Active Learning for a Recursive Non-Additive Emulator for Multi-Fidelity Computer Experiments

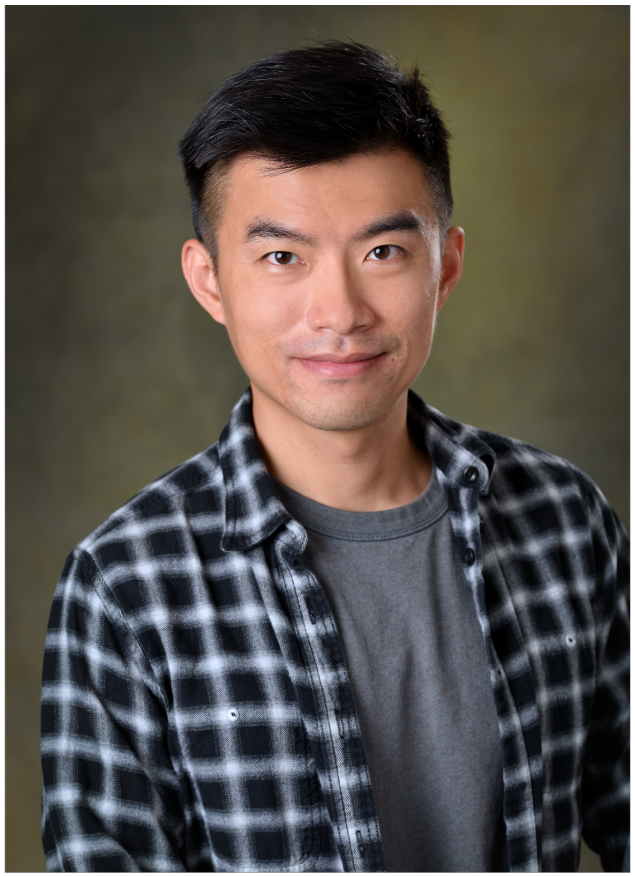
Junoh Heo and Chih-Li Sung

Department of Statistics and Probability
Michigan State University

南區研討會 @ NSYSU
June 27, 2024

MICHIGAN STATE
UNIVERSITY

Junoh Heo



Chih-Li Sung

Outline

Multi-Fidelity Simulations

- Computer models have been widely adopted to understand a real-world feature, phenomenon or event.
- Computer simulations are used to solve these models (e.g., finite element / finite difference) .

Multi-Fidelity Simulations

- Computer models have been widely adopted to understand a real-world feature, phenomenon or event.
- Computer simulations are used to solve these models (e.g., finite element / finite difference) .
- The simulation can be either
  - High-fidelity simulation: costly but close to the truth

Multi-Fidelity Simulations

- Computer models have been widely adopted to understand a real-world feature, phenomenon or event.
- Computer simulations are used to solve these models (e.g., finite element / finite difference) .
- The simulation can be either
  - High-fidelity simulation: costly but close to the truth
  - Low-fidelity simulation: cheaper but less accurate

Multi-Fidelity Simulations

- Computer models have been widely adopted to understand a real-world feature, phenomenon or event.
- Computer simulations are used to solve these models (e.g., finite element / finite difference) .
- The simulation can be either
  - High-fidelity simulation: costly but close to the truth
  - Low-fidelity simulation: cheaper but less accurate
  - (intermediate-fidelity simulation)

Motivated Example: Finite Element Simulations

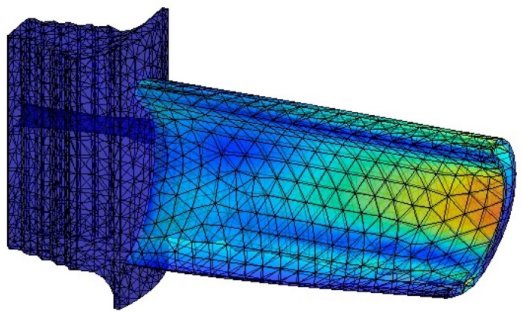
- Thermal stress of jet engine turbine blade can be analyzed through a static structural computer model.
- The model can be *numerically* solved via finite element method.

Motivated Example: Finite Element Simulations

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

Motivated Example: Finite Element Simulations

- Thermal stress of 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 thermal stress
- e.g., $\mathbf{x} = (0.23, 0.71)$

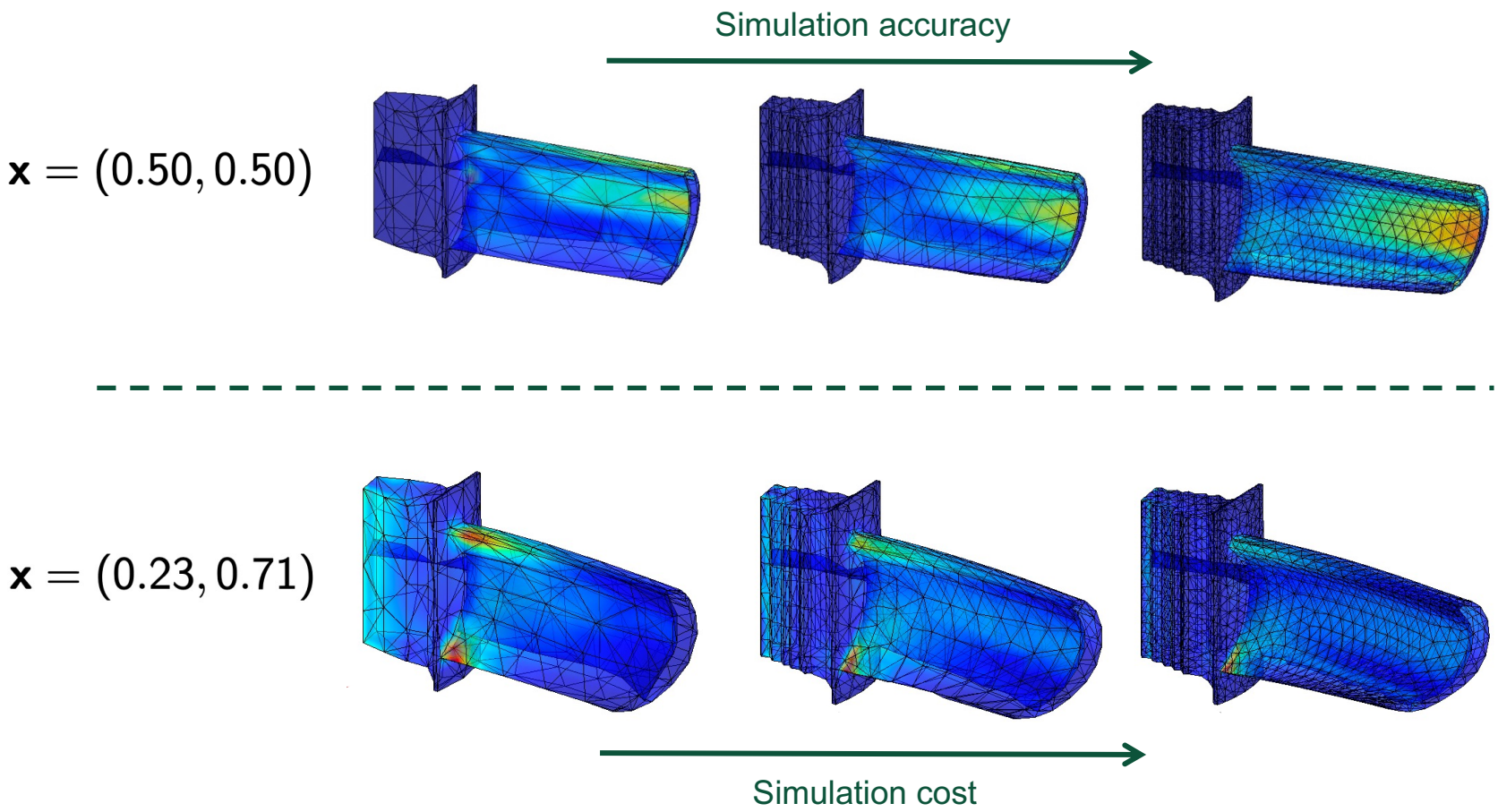


maximum of thermal stress $f(0.23, 0.71) = 20.3$

Multi-Fidelity Simulations

less accurate but cheaper

accurate but expensive



Statistical Emulation

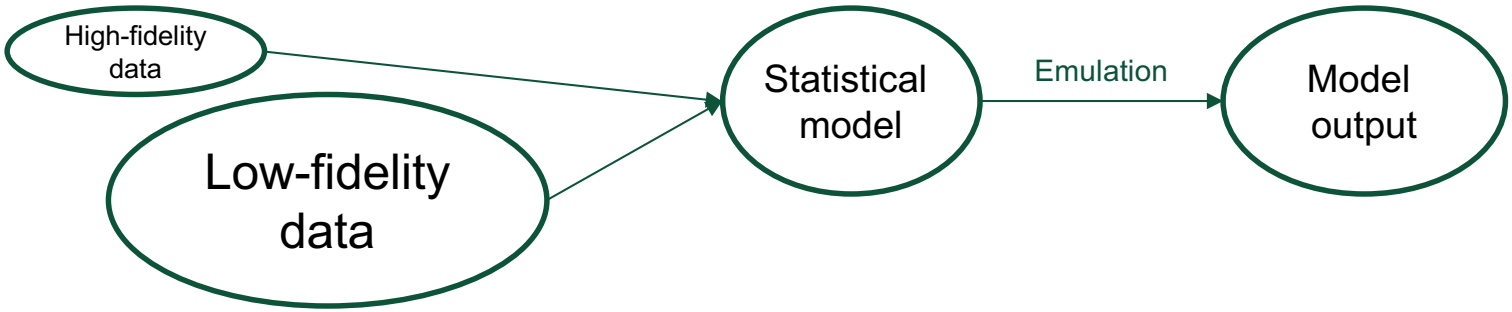
- Can we leverage both low- and high-fidelity simulations in order to

Statistical Emulation

- Can we leverage both low- and high-fidelity simulations in order to
  - maximize the accuracy of model predictions,
  - while minimizing the cost associated with the simulations?

