Non-Additive Models for Multi-Fidelity Computer Experiments

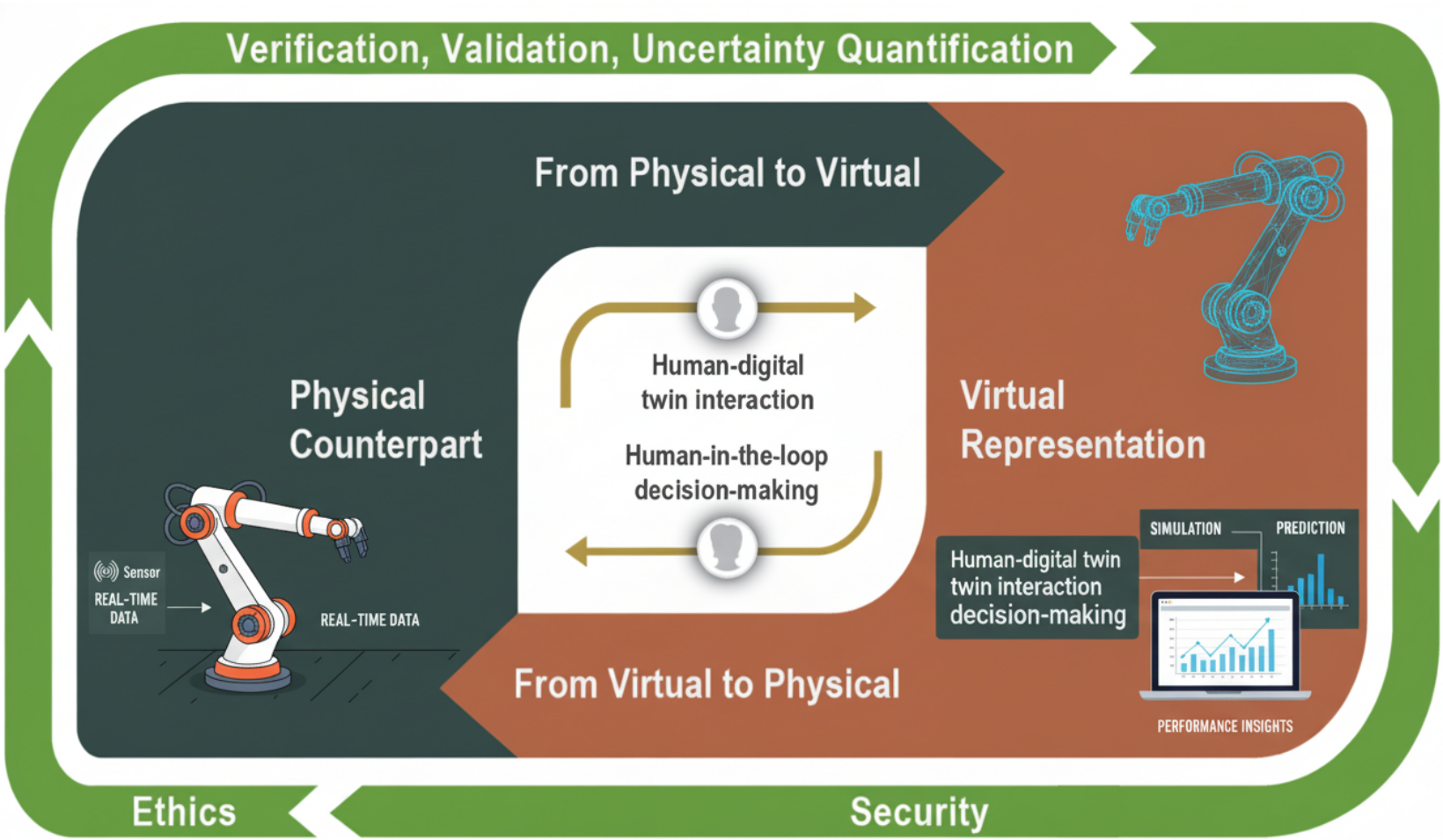
Junoh Heo

Assistant Professor
Department of Statistical Sciences
Wake Forest University
These works are supported by NSF DMS 2113407 and 2338018.

Statistics Seminar
School of Mathematical and Statistical Sciences, Clemson University
September 9, 2026

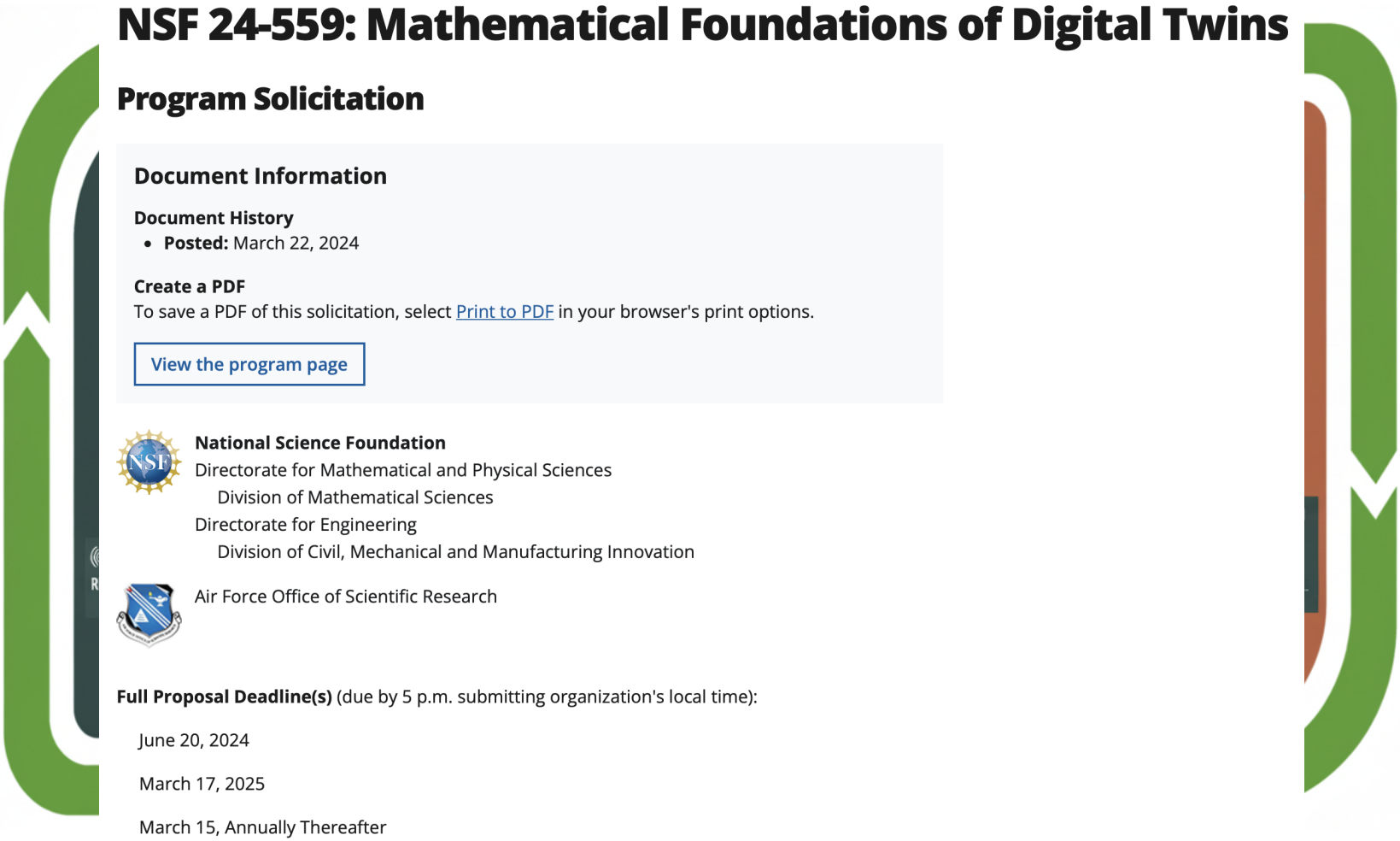
Outline

Digital Twins



National Academies of Sciences, Engineering, and Medicine (2023). Foundational Research Gaps and Future Directions for Digital Twins. https://doi.org/10.17226/26894

Digital Twins



NSF 24-559: Mathematical Foundations of Digital Twins

Program Solicitation

Document Information

Document History

- **Posted:** March 22, 2024

Create a PDF

To save a PDF of this solicitation, select Print to PDF in your browser's print options.

View the program page

National Science Foundation
Directorate for Mathematical and Physical Sciences
Division of Mathematical Sciences
Directorate for Engineering
Division of Civil, Mechanical and Manufacturing Innovation

Air Force Office of Scientific Research

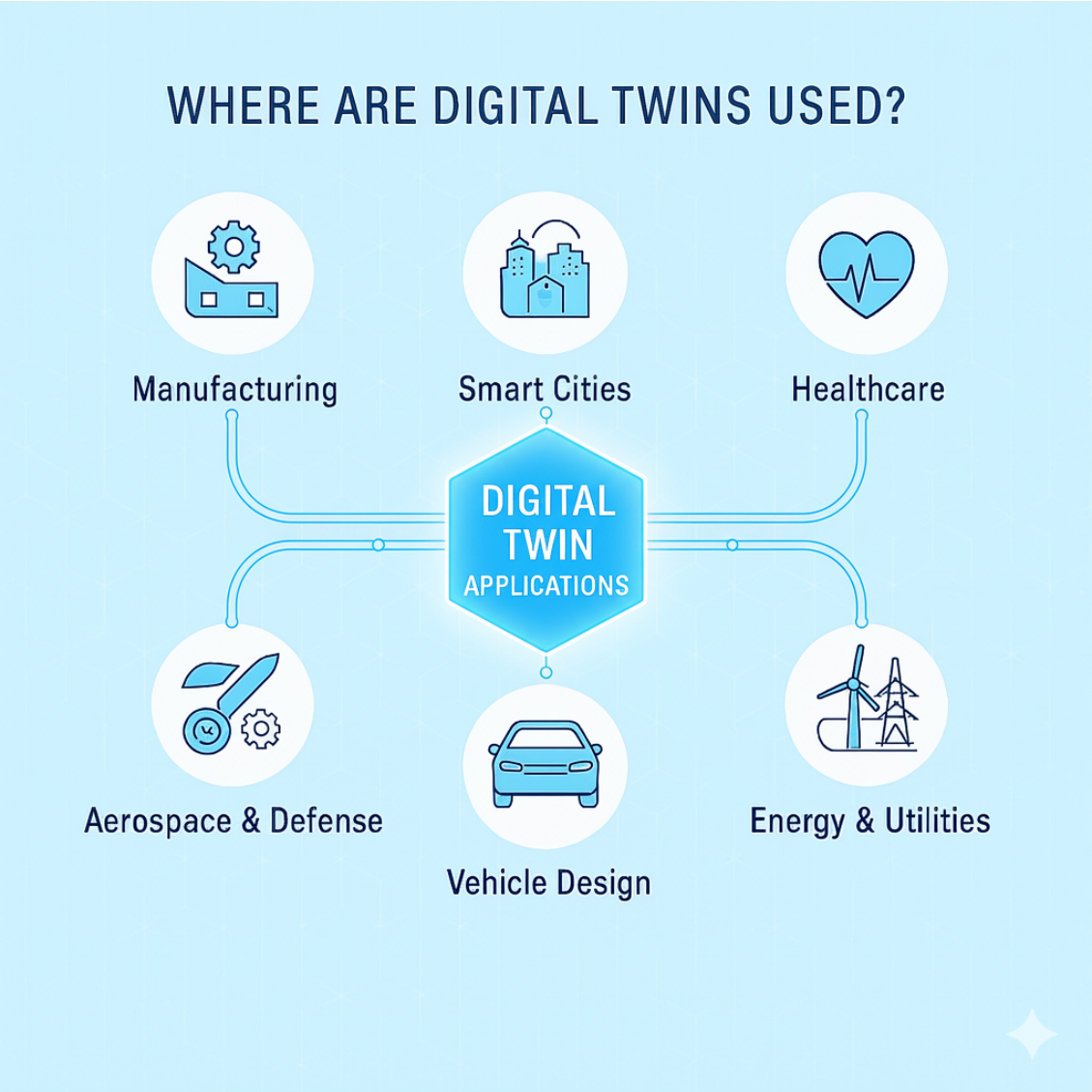
Full Proposal Deadline(s) (due by 5 p.m. submitting organization's local time):

June 20, 2024

March 17, 2025

March 15, Annually Thereafter

Digital Twins



Computer Experiments

- Real-world experiments are often too risky, costly, and time-consuming to perform directly.
- Computer models are mathematical representations of real-world systems.
- Computer simulations are used to solve these models (e.g., finite element / finite difference).

Computer Experiments

- No physical lab equipment.
- Deterministic: Same input $\rightarrow$ Same output (no random error)
- Computationally Expensive: A single simulation can take hours or days.

Surrogate Modeling

- The Challenge: We want to explore the design space (e.g., optimization, sensitivity analysis), but the simulation is too slow.
- The Solution: Construct a Surrogate Model (or Emulator).
- Goal: A statistical approximation that is:
  - **Fast** to evaluate.
  - **Accurate** enough to replace the expensive simulation.

Gaussian Process

- A popular statistical method for modeling computer experiments.
- Treat the simulator as a **Black Box** function $f(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^d$.
- **Input & Output**: $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ and $\mathbf{y} := (f(\mathbf{x}))_{\mathbf{x} \in \mathcal{X}}$.
- A Gaussian Process (GP) assumes a multivariate normal distribution over f:

$$\mathbf{y} \sim \mathcal{N}_n(\alpha \mathbf{1}_n, \tau^2 \mathbf{K}),$$

where $\mathbf{K} = K(\mathcal{X}, \mathcal{X})$ is the correlation matrix with $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

Kernel function

- Kernel function $k(\cdot, \cdot)$ determines the smoothness and behavior of GPs.
- Common choices include the squared exponential kernel and Matérn kernel (Stein, 1999).
- Anisotropic squared exponential kernel:

$$k(\mathbf{x}, \mathbf{x}') = \prod_{j=1}^{d} \exp\left(-\frac{(x_j - x_j')^2}{\theta_j}\right),$$

where $(\theta_1, \ldots, \theta_d)$ are the lengthscale hyperparameter.

Hyperparameter estimation

- Hyperparameters α, τ^2, and $\theta_1, \ldots, \theta_d$ are estimated by maximizing the log-likelihood:

$$-\frac{n}{2}\log 2\pi - \frac{n}{2}\log \tau^2 - \frac{1}{2}\log |\mathbf{K}| - \frac{1}{2\tau^2}(\mathbf{y} - \alpha \mathbf{1}_n)^\top \mathbf{K}^{-1}(\mathbf{y} - \alpha \mathbf{1}_n).$$

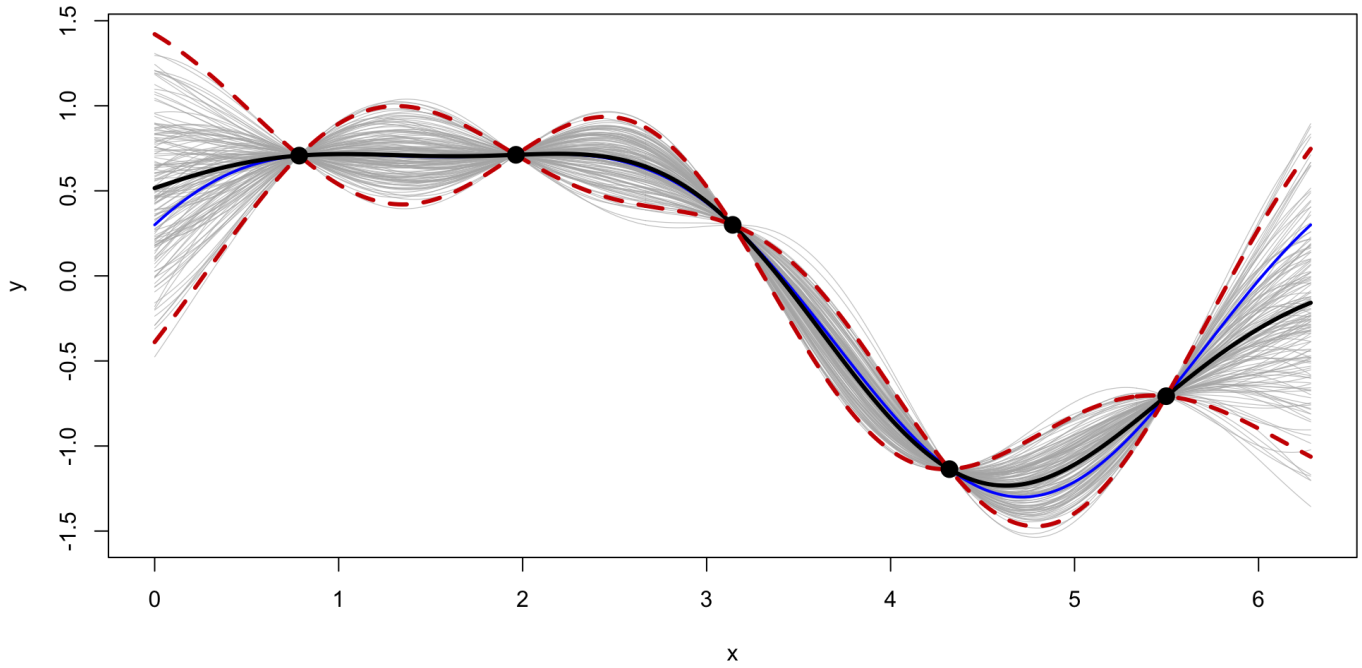
- Closed-form MLEs: $\hat{\alpha} = \frac{\mathbf{1}_n^\top \mathbf{K}^{-1}\mathbf{y}}{\mathbf{1}_n^\top \mathbf{K}^{-1}\mathbf{1}_n}$ and $\hat{\tau}^2 = \frac{(\mathbf{y}-\hat{\alpha}\mathbf{1}_n)^\top \mathbf{K}^{-1}(\mathbf{y}-\hat{\alpha}\mathbf{1}_n)}{n}$.

