Are physics-math-based iteration methods and physics-informed neural network methods in static magnetic field calculation more different or more similar?

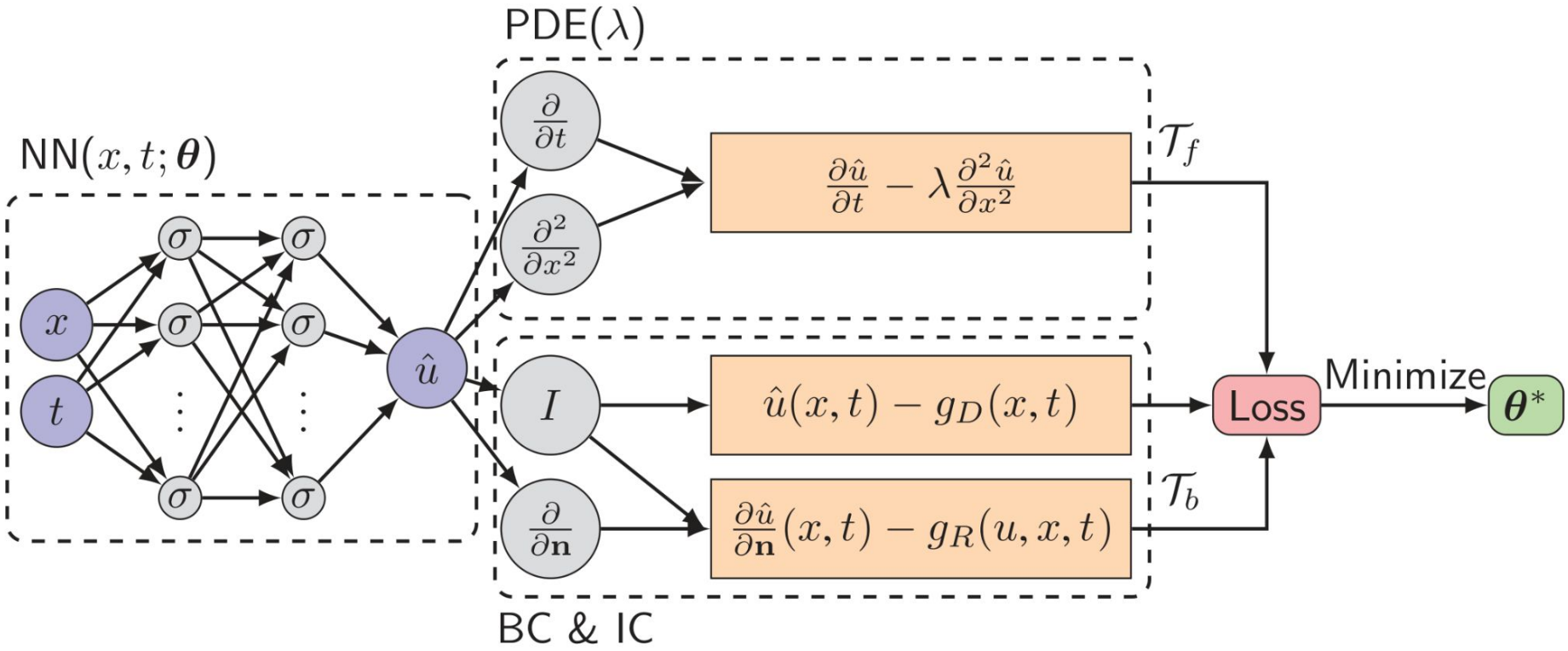
Mingyu Jeon[2], G. S. Choe[1,2], Sibaek Yi[2], Minseon Lee[1]

[1]School of Space Research, Kyung Hee University
[2]Department of Astronomy and Space Science, Kyung Hee University

Physics-informed neural networks (PINNs)

Procedure 2.1 The PINN algorithm for solving differential equations.

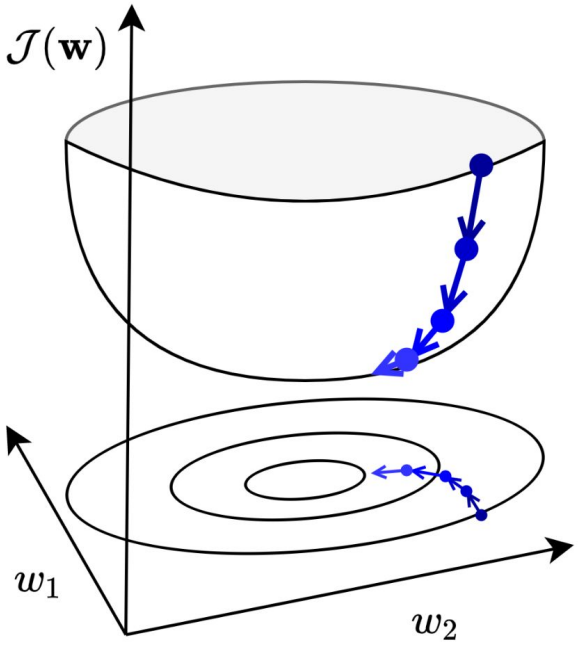
Step 1	Construct a neural network $\hat{u}(\mathbf{x};\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$.
Step 2	Specify the two training sets $\mathcal{T}_f$ and $\mathcal{T}_b$ for the equation and boundary/initial conditions.
Step 3	Specify a loss function by summing the weighted L^2 norm of both the PDE equation and boundary condition residuals.
Step 4	Train the neural network to find the best parameters $\boldsymbol{\theta}^*$ by minimizing the loss function $\mathcal{L}(\boldsymbol{\theta};\mathcal{T})$.



Lu et al. 2021

Gradient descent

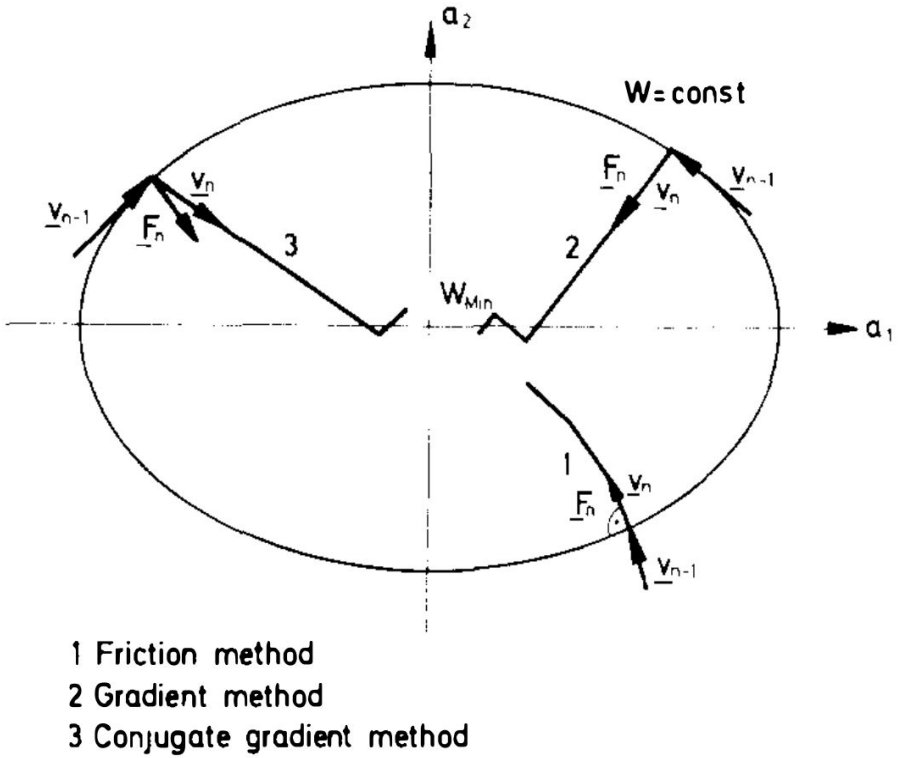
$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} \mathcal{J}$$



Grosse et al. 2021

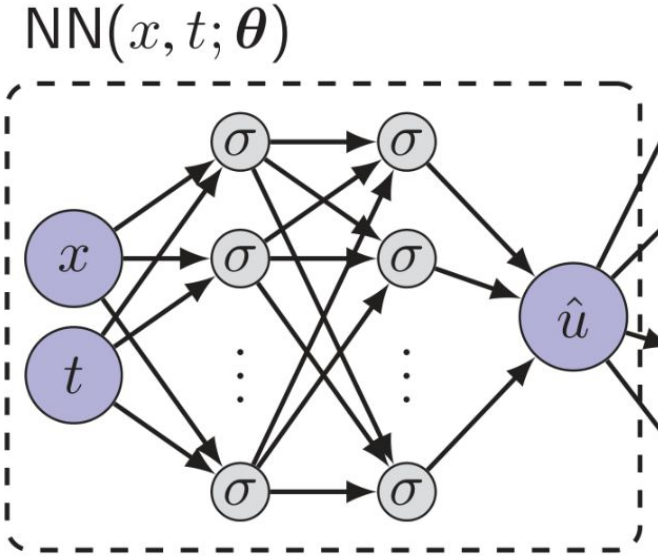
A 3D Code for MHD Equilibrium and Stability

R. CHODURA AND A. SCHLÜTER



Chodura et al. 1981

Automatic differentiation



$$z = wx + b$$

$$\hat{u} = \sigma(z)$$

$$\frac{\partial \hat{u}}{\partial z} = \sigma'(z)$$

$$\frac{\partial \hat{u}}{\partial x} = \frac{\partial z}{\partial x}\frac{\partial \hat{u}}{\partial z}$$

