A scalable adversarial learning approach using natural gradients for solving partial differential equations

Shu Liu (UCLA), Stanley Osher (UCLA), Wuchen Li (U of SC)

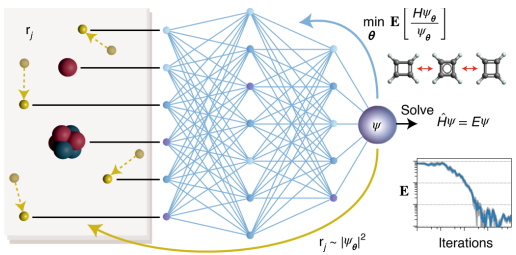
Department of Mathematics, UCLA

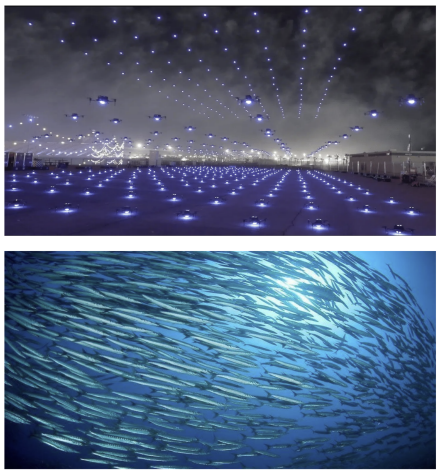
January 20, 2025

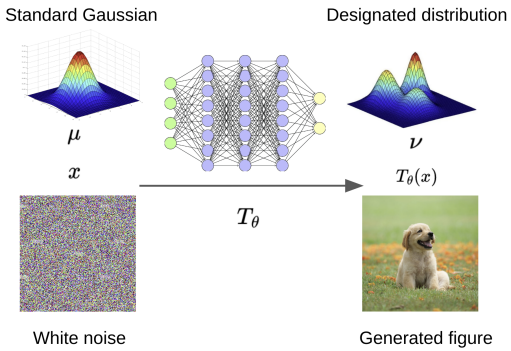
Table of Contents

Computing high-dimensional PDEs

- Growing demand for solving high-dimensional partial differential equations (PDEs) across diverse fields[1]:







- Due to *curse of dimensionality*, classical numerical methods face challenges.

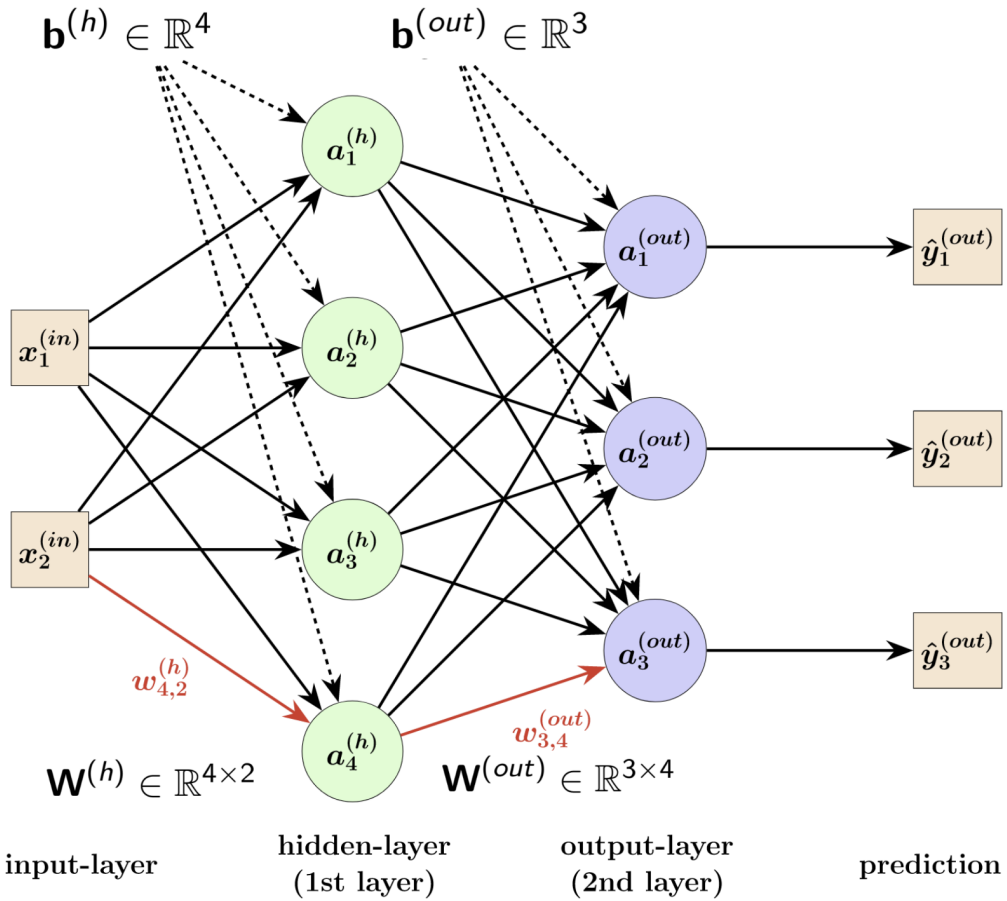
[1] J. Hermann, Z. Schätzle, F. Noé. Deep-neural-network solution of the electronic Schrödinger equation. Nature Chemistry, 2020.

Deep learning algorithms for PDEs

- **Solving specific PDE**:
  - Minimizing residue:
    - Physics-informed neural networks (**PINN**s) [Raissi, et al. 2019], etc.
  - Variational formula:
    - The Deep Ritz method (**DeepRitz**) [Yu, et al. 2017], etc.
  - Adversarial training:
    - Weak adversarial networks (**WAN**s) [Zang, et al. 2020], etc.
    - Residual-attention-based approach [McClenny, et al. 2020], etc.
  - Forward and Backward SDE (FBSDE) [E, et al. 2017], etc.
  - And many more...
- Operator learning:
  - Deep Operator Network [Lu, et al. 2021], Fourier Neural Operator [Li, et al. 2021], In-context operator learning [Yang, et al. 2023], etc.
  - Multi-Operator Multimodal learning [Liu, et al. 2023][Zhang, et al. 2024], etc.

What is a Neural Network?

A fully connected neural network: Multilayer Perceptron (MLP)



MLP: $f_\theta(\cdot) : \mathbb{R}^2 \to \mathbb{R}^3$.

- Input: $\mathbf{x}^{(in)} \in \mathbb{R}^2$,
- $\mathbf{a}^{(h)} = \sigma(\mathbf{W}^{(h)}\mathbf{x}^{(h)} + \mathbf{b}^{(h)})$,
- $\mathbf{a}^{(out)} = \sigma(\mathbf{W}^{(out)}\mathbf{a}^{(h)} + \mathbf{b}^{(out)})$,
- Output: $f_\theta(\mathbf{x}^{(in)}) = \mathbf{a}^{(out)}$.

