Physics Informed Machine Learning for Solving PDEs

Ming Zhong

Department of Applied Mathematics
Illinois Institute of Technology

December 2, 2022

Numerical PDE Lecture 01

Table of Contents

Partial Differential Equations

Many Kinds

A second-order linear constant coefficient PDE for u

$$Au_{xx} + 2Bu_{xy} + Cu_{yy} + Du_x + Eu_y + F = 0,$$

Partial Differential Equations

Many Kinds

A second-order linear constant coefficient PDE for u

$$Au_{xx} + 2Bu_{xy} + Cu_{yy} + Du_x + Eu_y + F = 0,$$

- $B^2 - AC > 0 \Rightarrow$ Elliptic (Laplace's Equation)

Partial Differential Equations

Many Kinds

A second-order linear constant coefficient PDE for u

$$Au_{xx} + 2Bu_{xy} + Cu_{yy} + Du_x + Eu_y + F = 0,$$

- $B^2 - AC > 0 \Rightarrow$ Elliptic (Laplace's Equation)
- $B^2 - AC = 0 \Rightarrow$ Parabolic (Heat Equation)

Partial Differential Equations

Many Kinds

A second-order linear constant coefficient PDE for u

$$Au_{xx} + 2Bu_{xy} + Cu_{yy} + Du_x + Eu_y + F = 0,$$

- $B^2 - AC > 0 \Rightarrow$ Elliptic (Laplace's Equation)
- $B^2 - AC = 0 \Rightarrow$ Parabolic (Heat Equation)
- $B^2 - AC < 0 \Rightarrow$ Hyperbolic (Wave Equation)

Partial Differential Equations

Many Kinds

A second-order linear constant coefficient PDE for u

$$Au_{xx} + 2Bu_{xy} + Cu_{yy} + Du_x + Eu_y + F = 0,$$

- $B^2 - AC > 0 \Rightarrow$ Elliptic (Laplace's Equation)
- $B^2 - AC = 0 \Rightarrow$ Parabolic (Heat Equation)
- $B^2 - AC < 0 \Rightarrow$ Hyperbolic (Wave Equation)
  - $u_t + (f(u))_x = 0$ Hyperbolic

Introduction

A Simple Heat Equation

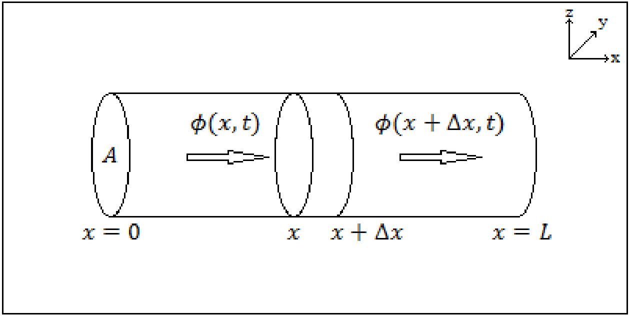


Figure: Heat transfer through a thin metal rod with insulated ends.

Introduction

A Simple Heat Equation, cont.

Let $u : [0, T] \times [0, L] \to \mathbb{R}$ such that

$$u_t(t, x) = \lambda u_{xx}(t, x), \quad u_t = \frac{\partial u}{\partial t} \text{ and } u_{xx} = \frac{\partial^2 u}{\partial x^2},$$
$$u(0, x) = f(x), \quad u(t, 0) = 0, \quad u(t, L) = 0.$$

Introduction

A Simple Heat Equation, cont.

Let $u : [0, T] \times [0, L] \to \mathbb{R}$ such that

$$u_t(t, x) = \lambda u_{xx}(t, x), \quad u_t = \frac{\partial u}{\partial t} \text{ and } u_{xx} = \frac{\partial^2 u}{\partial x^2},$$
$$u(0, x) = f(x), \quad u(t, 0) = 0, \quad u(t, L) = 0.$$

How to solve?

Introduction

A Simple Heat Equation, cont.

Let $u : [0, T] \times [0, L] \to \mathbb{R}$ such that

$$u_t(t,x) = \lambda u_{xx}(t,x), \quad u_t = \frac{\partial u}{\partial t} \text{ and } u_{xx} = \frac{\partial^2 u}{\partial x^2},$$
$$u(0,x) = f(x), \quad u(t,0) = 0, \quad u(t,L) = 0.$$

How to solve?

- Separation of variables

Introduction

A Simple Heat Equation, cont.

Let $u : [0, T] \times [0, L] \to \mathbb{R}$ such that

$$u_t(t, x) = \lambda u_{xx}(t, x), \quad u_t = \frac{\partial u}{\partial t} \text{ and } u_{xx} = \frac{\partial^2 u}{\partial x^2},$$
$$u(0, x) = f(x), \quad u(t, 0) = 0, \quad u(t, L) = 0.$$

How to solve?

- Separation of variables
- Finite Difference Method

Introduction

A Simple Heat Equation, cont.

Let $u : [0, T] \times [0, L] \to \mathbb{R}$ such that

$$u_t(t,x) = \lambda u_{xx}(t,x), \quad u_t = \frac{\partial u}{\partial t} \text{ and } u_{xx} = \frac{\partial^2 u}{\partial x^2},$$
$$u(0,x) = f(x), \quad u(t,0) = 0, \quad u(t,L) = 0.$$

How to solve?

- Separation of variables
- Finite Difference Method
- Finite Element, (Pseudo) Spectral Method, Finite Volume, etc.