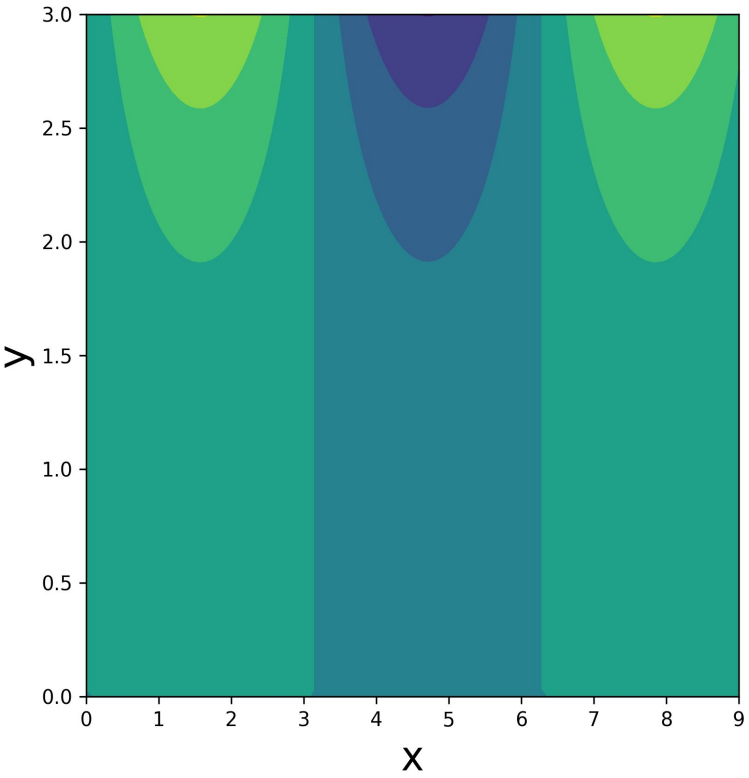
$$= w\sigma'(z)$$

PINN example: 2D Laplace equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad \Longrightarrow \qquad u(x, y) = \frac{\sin x}{\sin 9} \frac{\sinh y}{\sinh 3}$$