σ: nonlinear activation function.

Parameter:
$\theta = (\mathbf{W}^{(h)}, \mathbf{b}^{(h)}, \mathbf{W}^{(out)}, \mathbf{b}^{(out)})$.

[1] https://www.geo.fu-berlin.de/en/v/soga-r/index.html

Solving PDEs via deep learning

- Poisson equation:

$$-\Delta u = f, \text{ on } \Omega, \quad u = g, \text{ on } \partial\Omega. \tag{1}$$

  Assume $\Omega \subset \mathbb{R}^d$ is a bounded open region.
  Suppose (1) admits a unique solution u_*.

- We substitute $u(\cdot) : \mathbb{R}^d \to \mathbb{R}$ with a neural network $u_\theta(\cdot) : \mathbb{R}^d \to \mathbb{R}$ with trainable parameter $\theta \in \Theta \subset \mathbb{R}^m$.

- How we compute for an approximation solution $u_\theta(\cdot)$ of (1)?
  - Propose a loss function $L(\theta)$ that quantifies the discrepancy between u_θ and u_*.
  - Optimize $L(\theta)$ w.r.t. θ.

Paradigm of deep learning PDE solver

Algorithm Paradigm of deep learning PDE solver

Require: Given the PDE, neural network u_θ, the loss function $L(\theta)$.

1: Initialize the parameters θ, of the neural network(s) $u_\theta(\cdot)$.
2: **for** $iter = 1$ to N_{iter} **do**
3: Apply Monte-Carlo algorithm to approximate $L(\theta)$,
4: Apply auto-differentiation to evaluate $\nabla_\theta L(\theta)$,
5: Apply the optimizer with stepsize τ_u to update θ.
6: **end for**

Ensure: $u_\theta(\cdot)$

- We mainly focus on $L(\theta)$ and the optimizer in this research.

Key ingredients of a deep learning PDE solver

- Loss function $L(\theta)$
  - PINN: Minimize L^2 norm of residue.

$$\begin{aligned} L(\theta) &= \int_\Omega |-\Delta u_\theta - f|^2 d\mu + \lambda \int_{\partial\Omega} |u_\theta - g|^2 d\mu_{\partial\Omega} \\ &= \| -\Delta(u_\theta - u_*)\|^2_{L^2(\Omega)} + \lambda \|u_\theta - u_*\|^2_{L^2(\partial\Omega)}. \end{aligned}$$

Pathologies due to high differential order[2].

  - DeepRitz: Optimize the energy functional,

$$\begin{aligned} L(\theta) &= \int_\Omega \frac{1}{2}\|\nabla u_\theta\|^2 - f u_\theta d\mu + \lambda \int_{\partial\Omega} |u_\theta - g|^2 d\mu_{\partial\Omega} \\ &= \|\nabla(u_\theta - u_*)\|^2_{L^2(\Omega)} + \text{Const} + \lambda \|u_\theta - u_*\|^2_{L^2(\partial\Omega)}. \end{aligned}$$

Only applicable to PDEs with the variational formulas.

[2] Y. Park, C. Song, and M. Kang. Beyond Derivative Pathology of PINNs: Variable Splitting Strategy with Convergence Analysis. JMLR, 2024

Key ingredients of a deep learning PDE solver

- Loss function $L(\theta)$ (continued)
  - WAN: introduce a test function $\varphi_\eta \in H_0^1(\Omega)$, integration by parts yields

$$\begin{aligned} L(\theta,\eta) =& 2\log\left|\left\langle -\Delta u_\theta - f, \frac{\varphi_\eta}{\|\varphi_\eta\|}\right\rangle\right| + \lambda\|u_\theta - g\|_{L^2(\partial\Omega)} \\ =& \log\left(\frac{\langle\nabla(u_\theta - u_*), \nabla\varphi_\eta\rangle^2}{\|\varphi_\eta\|^2}\right) + \lambda\|u_\theta - u_*\|^2_{L^2(\partial\Omega)}. \end{aligned}$$

We consider

$$\inf_\theta \sup_\eta \; L(\theta,\eta).$$

Versatile scheme. But a challenging saddle point problem involving neural networks!

Key ingredients of a deep learning PDE solver

- Optimizer
  - First order optimizers:
    - Basic: Stochastic Gradient Descent (SGD)
    - Accelerated: Nesterov method...
    - With per-parameter learning rates: AdaGrad, RMSProp, **Adam**,... Fast but usually suffers from strong fluctuations.
  - Second order optimizers:
    - Quasi-Newton: **L-BFGS** method. Unstable in training.
    - **Natural gradient** method[Amari, 1998]. Broad applications. Fast convergence, robust w.r.t. random batch training

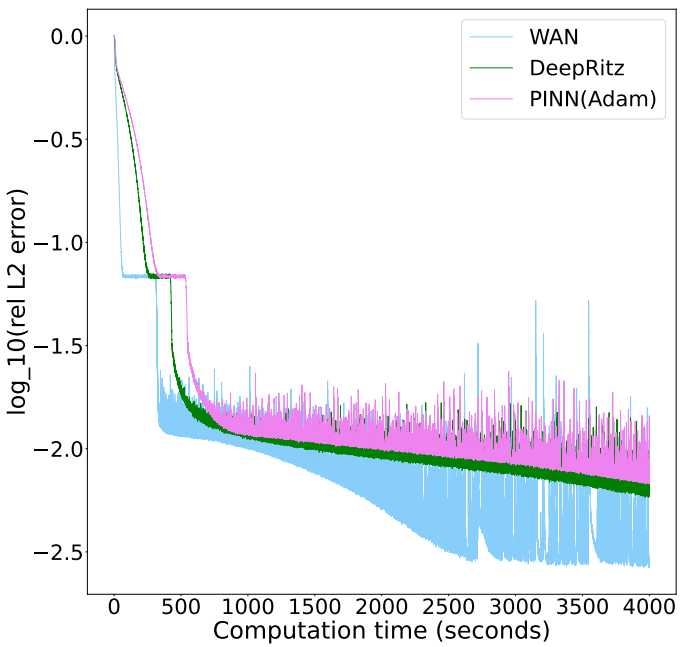
Performances of existing methods

Relative L^2 error vs. wall clock time(s).