Statistical Emulation

- Can we leverage both low- and high-fidelity simulations in order to
  - maximize the accuracy of model predictions,
  - while minimizing the cost associated with the simulations?
  - A cheaper statistical model emulating the model output based on the simulations with multiple fidelities
  - Often called emulator or surrogate model



Notation

fidelity level	1		2	3
output	$f_1(\mathbf{x})$		$f_2(\mathbf{x})$	$f_3(\mathbf{x})$
simulation cost	C_1	$<$	C_2	$<$ C_3

Notation

fidelity level	1		2	3
output	$f_1(\mathbf{x})$		$f_2(\mathbf{x})$	$f_3(\mathbf{x})$
simulation cost	C_1	$<$	C_2	$<$ C_3

- 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 $l = 1, \ldots, L$

Existing Methods

- The canonical approach is auto-regressive (AR) model (Kennedy and O'Hagan, 2000).
- AR model assumes additive structure of Gaussian processes (GPs).

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

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

Existing Methods

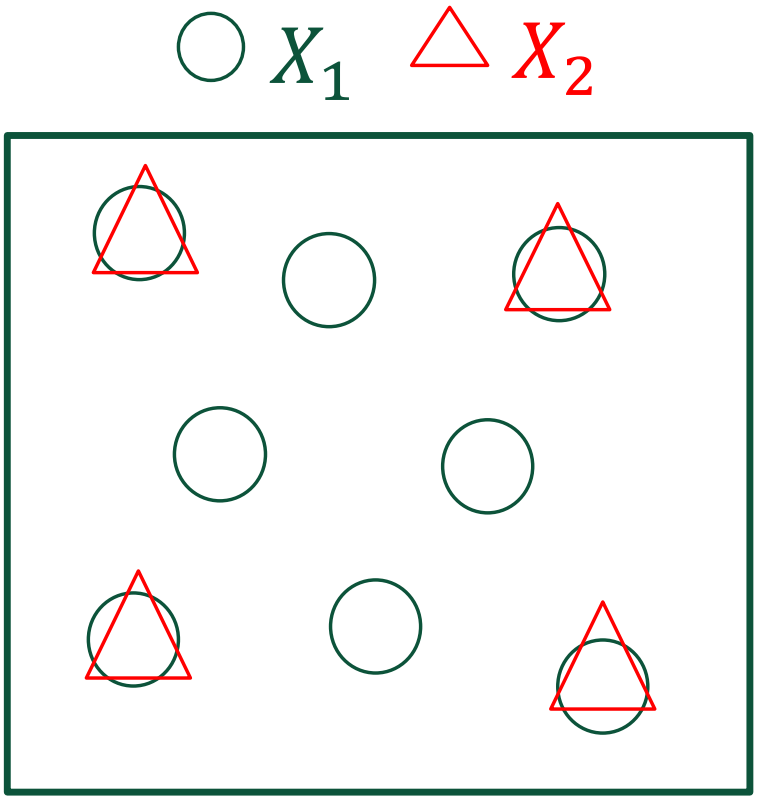
- Nested design, i.e.,

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

  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 et al., 2009; Qian, 2009).

Nested Design

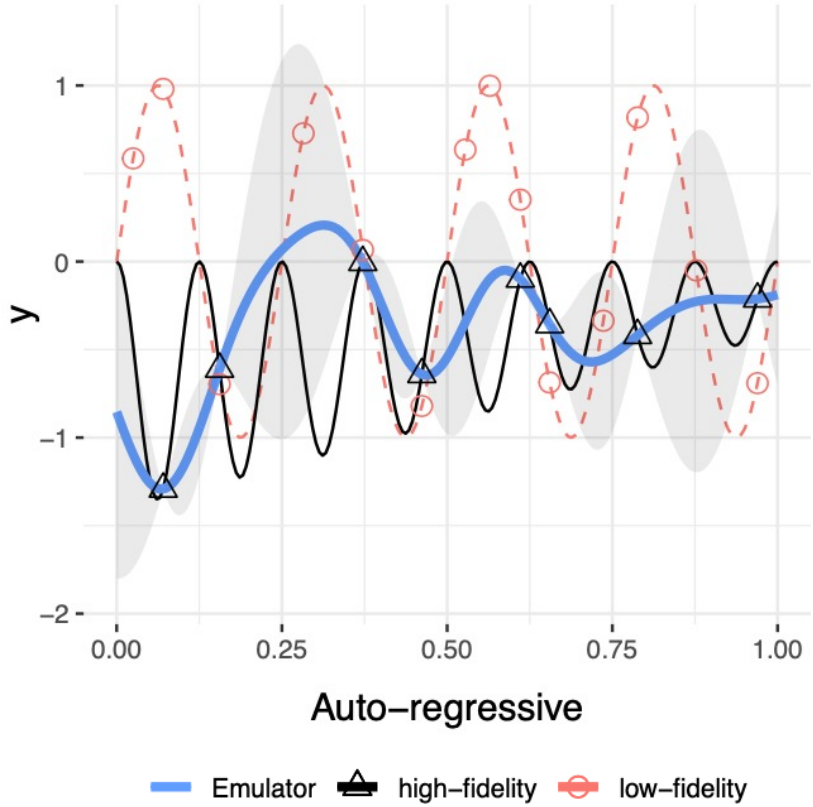


Existing Methods

- Q: Would it always follow an additive structure?

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 $f_2(x) = (x - \sqrt{2})f_1^2(x)$.

RNA emulator

- We propose 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!

RNA emulator

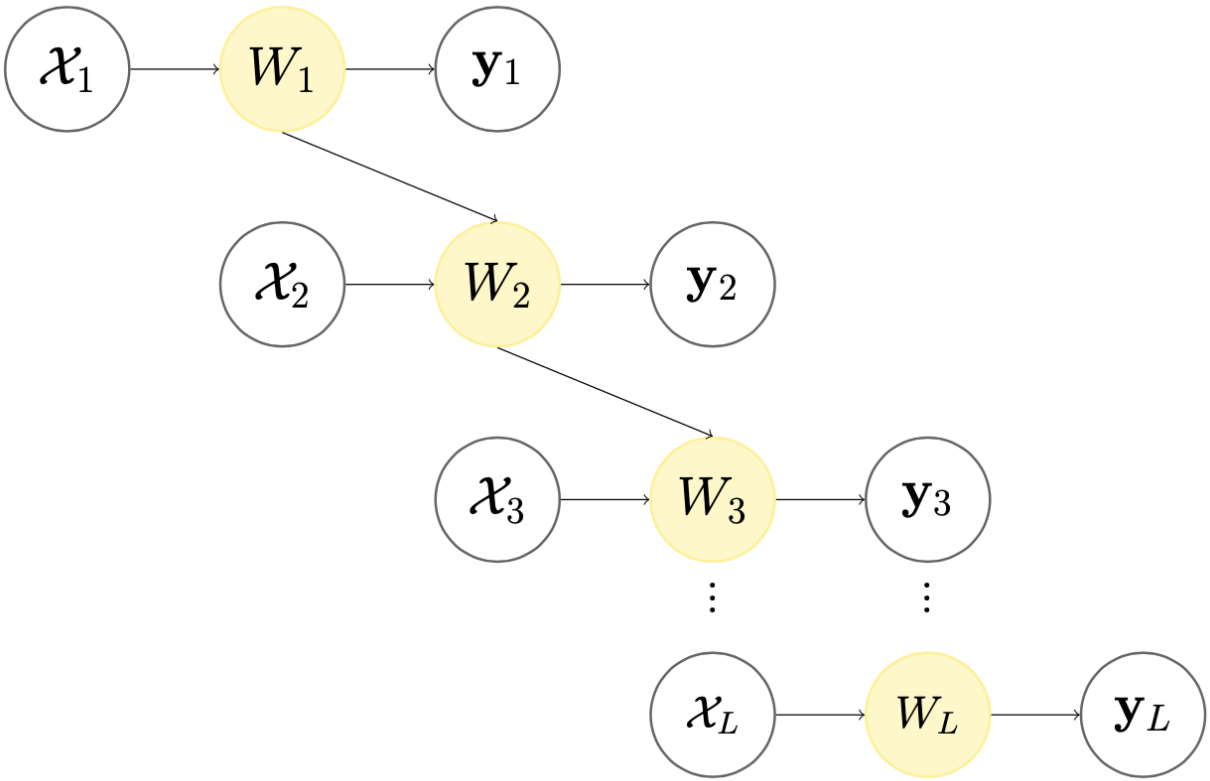
- We propose a Recursive Non-Additive emulator (RNA emulator) to overcome this limitation in a recursive fashion:

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

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

- 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 independent GP priors

RNA emulator



RNA emulator

RNA emulator

$$W_1(\mathbf{x}) \sim \mathcal{GP}(\alpha_1, \tau_1^2 K_1(\mathbf{x}, \mathbf{x}')),$$

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

where $\mathbf{z} = (\mathbf{x}, y)$, and $K_1(\mathbf{x}, \mathbf{x}')$ and $K_l(\mathbf{z}, \mathbf{z}')$ are a positive definite kernel.

RNA emulator

RNA emulator

$$W_1(\mathbf{x}) \sim \mathcal{GP}(\alpha_1, \tau_1^2 K_1(\mathbf{x}, \mathbf{x}')),$$

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

where $\mathbf{z} = (\mathbf{x}, y)$, and $K_1(\mathbf{x}, \mathbf{x}')$ and $K_l(\mathbf{z}, \mathbf{z}')$ are a positive definite kernel.