- The profile log-likelihood (up to constant) becomes

$$-\frac{n}{2}\log \hat{\tau}^2 - \frac{1}{2}\log |\mathbf{K}|.$$

- $\theta_1, \ldots, \theta_d$ are optimized using gradient-based algorithms (e.g., L-BFGS-B).

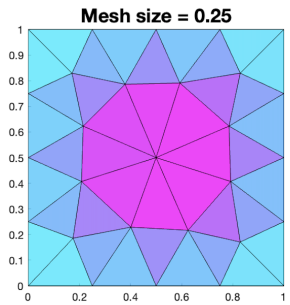
Why use GPs?

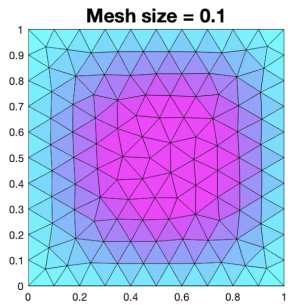


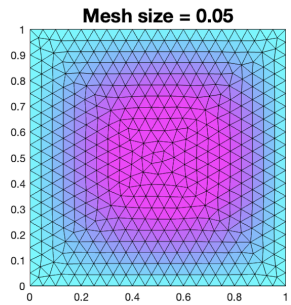
- Given training data $\{\mathbf{x}_i, y_i\}_{i=1}^n$, the GP predictive distribution at a new point $\mathbf{x}$ is:

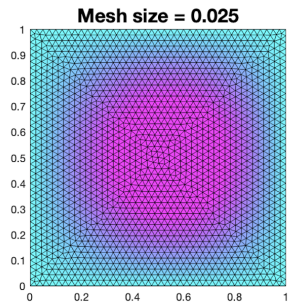
$$\mu(\mathbf{x}) = \alpha + k(\mathbf{x}, \mathcal{X})\mathbf{K}^{-1}(\mathbf{y} - \alpha\mathbf{1}_n),$$
$$\sigma^2(\mathbf{x}) = \tau^2\left[k(\mathbf{x}, \mathbf{x}) - k(\mathbf{x}, \mathcal{X})\mathbf{K}^{-1}k(\mathcal{X}, \mathbf{x})\right].$$

Multi-Fidelity Simulations









- Many simulators can be run at different **fidelity levels**.
- Fidelity is typically controlled by mesh resolution, time-step size, or convergence threshold.
- As a result, simulations can be run at different fidelities:
  - **High-fidelity** simulation: computationally expensive but accurate
  - **Low-fidelity** simulation: computationally cheaper but less accurate
  - (intermediate-fidelity simulation)

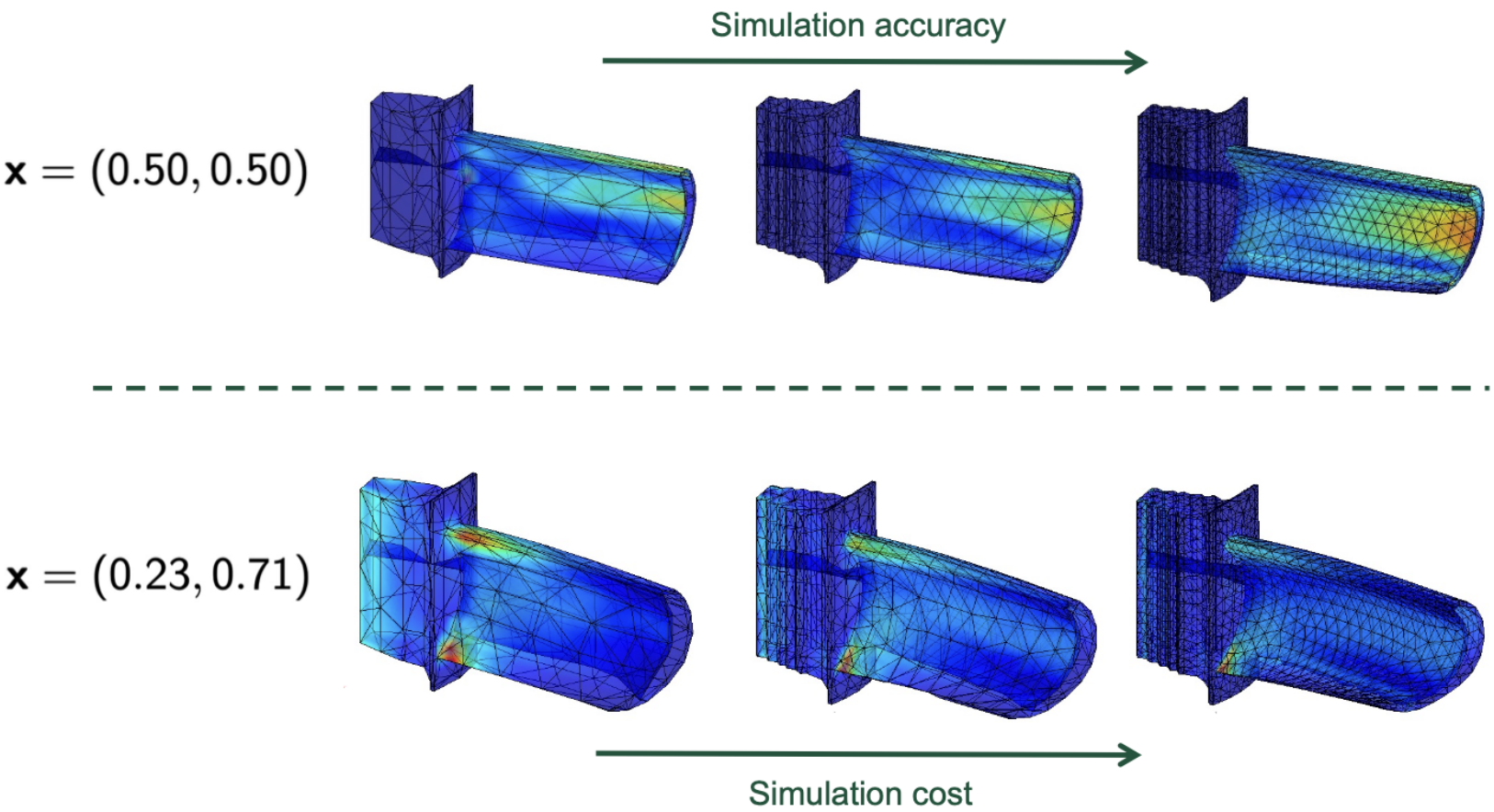
Motivating Example: Finite Element Simulations

- Thermal stress in a jet engine turbine blade can be analyzed through a static structural computer model.
- The model can be *numerically* solved via finite element method.
- **Input**: $\mathbf{x} = (x_1, x_2) = (\text{pressure}, \text{suction})$
- **Output**: $f(\mathbf{x})$: **maximum** of the thermal stress

Motivating Example: Finite Element Simulations

less accurate but cheaper

accurate but expensive



Statistical Emulation

- Can we leverage both low- and high-fidelity simulations in order to
  - maximize the accuracy of model predictions,
  - while minimizing the costs associated with the simulations?
  - By strategically integrating these simulations, we can potentially improve accuracy without sacrificing excessive computational resources.

Problem setup

- Let $f_l(\mathbf{x})$ represent the simulation output of the computer code with inputs $\mathbf{x} \in \Omega \subseteq \mathbb{R}^d$ at fidelity level $l = 1, \ldots, L$.
- **Goal**: Emulate $f_L(\mathbf{x})$.
- **Input**: $\mathcal{X}_l = \{\mathbf{x}_i^{[l]}\}_{i=1}^{n_l}$ for $l = 1, \ldots, L$.
- **Output**: $\mathbf{y}_l := (f_l(\mathbf{x}))_{\mathbf{x} \in \mathcal{X}_l}$ for $l = 1, \ldots, L$

Problem setup

- Assume **nested design**, i.e.,

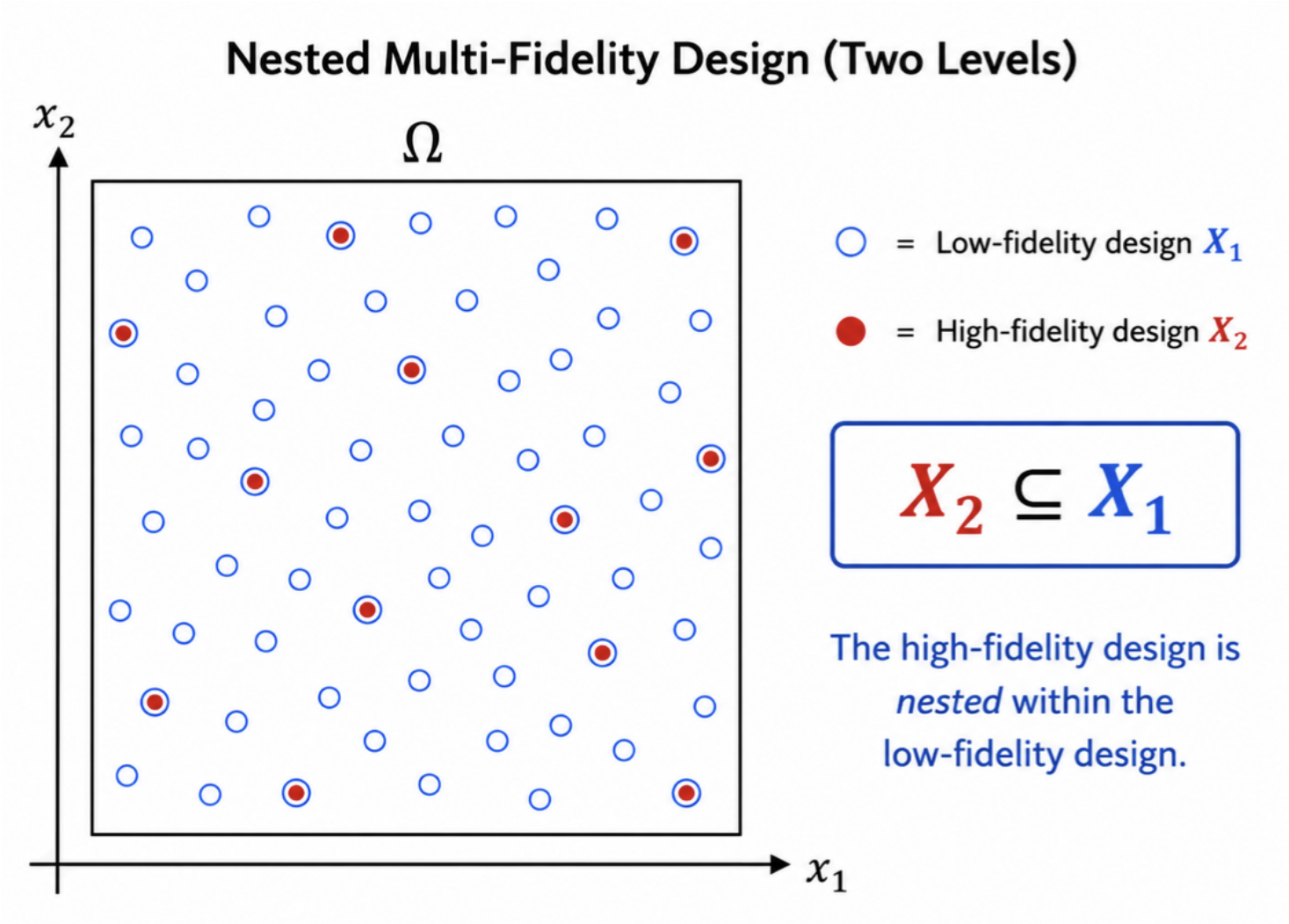
$$\mathcal{X}_L \subseteq \mathcal{X}_{L-1} \subseteq \cdots \subseteq \mathcal{X}_1 \subseteq \Omega,$$

  and $\mathbf{x}_i^{[l]} = \mathbf{x}_i^{[l-1]}$ for $i = 1, \ldots, n_l$.

- The nested property leads to more efficient inference in various multi-fidelity emulation approaches (Qian, 2009; Qian et al., 2009; Haaland and Qian, 2010).

- C_l denotes the simulation cost (e.g., in CPU hours) at fidelity level l with $0 < C_1 < C_2 < \ldots < C_L$.

Problem setup



Auto-regressive model

- The canonical approach is **auto-regressive** (AR) model (Kennedy & O'Hagan, 2000).

- AR model assumes additive structure of GPs.

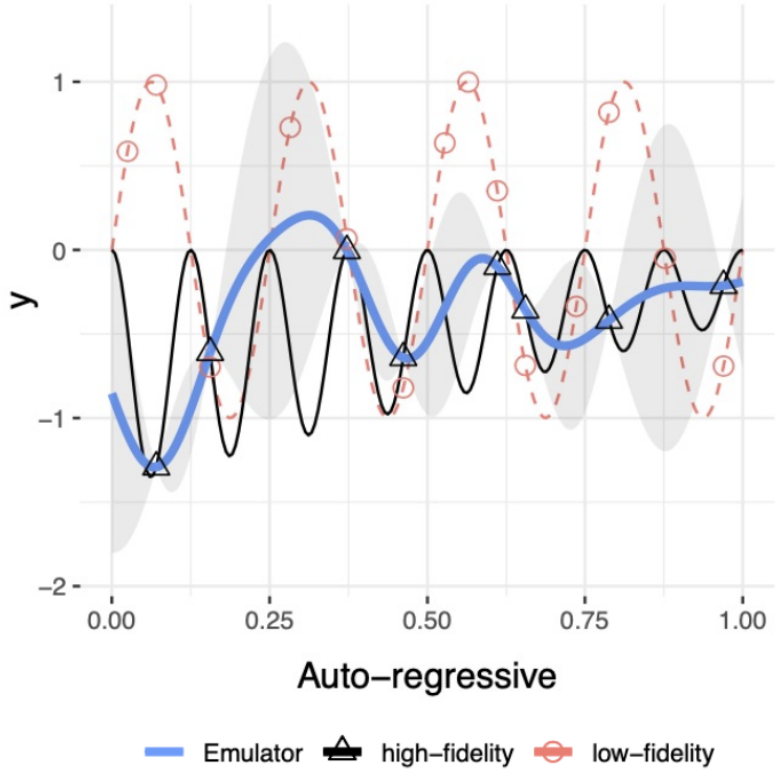
$$
\begin{aligned}
f_1(\mathbf{x}) &= Z_1(\mathbf{x}), \\
f_l(\mathbf{x}) &= \rho_{l-1} f_{l-1}(\mathbf{x}) + Z_l(\mathbf{x}), \quad \text{for} \quad 2 \leq l \leq L,
\end{aligned}
$$

  where $Z_1, \ldots, Z_L$ are independent GPs.

- Several extensions including Qian et al. (2006), Qian and Wu (2008), Le Gratiet (2013), Le Gratiet and Garnier (2014), and Perdikaris et al. (2017).

Existing Methods

- Q: Would it always follow an additive structure?



An example from Perdikaris et al. (2017), where $n_1 = 13$, $n_2 = 8$, $f_1(x) = \sin(8\pi x)$, and $f_2(x) = (x - \sqrt{2})f_1^2(x)$.

Recursive Non-Additive (RNA) model and Active learning

Heo, J. and Sung, C.-L. (2025).

Active learning for a recursive non-additive emulator for multi-fidelity computer experiments. *Technometrics*, 67(1):58–72.

