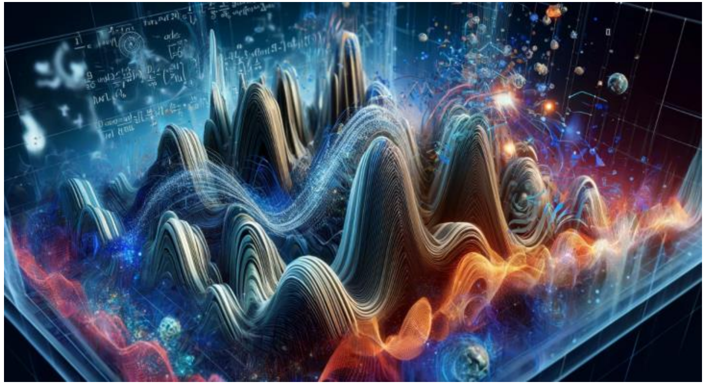
Solving High Frequency and Multi-Scale PDEs with Gaussian Processes

ICLR 2024

Shikai Fang*, Madison Cooley*, Da Long*,
Shibo Li, Robert M. Kirby, Shandian Zhe
∗Presenter: Shikai Fang



Github: github.com/xuangu-fang/Gaussian-Process-Slover-for-High-Freq-PDE

THE UNIVERSITY OF UTAH®

ML-based PDE Solver

- General form of PDE

Differential opeartor

Target solution

$$\boxed{\mathcal{F}}\underbrace{[u](\mathbf{x}) = f(\mathbf{x})}_{\text{Equations in domain}} \quad (\mathbf{x} \in \Omega), \quad \underbrace{\boxed{u}(\mathbf{x}) = g(\mathbf{x})}_{\text{Boundary conditions}} \quad (\mathbf{x} \in \partial\Omega),$$

- ML slovers of PINN[1] family:
  - Parameterized model (DNN) as the approx. of u: $\widehat{u}_{\boldsymbol{\theta}}(\mathbf{x}) \approx u_{\boldsymbol{\theta}}(\mathbf{x})$
  - Canonical objective func. : $\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} \; L_b(\boldsymbol{\theta}) + L_r(\boldsymbol{\theta}),$

Boundary term

Residual term

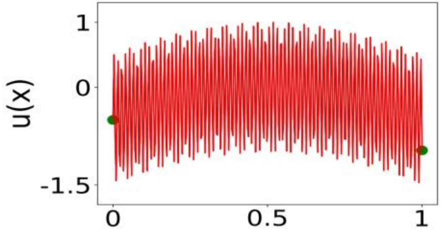
where $L_b(\boldsymbol{\theta}) = \frac{1}{N_b}\sum_{j=1}^{N_b}\left(\widehat{u}_{\boldsymbol{\theta}}(\mathbf{x}_b^j) - g(\mathbf{x}_b^j)\right)^2$

$L_r(\boldsymbol{\theta}) = \frac{1}{N_c}\sum_{j=1}^{N_c}\left(\mathcal{F}[\widehat{u}_{\boldsymbol{\theta}}](\mathbf{x}_c^j) - f(\mathbf{x}_c^j)\right)^2$

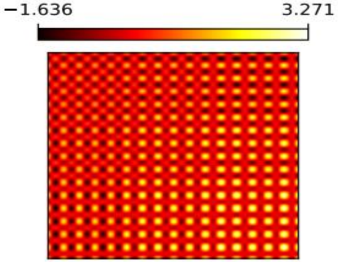
[1] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.

Hard cases: high-freq. + multi-scale PDEs

- Examples:



$\sin(500x) - 2(x - 0.5)^2$



$(\sin(x) + 0.1\sin(20x) + \cos(100x))\,(\sin(y) + 0.1\sin(20y) + \cos(100y))$

- Current NN-based ML solvers hard to handle such cases, because:
  - “Specturm bias”[1] in NN training
  - Easy to capture low-freq. info, hard to capture high-freq.

[1] Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A.(2019). On the spectral bias of neural networks. ICML

Motivation of GP-HM(our work)

Goals:

- Model the PDE solution in the frequency domain
- Estimate the target frequencies from covariance function

Kernel Learning &
Wiener-Khinchin Theorm

- Apply Gaussian Processes(GPs) with proper kernels as an alternative ML solver

$$\left\{\begin{array}{l} u(\cdot) \sim \mathcal{GP}(m(\cdot), \operatorname{cov}(\cdot, \cdot)) \\ \operatorname{cov}\left(\partial_{x_1 x_2} u(\mathbf{x}), u\left(\mathbf{x}'\right)\right) = \partial_{x_1 x_2} k\left(\mathbf{x}, \mathbf{x}'\right) \end{array}\right.$$

