Physics Informed Machine Learning for Solving PDEs

Ming Zhong

Department of Applied Mathematics
Illinois Institute of Technology

December 1, 2022

Numerical PDE Lecture 02

Table of Contents

Quick Recap

Last time

How to solve a simple Heat Equation with Dirichlet BC

Quick Recap

Last time

How to solve a simple Heat Equation with Dirichlet BC

- Separation of Variables (Heavy dependence of IC/BC)

Quick Recap

Last time

How to solve a simple Heat Equation with Dirichlet BC

- Separation of Variables (Heavy dependence of IC/BC)
    - Fourier expansion of the IC

Quick Recap

Last time

How to solve a simple Heat Equation with Dirichlet BC

- Separation of Variables (Heavy dependence of IC/BC)
    - Fourier expansion of the IC
    - Other BC (Neumann, Robin, Periodic) give different expression

Quick Recap

Last time

How to solve a simple Heat Equation with Dirichlet BC

- Separation of Variables (Heavy dependence of IC/BC)
  - Fourier expansion of the IC
  - Other BC (Neumann, Robin, Periodic) give different expression
- Finite Difference

Quick Recap

Last time

How to solve a simple Heat Equation with Dirichlet BC

- Separation of Variables (Heavy dependence of IC/BC)
  - Fourier expansion of the IC
  - Other BC (Neumann, Robin, Periodic) give different expression
- Finite Difference
  - Regular domain, exponential growth for dimension

Quick Recap

Last time

How to solve a simple Heat Equation with Dirichlet BC

- Separation of Variables (Heavy dependence of IC/BC)
    - Fourier expansion of the IC
    - Other BC (Neumann, Robin, Periodic) give different expression
- Finite Difference
    - Regular domain, exponential growth for dimension
    - Interpolation for inter-grid values, CFL condition

Quick Recap

Last time

How to solve a simple Heat Equation with Dirichlet BC

- Separation of Variables (Heavy dependence of IC/BC)
  - Fourier expansion of the IC
  - Other BC (Neumann, Robin, Periodic) give different expression
- Finite Difference
  - Regular domain, exponential growth for dimension
  - Interpolation for inter-grid values, CFL condition
- Physics Informed Neural Networks

Table of Contents

Physics Informed Machine Learning

Overview

Hybrid method

Physics Informed Machine Learning

Overview

Hybrid method

- Traditional Methods

Physics Informed Machine Learning

Overview

Hybrid method

- Traditional Methods
  - Speed and/or accuracy

Physics Informed Machine Learning

Overview

Hybrid method

- Traditional Methods
  - Speed and/or accuracy
  - Convergence

Physics Informed Machine Learning

Overview

Hybrid method

- Traditional Methods
  - Speed and/or accuracy
  - Convergence
  - Special Properties (Conservation, TVD, shock detection, etc.)

Physics Informed Machine Learning

Overview

Hybrid method

- Traditional Methods
  - Speed and/or accuracy
  - Convergence
  - Special Properties (Conservation, TVD, shock detection, etc.)
- Machine Learning

Physics Informed Machine Learning

Overview

Hybrid method

- Traditional Methods
  - Speed and/or accuracy
  - Convergence
  - Special Properties (Conservation, TVD, shock detection, etc.)
- Machine Learning
  - Mesh-free, no time integration

Physics Informed Machine Learning

Overview

Hybrid method

- Traditional Methods
  - Speed and/or accuracy
  - Convergence
  - Special Properties (Conservation, TVD, shock detection, etc.)
- Machine Learning
  - Mesh-free, no time integration
  - Auto-interpolation, auto-differentiation

Physics Informed Machine Learning

Overview

Hybrid method

- Traditional Methods
  - Speed and/or accuracy
  - Convergence
  - Special Properties (Conservation, TVD, shock detection, etc.)
- Machine Learning
  - Mesh-free, no time integration
  - Auto-interpolation, auto-differentiation
  - Data assimilation

Data Assimilation

Missing IC/BC[1]

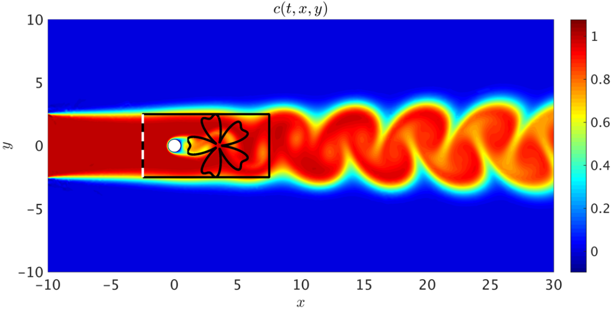


Figure: $2D$ Flow Past a Circular Cylinder.

[1] "Hidden Fluid Mechanics: A Navier-Stokes Informed Deep Learning Framework for Assimilating Flow Visualization Data", 2020.

Data Assimilation

HFM, cont.

$$
\begin{aligned}
c_t + uc_x + vc_y + wc_z - \mathrm{Pec}^{-1}(c_{xx} + c_{yy} + c_{zz}) &= 0,\\
d_t + ud_x + vd_y + wd_z - \mathrm{Pec}^{-1}(d_{xx} + d_{yy} + d_{zz}) &= 0,\\
u_t + uu_x + vu_y + wu_z + p_x + \mathrm{Re}^{-1}(u_{xx} + u_{yy} + u_{zz}) &= 0,\\
v_t + uv_x + vv_y + wv_z + p_y + \mathrm{Re}^{-1}(v_{xx} + v_{yy} + v_{zz}) &= 0,\\
w_t + uw_x + vw_y + ww_z + p_z + \mathrm{Re}^{-1}(w_{xx} + w_{yy} + w_{zz}) &= 0,\\
u_x + v_y + w_z &= 0
\end{aligned}
$$

With observation of c, d.