- Winner, INFORMS 2023 Quality, Statistics & Reliability Best Student Paper Competition
- Winner, 2024 ASA SPES + Q&P Student Paper Competition
- Most downloaded paper in Technometrics (2024-2025)

RNA emulator

- A Recursive Non-Additive emulator (RNA emulator) to overcome this limitation in a recursive fashion:

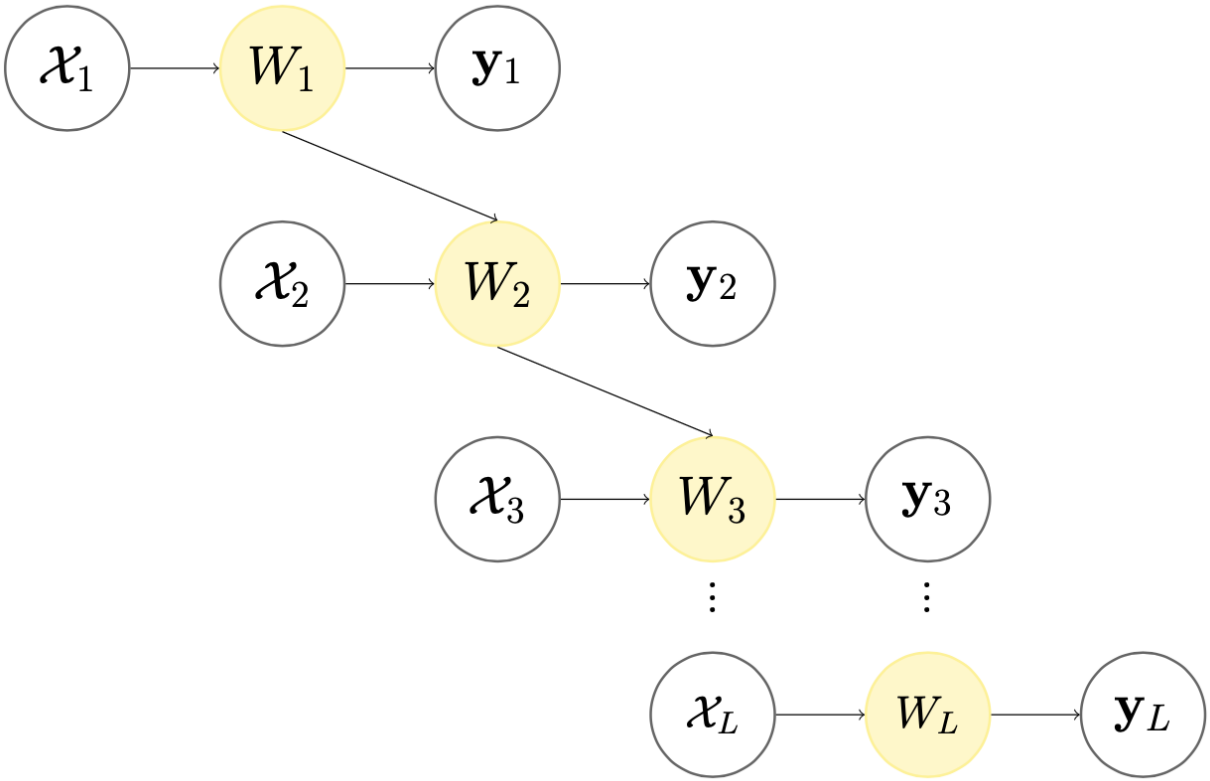
$$f_1(\mathbf{x}) = W_1(\mathbf{x}),$$

$$f_l(\mathbf{x}) = W_l(\mathbf{x}, f_{l-1}(\mathbf{x})), \quad l = 2, \cdots, L,$$

- The auto-regressive model ($f_l(\mathbf{x}) = \rho_{l-1} f_{l-1}(\mathbf{x}) + Z_l(\mathbf{x})$) becomes a special case!

- Model the relationship $\{W_l\}_{l=1}^{L}$ using GP priors.

RNA emulator



RNA emulator

$$W_1(\mathbf{x}) \sim GP(\alpha_1, \tau_1^2 k_1(\mathbf{x}, \mathbf{x}')),$$

$$W_l(\mathbf{z}) \sim GP(\alpha_l, \tau_l^2 k_l(\mathbf{z}, \mathbf{z}')), \quad l = 2, \cdots, L,$$

where $\mathbf{z} = (\mathbf{x}, y)$, and $k_1(\mathbf{x}, \mathbf{x}')$ and $k_l(\mathbf{z}, \mathbf{z}')$ are positive definite kernels.

- e.g., squared exponential kernel:

$$k_1(\mathbf{x}, \mathbf{x}') = \prod_{j=1}^{d} \exp\left(-\frac{(x_j - x_j')^2}{\theta_{1j}}\right)$$

$$k_l(\mathbf{z}, \mathbf{z}') = \exp\left(-\frac{(y - y')^2}{\theta_{ly}}\right) \prod_{j=1}^{d} \exp\left(-\frac{(x_j - x_j')^2}{\theta_{lj}}\right)$$

RNA emulator

- The observed simulations $\mathbf{y}_l$ follow a multivariate normal distribution:

$$\mathbf{y}_1 = W_1(\mathcal{X}_1) \sim \mathcal{N}_{n_1}(\alpha_1 \mathbf{1}_{n_1}, \tau_1^2 K_1(\mathcal{X}_1)), \quad \text{and}$$

$$\mathbf{y}_l = W_l(\mathcal{X}_l, f_{l-1}(\mathcal{X}_l)) \sim \mathcal{N}_{n_l}(\alpha_l \mathbf{1}_{n_l}, \tau_l^2 K_l(\mathcal{X}_l, f_{l-1}(\mathcal{X}_l))),$$

  for $l = 2, \cdots, L$.

- $\{K_1(\mathcal{X}_1)\}_{ij} = k_1(\mathbf{x}_i^{[1]}, \mathbf{x}_j^{[1]})$

- $\{K_l(\mathcal{X}_l, f_{l-1}(\mathcal{X}_l))\}_{ij} = k_l((\mathbf{x}_i^{[l]}, f_{l-1}(\mathbf{x}_i^{[l]})), (\mathbf{x}_j^{[l]}, f_{l-1}(\mathbf{x}_j^{[l]})))$

- $f_{l-1}(\mathcal{X}_l) = (\mathbf{y}_{l-1})_{1:n_l}$, because of the **nested assumption**!

Hyperparameter Estimation

- Hyperparameters $\left\{\alpha_l, \tau_l^2, \boldsymbol{\theta}_l\right\}_{l=1}^{L}$ can be estimated by maximum likelihood estimation:

$$-\frac{n_l}{2}\log(\tau_l^2) - \frac{1}{2}\log\left|K_l(\mathcal{X}_l, f_{l-1}(\mathcal{X}_l))\right| - \frac{1}{2\tau_l^2}(\mathbf{y}_l - \alpha_l \mathbf{1}_{n_l})^\top K_l(\mathcal{X}_l, f_{l-1}(\mathcal{X}_l))^{-1}(\mathbf{y}_l - \alpha_l \mathbf{1}_{n_l}).$$

Posterior of $f_L(\mathbf{x})$ for a new input $\mathbf{x}$

- Based on the properties of conditional multivariate normal distribution, it follows that

$$f_l(\mathbf{x})|\mathbf{y}_l, f_{l-1}(\mathbf{x}) \sim \mathcal{N}(\mu_l(\mathbf{x}, f_{l-1}(\mathbf{x})), \sigma_l^2(\mathbf{x}, f_{l-1}(\mathbf{x})))$$

for $l = 2, \ldots, L$ with

$$\mu_l(\mathbf{x}, f_{l-1}(\mathbf{x})) = \alpha_l \mathbf{1}_{n_l} + \mathbf{k}_l(\mathbf{x}, f_{l-1}(\mathbf{x}))^\top K_l(\mathcal{X}_l, f_{l-1}(\mathcal{X}_l))^{-1}(\mathbf{y}_l - \alpha_l \mathbf{1}_{n_l}),$$
$$\sigma_l^2(\mathbf{x}, f_{l-1}(\mathbf{x})) = \tau_l^2(1 - \mathbf{k}_l(\mathbf{x}, f_{l-1}(\mathbf{x}))^\top K_l(\mathcal{X}_l, f_{l-1}(\mathcal{X}_l))^{-1}\mathbf{k}_l(\mathbf{x}, f_{l-1}(\mathbf{x}))).$$

- Posterior of $f_L(\mathbf{x})$ at a new input $\mathbf{x}$: $p(f_L(\mathbf{x})|\mathbf{y}_1, \ldots, \mathbf{y}_L) =$

$$= \int \cdots \int p(f_L(\mathbf{x})|\mathbf{y}_L, f_{l-1}(\mathbf{x}))p(f_{L-1}(\mathbf{x})|\mathbf{y}_{L-1}, f_{L-2}(\mathbf{x})) \cdots p(f_1(\mathbf{x})|\mathbf{y}_1)\mathrm{d}(f_{l-1}(\mathbf{x})) \ldots \mathrm{d}(f_1(\mathbf{x})).$$

Remark on NARGP by Perdikaris et al. (2017)

- **Nonlinear Auto-Regressive GP** (NARGP) proposed by Perdikaris et al. (2017) also adopts the recursive scheme, but they rely on

  1. Additive form of the kernel:

$$K_l(\mathbf{z}, \mathbf{z}') = K_{l1}(\mathbf{x}, \mathbf{x}')K_{l2}(f_{l-1}(\mathbf{x}), f_{l-1}(\mathbf{x}')) + K_{l3}(\mathbf{x}, \mathbf{x}'),$$

  2. Monte Carlo integration for the intractable posterior distribution:

$$p(f_L(\mathbf{x})|\mathbf{y}_1, \dots, \mathbf{y}_L)$$
$$= \int \cdots \int p(f_L(\mathbf{x})|\mathbf{y}_L, f_{L-1}(\mathbf{x}))p(f_{L-1}(\mathbf{x})|\mathbf{y}_{L-1}, f_{L-2}(\mathbf{x})) \cdots p(f_1(\mathbf{x})|\mathbf{y}_1)\mathrm{d}(f_{L-1}(\mathbf{x})) \dots \mathrm{d}(f_1(\mathbf{x})).$$

RNA emulator

In contrast...

- RNA emulator adopts the natural form of popular kernel choices:

$$K_1(\mathbf{x}, \mathbf{x}') = \prod_{j=1}^{d} \phi(x_j, x_j'; \theta_{1j}),$$

$$K_l(\mathbf{z}, \mathbf{z}') = \phi(y, y'; \theta_{ly}) \prod_{j=1}^{d} \phi(x_j, x_j'; \theta_{lj}), \quad l = 2, \cdots, L,$$

- With these kernel choices, RNA emulator has the closed-form posterior mean and variance of $f_l(\mathbf{x})$ in a recursive fashion!

The closed-form expression of RNA emulator

Proposition 1: The closed-form expressions

- Under the **squared exponential kernel**, the posterior mean and variance can be obtained as follows:

$$\mu_l^*(\mathbf{x}) := \mathbb{E}[f_l(\mathbf{x})|\mathbf{y}_1,\ldots,\mathbf{y}_l]$$
$$= \alpha_l + \sum_{i=1}^{n_l} r_i \prod_{j=1}^{d} \exp\left(-\frac{(x_j - x_{ij}^{[l]})^2}{\theta_{lj}}\right) \frac{1}{\sqrt{1+2\frac{\sigma_{l-1}^{*2}(\mathbf{x})}{\theta_{ly}}}} \exp\left(-\frac{(y_i^{[l-1]} - \mu_{l-1}^*(\mathbf{x}))^2}{\theta_{ly}+2\sigma_{l-1}^{*2}(\mathbf{x})}\right),$$

$$\sigma_l^{*2}(\mathbf{x}) := \mathbb{V}[f_l(x)|\mathbf{y}_1,\ldots,\mathbf{y}_l] = \tau_l^2 - (\mu_l^*(\mathbf{x}) - \alpha_l)^2 +$$
$$\left(\sum_{i,k=1}^{n_l} \zeta_{ik}\left(r_i r_k - \tau_l^2(\mathbf{K}_l^{-1})_{ik}\right) \prod_{j=1}^{d} \exp\left(-\frac{(x_j - x_{ij}^{[l]})^2 + (x_j - x_{kj}^{[l]})^2}{\theta_{lj}}\right)\right).$$

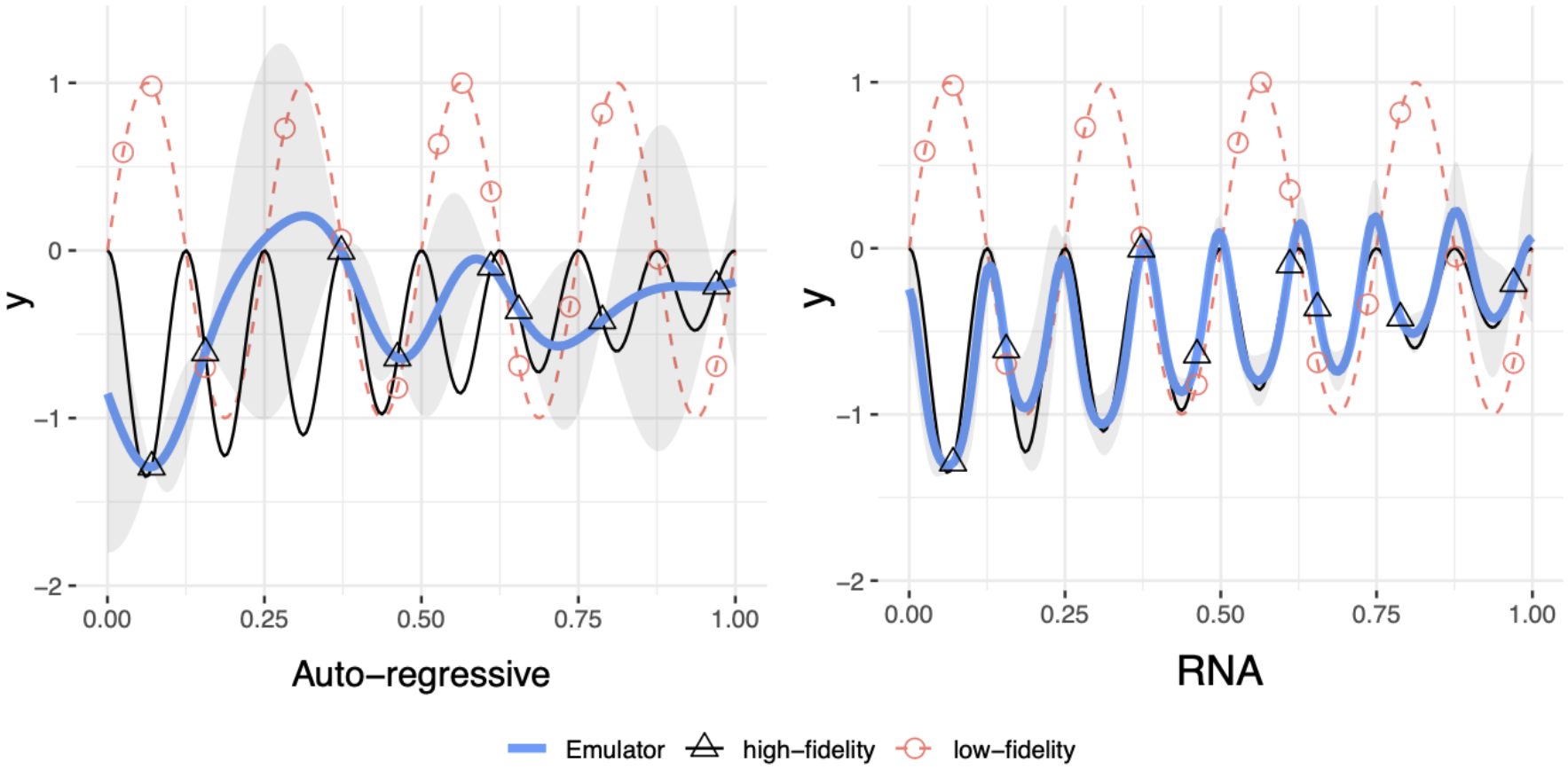
The closed-form expression of RNA emulator

Proposition 2: Interpolation property

The RNA emulator exhibits interpolation property. That is, $\mu_l^*(\mathcal{X}_l) = \mathbf{y}_l$, and $\sigma_l^{*2}(\mathcal{X}_l) = \mathbf{0}_{n_l}$ for $l = 1, \ldots, L$.