- e.g., squared exponential kernel:

$$K_1(\mathbf{x}, \mathbf{x}') = \prod_{j=1}^{d} \exp\left(-\frac{(x_j - x_j')^2}{\theta_{1j}}\right)$$

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

Gaussian Process (GP)

- 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 $l = 2, \ldots, L$.

Gaussian Process (GP)

- 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 $l = 2, \ldots, L$.

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

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

Gaussian Process (GP)

- 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 $l = 2, \ldots, L$.

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

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

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

Parameter Estimation

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

$$n_l \log(\tau_l^2) + \log(\det(K_l(\mathcal{X}_l, f_{l-1}(\mathcal{X}_l)))) + \frac{1}{\tau_l^2}(\mathbf{y}_l - \alpha_l \mathbf{1}_{n_l})^T K_l(\mathcal{X}_l, f_{l-1}(\mathcal{X}_l))^{-1}(\mathbf{y}_l - \alpha_l \mathbf{1}_{n_l}).$$

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

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

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

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

- 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:

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

  2. Monte Carlo integration for the intractable posterior distribution:

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

RNA emulator

In contrast...

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

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

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

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 (Kyzyurova et al., 2018; Ming and Guillas, 2021):

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

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

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 $\sigma_l^{*2}(\mathcal{X}_l) = \mathbf{0}_{n_l}$.

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 $\sigma_l^{*2}(\mathcal{X}_l) = \mathbf{0}_{n_l}$.

- The closed form expressions can be derived under a Matérn kernel with the smoothness parameter $\nu = 1.5$ and $\nu = 2.5$ as well.

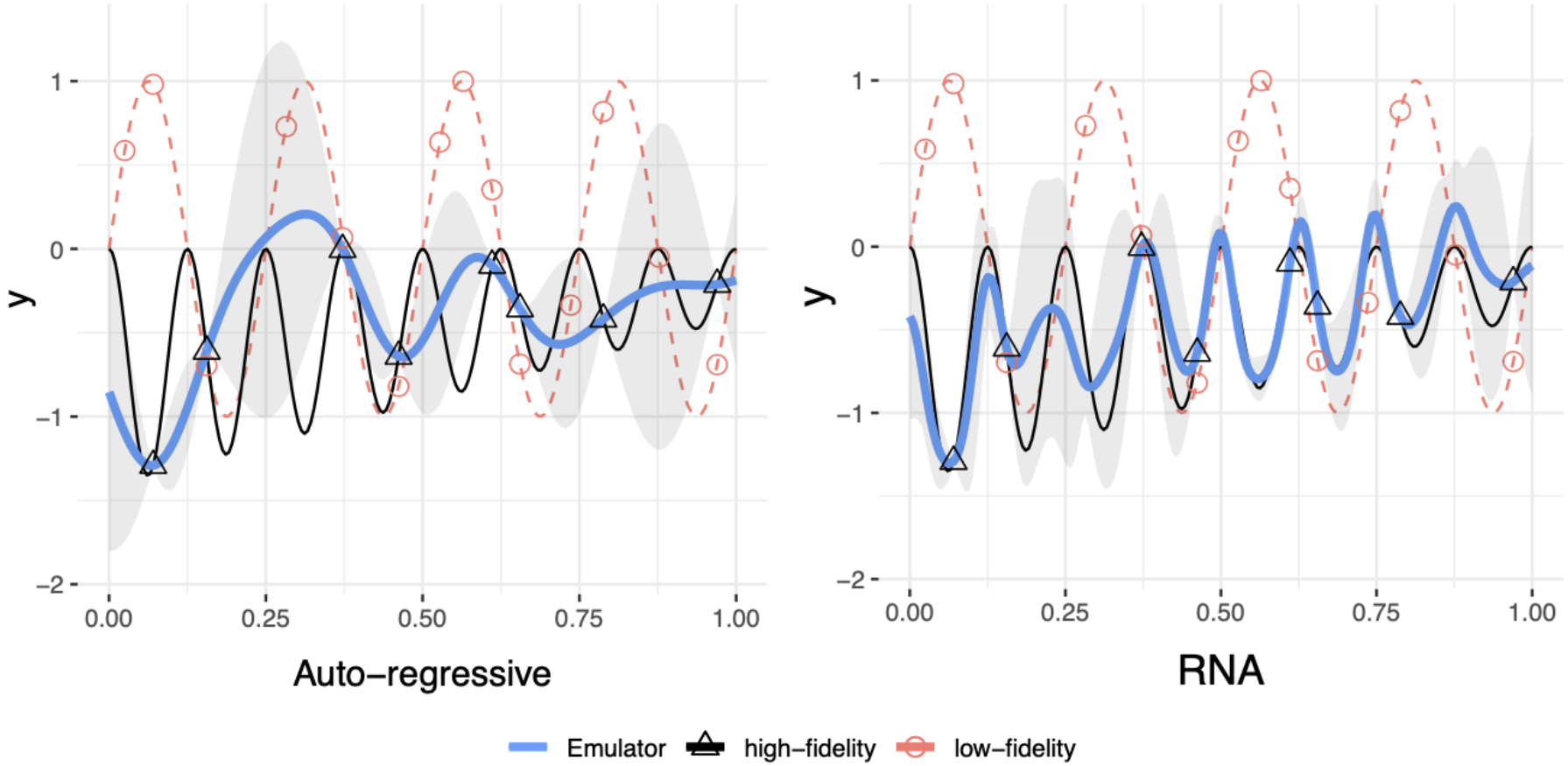
The closed form expression of RNA emulator

Proposition 2: Interpolation property

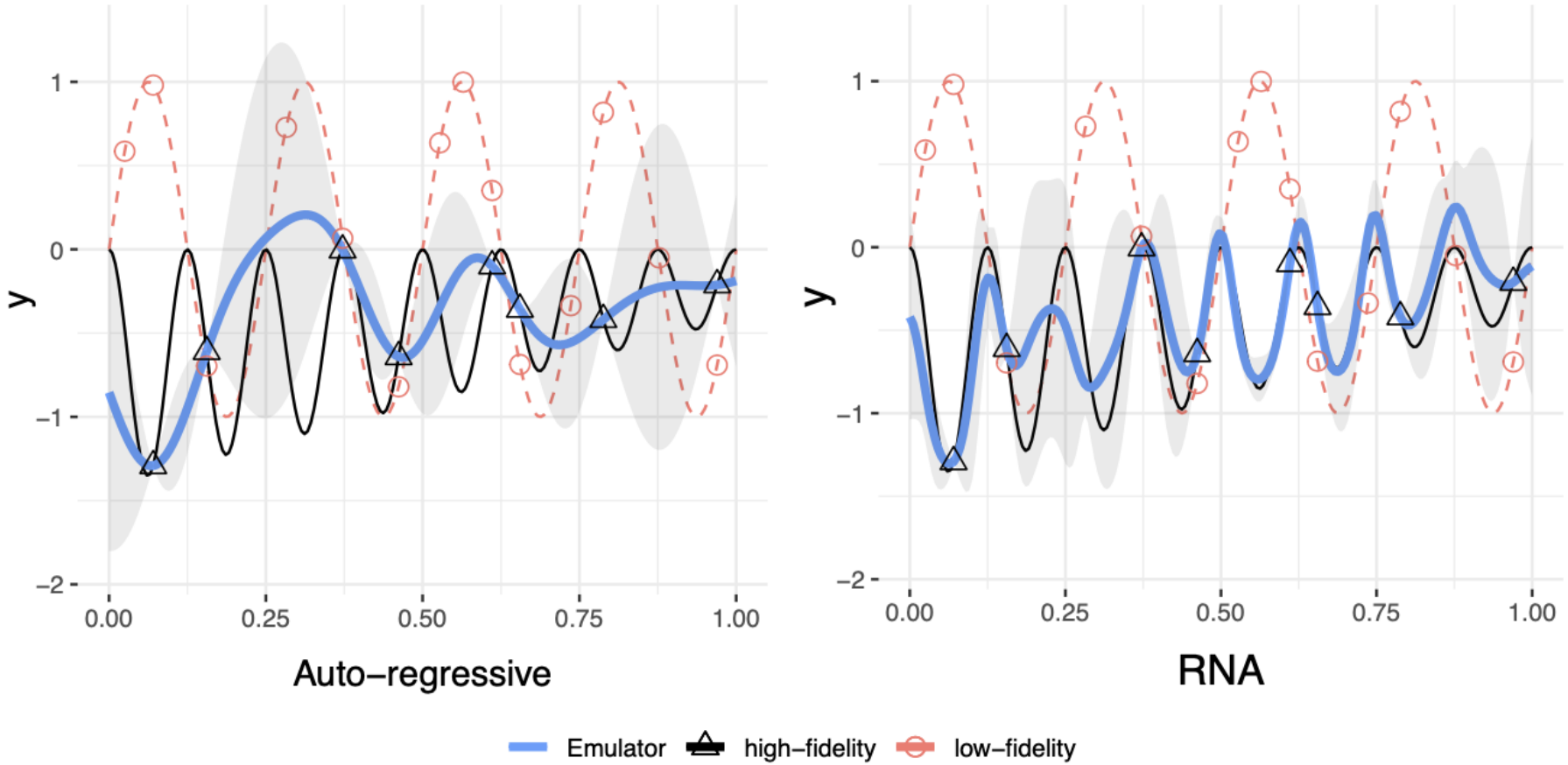
The RNA emulator exhibits interpolation property. That is, $\mu_l^*(\mathcal{X}_l) = \mathbf{y}_l$, and $\sigma_l^{*2}(\mathcal{X}_l) = \mathbf{0}_{n_l}$.

