Diffusion Non-Additive Model for Multi-Fidelity Simulations with Tunable Precision

Chih-Li Sung

Department of Statistics and Probability
Michigan State University

2026 Quality and Productivity Research Conference (QPRC)
Santa Fe, New Mexico, USA
June 23, 2026

MICHIGAN STATE
UNIVERSITY

Outline

Collaborators and Students

Junoh Heo
Wake Forest

Romain Boutelet
MSU

Wenjia Wang
NUS

Chih-Li Sung
MSU

Multi-Fidelity Simulations

- Computer models are widely used to study real-world systems and phenomena.
- Computer simulations numerically solve these models using methods such as finite elements or finite differences.

Multi-Fidelity Simulations

- Computer models are widely used to study real-world systems and phenomena.
- Computer simulations numerically solve these models using methods such as finite elements or finite differences.
- Simulations are often available at different levels of fidelity:
  - High-fidelity simulation: costly but close to the truth

Multi-Fidelity Simulations

- Computer models are widely used to study real-world systems and phenomena.
- Computer simulations numerically solve these models using methods such as finite elements or finite differences.
- Simulations are often available at different levels of fidelity:
  - High-fidelity simulation: costly but close to the truth
  - Low-fidelity simulation: cheaper but less accurate

Multi-Fidelity Simulations

- Computer models are widely used to study real-world systems and phenomena.
- Computer simulations numerically solve these models using methods such as finite elements or finite differences.
- Simulations are often available at different levels of fidelity:
  - High-fidelity simulation: costly but close to the truth
  - Low-fidelity simulation: cheaper but less accurate
  - Intermediate-fidelity simulation: between high and low fidelity

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 using the finite element method.

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 using the finite element method.
- **Input**: $\mathbf{x} = (x_1, x_2) = (\text{pressure}, \text{suction})$

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 using the finite element method.
- **Input**: $\mathbf{x} = (x_1, x_2) = (\text{pressure}, \text{suction})$
- **Output**: $f(\mathbf{x})$: **maximum thermal stress**

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 using the finite element method.
- **Input**: $\mathbf{x} = (x_1, x_2) = (\text{pressure}, \text{suction})$
- **Output**: $f(\mathbf{x})$: **maximum thermal stress**
- Example: $\mathbf{x} = (0.23, 0.71)$

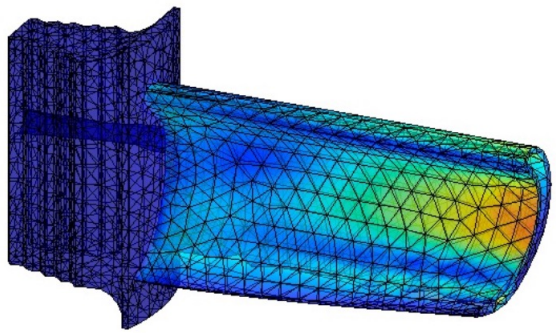
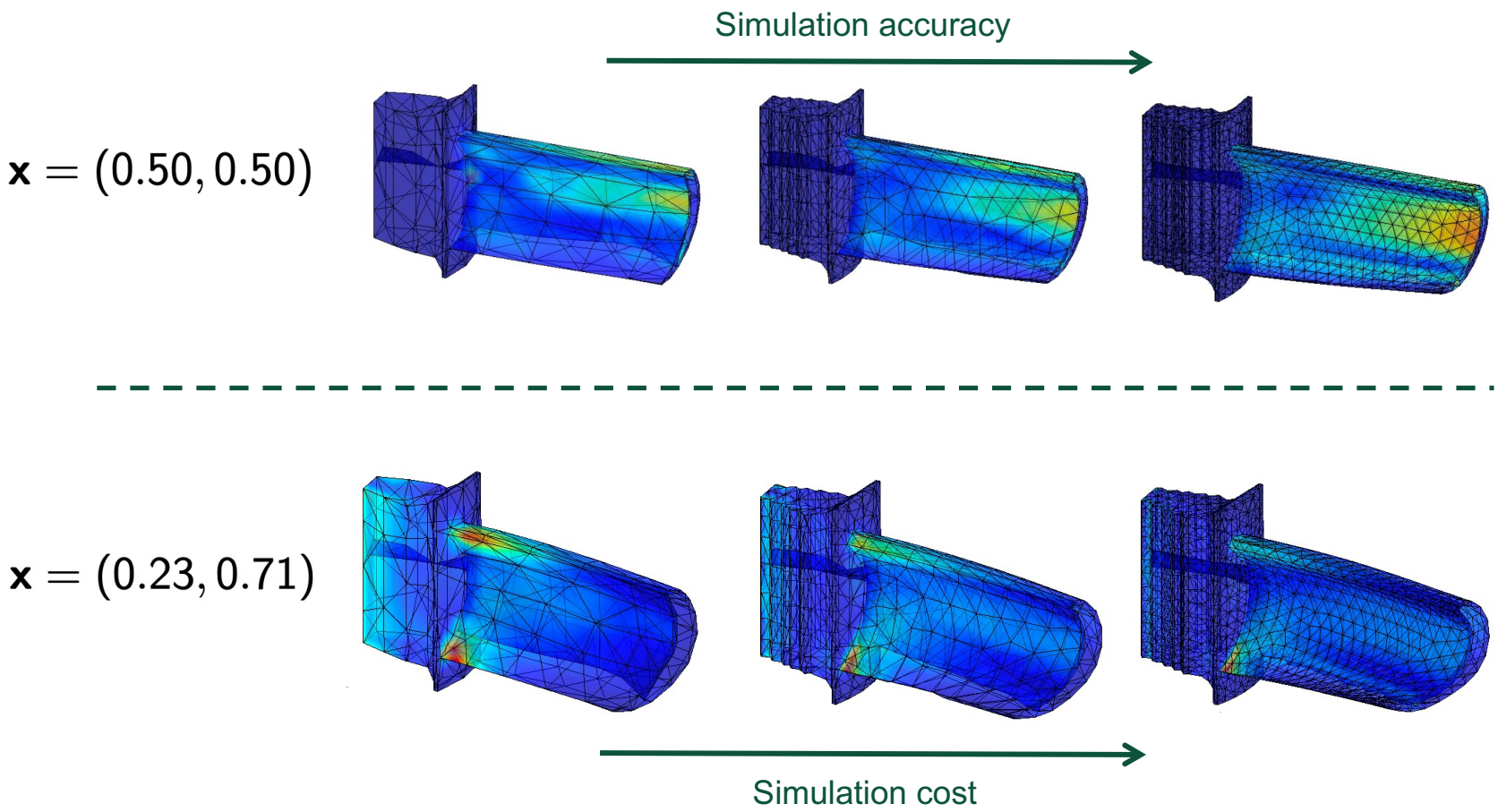


