The Application of Gaussian Processes in Time Series and PDE-solving

RUC 2025-09-22

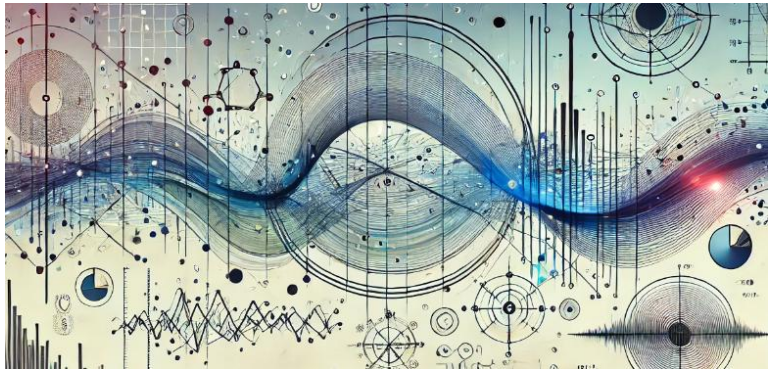
Presenter: 方楒楷 Shikai Fang

BayOTIDE: Bayesian Online Multivariate Time Series Imputation with Functional Decomposition

Spotlight Paper of ICML 2024

Presenter: Shikai Fang

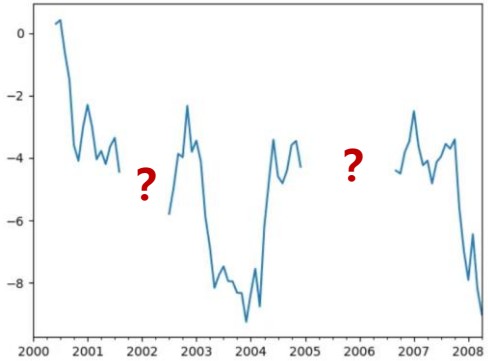
Shikai Fang, Qingsong Wen,
Yingtao Luo, Shandian Zhe, Liang Sun

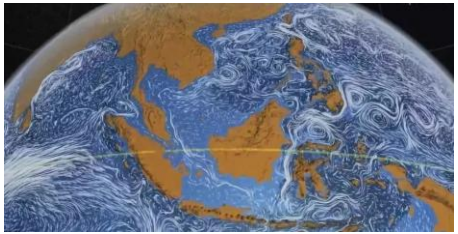


https://github.com/xuangu-fang/BayOTIDE

Imputation of Multi-Var. Time Series

- Time series data is ubiquitous
- The same with missing values…
- **Robust and efficient imputation** is crucial





Climate

Energy

Traffic

Finance

Limitations of current methods

- Regulate timestamp -> underuse **continuous** temporal info.
- Sensitive to noise -> call for **uncertainty-aware** model
- Black-box model -> lack of interpretability
- **Offline** infer. -> not efficient for fast-generated streaming seq.

BayOTIDE: Bayesian online TS Imputation

Properties / Methods	**BayOTIDE**	TIDER	Statistic-based	DNN-based	Diffusion-based
Uncertainty-aware	✓	✗	✗	✓̸	✓
Interpretability	✓	✓	✓	✗	✗
Continuous modeling	✓	✗	✗	✗	✓̸
Inference manner	**online**	offline	offline	offline	offline

Table 1: Comparison of *BayOTIDE* and main-stream multivariate time series imputation methods. ✓̸ means only partial models in the family have the property, or it's not clear in the original paper. For example, only deep models with probabilistic modules can offer uncertainty quantification, such as GP-VAE (Fortuin et al., 2020), but most deep models cannot. The diffusion-based CSDI (Tashiro et al., 2021) and CSBI (Chen et al., 2023) take timestamps as input, but the model is trained with discretized time embedding.

Method: Function Decomposition

- **Imputation ⇔ Low-rank function approximation**

$$\underset{t \to \mathbb{R}^D}{\mathbf{X}(t)} = \mathbf{U}\mathbf{V}(t) = \underset{\mathbb{R}^{D\times D_r}}{[\mathbf{U}_{\text{trend}}}, \underset{\mathbb{R}^{D\times D_s}}{\mathbf{U}_{\text{season}}}] \begin{bmatrix} \mathbf{v}_{\text{trend}}(t), \\ \mathbf{v}_{\text{season}}(t) \end{bmatrix}$$

Weights

$$D_r + D_s << D$$

Latent temporal functions (trend + seasonal factors)

$$\mathbf{v}_{\text{trend}}(t) = \text{concat}[v^i_{\text{trend}}(t)]_{i=1\ldots D_r},$$
$$\mathbf{v}_{\text{season}}(t) = \text{concat}[v^j_{\text{season}}(t)]_{j=1\ldots D_s},$$

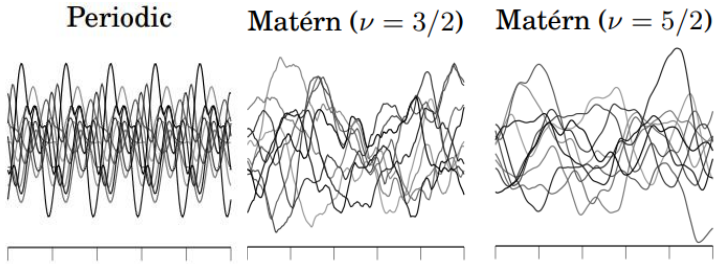
Gaussian Processes (GP) as function estimator

$$\begin{cases} \mathbf{v}_{\text{trend}}(t) = \text{concat}[v^i_{\text{trend}}(t)]_{i=1\ldots D_r}, \\ \mathbf{v}_{\text{season}}(t) = \text{concat}[v^j_{\text{season}}(t)]_{j=1\ldots D_s}, \end{cases}$$

$$\begin{cases} v^i_{\text{trend}}(t) \sim \mathcal{GP}(0, \kappa_{\text{Matérn}}) \\ v^j_{\text{season}}(t) \sim \mathcal{GP}(0, \kappa_{\text{periodic}}) \end{cases}$$