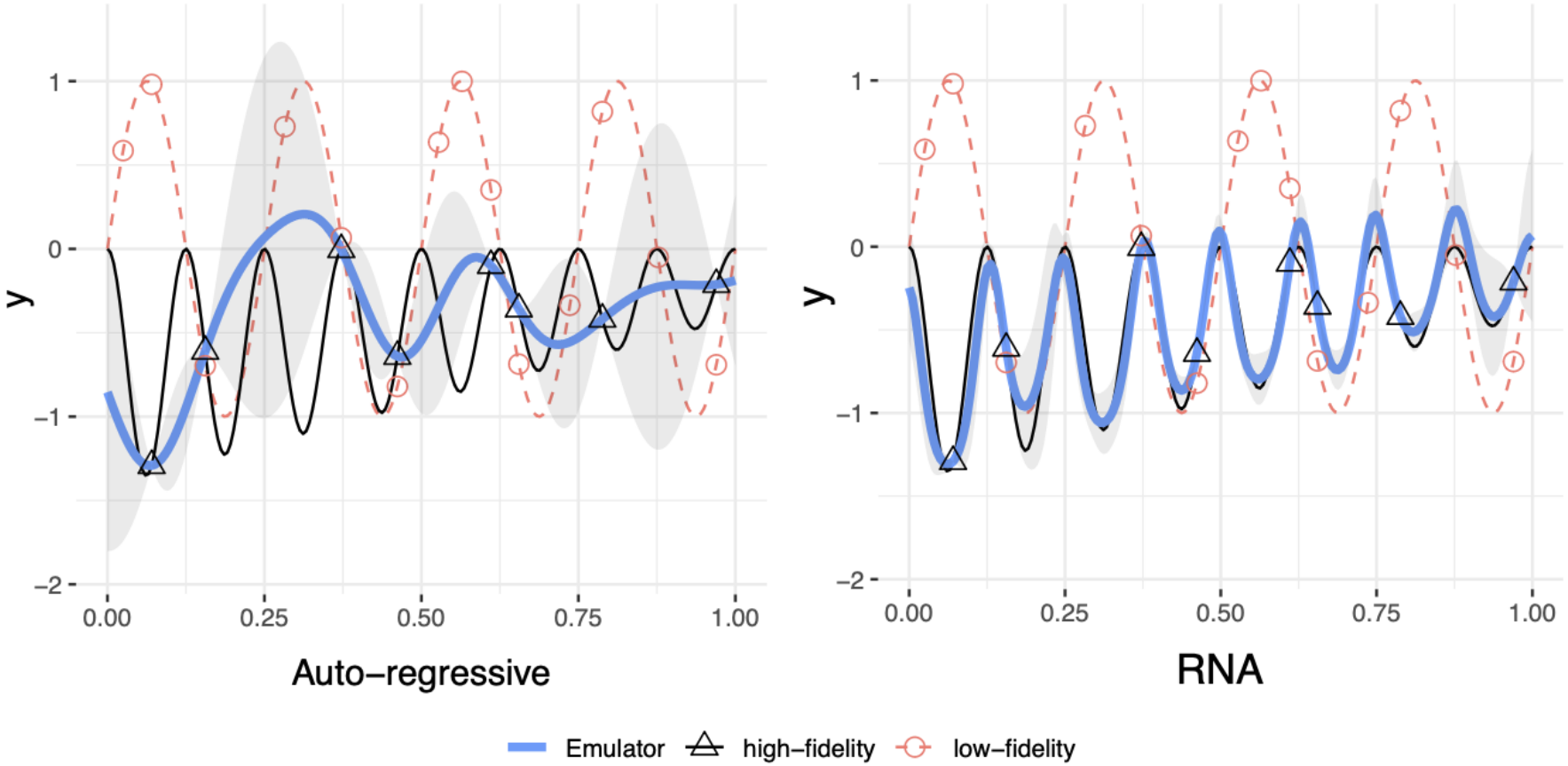
- The closed-form expressions can be derived under a Matérn kernel with the smoothness parameter $\nu = 1.5$ and $\nu = 2.5$ as well.
- Adopt the moment matching method to approximate the posterior distribution.

RNA emulator



An illustration of the posterior distribution with an example from Perdikaris et al. (2017), where $n_1 = 13$, $n_2 = 8$, $f_1(x) = \sin(8\pi x)$, and $f_2(x) = (x - \sqrt{2})f_1^2(x)$.

After emulating...



However, the emulator still holds the **uncertainty** in some region!

Active Learning

- Active learning
  - is also known as sequential design,
  - sequentially searches for and acquires new data points at optimal location by a given criterion,
  - aims to achieve enhanced accuracy while managing the limited resources,
  - is crucial for computationally expensive simulations.
- Well-established for single-fidelity GP emulators, but research for multi-fidelity computer simulations is scarce and more challenging.

Active Learning for RNA emulator

- In multi-fidelity simulation, active learning requires
  - identifying optimal input locations,
  - identifying fidelity levels,
  - accounting for the respective simulation costs simultaneously.
- Four active learning strategies (ALD, ALM, ALC, and ALMC) for RNA emulator will be introduced.
- The nested structure assumption implies that, in order to run the simulation $f_l(\mathbf{x}_{n_l+1}^{[l]})$, we need to run $f_k(\mathbf{x}_{n_k+1}^{[k]})$ with $\mathbf{x}_{n_k+1}^{[k]} = \mathbf{x}_{n_l+1}^{[l]}$ for all $1 \leq k \leq l$.

Active Learning Decomposition (ALD)

- Select the next point that maximizes the posterior predictive variance $\sigma_L^{*2}(x)$.

- We can rewrite the posterior variance, for example $L = 2$:

$$\begin{aligned}\sigma_2^{*2}(x) &= \mathbb{V}\left[\mathbb{E}[f_2(x)|f_1(x), \mathbf{y}_1, \mathbf{y}_2]\right] + \mathbb{E}\left[\mathbb{V}[f_2(x)|f_1(x), \mathbf{y}_1, \mathbf{y}_2]\right] \\ &:= V_1(x) + V_2(x)\end{aligned}$$

- To account for the simulation cost C_l, choose the next point $\mathbf{x}_{n_l+1}^{[l]}$ at level l by maximizing ALD criterion:

$$(l, \mathbf{x}_{n_l+1}^{[l]}) = \underset{k \in \{1,2\}; \mathbf{x} \in \Omega}{\arg\max} \frac{V_k(\mathbf{x})}{\sum_{j=1}^{k} C_j}.$$

- The closed-form expression facilitates the computation of ALD.

Active Learning MacKay (ALM)

- Select the next point that maximizes the posterior predictive variance $\sigma_l^{*2}(x)$ (MacKay, 1992).

- To account for the simulation cost C_l, choose the next point $\mathbf{x}_{n_l+1}^{[l]}$ at level l by maximizing ALM criterion:

$$(l, \mathbf{x}_{n_l+1}^{[l]}) = \underset{k \in \{1,\ldots,L\}; \mathbf{x} \in \Omega}{\arg\max} \frac{\sigma_k^{*2}(\mathbf{x})}{\sum_{j=1}^{k} C_j}.$$

- The closed-form expression facilitates the computation of ALM.

Active Learning Cohn (ALC)

- Select an input location that maximizes the variance reduction **across the entire input space** after running this selected simulation (Cohn, 1993).

- Choose the next point $\mathbf{x}_{n_l+1}$ at fidelity level l by maximizing the ALC criterion:

$$(l, \mathbf{x}_{n_l+1}^{[l]}) = \underset{k \in \{1,\ldots,L\}; \mathbf{x} \in \Omega}{\arg\max} \frac{\Delta\sigma_L^2(k, \mathbf{x})}{\sum_{j=1}^{k} C_j},$$

where $\Delta\sigma_L^2(k, \mathbf{x}) = \int_\Omega \left\{\sigma_L^{*2}(\boldsymbol{\xi}) - \tilde{\sigma}_L^{*2}(\boldsymbol{\xi}; k, \mathbf{x})\right\} \mathrm{d}\boldsymbol{\xi}$ is the **average reduction in variance** (of the highest-fidelity emulator) with a choice of the fidelity level k and the input location $\mathbf{x}$.

Two-step approach: ALMC

- Inspired by Le Gratiet and Cannamela (2015), consider the combination of ALM and ALC.
- First, the optimal input location is selected by maximizing the posterior predictive variance of the highest fidelity emulator:

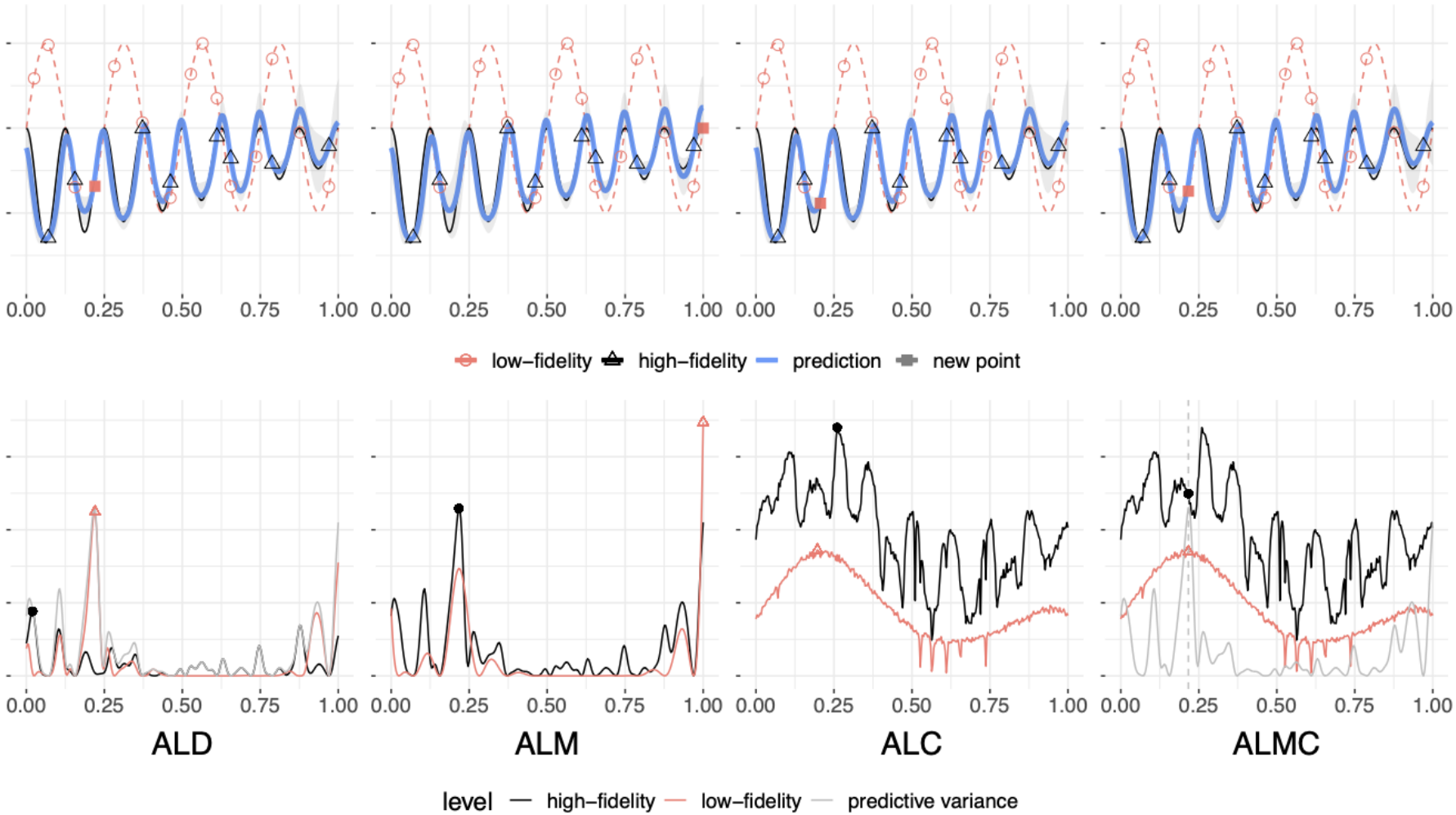
$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\Omega} \sigma_L^{*2}(\mathbf{x}).$$

- Then, the ALC criterion determines the fidelity level with the identified input location:

$$l = \arg\max_{k\in\{1,\ldots,L\}} \frac{\Delta\sigma_L^2(k,\mathbf{x}^*)}{\sum_{j=1}^{k} C_j},$$

which aims to maximize the ratio between the variance reduction and the associated simulation cost.

Demonstration



Numerical studies 1: Emulation performance

- 6 different functions with 2 or 3 levels of fidelity.
- Compare proposed emulator `RNAmf` with `Cokriging` (Le Gratiet and Garnier, 2014) and `NARGP` (Perdikaris et al., 2017).
- 100 repetitions with $n_{\text{test}} = 1000$ random test input locations generated by space-filling designs.
- Evaluate the prediction performance based on two criteria:
  - the root-mean-square error (RMSE)
  - continuous rank probability score (CRPS) (Gneiting and Raftery, 2007)

Numerical studies 1: Emulation performance

- Two-level Perdikaris function (Perdikaris et al., 2017),

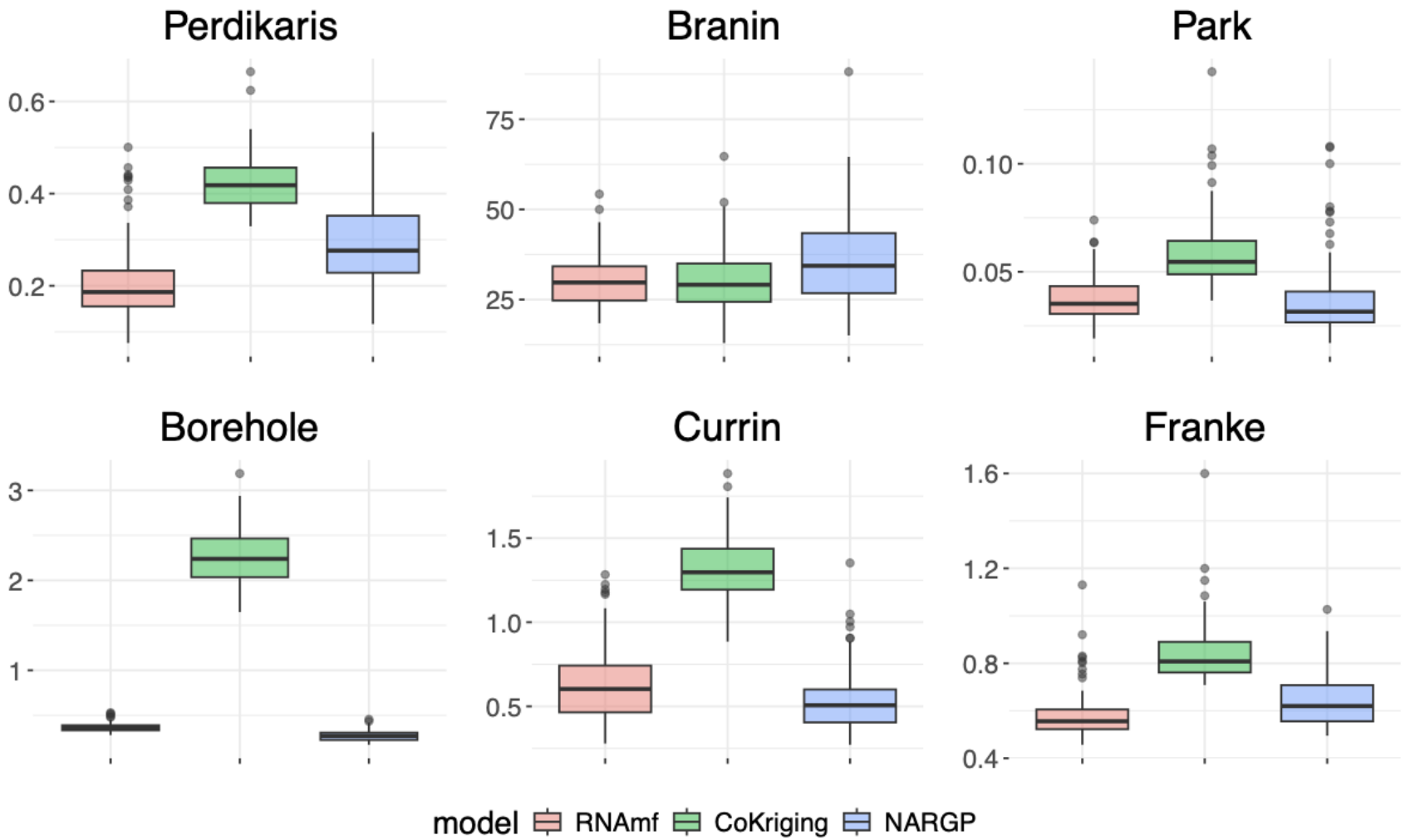
$$\begin{cases} f_1(x) = \sin(8\pi x) \\ f_2(x) = (x - \sqrt{2}) f_1^2(x) \end{cases} \text{for } x \in [0, 1],$$