$$u(x,\, 3) = \frac{\sin x}{\sin 9}$$

$u(0,\, y) = 0$

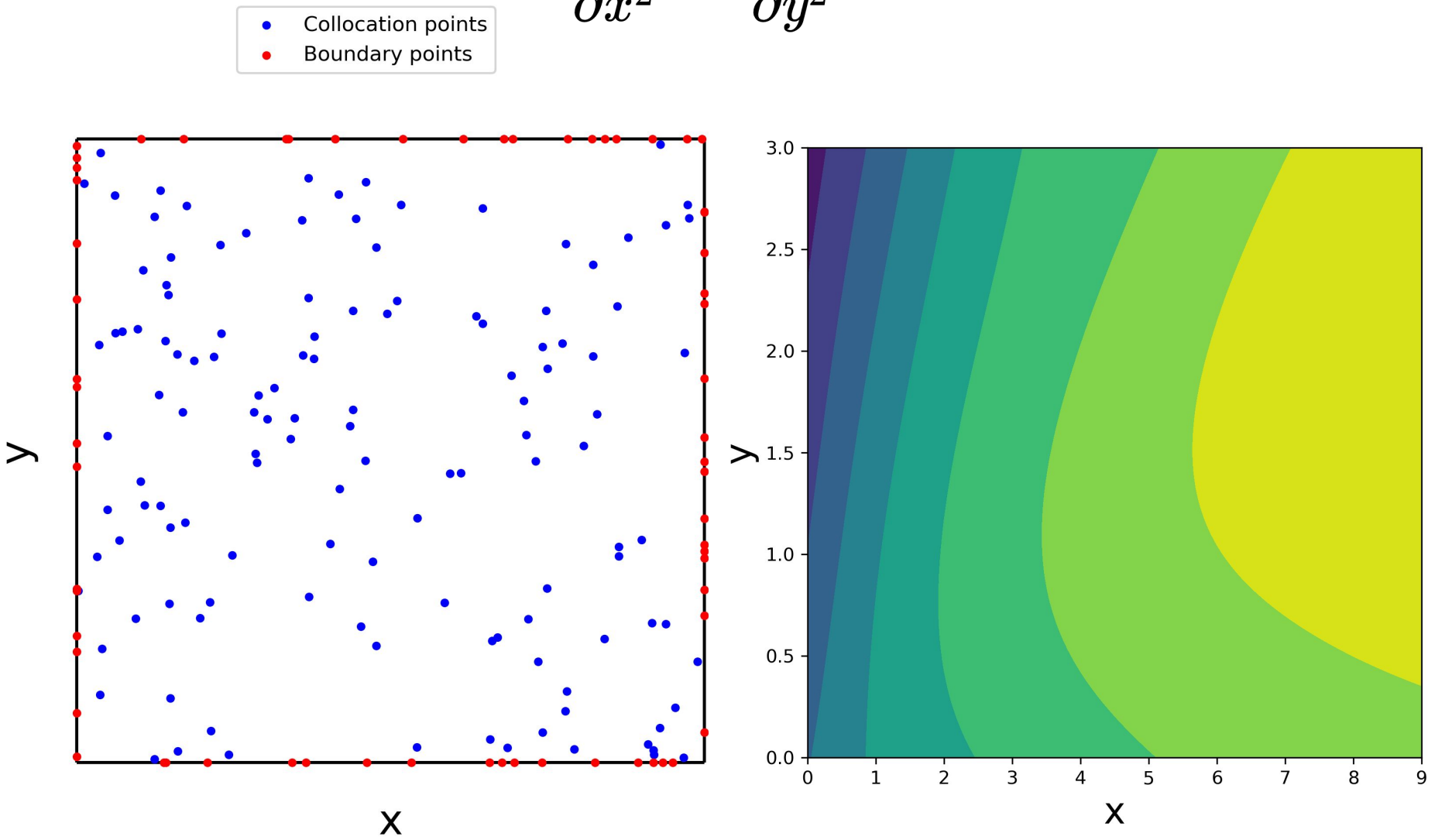


$u(9,\, y) = \dfrac{\sinh y}{\sinh 3}$

$$u(x,\, 0) = 0$$

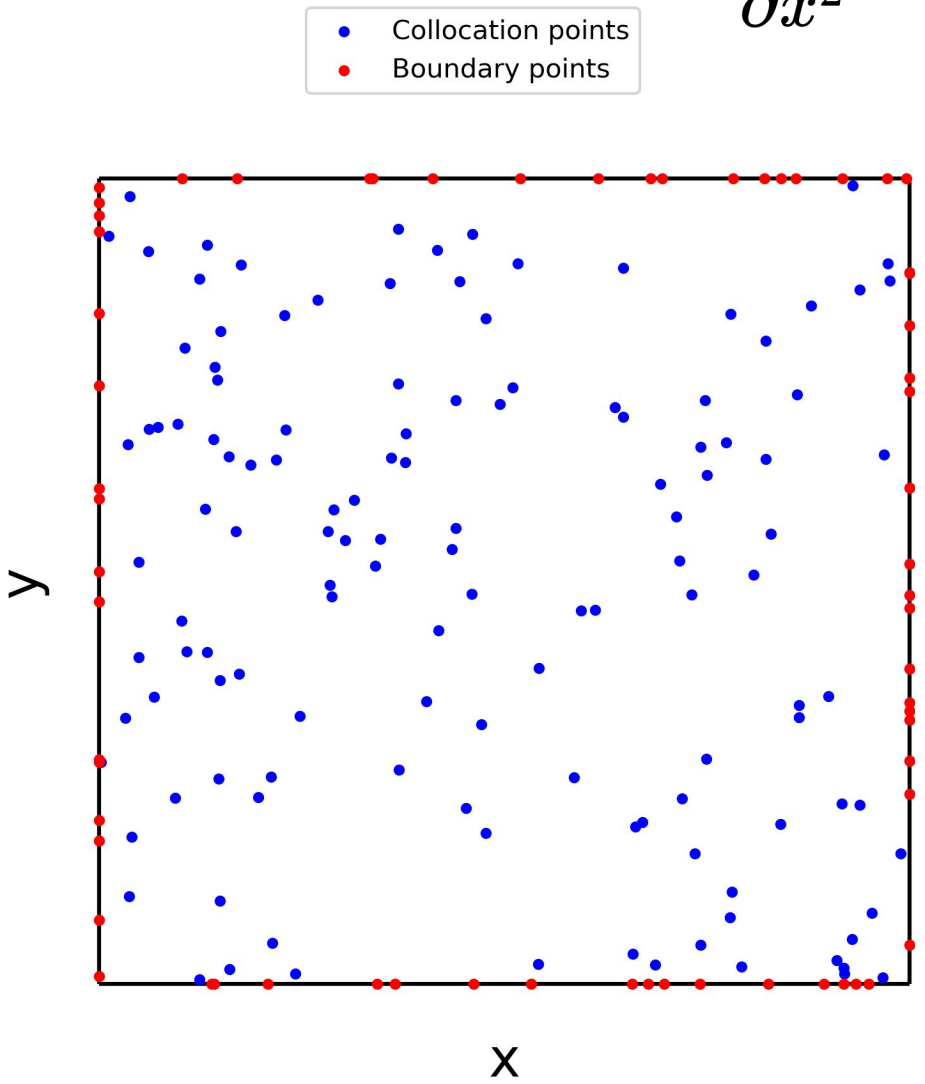
PINN example: 2D Laplace equation

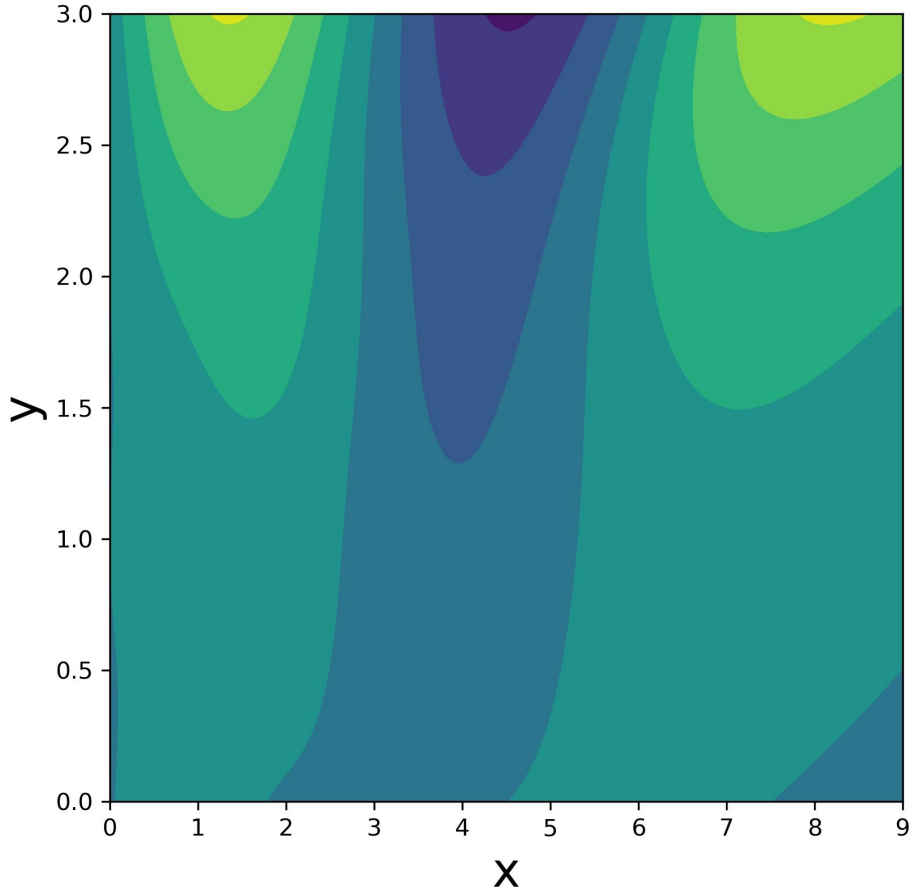
$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$



PINN example: 2D Laplace equation

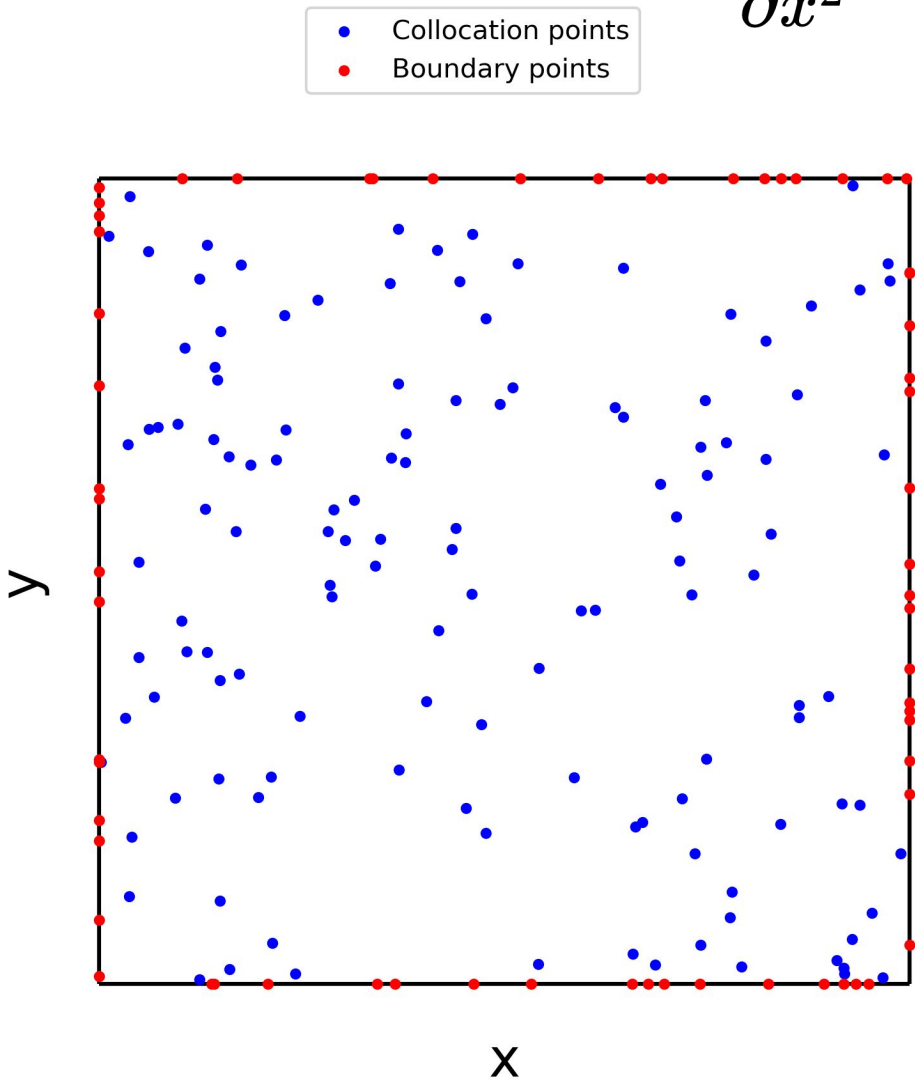
$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

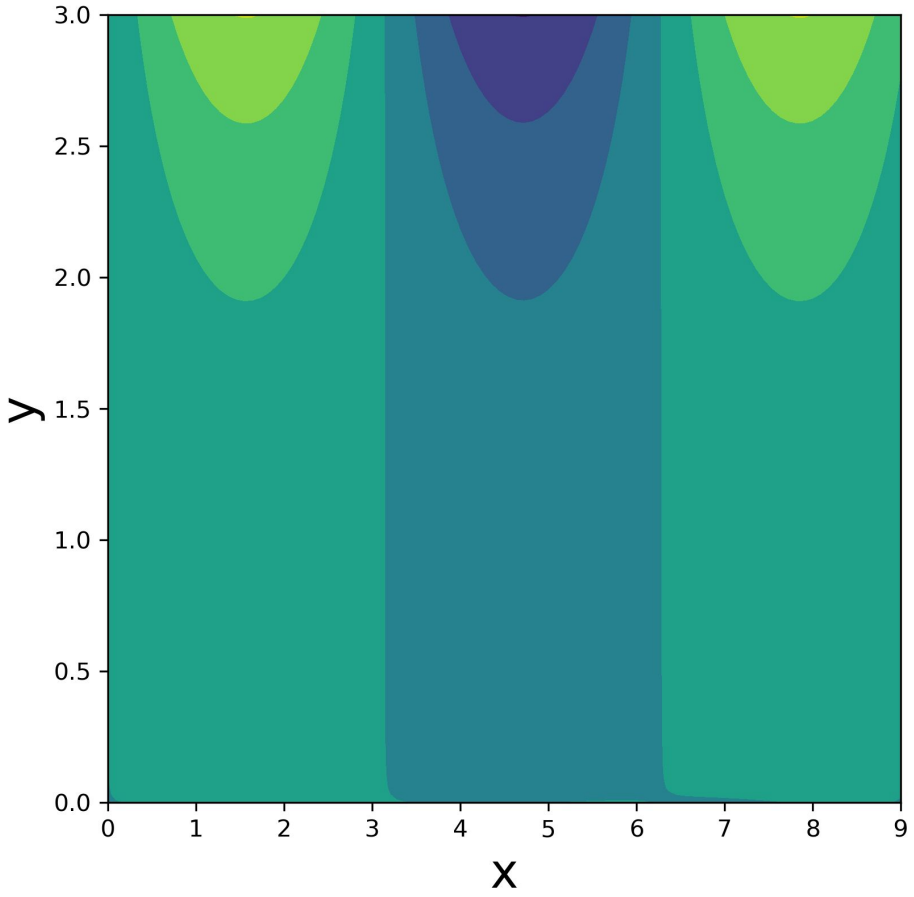




PINN example: 2D Laplace equation

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





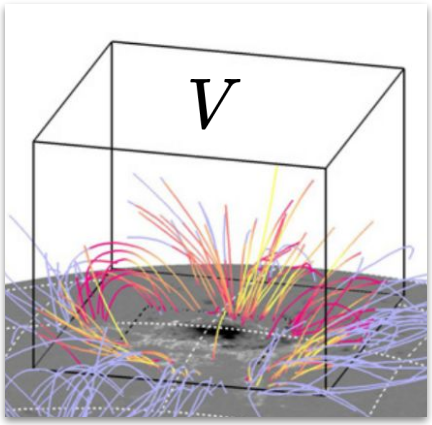
Optimization Method for Force-Free Fields

Wheatland et al. 2000

We have to solve $\mathbf{J} \times \mathbf{B} = \mathbf{0}$ with $\nabla \cdot \mathbf{B} = 0$.

$$L = \int_V \left[\frac{|\mathbf{J} \times \mathbf{B}|^2}{B^2} + |\nabla \cdot \mathbf{B}|^2 \right] \mathrm{d}V.$$

$L \to 0$ as the field becomes force-free.