Other Forms of NN Solver

References

- DeepONet: "Learning Nonlinear Operators for Identifying Differential Equations based on the Universal Approximation Theorem of Operators", 2020.

Other Forms of NN Solver

References

- DeepONet: "Learning Nonlinear Operators for Identifying Differential Equations based on the Universal Approximation Theorem of Operators", 2020.
- LSNN: "Deep Least-squares Methods: an Unsupervied Learning-based Numerical Methods for Solving Elliptic PDEs", 2020.

Other Forms of NN Solver

References

- DeepONet: "Learning Nonlinear Operators for Identifying Differential Equations based on the Universal Approximation Theorem of Operators", 2020.
- LSNN: "Deep Least-squares Methods: an Unsupervied Learning-based Numerical Methods for Solving Elliptic PDEs", 2020.
- CVPINN: "Thermodynamically Consistent Physics-informed Neural Networks for Hyperbolic Systems", 2020.

Weaknesses

Multi-Object Losses[2]

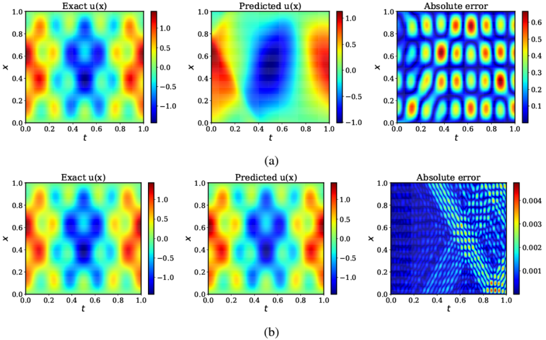


Figure: $1D$ Wave Equation ($u_{tt} = ku_{xx}$).

[2] "When and Why PINNs Fail to Train: a Neural Tanget Kernel Perspective", 2020.

Weaknesses

No Time Marching[3]

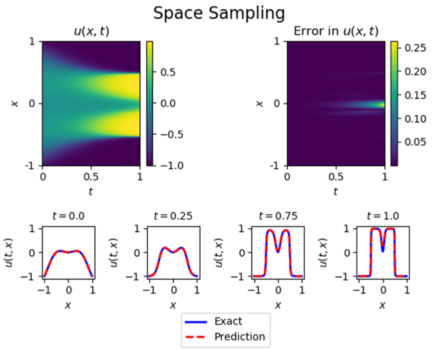
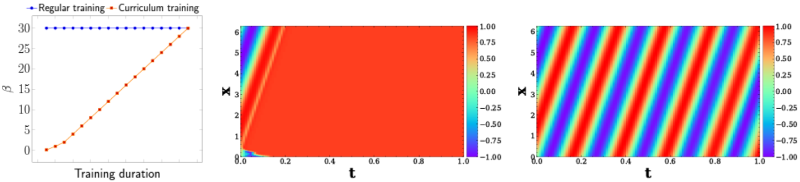


Figure: $1D$ Allen-Chan Equation.

[3] "Solving Allen-Cahn and Cahn-Hilliard Equations using the Adaptive Physics Informed Neural Networks", 2020.

Weakness

No Time Marching[4]



(a) *Curriculum regularization schematic*

(b) *Regular training PINN solution for $\beta = 30$*

(c) *Curriculum training PINN solution for $\beta = 30$*

Figure: $1D$ Transport Equation.

[4] "Characterizing Possible Failure Modes in Physics-informed Neural Networks", 2021.

Weaknesses

No Time Marching[5]

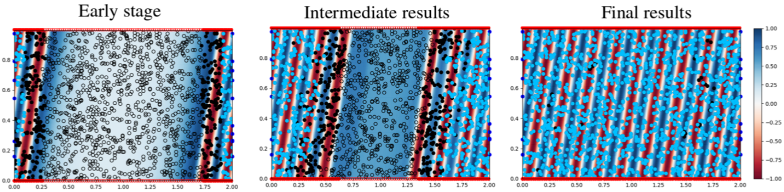


Figure: $1D$ Transport Equation.

[5] "Improved Training of Physics-informed Neural Networks with Model Ensembles", 2022.

Weaknesses

Shock Capturing[6]

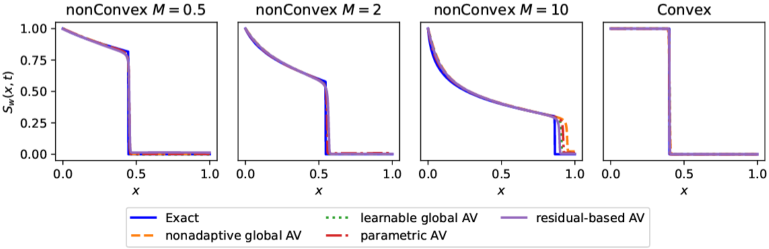


Figure: $1D$ Buckey-Leverett Equation.