- Two-level Park function (Park, 1991; Xiong et al., 2013),

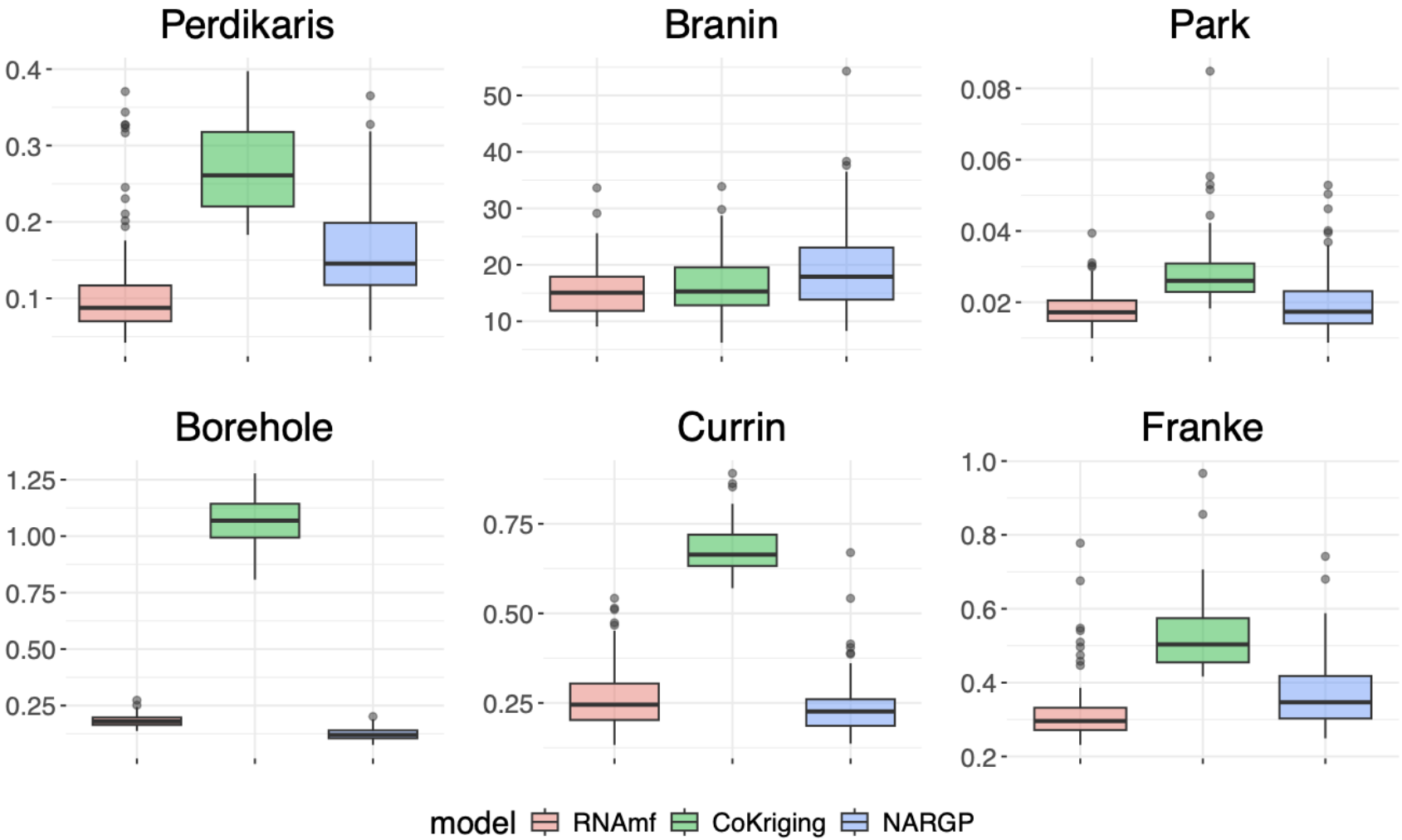
$$\begin{cases} f_1(\mathbf{x}) = f_2(\mathbf{x}) + \frac{\sin(x_1)}{10} f_2(\mathbf{x}) - 2x_1 + x_2^2 + x_3^2 + 0.5 \\ f_2(\mathbf{x}) = \frac{x_1}{2} \left[\sqrt{1 + (x_2 + x_3^2) \frac{x_4}{x_1^2}} - 1 \right] + (x_1 + 3x_4) \exp\left(1 + \sin(x_3)\right) \end{cases} \text{for } \mathbf{x} \in [0, 1]^4.$$

	Perdikaris	Branin	Park	Borehole	Currin	Franke
d	1	2	4	8	2	2
n_1	13	20	40	60	20	20
n_2	8	15	20	30	10	15
n_3		10				10

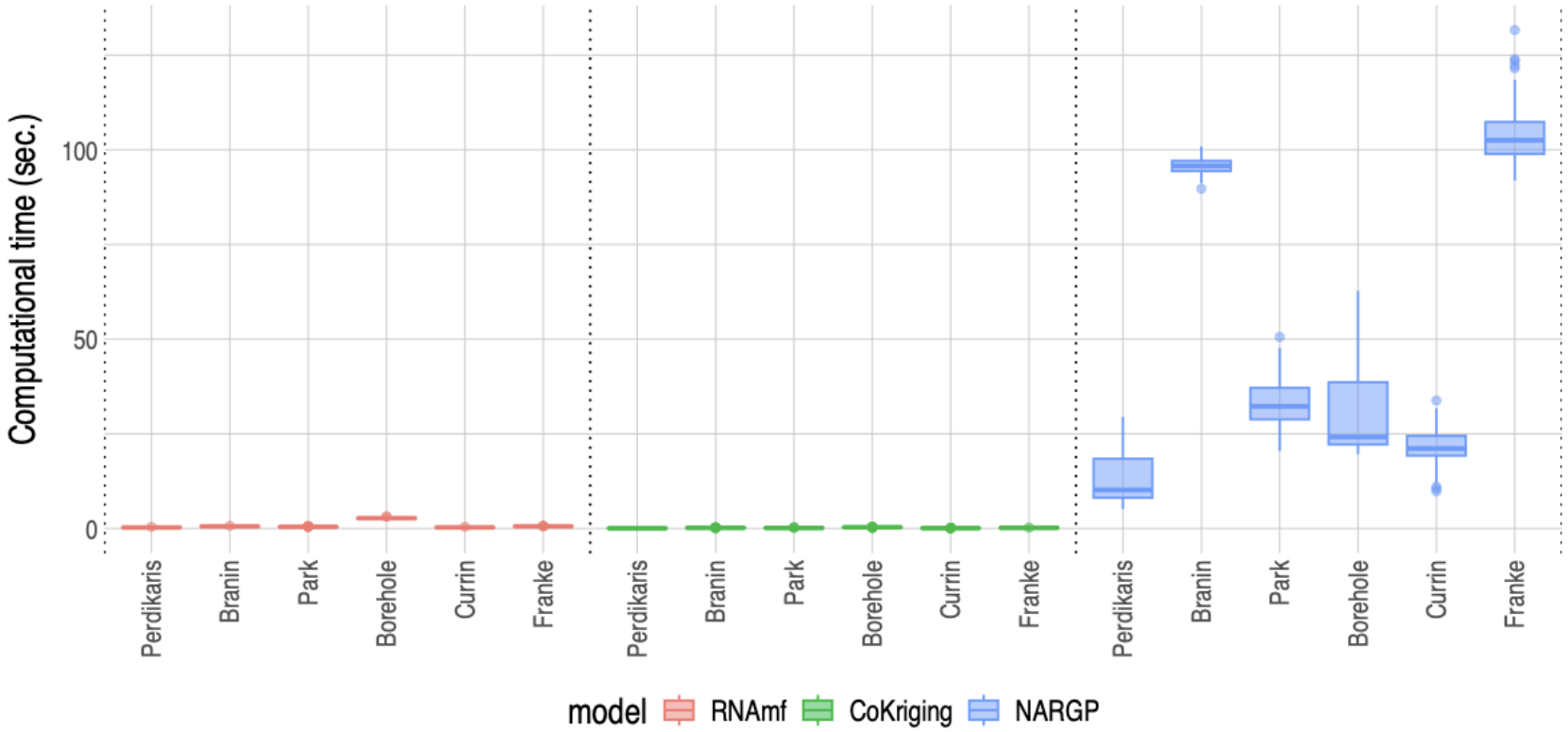
Numerical studies 1: RMSE



Numerical studies 1: CRPS



Numerical studies 1: Computational time

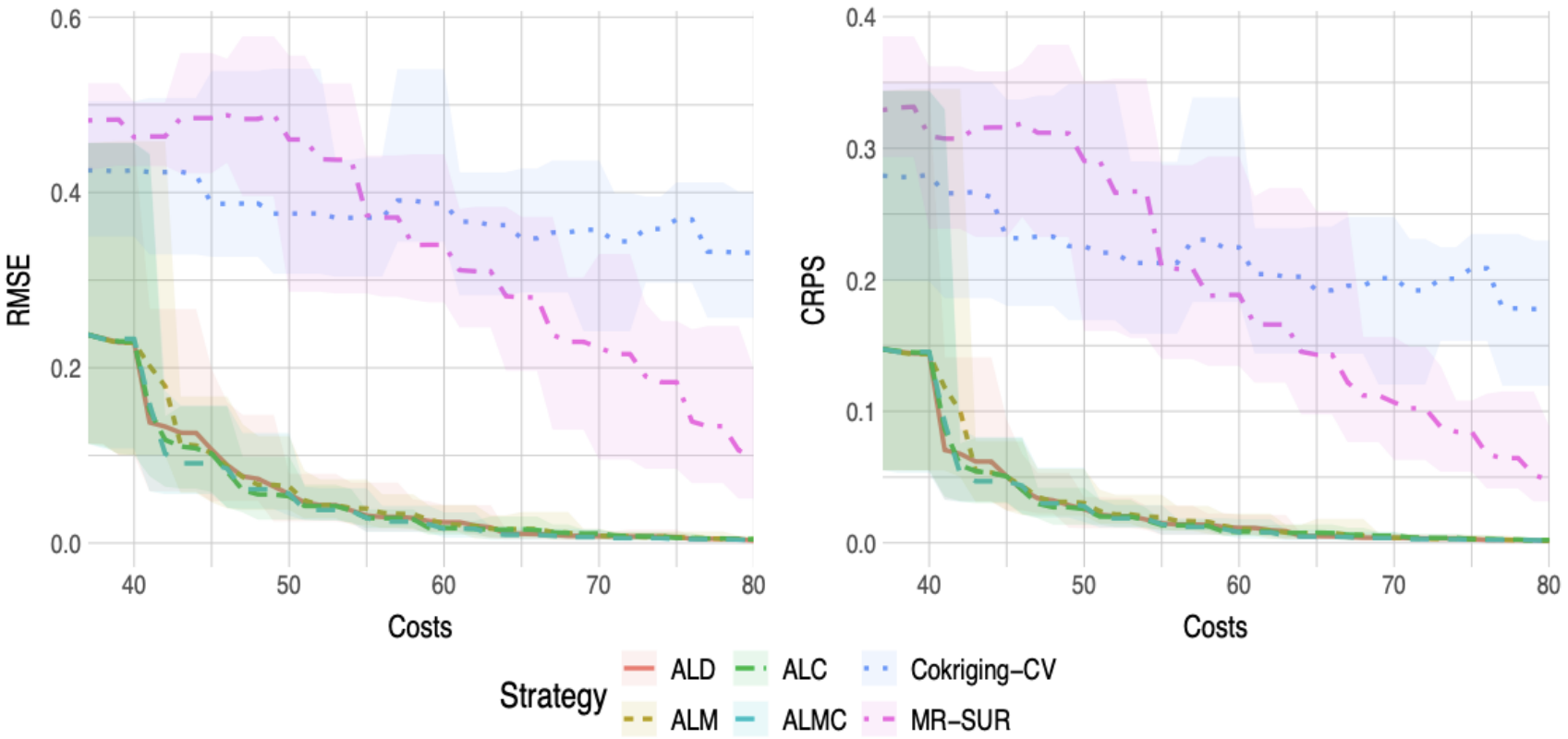


Computational time of six synthetic examples across 100 repetitions.

Numerical studies 2: Active learning performance

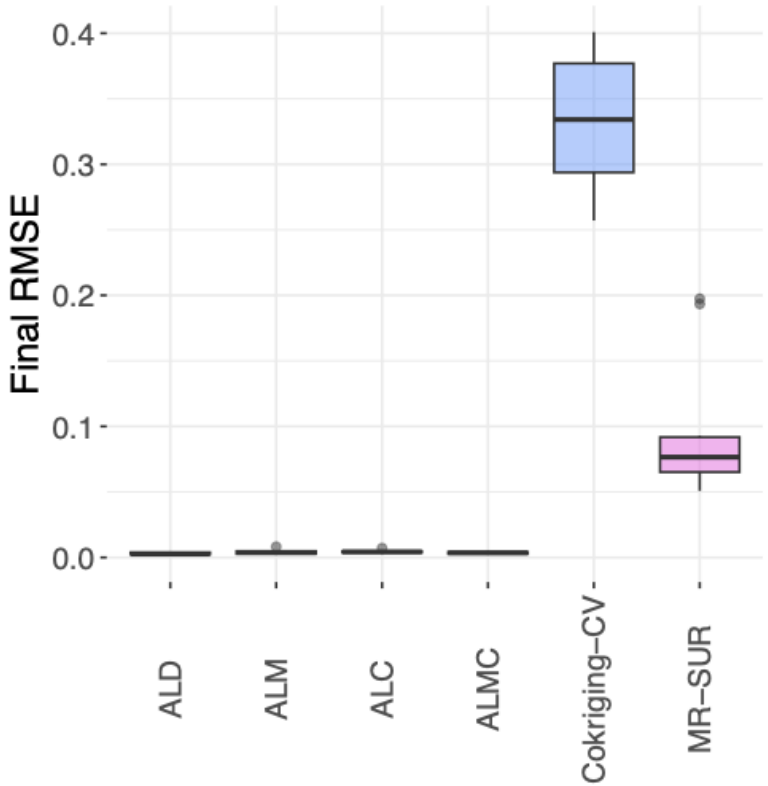
- Perdikaris function (1-dim).
- Compare four proposed strategies `ALD`, `ALM`, `ALC`, and `ALMC`, with
  - a cokriging-based sequential design (`CoKriging-CV`) (Le Gratiet and Cannamela, 2015)
  - a sequential design maximizing the rate of stepwise uncertainty reduction using the AR model (`MR-SUR`) (Stroh et al., 2022)
- Simulation costs of low- and high-fidelity simulators are $C_1 = 1$ and $C_2 = 3$.
- Total simulation budget of $C_{\text{total}} = 80$.
- 10 repetitions with different initial designs.

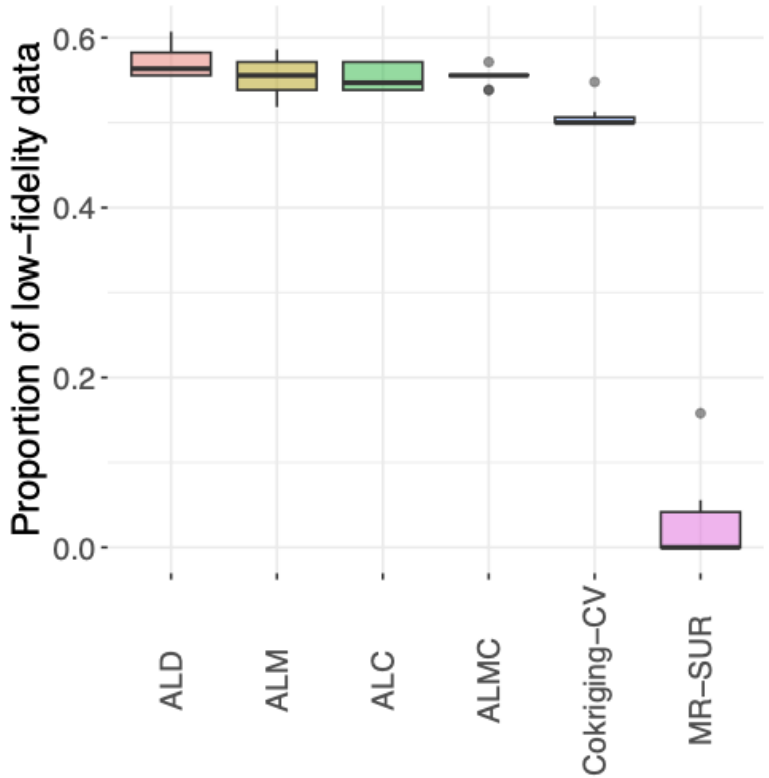
Numerical studies 2: Active learning performance



RMSE and CRPS for the Perdikaris function with respect to the simulation cost.

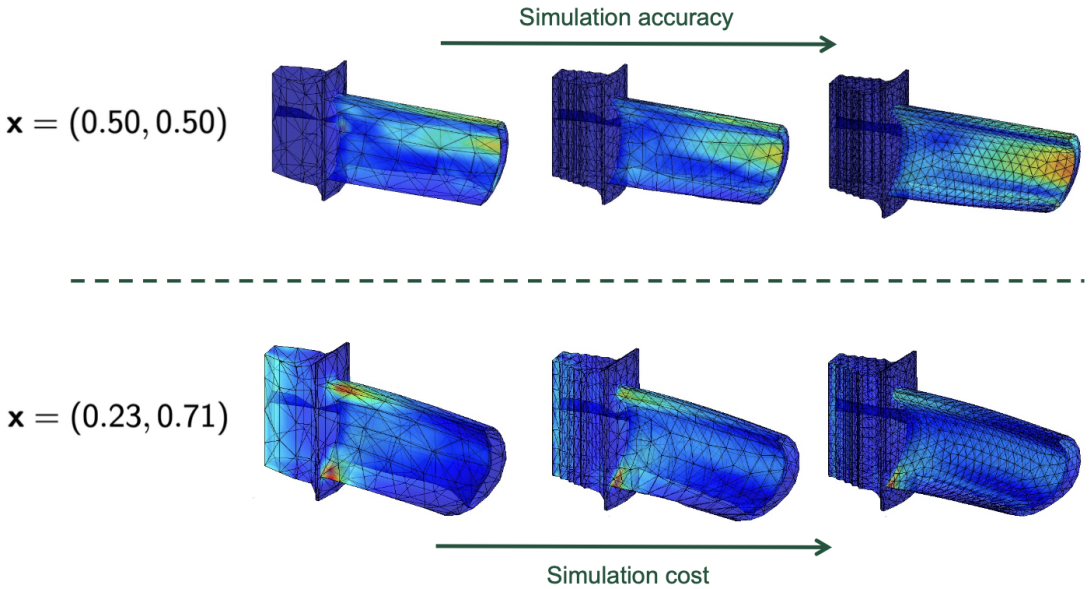
Numerical studies 2: Active learning performance





Final RMSE (left) and proportion of AL acquisitions choosing low-fidelity data (right).