DeRosa et al. 2009

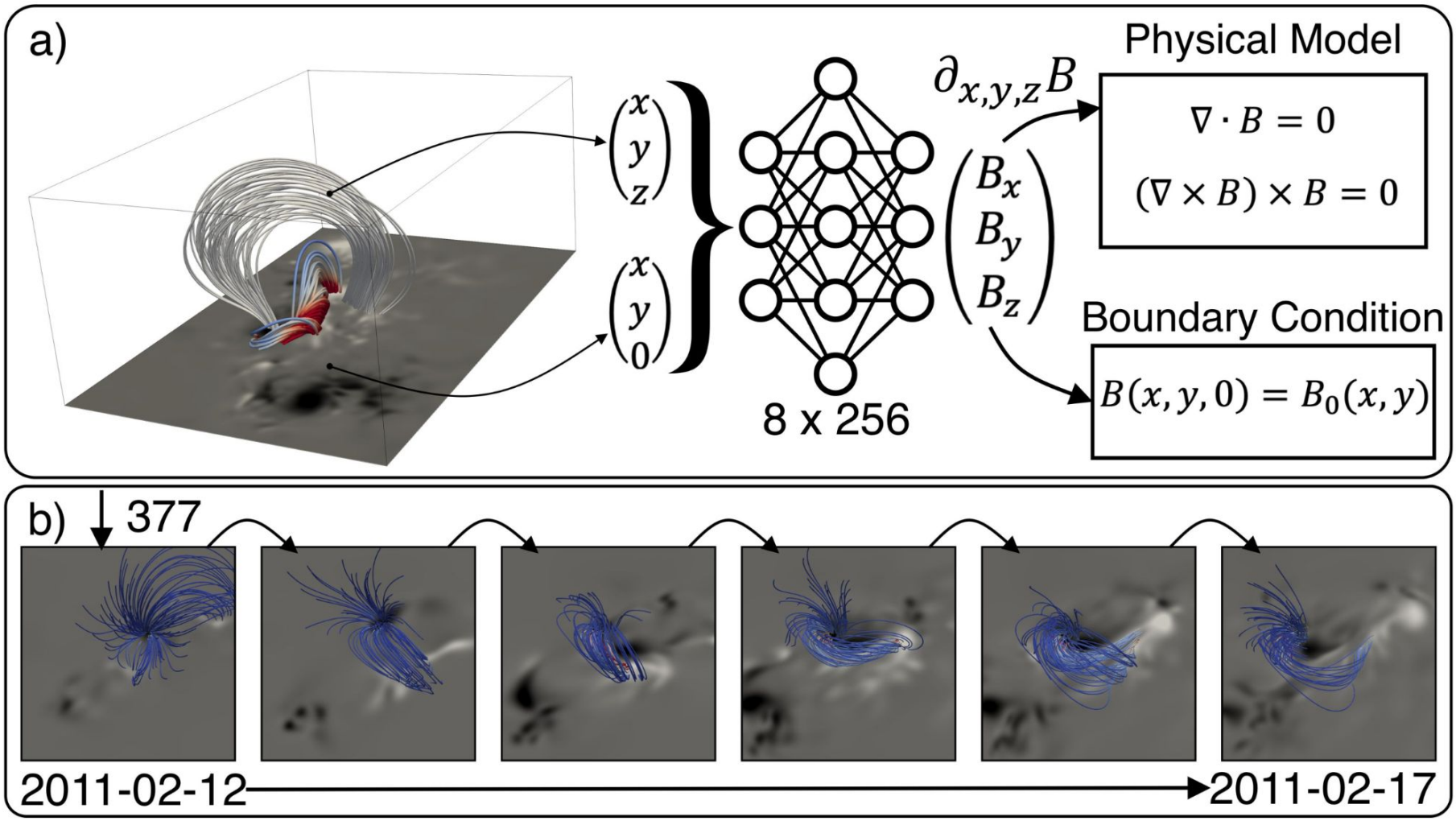
$$\frac{1}{2}\frac{dL}{dt} = -\int_V \frac{\partial \mathbf{B}}{\partial t} \cdot \tilde{\mathbf{F}} \mathrm{d}V - \int_{S=\partial V} \frac{\partial \mathbf{B}}{\partial t} \cdot \tilde{\mathbf{G}} \mathrm{d}S.$$

Here t is a time-like parameter increasing with the iteration steps.

$$\text{If } \frac{\partial \mathbf{B}}{\partial t} = \mu \tilde{\mathbf{F}} \ (\mu > 0) \text{ in } V \text{ and if } \frac{\partial \mathbf{B}}{\partial t} = 0 \text{ at } \partial V, \ \frac{dL}{dt} < 0.$$

The route toward the solution is clear and transparent.

PINNs for Force-Free Fields



Jarolim et al. 2022

Forward problem and Inverse problem

$$\frac{\partial u}{\partial t} + \lambda_1 u \frac{\partial u}{\partial x} - \lambda_2 \frac{\partial^2 u}{\partial x^2} = 0$$

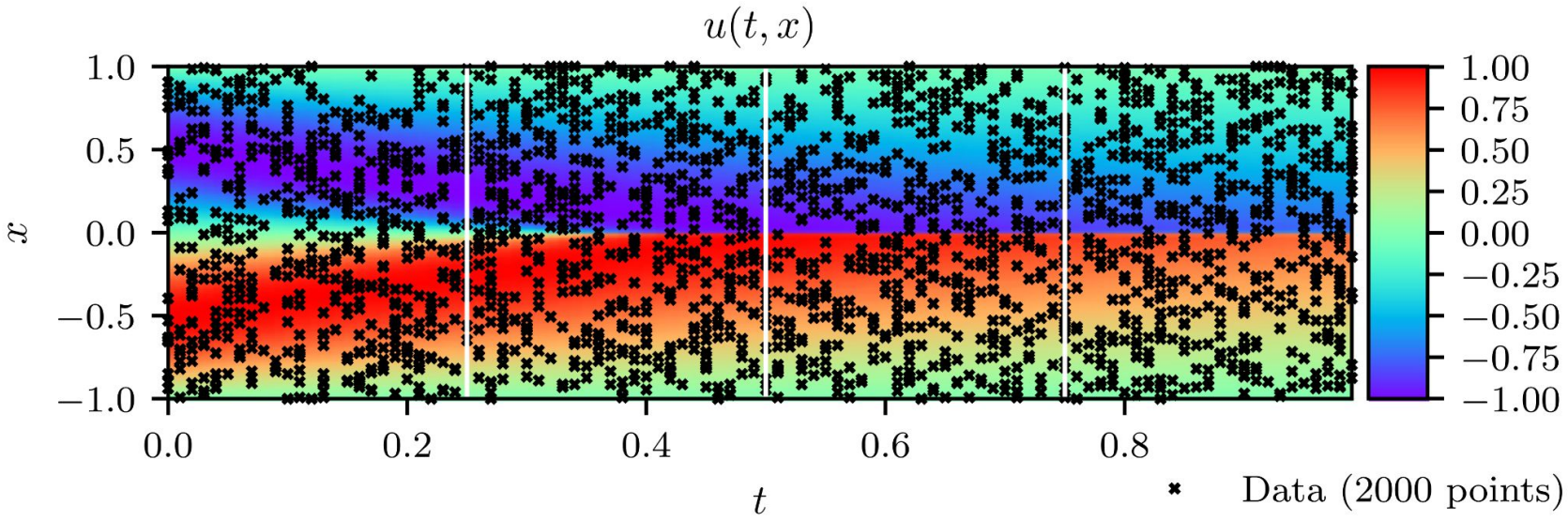
$$\text{What is } u? \quad \Longrightarrow \quad \text{Forward problem}$$

$$\text{What are } \lambda_1,\ \lambda_2? \quad \Longrightarrow \quad \text{Inverse problem}$$

Inverse problem

$$\frac{\partial u}{\partial t} + \lambda_1 u \frac{\partial u}{\partial x} - \lambda_2 \frac{\partial^2 u}{\partial x^2} = 0$$

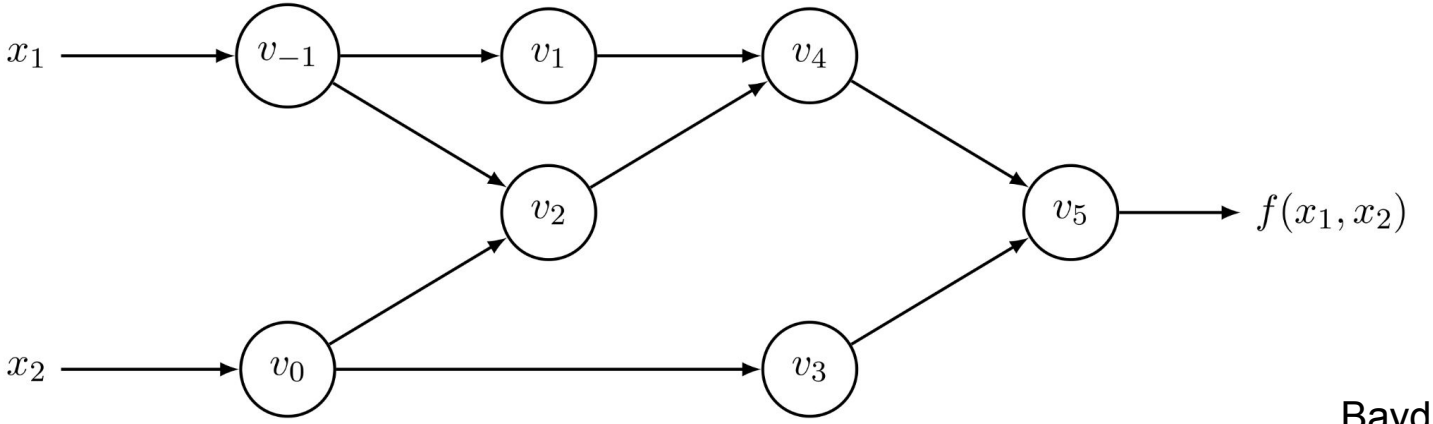
Correct PDE	$u_t + uu_x - 0.0031831u_{xx} = 0$
Identified PDE (clean data)	$u_t + 0.99915uu_x - 0.0031794u_{xx} = 0$
Identified PDE (1% noise)	$u_t + 1.00042uu_x - 0.0032098u_{xx} = 0$