Figure: Thermal stress profile with $f(0.23, 0.71) = 20.3$.

Multi-Fidelity Simulations

less accurate but cheaper

accurate but expensive



Multi-Fidelity Emulation

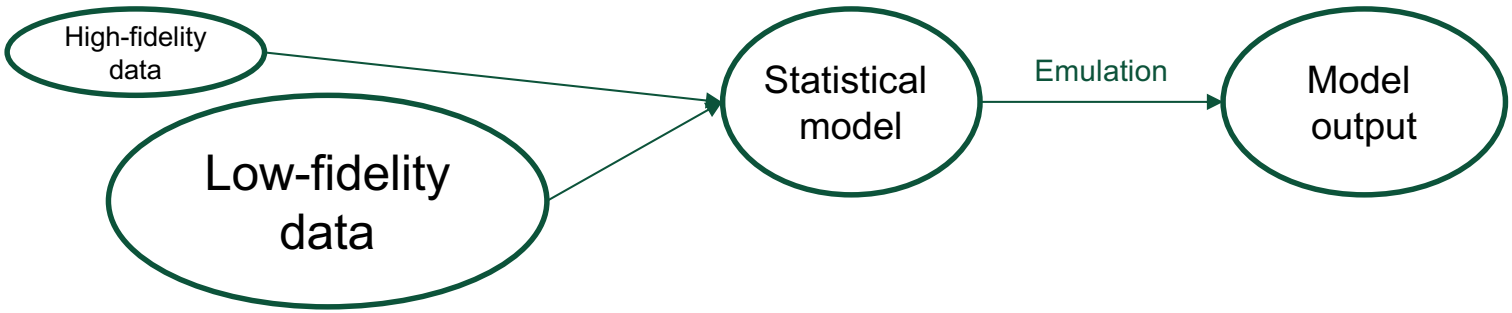
- Can we leverage both low-fidelity and high-fidelity simulations to build a statistical emulator?

Multi-Fidelity Emulation

- Can we leverage both low-fidelity and high-fidelity simulations to build a statistical emulator?
- A statistical emulator, also known as a surrogate model, approximates the output of the high-fidelity simulator:

$$\hat{f}(\mathbf{x}) \approx f(\mathbf{x}),$$

where $f(\mathbf{x})$ is the high-fidelity simulator and $\mathbf{x}$ is the input.



Jet Blade Simulations

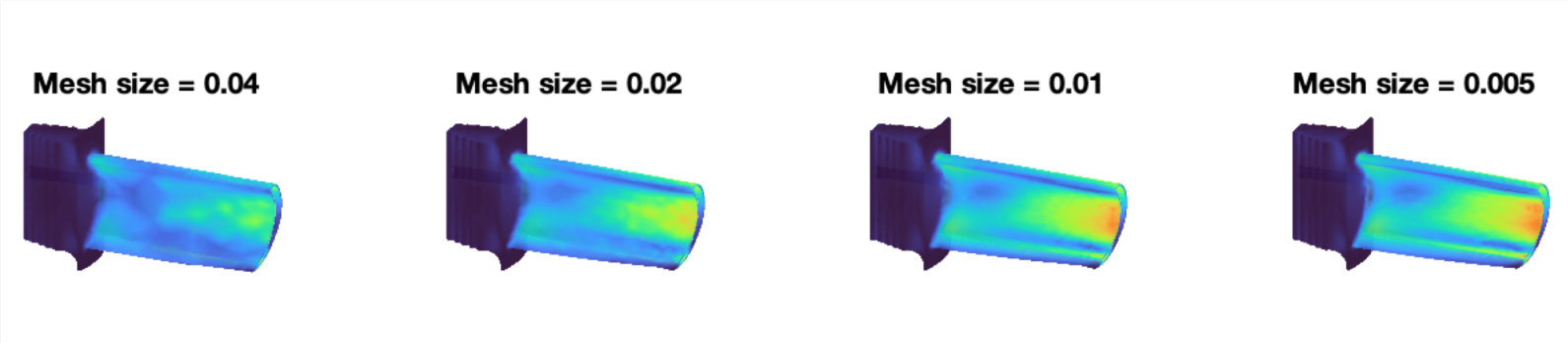


Figure: Jet blade simulations with different mesh configurations

- The fidelity level is controlled by the mesh size.