- Sampling: uniform sampler on Ω and $\partial\Omega$;
- NN architecture: Multilayer Perceptron (MLP);

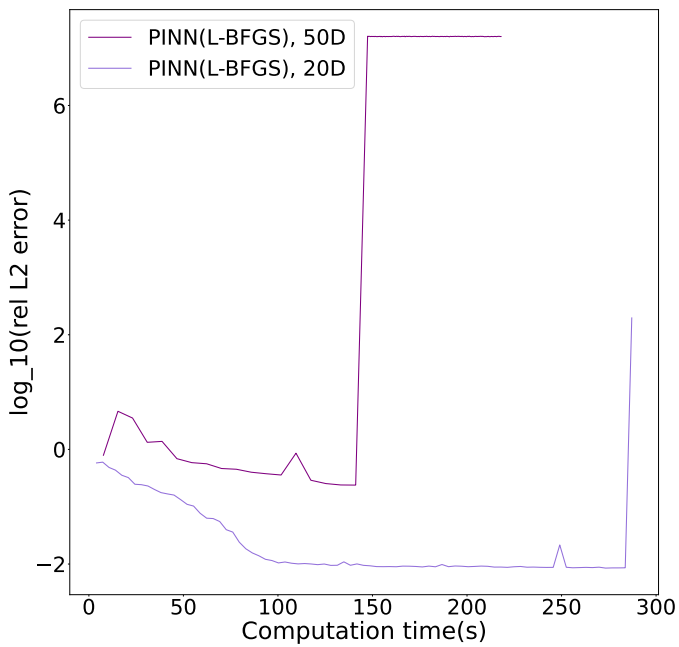
PINN/DeepRitz/WAN + Adam

50D, Poisson equation



PINN + L-BFGS

PINN with L-BFGS (20D 50D)

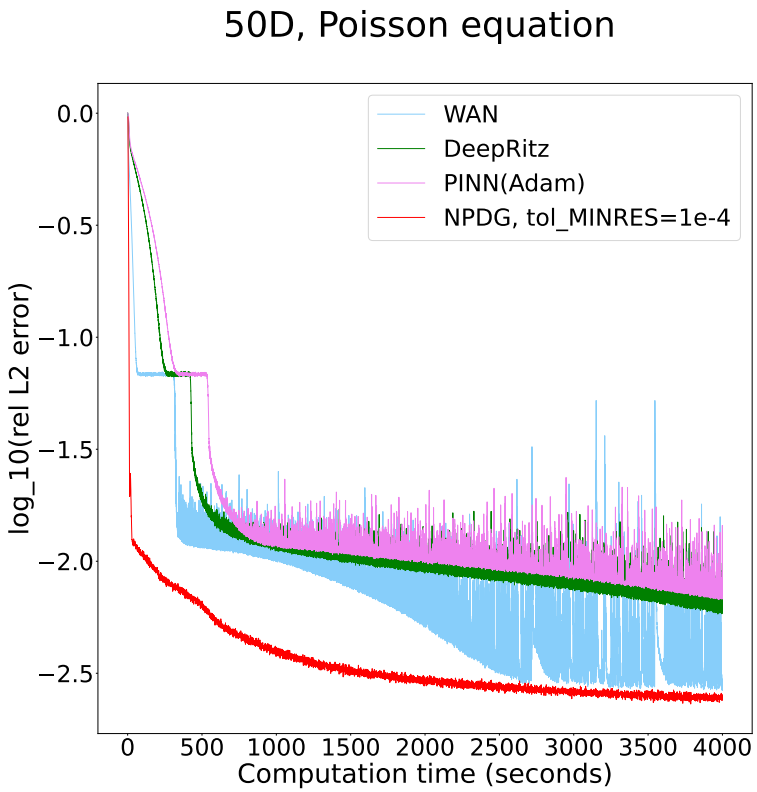


Performances of existing methods

Relative L^2 error vs. wall clock time(s).

- Sampling: uniform sampler on Ω and $\partial\Omega$;
- NN architecture: Multilayer Perceptron (MLP);

Proposed algorithm:
Stability, Efficiency, Versatility.



Related works

Various preconditioned deep-learning PDE-solver

- Multigrid-augmented method: [Azulay, et al. 2022]
- Domain decomposition strategy: [Kopaničáková]
- Incomplete LU preconditioning: [Liu, et al. 2024]
- Gauss-Newton for variational problems: [Hao, et al. 2024]
- **Natural gradient** method for PINNs (Distilling the Hessian for the residual functional for preconditioning):
  - Direct forming and inverting the precondition matrix via least square problem: [Müller, et al. 2023];
  - Utilizing K-FAC approximation: [Dangel, et al. 2024].
- And many more...

Table of Contents

Novel scheme: Adversarial learning scheme + Natural gradient $\Longrightarrow$ **Natural Primal-Dual Gradient (NPDG)**

Let us derive the algorithm for the Poisson equation:

$$-\Delta u = f, \text{ on } \Omega, \quad u = g, \text{ on } \partial\Omega. \tag{2}$$

- Denote the functional

$$\mathcal{F} : H^2(\Omega) \to L^2(\Omega) \times L^2(\partial\Omega),\ u \mapsto (-\Delta u - f, \mathcal{B}u - g),$$

  where $\mathcal{B}$ denotes the trace operator.
- It suffices to solve the root-finding problem $\mathcal{F}(u) = 0$.
- This can be reformulated as:

$$\inf_{u \in H^2(\Omega)} \frac{1}{2}\|\mathcal{F}(u)\|^2_{\mathbb{L}^2} := \frac{1}{2}(\| - \Delta u - f\|^2_{L^2(\Omega)} + \|\mathcal{B}u - g\|^2_{L^2(\partial\Omega)}).$$

$\mathbb{L}^2 := L^2(\Omega) \times L^2(\partial\Omega)$

A saddle point scheme

- Recall Legendre transform: $\frac{1}{2}\|\cdot\|_{\mathbb{L}^2}^2 = \sup_{v\in\mathbb{L}^2} \langle\cdot, v\rangle_{\mathbb{L}^2} - \frac{1}{2}\|v\|_{\mathbb{L}^2}^2$.
- We formulate

$$\inf_{u\in H^2(\Omega)} \sup_{\substack{\varphi\in L^2(\Omega),\\ \psi\in L^2(\partial\Omega)}} \mathscr{E}_0(u,\varphi,\psi) := \langle \mathcal{F}(u), (\varphi,\psi)\rangle_{\mathbb{L}^2} - \frac{1}{2}\|(\varphi,\psi)\|_{\mathbb{L}^2}^2$$

$$= \langle -\Delta u - f, \varphi\rangle_{L^2(\Omega)} + \langle u - g, \psi\rangle_{L^2(\partial\Omega)} - \frac{1}{2}\|(\varphi,\psi)\|_{\mathbb{L}^2}^2.$$