- 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. That is, $f_L(\mathbf{x})|\mathbf{y}_1, \ldots, \mathbf{y}_L \sim \mathcal{N}(\mu_L^*(\mathbf{x}), \sigma_L^{*2}(\mathbf{x}))$.
- R package called RNAmf is available.

RNA emulator



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.

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.
- 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.

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.

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 $f_l(\mathbf{x}^{[l]}_{n_l+1})$, we need to run $f_k(\mathbf{x}^{[k]}_{n_k+1})$ with $\mathbf{x}^{[k]}_{n_k+1} = \mathbf{x}^{[l]}_{n_l+1}$ for all $1 \leq k \leq l$.

Active Learning Decomposition (ALD)

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

Active Learning Decomposition (ALD)

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

- Suppose $L = 2$. We have

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

- 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:

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

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

Active Learning MacKay (ALM)

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

Active Learning MacKay (ALM)

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

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

- The closed-form expression of $\sigma_k^{*2}(\mathbf{x})$ facilitates the computation of ALM criterion.

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).

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:

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

where $\Delta\sigma^2_L(k,\mathbf{x}) = \int_\Omega \left\{\sigma^{*2}_L(\boldsymbol{\xi}) - \tilde{\sigma}^{*2}_L(\boldsymbol{\xi}; k, \mathbf{x})\right\} \mathrm{d}\boldsymbol{\xi}$ 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.

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:

$$\mathbf{x}^* = \underset{\mathbf{x}\in\Omega}{\operatorname{argmax}}\, \sigma_L^{*2}(\mathbf{x}).$$

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:

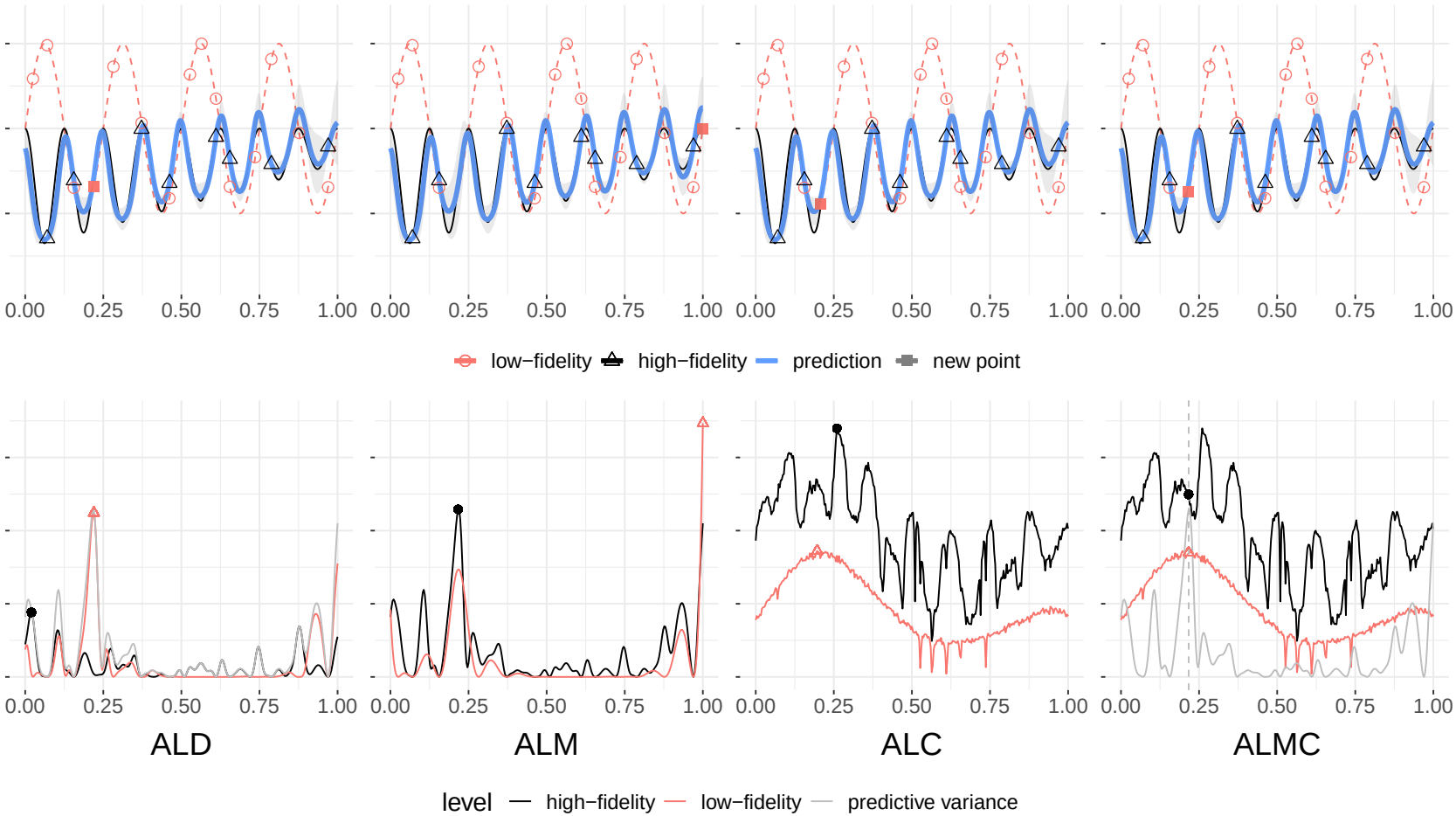
$$\mathbf{x}^* = \underset{\mathbf{x}\in\Omega}{\operatorname{argmax}}\, \sigma_L^{*2}(\mathbf{x}).$$

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

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

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

Demonstration



Numerical studies: 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: Emulation performance

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

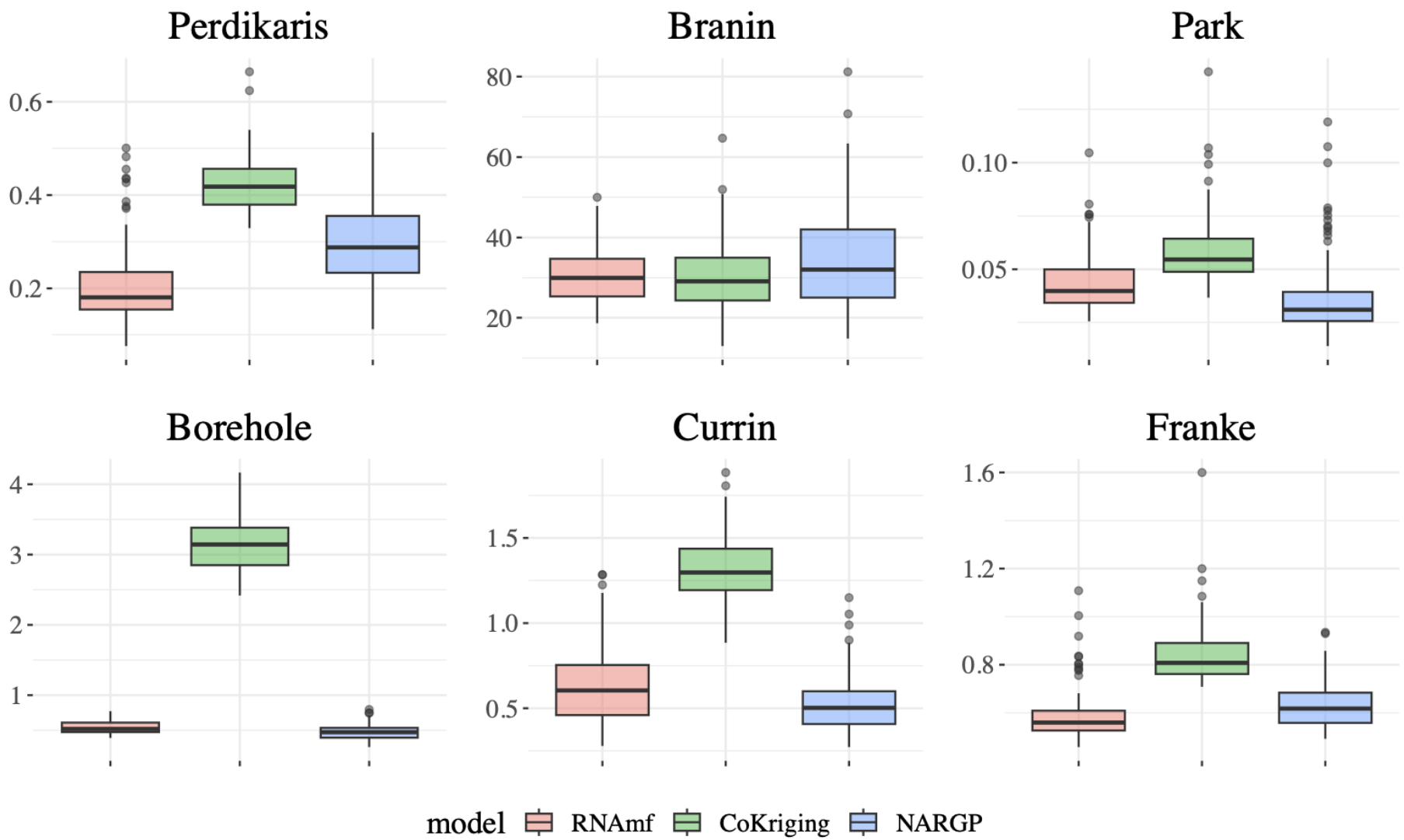
$$\begin{cases} f_1(x) = \sin(8\pi x) \\ f_2(x) = (x - \sqrt{2}) f_1^2(x) \end{cases} \text{for } x \in [0, 1],$$