Introduction

Separation of Variables

Assume $u(t,x) = F(x)G(t)$, since $u_t = \lambda u_{xx}$,

$$F(x)\frac{dG(t)}{dt} = \lambda G(t)\frac{d^2F(x)}{dx^2},$$

Introduction

Separation of Variables

Assume $u(t,x) = F(x)G(t)$, since $u_t = \lambda u_{xx}$,

$$F(x)\frac{dG(t)}{dt} = \lambda G(t)\frac{d^2F(x)}{dx^2},$$

with $C > 0$,

$$F(x) = -C\frac{d^2F(x)}{dx^2} \quad \text{and} \quad G(t) = -\lambda C\frac{dG(t)}{dt}$$

Introduction

Separation of Variables

Assume $u(t,x) = F(x)G(t)$, since $u_t = \lambda u_{xx}$,

$$F(x)\frac{dG(t)}{dt} = \lambda G(t)\frac{d^2F(x)}{dx^2},$$

with $C > 0$,

$$F(x) = -C\frac{d^2F(x)}{dx^2} \quad \text{and} \quad G(t) = -\lambda C\frac{dG(t)}{dt}$$

Then

$$F(x) = a_1\sin(Cx) + a_2\cos(Cx) \quad \text{and} \quad G(t) = a_3 e^{-\lambda Ct}.$$

Introduction

Separation of Variables, cont.

Since $u(0, x) = G(0)F(x)$,

$$f(x) = \sum_{n=1}^{\infty} B_n \sin(\frac{n\pi x}{L}), \quad B_n = \frac{2}{L}\int_0^L f(x)\sin(\frac{n\pi x}{L})\,dx,$$

Introduction

Separation of Variables, cont.

Since $u(0, x) = G(0)F(x)$,

$$f(x) = \sum_{n=1}^{\infty} B_n \sin(\frac{n\pi x}{L}), \quad B_n = \frac{2}{L}\int_0^L f(x)\sin(\frac{n\pi x}{L})\,dx,$$

Put everything back in

$$u(t, x) = \sum_{n=1}^{\infty} B_n \sin(\frac{n\pi x}{L}) e^{-\lambda(\frac{n\pi}{L})^2 t}.$$

A Simple Heat Equation

Analytic Solutions: Examples

(a) if $f(x) = 6\sin(\frac{\pi x}{L})$

A Simple Heat Equation

Analytic Solutions: Examples

(a) if $f(x) = 6\sin(\frac{\pi x}{L})$, then $B_1 = 6$ and $B_n = 0$ for $n \neq 1$, thus

$$u(t,x) = 6\sin(\frac{\pi x}{L})e^{-\lambda(\frac{\pi}{L})^2 t}.$$

A Simple Heat Equation

Analytic Solutions: Examples

(a) if $f(x) = 6\sin(\frac{\pi x}{L})$, then $B_1 = 6$ and $B_n = 0$ for $n \neq 1$, thus

$$u(t, x) = 6\sin(\frac{\pi x}{L})e^{-\lambda(\frac{\pi}{L})^2 t}.$$

(b) if $f(x) = 12\sin(\frac{9\pi x}{L}) - 7\sin(\frac{4\pi x}{L})$

A Simple Heat Equation

Analytic Solutions: Examples

(a) if $f(x) = 6\sin(\frac{\pi x}{L})$, then $B_1 = 6$ and $B_n = 0$ for $n \neq 1$, thus

$$u(t,x) = 6\sin(\frac{\pi x}{L})e^{-\lambda(\frac{\pi}{L})^2 t}.$$

(b) if $f(x) = 12\sin(\frac{9\pi x}{L}) - 7\sin(\frac{4\pi x}{L})$, then $B_4 = -7$, $B_9 = 12$ and $B_n = 0$ for $n \neq 4, 9$, hence

$$u(t,x) = -7\sin(\frac{4\pi x}{L})e^{-\lambda(\frac{4\pi}{L})^2 t} + 12\sin(\frac{9\pi x}{L})e^{-\lambda(\frac{9\pi}{L})^2 t}$$

A Simple Heat Equation

Analytic Solutions: Examples, cont.

(c) if $f(x) = 20$

A Simple Heat Equation

Analytic Solutions: Examples, cont.

(c) if $f(x) = 20$, then

$$\begin{aligned} B_n &= \frac{2}{L}\int_0^L 20\sin(\frac{n\pi x}{L})\,dx \\ &= \frac{2}{L}(\frac{20L(1-\cos(n\pi))}{n\pi}) = \frac{40(1-(-1)^n)}{n\pi}, \end{aligned}$$

A Simple Heat Equation

Analytic Solutions: Examples, cont.

(c) if $f(x) = 20$, then

$$B_n = \frac{2}{L}\int_0^L 20\sin(\frac{n\pi x}{L})\,dx$$
$$= \frac{2}{L}(\frac{20L(1-\cos(n\pi))}{n\pi}) = \frac{40(1-(-1)^n)}{n\pi},$$

hence

$$u(t,x) = \sum_{n=1}^{\infty}\frac{40(1-(-1)^n)}{n\pi}\sin(\frac{n\pi x}{L})e^{-\lambda(\frac{n\pi}{L})^2 t}.$$

A Simple Heat Equation

Numerical Solutions

Summary

A Simple Heat Equation

Numerical Solutions

Summary

- The key is being to integrate to get B_n.

A Simple Heat Equation

Numerical Solutions

Summary

- The key is being to integrate to get B_n.
- Actual computation, we have to stop the infinite sum somewhere to compute $u(t, x)$.

A Simple Heat Equation

Numerical Solutions

Summary

- The key is being to integrate to get B_n.
- Actual computation, we have to stop the infinite sum somewhere to compute $u(t, x)$.
- Different formulas when $u(t, 0) \neq 0$ or $u(t, L) \neq 0$ or some other types of conditions.

A Simple Heat Equation

Numerical Solutions

Summary

- The key is being to integrate to get B_n.
- Actual computation, we have to stop the infinite sum somewhere to compute $u(t, x)$.
- Different formulas when $u(t, 0) \neq 0$ or $u(t, L) \neq 0$ or some other types of conditions.

4 major kinds of numerical methods

- Finite Difference, Finite Element, Finite Volume and Spectral Method (Pseudo Spectral Method).