- $H_0^1(\Omega)$ is dense in $L^2(\Omega)$, fix $\varphi \in H_0^1(\Omega)$,

$$\inf_{u\in H^2(\Omega)} \sup_{\substack{\varphi\in H_0^1(\Omega),\\ \psi\in L^2(\partial\Omega)}} \mathscr{E}_0(u,\varphi,\psi) := \int_\Omega \nabla u\cdot\nabla\varphi - f\varphi dx + \int_{\partial\Omega}(u-g)\psi d\sigma$$

$$- \frac{1}{2}\left(\int_\Omega \varphi^2 dx + \int_{\partial\Omega}\psi^2 d\sigma\right).$$

Leveraging Primal-Dual Hybrid Gradient (PDHG)[1] algorithm

$$\begin{bmatrix} \varphi_{n+1} \\ \psi_{n+1} \end{bmatrix} = \underset{\substack{\varphi\in H_0^1(\Omega) \\ \psi\in L^2(\partial\Omega)}}{\operatorname{argmin}} \left\{ \frac{1}{2\tau_\varphi}\left(\|\varphi-\varphi_n\|^2_{L^2(\Omega)} + \|\psi-\psi_n\|^2_{L^2(\partial\Omega)}\right) - \mathscr{E}_0(u_n,\varphi,\psi) \right\},$$

$$\begin{bmatrix} \widetilde{\varphi}_{n+1} \\ \widetilde{\psi}_{n+1} \end{bmatrix} = \begin{bmatrix} \varphi_{n+1} \\ \psi_{n+1} \end{bmatrix} + \omega \left(\begin{bmatrix} \varphi_{n+1} \\ \psi_{n+1} \end{bmatrix} - \begin{bmatrix} \varphi_{n} \\ \psi_{n} \end{bmatrix} \right),$$

$$u_{n+1} = \underset{u\in H^2(\Omega)}{\operatorname{argmin}} \left\{ \frac{1}{2\tau_u}\left(\|u-u_n\|^2_{L^2(\Omega)} + \|\mathcal{B}u-\mathcal{B}u_n\|^2_{L^2(\partial\Omega)}\right) + \mathscr{E}_0(u,\widetilde{\varphi}_{n+1},\widetilde{\psi}_{n+1}) \right\}.$$

- Proximal Grad Ascent + Extrapolation + Proximal Grad Descent.
- τ_φ, τ_u: stepsizes, ω: extrapolation coefficient.
- Successful applications on conservation laws, reaction-diffusion equations[3], etc. (1D, 2D)

---

[1] M. Zhu, T. Chan. An efficient primal-dual hybrid gradient algorithm for total variation image restoration. UCLA CAM Report, 2008.
A. Chambolle, T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. JMIV, 2011.

[3] Shu Liu, Siting Liu, Stanley Osher, Wuchen Li. A first-order computational algorithm for reaction-diffusion type equations via primal-dual hybrid gradient method. JCP, 2023.

Ill-conditioned problem and its preconditioning

Experience gained from classical methods:

- Solve Poisson equation on $\Omega = [0, 1]$ with finite difference scheme. Suppose Ω is discretized into $N_x + 1$ subintervals.
- Denote $A = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} \in \mathbb{R}^{N_x \times N_x}$ as discrete Laplace.
- Root-finding problem $Au - b = 0$.
- Assoc. saddle point problem: $\min_u \max_p \left\{ p^\top (Au - b) - \frac{1}{2}\|p\|^2 \right\}$.
- PDHG algorithm yields the convergence rate $\sqrt{1 - \frac{c}{\kappa(A)^2}} = 1 - \mathcal{O}(\frac{1}{N_x^4})$ ($\kappa(A) = \mathcal{O}(N_x^2)$). **Very slow convergence!**
- **Remedy**: Find an easy-to-invert precondition matrix $M \approx A$ and consider $M^{-1}Au - M^{-1}b = 0$.

Introduce preconditioning

- PDHG algorithm suffers from the ill-conditioning of $\mathcal{F}(u)$.
- How do we apply the precondition operator, say, Δ^{-1} to $\mathcal{F}(u)$? How to realize the computation? 🤔
  Seems intractable.
- 💡 Preconditioning on metric terms of the proximal steps:

$$\|u - u_n\|^2_{L^2(\Omega)} \quad \text{replaced by} \quad \|\mathcal{M}_u u - \mathcal{M}_u u_n\|^2_{L^2(\Omega)}.$$

$$\|\varphi - \varphi_n\|^2_{L^2(\Omega)} \quad \text{replaced by} \quad \|\mathcal{M}_\varphi \varphi - \mathcal{M}_\varphi \varphi_n\|^2_{L^2(\Omega)}.$$

  $\mathcal{M}_u$, $\mathcal{M}_\varphi$ are precondition operators for u, φ.
- How to pick $\mathcal{M}_u$, $\mathcal{M}_\varphi$?

Key idea:

- Recall u_* is the solution to (2), and $-\Delta u_* = f$,

$$\begin{aligned}\langle \mathcal{F}(u), (\varphi, \psi)\rangle_{\mathbb{L}^2} &= \int_\Omega (-\Delta u - f)\varphi \, dx + \int_{\partial\Omega} (u - g)\psi d\sigma \\ &= \int_\Omega (\nabla u - \nabla u_*)\cdot \nabla\varphi \, dx + \int_{\partial\Omega} (\mathcal{B}u - \mathcal{B}u_*)\psi d\sigma. \\ &= \Big\langle \underbrace{\begin{pmatrix} \nabla u \\ \mathcal{B}u \end{pmatrix}}_{\mathbf{U}} - \underbrace{\begin{pmatrix} \nabla u_* \\ \mathcal{B}u_* \end{pmatrix}}_{\mathbf{U}_*}, \underbrace{\begin{pmatrix} \nabla\varphi \\ \psi \end{pmatrix}}_{\mathbf{\Phi}} \Big\rangle_{\mathbb{L}^2}\end{aligned}$$

- We modify $\mathscr{E}_0(u, \varphi, \psi)$ as:

$$\begin{aligned}\mathscr{E}(u, \varphi, \psi) :=& \langle \mathcal{F}(u), (\varphi, \psi)\rangle_{\mathbb{L}^2} - \frac{1}{2}\left(\int_\Omega \|\nabla\varphi\|^2 dx + \int_{\partial\Omega} \psi^2 d\sigma\right) \\ =& \langle \mathbf{U} - \mathbf{U}_*, \mathbf{\Phi}\rangle_{\mathbb{L}^2} - \frac{1}{2}\|\mathbf{\Phi}\|^2_{\mathbb{L}^2}.\end{aligned}$$

- $\|\mathbf{U} - \mathbf{U}_n\|^2_{\mathbb{L}^2} = \|\nabla u - \nabla u_n\|^2_{L^2(\Omega)} + \|\mathcal{B}u - \mathcal{B}u_n\|^2_{L^2(\partial\Omega)}$.
- This motivates us to pick $\mathcal{M}_u = \nabla$, and similarly, $\mathcal{M}_\varphi = \nabla$.