Jet Blade Simulations

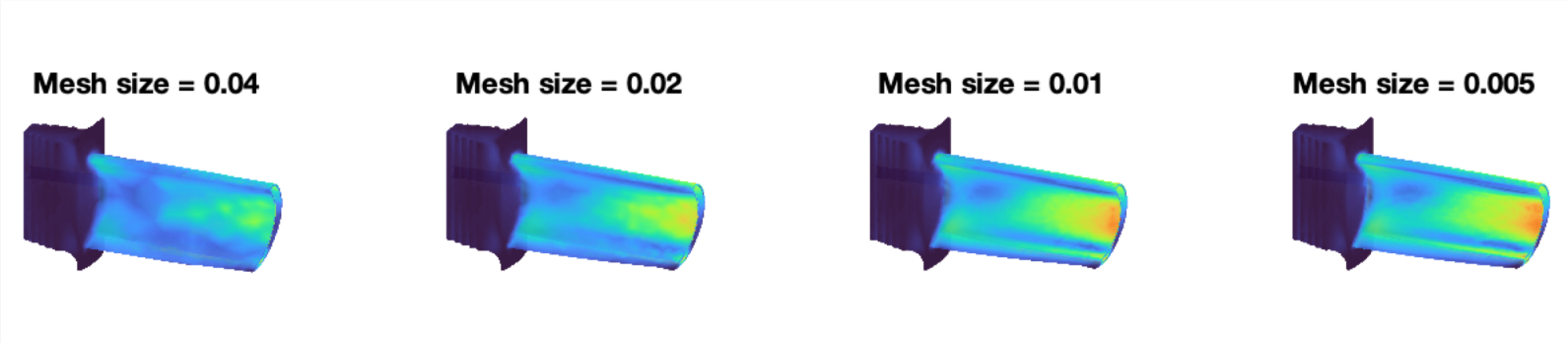


Figure: Jet blade simulations with different mesh configurations

- The fidelity level is controlled by the mesh size.
- Q1: Can we account for the mesh size and extrapolate to the **exact finite element solution** (mesh size $= 0$)?

Jet Blade Simulations

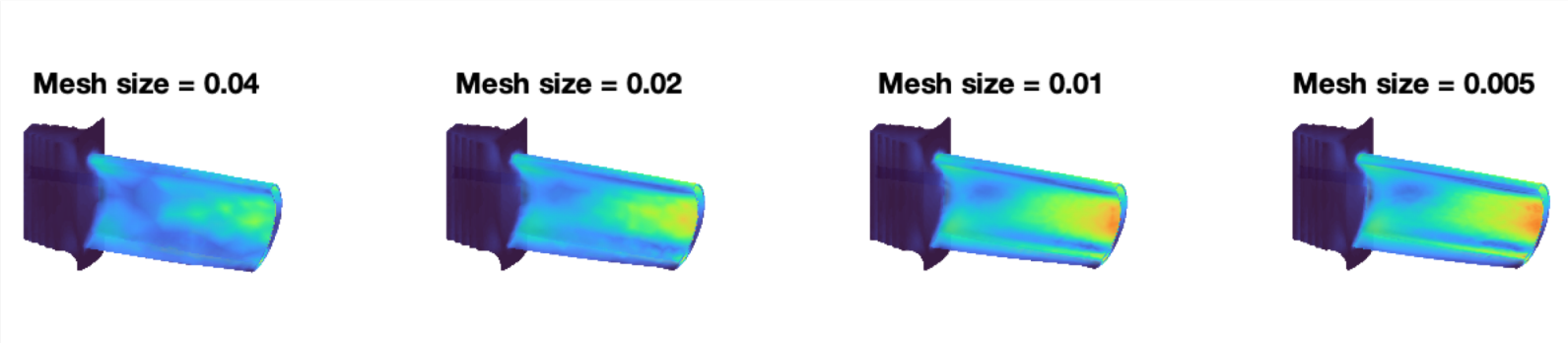


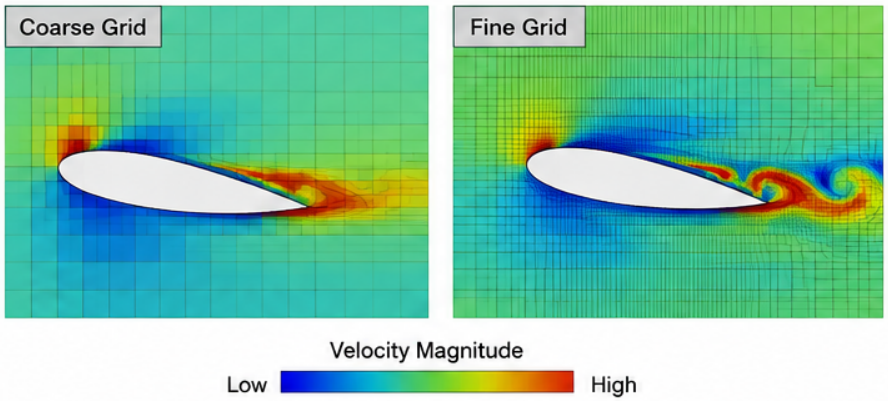
Figure: Jet blade simulations with different mesh configurations

- The fidelity level is controlled by the mesh size.
- Q1: Can we account for the mesh size and extrapolate to the **exact finite element solution** (mesh size $= 0$)?
- Q2: How should we choose both the input locations and the mesh sizes when running simulations?