Raissi et al. 2019

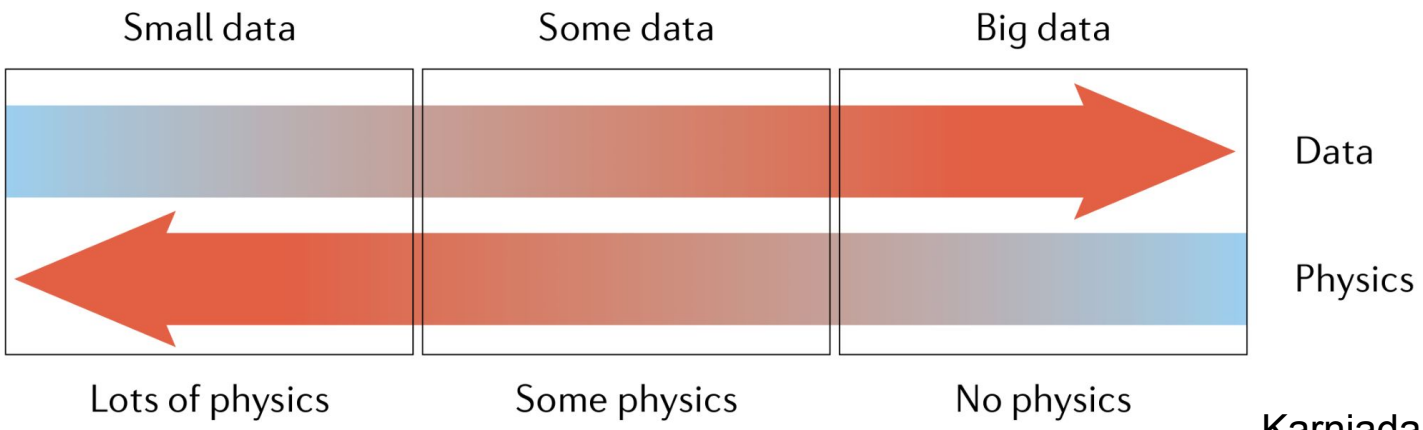
Two Elements of PINNs

Numerical methods with automatic differentiation



Baydin et al. 2018

Machine learning techniques with physical constraints



Karniadakis et al. 2021

Conclusion

- Recently, **physics-informed neural networks (PINNs)** are proposed as another method to find **numerical solutions of differential equations**.

- The overall strategy of PINNs is not quite different from that of traditional numerical methods in that both methods pursue paths **leading to extrema in certain optimization problems**.

- The crux of PINNs is the **automatic differentiation** (algorithmic differentiation), which does not require local differencing of variables, but involves their values at all collocation points.

- **PINNs do not "learn" from any provided solutions.** The role of the provided data for training in conventional machine learning is taken by boundary and/or initial conditions in PINNs.

Thank You
For Your Attention

Automatic differentiation (Algorithmic diff.)

Consider a Function $F : \mathbb{R}^2 \to \mathbb{R}, \;\; F(x, y) = 3(x+y)^2 + \sin(xy).$

What are $\dfrac{\partial F}{\partial x}$ and $\dfrac{\partial F}{\partial y}$ for specific values of x and y?

$$
\begin{aligned}
f_1(x, y) &= x + y, & f_4(x, y) &= x \times y, \\
f_2(f_1) &= f_1^2, & f_5(f_4) &= \sin f_4 \\
f_3(f_2) &= 3 \times f_2, & & \\
& & f_6(f_3, f_5) &= f_3 + f_5.
\end{aligned}
$$

Thus, $F(x, y) = f_6(f_3(f_2(f_1(x, y))), f_5(f_4(x, y)))$ and

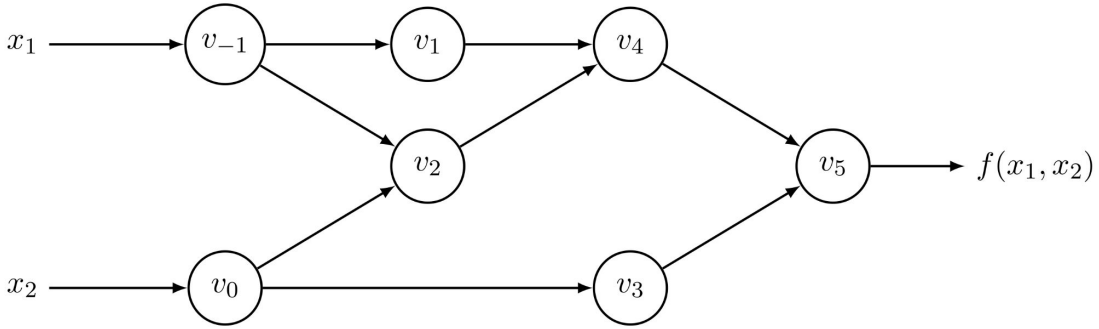
$$
\begin{aligned}
\frac{\partial F}{\partial x} &= \frac{\partial f_6}{\partial f_3}\frac{df_3}{df_2}\frac{df_2}{df_1}\frac{\partial f_1}{\partial x} + \frac{\partial f_6}{\partial f_5}\frac{df_5}{df_1}\frac{\partial f_4}{\partial x} \\
&= 1 \cdot 3 \cdot (2f_1) \cdot 1 + 1 \cdot (\cos f_4) \cdot y = 6(x+y) + y\cos(xy).
\end{aligned}
$$

We can evaluate $\partial F / \partial y$ in a similar way.

Here x and y are <u>not symbols</u>, but specific numbers.

We just make the computer know the derivative functions of the intrinsic operations (+, −, *, /, **) and functions (sin, cos, exp, etc.) built into it.

Automatic differentiation - Forward mode



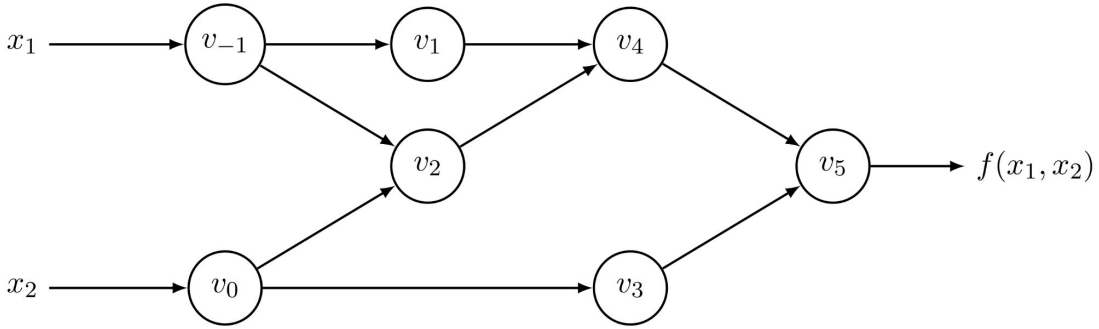
$$\dot{v}_i = \frac{\partial v_i}{\partial x_1}$$

Table 2: Forward mode AD example, with $y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$ evaluated at $(x_1, x_2) = (2, 5)$ and setting $\dot{x}_1 = 1$ to compute $\frac{\partial y}{\partial x_1}$. The original forward evaluation of the primals on the left is augmented by the tangent operations on the right, where each line complements the original directly to its left.

Forward Primal Trace			Forward Tangent (Derivative) Trace		
v_{-1}	$= x_1$	$= 2$	$\dot{v}_{-1}$	$= \dot{x}_1$	$= 1$
v_0	$= x_2$	$= 5$	$\dot{v}_0$	$= \dot{x}_2$	$= 0$
v_1	$= \ln v_{-1}$	$= \ln 2$	$\dot{v}_1$	$= \dot{v}_{-1}/v_{-1}$	$= 1/2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$	$\dot{v}_2$	$= \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$	$= 1 \times 5 + 0 \times 2$
v_3	$= \sin v_0$	$= \sin 5$	$\dot{v}_3$	$= \dot{v}_0 \times \cos v_0$	$= 0 \times \cos 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$	$\dot{v}_4$	$= \dot{v}_1 + \dot{v}_2$	$= 0.5 + 5$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$	$\dot{v}_5$	$= \dot{v}_4 - \dot{v}_3$	$= 5.5 - 0$
y	$= v_5$	$= 11.652$	$\dot{y}$	$= \dot{v}_5$	$= 5.5$

Baydin et al. 2018

Automatic differentiation - Reverse mode



$$\bar{v}_i = \frac{\partial y_j}{\partial v_i}$$

Table 3: Reverse mode AD example, with $y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$ evaluated at $(x_1, x_2) = (2, 5)$. After the forward evaluation of the primals on the left, the adjoint operations on the right are evaluated in reverse (cf. Figure 1). Note that both $\frac{\partial y}{\partial x_1}$ and $\frac{\partial y}{\partial x_2}$ are computed in the same reverse pass, starting from the adjoint $\bar{v}_5 = \bar{y} = \frac{\partial y}{\partial y} = 1$.