A Simple Heat Equation

Finite Difference Method[1]

Recall, $u_t = \lambda u_{xx}$ for $(t, x) \in [0, T] \times [0, L]$, then for $0 \leq t < t + k \leq T$

$$u_t(t, x) \approx \frac{u(t + k, x) - u(t, x)}{k}.$$

[1] A. Iserles, "A First Course in the Numerical Analysis of Differential Equations", Second Edition, 2009.

A Simple Heat Equation

Finite Difference Method[1]

Recall, $u_t = \lambda u_{xx}$ for $(t, x) \in [0, T] \times [0, L]$, then for $0 \leq t < t + k \leq T$

$$u_t(t, x) \approx \frac{u(t + k, x) - u(t, x)}{k}.$$

and for $0 \leq x - h < x < x + h \leq L$

$$u_{xx}(t, x) \approx \frac{u(t, x + h) - 2u(t, x) + u(t, x - h)}{h^2};$$

[1] A. Iserles, "A First Course in the Numerical Analysis of Differential Equations", Second Edition, 2009.

A Simple Heat Equation

Finite Difference Method, cont.

Hence

$$\frac{u(t+k,x)-u(t,x)}{k} \approx \lambda \frac{u(t,x+h)-2u(t,x)+u(t,x-h)}{h^2}.$$

A Simple Heat Equation

Finite Difference Method, cont.

Hence

$$\frac{u(t+k,x)-u(t,x)}{k} \approx \lambda \frac{u(t,x+h)-2u(t,x)+u(t,x-h)}{h^2}.$$

Re-arranging

$$u(t+k,x) = u(t,x) \\ \lambda \frac{k}{h^2}(u(t,x+h)-2u(t,x)+u(t,x-h)).$$

A Simple Heat Equation

Finite Difference Method, cont.

Algorithm

Starting from $t = 0$

A Simple Heat Equation

Finite Difference Method, cont.

Algorithm

Starting from $t = 0$

- Advance $u(0, x) = f(x)$ with $u(t, 0) = u(t, L) = 0$ from $t = 0$ to $t = k$ using the formula

A Simple Heat Equation

Finite Difference Method, cont.

Algorithm

Starting from $t = 0$

- Advance $u(0, x) = f(x)$ with $u(t, 0) = u(t, L) = 0$ from $t = 0$ to $t = k$ using the formula
- Repeat until $t = T$.

A Simple Heat Equation

Finite Difference Method, cont.

Algorithm

Starting from $t = 0$

- Advance $u(0, x) = f(x)$ with $u(t, 0) = u(t, L) = 0$ from $t = 0$ to $t = k$ using the formula
- Repeat until $t = T$.

Prepare the grid

- $0 = t_0 < t_1 < \cdots < t_N = T$ and $t_n - t_{n-1} = k$ for $n = 1, \ldots, N$.
- $0 = x_0 < x_1 < \cdots < x_M = L$ and $x_m - x_{m-1} = h$ for $m = 1, \ldots, M$.

A Simple Heat Equation

Implementation

Obtain $\hat{u}_m^n \approx u(t_n, x_m)$

$$\hat{u}_m^{n+1} = \hat{u}_m^n + \mu(\hat{u}_{m+1}^n - 2\hat{u}_m^n + \hat{u}_{m-1}^n),$$

for $n = 1, \ldots, N$ and $m = 1, \ldots, M-1$.

A Simple Heat Equation

Implementation

Obtain $\hat{u}_m^n \approx u(t_n, x_m)$

$$\hat{u}_m^{n+1} = \hat{u}_m^n + \mu(\hat{u}_{m+1}^n - 2\hat{u}_m^n + \hat{u}_{m-1}^n),$$

for $n = 1, \ldots, N$ and $m = 1, \ldots, M - 1$.

- $\mu = \lambda \frac{k}{h^2}$: Courant Number.

A Simple Heat Equation

Implementation

Obtain $\hat{u}_m^n \approx u(t_n, x_m)$

$$\hat{u}_m^{n+1} = \hat{u}_m^n + \mu(\hat{u}_{m+1}^n - 2\hat{u}_m^n + \hat{u}_{m-1}^n),$$

for $n = 1, \ldots, N$ and $m = 1, \ldots, M - 1$.

- $\mu = \lambda \frac{k}{h^2}$: Courant Number.
- $h, k \to 0 \Rightarrow \hat{u}_m^n \to u(t_n, x_m)$.

A Simple Heat Equation

Implementation

Obtain $\hat{u}_m^n \approx u(t_n, x_m)$

$$\hat{u}_m^{n+1} = \hat{u}_m^n + \mu(\hat{u}_{m+1}^n - 2\hat{u}_m^n + \hat{u}_{m-1}^n),$$

for $n = 1, \ldots, N$ and $m = 1, \ldots, M-1$.

- $\mu = \lambda \frac{k}{h^2}$: Courant Number.
- $h, k \to 0 \Rightarrow \hat{u}_m^n \to u(t_n, x_m)$.
- $\mu \leq \frac{1}{2}$ (CFL condition).

A Simple Heat Equation

Implementation, cont.

Let

$$\mathbf{U}_n = \begin{bmatrix} \hat{u}_1^n \\ \vdots \\ \hat{u}_{M-1}^n \end{bmatrix} \in \mathbb{R}^{M-1}$$

A Simple Heat Equation

Implementation, cont.

Let

$$\mathbf{U}_n = \begin{bmatrix} \hat{u}_1^n \\ \vdots \\ \hat{u}_{M-1}^n \end{bmatrix} \in \mathbb{R}^{M-1}$$

then

$$\hat{u}_m^{n+1} = \hat{u}_m^n + \mu(\hat{u}_{m+1}^n - 2\hat{u}_m^n + \hat{u}_{m-1}^n), \quad 1 \le m \le M-1$$

A Simple Heat Equation

Implementation, cont.