Facts of GPs:

- Powerful prob. functional model
- Characterized by kernel:



- Non-scalable : O(N^3) time cost

State-Space Gaussian Process

Linear-Cost GP with **Chain-Structure**

Temporal GPs

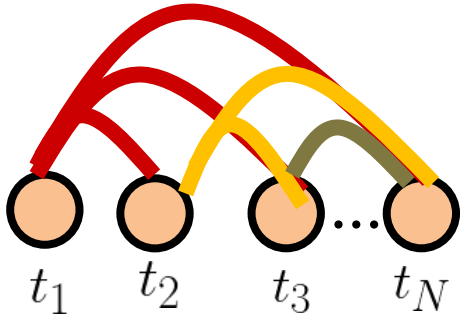
Temporal States:

LTI-SDE

$$\frac{\mathrm{d}\gamma_{\mathbf{r}}(t)}{\mathrm{d}t} = \mathbf{F}\gamma_{\mathbf{r}} + \mathbf{L}\xi(t)$$

discrete form

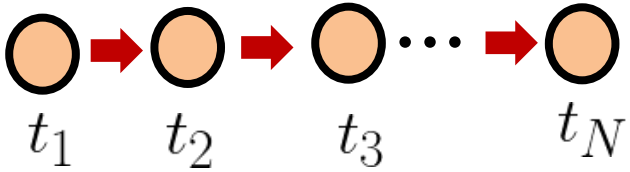
State Space Model



Space: $\mathcal{O}(N^2)$

Time: $\mathcal{O}(N^3)$

$$p\left(\gamma_{\mathbf{r}}\left(t_{n+1}\right) \mid \gamma_{\mathbf{r}}\left(t_n\right)\right) = \mathcal{N}\left(\gamma_{\mathbf{r}}\left(t_{n+1}\right) \mid \mathbf{A}_n\gamma_{\mathbf{r}}\left(t_n\right), \mathbf{Q}_n\right)$$



Space: $\mathcal{O}(N)$

Time: $\mathcal{O}(N)$

Joint Prob. of BayOTIDE

Noise

Channel-wise weight

Joint prob. $$p(\mathbf{Y}, \mathbf{V}(t), \mathbf{U}, \tau) = \mathrm{Gam}(\tau \mid a_0, b_0) \prod_{d=1}^{D} \mathcal{N}(\mathbf{u}^d \mid \mathbf{0}, \mathbf{I})$$

O(N^3) $$\cdot \prod_{j=1}^{D_s} \mathcal{GP}(0, \kappa_{\text{periodic}}) \prod_{i=1}^{D_r} \mathcal{GP}(0, \kappa_{\text{Matérn}}) \cdot p(\mathbf{Y}|\mathbf{U}, \mathbf{V}(t), \tau)$$

Likelihood

O(N)! $$P(\mathbf{Z}(t_1)) \prod_{i=1}^{N-1} P(\mathbf{Z}(t_{n+1})|\mathbf{Z}(t_n))$$

States of temporal factors.

Bayesian Online Learning

- **Online learning/ Straming Inference:**
  data come, model update, date drops

- Princinple: Incremental version of Bayes' rule:

Posterior on old data

$$p(\boldsymbol{\theta} \mid \mathcal{D}_{\text{old}} \cup \mathcal{D}_{\text{new}}) \propto p(\boldsymbol{\theta} \mid \mathcal{D}_{\text{old}})\, p(\mathcal{D}_{\text{new}} \mid \boldsymbol{\theta})$$

Posterior on all data

Likelihood on current model

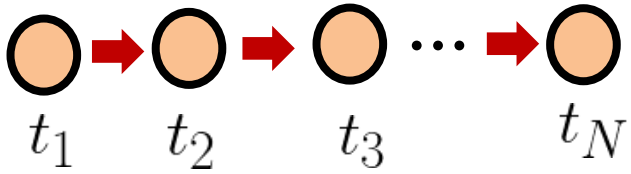
Online Learning ⇔ Kalman Filter!

Incremental version of Bayes' rule:

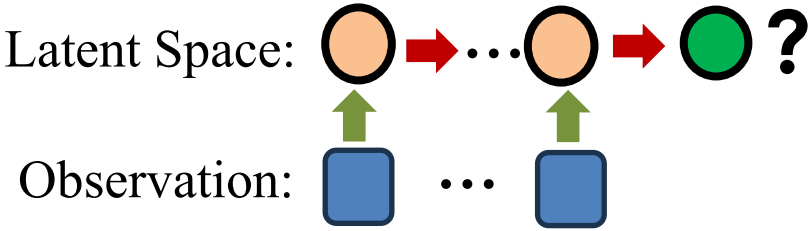
$$p(\boldsymbol{\theta} \mid \mathcal{D}_{\text{old}} \cup \mathcal{D}_{\text{new}}) \propto p(\boldsymbol{\theta} \mid \mathcal{D}_{\text{old}})\, p(\mathcal{D}_{\text{new}} \mid \boldsymbol{\theta})$$

+

Chain-Struture of latent factors



→ **Kalman Filter!**

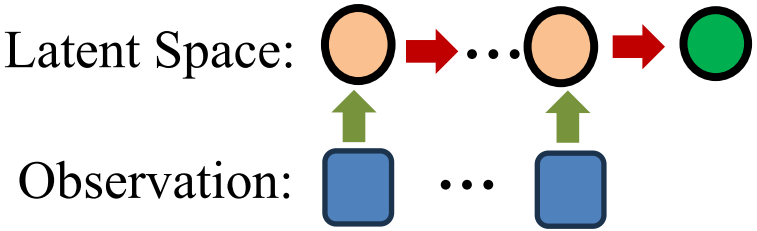


Moment Matching & Message Passing

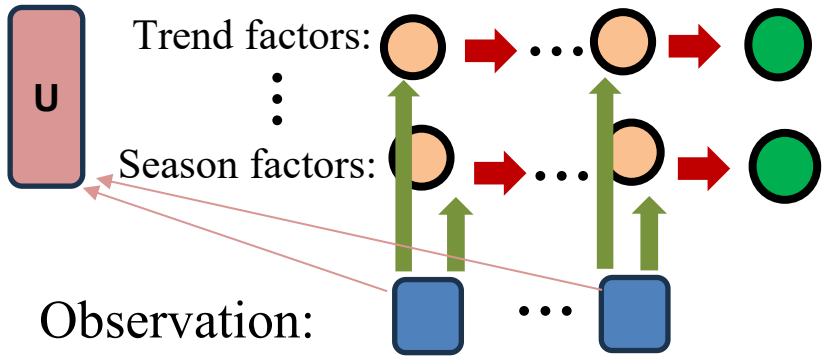
Last Challenge:

- One obvervation ? States of multi-factors + weights + noise

- Classical Kalman Filter



- BayOTIDE



Moment Matching & Message Passing

- Conditional Moment-Matching

$$p(y_{n+1}^d \mid \Theta) \approx \mathcal{Z} f_{n+1}^d(\mathbf{Z}(t_{n+1})) f_{n+1}^d(\mathbf{u}_d) f_{n+1}^d(\tau)$$

Observation llk. (Gaussian)

Msgs to **multi**-factors (Gaussian)

Msgs to **weights and noise** (Gaussian & Gamma)

- Message passing and merging

$$q(\tau|\mathcal{D}_{t_{n+1}}) = q(\tau|\mathcal{D}_{t_n}) \prod_{d=1}^{D} f_{n+1}^d(\tau)$$

$$q(\mathbf{u}^d|\mathcal{D}_{t_{n+1}}) = q(\mathbf{u}^d|\mathcal{D}_{t_n}) f_{n+1}^d(\mathbf{u}^d)$$

$$q(\mathbf{Z}(t_{n+1})) = q(\mathbf{Z}(t_n)) p(\mathbf{Z}(t_{n+1})|\mathbf{Z}(t_n)) \prod_{d=1} f_{n+1}^d(\mathbf{Z}(t_{n+1}))$$

All closed-form update!

Algorithm of BayOTIDE

Algorithm 1 *BayOTIDE*

Input: observation $\mathbf{Y} = \{\mathbf{y}_n\}_{n=1}^N$ over $\{t_n\}_{n=1}^N$, D_s, D_r, the kernel hyperparameters.
Initialize $q(\tau), q(\mathcal{W}), \{q(\mathbf{Z}(t_n))\}_{n=1}^N$.
for $t = 1$ **to** N **do**
  Approximate messages by (12) for all observed channels in parallel.
  Update posterior of τ and $\mathbf{U}$ by (13) and (14) for all observed channels in parallel.
  Update posteriors of $\mathbf{Z}(t)$ using Kalman filter by (15).
end for
Run RTS smoother to obtain the full posterior of $\mathbf{Z}(t)$.
Return: $q(\tau), q(\mathcal{W}), \{q(\mathbf{Z}(t_n))\}_{n=1}^N$

Could be irregulate

- **Parallel over channels**
- **Online update for all para.**

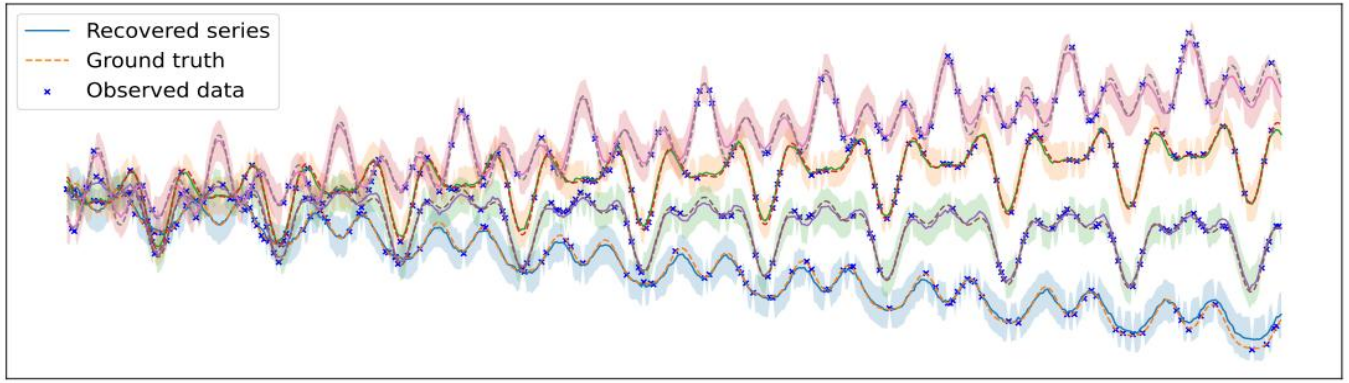
Allow prob. imputation over **arbitrary timestamp** (even never seen in training)

Experiments: Simulation

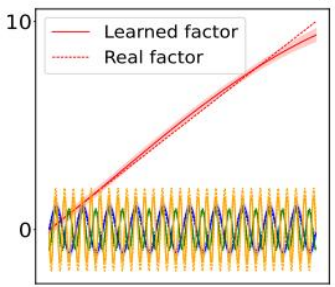
- **20% noisy observations**

$$\mathbf{U} = \begin{pmatrix} 1 & 1 & -2 & -2 \\ 0.4 & 1 & 2 & -1 \\ -0.3 & 2 & 1 & -1 \\ -1 & 1 & 1 & 0.5 \end{pmatrix},$$

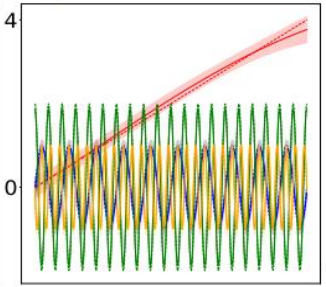
$$\mathbf{V}(t) = \begin{pmatrix} 10t, \\ \sin(20\pi t), \\ \cos(40\pi t), \\ \sin(60\pi t) \end{pmatrix}.$$



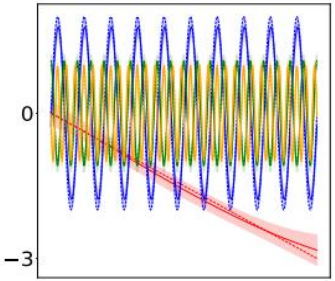
(a) Imputation results of the four-channel synthetic time series.