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

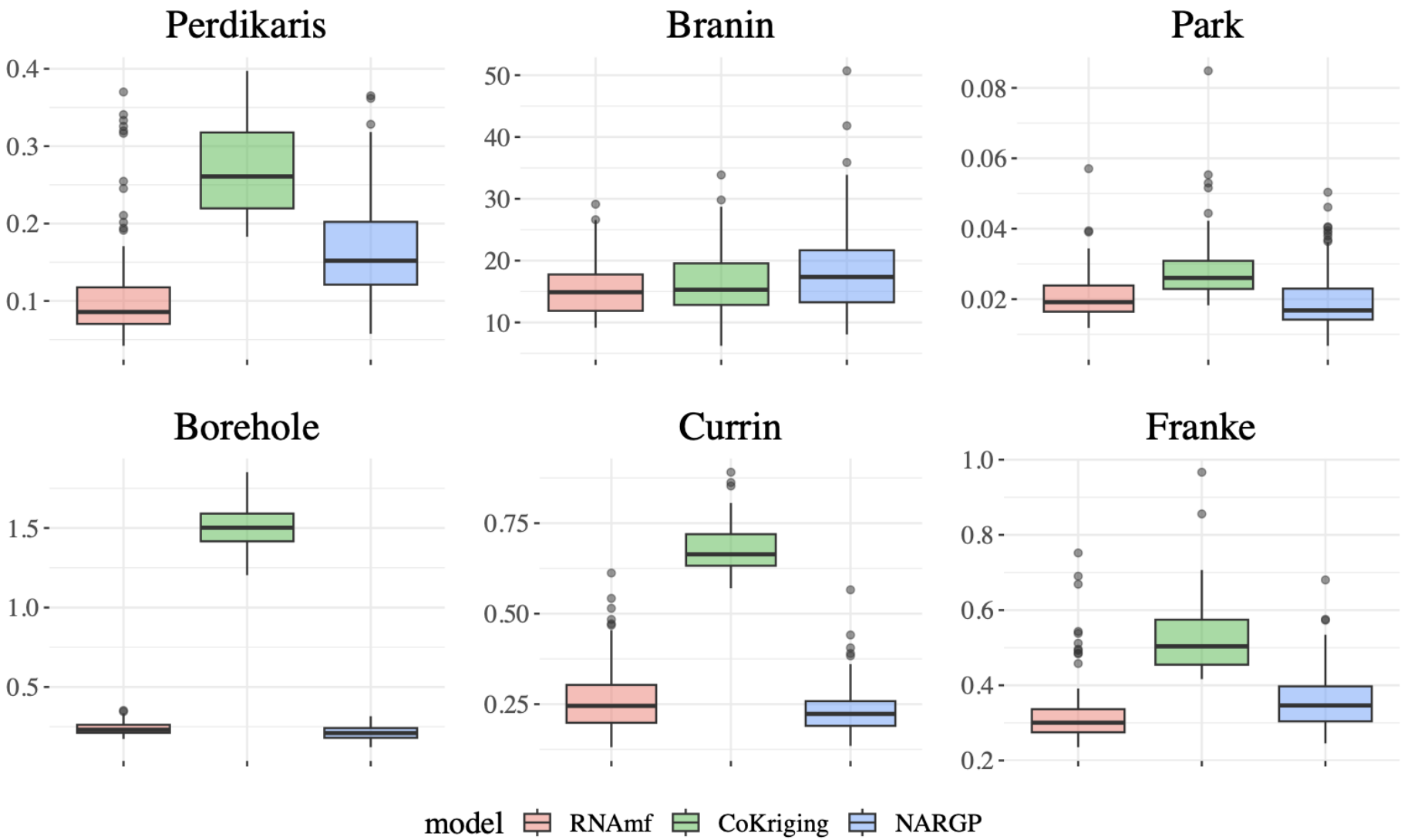
$$\begin{cases} f_1(\mathbf{x}) = f_2(\mathbf{x}) + \frac{\sin(x_1)}{10} f_2(\mathbf{x}) - 2x_1 + x_2^2 + x_3^2 + 0.5 \\ f_2(\mathbf{x}) = \frac{x_1}{2} \left[\sqrt{1 + (x_2 + x_3^2) \frac{x_4}{x_1^2}} - 1 \right] + (x_1 + 3x_4) \exp\left(1 + \sin(x_3)\right) \end{cases} \text{for } \mathbf{x} \in [0,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

	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

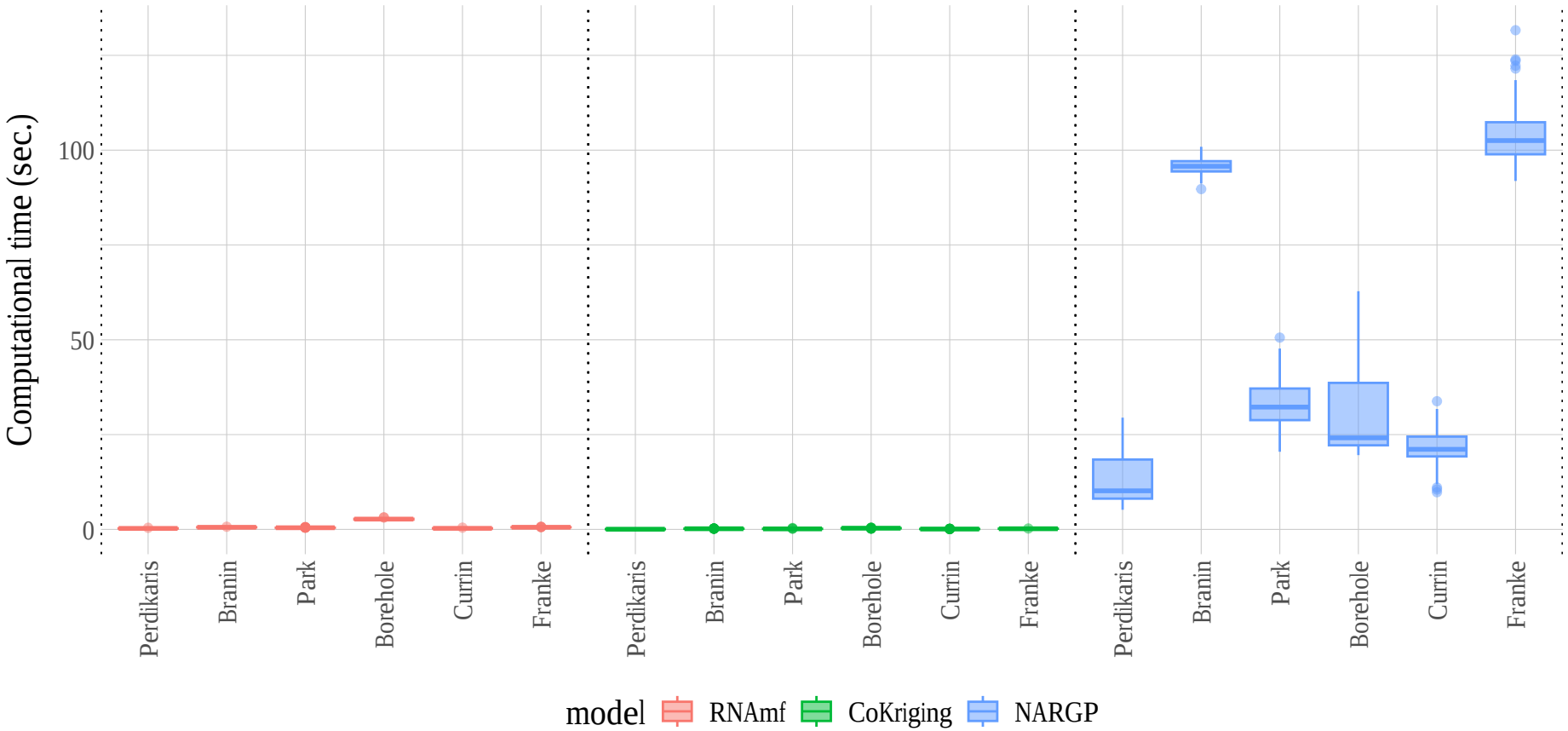
Numerical studies: RMSE



Numerical studies: CRPS



Numerical studies: Computational time



Computational time of six synthetic examples across 100 repetitions.