Forward Primal Trace		
$v_{-1} = x_1$		$= 2$
$v_0 = x_2$		$= 5$
$v_1 = \ln v_{-1}$		$= \ln 2$
$v_2 = v_{-1} \times v_0$		$= 2 \times 5$
$v_3 = \sin v_0$		$= \sin 5$
$v_4 = v_1 + v_2$		$= 0.693 + 10$
$v_5 = v_4 - v_3$		$= 10.693 + 0.959$
$y = v_5$		$= 11.652$

Reverse Adjoint (Derivative) Trace		
$\bar{x}_1 = \bar{v}_{-1}$		$= \mathbf{5.5}$
$\bar{x}_2 = \bar{v}_0$		$= \mathbf{1.716}$
$\bar{v}_{-1} = \bar{v}_{-1} + \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}}$	$= \bar{v}_{-1} + \bar{v}_1 / v_{-1}$	$= 5.5$
$\bar{v}_0 = \bar{v}_0 + \bar{v}_2 \frac{\partial v_2}{\partial v_0}$	$= \bar{v}_0 + \bar{v}_2 \times v_{-1}$	$= 1.716$
$\bar{v}_{-1} = \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}}$	$= \bar{v}_2 \times v_0$	$= 5$
$\bar{v}_0 = \bar{v}_3 \frac{\partial v_3}{\partial v_0}$	$= \bar{v}_3 \times \cos v_0$	$= -0.284$
$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_1 = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$	$= \bar{v}_5 \times (-1)$	$= -1$
$\bar{v}_4 = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$	$= \bar{v}_5 \times 1$	$= 1$
$\bar{v}_5 = \bar{y}$		$= 1$

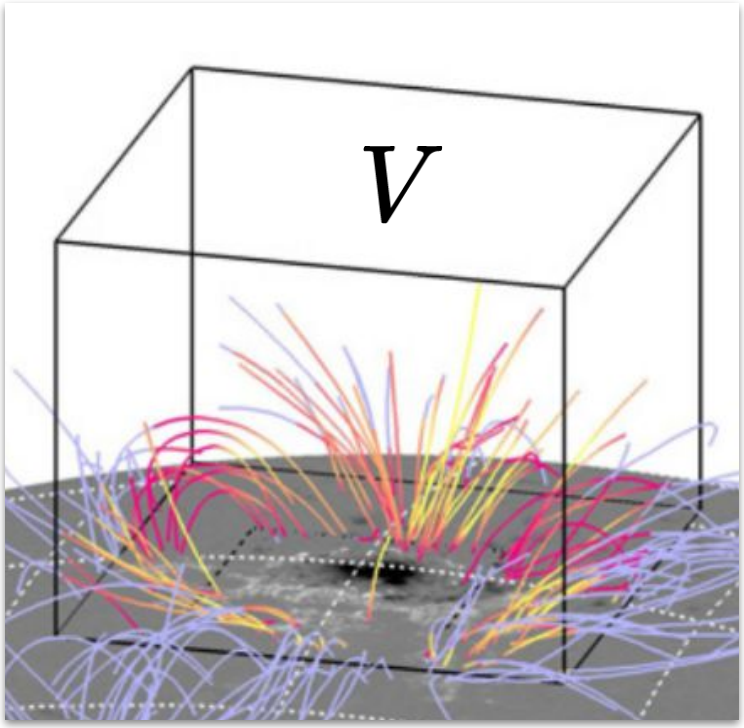
Baydin et al. 2018

Techniques to calculate derivatives

Technique	Advantage(s)	Drawback(s)
Hand-coded analytical derivative = Manual differentiation	Exact and often fastest method.	Time consuming to code, error prone, and not applicable to problems with implicit solutions. Not automated.
Finite differentiation = Numerical differentiation	Easy to code.	Subject to floating point precision errors and slow, especially in high dimensions, as the method requires at least D evaluations, where D is the number of partial derivatives required.
Symbolic differentiation	Exact, generates symbolic expressions.	Memory intensive and slow. Cannot handle statements such as unbounded loops.
Automatic differentation	Exact, speed is comparable to hand-coding derivatives, highly applicable.	Needs to be carefully implemented, alhough this is already done in several packages.

Margossian 2019

Optimization Method for Force-Free Fields



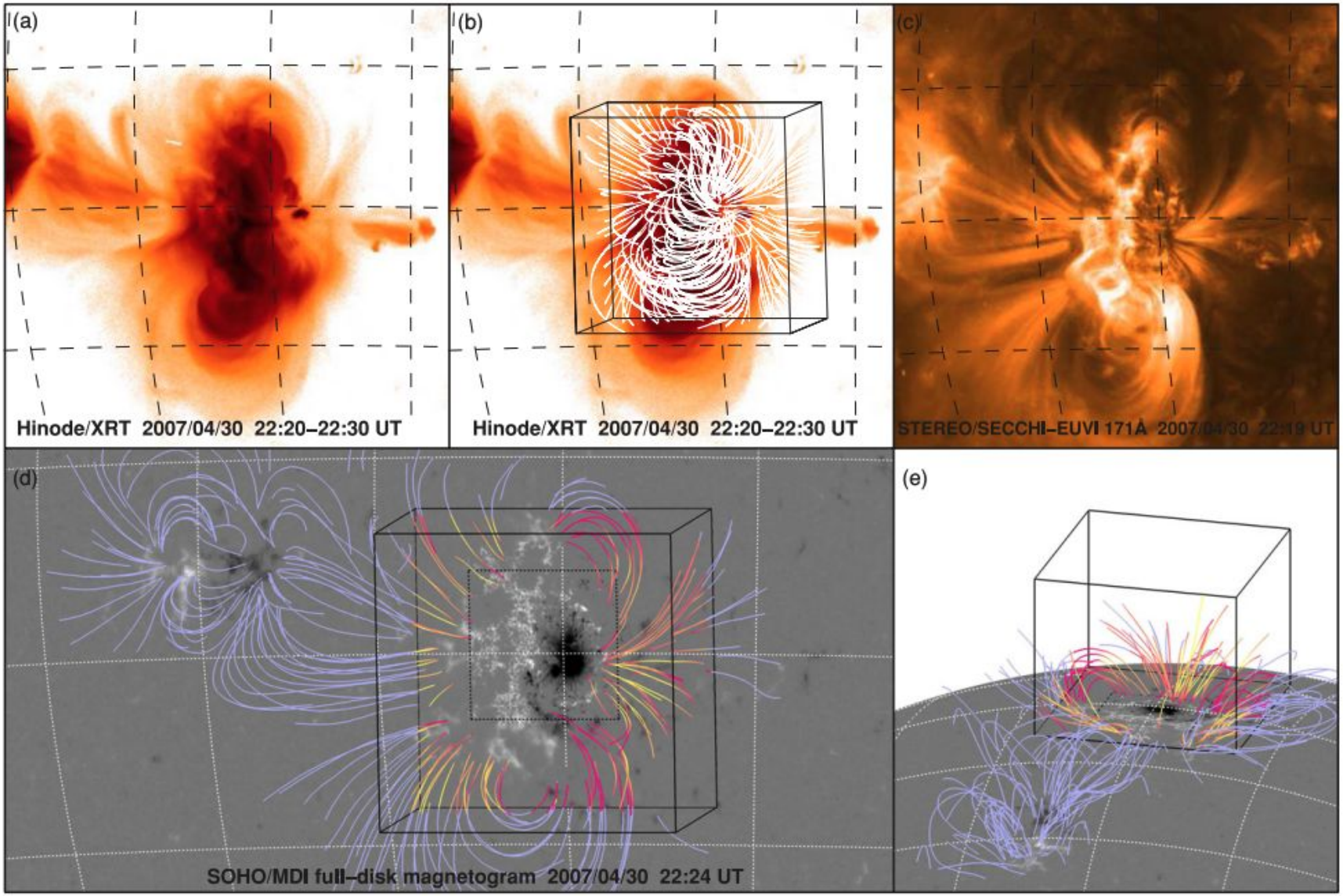
DeRosa et al. 2009

$$(\nabla \times \mathbf{B}) \times \mathbf{B} = \mathbf{0}$$
$$\nabla \cdot \mathbf{B} = 0 \qquad \text{in } V$$

$$L = \int_V \left[\frac{|(\nabla \times \mathbf{B}) \times \mathbf{B}|^2}{B^2} + |\nabla \cdot \mathbf{B}|^2 \right] \mathrm{d}V$$