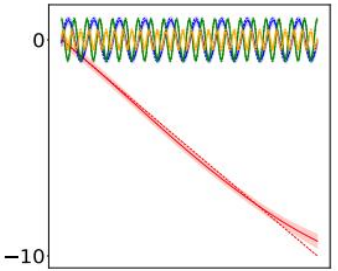
(b) Channel#1's factors



(c) Channel#2's factors



(d) Channel#3's factors



(e) Channel#4's factors

Experiments on real-world tasks

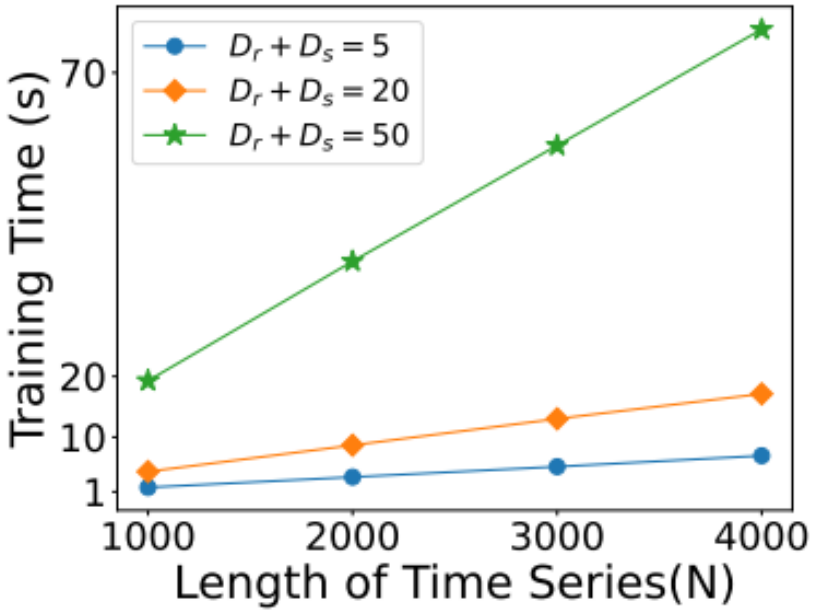
Online beats offline!

Observed-ratio=50%	*Traffic-GuangZhou*			*Solar-Power*			*Uber-Move*		
Metrics	RMSE	MAE	CRPS	RMSE	MAE	CRPS	RMSE	MAE	CRPS
Deterministic & Offline									
SimpleMean	9.852	7.791	-	3.213	2.212	-	5.183	4.129	-
BRITS	4.874	3.335	-	2.842	1.985	-	2.180	1.527	-
NAOMI	5.986	4.543	-	2.918	2.112	-	2.343	1.658	-
SAITS	4.839	3.391	-	2.791	1.827	-	1.998	1.453	-
TIDER	4.708	3.469	-	**1.679**	0.838	-	1.959	1.422	-
Probabilistic & Offline									
Multi-Task GP	4.887	3.530	0.092	2.847	1.706	0.203	3.625	2.365	0.121
GP-VAE	4.844	3.419	0.084	3.720	1.810	0.368	5.399	3.622	0.203
CSDI	4.813	3.202	0.076	2.276	0.804	0.166	1.982	1.437	0.072
CSBI	4.790	3.182	0.074	2.097	1.033	0.153	1.985	1.441	0.075
Probabilistic & Online									
BayOTIDE-fix weight	11.032	9.294	0.728	5.245	2.153	0.374	5.950	4.863	0.209
BayOTIDE-trend only	4.188	2.875	0.059	1.789	0.791	0.132	2.052	1.464	0.067
BayOTIDE	**3.820**	**2.687**	**0.055**	1.699	**0.734**	**0.122**	**1.901**	**1.361**	**0.062**

Table 2: RMSE, MAE and CRPS scores of imputation results of all methods on three datasets with observed ratio = 50%.

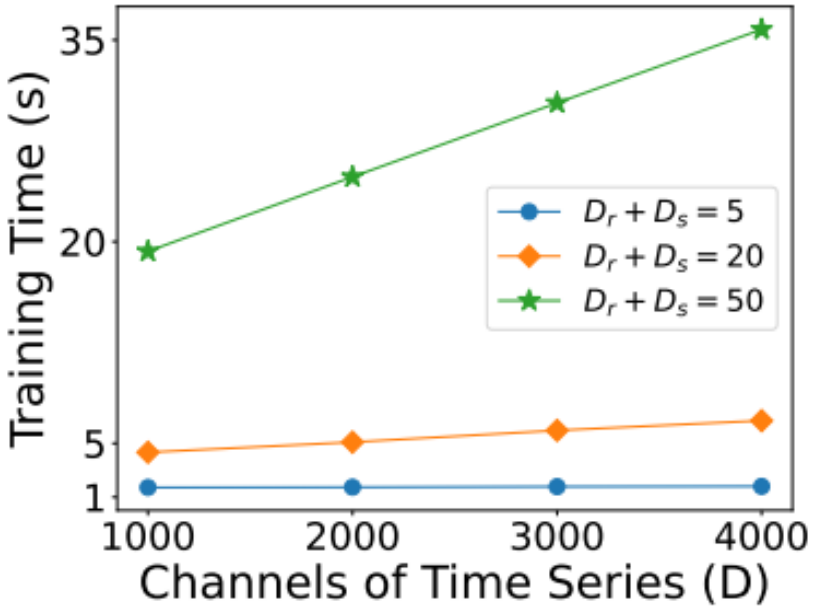
Scalability of BayOTIDE

D fixed as 1000



(b) Scalability over time series length.

N fixed as 1000



(c) Scalability over the channels number.