Other Applications

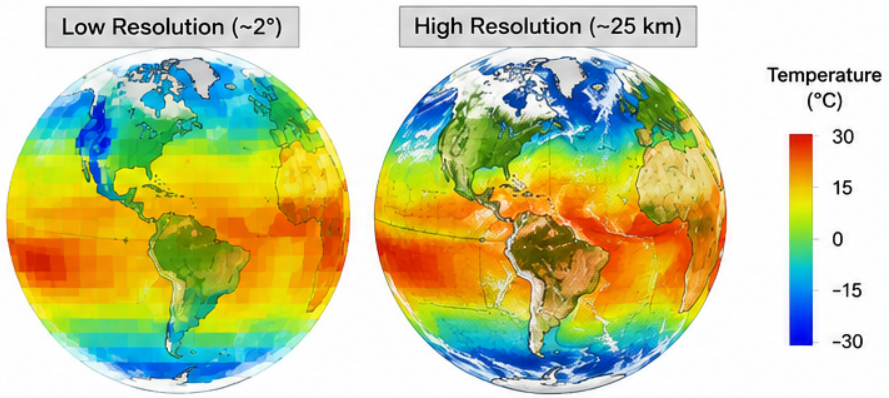
Computational Fluid Dynamics (CFD)

coarse-grid vs. fine-grid simulations



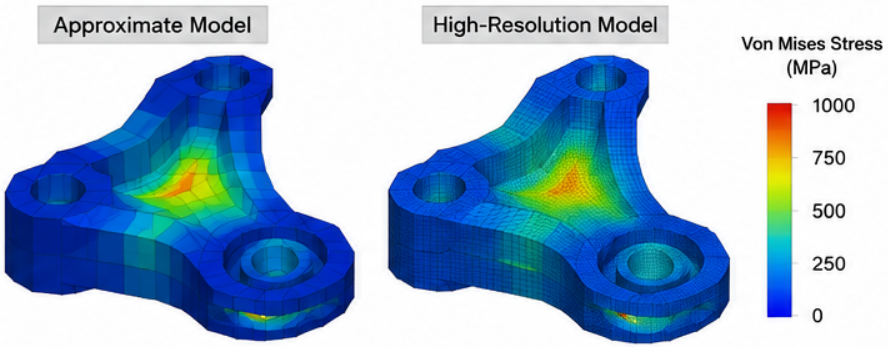
Climate Modeling

low-resolution vs. high-resolution models



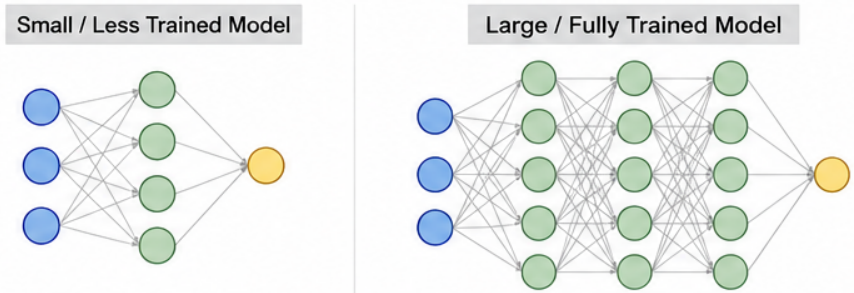
Materials and Manufacturing

approximate vs. high-resolution simulations



Machine Learning / AI

small models vs. fully trained systems



Common theme: combine **cheap approximate** information with **expensive accurate** information.

Problem Setup

Let $f(t_l, \mathbf{x})$ denote the scalar output of a simulator with input $\mathbf{x} \in \Omega \subseteq \mathbb{R}^d$ and tuning parameter t_l.

- Simulations are available at L fidelity levels:

$$t_1 > t_2 > \cdots > t_L > 0.$$

- Smaller t_l corresponds to higher fidelity but higher computational cost.
- **Goal**: build an emulator for the exact solution

$$f(0, \mathbf{x}),$$

which is typically infeasible to evaluate directly.