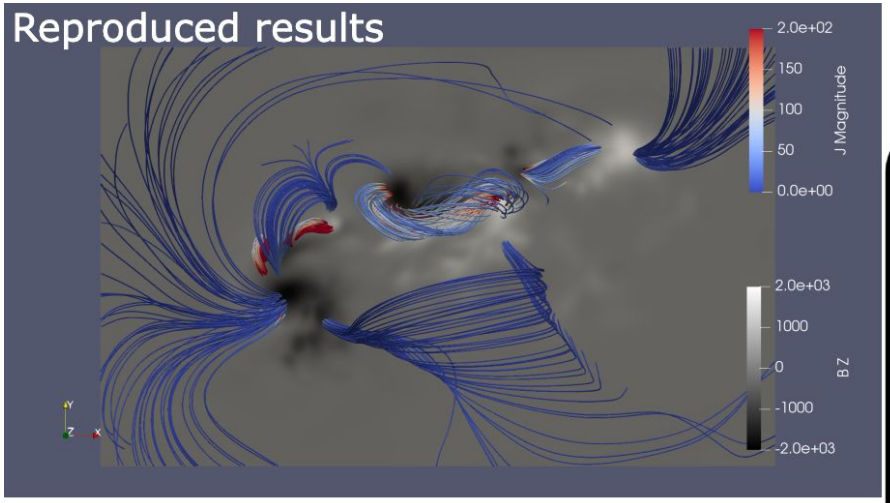
Wheatland et al. 2000

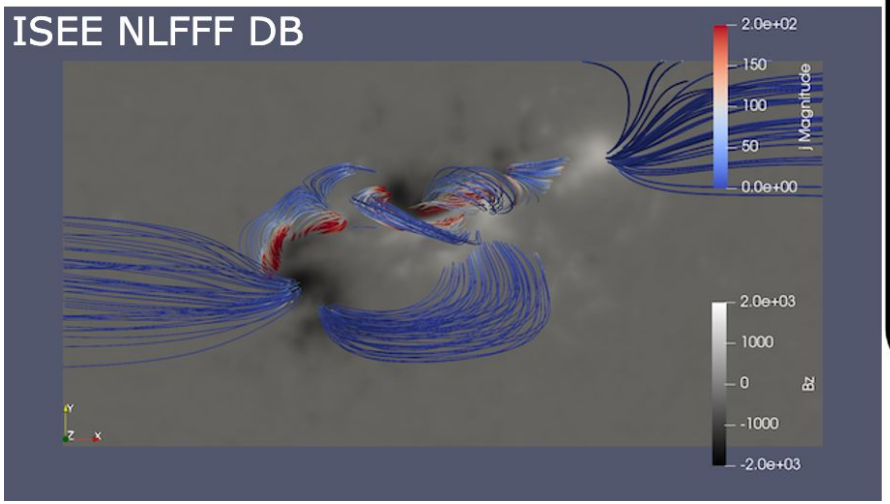
Nonlinear force-free fields (NLFFFs)



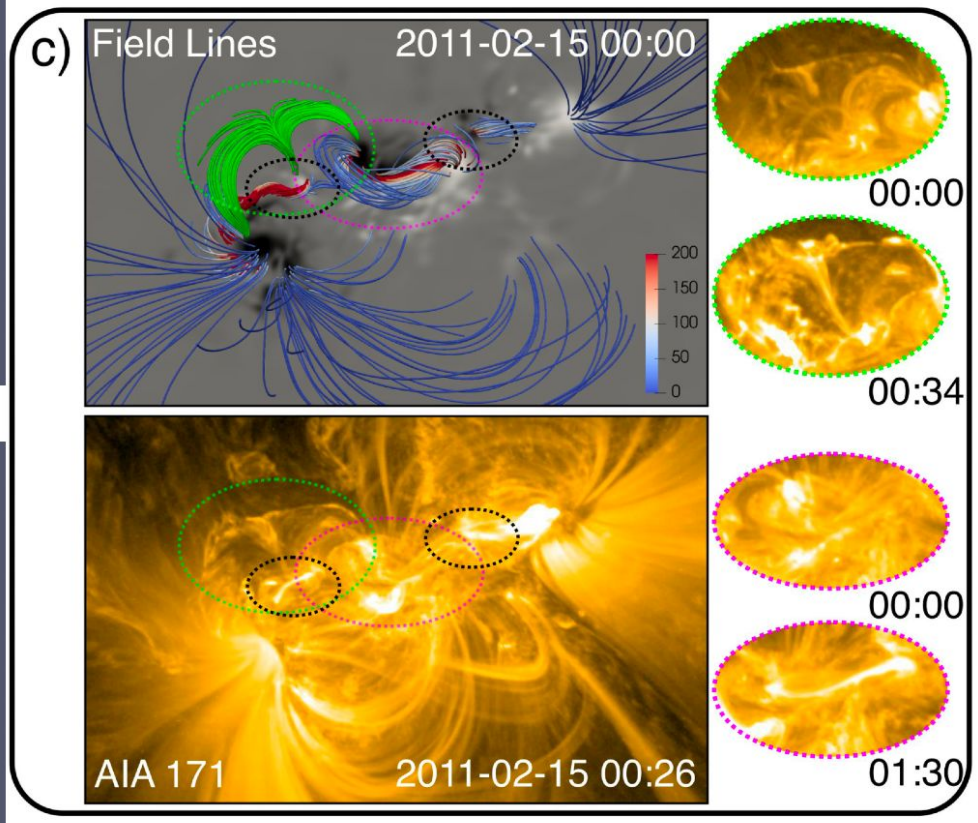
DeRosa et al. 2009

PINNs for Force-Free Fields

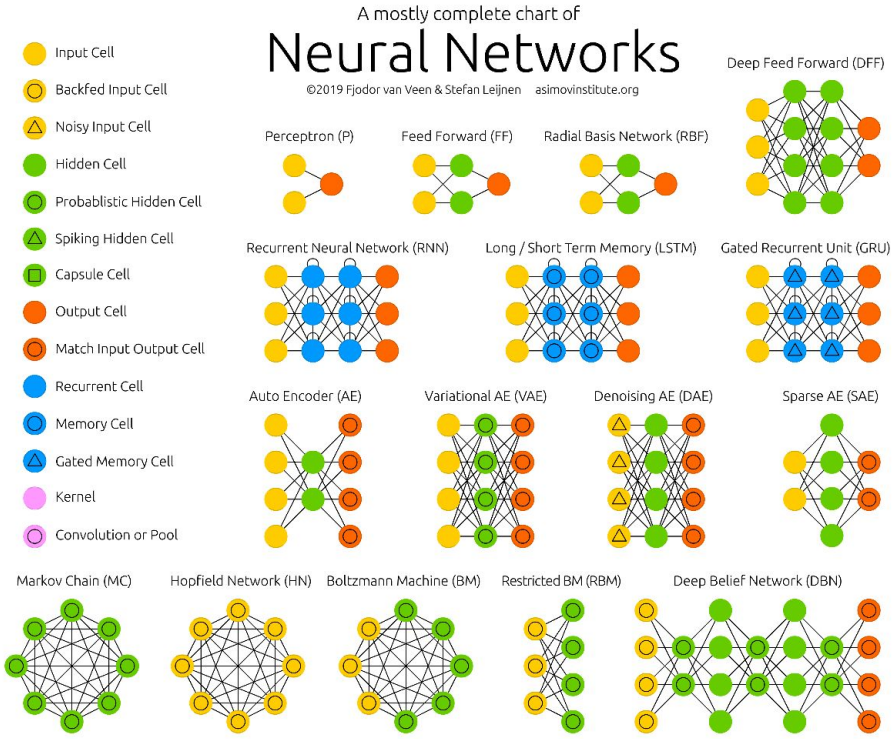


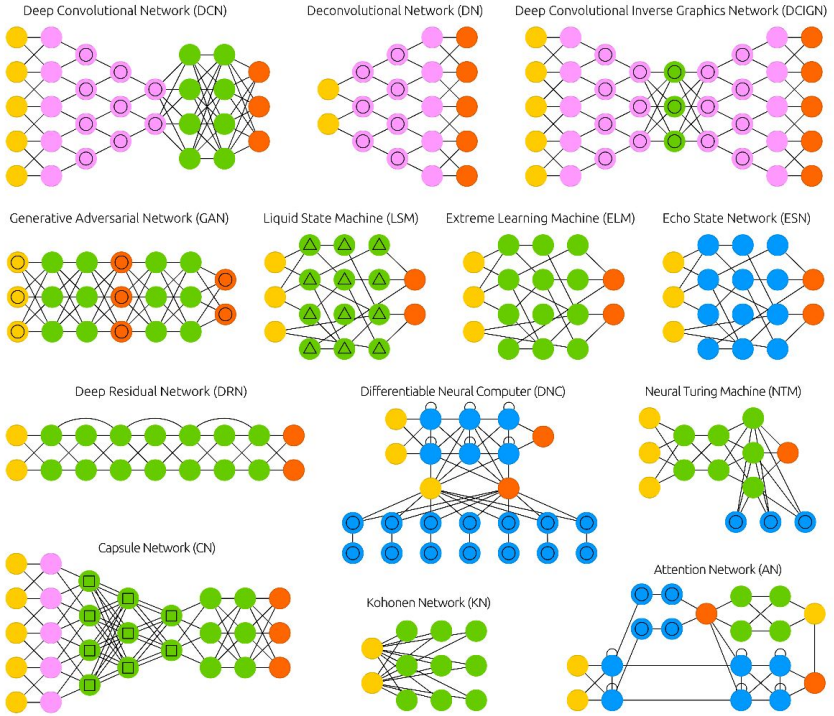


Paper results



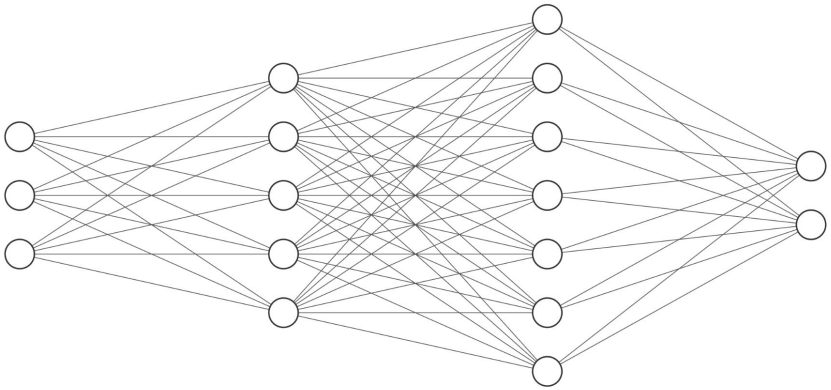
Neural Networks





https://www.asimovinstitute.org/neural-network-zoo/

Feedforward neural network (FNN) = Multilayer perceptron (MLP)