Take Home Messages

- **Decomposition** matters for high-order time series
- **GP** is powerful for latent functional model
- **SS-GP + Bayes MP** offers **linear** complexity and enables **online** inference

知乎解读:

Github Repo

https://github.com/xuangu-fang/BayOTIDE

Solving High Frequency and Multi-Scale PDEs with Gaussian Processes

NAHOMCON 2024

June 19th, 2024

Presenter: Shikai Fang

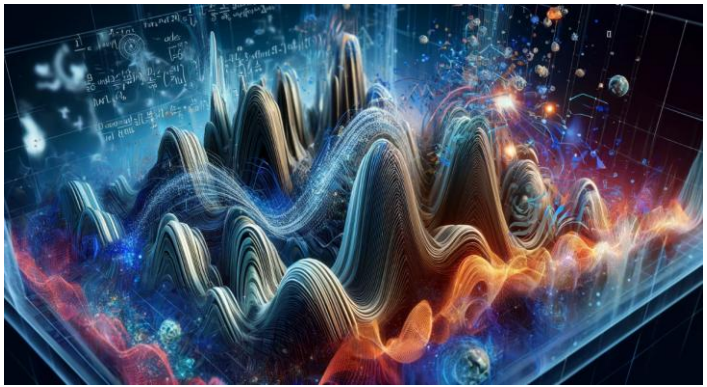
Shikai Fang*, Madison Cooley*, Da Long*,
Shibo Li, Robert M. Kirby, Shandian Zhe

Published at ICLR 2024

ICLR

SCI

KAHLERT SCHOOL OF COMPUTING



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

PDE: First Principle of Scientific Computing

- General form of PDE

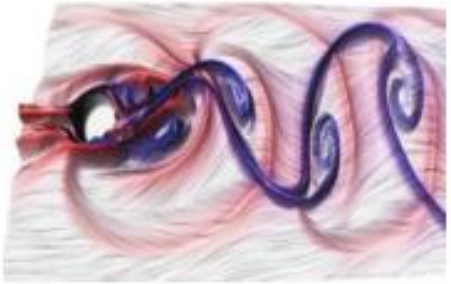
Differential opeartor

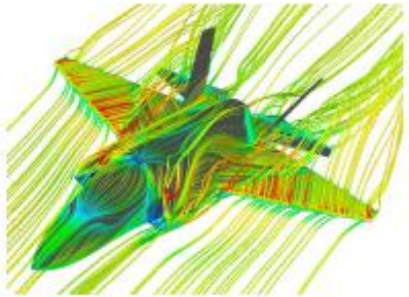
Target solution

$$\mathcal{F}[u](\mathbf{x}) = f(\mathbf{x}) \quad (\mathbf{x} \in \Omega), \qquad u(\mathbf{x}) = g(\mathbf{x}) \quad (\mathbf{x} \in \partial\Omega),$$

Equations in domain

Boundary conditions





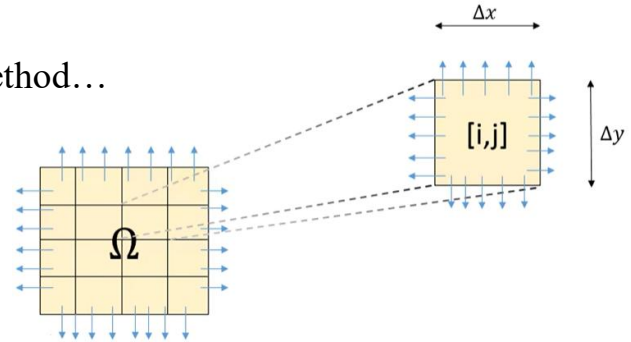
Numerical Solver & ML Solver

- Numerical Solver:

  - Finite Element Method (FEM), Finite Volume Method (FVM), Spectrum method…

  👍 : Solid theory, high stability, mature software libraries

  ❓: Not scalable enough for high-dim. cases, many "tricks" needed



- ML slovers of **PINN**[Raissi, JCP 2019] family:

  - **Deep Neural Network (DNN)** as the approx. of solution: $\widehat{u}_{\boldsymbol{\theta}}(\mathbf{x}) \approx u_{\boldsymbol{\theta}}(\mathbf{x})$

  - Canonical objective func. : $\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} \; L_b(\boldsymbol{\theta}) + L_r(\boldsymbol{\theta}),$

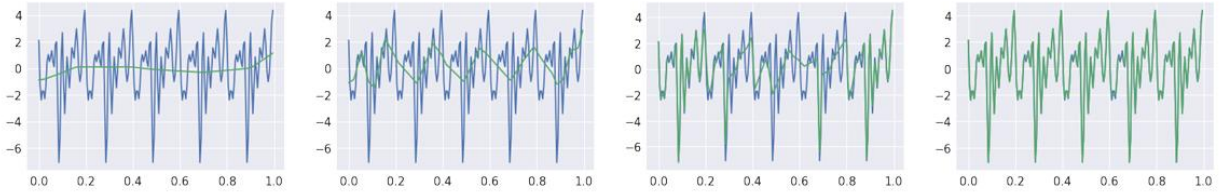
Boundary term **Residual term**

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

Hard cases for PINN: high-frequency

- “Specturm bias of DNN” [Rahaman. ICML 2019]：

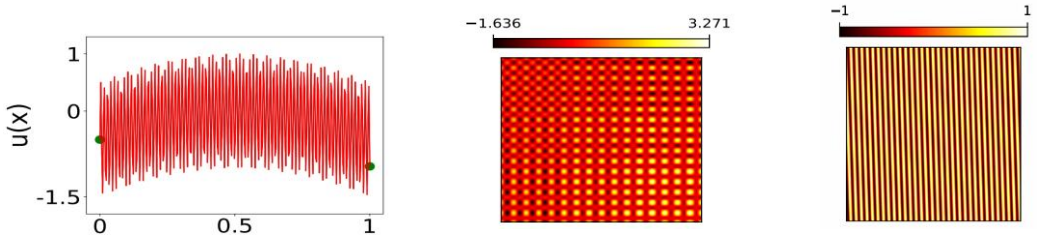
  DNNs are easy to capture low-freq. info, but much hard to learn high-freq.