Revisit motivating example

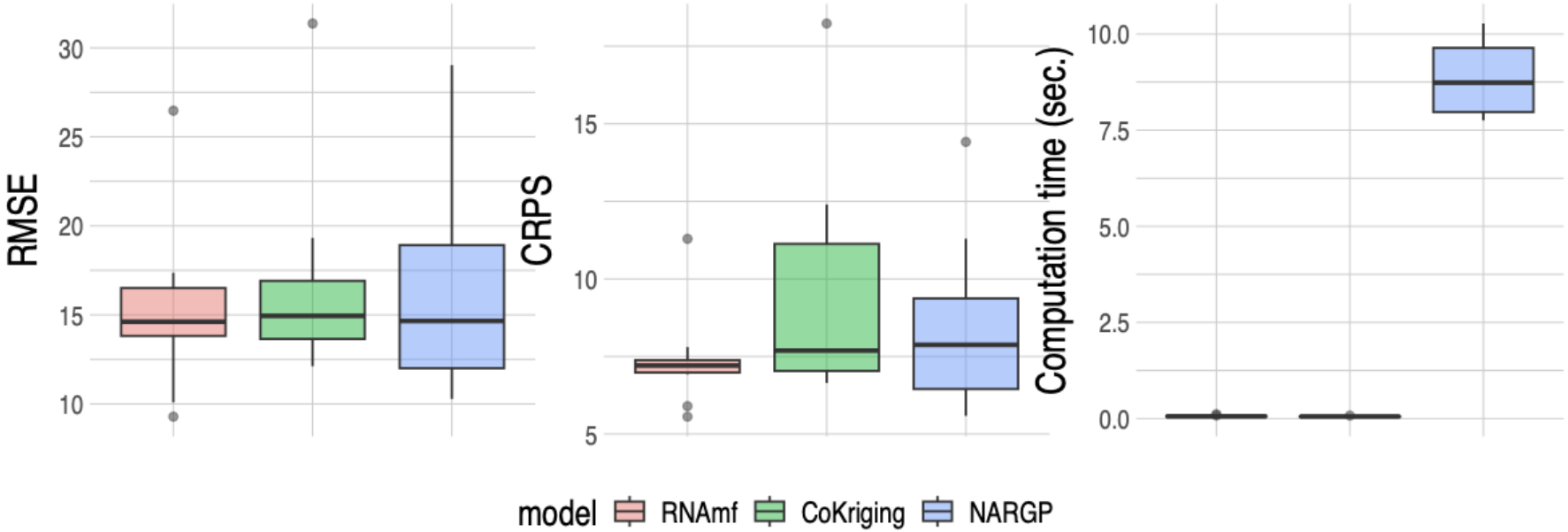


- **Input**: $\mathbf{x} = (x_1, x_2) = (\text{pressure}, \text{suction}) \in \Omega = [0.25, 0.75]^2$
- **Output**: $f(\mathbf{x})$: **maximum** of the thermal stress profile

Revisit motivating example

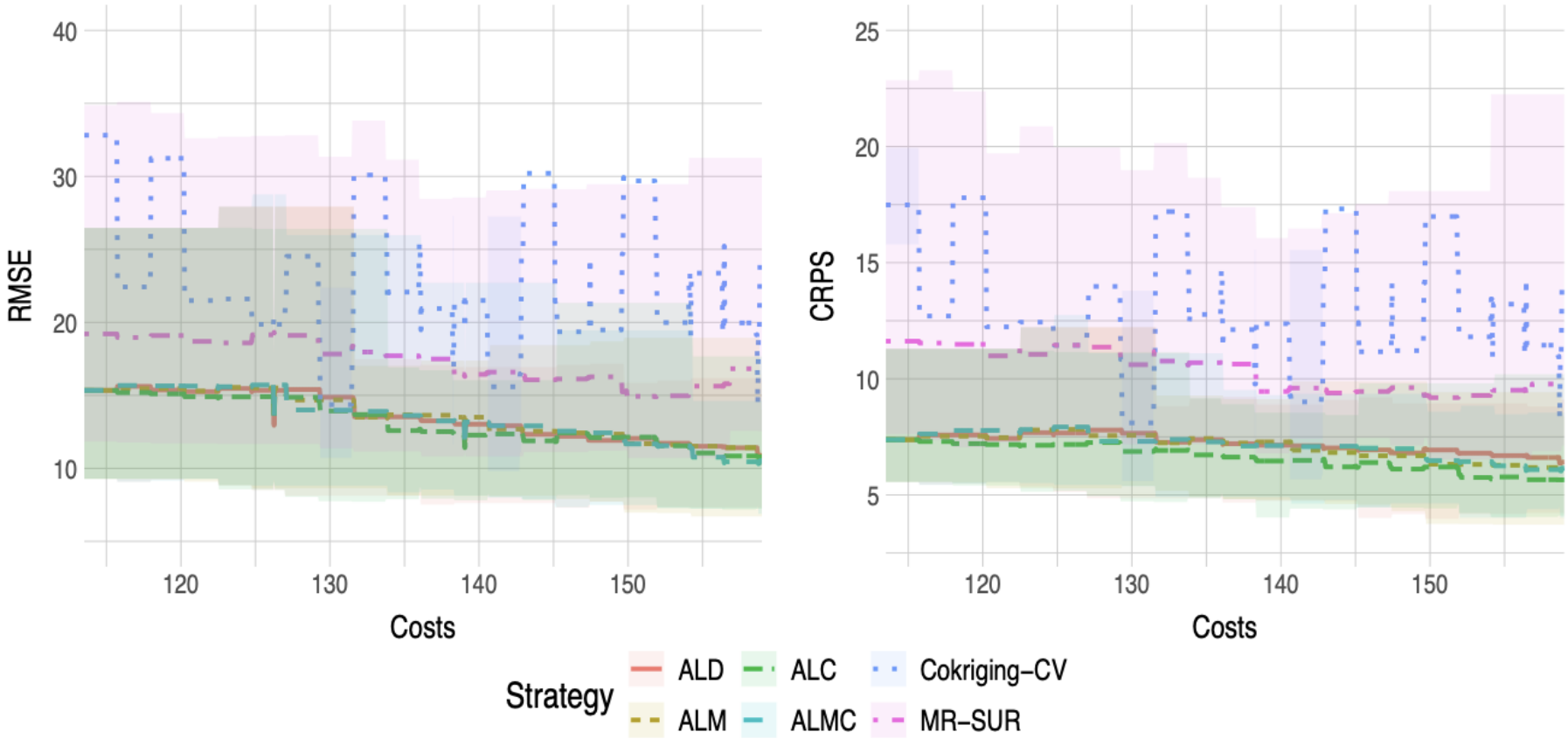
- Perform the finite element simulations with $n_1 = 20$ and $n_2 = 10$.
- The simulation time of the finite element simulations, which are respectively $C_1 = 2.25$ and $C_2 = 6.85$ (seconds) will be used for active learning.
- 10 repetitions with $n_{\text{test}} = 100$ random test input locations generated by a space-filling design.
- Perform finite element simulations using the Partial Differential Equation Toolbox in MATLAB.

Blade: Emulation performance



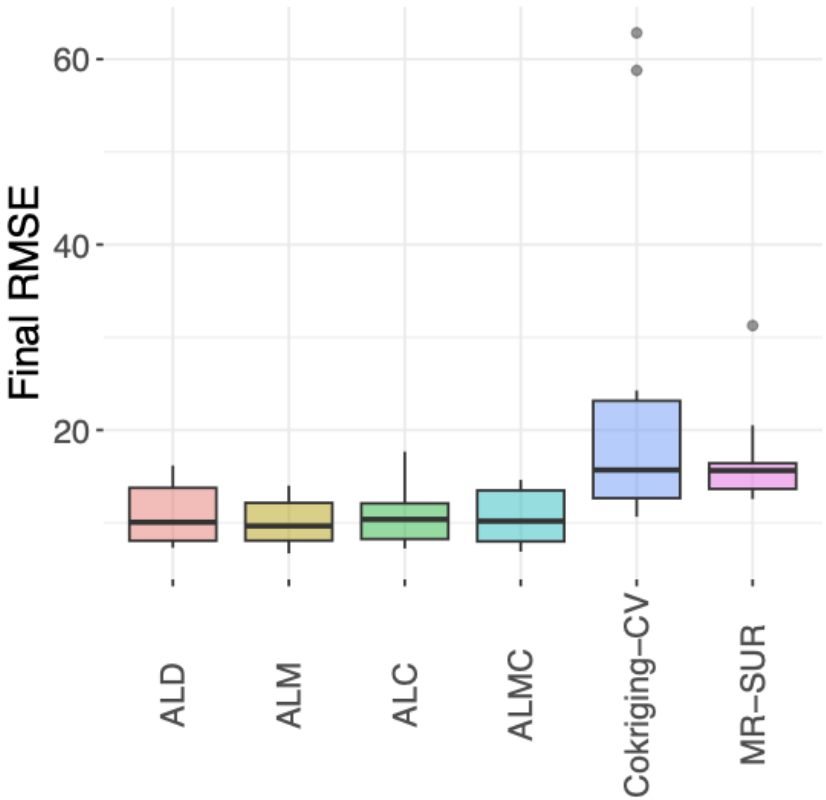
RMSE, CRPS, and computation time across 10 repetitions in the turbine blade application.

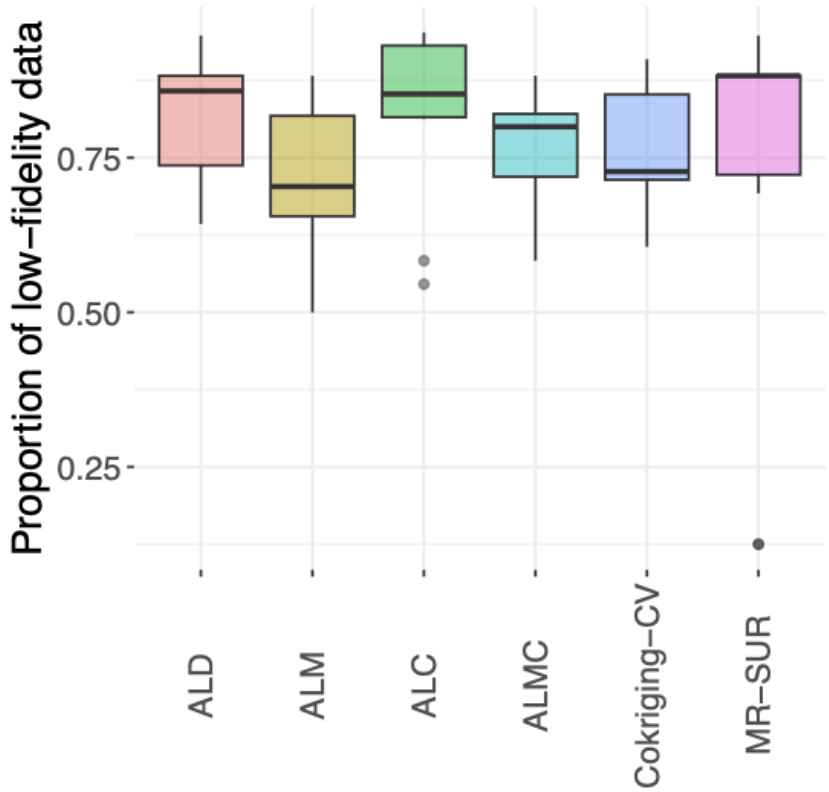
Blade: Active learning performance



RMSE and CRPS for the Blade simulations with respect to the cost.

Blade: Active learning performance





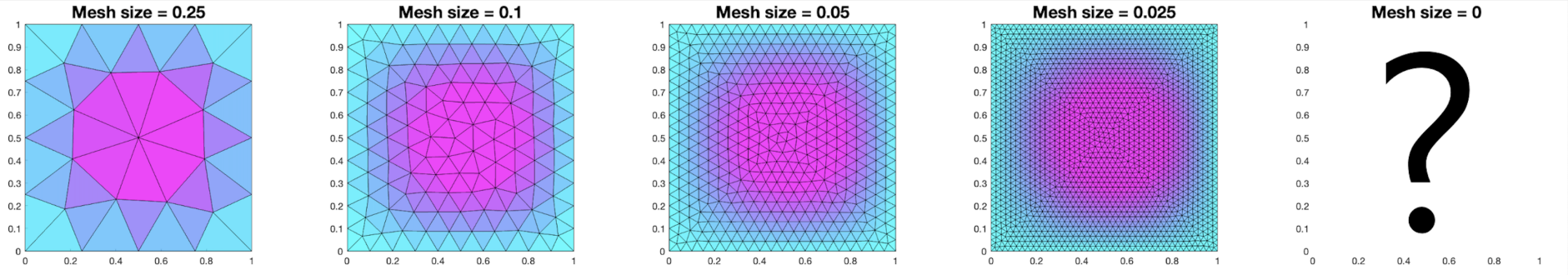
Final RMSE (left) and proportion of AL acquisitions choosing low-fidelity data (right).

Diffusion Non-Additive (DNA) model and Non-nested design

Heo, J., Boutelet-Smith, R., Wang, W., and Sung, C.-L. (2026+).

Diffusion non-additive model for multi-fidelity simulations with tunable precision. *arXiv:2506.08328*.

Remaining Questions



FEM solutions of Poisson's equation with different mesh configurations.

- Q: Can we extrapolate to the **exact solution** of the finite element simulations (i.e., mesh size $\rightarrow 0$)?
- Q: Can we account for the mesh size and its **interaction** with the input variables?

Problem Reformulation

- Let $f_l(\mathbf{x}) := f(t_l, \mathbf{x})$ with tuning parameter t_l at fidelity level $l = 1, \ldots, L$.
- t_l denote the tuning parameter (e.g., mesh sizes) at fidelity level l with $t_1 > t_2 > \ldots > t_L > 0$.
- Most existing works (including AR model and RNA model) focus on the highest available fidelity simulator $f_L(\mathbf{x})$!

Existing Methods

- Non-Stationary Model (Tuo et al., 2014):
  The response variable at input location $\mathbf{x} \in \mathcal{D}$ with mesh size $t \in \mathcal{T}$ is assumed to be:

$$f(\mathbf{x}, t) = \underbrace{\varphi(\mathbf{x})}_{\text{exact solution}} + \underbrace{\delta(\mathbf{x}, t)}_{\text{error}},$$

  where $\varphi(\mathbf{x}) := f(\mathbf{x}, 0)$, and $\delta(\mathbf{x}, t)$ are realizations of two mutually independent GPs, respectively.

- Since $f(\mathbf{x}, t)$ must equal to the exact solution $\varphi(\mathbf{x})$ as $t \to 0$, we need δ to satisfy $\delta(\mathbf{x}, t) \xrightarrow{t \to 0} 0$, for all $\mathbf{x}$.

Existing Methods

- Tuo et al. (2014) considers a Brownian motion kernel $K_{BM}(t, t') = min(t, t')^l$
- Ji et al. (2024) extends Tuo et al. (2014) to multiple tuning parameters.
- Boutelet-Smith and Sung (2026) extends Tuo et al. (2014) to the fractional Brownian motion:

$$K_H(t, t') = \left\{ \frac{1}{2} \left(t^{2H} + (t')^{2H} + |t - t'|^{2H} \right) \right\}^{\frac{l}{2H}}, \quad 0 \leq H \leq 1.$$

Diffusion Non-Additive (DNA) Model

Diffusion Non-Additive (DNA) Model

The response variable at input location $\mathbf{x}$ with mesh size t_l is assumed to be:

$$f(\mathbf{x}, t_1) = W_1(\mathbf{x}),$$
$$f(\mathbf{x}, t_l) = W(t_l, \mathbf{x}, f(\mathbf{x}, t_{l-1})), \quad l = 2, \ldots, L.$$

where $t_1 > t_2 > \cdots > t_L > 0$ and W is a realization of a GP.

Remark 1: The relationships between fidelity levels are no longer additive.

Remark 2: The term "diffusion" is inspired by its analogy in generative models.