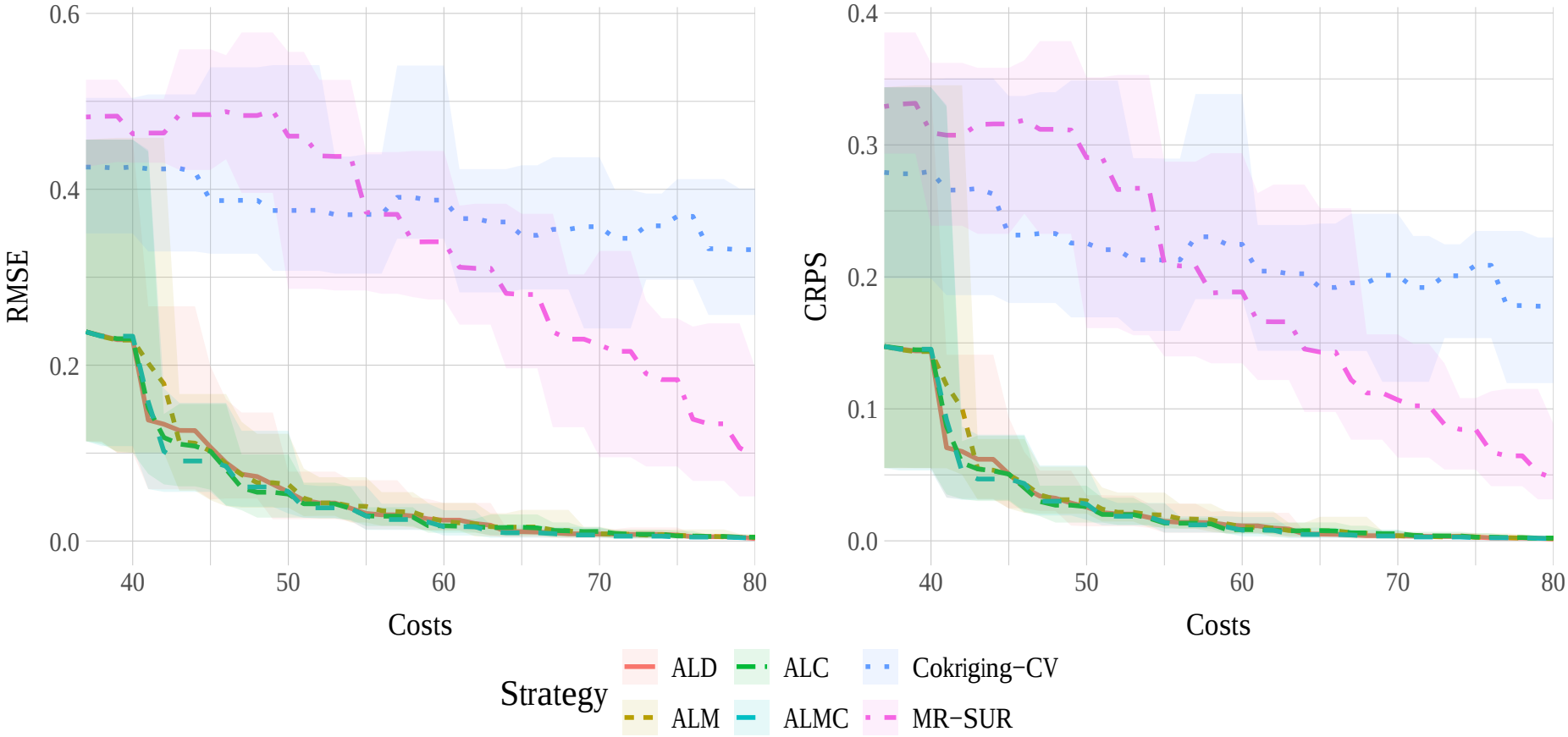
Active learning performance (Perdikaris)

- Perdikaris function (1-dim).
- Compare three 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)

Active learning performance (Perdikaris)

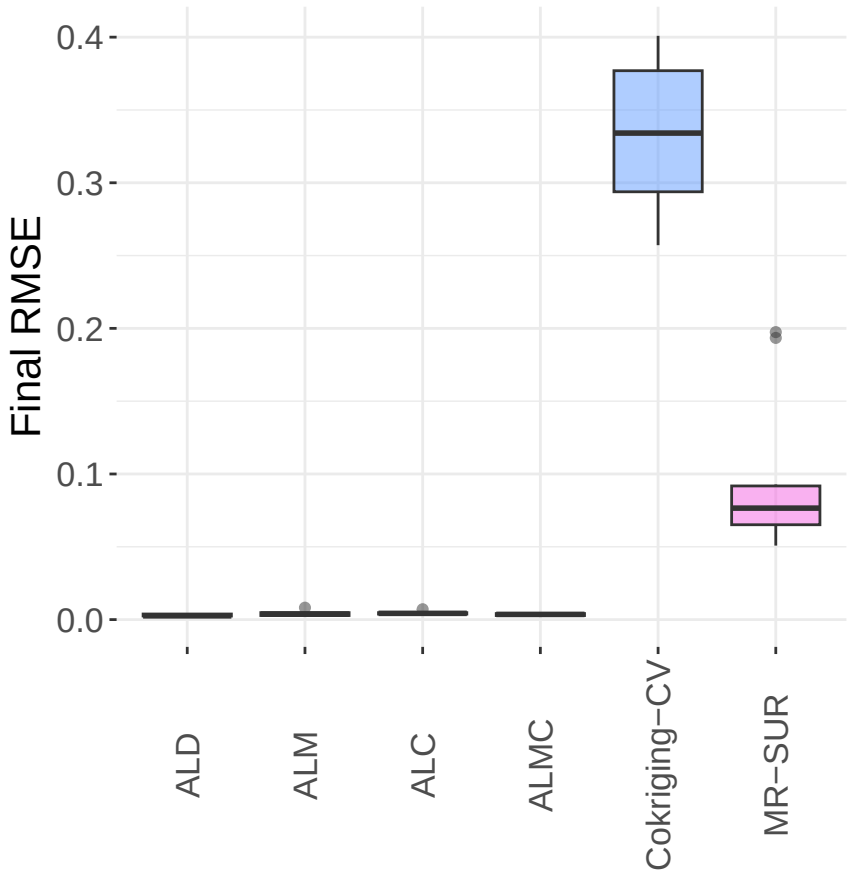
- Perdikaris function (1-dim).
- Compare three proposed strategies `ALD`, `ALM`, `ALC`, and `ALMC`, with
  - a cokriging-based sequential design (`CoKriging-CV`) (Le Gratiet and Cannamela, 2015)
  - a sequential design maximizing the rate of stepwise uncertainty reduction using the AR model (`MR-SUR`) (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.

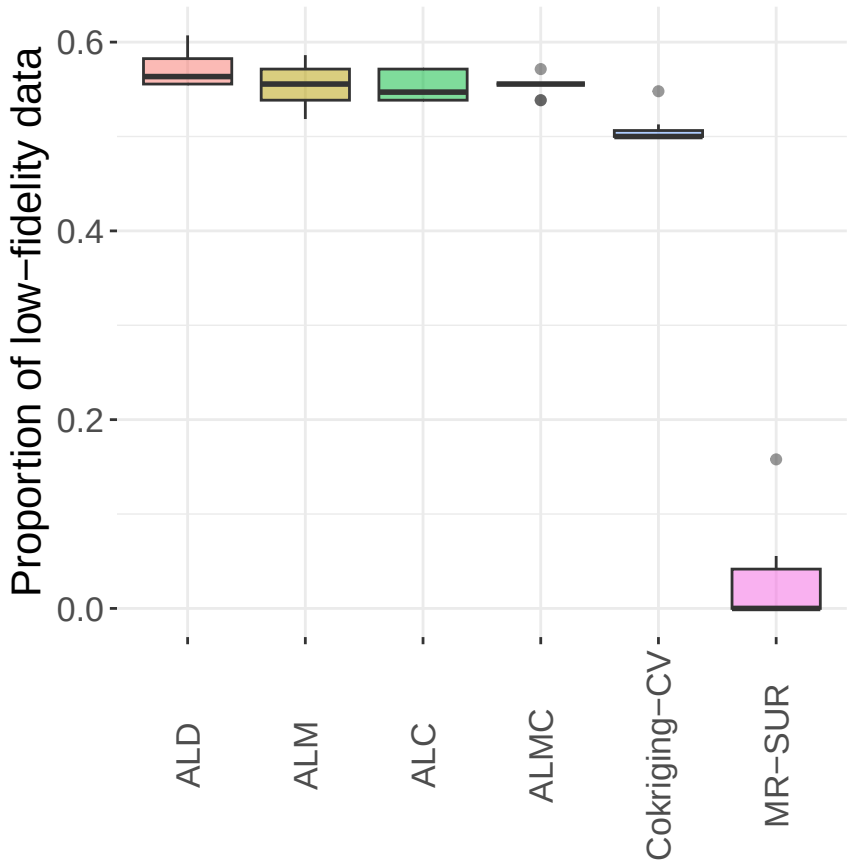
Active learning performance (Perdikaris)



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

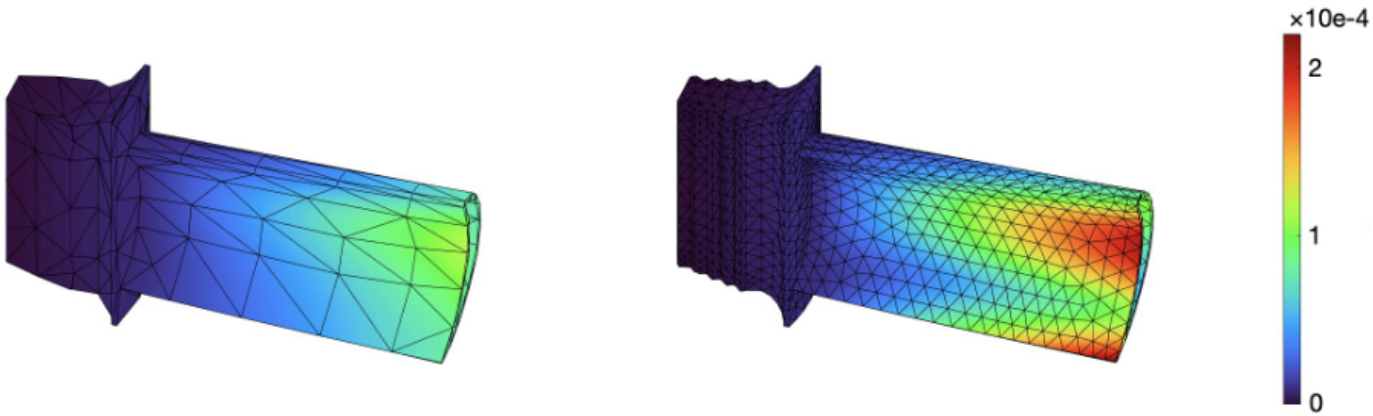
Active learning performance (Perdikaris)





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

Revisit motivated example



Low-fidelity (left) and high-fidelity (right) simulations at $\mathbf{x} = (0.5, 0.45)$.