(a) Iteration 100 (b) Iteration 1000 (c) Iteration 10000 (d) Iteration 80000

Figure 2. The learnt function (green) overlayed on the target function (blue) as the training progresses. The target function is a superposition of sinusoids of frequencies $\kappa = (5, 10, ..., 45, 50)$, equal amplitudes and randomly sampled phases.

- PINN is hard to handle PDEs with high-freq. & multi-scale components



[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 solutions in frequency domain
- Build a frequency-aware surrogate model

How?

- Gaussian Processes(GPs)!

$$\begin{cases} u(\cdot) \sim \mathcal{GP}(m(\cdot), \mathrm{cov}(\cdot,\cdot)) \\ \mathrm{cov}\left(\partial_{x_1x_2}u(\mathbf{x}), u\left(\mathbf{x}'\right)\right) = \partial_{x_1x_2}k\left(\mathbf{x}, \mathbf{x}'\right) \\ \mu(\mathbf{x}) = \mathrm{cov}(f(\mathbf{x}), \mathbf{f})\mathbf{K}^{-1}\mathbf{f} \end{cases}$$

Facts of GPs:

- Expressive function estimator
- Characterized by kernel/cov func.
- Nice property to track derivative

Model of GP-HM

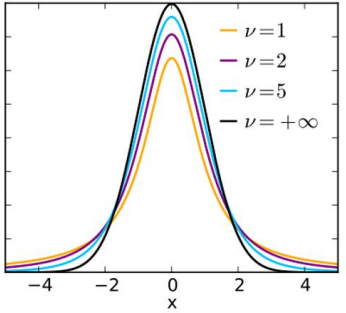
- Model PDE solution's **power spectrum** with a mixture of student-t (St) distribution

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

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

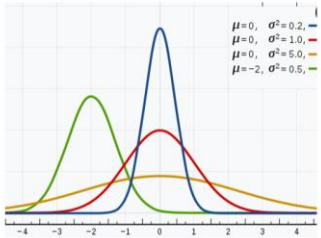
Non-negative component weight

One principle frequency μ_q



- Alternative: a mixture of Gaussian distribution

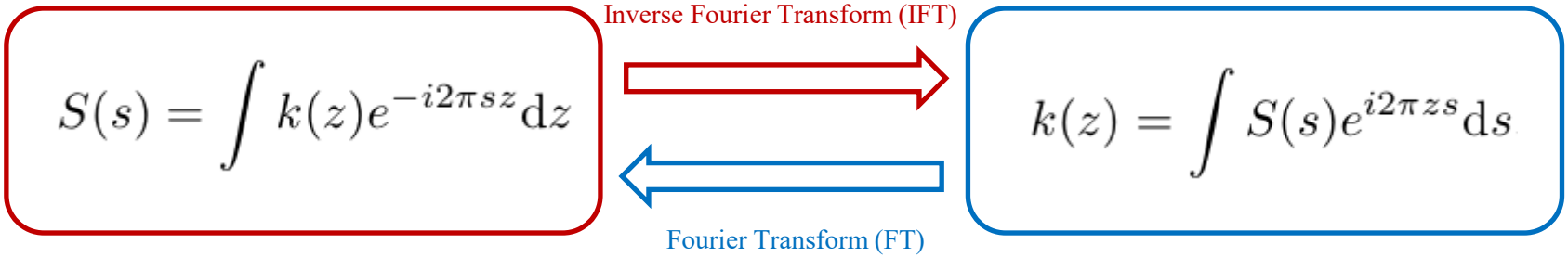
$$S(s) = \sum_{q=1}^{Q} w_q \mathcal{N}(s; \mu_q, \rho_q^2)$$



From spectrum to covariance

Wiener-Khinchin theorem:

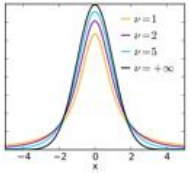
For a stationary random process, its power spectrum and the auto-correlation form a **Fourier pair.**

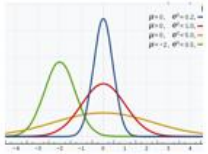


Kernel of GPs!

From spectrum to covariance

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





Wiener-Khinchin theorem &
Inverse Fourier transform (**IFT**)

GP kernels!

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

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

(known as *spectral mixture kernel*)

GP with Stm/GM kernel as **frequency-aware** surrogate model:

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

Jeffreys prior for shrinkage effect

We don't know the optimal number of freq. component !

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

Kernel parameters to learn

Solution:

Start from a large Q (e.g., 50), optimize the component weights in log domain

$$\overline{w}_q = \log(w_q)$$

Assign each component weights with Jeffreys priors:

$$p(w_q) = p(\overline{w}_q) \left| \frac{\mathrm{d}\overline{w}_q}{\mathrm{d}w_q} \right| \propto \frac{1}{w_q}$$

Priors with strong **shrinkage effect!**

Most components will be "sparse out"!

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

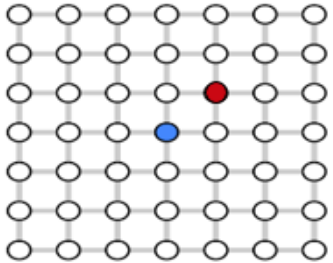
$$\mathcal{L}(\mathcal{U}, \Theta, \tau_1, \tau_2) = \log \mathcal{N}(\mathrm{vec}(\mathcal{U}) | \mathbf{0}, \mathbf{C}) + \lambda_b \cdot \log \mathcal{N}(\mathbf{g} | \mathbf{u}_b, \tau_1^{-1}\mathbf{I}) + \log \mathcal{N}(\mathbf{0} | \mathrm{vec}(\mathcal{H}), \tau_2^{-1}\mathbf{I})$$

混合尺度-多频先验　　边界条件似然函数　　方程梯度似然函数

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

Structured kernel for efficient computation

For grids with resolution $M_1 \times \ldots \times M_d$, we will have $M = \prod M_d$ 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: $\mathcal{O}\left(\sum M_d^2 + M\right)$

Linear cost respect of # allocation point!