[1] Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A.(2019). On the spectral bias of neural networks. ICML\\

Model of GP-HM

- Model PDE solution’s **power spectrum** with a mixture of student-t (St) distribution

(distribution of function in frequency domain: norm of FT[u])

$$S(s) = \sum_{q=1}^{Q} \underbrace{w_q}_{\text{Non-negative component weight}} \underbrace{\mathrm{St}(s; \mu_q, \rho_q^2, \nu)}_{\text{One principle frequency } \mu_q},$$

- Alternative: a mixture of Gaussian distribution

$$S(s) = \sum_{q=1}^{Q} w_q \mathcal{N}(s; \mu_q, \rho_q^2)$$

[1] Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A.(2019). On the spectral bias of neural networks. ICML\\

From spectrum to covariance

- Apply Wiener-Khinchin theorem: transform spectrum to valid covariance function (kernel)

From the mixure of student-t:

$$k_{\text{StM}}(x,x') = \sum\nolimits_{q=1}^{Q} w_q \gamma_{\nu,\rho_q}(x,x') \cos(2\pi\mu_q(x-x')),$$

From the mixure of Gaussian: (known as *spectral mixture kernel*)

$$k_{\text{GM}}(x,x') = \sum\nolimits_{q=1}^{Q} w_q \exp\left(-\rho_q^2(x-x')^2\right) \cdot \cos(2\pi(x-x')\mu_q).$$

Objective and inference

- Parameters to learn:
  - Solution values at grid points: $\mathcal{U} = \{u(\mathbf{x}) | \mathbf{x} \in \mathcal{G}\}$, which is an $M_1 \times \ldots \times M_d$ array.
  - Kernel parameters (freq., weights…): Θ,
  - Observation noises (in domain & boundary): τ_1 and τ_2

- Inference: maximize log joint probability

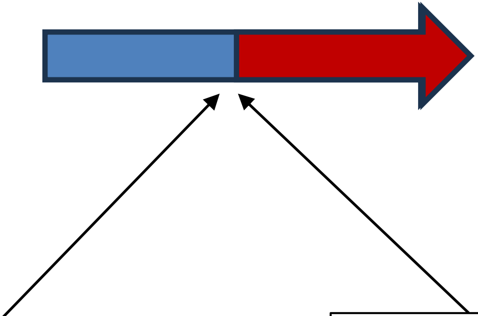
$$\mathcal{L}(\mathcal{U}, \Theta, \tau_1, \tau_2) = \underbrace{\log \mathcal{N}(\text{vec}(\mathcal{U})|\mathbf{0}, \mathbf{C})}_{\substack{\text{GP priors of the solution} \\ \text{(compute from the kernel)}}} + \underbrace{\lambda_b \cdot \log \mathcal{N}(\mathbf{g}|\mathbf{u}_b, \tau_1^{-1}\mathbf{I})}_{\substack{\text{Likelihood of the boundary} \\ \text{conditions}}} + \underbrace{\log \mathcal{N}(\mathbf{0}|\text{vec}(\mathcal{H}), \tau_2^{-1}\mathbf{I})}_{\substack{\text{Likelihood on the differential} \\ \text{terms in domain}}}$$

Recap: loss func. of PINN : $\boldsymbol{\theta}^* = \text{argmin}_{\boldsymbol{\theta}} \;\; L_b(\boldsymbol{\theta}) + L_r(\boldsymbol{\theta}),$

Structured kernel for efficient computation

For grids with resolution $M_1 \times \ldots \times M_d$, we will have $\boxed{M = \prod M_d}$ allocation points

Original GP cost:
- Time: $\mathcal{O}(M^3)$
- Space: $\mathcal{O}(M^2)$



GP-HM cost:
- Time: $\mathcal{O}\left(\sum M_d^3 + \left(\sum M_d\right) M\right)$
- Space: $\mathcal{O}\left(\sum M_d^2 + M\right)$