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

Important References

Gaussian Processes for Machine Learning

Two books

- Gaussian Processes for Machine Learning, Rasmussen and Williams, 2006
- Operator-Adapted Wavelets, Fast Solvers, and Numerical Homogenization: from a Game Theory Approach to Numerical Approximation and Algorithm Design, Owhadi, 2019.

PIGP

Many Kinds

1 : "Machine Learning of Linear Differential Equations using Gaussian Processes", 2017.

PIGP

Many Kinds

1 : “Machine Learning of Linear Differential Equations using Gaussian Processes”, 2017.

2 : “Numerical Gaussian Processes for Time-Dependent and Nonlinear Partial Differential Equations”, 2018.

PIGP

Many Kinds

1 : “Machine Learning of Linear Differential Equations using Gaussian Processes”, 2017.

2 : “Numerical Gaussian Processes for Time-Dependent and Nonlinear Partial Differential Equations”, 2018.

3 : “Solving and Learning Nonlinear PDEs with Gaussian Processes”, 2021.

PIGP

Many Kinds

1 : “Machine Learning of Linear Differential Equations using Gaussian Processes”, 2017.

2 : “Numerical Gaussian Processes for Time-Dependent and Nonlinear Partial Differential Equations”, 2018.

3 : “Solving and Learning Nonlinear PDEs with Gaussian Processes”, 2021.

4 : “AutoIP: A United Framework to Integrate Physics into Gaussian Processes”, 2022.

PIGP

Many Kinds

1 : "Machine Learning of Linear Differential Equations using Gaussian Processes", 2017.

2 : "Numerical Gaussian Processes for Time-Dependent and Nonlinear Partial Differential Equations", 2018.

3 : "Solving and Learning Nonlinear PDEs with Gaussian Processes", 2021.

4 : "AutoIP: A United Framework to Integrate Physics into Gaussian Processes", 2022.

5 : Bayesian Numerical PDE, 2023.

GP for Linear PDEs

Approach One[7]

Consider

$$\mathcal{L}_{\mathbf{x}} u(\mathbf{x}) = f(\mathbf{x}), \quad \mathcal{L}_{\mathbf{x}} \text{ is a linear differential operator.}$$

[7] Reference [1].

GP for Linear PDEs

Approach One[7]

Consider

$$\mathcal{L}_{\mathbf{x}} u(\mathbf{x}) = f(\mathbf{x}), \quad \mathcal{L}_{\mathbf{x}} \text{ is a linear differential operator.}$$

Examples

[7] Reference [1].

GP for Linear PDEs

Approach One[7]

Consider

$$\mathcal{L}_{\mathbf{x}} u(\mathbf{x}) = f(\mathbf{x}), \quad \mathcal{L}_{\mathbf{x}} \text{ is a linear differential operator.}$$

Examples

- For $\Delta_{\mathbf{x}} u(\mathbf{x}) = f(\mathbf{x})$ (Laplace Equation), $\mathcal{L}_{\mathbf{x}} = \Delta_{\mathbf{x}}$.

[7] Reference [1].

GP for Linear PDEs

Approach One[7]

Consider

$$\mathcal{L}_{\mathbf{x}} u(\mathbf{x}) = f(\mathbf{x}), \quad \mathcal{L}_{\mathbf{x}} \text{ is a linear differential operator.}$$

Examples

- For $\Delta_{\mathbf{x}} u(\mathbf{x}) = f(\mathbf{x})$ (Laplace Equation), $\mathcal{L}_{\mathbf{x}} = \Delta_{\mathbf{x}}$.
- For $\partial_t u(\mathbf{x}) + c\partial_x u(\mathbf{x}) = f(\mathbf{x})$ (Linear Advection), $\mathcal{L}_{\mathbf{x}} = \partial_{x_1} + c\partial_{x_2}$ where $x_1 = t$ and $x_2 = x$ and $\mathbf{x} = (x_1, x_2)$.

[7] Reference [1].

GP for Linear PDEs

Approach One[7]

Consider

$$\mathcal{L}_{\mathbf{x}} u(\mathbf{x}) = f(\mathbf{x}), \quad \mathcal{L}_{\mathbf{x}} \text{ is a linear differential operator.}$$

Examples

- For $\Delta_{\mathbf{x}} u(\mathbf{x}) = f(\mathbf{x})$ (Laplace Equation), $\mathcal{L}_{\mathbf{x}} = \Delta_{\mathbf{x}}$.
- For $\partial_t u(\mathbf{x}) + c\partial_x u(\mathbf{x}) = f(\mathbf{x})$ (Linear Advection), $\mathcal{L}_{\mathbf{x}} = \partial_{x_1} + c\partial_{x_2}$ where $x_1 = t$ and $x_2 = x$ and $\mathbf{x} = (x_1, x_2)$.
- For $\partial_t u(\mathbf{x}) - k\partial_x^2 u(\mathbf{x}) = f(\mathbf{x})$ (Heat Equation), $\mathcal{L}_{\mathbf{x}} = \partial_{x_1} - k\partial_{x_2}^2$.

[7] Reference [1].

GP for Linear PDEs

Approach One, cont.

GP Prior

If $u(\mathbf{x}) \sim \mathcal{N}(0, K_{uu}(\mathbf{x}, \mathbf{x}'; \theta))$, then

$$f(\mathbf{x}) \sim \mathcal{N}(0, K_{ff}(\mathbf{x}, \mathbf{x}'; \theta))$$

and

$$K_{ff}(\mathbf{x}, \mathbf{x}'; \theta) = \mathcal{L}_{\mathbf{x}} \mathcal{L}_{\mathbf{x}'} K_{uu}(\mathbf{x}, \mathbf{x}'; \theta).$$

Here θ is the hyper-parameter for the GP.