PDHG with preconditioning

The preconditioned PDHG for solving (2) yields

$$\begin{bmatrix} \varphi_{n+1} \\ \psi_{n+1} \end{bmatrix} = \underset{\substack{\varphi \in H_0^1(\Omega) \\ \psi \in L^2(\partial\Omega)}}{\operatorname{argmin}} \left\{ \frac{1}{2\tau_\varphi} (\|\nabla\varphi - \nabla\varphi_n\|_{L^2(\Omega)}^2 + \|\psi - \psi_n\|_{L^2(\partial\Omega)}^2) - \mathscr{E}(u_n, \varphi, \psi) \right\},$$

$$\begin{bmatrix} \widetilde{\varphi}_{n+1} \\ \widetilde{\psi}_{n+1} \end{bmatrix} = \begin{bmatrix} \varphi_{n+1} \\ \psi_{n+1} \end{bmatrix} + \omega \left(\begin{bmatrix} \varphi_{n+1} \\ \psi_{n+1} \end{bmatrix} - \begin{bmatrix} \varphi_n \\ \psi_n \end{bmatrix} \right),$$

$$u_{n+1} = \underset{u \in H^2(\Omega)}{\operatorname{argmin}} \left\{ \frac{1}{2\tau_u} (\|\nabla u - \nabla u_n\|_{L^2(\Omega)}^2 + \|\mathcal{B}u - \mathcal{B}u_n\|_{L^2(\partial\Omega)}^2) + \mathscr{E}(u, \widetilde{\varphi}_{n+1}, \widetilde{\psi}_{n+1}) \right\}.$$

- The algorithm is at functional level.
- Set $u = u_\theta$, $\varphi = \varphi_\eta$, $\psi = \psi_\xi$ as neural networks.
- Instead of updating u, φ, ψ, we update parameters $\theta \in \mathbb{R}^{m_\theta}, \eta \in \mathbb{R}^{m_\eta}, \xi \in \mathbb{R}^{m_\xi}$.

PDHG with preconditioning

Preconditioned PDHG for neural networks

$$\begin{bmatrix} \eta^{n+1} \\ \xi^{n+1} \end{bmatrix} = \underset{\substack{\eta\in\Theta_\eta \\ \xi\in\Theta_\xi}}{\operatorname{argmin}} \left\{ \frac{1}{2\tau_\varphi}(\|\nabla\varphi_\eta - \nabla\varphi_{\eta^n}\|^2_{L^2(\Omega)} + \|\psi_\xi - \psi_{\xi^n}\|^2_{L^2(\partial\Omega)}) - \mathscr{E}(u_{\theta^n}, \varphi_\eta, \psi_\xi) \right\},$$

$$\begin{bmatrix} \widetilde{\varphi}^{n+1} \\ \widetilde{\psi}^{n+1} \end{bmatrix} = \begin{bmatrix} \varphi_{\eta^{n+1}} \\ \psi_{\xi^{n+1}} \end{bmatrix} + \omega \left(\begin{bmatrix} \varphi_{\eta^{n+1}} \\ \psi_{\xi^{n+1}} \end{bmatrix} - \begin{bmatrix} \varphi_{\eta^n} \\ \psi_{\xi^n} \end{bmatrix} \right),$$

$$\theta^{n+1} = \underset{\theta\in\Theta_\theta}{\operatorname{argmin}} \left\{ \frac{1}{\tau_u}(\|\nabla u_\theta - \nabla u_{\theta^n}\|^2_{L^2(\Omega)} + \|\mathcal{B}u_\theta - \mathcal{B}u_{\theta^n}\|^2_{L^2(\partial\Omega)}) + \mathscr{E}(u_\theta, \widetilde{\varphi}^{n+1}, \widetilde{\psi}^{n+1}) \right\}.$$

- The algorithm is at functional level.
- Set $u = u_\theta$, $\varphi = \varphi_\eta$, $\psi = \psi_\xi$ as neural networks.
- Instead of updating u, φ, ψ, we update parameters $\theta \in \mathbb{R}^{m_\theta}, \eta \in \mathbb{R}^{m_\eta}, \xi \in \mathbb{R}^{m_\xi}$.

PDHG with preconditioning

Preconditioned PDHG for neural networks

$$\begin{bmatrix} \eta^{n+1} \\ \xi^{n+1} \end{bmatrix} = \underset{\eta,\xi}{\operatorname{argmin}} \left\{ \frac{1}{2\tau_\varphi} \Big(\overbrace{\|\nabla\varphi_\eta - \nabla\varphi_{\eta^n}\|^2_{L^2(\Omega)} + \|\psi_\xi - \psi_{\xi^n}\|^2_{L^2(\partial\Omega)}}^{d_2^2((\eta,\xi),(\eta^n,\xi^n))} \Big) - \mathscr{E}(u_{\theta^n}, \varphi_\eta, \psi_\xi) \right\},$$

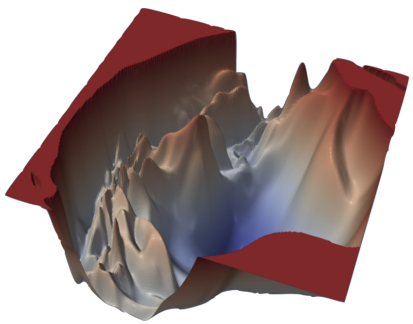
$$\begin{bmatrix} \widetilde{\varphi}^{n+1} \\ \widetilde{\psi}^{n+1} \end{bmatrix} = \begin{bmatrix} \varphi_{\eta^{n+1}} \\ \psi_{\xi^{n+1}} \end{bmatrix} + \omega \left(\begin{bmatrix} \varphi_{\eta^{n+1}} \\ \psi_{\xi^{n+1}} \end{bmatrix} - \begin{bmatrix} \varphi_{\eta^n} \\ \psi_{\xi^n} \end{bmatrix} \right),$$

$$\theta^{n+1} = \underset{\theta}{\operatorname{argmin}} \left\{ \frac{1}{2\tau_u} \Big(\underbrace{\|\nabla u_\theta - \nabla u_{\theta^n}\|^2_{L^2(\Omega)} + \|\mathcal{B}u_\theta - \mathcal{B}u_{\theta^n}\|^2_{L^2(\partial\Omega)}}_{d_1^2(\theta,\theta^n)} \Big) + \mathscr{E}(u_\theta, \widetilde{\varphi}^{n+1}, \widetilde{\psi}^{n+1}) \right\}.$$

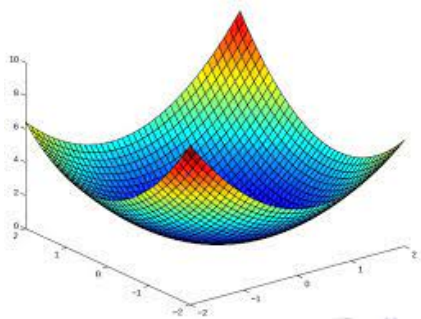
- The algorithm is at functional level.
- Set $u = u_\theta$, $\varphi = \varphi_\eta$, $\psi = \psi_\xi$ as neural networks.
- Instead of updating u, φ, ψ, we update parameters $\theta \in \mathbb{R}^{m_\theta}, \eta \in \mathbb{R}^{m_\eta}, \xi \in \mathbb{R}^{m_\xi}$.