- Product kernel and cross covariance:

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

$$\mathrm{cov}(\partial_{x_1 x_2} u(\mathbf{x}), u(\mathbf{x}')) = \partial_{x_1}\kappa(x_1, x_1') \cdot \partial_{x_2}\kappa(x_2, x_2') \cdot \prod_{j \neq 1,2} \kappa(x_j, x_j').$$

- Kronecker product structure of kernel matrix

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

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

Numerical results

Method	Poission-1D					Poission-2D		1D Allen-cahn		2D Allen-cahn	1D Advection
	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}
PINN	1.36e0	1.40e0	1.00e0	1.42e1	6.03e-1	1.63e0	9.99e-1	1.41e0	1.14e1	1.96e1	1.00e0
W-PINN	1.31e0	2.65e-1	1.86e0	2.60e1	6.94e-1	1.63e0	6.75e-1	1.34e0	1.45e1	2.03e1	1.01e0
RFF-PINN	4.97e-4	2.00e-5	7.29e-2	2.80e-1	5.74e-1	1.69e0	7.99 e-1	1.24e-3	2.46e-1	7.17e-1	9.96e-1
Rowdy	1.70e0	1.00e0	1.00e0	1.01e0	1.03e0	2.24e1	7.36e-1	1.30e0	1.31e0	1.18e0	1.03e0
Spectral method	2.36e-2	3.47e0	1.02e0	1.02e0	9.98e-1	1.58e-2	1.04e0	2.34e-2	2.45e1	2.45e1	2.67e0
Chebfun	**3.05e-11**	**1.17e-11**	**5.81e-11**	**1.14e-10**	**8.95e-10**	N/A	N/A	**1.39e-08**	**2.94e-10**	N/A	1.39e0
Finite Difference	5.58e-1	4.78e-2	2.34e-1	1.47e0	1.40e0	2.33e-1	1.75e-2	2.32e-01	2.36e-1	3.23e0	1.29e-1
GP-SE	2.70e-2	9.99e-1	9.99e-1	3.19e-1	9.75e-1	9.99e-1	9.53e-1	2.74e-2	1.06e-2	3.48e-1	9.99e-1
GP-Matérn	3.32e-2	9.8e-1	5.15e-1	1.83e-2	6.27e-1	6.28e-1	3.54e-2	3.32e-2	5.16e-2	2.96e-1	9.99e-1
GP-HM-GM	**3.99e-7**	2.73e-3	3.92e-6	1.55e-6	1.82e-3	**6.46e-5**	1.06e-3	**4.91e-6**	**4.24e-6**	5.78e-3	3.59e-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**	7.71e-6	4.76e-6	**2.99e-3**	**9.08e-4**

Relative L_2 error. $u_1 = \sin(100x)$, $u_2 = \sin(x) + 0.1\sin(20x) + 0.05\cos(100x)$, $u_3 = \sin(6x)\cos(100x)$, $u_4 = x\sin(200x)$, $u_5 = \sin(500x) - 2(x-0.5)^2$, $u_6 = \sin(100x)\sin(100y)$, $u_7 = \sin(6x)\sin(20x) + \sin(6y)\sin(20y)$, $u_8 = \sin(100x)$, $u_9 = \sin(6x)\cos(100x)$,$u_{10} = (\sin(x) + 0.1\sin(20x) + \cos(100x)) \cdot (\sin(y) + 0.1\sin(20y) + \cos(100y))$ and $u_{11} = \sin(x - 200t)$.

Numerical results: visualization-1d

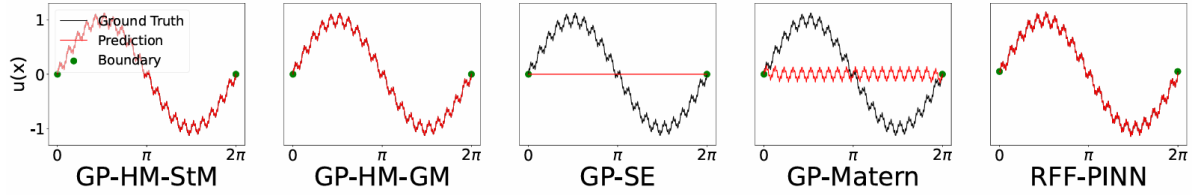


Figure 1: Prediction for the 1D Poisson equation with solution $\sin(x) + 0.1\sin(20x) + 0.05\cos(100x)$.

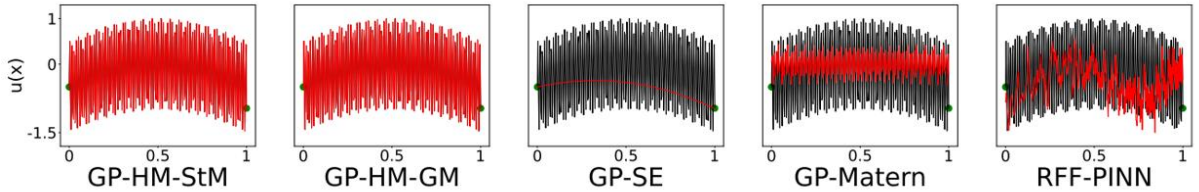


Figure 2: Prediction for the 1D Poisson equation with solution $\sin(500x) - 2(x - 0.5)^2$.

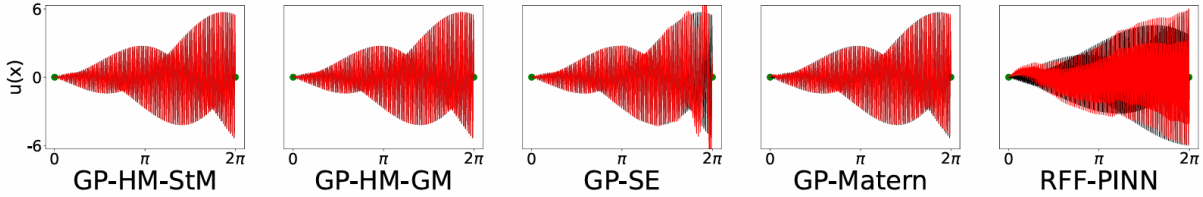


Figure 5: Prediction for the 1D Poisson equation with solution $x\sin(200x)$.