- Product kernel and cross covariance:

$$\mathrm{cov}(f(\mathbf{x}), f(\mathbf{x}')) = \kappa(\mathbf{x}, \mathbf{x}'|\Theta) = \prod\nolimits_{j=1}^{d} k_{\mathrm{StM}}(x_j, x_j'|\boldsymbol{\theta}_q),$$

$$\mathrm{cov}(\partial_{x_1 x_2} u(\mathbf{x}), u(\mathbf{x}'))$$

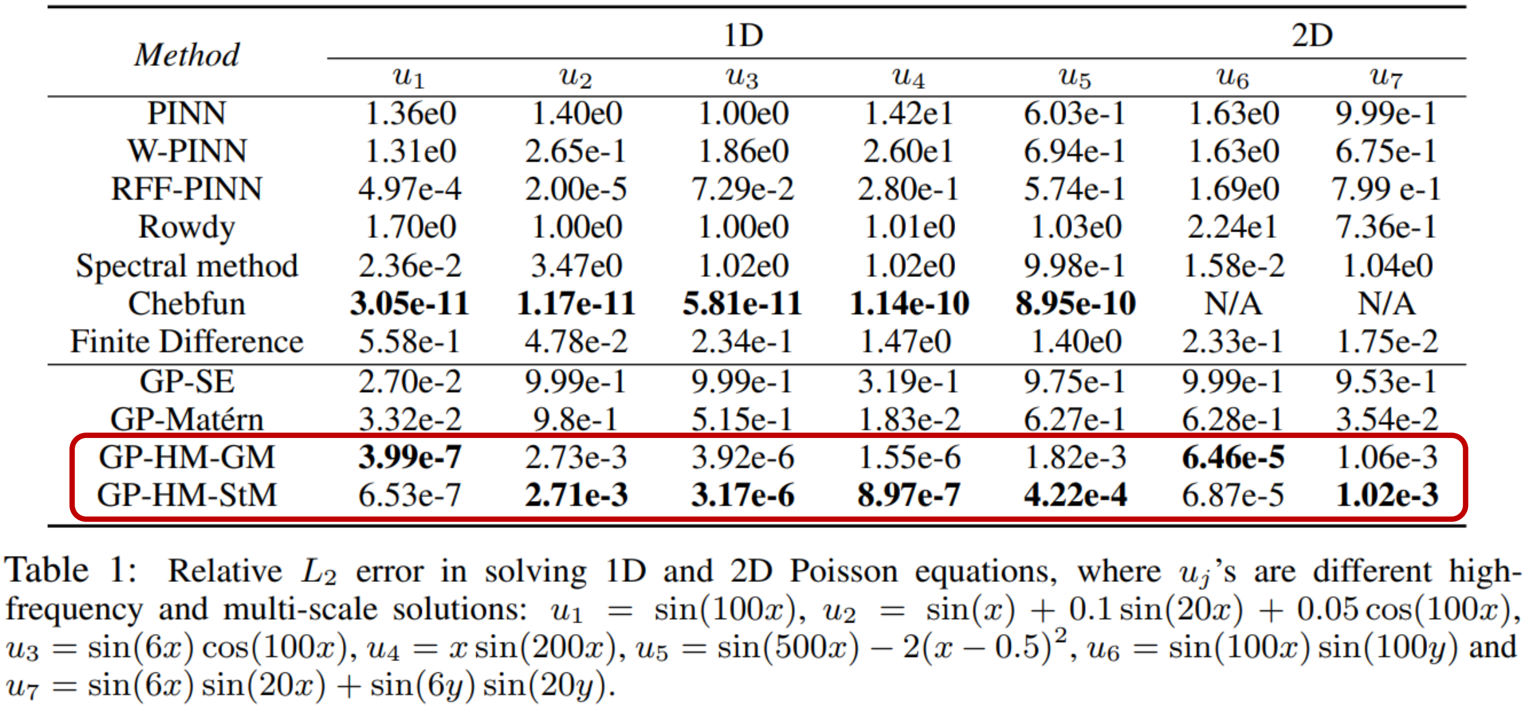
$$= \partial_{x_1} \kappa(x_1, x_1') \cdot \partial_{x_2} \kappa(x_2, x_2') \cdot \prod\nolimits_{j \neq 1,2} \kappa(x_j, x_j').$$

- Kronecker product structure of kernel matrix

$$\log |\mathbf{C}| = \sum\nolimits_{j=1}^{d} \frac{M}{M_j} \log |\mathbf{C}_j|,$$

$$\mathbf{C}^{-1}\mathrm{vec}(\mathcal{U}) = \mathrm{vec}\left(\mathcal{U} \times_1 \mathbf{C}_1^{-1} \times_2 \ldots \times_d \mathbf{C}_d^{-1}\right)$$