Let $\mathcal{N}^L(\mathbf{x}) : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ be an L-layer neural network, or an $(L-1)$-hidden layer neural network, with N_ℓ neurons in the ℓth layer ($N_0 = d_{\text{in}}$, $N_L = d_{\text{out}}$). Let us denote the weight matrix and bias vector in the ℓth layer by $\boldsymbol{W}^\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and $\mathbf{b}^\ell \in \mathbb{R}^{N_\ell}$, respectively. Given a nonlinear activation function σ, which is applied elementwisely, the FNN is recursively defined as follows:

$$\begin{aligned}
\text{input layer:} \quad & \mathcal{N}^0(\mathbf{x}) = \mathbf{x} \in \mathbb{R}^{d_{\text{in}}}, \\
\text{hidden layers:} \quad & \mathcal{N}^\ell(\mathbf{x}) = \sigma(\boldsymbol{W}^\ell \mathcal{N}^{\ell-1}(\mathbf{x}) + \boldsymbol{b}^\ell) \in \mathbb{R}^{N_\ell} \quad \text{for} \quad 1 \le \ell \le L-1, \\
\text{output layer:} \quad & \mathcal{N}^L(\mathbf{x}) = \boldsymbol{W}^L \mathcal{N}^{L-1}(\mathbf{x}) + \boldsymbol{b}^L \in \mathbb{R}^{d_{\text{out}}};
\end{aligned}$$

see also a visualization of a neural network in Figure 1. Commonly used activation functions include the logistic sigmoid $1/(1+e^{-x})$, the hyperbolic tangent (tanh), and the rectified linear unit (ReLU, $\max\{x, 0\}$).

Lu et al. 2021

Universal approximation theorem

Neural Networks, Vol. 2, pp. 359–366, 1989
Printed in the USA. All rights reserved.

ORIGINAL CONTRIBUTION

Multilayer Feedforward Networks are Universal Approximators

KURT HORNIK

Technische Universität Wien

MAXWELL STINCHCOMBE AND HALBERT WHITE

University of California, San Diego

(*Received* 16 *September* 1988; *revised and accepted* 9 *March* 1989)

Abstract—*This paper rigorously establishes that standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators.*

Keywords—Feedforward networks, Universal approximation, Mapping networks, Network representation capability, Stone-Weierstrass Theorem, Squashing functions, Sigma-Pi networks, Back-propagation networks.

Hornik et al. 1989

Universal approximation theorem

Feed-forward neural nets (FNNs) with enough neurons can simultaneously and uniformly approximate any function and its partial derivatives.

$\mathcal{F}$: The family of all the functions that can be represented by our chosen neural network architecture.

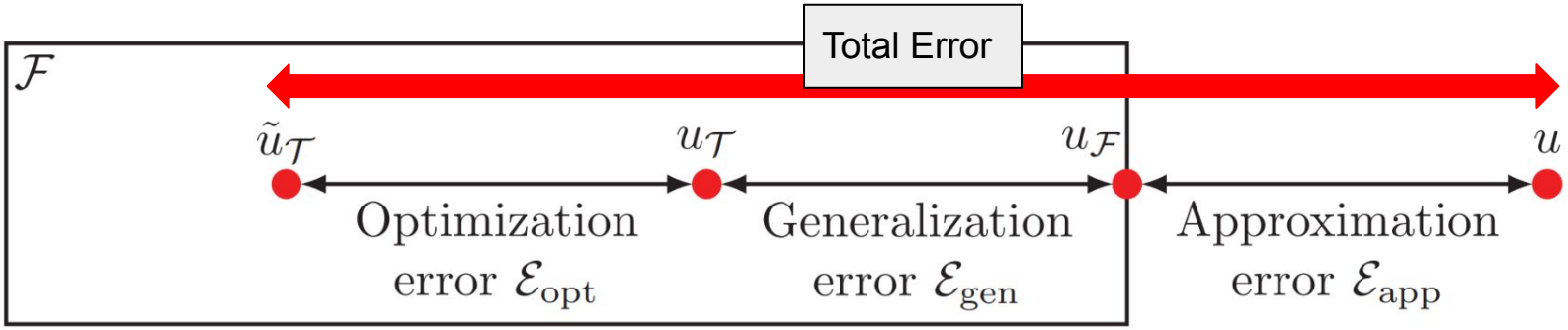


Fig. 3 *Illustration of errors of a PINN. The total error consists of the approximation error, the optimization error, and the generalization error. Here, u is the PDE solution, $u_{\mathcal{F}}$ is the best function close to u in the function space $\mathcal{F}$, $u_{\mathcal{T}}$ is the neural network whose loss is at a global minimum, and $\tilde{u}_{\mathcal{T}}$ is the function obtained by training a neural network.*

Lu et al. 2021

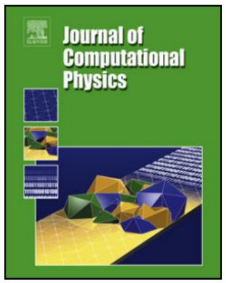
Physics-informed neural networks (PINNs)

Journal of Computational Physics 378 (2019) 686–707

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp



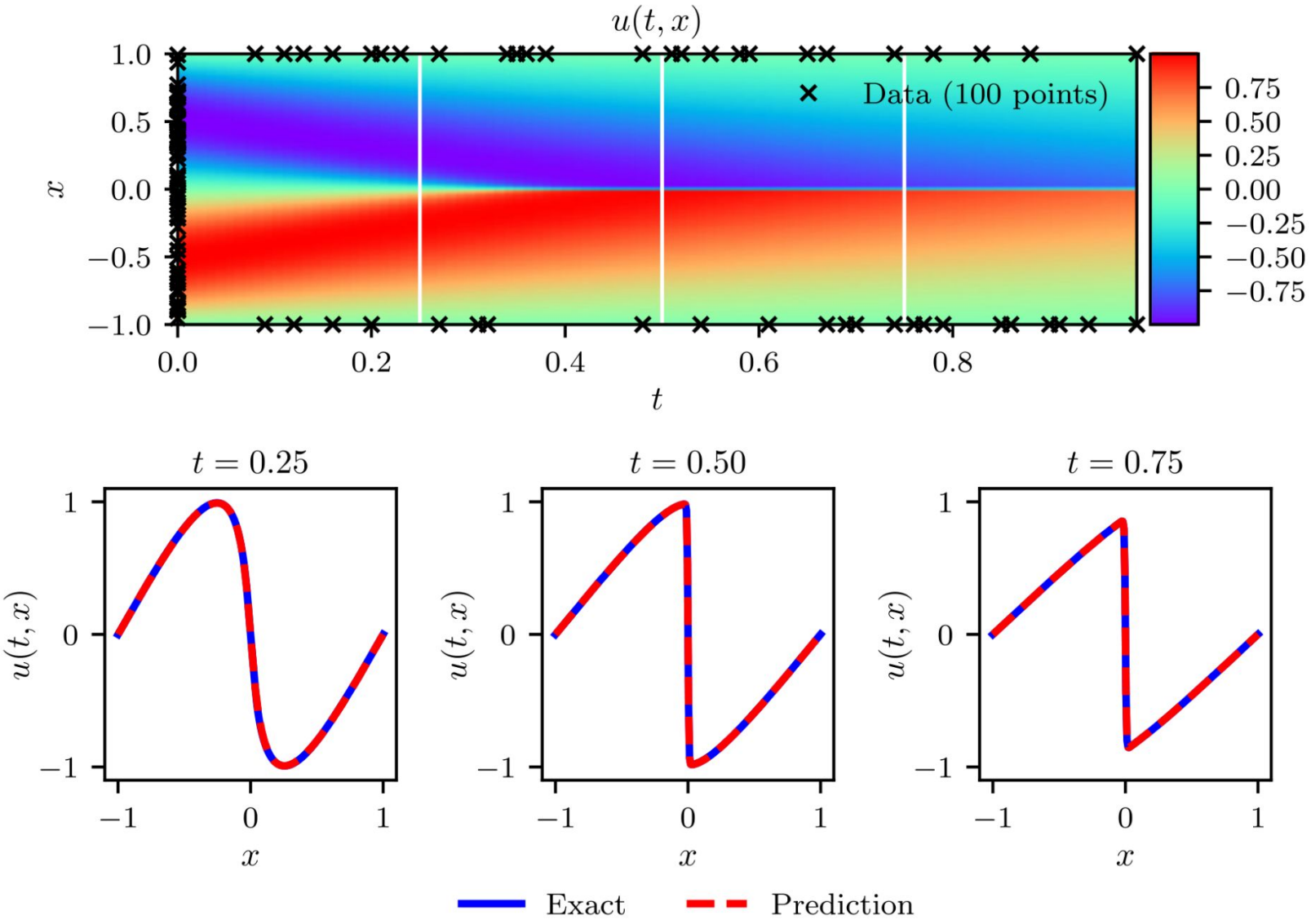
Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations

M. Raissi [a], P. Perdikaris [b,*], G.E. Karniadakis [a]

[a] *Division of Applied Mathematics, Brown University, Providence, RI, 02912, USA*
[b] *Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA, 19104, USA*

Raissi et al. 2019

Forward problem



Raissi et al. 2019