse problem)

Conclusions

- **Contribution**

Majority of Real applications

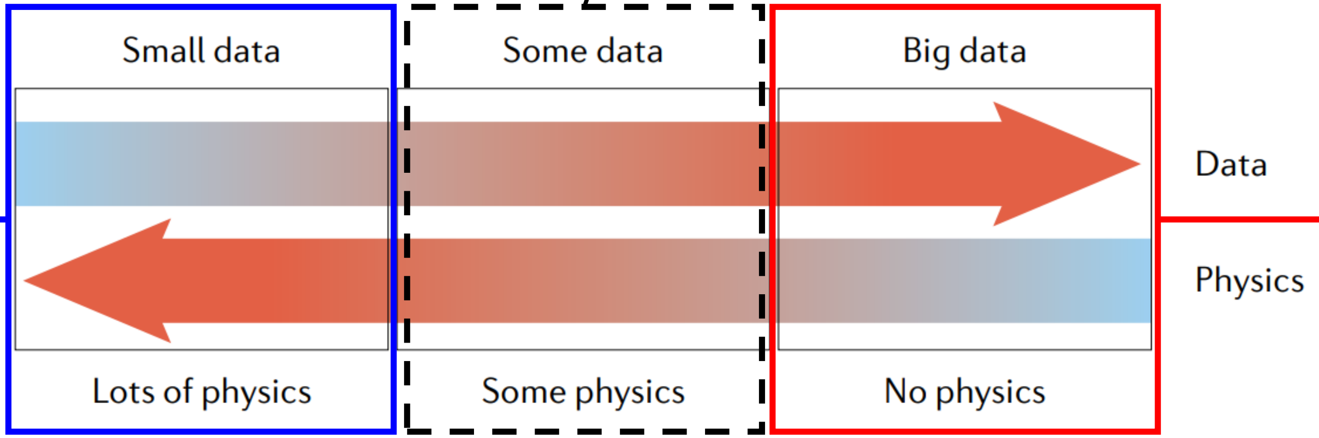
Inference of parameters and missing functional terms in PDE while simultaneously recovering the solution

Inverse problem

Only data available are initial/boundary conditions.

Available many data

Forward problem



Original data-driven approach

Specific governing PDEs and associated parameters are precisely known.

Governing physical law is not be known.

[Relation data and physics †]

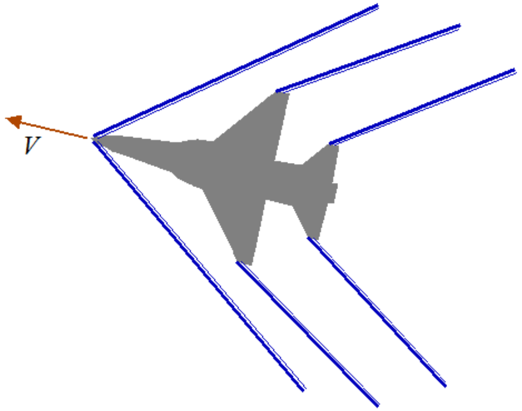
† G. E. Karniadakis et al., *Nature Reviews Physics*, 2021

Conclusions

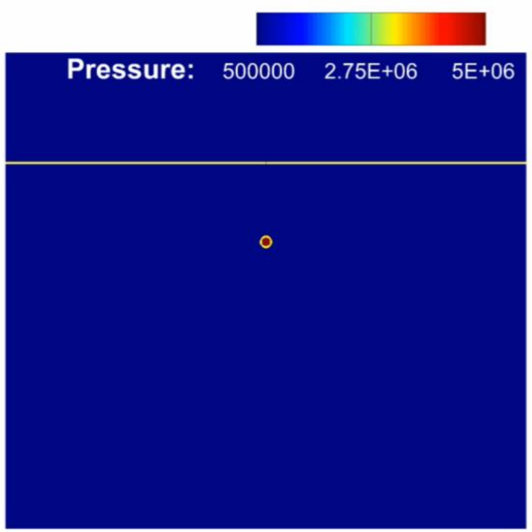
- **Contribution**
  - Physics-informed NN, a new class of universal function approximators that can reflect underlying physical laws is introduced.
  - Two algorithms are suggested.
    - Solutions to general nonlinear PDEs are inferred. (Forward problem)
    - Efficient physics-informed surrogate model is constructed. (Inverse problem)
- **Future works**
  - How deep/wide should the a NN be?, How much data is needed?
  - Why is the algorithms not suffering from local optima for the parameters of the differential operator?
  - Does the network suffer from vanishing gradients for high-order differential operators?,
    Could this be mitigated by using different activation function?
  - Are the MSE and SSE the appropriate loss functions?

General Limitation of PINN

- **Fundamental Issue**
  - Bad convergence at discontinuity point and singularity
    - Weak form: differential equations → integral equations
    - Domain decomposition: multiple sub-domain with separate neural networks
- **Neural Networks Issue**
  - Unbalanced/non-defined loss optimization
    - $\mathcal{L} = \omega_f \mathcal{L}_f + \omega_g \mathcal{L}_g + \omega_h \mathcal{L}_h + \cdots + \omega_{\mathrm{IC}} \mathcal{L}_{\mathrm{IC}}$
      - Normalization
      - Adaptive loss weights



[Diagram in case of aerospace]



[Numerical simulation of underwater explosions]

Thank you