GP for Linear PDEs

Approach One, cont.

Problem Statement

Given

$$\mathbf{y}_u = u(\mathbf{X}_u) + \epsilon_u \quad \text{and} \quad \mathbf{y}_f = f(\mathbf{X}_f) + \epsilon_f$$

where $\epsilon_u \sim \mathcal{N}(0, \sigma_u^2 \mathrm{Id}_u)$ and $\epsilon_f \sim \mathcal{N}(0, \sigma_f^2 \mathrm{Id}_f)$, how can we find the θ?

GP for Linear PDEs

Approach One, cont.

Problem Statement

Given

$$\mathbf{y}_u = u(\mathbf{X}_u) + \epsilon_u \quad \text{and} \quad \mathbf{y}_f = f(\mathbf{X}_f) + \epsilon_f$$

where $\epsilon_u \sim \mathcal{N}(0, \sigma_u^2 \mathrm{Id}_u)$ and $\epsilon_f \sim \mathcal{N}(0, \sigma_f^2 \mathrm{Id}_f)$, how can we find the θ?

Find θ such that $-\log(P(\mathbf{y}\Big|\theta, \sigma_u^2, \sigma_f^2))$ is minimized

GP for Linear PDEs

Approach One, cont.

Here

$$-\log(P(\mathbf{y}\Big|\theta, \sigma_u^2, \sigma_f^2)) = \frac{1}{2}\log(\big|K\big|) + \frac{1}{2}\mathbf{y}^\top K^{-1}\mathbf{y} + \frac{N}{2}\log(2\pi),$$

where

$$\mathbf{y} = \begin{bmatrix}\mathbf{y}_u \\ \mathbf{y}_f\end{bmatrix} \quad \text{and} \quad \mathbf{y}\Big|\theta, \sigma_u^2, \sigma_f^2 \sim \mathcal{N}(0, K),$$

GP for Linear PDEs

Approach One, cont.

$$K = \begin{bmatrix} K_{uu}(\mathbf{X}_u, \mathbf{X}_u; \theta) + \sigma_u^2 \mathrm{Id}_u & K_{uf}(\mathbf{X}_u, \mathbf{X}_f; \theta) \\ K_{fu}(\mathbf{X}_f, \mathbf{X}_u; \theta) & K_{ff}(\mathbf{X}_f, \mathbf{X}_f; \theta) + \sigma_f^2 \mathrm{Id}_f \end{bmatrix}.$$

Moreover

$$K_{uf}(\mathbf{x}, \mathbf{x}'; \theta) = \mathcal{L}_{\mathbf{x}'} K_{uu}(\mathbf{x}, \mathbf{x}'; \theta) \quad \text{and} \quad \theta)$$

and

$$K_{fu}(\mathbf{x}, \mathbf{x}'; \theta) = \mathcal{L}_{\mathbf{x}} K_{uu}(\mathbf{x}, \mathbf{x}').$$

The minimizater is found using L-BFGS.

GP for Linear PDEs

Approach One, cont.

Next?

How to predict?

GP for Linear PDEs

Approach One, cont.

Next?

How to predict?

Use the fact that

$$u(\mathbf{x})\Big|\mathbf{y} \sim \mathcal{N}(\bar{u}(\mathbf{x}), s_u^2(\mathbf{x}))$$

and

$$f(\mathbf{x})\Big|\mathbf{y} \sim \mathcal{N}(\bar{f}(\mathbf{x}), s_f^2(\mathbf{x})),$$

GP for Linear PDEs

Approach One, cont.

Where

$$\bar{u}(\mathbf{x}) = \mathbf{q}_u^\top K^{-1}\mathbf{y}, \quad s_u^2(\mathbf{x}) = K_{uu}(\mathbf{x},\mathbf{x}) - \mathbf{q}_u^\top K^{-1}\mathbf{q}_u,$$
$$\mathbf{q}_u^\top(\mathbf{x}) = \begin{bmatrix} K_{uu}(\mathbf{x},\mathbf{X}_u) & K_{uf}(\mathbf{x},\mathbf{X}_f) \end{bmatrix}.$$

and

$$\bar{f}(\mathbf{x}) = \mathbf{q}_f^\top K^{-1}\mathbf{y}, \quad s_f^2(\mathbf{x}) = K_{ff}(\mathbf{x},\mathbf{x}) - \mathbf{q}_f^\top K^{-1}\mathbf{q}_f,$$
$$\mathbf{q}_f^\top(\mathbf{x}) = \begin{bmatrix} K_{fu}(\mathbf{x},\mathbf{X}_u) & K_{ff}(\mathbf{x},\mathbf{X}_f) \end{bmatrix}.$$

GP for Linear PDEs

Conclusion

Summary

GP for Linear PDEs

Conclusion

Summary

- Clean implementation.

GP for Linear PDEs

Conclusion

Summary

- Clean implementation.
- Easy to train.

GP for Linear PDEs

Conclusion

Summary

- Clean implementation.
- Easy to train.
- Uncertainty quantification.

GP for Linear PDEs

Conclusion

Summary

- Clean implementation.
- Easy to train.
- Uncertainty quantification.
- Differentiation is done on the kernel function.