Nonseparable Kernel

- The **mean function** of W is assumed to be a constant α.
- The **covariance function** of W is assumed to be a nonseparable kernel function (Cressie and Huang, 1999; Gneiting, 2002):

$$K\left((t,\mathbf{x},y),(t',\mathbf{x}',y')\right) = \left(\frac{(t-t')^2}{\theta_t}+1\right)^{-\left(\frac{\beta(d+1)}{2}+\delta\right)} \times \exp\left(-\left(\frac{(t-t')^2}{\theta_t}+1\right)^{-\beta}\left[\frac{(y-y')^2}{\theta_y}+\sum_{j=1}^{d}\frac{(x_j-x_j')^2}{\theta_j}\right]\right)$$

- $0 \leq \beta \leq 1$ allows for the interaction of $(\mathbf{x}, y)$ and t.
- When $\beta = 0$, it becomes separable.
- Parameters can be estimated via MLE.

The closed-form expression of DNA model

Proposition 3: The closed-form expressions

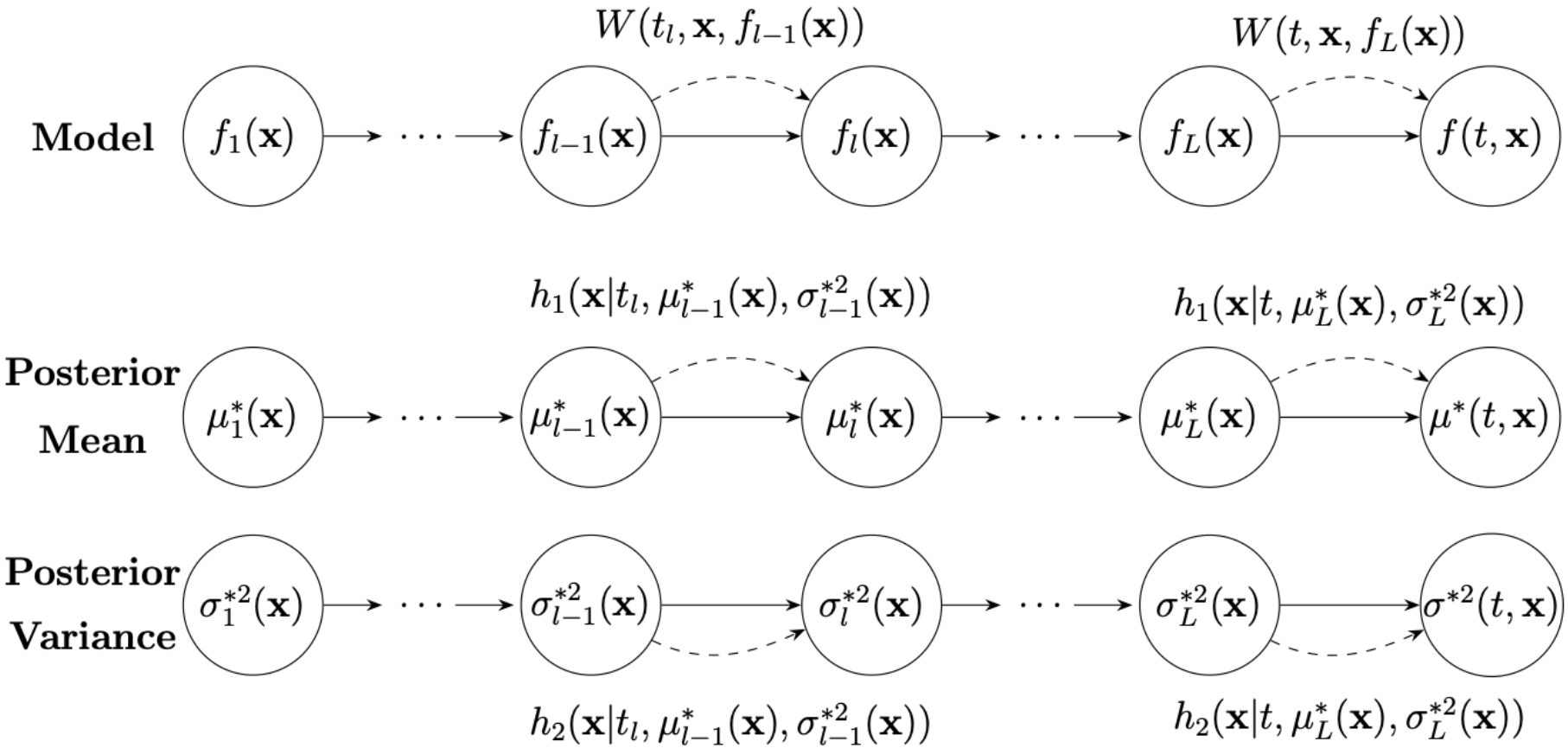
- Suppose that the design points are nested. The posterior mean and variance of $f(\mathbf{x}, t)$ given the data Y_N for $0 \leq t < t_L$ can be expressed in a recursive fashion:

$$\mu^*(t, \mathbf{x}) = \mathbb{E}[f(\mathbf{x}, t) \mid Y_N] = h_1(\mathbf{x} \mid t, \mu_L^*(\mathbf{x}), \sigma_L^{*2}(\mathbf{x})),$$

$$\sigma^{*2}(t, \mathbf{x}) = \mathbb{V}[f(\mathbf{x}, t) \mid Y_N] = h_2(\mathbf{x} \mid t, \mu_L^*(\mathbf{x}), \sigma_L^{*2}(\mathbf{x})),$$

where $h_1(\mathbf{x} \mid t, \mu, \sigma^2)$ and $h_2(\mathbf{x} \mid t, \mu, \sigma^2)$ both have closed-form expressions.

Diffusion Representation



The Markov chain representation of the hierarchical modeling structure.

Convergence Error Bound

Theorem 1: Graph-based Error Bound

Under standard regularity and Lipschitz conditions, the pointwise error satisfies, for all $(t, x) \in \Omega$, for any $a > 0$, there exists a constant C and c_X such that:

$$
\begin{aligned}
|f(t,x) - \hat{f}(t,x)| \leq\ & (B_{L+1} + M_{L+1})(t_L - t) \\
& + \sum_{l=2}^{L} \left(\prod_{s=l}^{L} M_{s+1} \right) C \left(\sqrt{1 + B_{l-1}^2}\, c_X \right)^a n_l^{-a/d} \|g_l\|_{\mathcal{N}_{K_l}(M_{l-1})} \\
& + \left(\prod_{s=1}^{L} M_{s+1} \right) C c_X^a n_1^{-a/d} \|g_1\|_{\mathcal{N}_{K_1}(X)}.
\end{aligned}
$$

- The total prediction error decomposes into three components: Extrapolation error + Intermediate errors + Base error.

Optimal Sample Size

Theorem 2: Optimal Sample Size

Suppose $\text{cost}_l = t_l^{-\gamma}$, and assume $m = \max_l\{M_l\}$. Then

$$n_l = \frac{B}{\sum_{k=1}^{L} m^{\frac{(L-k+1)d}{a+d}} \cdot t_k^{-\frac{a\gamma}{a+d}}} \left(m^{l-L-1} t_l^{-\gamma}\right)^{-\frac{d}{a+d}},$$

where B is the total budget.

- Provides theoretical guidance on how to optimally allocate a computational budget across different fidelities.

Numerical studies: Emulation performance

- 4 different functions with 5 or 6 levels of fidelity.
- Compare proposed model `DNAmf` with `CoKriging` (Le Gratiet and Garnier, 2014), `NARGP` (Perdikaris et al., 2017), `RNAmf` (Heo and Sung, 2024), `BM` (Tuo et al., 2014), and `FBM` (Boutelet-Smith and Sung, 2026).
- `CoKriging`, `NARGP`, and `RNAmf` emulate $f(t_L, \mathbf{x})$.
- `BM`, `FBM`, and `DNAmf` target $f(0, \mathbf{x})$.
- $t_l = ct_{l-1}$, and $n_{l-1} = \lfloor c^{-\gamma} n_l \rceil$, where $c = 0.7$, and $\gamma = 2$.
- 100 repetitions with $n_{\text{test}} = 100$ random test input locations.
- RMSE & CRPS.

Numerical studies: Emulation performance

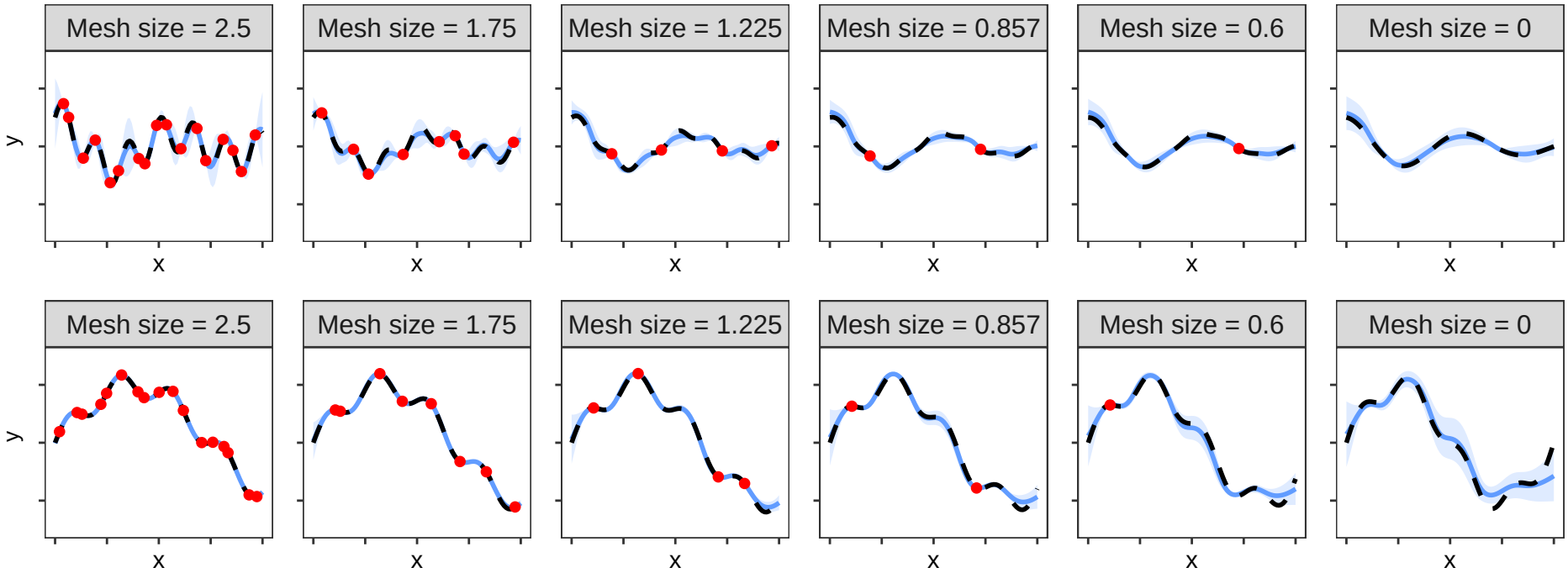
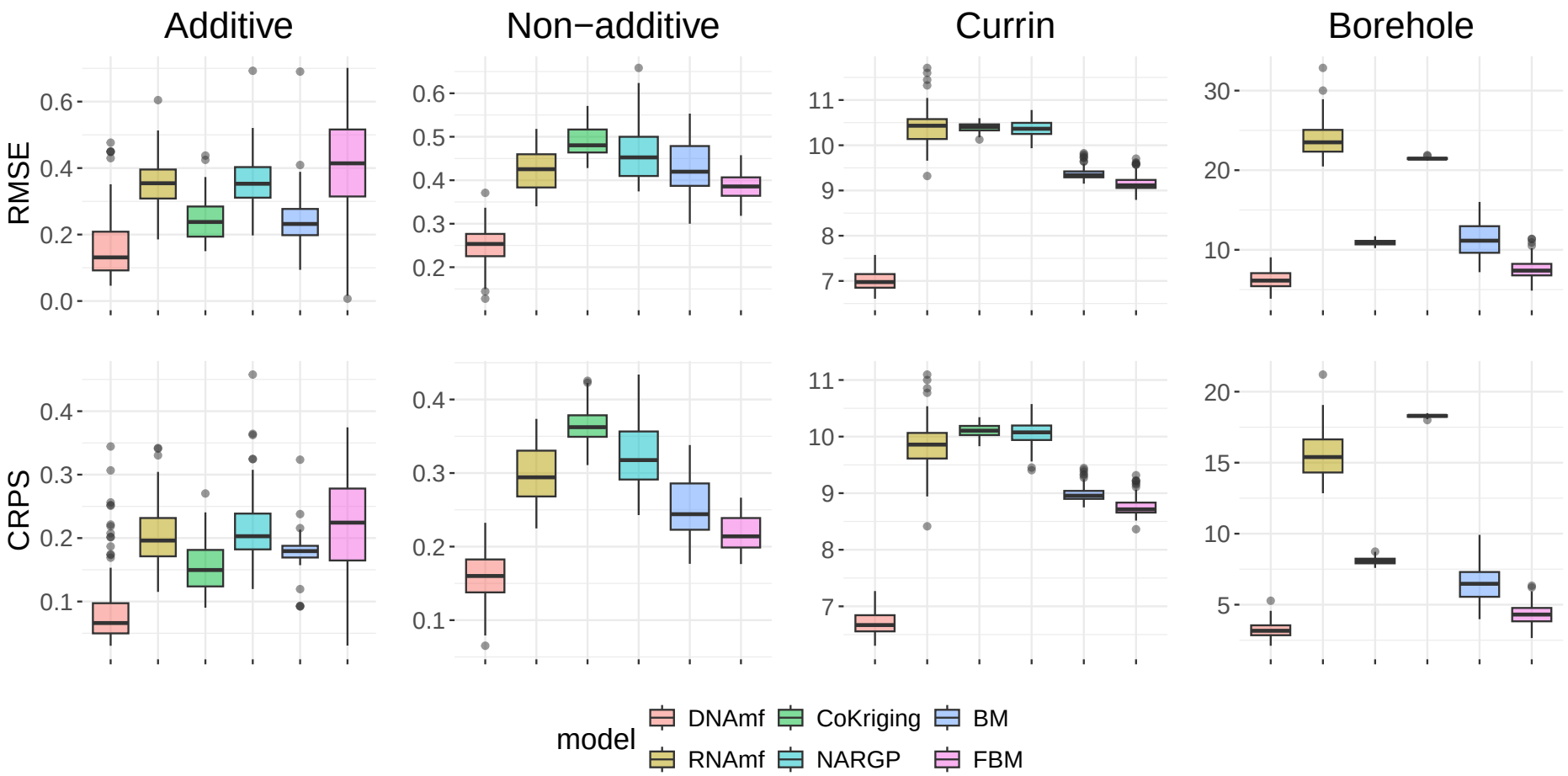


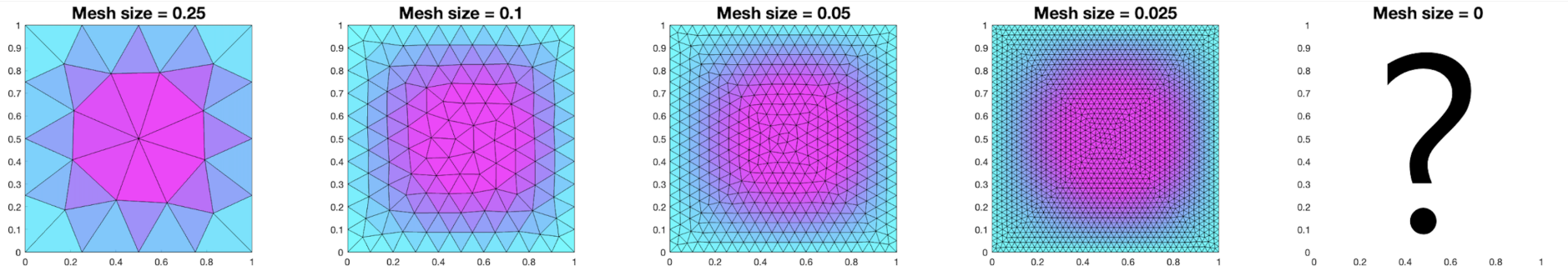
Illustration of the posterior distributions for the additive function $f(t,x) = \exp(-1.4x)\cos(3.5\pi x) + t^2\sin(40x)/10$ (top row) and the non-additive function (bottom row) $f(t,x) = \sin(\frac{10\pi x}{5+t}) + 0.2\sin(8\pi x)$.