Let

$$\mathbf{U}_n = \begin{bmatrix} \hat{u}_1^n \\ \vdots \\ \hat{u}_{M-1}^n \end{bmatrix} \in \mathbb{R}^{M-1}$$

then

$$\hat{u}_m^{n+1} = \hat{u}_m^n + \mu(\hat{u}_{m+1}^n - 2\hat{u}_m^n + \hat{u}_{m-1}^n), \quad 1 \le m \le M-1$$

becomes

$$\mathbf{U}_{n+1} = \mathbf{U}_n + \mu A \mathbf{U}_n.$$

A Simple Heat Equation

Implementation, cont.

Here

$$A = \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ & \ddots & \ddots & \ddots & \\ \cdots & 0 & 1 & -2 & 1 \\ \cdots & & 0 & 1 & -2 \end{bmatrix} \in \mathbb{R}^{(M-1)\times(M-1)},$$

and

$$\mathbf{U}_0 = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_{M-1}) \end{bmatrix}$$

A Simple Heat Equation

Numerical Solutions - Example

A simple test

- Let $L = 2,\ T = 2,\ \lambda = 1,\ f(x) = 6\sin(\frac{\pi x}{2})$, so

$$u(t,x) = 6\sin(\frac{\pi x}{2})e^{-(\frac{\pi}{2})^2 t}$$

- Choose $h = \frac{1}{10}$, since $k \le \frac{h^2}{2}$, we take

$$k = \frac{1}{200}.$$

- Compare the finite difference solution to the true solution side by side.

Numerical PDE

Brief Conclusion

Summary

Numerical PDE

Brief Conclusion

Summary

- Able to take care of general IC/BCs and $f(x)$.

Numerical PDE

Brief Conclusion

Summary

- Able to take care of general IC/BCs and $f(x)$.
- It generates a large linear system (sparse, symmetric, negative-definite) at each time step.

Numerical PDE

Brief Conclusion

Summary

- Able to take care of general IC/BCs and $f(x)$.
- It generates a large linear system (sparse, symmetric, negative-definite) at each time step.
- Generalization to $2D/3D$ requires extra attention and increasing storage for A.

Numerical PDE

Brief Conclusion

Summary

- Able to take care of general IC/BCs and $f(x)$.
- It generates a large linear system (sparse, symmetric, negative-definite) at each time step.
- Generalization to $2D/3D$ requires extra attention and increasing storage for A.
- $\lambda\frac{k}{h^2} \leq \frac{1}{2}$ restricts Δt.

Numerical PDE

Brief Conclusion

Summary

- Able to take care of general IC/BCs and $f(x)$.
- It generates a large linear system (sparse, symmetric, negative-definite) at each time step.
- Generalization to $2D/3D$ requires extra attention and increasing storage for A.
- $\lambda \frac{k}{h^2} \leq \frac{1}{2}$ restricts Δt.
- No inter grid point values, no derivatives, etc.

Numerical PDE

Brief Conclusion

Summary

- Able to take care of general IC/BCs and $f(x)$.
- It generates a large linear system (sparse, symmetric, negative-definite) at each time step.
- Generalization to $2D/3D$ requires extra attention and increasing storage for A.
- $\lambda\frac{k}{h^2} \leq \frac{1}{2}$ restricts Δt.
- No inter grid point values, no derivatives, etc.
- Cannot handle irregular domain.

Numerical PDE

Brief Conclusion, cont.

What about other methods?

Numerical PDE

Brief Conclusion, cont.

What about other methods?

- Finite Element Method

Numerical PDE

Brief Conclusion, cont.

What about other methods?

- Finite Element Method
  - weak formulation, mesh generation, basis, integrals, large and sparse linear system, etc.

Numerical PDE

Brief Conclusion, cont.

What about other methods?

- Finite Element Method
  - weak formulation, mesh generation, basis, integrals, large and sparse linear system, etc.
- Spectral Method

Numerical PDE

Brief Conclusion, cont.

What about other methods?

- Finite Element Method
  - weak formulation, mesh generation, basis, integrals, large and sparse linear system, etc.
- Spectral Method
  - periodic BC, Fourier/Chebyshev basis, Fast Fourier Transform, small but dense linear system, etc.

Numerical PDE

Scientific Machine Learning

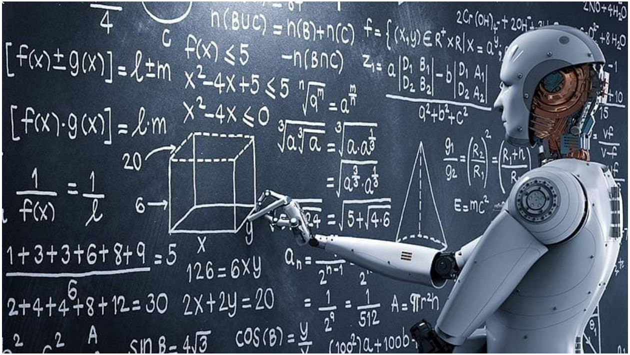


Figure: Can we teach the computer to learn how to solve PDEs?

Table of Contents

Machine Learning for Solving PDEs

Physics Informed Machine Learning

Develop hybrid methods with machine learning so that

Machine Learning for Solving PDEs

Physics Informed Machine Learning

Develop hybrid methods with machine learning so that

- Mesh-free, auto-interpolation, auto-differentiation, complex geometric domain (high-dimensional domain), data-driven basis functions, various kind of IC/BCs.

Machine Learning for Solving PDEs

Physics Informed Machine Learning

Develop hybrid methods with machine learning so that

- Mesh-free, auto-interpolation, auto-differentiation, complex geometric domain (high-dimensional domain), data-driven basis functions, various kind of IC/BCs.
- flexibility to incorporate more equations (or inequalities), uncertainty quantification.

Machine Learning for Solving PDEs

Physics Informed Machine Learning

Develop hybrid methods with machine learning so that

- Mesh-free, auto-interpolation, auto-differentiation, complex geometric domain (high-dimensional domain), data-driven basis functions, various kind of IC/BCs.
- flexibility to incorporate more equations (or inequalities), uncertainty quantification.
- data assimilation to discover parametric structure or more.