GP for Linear PDEs

Conclusion

Summary

- Clean implementation.
- Easy to train.
- Uncertainty quantification.
- Differentiation is done on the kernel function.

But

- Only for linear PDEs.
- Application when there are data for u and f.

GP for Time-dependent PDEs

Approach Two[8]

Consider

$$u_t = \mathcal{L}_{\mathbf{x}} u, \quad \mathbf{x} \in \Omega, \quad t \in [0, T].$$

[8] Reference [2].

GP for Time-dependent PDEs

Approach Two[8]

Consider

$$u_t = \mathcal{L}_{\mathbf{x}} u, \quad \mathbf{x} \in \Omega, \quad t \in [0, T].$$

Via the explicit Euler scheme for u_t, we end up with

$$u^n = u^{n-1} + \Delta t \mathcal{L}_{\mathbf{x}} u^{n-1} = \mathcal{B}_x(\Delta t) u^{n-1},$$

Here

$$u^n(\mathbf{x}) = u(t_n, \mathbf{x}).$$

[8] Reference [2].

GP for Time-dependent PDEs

Approach Two, cont.

Put on a Gaussian Process prior on u^{n-1}, i.e.

$$u(t_{n-1}, \mathbf{x}) \sim \mathcal{N}(0, K_{uu}^{n-1,n-1}(\mathbf{x}, \mathbf{x}'; \theta)),$$

GP for Time-dependent PDEs

Approach Two, cont.

Put on a Gaussian Process prior on u^{n-1}, i.e.

$$u(t_{n-1}, \mathbf{x}) \sim \mathcal{N}(0, K_{uu}^{n-1,n-1}(\mathbf{x}, \mathbf{x}'; \theta)),$$

then

$$\begin{bmatrix} u(t_n, \mathbf{x}) \\ u(t_{n-1}, \mathbf{x}) \end{bmatrix} \sim \mathcal{N}(0, \begin{bmatrix} K_{uu}^{n,n} & K_{uu}^{n,n-1} \\ k_{uu}^{n-1,n} & K_{uu}^{n-1,n-1} \end{bmatrix}).$$

GP for Time-dependent PDEs

Conclusion

Summary

GP for Time-dependent PDEs

Conclusion

Summary

- Able to handle time-dependent PDE.

GP for Time-dependent PDEs

Conclusion

Summary

- Able to handle time-dependent PDE.
- No need to have observation data inside the computational domain (start out from IC/BC).

GP for Time-dependent PDEs

Conclusion

Summary

- Able to handle time-dependent PDE.
- No need to have observation data inside the computational domain (start out from IC/BC).
- Applicable to other numerical integration schemes (even implicit).

GP for Time-dependent PDEs

Conclusion

Summary

- Able to handle time-dependent PDE.
- No need to have observation data inside the computational domain (start out from IC/BC).
- Applicable to other numerical integration schemes (even implicit).

But

- Different numerical integration schemes would require different $\mathcal{B}_x(\Delta t)$.
- For nonlinear PDEs, need to linearize it using u^{n-1}.

PIGP for Nonlinear PDE

Approach Three[9]

A Simple Nonlinear Elliptic PDE Problem

Consider $\Omega \subset \mathbb{R}^d$ ($d \geq 1$, open and bounded) with Lipshitz boundary $\partial\Omega$, and for $u^* : \Omega \to \mathbb{R}$ satisfying

$$\begin{aligned} -\Delta u^*(\mathbf{x}) + \tau(u^*(\mathbf{x})) = f(\mathbf{x}), &\quad \mathbf{x} \in \Omega \\ u^*(\mathbf{x}) = g(\mathbf{x}), &\quad \mathbf{x} \in \partial\Omega. \end{aligned}$$

Here $f : \Omega \to \mathbb{R}$, $g : \partial\Omega \to \mathbb{R}$, and $\tau : \mathbb{R} \to \mathbb{R}$ are given continuous functions.

[9]Reference [3].

PIGP for Nonlinear PDE

Approach Three, cont.

The PIGP method finds the approximation to u^* from the following

$$\min_{u \in \mathcal{U}} \left|u\right|_K, \quad \text{s.t.}$$
$$-\Delta u(\mathbf{x}_m) + \tau(u(\mathbf{x}_m)) = f(\mathbf{x}_m), \quad m = 1, \ldots, M_\Omega$$
$$u(\mathbf{x}_m) = g(\mathbf{x}_m), \quad m = M_\Omega + 1, \ldots, M.$$

PIGP for Nonlinear PDE

Approach Three, cont.

The PIGP method finds the approximation to u^* from the following

$$
\begin{aligned}
&\min_{u \in \mathcal{U}} \left|u\right|_K, \quad \text{s.t.} \\
&-\Delta u(\mathbf{x}_m) + \tau(u(\mathbf{x}_m)) = f(\mathbf{x}_m), \quad m = 1, \dots, M_\Omega \\
&u(\mathbf{x}_m) = g(\mathbf{x}_m), \quad m = M_\Omega + 1, \dots, M.
\end{aligned}
$$

Here

- $\mathcal{U}$ is a RKHS (Reproducing Kernel Hilbert Space) associated with K and the RKHS norm is $\left|\cdot\right|_K$.

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Moreover

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Moreover

- A non-degenerate, symmetric, and positive definite kernel function $K : \bar{\Omega} \times \bar{\Omega} \to \mathbb{R}$.