Key idea:

$$-\mathscr{E} = -\mathscr{E}(u_\theta, \varphi_\eta, \psi_\xi) \qquad -\mathscr{E} = -\langle \mathbf{U}_\theta - \mathbf{U}_*, \mathbf{\Phi}_{\eta,\xi} \rangle_{\mathbb{L}^2} + \frac{1}{2}\|\mathbf{\Phi}_{\eta,\xi}\|^2_{\mathbb{L}^2}$$



(η, ξ)'s point of view[4].



$\mathbf{\Phi}_{\eta,\xi}$'s point of view.

The intrinsic distance between (η, ξ) and (η^n, ξ^n) should be induced by

$$d_2^2((\eta, \xi), (\eta^n, \xi^n)) = \|\mathbf{\Phi}_{\eta,\xi} - \mathbf{\Phi}_{\eta^n,\xi^n}\|^2_{\mathbb{L}^2},$$

rather than ℓ^2 distance $\|(\eta, \xi) - (\eta^n, \xi^n)\|_2$.

[4] H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein. Visualizing the Loss Landscape of Neural Nets. Advances in neural information processing systems, 2018.

Natural Gradient Descent

- The update of θ^{n+1}:

$$\theta^{n+1} = \underset{\theta \in \mathbb{R}^{m_\theta}}{\operatorname{argmin}} \left\{ \frac{1}{2\tau_u} \left(d_1^2(\theta, \theta^n) + \mathscr{E}(u_\theta, \widetilde{\varphi}_{n+1}, \widetilde{\psi}_{n+1}) \right) \right\}.$$

- The *time-explicit* scheme:

$$\theta^{n+1} = \theta^n - \tau_u \underbrace{M_p(\theta^n)^\dagger \nabla_\theta \mathscr{E}(u_{\theta^n}, \widetilde{\varphi}_{n+1}, \widetilde{\psi}_{n+1})}_{\text{Nature Gradient}},$$

$M_p(\theta^n)$ is induced by:

$$d_1^2(\theta, \theta^n) = \|\nabla u_\theta - \nabla u_{\theta^n}\|_{L^2(\Omega)}^2 + \|\mathcal{B} u_\theta - \mathcal{B} u_{\theta^n}\|_{L^2(\partial\Omega)}^2 \approx (\theta - \theta^n)^\top M_p(\theta^n)(\theta - \theta^n).$$

$$(M_p(\theta))_{ij} = \int_\Omega \frac{\partial}{\partial \theta_i} \nabla u_\theta(x) \cdot \frac{\partial}{\partial \theta_j} \nabla u_\theta(x) dx + \int_{\partial\Omega} \frac{\partial u_\theta(y)}{\partial \theta_i} \frac{\partial u_\theta(y)}{\partial \theta_j} d\sigma.$$

NPDG algorithm

We update ξ^{n+1} and θ^{n+1} by similar techniques. This yields the **Natural Primal-Dual Gradient** (NPDG) algorithm.

$$\begin{bmatrix} \eta^{n+1} \\ \xi^{n+1} \end{bmatrix} = \begin{bmatrix} \eta^{n} \\ \xi^{n} \end{bmatrix} + \tau_\varphi \begin{bmatrix} M_d(\eta^n)^\dagger \nabla_\eta \mathscr{E}(u_{\theta^n}, \varphi_{\eta^n}, \psi_{\xi^n}) \\ M_{bdd}(\xi^n)^\dagger \nabla_\xi \mathscr{E}(u_{\theta^n}, \varphi_{\eta^n}, \psi_{\xi^n}) \end{bmatrix},$$

$$\begin{bmatrix} \widetilde{\varphi}_{n+1} \\ \widetilde{\psi}_{n+1} \end{bmatrix} = \begin{bmatrix} \varphi_{\eta^{n+1}} \\ \psi_{\xi^{n+1}} \end{bmatrix} + \omega \left(\begin{bmatrix} \varphi_{\eta^{n+1}} \\ \psi_{\xi^{n+1}} \end{bmatrix} - \begin{bmatrix} \varphi_{\eta^{n}} \\ \psi_{\xi^{n}} \end{bmatrix} \right),$$

$$\theta^{n+1} = \theta^n - \tau_u M_p(\theta^n)^\dagger \nabla_\theta \mathscr{E}(u_{\theta^n}, \widetilde{\varphi}_{n+1}, \widetilde{\psi}_{n+1}).$$

- More general cases:

$$\mathcal{L}u + \mathcal{N}u = f \text{ on } \Omega, \quad \mathcal{B}u = g \text{ on } \partial\Omega,$$

  in which linear operator $\mathcal{L}$ splits as $\mathcal{D}_d{}^* \widetilde{\mathcal{L}} \mathcal{D}_p$. We pick $\mathcal{M}_u = \mathcal{D}_p$, $\mathcal{M}_\varphi = \mathcal{D}_d$ and apply the NPDG algorithm.

- Monge-Ampère equation arising from L^2 Optimal Transport problem.

Implementation

- Computation of $\nabla_\eta \mathscr{E}(u_{\theta^n}, \varphi_{\eta^n}, \psi_{\xi^n})$:
  - Monte-Carlo Approximation + Auto-Differentiation
- Evaluation of Natural Gradient $M_d(\eta^n)^\dagger \nabla_\eta \mathscr{E}(u_{\theta^n}, \varphi_{\eta^n}, \psi_{\xi^n})$:
  - ⚠️ $M_d(\eta^n)$ is prohibitively **expensive** to compute.
  - Krylov subspace iteration: **MINRES** (Minimal residual) method: Only requires matrix-vector multiplication.
  - Matrix-vector multiplication $M_d(\eta^n)\mathbf{v}$ can be efficiently computed using auto-differentiations.