Machine Learning for Solving PDEs

PINN and PIGP

Major methods

- Deep Neural Networks (PINN).
- Gaussian Processes (PIGP).

Machine Learning for Solving PDEs

PINN and PIGP

Major methods

- Deep Neural Networks (PINN).
- Gaussian Processes (PIGP).

Consider solutions of the following form

$$u_{app}(t, \mathbf{x}) = \sum_{n=1}^{N} \alpha_n \psi_n(t, \mathbf{x}), \quad \psi_n \text{ is either a NN or GP},$$

where both α_n and ψ_n are learned from data.

Machine Learning for Solving PDEs

References

Further reading

Machine Learning for Solving PDEs

References

Further reading

- Least Square method: https://www.math.purdue.edu/~caiz/paper.html

Machine Learning for Solving PDEs

References

Further reading

- Least Square method: https://www.math.purdue.edu/~caiz/paper.html
- Pseudo Spectral Method: https://arxiv.org/pdf/1606.05432

Machine Learning for Solving PDEs

References

Further reading

- Least Square method: https://www.math.purdue.edu/~caiz/paper.html
- Pseudo Spectral Method: https://arxiv.org/pdf/1606.05432
- Crunch Group at Brown: https://www.brown.edu/research/projects/crunch/home

Machine Learning for Solving PDEs

References

Further reading

- Least Square method: https://www.math.purdue.edu/~caiz/paper.html
- Pseudo Spectral Method: https://arxiv.org/pdf/1606.05432
- Crunch Group at Brown: https://www.brown.edu/research/projects/crunch/home
- Review Paper: https://arxiv.org/abs/2201.05624

Physics Informed Neural Network[2]

General Setup

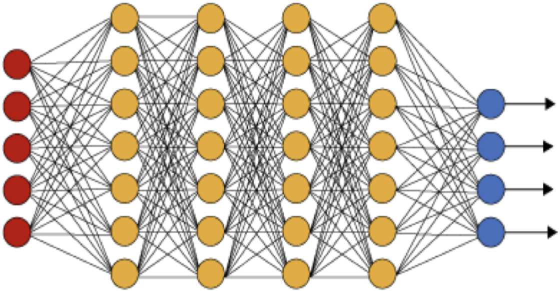


Figure: Feedforward Neural Network.

[2] "Physics-informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems involving Nonlinear Partial Differential Equations", 2019.

Physics Informed Neural Network (PINN)

General Setup, cont.

u_{NN} is a feedforward neural network (aka a multi-layer preceptron)

$$u_{NN}(\mathbf{x};\theta) = L_K \circ F_{act} \circ L_{K-1} \circ \cdots \circ F_{act} \circ L_2 \circ F_{act} \circ L_1(\mathbf{x}).$$

Here $\mathbf{x} = (t, x)$, etc.

Physics Informed Neural Network (PINN)

General Setup, cont.

u_{NN} is a feedforward neural network (aka a multi-layer preceptron)

$$u_{NN}(\mathbf{x};\theta) = L_K \circ F_{act} \circ L_{K-1} \circ \cdots \circ F_{act} \circ L_2 \circ F_{act} \circ L_1(\mathbf{x}).$$

Here $\mathbf{x} = (t, x)$, etc.

- It has an input layer, $(K-1)$ hidden layers, and an output layer for some $1 < K \in \mathbb{N}$.

Physics Informed Neural Network (PINN)

General Setup, cont.

u_{NN} is a feedforward neural network (aka a multi-layer preceptron)

$$u_{NN}(\mathbf{x};\theta) = L_K \circ F_{act} \circ L_{K-1} \circ \cdots \circ F_{act} \circ L_2 \circ F_{act} \circ L_1(\mathbf{x}).$$

Here $\mathbf{x} = (t, x)$, etc.

- It has an input layer, $(K-1)$ hidden layers, and an output layer for some $1 < K \in \mathbb{N}$.
- The k^{th} hidden layer with w_k neurons is given an input $\mathbf{z}_k \in \mathbb{R}^{w_k}$, transformed by a linear map L_k then activated by F_{act}.

Physics Informed Neural Network (PINN)

General Setup, cont.

u_{NN} is a feedforward neural network (aka a multi-layer preceptron)

$$u_{NN}(\mathbf{x};\theta) = L_K \circ F_{act} \circ L_{K-1} \circ \cdots \circ F_{act} \circ L_2 \circ F_{act} \circ L_1(\mathbf{x}).$$

Here $\mathbf{x} = (t, x)$, etc.

- It has an input layer, $(K-1)$ hidden layers, and an output layer for some $1 < K \in \mathbb{N}$.
- The k^{th} hidden layer with w_k neurons is given an input $\mathbf{z}_k \in \mathbb{R}^{w_k}$, transformed by a linear map L_k then activated by F_{act}.
- $\circ$: function composition.

Physics Informed Neural Network (PINN)

General Setup, cont.

Moreover

Physics Informed Neural Network (PINN)

General Setup, cont.

Moreover

- F_{act} is a scalar (non-linear) activation function applied component wise. Popular choice: hyperbolic tangent or ReLU.

Physics Informed Neural Network (PINN)

General Setup, cont.

Moreover

- F_{act} is a scalar (non-linear) activation function applied component wise. Popular choice: hyperbolic tangent or ReLU.
- $L_k \mathbf{z}_k = A_k \mathbf{z}_k + \mathbf{b}_k$, $A_k \in \mathbb{R}^{w_{k+1} \times w_k}$, $\mathbf{z}_k \in \mathbb{R}^{w_k}$, $\mathbf{b}_k \in \mathbb{R}^{w_{k+1}}$.

Physics Informed Neural Network (PINN)

General Setup, cont.