Training Data

At fidelity level l, simulations are run at design points

$$\mathcal{X}_l = \{\mathbf{x}_i^{[l]}\}_{i=1}^{n_l} \subset \Omega.$$

The corresponding outputs are

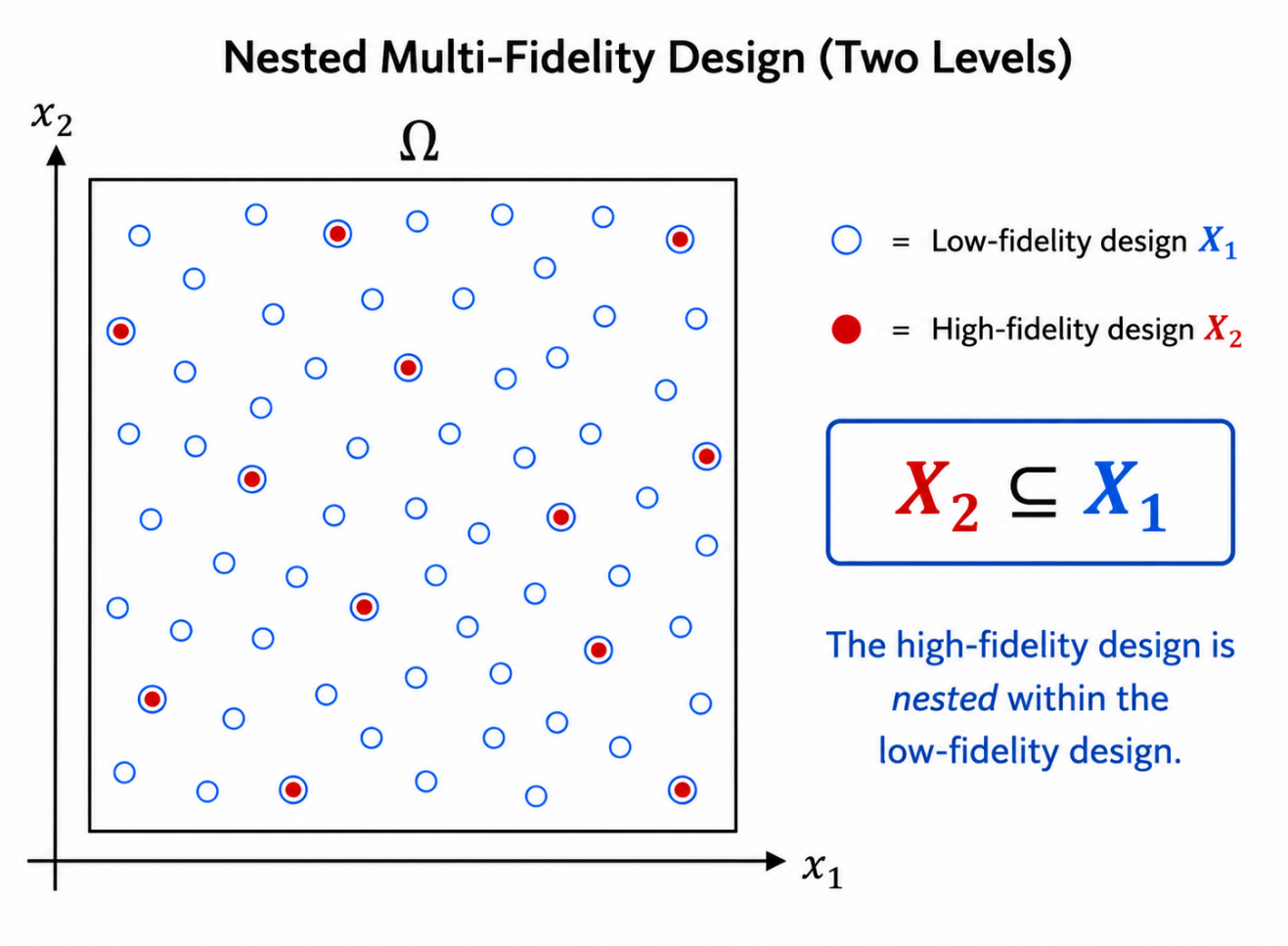
$$\mathbf{y}_l = \{f(t_l, \mathbf{x})\}_{\mathbf{x}\in\mathcal{X}_l}, \qquad y_i^{[l]} = f(t_l, \mathbf{x}_i^{[l]}).$$

- $\mathcal{X}_l$: design points at fidelity level l.
- $\mathbf{y}_l$: simulation outputs at fidelity level l.
- The emulator learns from all fidelity levels jointly.

Nested Multi-Fidelity Design

We first assume a sequentially nested design:

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



Existing Methods: Discrete Fidelity Levels

- The canonical approach is the auto-regressive (AR) model (Kennedy and O'Hagan, 2000):

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

  where $f_l(\mathbf{x})$ is the scalar simulation output at fidelity level l, and each Z_l is modeled as a Gaussian process (GP).

Existing Methods: Discrete Fidelity Levels

- The canonical approach is the auto-regressive (AR) model (Kennedy and O'Hagan, 2000):

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

  where $f_l(\mathbf{x})$ is the scalar simulation output at fidelity level l, and each Z_l is modeled as a Gaussian process (GP).

- The recursive non-additive (RNA) model (Heo and Sung, 2025) generalizes the AR structure:

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

  where $W_l(\cdot,\cdot)$ is modeled as a GP over the augmented input $\big(\mathbf{x}, f_{l-1}(\mathbf{x})\big)$, allowing a flexible nonlinear relationship between adjacent fidelity levels.

Existing Methods: Continuous Fidelity Level

- Model the simulator using both the input $\mathbf{x}$ and a continuous fidelity parameter t.

Existing Methods: Continuous Fidelity Level

- Model the simulator using both the input $\mathbf{x}$ and a continuous fidelity parameter t.
- Tuo et al. (2014) decomposes the simulator as

$$f(\mathbf{x}, t) = \underbrace{\varphi(\mathbf{x})}_{\text{exact solution } f(\mathbf{x},0)} + \underbrace{\delta(\mathbf{x}, t)}_{\text{numerical error}},$$

where φ and δ are modeled as independent Gaussian processes.