Table of Contents

Time-continuous NPDG algorithm

- **NPDG flow**: time-continuous version of the NPDG algorithm:

$$\begin{aligned}\dot{\eta}_t &= M_d(\eta_t)^\dagger \nabla_\eta \mathscr{E}(u_{\theta_t}; \varphi_{\eta_t}, \psi_{\xi_t}),\\ \dot{\xi}_t &= M_{bdd}(\xi_t)^\dagger \nabla_\xi \mathscr{E}(u_{\theta_t}, \varphi_{\eta_t}, \psi_{\xi_t}),\\ \dot{\theta}_t &= -M_p(\theta_t)^\dagger \nabla_\theta \mathscr{E}(u_{\theta_t}; \varphi_t + \gamma\dot{\varphi}_t, \psi_t + \gamma\dot{\psi}_t),\end{aligned}$$

  as $\tau_u, \tau_\varphi \to 0$ and $\omega\tau_\varphi \to \gamma > 0$.

- Consider the linear equation:

$$\mathcal{L}u = f, \text{ on } \Omega, \quad \mathcal{B}u = g, \text{ on } \partial\Omega, \tag{3}$$

  where $\mathcal{L} = \mathcal{D}_d{}^* \widetilde{\mathcal{L}} \mathcal{D}_p$. Suppose (3) admits unique classical solution u_*.

- Apply NPDG flow with $\mathcal{M}_\varphi = \mathcal{D}_d, \mathcal{M}_u = \mathcal{D}_p$ to (3).

A posteriori convergence analysis of NPDG flow[5]

- Suppose $\{(\theta_t, \eta_t, \xi_t)\}_{0\le t\le T}$ solves the NPDG-flow.
- Denote $\widetilde{\kappa}$ as the condition number of $\widetilde{\mathcal{L}}$ w.r.t. L^2 norm.
- Assume α, β_1, β_2 describe the approximation quality of tangent subspaces spanned by neural networks $u_\theta, \varphi_\eta, \psi_\xi$.

Suppose $\alpha + \beta_1 < \frac{1}{\widetilde{\kappa}^2}$, $\beta_2 < 1$, if γ further satisfies

$$\left(\frac{1}{\widetilde{\kappa}^2} - (\alpha+\beta_1)\right)\cdot(1-\beta_2) > \frac{((1+\beta_1)\gamma + \beta_2 + \alpha|1-\gamma|)^2}{4\gamma}.$$

There exists $r > 0$ depending on $\widetilde{\kappa}$, such that, for $0 \le t \le T$,

$$\|\mathcal{M}_u(u_{\theta_t} - u_*)\|^2_{L^2(\Omega)} + \|\mathcal{B}(u_{\theta_t} - u_*)\|^2_{L^2(\partial\Omega)} \le 2C_0 \exp(-rt).$$

[5] S. Liu, S. Osher, W. Li. A Natural Primal-Dual Hybrid Gradient Method for Adversarial Neural Network Training on Solving Partial Differential Equations. arXiv:2411.06278, 2024.

Table of Contents

Elliptic equation with variable coefficients $d = 20$

- We consider:

$$-\nabla \cdot (\kappa(x)\nabla u(x)) = f(x), \quad u(y) = g(y) \text{ on } \partial\Omega.$$

  - $\Omega = [-1, 1]^d$ with even dimension d.
  - $\kappa(x) = \frac{x^\top \Lambda x + 1}{2}$, with $\Lambda = \mathrm{diag}(\lambda_0, \lambda_1, \ldots, \lambda_0, \lambda_1)$, $\lambda_0 = 1, \lambda_1 = 4$.
  - $f(x) = -\frac{\mathrm{Tr}(\Lambda^{-1})}{2}(x^\top \Lambda x + 1) - \|x\|^2$, $g(y) = \frac{1}{2} y^\top \Lambda^{-1} y$.
  - The solution $u_*(x) = \frac{1}{2} x^\top \Lambda^{-1} x$.

- Pick u, φ, ψ as MLPs with hidden dimension 256, number of hidden layers 4, and $\mathrm{softplus}(\cdot)$[6] activation function.

- $\mathcal{M}_p = \nabla$, $\mathcal{M}_d = \nabla$.

[6] $\mathrm{softplus}(x) = \frac{1}{\beta} \log(1 + \exp(\beta x))$, $\beta = \frac{1}{4}$.

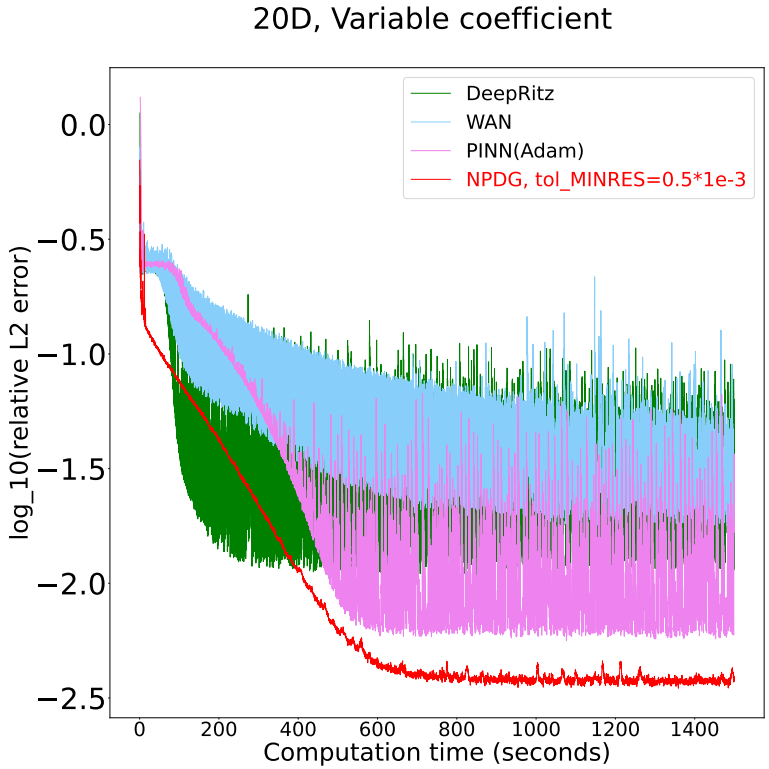


Figure: Plots of relative L^2 error vs. wall clock time(s), comparing NPDG with PINN, DeepRitz, and WAN.

- **Fix NN architecture** as MLP and use the **same sample size** for all tested methods.

Performance comparison on different types of PDEs

- Fix NN architecture as MLP. Use the same sample size.
- We test PINN, DeepRitz, and WAN using different learning rates and present the *optimal* performance.