Numerical results



Method	1D					2D	
	u_1	u_2	u_3	u_4	u_5	u_6	u_7
PINN	1.36e0	1.40e0	1.00e0	1.42e1	6.03e-1	1.63e0	9.99e-1
W-PINN	1.31e0	2.65e-1	1.86e0	2.60e1	6.94e-1	1.63e0	6.75e-1
RFF-PINN	4.97e-4	2.00e-5	7.29e-2	2.80e-1	5.74e-1	1.69e0	7.99 e-1
Rowdy	1.70e0	1.00e0	1.00e0	1.01e0	1.03e0	2.24e1	7.36e-1
Spectral method	2.36e-2	3.47e0	1.02e0	1.02e0	9.98e-1	1.58e-2	1.04e0
Chebfun	**3.05e-11**	**1.17e-11**	**5.81e-11**	**1.14e-10**	**8.95e-10**	N/A	N/A
Finite Difference	5.58e-1	4.78e-2	2.34e-1	1.47e0	1.40e0	2.33e-1	1.75e-2
GP-SE	2.70e-2	9.99e-1	9.99e-1	3.19e-1	9.75e-1	9.99e-1	9.53e-1
GP-Matérn	3.32e-2	9.8e-1	5.15e-1	1.83e-2	6.27e-1	6.28e-1	3.54e-2
GP-HM-GM	**3.99e-7**	2.73e-3	3.92e-6	1.55e-6	1.82e-3	**6.46e-5**	1.06e-3
GP-HM-StM	6.53e-7	**2.71e-3**	**3.17e-6**	**8.97e-7**	**4.22e-4**	6.87e-5	**1.02e-3**

Table 1: Relative L_2 error in solving 1D and 2D Poisson equations, where u_j's are different high-frequency and multi-scale solutions: $u_1 = \sin(100x)$, $u_2 = \sin(x) + 0.1\sin(20x) + 0.05\cos(100x)$, $u_3 = \sin(6x)\cos(100x)$, $u_4 = x\sin(200x)$, $u_5 = \sin(500x) - 2(x-0.5)^2$, $u_6 = \sin(100x)\sin(100y)$ and $u_7 = \sin(6x)\sin(20x) + \sin(6y)\sin(20y)$.

Numerical results: visulization

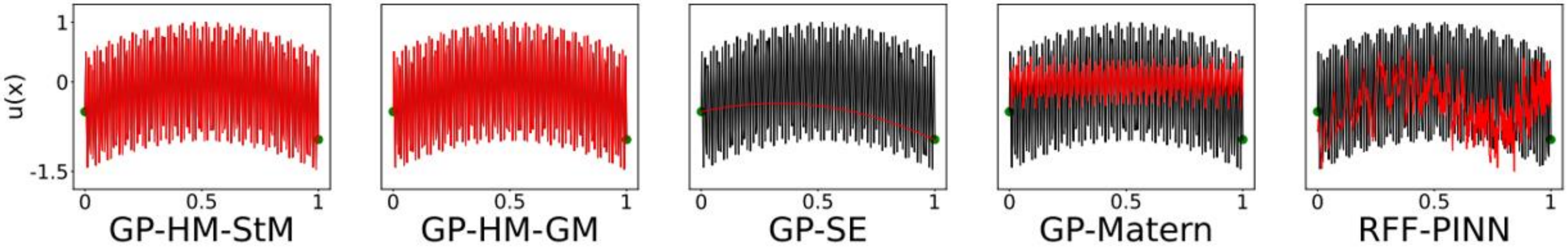


Figure 2: Prediction for the 1D Poisson equation with solution $\sin(500x) - 2(x-0.5)^2$.

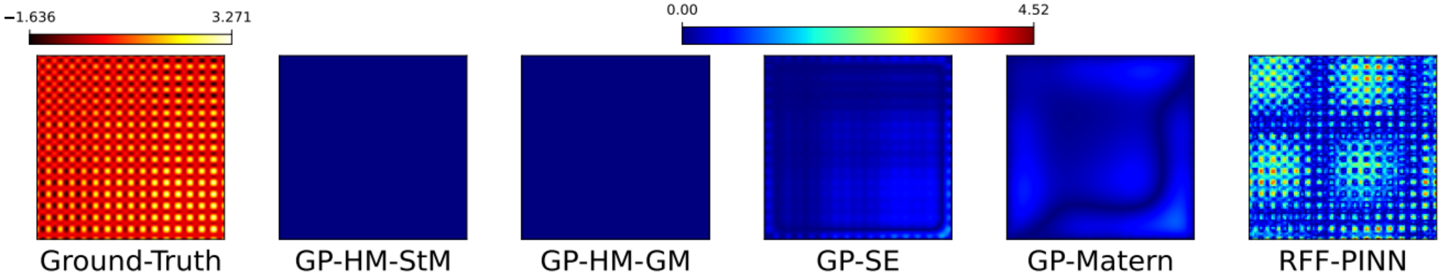


Figure 3: Point-wise solution error for 2D Allen-cahn equation, and the solution is $(\sin(x) + 0.1\sin(20x) + \cos(100x))(\sin(y) + 0.1\sin(20y) + \cos(100y))$.

Thank you!

Q&A