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Moreover

- A non-degenerate, symmetric, and positive definite kernel function $K : \bar{\Omega} \times \bar{\Omega} \to \mathbb{R}$.
- K is chosen so that $\mathcal{U} \in C^2(\Omega) \cap C(\bar{\Omega})$.

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Moreover

- A non-degenerate, symmetric, and positive definite kernel function $K : \bar{\Omega} \times \bar{\Omega} \to \mathbb{R}$.
- K is chosen so that $\mathcal{U} \in C^2(\Omega) \cap C(\bar{\Omega})$.
- $1 \leq M_\Omega < M < \infty$ with $\mathbf{x}_1, \ldots, \mathbf{x}_{M_\Omega} \in \Omega$ and $\mathbf{x}_{M_\Omega+1}, \ldots, \mathbf{x}_M \in \partial\Omega$.

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Moreover

- A non-degenerate, symmetric, and positive definite kernel function $K : \bar{\Omega} \times \bar{\Omega} \to \mathbb{R}$.
- K is chosen so that $\mathcal{U} \in C^2(\Omega) \cap C(\bar{\Omega})$.
- $1 \leq M_\Omega < M < \infty$ with $\mathbf{x}_1, \ldots, \mathbf{x}_{M_\Omega} \in \Omega$ and $\mathbf{x}_{M_\Omega+1}, \ldots, \mathbf{x}_M \in \partial\Omega$.
- A minimier $u^\dagger$ can be interpreted as a MAP estimator of a GP $\xi \sim \mathcal{N}(0, \mathcal{K})$ ($\mathcal{K}$ is the integral operator with kernel K).

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Finite Dimensional Representation

$$z_m^{(1)} = u(\mathbf{x}_m) \quad \text{and} \quad z_m^{(2)} = -\Delta u(\mathbf{x}_m),$$

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Finite Dimensional Representation

$$z_m^{(1)} = u(\mathbf{x}_m) \quad \text{and} \quad z_m^{(2)} = -\Delta u(\mathbf{x}_m),$$

Define

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Finite Dimensional Representation

$$z_m^{(1)} = u(\mathbf{x}_m) \quad \text{and} \quad z_m^{(2)} = -\Delta u(\mathbf{x}_m),$$

Define

- $\phi_m^{(1)} = \delta_{\mathbf{x}_m}$ and $\phi_m^{(2)} = \delta_{\mathbf{x}_m} \circ (-\Delta)$ where $\delta_{\mathbf{x}}$ is the Dirac delta function centered at $\mathbf{x}$.

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Finite Dimensional Representation

$$z_m^{(1)} = u(\mathbf{x}_m) \quad \text{and} \quad z_m^{(2)} = -\Delta u(\mathbf{x}_m),$$

Define

- $\phi_m^{(1)} = \delta_{\mathbf{x}_m}$ and $\phi_m^{(2)} = \delta_{\mathbf{x}_m} \circ (-\Delta)$ where $\delta_{\mathbf{x}}$ is the Dirac delta function centered at $\mathbf{x}$.
- $\mathbf{z} = \left[(\mathbf{z}^{(1)})^\top \quad (\mathbf{z}^{(2)})^\top \right]$ where $\mathbf{z}^{(i)}$ is the concatenation of $z_m^{(i)}$ for $i = 1, 2$ respectively.

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Finite Dimensional Representation

$$z_m^{(1)} = u(\mathbf{x}_m) \quad \text{and} \quad z_m^{(2)} = -\Delta u(\mathbf{x}_m),$$

Define

- $\phi_m^{(1)} = \delta_{\mathbf{x}_m}$ and $\phi_m^{(2)} = \delta_{\mathbf{x}_m} \circ (-\Delta)$ where $\delta_{\mathbf{x}}$ is the Dirac delta function centered at $\mathbf{x}$.
- $\mathbf{z} = \left[(\mathbf{z}^{(1)})^\top \quad (\mathbf{z}^{(2)})^\top\right]$ where $\mathbf{z}^{(i)}$ is the concatenation of $z_m^{(i)}$ for $i = 1, 2$ respectively.
- $\phi = \left[(\phi^{(1)})^\top \quad (\phi^{(2)})^\top\right]$ where $\phi^{(i)}$ is the concatenation of $\phi_m^{(i)}$ for $i = 1, 2$ respectively.

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Furthermore

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Furthermore

- $\Theta(\mathbf{x}, \phi)$ is the $(M + M_\Omega)$-dim vector with entries $\int K(\mathbf{x}, \mathbf{x}')\phi_m(\mathbf{x}')\, d\mathbf{x}'$.

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Furthermore

- $\Theta(\mathbf{x}, \phi)$ is the $(M + M_\Omega)$-dim vector with entries $\int K(\mathbf{x}, \mathbf{x}')\phi_m(\mathbf{x}')\,d\mathbf{x}'$.
- $\Theta(\phi, \phi)$ is the $(M + M_\Omega) \times (M + M_\Omega)$ matrix with entries $\int K(\mathbf{x}, \mathbf{x}')\phi_m(\mathbf{x})\phi_n(\mathbf{x}')\,d\mathbf{x}d\mathbf{x}'$.

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Furthermore

- $\Theta(\mathbf{x}, \phi)$ is the $(M + M_\Omega)$-dim vector with entries $\int K(\mathbf{x}, \mathbf{x}')\phi_m(\mathbf{x}')\, d\mathbf{x}'$.
- $\Theta(\phi, \phi)$ is the $(M + M_\Omega) \times (M + M_\Omega)$ matrix with entries $\int K(\mathbf{x}, \mathbf{x}')\phi_m(\mathbf{x})\phi_n(\mathbf{x}')\, d\mathbf{x}d\mathbf{x}'$.