Equation	δ	d	PINN(Adam)	Deep Ritz	WAN	NPDG
Poisson $-\Delta u = f$	0.005	10D	44.83	43.45	51.65	**40.98**
		20D	160.82	183.49	460.12	**110.42**
		50D	1989.06	1452.29	2117.24	**821.24**
VarCoeff $-\nabla \cdot (\kappa \nabla u) = f$	0.005	10D	–	–	–	**281.26**
		20D	–	–	–	**998.09**
		50D	–	–	–	**13731.33**
Nonlinear Elliptic $\frac{1}{2}\|\nabla u\|^2 + V = \Delta u$	0.05	5D	–	N.A.	–	**1894.89**

Figure: GPU time (s) required to reach relative L^2 accuracy δ when different methods are applied to PDEs with different dimensions posed with Dirichlet boundary condition. ("–" the accuracy is never achieved; "N.A." method is not applicable.)

Monge-Ampère equation arising from L^2-OT problem

- ρ_0, ρ_1 – probability density functions.
- Monge-Ampère equation:

$$|\det(D^2 u(x))| = \frac{\rho_0(x)}{\rho_1(\nabla u(x))}, \ \rho_0 dx - a.e., \quad u \text{ is convex on } \mathbb{R}^d.$$

- L^2 Optimal Transport (OT) problem (Monge problem):

$$\min_{T:\mathbb{R}^d \to \mathbb{R}^d, T_\sharp \mu = \nu} \int_{\mathbb{R}^d} \| T(x) - x \|^2 d\mu. \quad \mu = \rho_0 dx, \ \nu = \rho_1 dx$$

Fact: OT map $T_* = \nabla u_*$, with u_* solves the Monge-Ampère equation.

To compute $T_* = \nabla u_*$, apply **NPDG** algorithm to the **primal-dual** problem assoc. with the L^2 OT problem.

OT from Gaussian to Gaussian mixture in $\mathbb{R}^{50}$

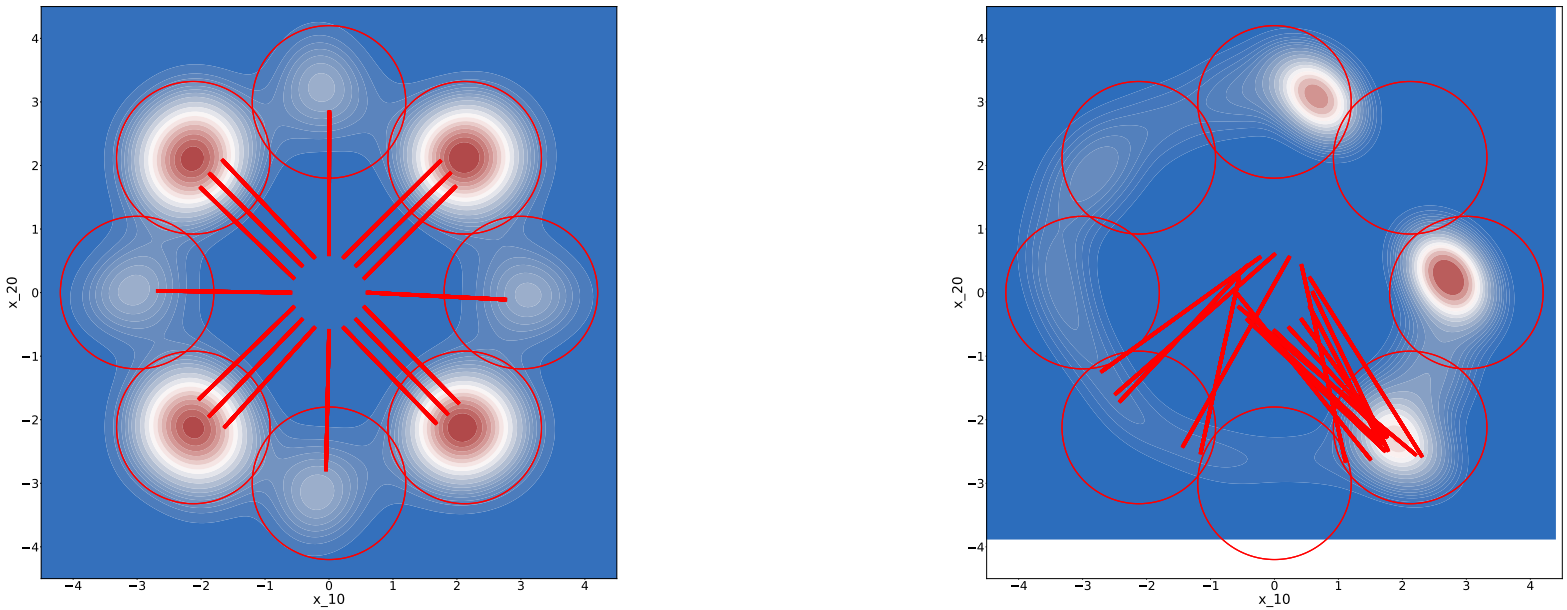


Figure: Plots of the pushforwarded density $T_{\theta\sharp}\rho_0$ by Kernel Density Estimation (KDE). **Left**: NPDG method; **Right**: Primal-Dual algorithm using Adam[7]. Red lines indicate the computed OT map $T_\theta(\cdot)$.

[7] Jiaojiao Fan*, Shu Liu*, Shaojun Ma, Haomin Zhou, Yongxin Chen. Neural Monge Map estimation and its applications. TMLR, 2023

Table of Contents

Summary of the research

NPDG = Adversarial training of NNs + Natural Gradients.

- Adversarial training:
  - Computational efficiency: lower order of differentiation.
  - Versatility: Applicable to PDEs possessing saddle point scheme.
- Natural Gradient:
  - Respect the mathematical nature: PDE-informed precondition.
  - Fast & Stable convergence.

On-going & future plans

Ongoing: improvements of NPDG algorithm:

- Jacobi-free method: Finite difference instead of auto-differentiation.
- Kronecker-Factored Approximate Curvature (KFAC) method for more efficient evaluation of natural gradient[8].

[8] Felix Dangel, Johannes Müller, and Marius Zeinhofer. Kronecker-Factored Approximate Curvature for Physics-Informed Neural Networks. NeurIPS, 2024.

NPDG: Efficient scientific machine learning (SciML) paradigm.

Large-scale saddle point/constrained optimization problems:

- Enhancing the training of Generative Adversarial Networks (GANs).
- Optimal Transport (Monge) problem with general cost[9].
- High-dimensional Mean-Field Control problems.

[9] Jiaojiao Fan*, Shu Liu*, Shaojun Ma, Haomin Zhou, Yongxin Chen. Neural Monge Map estimation and its applications. TMLR, 2023

Reference:

Shu Liu, Stanley Osher, Wuchen Li. A Natural Primal-Dual Hybrid Gradient Method for Adversarial Neural Network Training on Solving Partial Differential Equations. arXiv:2411.06278, 2024.

We welcome any comments and suggestions.

Thank you!