Numerical studies: Emulation performance

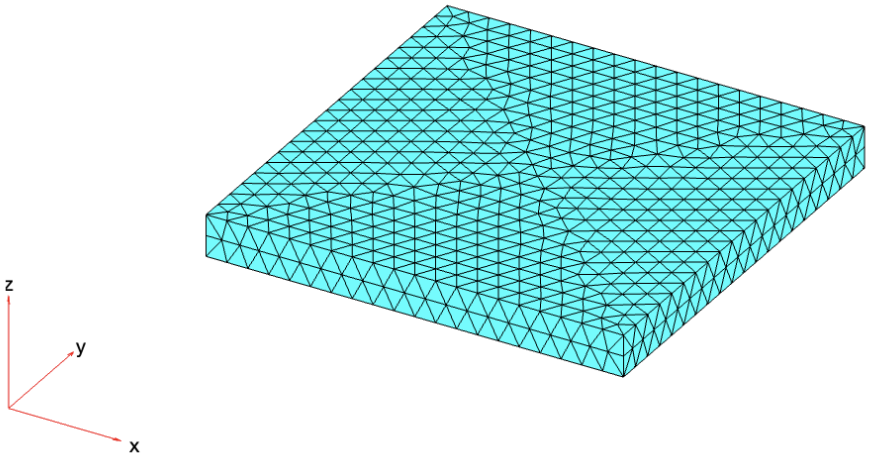


RMSEs and CRPSs of four synthetic examples across 100 repetitions.

Real Applications



Poisson's equation

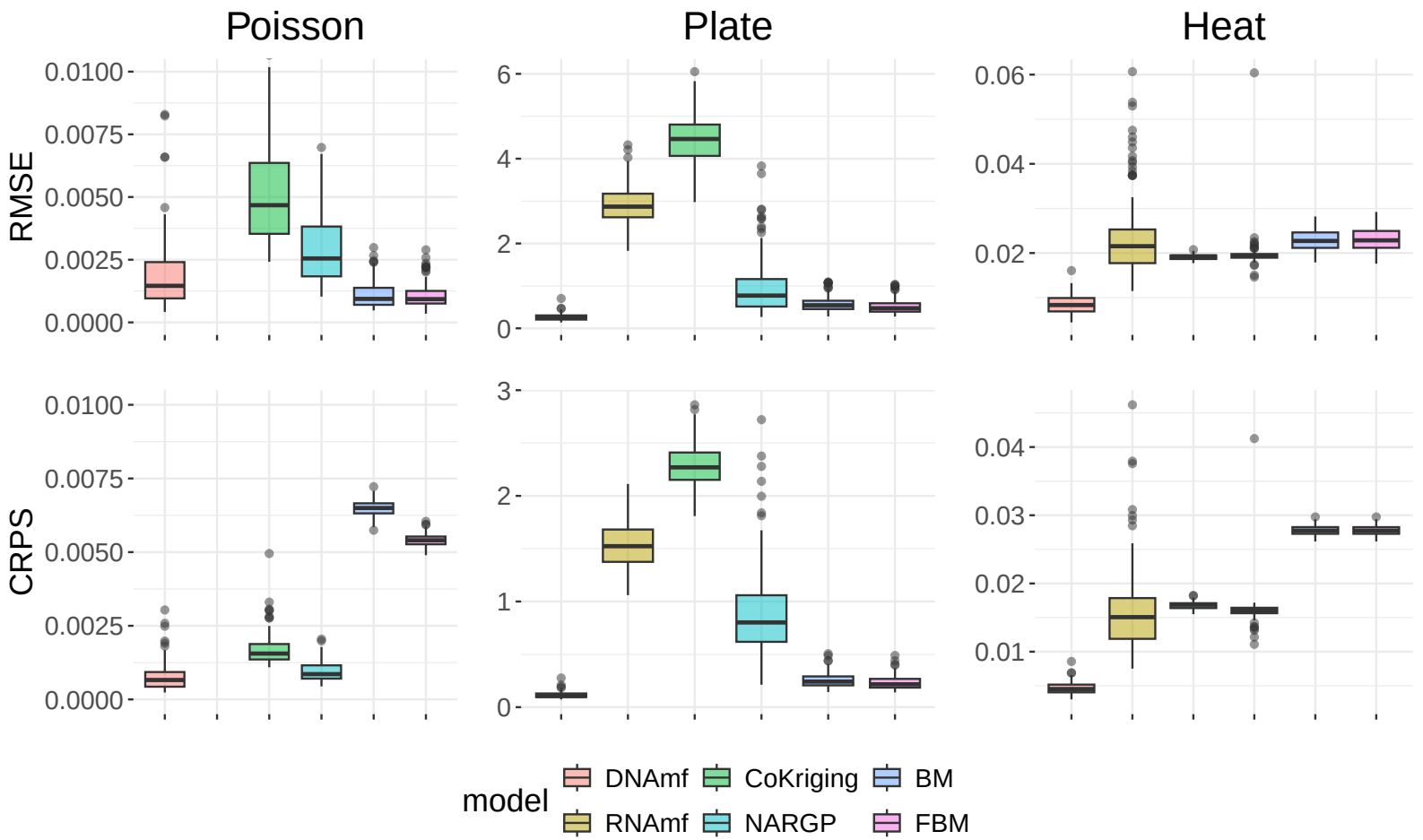


Vibration of square plate

$$\frac{\partial u(x,s)}{\partial s} = D\frac{\partial^2 u(x,s)}{\partial x^2}$$

Heat equation

Real Applications



RMSEs and CRPSs of three real case studies across 100 repetitions.

Without Nested Design?

- The RNA and DNA models require a nested design, i.e.,

$$\mathcal{X}_L \subseteq \mathcal{X}_{L-1} \subseteq \cdots \subseteq \mathcal{X}_1 \subseteq \Omega,$$

- Q: Can these models be constructed and still provide accurate predictions without a nested design?

RNA/DNA Model with Non-Nested Design

- Pseudo outputs can be imputed using **SEM Algorithm** (Ming et al., 2023)!.
- Let $\mathcal{X}_L^* = \mathcal{X}_L$, and $\mathcal{X}_l^* = \mathcal{X}_l \cup \mathcal{X}_{l+1}^*$ for $l = 1, \ldots, L-1$ with $\mathcal{X}_l \cap \mathcal{X}_{l+1} = \varnothing$, then

$$\mathcal{X}_L^* \subseteq \mathcal{X}_{L-1}^* \subseteq \cdots \subseteq \mathcal{X}_1^* \subseteq \Omega.$$

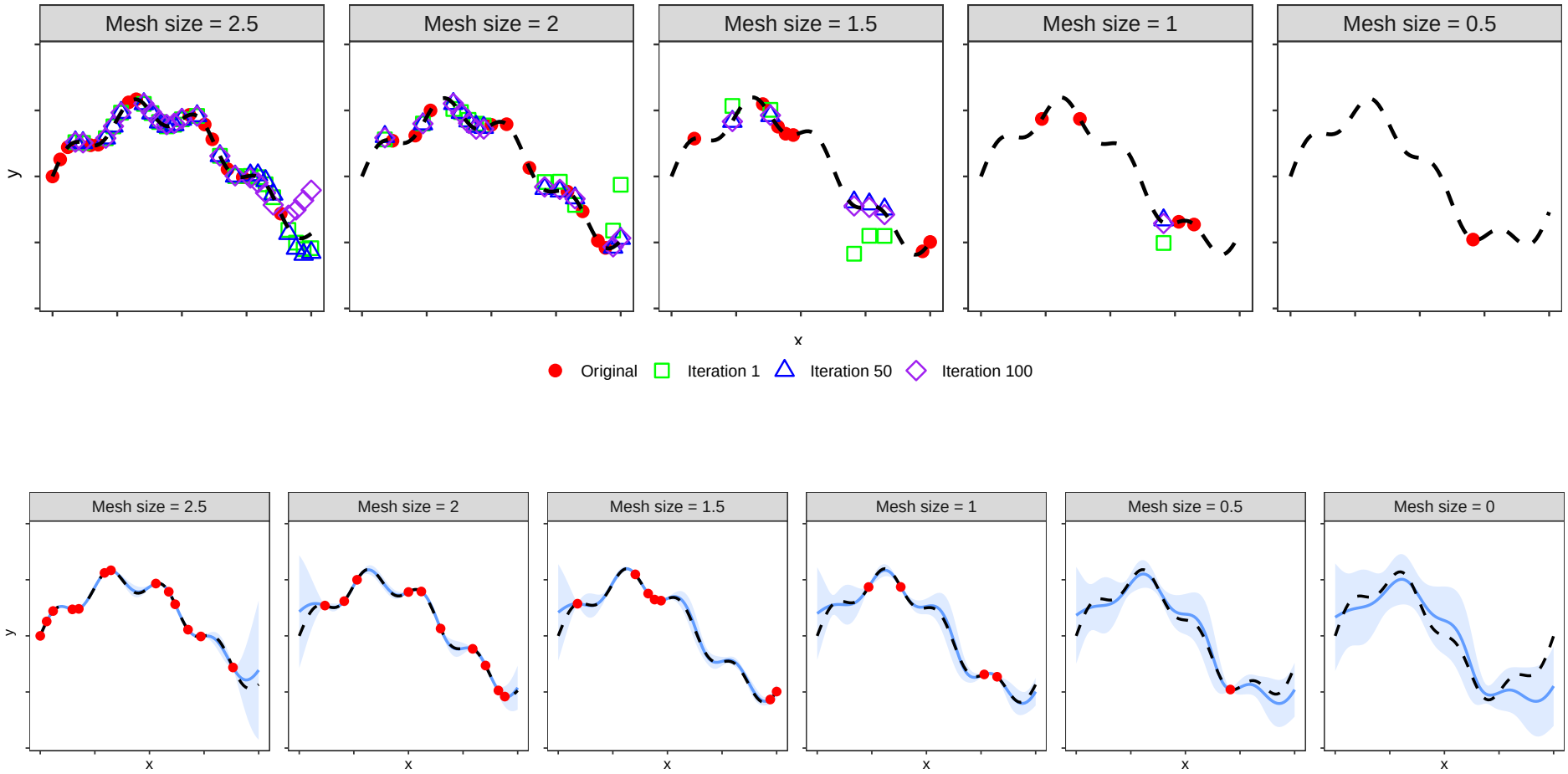
- Define *pseudo* inputs and outputs:

$$\widetilde{\mathcal{X}}_l := \mathcal{X}_l^* \setminus \mathcal{X}_l, \quad \widetilde{\mathbf{y}}_l := f_l(\widetilde{\mathcal{X}}_l)$$

- The *pseudo-complete* output is:

$$\mathbf{y}_l^* := f_l(\mathcal{X}_l^*)$$

DNA Model with Non-Nested Design



Conclusion

Conclusion

- The RNA model with four active learning strategies (ALD, ALM, ALC, and ALMC) and DNA model are introduced.
- Closed-form posterior expressions are derived.
- The DNA model enables interpretable connections between the inputs and the tuning parameter.
- The models accommodate non-nested designs.

Thank you!

Numerical studies: Emulation performance

$$
\begin{cases}
f(t,x) = \exp(-1.4x)\cos(3.5\pi x) + t^2\sin(40x)/10, \quad x \in [0,1], \\
f(t,x) = \sin\left(\frac{10\pi x}{5+t}\right) + 0.2\sin(8\pi x), \quad x \in [0,1], \\
f(t,\mathbf{x}) = \left[\exp(-4t) - \exp\left(-\frac{1}{2x_2}\right)\right] \frac{2300x_1^3+1900x_1^2+2092x_1+60}{100x_1^3+500x_1^2+4x_1+20}, \quad \mathbf{x} = (x_1,x_2) \in [0,1]^2. \\
f(t,\mathbf{x}) = \frac{(2\pi-t^2)T_u(H_u-H_l)}{\log(r/r_w)\left(1+t^2+\frac{2LT_u}{\log(r/r_w)r_w^2K_w}+\frac{T_u}{T_l}\right)},
\end{cases}
$$

	n_1	n_2	n_3	n_4	n_5	n_6	d	t_L
Additive	17	8	4	2	1		1	2.5
Non-additive	17	8	4	2	1		1	2.5
Currin	71	35	17	8	4	2	2	2.5
Borehole	71	35	17	8	4	2	8	2.5

Real Applications

	n_1	n_2	n_3	n_4	n_5	d	t_L	c	γ
Poisson	17	11	7	5	3	1	0.1	0.65	1
Plate	28	21	15	11	8	3	0.55	0.9	3
Heat	25	15	9	5	3	1	0.5	0.7	1.5

The closed-form expression of DNA model

The closed-form expressions

- Under the **nonseparable squared exponential kernel**, the posterior mean and variance can be obtained as follows:

$$h_1(\mathbf{x}|t,\mu,\sigma^2) = \alpha + \sqrt{\frac{\theta_y}{\theta_y+2\sigma^2}}\sum_{i=1}^{N-1} r_i c_i^{\frac{d+1}{2}+\frac{\delta}{\beta}} \times \exp\left(-c_i\left\{\sum_{j=1}^{d}\frac{(x_j-(\mathbf{X}_{-1})_{ij})^2}{\theta_j}+\frac{((Y_{-L})_i-\mu)^2}{\theta_y+2\sigma^2}\right\}\right),$$

$$h_2(\mathbf{x}|t,\mu,\sigma^2) = \tau^2-(\mu-\alpha)^2+\left(\sum_{i,k=1}^{N-1}\zeta_{ik}(t,\mu,\sigma^2)\left(r_i r_k-\tau^2(\mathbf{K}^{-1})_{ik}\right)(c_i c_k)^{\frac{d+1}{2}+\frac{\delta}{\beta}} \times \exp\left(-\sum_{j=1}^{d}\frac{c_i(x_j-(\mathbf{X}_{-1})_{ij})^2+c_k(x_j-(\mathbf{X}_{-1})_{kj})^2}{\theta_j}\right)\right),$$

The closed-form expression of DNA model

The closed-form expressions

where $r_i = (\mathbf{K}^{-1}(Y_{-1} - \alpha \mathbf{1}_{N_{-1}}))_i$, $c_i := c(t, \mathbf{t}_i) = \left(\frac{(t-\mathbf{t}_i)^2}{\theta_t} + 1\right)^{-\beta}$ and

$$\zeta_{ik}(t, \mu, \sigma^2) = \sqrt{\frac{\theta_y}{\theta_y + 2(c_i + c_k)\sigma^2}} \times \exp\left(-\frac{c_i\left((Y_{-L})_i - \mu\right)^2 + c_k\left((Y_{-L})_k - \mu\right)^2 + \frac{2}{\theta_y} c_i c_k \sigma^2 \left((Y_{-L})_i - (Y_{-L})_k\right)^2}{\theta_y + 2(c_i + c_k)\sigma^2}\right).$$

SEM Algorithm

- **Initialization**: Fit independent GPs at each fidelity level and impute initial *pseudo* outputs $\{\widetilde{\mathbf{y}}_l^{(0)}\}_{l=1}^{L-1}$ using posterior means.

- **E-step (Imputation)**:
  - Sample $\widetilde{\mathbf{y}}_1^{(m)} \sim p(\widetilde{\mathbf{y}}_1 \mid \mathbf{y}_1; \hat{\alpha}_1, \hat{\tau}_1^2, \hat{\boldsymbol{\theta}}_1)$
  - Sample $\{\widetilde{\mathbf{y}}_l^{(m)}\}_{l=2}^{L-1} \sim p(\{\widetilde{\mathbf{y}}_l\}_{l=2}^{L-1} \mid Y_{-1}; \hat{\boldsymbol{\varphi}}^{(m-1)}, Y_{-L}^{*(m-1)})$

- **M-step**: With the pseudo-complete dataset, update parameters by maximizing the likelihood:

$$\left\{\{\mathbf{y}_l^{*(m)}\}_{l=1}^{L},\ \{\mathcal{X}_l^{*}\}_{l=1}^{L},\ \{t_l\}_{l=1}^{L}\right\}$$

SEM Algorithm

- Repeat the E- and M-steps for M iterations with burn-in period $B = \lceil 0.75M \rceil$.
- $\hat{\alpha} = \frac{1}{M-B} \sum_{m=B+1}^{M} \hat{\alpha}^{(m)}$.
- SEM generates a Markov chain that fluctuates near the maximum of the complete-data likelihood.

DNA Model with Non-Nested Design

- For prediction, generate M pseudo-complete outputs using the estimated parameters: $Y^{*(m)} := \{\mathbf{y}_l^{*(m)}\}_{l=1}^{L}, \quad m = 1, \ldots, M$
- For each $Y^{*(m)}$, compute the posterior mean and variance:

$$\mu^{(m)}(t, \mathbf{x}) = \mathbb{E}[f(t, \mathbf{x}) | Y^{*(m)}], \quad \sigma^{2(m)}(t, \mathbf{x}) = \mathbb{V}[f(t, \mathbf{x}) | Y^{*(m)}]$$

- Final predictive mean and variance are obtained by averaging over M samples:

$$\mu^*(t, \mathbf{x}) \approx \frac{1}{M} \sum_{m=1}^{M} \mu^{(m)}(t, \mathbf{x})$$

$$\sigma^{2*}(t, \mathbf{x}) \approx \frac{1}{M} \sum_{m=1}^{M} \left(\mu^{(m)}(t, \mathbf{x})^2 + \sigma^{2(m)}(t, \mathbf{x}) \right) - \mu^*(t, \mathbf{x})^2$$