Moreover

- F_{act} is a scalar (non-linear) activation function applied component wise. Popular choice: hyperbolic tangent or ReLU.
- $L_k\mathbf{z}_k = A_k\mathbf{z}_k + \mathbf{b}_k$, $A_k \in \mathbb{R}^{w_{k+1}\times w_k}$, $\mathbf{z}_k \in \mathbb{R}^{w_k}$, $\mathbf{b}_k \in \mathbb{R}^{w_{k+1}}$.
- Our setting: $w_K = 1$, a total of $1 + \sum_{k=1}^{K-1} w_k$ neurons, and $\theta = \{A_k, \mathbf{b}_k\}$ and $\theta \in \Theta \subset \mathbb{R}^{\sum_{k=1}^{K-1}(w_k w_{k+1} + w_{k+1})}$.

Physics Informed Neural Networks

General Setup

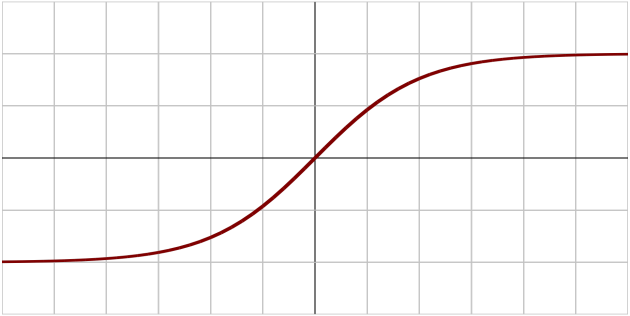


Figure: Activation Function: Hyperbolic Tangent.

Physics Informed Neural Networks

General Setup

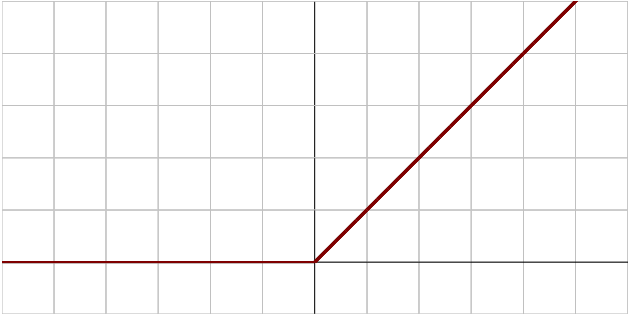


Figure: Activation Function: ReLU function.

PINN

Solving PDEs

Find θ_* by training from minimizing a certain loss function,

$$\theta_* = \operatorname*{argmin}_{\theta \in \Theta} \{\text{Loss}(\theta)\}, \quad \text{but Loss}(\theta) = ??.$$

PINN

Solving PDEs

Find θ_* by training from minimizing a certain loss function,

$$\theta_* = \underset{\theta \in \Theta}{\operatorname{argmin}} \{\text{Loss}(\theta)\}, \quad \text{but Loss}(\theta) = ??.$$

Normal Machine Learning

$$\text{Loss}(\theta) = \frac{1}{M} \sum_{i=1}^{M} \left| u_{NN}(\mathbf{x}_i; \theta) - y_i \right|^2$$

where $y_i = u(\mathbf{x}_i) + \text{noise}$.

PINN

Solving PDEs, the Loss

PDE Problem

For $\Omega \subset \mathbb{R}^{w_1}$ with Lipschitz boundary $\partial\Omega$ and $u : \Omega \to \mathbb{R}$ such that

$$\mathcal{P}(u)(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega; \quad \mathcal{B}(u)(\mathbf{x}) = g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega.$$

PINN

Solving PDEs, the Loss

PDE Problem

For $\Omega \subset \mathbb{R}^{w_1}$ with Lipschitz boundary $\partial\Omega$ and $u : \Omega \to \mathbb{R}$ such that

$$\mathcal{P}(u)(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega; \quad \mathcal{B}(u)(\mathbf{x}) = g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega.$$

- Solving PDEs, no $u(\mathbf{x}_i)$ is available in Ω.

PINN

Solving PDEs, the Loss

PDE Problem

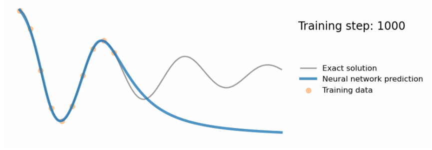
For $\Omega \subset \mathbb{R}^{w_1}$ with Lipschitz boundary $\partial\Omega$ and $u : \Omega \to \mathbb{R}$ such that

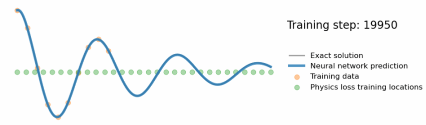
$$\mathcal{P}(u)(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega; \quad \mathcal{B}(u)(\mathbf{x}) = g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega.$$

- Solving PDEs, no $u(\mathbf{x}_i)$ is available in Ω.
- Only BC data on $\partial\Omega$ and PDE inside Ω (Extrapolation)

PINN

PDE Loss Provides Extrapolation





PINN

Solving PDEs, the Loss

Hence

$$\text{Loss}(\theta) = \text{Loss}_{PDE}(\theta) + \text{Loss}_{BC}(\theta)$$

PINN

Solving PDEs, the Loss

Hence

$$\text{Loss}(\theta) = \text{Loss}_{PDE}(\theta) + \text{Loss}_{BC}(\theta)$$

Then for $\mathbf{x}_i^{CL} \in \Omega$

$$\text{Loss}_{PDE}(\theta) = \frac{1}{N_{CL}} \sum_{i=1}^{N_{CL}} \left| \mathcal{P}(u_{NN}(\mathbf{x}_i^{CL}; \theta)) \right|^2,$$

PINN

Solving PDEs, the Loss

Hence

$$\text{Loss}(\theta) = \text{Loss}_{PDE}(\theta) + \text{Loss}_{BC}(\theta)$$

Then for $\mathbf{x}_i^{CL} \in \Omega$

$$\text{Loss}_{PDE}(\theta) = \frac{1}{N_{CL}} \sum_{i=1}^{N_{CL}} \left| \mathcal{P}(u_{NN}(\mathbf{x}_i^{CL}; \theta)) \right|^2,$$

And for $\mathbf{x}_i^{BC} \in \partial\Omega$

$$\text{Loss}_{BC}(\theta) = \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} \left| \mathcal{B}(u_{NN}(\mathbf{x}_i^{BC}; \theta)) - g(\mathbf{x}_i^{BC}) \right|^2.$$

PINN

Solving PDEs, cont.