Existing Methods: Continuous Fidelity Level

- Model the simulator using both the input $\mathbf{x}$ and a continuous fidelity parameter t.
- Tuo et al. (2014) decomposes the simulator as

$$f(\mathbf{x}, t) = \underbrace{\varphi(\mathbf{x})}_{\text{exact solution } f(\mathbf{x},0)} + \underbrace{\delta(\mathbf{x}, t)}_{\text{numerical error}},$$

  where φ and δ are modeled as independent Gaussian processes.
- Boutelet and Sung (2025) extends this framework with an adaptive nonstationary kernel inspired by fractional Brownian motion, together with active learning for selecting both $\mathbf{x}$ and t.

Heo, J., Boutelet, R., Wang, W. and **Sung, C.-L.** (2025)

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

MICHIGAN STATE
UNIVERSITY

Diffusion Non-Additive (DNA) Model

Diffusion Non-Additive (DNA) Model

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

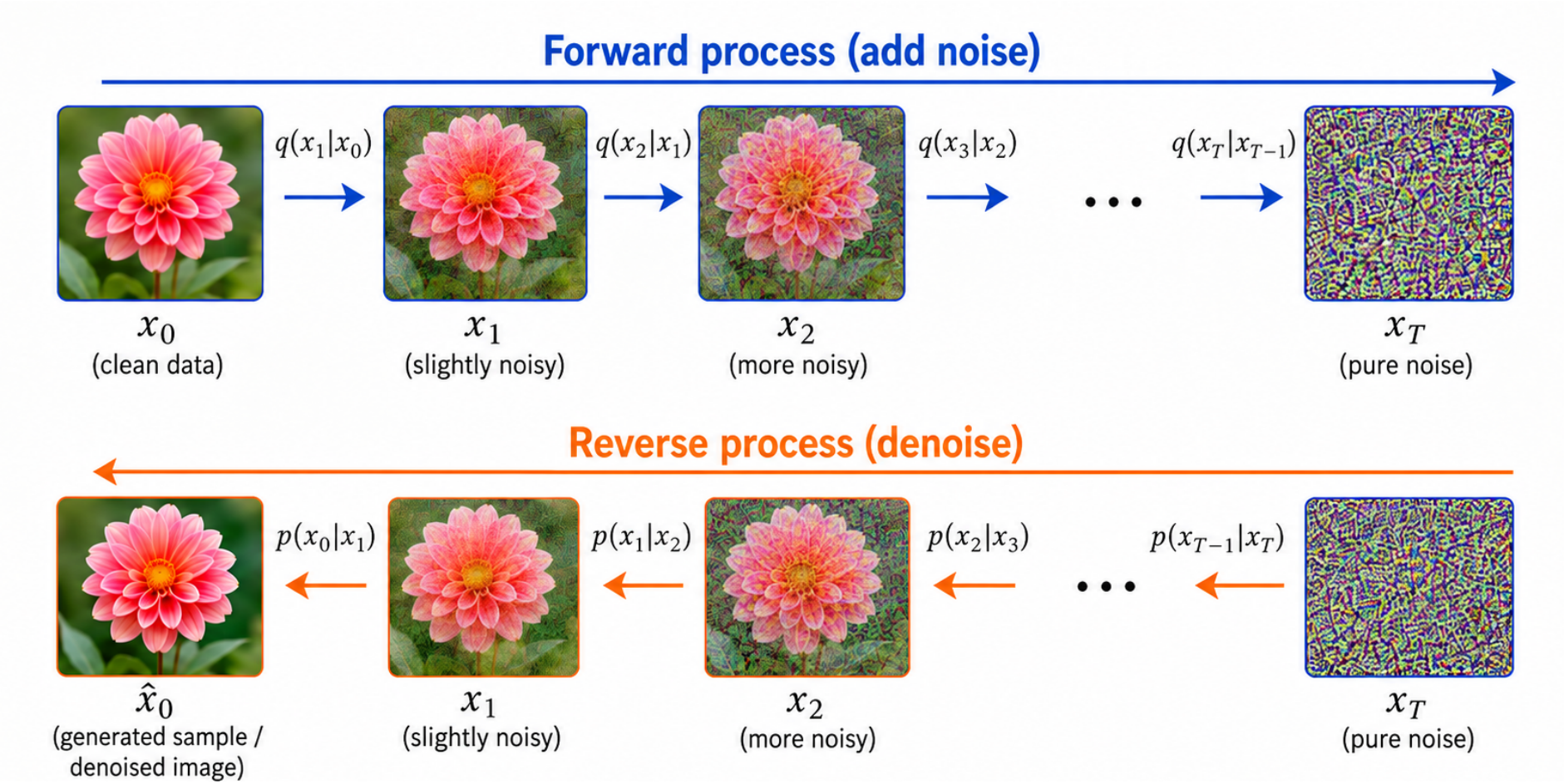
$$f(\mathbf{x}, t_l) = W(t_l, \mathbf{x}, f(\mathbf{x}, t_{l-1})), \quad l = 2, \ldots, L.$$