Thus

$$u(\mathbf{x}) = \Theta(\mathbf{x}, \phi)\Theta(\phi, \phi)^{-1}\mathbf{z};$$

and

$$\left|u\right|_K^2 = \mathbf{z}^\top\Theta(\phi, \phi)^{-1}\mathbf{z}$$

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Define

$$F(\mathbf{z}) = \begin{cases} z_m^{(2)} + \tau(z_m^{(1)}), & m = 1, \dots, M_\Omega \\ z_m^{(1)}, & m = M_\Omega + 1, \dots, M \end{cases}$$

and

$$\mathbf{y} = \begin{cases} f(\mathbf{x}_m), & m = 1, \dots, M_\Omega \\ g(\mathbf{x}_m), & m = M_\Omega + 1, \dots, M \end{cases}$$

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Define

$$F(\mathbf{z}) = \begin{cases} z_m^{(2)} + \tau(z_m^{(1)}), & m = 1, \ldots, M_\Omega \\ z_m^{(1)}, & m = M_\Omega + 1, \ldots, M \end{cases}$$

and

$$\mathbf{y} = \begin{cases} f(\mathbf{x}_m), & m = 1, \ldots, M_\Omega \\ g(\mathbf{x}_m), & m = M_\Omega + 1, \ldots, M \end{cases}$$

The original optimal recovery problem becomes

$$\min_{\mathbf{z} \in \mathbb{R}^{M+M_\Omega}} \mathbf{z}^\top \Theta(\phi, \phi)^{-1} \mathbf{z}, \quad \text{s.t.} \quad F(\mathbf{z}) = \mathbf{y}.$$

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Or the regularized version

$$\min_{\mathbf{z}\in\mathbb{R}^{M+M_\Omega}} \mathbf{z}^\top \Theta(\phi,\phi)^{-1}\mathbf{z} + \frac{1}{\beta^2}\left|F(\mathbf{z}) - \mathbf{y}\right|^2_{\mathbb{R}^{M+M_\Omega}}.$$

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Or the regularized version

$$\min_{\mathbf{z}\in\mathbb{R}^{M+M_\Omega}} \mathbf{z}^\top \Theta(\phi,\phi)^{-1}\mathbf{z} + \frac{1}{\beta^2}\left|F(\mathbf{z}) - \mathbf{y}\right|^2_{\mathbb{R}^{M+M_\Omega}}.$$

- Choosing a proper β?

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Or the regularized version

$$\min_{\mathbf{z}\in\mathbb{R}^{M+M_\Omega}} \mathbf{z}^\top \Theta(\phi,\phi)^{-1}\mathbf{z} + \frac{1}{\beta^2}\left|F(\mathbf{z}) - \mathbf{y}\right|^2_{\mathbb{R}^{M+M_\Omega}}.$$

- Choosing a proper β?
- Why does β have to be a scalar?

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Or the regularized version

$$\min_{\mathbf{z}\in\mathbb{R}^{M+M_\Omega}} \mathbf{z}^\top \Theta(\phi,\phi)^{-1}\mathbf{z} + \frac{1}{\beta^2}\left|F(\mathbf{z}) - \mathbf{y}\right|^2_{\mathbb{R}^{M+M_\Omega}}.$$

- Choosing a proper β?
- Why does β have to be a scalar?
- Somewhat easier than the constrained problem

PIGP for Nonlinear PDE

Nonlinear Elliptic PDEs, cont.

Or the regularized version

$$\min_{\mathbf{z}\in\mathbb{R}^{M+M_\Omega}} \mathbf{z}^\top \Theta(\phi,\phi)^{-1}\mathbf{z} + \frac{1}{\beta^2}\left|F(\mathbf{z}) - \mathbf{y}\right|^2_{\mathbb{R}^{M+M_\Omega}}.$$

- Choosing a proper β?
- Why does β have to be a scalar?
- Somewhat easier than the constrained problem
- But the constrained problem can be reduced further

PIGP for Nonlinear PDE

Numerical Tests

We take

- $d = 2$, $\Omega = (0, 1)^2$, $\tau(u) = 0$ or u^3, and $g(\mathbf{x}) = 0$.
- $u^*(\mathbf{x}) = \sin(\pi x_1)\sin(\pi x_2) + 4\sin(4\pi x_1)\sin(4\pi x_2)$, and find $f(\mathbf{x})$ accordingly.
- Gaussian kernel

$$K(\mathbf{x}, \mathbf{x}'; \sigma) = \exp\left(-\frac{\left|\mathbf{x} - \mathbf{x}'\right|^2_{\ell_2(\mathbb{R}^d)}}{2\sigma^2}\right).$$

- Special care of Θ.
- $M = 1024$ and $M_\Omega = 900$ and $\sigma = M^{-\frac{1}{4}}$.
- $\eta = 10^{-5}$ and $\beta = 10^{-5}$.