Different weighting for the losses

$$\text{Loss}(\theta) = \lambda_{PDE}\text{Loss}_{PDE}(\theta) + \lambda_{BC}\text{Loss}_{BC}(\theta).$$

With preset $\lambda_{PDE}, \lambda_{BC}$.

PINN

Solving PDEs, cont.

Different weighting for the losses

$$\text{Loss}(\theta) = \lambda_{PDE}\text{Loss}_{PDE}(\theta) + \lambda_{BC}\text{Loss}_{BC}(\theta).$$

With preset $\lambda_{PDE}, \lambda_{BC}$.

- Computed using NTK: "When and Why PINNs Fail to Train: A Neural Tangent Kernel Perspective", 2020.

PINN

Solving PDEs, cont.

Data-driven weighting[3]

$$
\begin{aligned}
\mathrm{Loss}(\theta) = &\sum_{i=1}^{N_{CL}} f_{mask}(\lambda_i^{CL})\Big|\mathcal{P}(u_{NN}(\mathbf{x}_i^{CL};\theta))\Big|^2 \\
&+ \sum_{i=1}^{N_{BC}} f_{mask}(\lambda_i^{BC})\Big|\mathcal{B}(u_{NN}(\mathbf{x}_i^{BC};\theta)) - g(\mathbf{x}_i^{BC})\Big|^2
\end{aligned}
$$

[3] "Self-adaptive Physics-informed Neural Networks using a Soft Attention Mechanism", 2020.

PINN

A Simple Heat Equation

Recall

$$
\begin{aligned}
u_t(t,x) = \lambda u_{xx}(t,x), &\quad (t,x) \in (0,T] \times [0,L] \\
u(0,x) = f(x), \quad u(t,0) = 0, &\quad u(t,L) = 0.
\end{aligned}
$$

So $\Omega = (0,T] \times (0,L)$,

$$
\mathcal{P}(u)(\mathbf{x}) = u_t(\mathbf{x}) - \lambda u_{xx}(\mathbf{x}) = 0, \quad \mathbf{x} = (t,x) \in \Omega,
$$

$$
\mathcal{B}(u)(\mathbf{x}) = g(\mathbf{x}) = \begin{cases} f(x), & \mathbf{x} \in \{0\} \times [0,L], \\ 0, & \mathbf{x} \in [0,T] \times \{0\}, \\ 0, & \mathbf{x} \in [0,T] \times \{L\} \end{cases}
$$

PINN

A Simple Heat Equation, cont.

Next

$$\text{Loss}_{PDE}(\theta) = \frac{1}{N_{CL}} \sum_{i=1}^{N_{CL}} \left| \frac{\partial u_{NN}(\mathbf{x}_i^{CL};\theta)}{\partial t} - \lambda \frac{\partial^2 u_{NN}(\mathbf{x}_i^{CL};\theta)}{\partial x^2} \right|^2.$$

The BC has 3 different parts, BC_1, BC_2, BC_3:

$$\begin{aligned}\text{Loss}_{BC}(\theta) = \frac{1}{N_{BC_1}}\text{Loss}_{BC_1}(\theta) + \frac{1}{N_{BC_2}}\text{Loss}_{BC_2}(\theta) \\ + \frac{1}{N_{BC_3}}\text{Loss}_{BC_3}(\theta)\end{aligned}$$

PINN

A Simple Heat Equation, cont.

For BC_1, $\mathbf{x}_i^{BC_1} = (0, x_i^{BC_1}), \quad x_i^{BC_1} \in [0, L]$

$$\text{Loss}_{BC_1}(\theta) = \frac{1}{N_{BC_1}} \sum_{i=1}^{BC_1} \left| u_{NN}(\mathbf{x}_i^{BC_1}; \theta) - g(\mathbf{x}_i^{BC_1}) \right|^2,$$

For BC_2, $\mathbf{x}_i^{BC_2} = (t_i^{BC_2}, 0), \quad t_i^{BC_2} \in [0, T]$

$$\text{Loss}_{BC_2}(\theta) = \frac{1}{N_{BC_2}} \sum_{i=1}^{BC_2} \left| u_{NN}(\mathbf{x}_i^{BC_2}; \theta) \right|^2,$$

PINN

A Simple Heat Equation, cont.

For BC_3, $\mathbf{x}_i^{BC_3} = (t_i^{BC_3}, L), \quad t_i^{BC_3} \in [0, T]$,

$$\text{Loss}_{BC_3}(\theta) = \frac{1}{N_{BC_3}} \sum_{i=1}^{BC_3} \left| u_{NN}(\mathbf{x}_i^{BC_3}; \theta) \right|^2.$$

PINN

Test

A simple test

- Let $L = 2,\ T = 2,\ \lambda = 1,\ f(x) = 6\sin(\frac{\pi x}{2})$, so

$$u(t, x) = 6\sin(\frac{\pi x}{2})e^{-(\frac{\pi}{2})^2 t}.$$

- Choose $N_{CL} = 2048$ and $N_{BC} = 64 * 3$.
- 7-hidden layers, and 20 neurons on each layer.
- Adam training with learning rate at $5 \cdot 10^{-4}$ and 5000 iterations.
- Compare the finite difference solution to the true solution side by side.