- **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

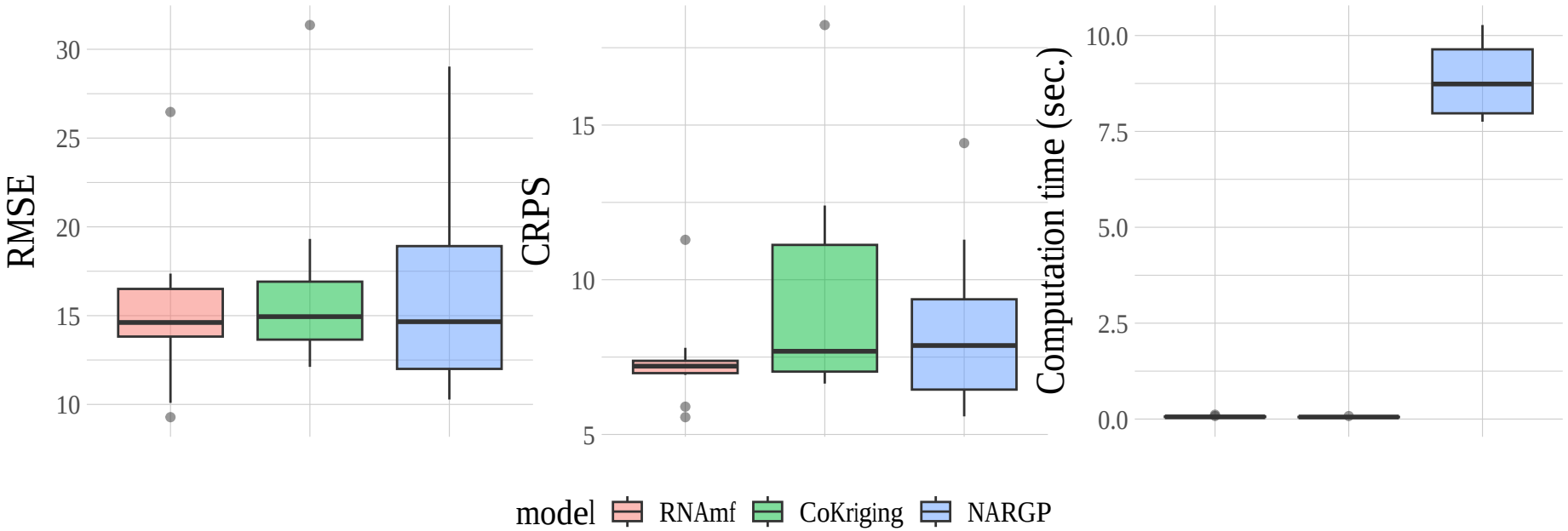
Revisit motivated example

- 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.

Revisit motivated example

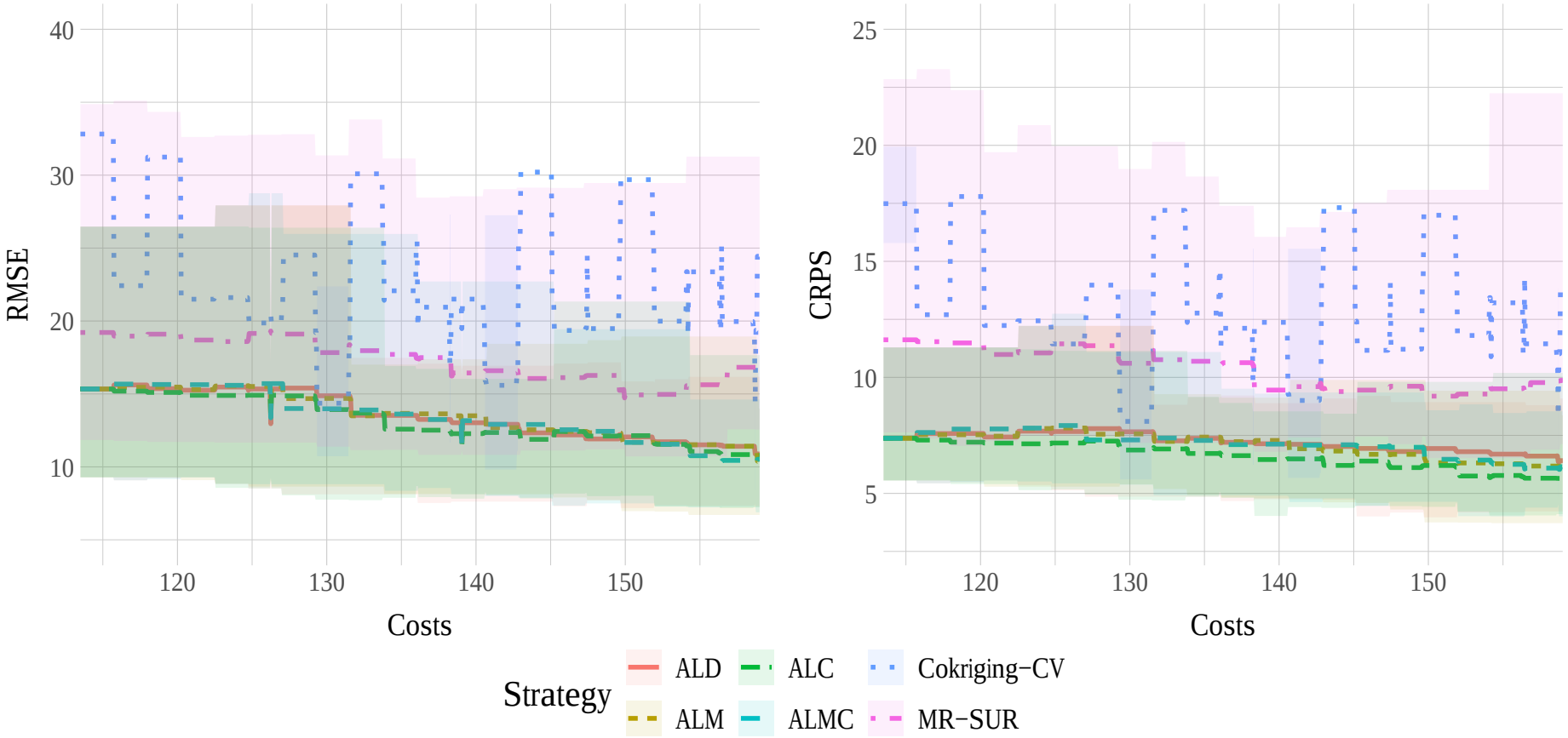
- Perform the finite element simulations with $n_1 = 20$ and $n_2 = 10$.
- The simulation time of the finite element simulations, which are respectively $C_1 = 2.25$ and $C_2 = 6.85$ (seconds) will be used for active learning.
- 10 repetitions with $n_{\text{test}} = 100$ random test input locations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

Blade: Emulation performance



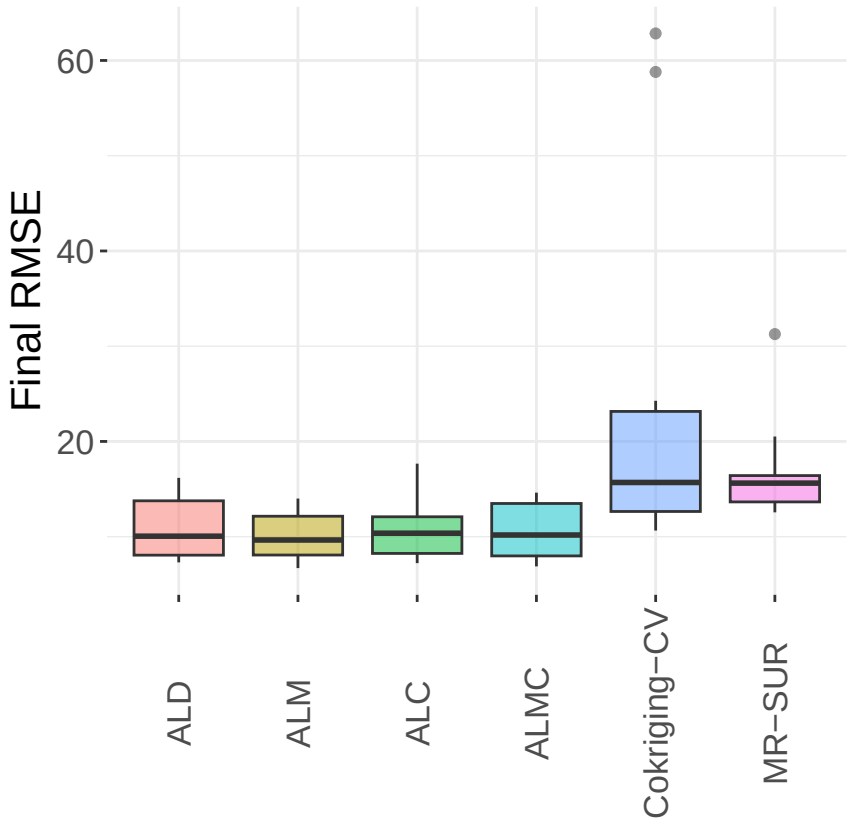
RMSE, CRPS, and computation time across 10 repetitions in the turbine blade application.