Numerical results: visualization-2d

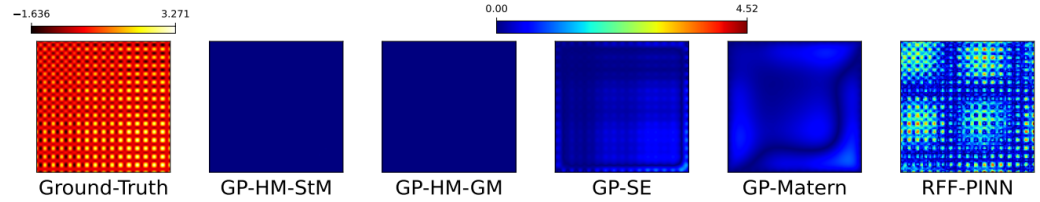


Figure 3: Point-wise solution error for 2D Allen-cahn equation, and the solution is $(\sin(x) + 0.1\sin(20x) + \cos(100x))\,(\sin(y) + 0.1\sin(20y) + \cos(100y))$.

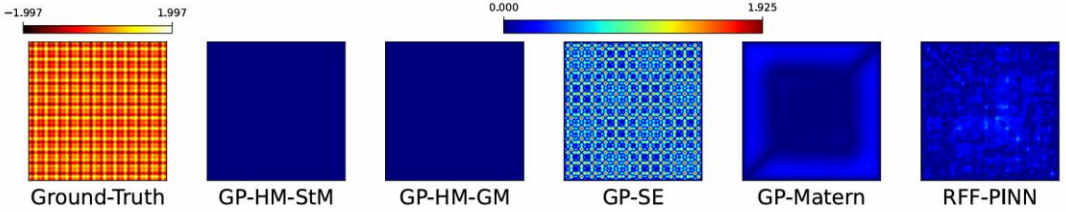


Figure 6: Point-wise solution error for 2D Poisson equation and the solution is $u(x) = \sin(6x)\sin(20x) + \sin(6y)\sin(20y)$.

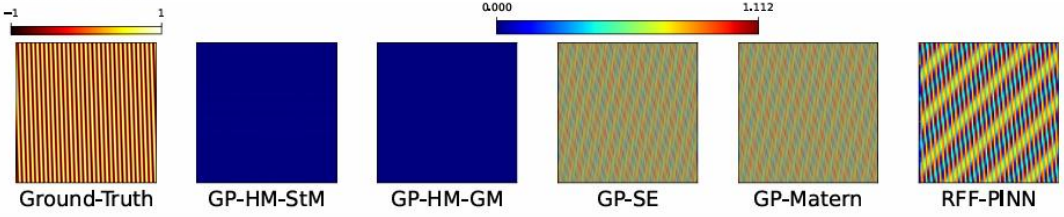
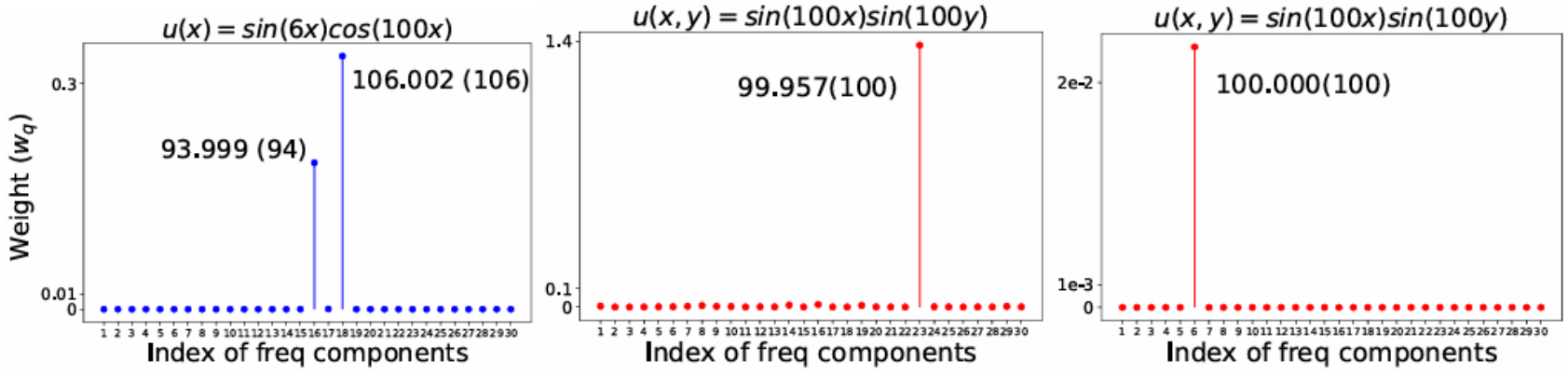


Figure 7: Point-wise solution error for 1D Advection equation and the solution is $\sin(x - 200t)$.

Numerical results: learned frequency



(a) Poisson-1D (b) Poisson-2D x-dim (c) Poisson-2D y-dim

Figure 4: The learned component weights and frequency values. For each number pair a(b) in the figure, "a" is the learned frequency by GP-HM, and "b" is the ground-truth. The expressions on the top are the solutions.

Take Home Messages

- Where NN works, **GP** should also work (as you can **scale it up**)
- **GP** is flexible to inject domain first principal as **prior**
  - **Mix-freq. in PDE**
  - **Season-Trend in TS**
- **Wiener-Khinchin theorem** is super helpful to inject design in freq. domain

知乎解读:

Github Repo

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

Thank you!
Q&A

个人主页:
xuangu-fang.github.io

知乎/小红书:
Id: “方轩固”

Email: xuangufang@gmail.com
Wechat: xuangu1996