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

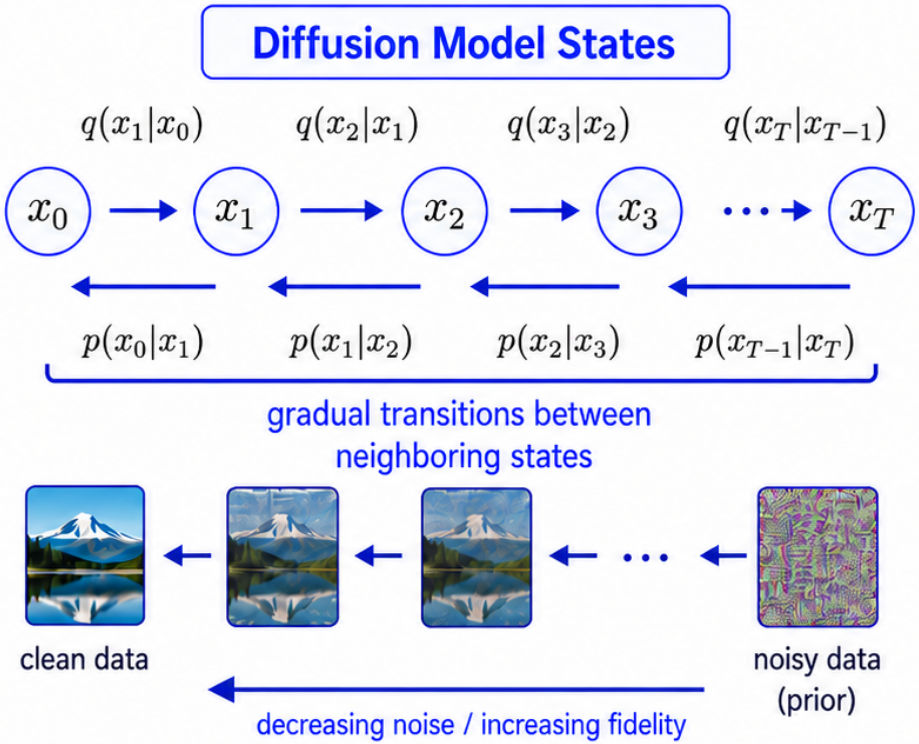
Remark 1: The relationships between fidelity levels are no longer additive.

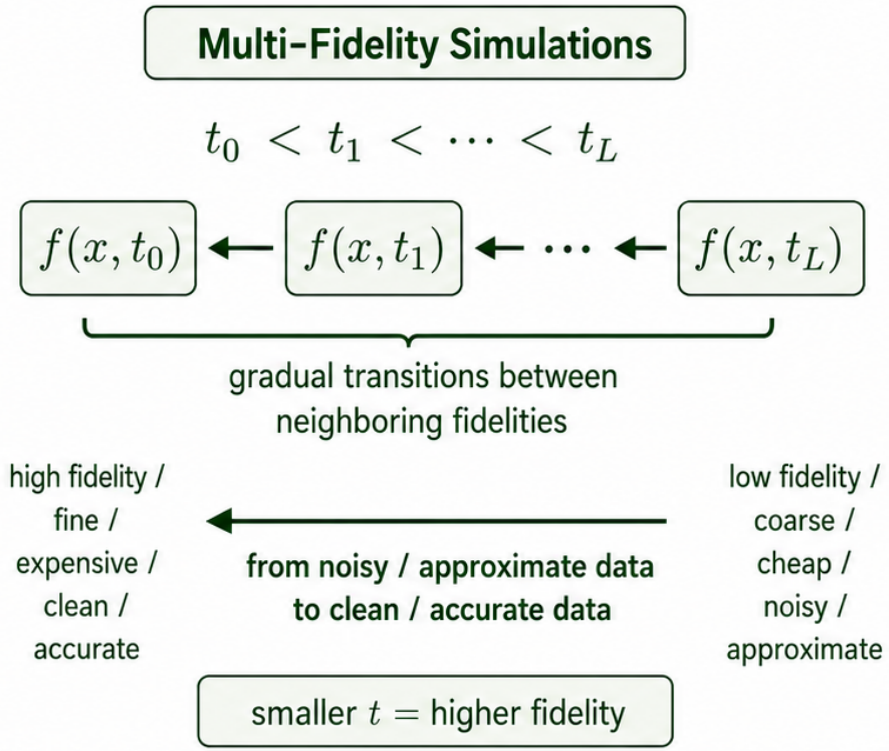
Remark 2: The term "diffusion" is inspired by its analogy in generative models.

Diffusion Model: Forward and Reverse Processes



Inspiration from Diffusion Models





Gaussian Process Emulator

Key idea

A Gaussian process defines a probability model over an unknown function:

$$g(\mathbf{x}) \sim \mathcal{GP}\big(\alpha, K_\theta(\mathbf{x}, \mathbf{x}')\big).$$

- For observed outputs at design points $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$,

$$\mathbf{y} = \big(g(\mathbf{x}_1), \ldots, g(\mathbf{x}_n)\big)^T \sim \mathcal{N}_n(\alpha \mathbf{1}_n, \mathbf{K}),$$

  where $\mathbf{K}_{ij} = K_\theta(\mathbf{x}_i, \mathbf{x}_j)$.

- After observing data, the posterior predictive distribution $g(\mathbf{x})|\mathbf{y}$ is Gaussian, giving closed-form prediction and uncertainty quantification.

In our setting, GP priors are placed on transition functions between fidelity levels.