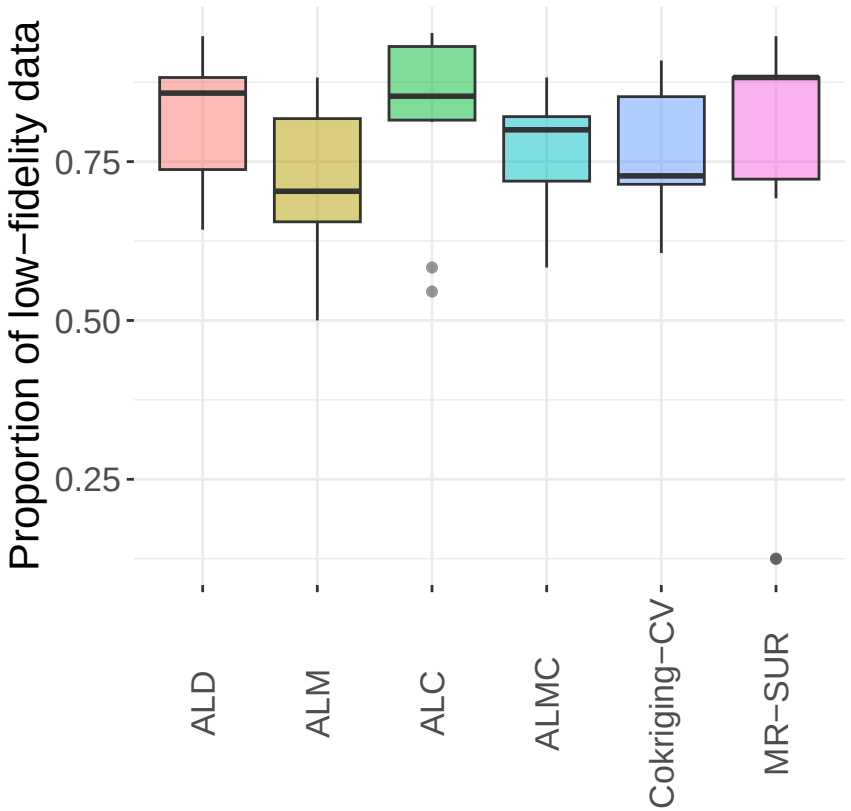
Blade: Active learning performance



RMSE and CRPS for the Park function with respect to the cost.

Blade: Active learning performance





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

Conclusion

- We propose a new model (RNA emulator) and three corresponding active learning strategies (ALD, ALM, ALC, and ALMC).

Conclusion

- We propose a new model (RNA emulator) and three corresponding active learning strategies (ALD, ALM, ALC, and ALMC).
- RNA emulator provides the closed-form expressions for both the posterior mean and variance under common kernel choices.

Conclusion

- We propose a new model (RNA emulator) and three corresponding active learning strategies (ALD, ALM, ALC, and ALMC).
- RNA emulator provides the closed-form expressions for both the posterior mean and variance under common kernel choices.
- Active learnings are facilitated by these closed form expressions.

Conclusion

- We propose a new model (RNA emulator) and three corresponding active learning strategies (ALD, ALM, ALC, and ALMC).
- RNA emulator provides the closed-form expressions for both the posterior mean and variance under common kernel choices.
- Active learnings are facilitated by these closed form expressions.
- Numerical studies and real application show the effectiveness of our approach.
- R package RNAmf is available on CRAN (Heo and Sung, 2024).

Arxiv

Accepted by Technometrics

Statistics > Methodology

[Submitted on 21 Sep 2023 (v1), last revised 4 Jun 2024 (this version, v3)]

Active Learning for a Recursive Non-Additive Emulator for Multi-Fidelity Computer Experiments

Junoh Heo, Chih-Li Sung

Computer simulations have become essential for analyzing complex systems, but high-fidelity simulations often come with significant computational costs. To tackle this challenge, multi-fidelity computer experiments have emerged as a promising approach that leverages both low-fidelity and high-fidelity simulations, enhancing both the accuracy and efficiency of the analysis. In this paper, we introduce a new and flexible statistical model, the Recursive Non-Additive (RNA) emulator, that integrates the data from multi-fidelity computer experiments. Unlike conventional multi-fidelity emulation approaches that rely on an additive auto-regressive structure, the proposed RNA emulator recursively captures the relationships between multi-fidelity data using Gaussian process priors without making the additive assumption, allowing the model to accommodate more complex data patterns. Importantly, we derive the posterior predictive mean and variance of the emulator, which can be efficiently computed in a closed-form manner, leading to significant improvements in computational efficiency. Additionally, based on this emulator, we introduce four active learning strategies that optimize the balance between accuracy and simulation costs to guide the selection of the fidelity level and input locations for the next simulation run. We demonstrate the effectiveness of the proposed approach in a suite of synthetic examples and a real-world problem. An R package RNAmf for the proposed methodology is provided on CRAN.

Comments: 37 pages for the paper including references, 17 pages for supplementary
Subjects: **Methodology (stat.ME)**
Cite as: arXiv:2309.11772 **[stat.ME]**
(or arXiv:2309.11772v3 **[stat.ME]** for this version)
https://doi.org/10.48550/arXiv.2309.11772

Submission history

From: Junoh Heo [view email]
[v1] Thu, 21 Sep 2023 04:11:35 UTC (1,027 KB)
[v2] Tue, 9 Apr 2024 19:43:12 UTC (523 KB)
[v3] Tue, 4 Jun 2024 17:38:27 UTC (534 KB)

R package (CRAN)

RNAmf: Recursive Non-Additive Emulator for Multi-Fidelity Data

Performs RNA emulation and active learning proposed by Heo and Sung (2023+) <doi:10.48550/arXiv.2309.11772> for multi-fidelity computer experiments. The RNA emulator is particularly useful when the simulations with different fidelity level are nonlinearly correlated. The hyperparameters in the model are estimated by maximum likelihood estimation.

Version:	0.1.2
Imports:	plgp, stats, lhs, doParallel, foreach
Suggests:	knitr, rmarkdown
Published:	2024-03-22
DOI:	10.32614/CRAN.package.RNAmf
Author:	Junoh Heo [aut, cre], Chih-Li Sung [aut]
Maintainer:	Junoh Heo <heojunoh at msu.edu>
License:	MIT + file LICENSE
NeedsCompilation:	no
CRAN checks:	RNAmf results

Documentation:

Reference manual: RNAmf.pdf

Downloads:

Package source:	RNAmf_0.1.2.tar.gz
Windows binaries:	r-devel: RNAmf_0.1.2.zip, r-release: RNAmf_0.1.2.zip, r-oldrel: RNAmf_0.1.2.zip
macOS binaries:	r-release (arm64): RNAmf_0.1.2.tgz, r-oldrel (arm64): RNAmf_0.1.2.tgz, r-release (x86_64): RNAmf_0.1.2.tgz, r-oldrel (x86_64): RNAmf_0.1.2.tgz
Old sources:	RNAmf archive

Linking:

Reproducibility

README

Active Learning for a Recursive Non-Additive Emulator for Multi-Fidelity Computer Experiments (Reproducibility)

Junoh Heo, Chih-Li Sung Jun 3, 2024

This instruction aims to reproduce the results in the paper "*Active Learning for a Recursive Non-Additive Emulator for Multi-Fidelity Computer Experiments*".

The following results are reproduced in this file

- Section 5.1: Figures 7, S14, and 8
- Section 5.2: Figures 9, 10, S15, and S16
- Section 6: Figure 11, 12 and 13

The approximate running times for each section are as follows:

- Section 5.1: ~9 hours
- Section 5.2: ~48 hours

Cohn, D. (1993). Neural network exploration using optimal experiment design. *Advances in Neural Information Processing Systems*, 6:1071–1083.

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378.

Kennedy, M. C. and O'Hagan, A. (2000). Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13.

Kyzyurova, K. N., Berger, J. O., and Wolpert, R. L. (2018). Coupling computer models through linking their statistical emulators. *SIAM/ASA Journal on Uncertainty Quantification*, 6(3):1151–1171.

Le Gratiet, L. (2013). Bayesian analysis of hierarchical multifidelity codes. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):244–269.

Le Gratiet, L. and Cannamela, C. (2015). Cokriging-based sequential design strategies using fast cross-validation techniques for multi-fidelity computer codes. *Technometrics*, 57(3):418–427.

Le Gratiet, L. and Garnier, J. (2014). Recursive co-kriging model for design of computer experiments with multiple levels of fidelity. *International Journal for Uncertainty Quantification*, 4(5):365–386.

MacKay, D. J. C. (1992). Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604.

Ming, D. and Guillas, S. (2021). Linked Gaussian process emulation for systems of computer models using Matérn kernels and adaptive design. *SIAM/ASA Journal on Uncertainty Quantification*, 9(4):1615–1642.

Park, J. S. (1991). *Tuning Complex Computer Codes to Data and Optimal Designs*. University of Illinois at Urbana-Champaign.

Perdikaris, P., Raissi, M., Damianou, A., Lawrence, N. D., and Karniadakis, G. E. (2017). Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2198):20160751.

Qian, P. Z. G. (2009). Nested latin hypercube designs. *Biometrika*, 96(4):957–970.

Qian, P. Z. G., Ai, M., and Wu, C. F. J. (2009). Construction of nested space-filling designs. *Annals of Statistics*, 37(6A):3616–3643.

Qian, P. Z. G. and Wu, C. F. J. (2008). Bayesian hierarchical modeling for integrating low-accuracy and high-accuracy experiments. *Technometrics*, 50(2):192–204.

Qian, Z., Seepersad, C. C., Joseph, V. R., Allen, J. K., and Wu, C. F. J. (2006). Building surrogate models based on detailed and approximate simulations. *Journal of Mechanical Design*, 128(4):668–677.

Stroh, R., Bect, J., Demeyer, S., Fischer, N., Marquis, D., and Vazquez, E. (2022). Sequential design of multi-fidelity computer experiments: maximizing the rate of stepwise uncertainty reduction. *Technometrics*, 64(2):199–209.

Xiong, S., Qian, P. Z. G., and Wu, C. F. J. (2013). Sequential design and analysis of high-accuracy and low-accuracy computer codes. *Technometrics*, 55(1):37–46.

Thank You!

Thank NSF DMS 2113407 and 2338018 for supporting this work.

MICHIGAN STATE
UNIVERSITY