PIGP for Nonlinear PDE

A Simple Heat Equation

Recall

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

so $(\mathbf{x} = (t,x))$

$$z_m^{(1)} = u(\mathbf{x}_m), \quad z_m^{(2)} = u_t(\mathbf{x}_m), \quad \text{and} \quad z_m^{(3)} = u_{xx}(\mathbf{x}_m).$$

Here

$$\phi_m^{(1)} = \delta_{\mathbf{x}_m}, \quad \phi_m^{(2)} = \delta_{\mathbf{x}_m} \circ \frac{\partial}{\partial t}, \quad \text{and} \quad \phi_m^{(3)} = \delta_{\mathbf{x}_m} \circ \frac{\partial^2}{\partial x^2}.$$

PIGP for Nonlinear PDE

A Simple Heat Equation, cont.

Thus

$$
\begin{aligned}
\text{PDE}: & 0 * z_m^{(1)} + z_m^{(2)} - \lambda z_m^{(3)} = 0, \\
\text{BC}: & z_m^{(1)} = \begin{cases} f(x_m), & \mathbf{x}_m \in \{0\} \times [0, L] \\ 0, & \mathbf{x}_m \in [0, T] \times \{0\} \\ 0, & \mathbf{x}_m \in [0, T] \times \{L\} \end{cases}
\end{aligned}
$$

PIGP for Nonlinear PDE

A Simple Heat Equation, cont.

Thus

$$
\begin{aligned}
\text{PDE}: & 0 * z_m^{(1)} + z_m^{(2)} - \lambda z_m^{(3)} = 0, \\
\text{BC}: & z_m^{(1)} = \begin{cases} f(x_m), & \mathbf{x}_m \in \{0\} \times [0, L] \\ 0, & \mathbf{x}_m \in [0, T] \times \{0\} \\ 0, & \mathbf{x}_m \in [0, T] \times \{L\} \end{cases}
\end{aligned}
$$

Demo

PIGP for Nonlinear PDE

Conclusion

PIGP can even handle

$$\left|\nabla u(\mathbf{x})\right|^2 = f(\mathbf{x})^2 + \epsilon\Delta u(\mathbf{x}), \quad \text{Eikonal PDE}$$

and

$$u_t + uu_x = \nu u_{xx}, \quad \text{Viscous Burgers}$$

Summary:

PIGP for Nonlinear PDE

Conclusion

PIGP can even handle

$$\left|\nabla u(\mathbf{x})\right|^2 = f(\mathbf{x})^2 + \epsilon \Delta u(\mathbf{x}), \quad \text{Eikonal PDE}$$

and

$$u_t + u u_x = \nu u_{xx}, \quad \text{Viscous Burgers}$$

Summary:

- Smaller number of “collocation” points

PIGP for Nonlinear PDE

Conclusion

PIGP can even handle

$$\left|\nabla u(\mathbf{x})\right|^2 = f(\mathbf{x})^2 + \epsilon \Delta u(\mathbf{x}), \quad \text{Eikonal PDE}$$

and

$$u_t + u u_x = \nu u_{xx}, \quad \text{Viscous Burgers}$$

Summary:

- Smaller number of “collocation” points
- Uncertainty quantification

PIGP for Nonlinear PDE

Conclusion

PIGP can even handle

$$\left|\nabla u(\mathbf{x})\right|^2 = f(\mathbf{x})^2 + \epsilon \Delta u(\mathbf{x}), \quad \text{Eikonal PDE}$$

and

$$u_t + u u_x = \nu u_{xx}, \quad \text{Viscous Burgers}$$

Summary:

- Smaller number of “collocation” points
- Uncertainty quantification
- $\Omega \subset \mathbb{R}^d$ with $d \gg 1$ and irreuglar.

PIGP for Nonlinear PDE

Conclusion

PIGP can even handle

$$\left|\nabla u(\mathbf{x})\right|^2 = f(\mathbf{x})^2 + \epsilon \Delta u(\mathbf{x}), \quad \text{Eikonal PDE}$$

and

$$u_t + u u_x = \nu u_{xx}, \quad \text{Viscous Burgers}$$

Summary:

- Smaller number of "collocation" points
- Uncertainty quantification
- $\Omega \subset \mathbb{R}^d$ with $d \gg 1$ and irreuglar.
- Auto-interpolation and auto-differentiation

PIGP for Nonlinear PDE

Conclusion

PIGP can even handle

$$\left|\nabla u(\mathbf{x})\right|^2 = f(\mathbf{x})^2 + \epsilon \Delta u(\mathbf{x}), \quad \text{Eikonal PDE}$$

and

$$u_t + u u_x = \nu u_{xx}, \quad \text{Viscous Burgers}$$

Summary:

- Smaller number of "collocation" points
- Uncertainty quantification
- $\Omega \subset \mathbb{R}^d$ with $d \gg 1$ and irreuglar.
- Auto-interpolation and auto-differentiation
- However, heavy dependence on kernel

Table of Contents

Deep Learning Software

How to install them

- Python
  - TensorFlow: https://www.tensorflow.org/
  - PyTorch: https://pytorch.org/
  - JAX: https://jax.readthedocs.io/en/latest/
- Julia: SciML package https://sciml.ai/
- MATLAB
- Google Colab or Amazon AWS (Microsft? IBM?)

They can work with

- CPU (Slow!!)
- GPU (CUDA or M1/M2 Chips)

Table of Contents

Physics Informed Machine Learning

Conclusion

We have shown

- Solving a heat equation, viscous Burgers, and RTE using PINN.
- Solving elliptic PDEs, a heat equation, viscous Burgers, Eikonal using PIGP.
- Both methods have their own strength and weaknesses: auto-interpolation, mesh-less, etc.

Thank You!