PINN

Nonlinear Diffusion and Advection - Burgers

Consider $u : [0, T] \times [-L, L]$

$$u_t(t, x) + uu_x = \nu u_{xx}(t, x), \quad \text{Viscous Burgers,}$$
$$u(0, x) = f(x), \quad u(t, -L) = 0, \quad u(t, L) = 0.$$

Therefore $\Omega = (0, T] \times (-L, L)$ and $\mathbf{x} = (t, x)$

$$\mathcal{P}(u)(\mathbf{x}) = u_t(\mathbf{x}) + uu_x - \nu u_{xx}(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega,$$

$$\mathcal{B}(u)(\mathbf{x}) = g(\mathbf{x}) = \begin{cases} f(x), & \mathbf{x} \in \{0\} \times [-L, L], \\ 0, & \mathbf{x} \in [0, T] \times \{-L\}, \\ 0, & \mathbf{x} \in [0, T] \times \{L\} \end{cases}$$

PINN

Nonlinear Diffusion and Advection - Burgers, cont.

Hence

$$\begin{aligned}\text{Loss}_{PDE}(\theta) = \frac{1}{N_{CL}} \sum_{i=1}^{N_{CL}} | & \frac{\partial u_{NN}(\mathbf{x}_i^{CL}; \theta)}{\partial t} \\ & + u_{NN}(\mathbf{x}_i^{CL}; \theta) \frac{\partial u_{NN}(\mathbf{x}_i^{CL}; \theta)}{\partial x} \\ & - \nu \frac{\partial^2 u_{NN}(\mathbf{x}_i^{CL}; \theta)}{\partial x^2} |^2.\end{aligned}$$

The BC has 3 different parts, BC_1, BC_2, BC_3, $\text{Loss}_{BC}(\theta) = \sum_{i=1}^{N} \frac{1}{N_{BC_i}} \text{Loss}_{BC_i}(\theta)$.

PINN

Nonlinear Diffusion and Advection - Burgers, cont.

For BC_1, $\mathbf{x}_i^{BC_1} = (0, x_i^{BC_1}), x_i^{BC_1} \in [0, L]$,

$$\text{Loss}_{BC_1}(\theta) = \frac{1}{N_{BC_1}} \sum_{i=1}^{BC_1} \left| u_{NN}(\mathbf{x}_i^{BC_1}; \theta) - g(\mathbf{x}_i^{BC_1}) \right|^2,$$

For BC_2, $\mathbf{x}_i^{BC_2} = (t_i^{BC_2}, 0), t_i^{BC_2} \in [0, T]$

$$\text{Loss}_{BC_2}(\theta) = \frac{1}{N_{BC_2}} \sum_{i=1}^{BC_2} \left| u_{NN}(\mathbf{x}_i^{BC_2}; \theta) \right|^2,$$

PINN

Nonlinear Diffusion and Advection - Burgers, cont.

For BC_3, $\mathbf{x}_i^{BC_3} = (t_i^{BC_3}, L), \quad t_i^{BC_3} \in [0, T]$,

$$\mathrm{Loss}_{BC_3}(\theta) = \frac{1}{N_{BC_3}} \sum_{i=1}^{BC_3} \left| u_{NN}(\mathbf{x}_i^{BC_3}; \theta) \right|^2.$$

PINN

Burgers - test

A simple test

- Let $L = 1$, $T = \frac{3}{\pi}$, $\nu = \frac{0.01}{\pi}$, $f(x) = -\sin(\pi x)$, the true solution is computed using the Hopf-Cole transformation.
- Choose $N_{CL} = 2048$ and $N_{BC} = 64 * 3$.
- 7-hidden layers, and 20 neurons on each layer.
- Adam training with learning rate at $5 \cdot 10^{-4}$ and 5000 iterations.
- Compare the finite difference solution to the true solution side by side.

PINN

Other Equations

How about differential-integral equation, e.g. the radiative transfer equation[4]? For $(x, \mu) \in [0, 1] \times [-1, 1]$

$$\mu \frac{\partial u(x, \mu)}{\partial x} + u(x, \mu) - \frac{x}{2} \int_{-1}^{1} \Phi(\mu, \mu') u(x, \mu') \, d\mu' = 0,$$

with only inflow boundary condition

$$u(0, \mu) = 1, \quad \mu \in (0, 1]; \quad u(1, \mu) = 0, \quad \mu \in [-1, 0).$$

[4] "Physics Informed Neural Networks for Simulating Radiative Transer", 2021.

PINN

Other Equations, cont.

How about scalar conservation laws with only initial condition[5]?

$$u_t + (f(u))_x = 0, \quad u(0, x) = u_0(x).$$

We add data-driven artificial viscosity, i.e., solve

$$u_t + (f(u))_x = \nu u_{xx}, \quad u(0, x) = u_0(x).$$

Here $\nu > 0$ is learned from data.

---

[5] "Physics-informed Neural Networks with Adaptive Localized Artificial Viscosity", 2022.

PINN

Other Equations, cont.

How about scalar conservation laws with only initial condition[5]?

$$u_t + (f(u))_x = 0, \quad u(0, x) = u_0(x).$$

We add data-driven artificial viscosity, i.e., solve

$$u_t + (f(u))_x = \nu u_{xx}, \quad u(0, x) = u_0(x).$$

Here $\nu > 0$ is learned from data. when $u \in R^{2,3}$, e.g. Euler System, then?

[5] "Physics-informed Neural Networks with Adaptive Localized Artificial Viscosity", 2022.

PINN

Conclusion

Summary:

PINN

Conclusion

Summary:

- Feasible way to solve many different PDEs, including differential-integral equation and hyperbolic PDEs

PINN

Conclusion

Summary:

- Feasible way to solve many different PDEs, including differential-integral equation and hyperbolic PDEs
- Incorporate more physics constrains, such as adding artificial viscosity

PINN

Conclusion

Summary:

- Feasible way to solve many different PDEs, including differential-integral equation and hyperbolic PDEs
- Incorporate more physics constrains, such as adding artificial viscosity
- Handle stiff PDEs such Allen-Cahn with self-adaptive weights

PINN

Conclusion

Summary:

- Feasible way to solve many different PDEs, including differential-integral equation and hyperbolic PDEs
- Incorporate more physics constrains, such as adding artificial viscosity
- Handle stiff PDEs such Allen-Cahn with self-adaptive weights
- Similar to pseudo-spectral method, data-driven basis