DNA Model

Diffusion Non-Additive Model

$$\begin{cases} f_1(\mathbf{x}) = W_1(\mathbf{x}), \\ f_l(\mathbf{x}) = W(t_l, \mathbf{x}, f_{l-1}(\mathbf{x})), \quad \text{for} \quad l = 2, \ldots, L, \end{cases}$$

where

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

$$W(t, \mathbf{x}, y) \sim \mathcal{GP}\left(\alpha, \tau^2 K\left((t, \mathbf{x}, y), (t', \mathbf{x}', y')\right)\right).$$

- W: unknown transition function from one fidelity level to the next.
- The GP prior on W gives both prediction and uncertainty quantification.

Diffusion Representation

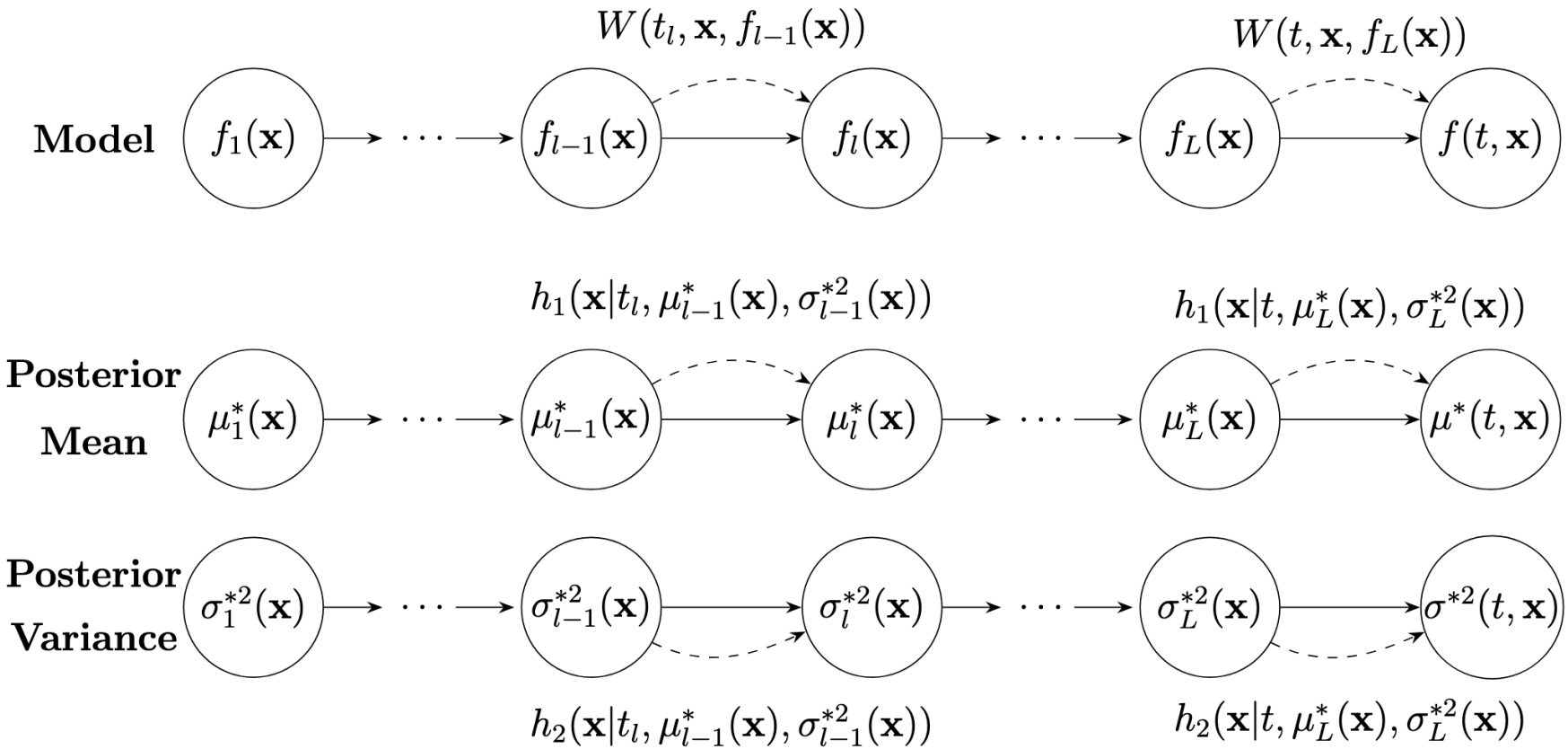


Figure: The Markov chain representation of the hierarchical modeling structure.

Likelihood Construction

The lowest-fidelity output is modeled as

$$\mathbf{y}_1 \sim \mathcal{N}_{n_1}\left(\alpha_1 \mathbf{1}_{n_1}, \tau_1^2 \mathbf{K}_1\right),$$

and for the remaining fidelity levels,

$$Y_{-1} = \begin{bmatrix} \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_L \end{bmatrix} \sim \mathcal{N}_{N_{-1}}\left(\alpha \mathbf{1}_{N_{-1}}, \tau^2 \mathbf{K}\right), \qquad N_{-1} = \sum_{l=2}^{L} n_l.$$

where $\mathbf{K}$ is an $N_{-1} \times N_{-1}$ matrix with elements

$$\mathbf{K}_{ij} = K\left(((\mathbf{t}_{-1})_i, (\mathbf{X}_{-1})_i, (Y_{-L})_i), ((\mathbf{t}_{-1})_j, (\mathbf{X}_{-1})_j, (Y_{-L})_j)\right),$$

and $Y_{-L} = (\mathbf{y}_1[1:n_2]^T, \mathbf{y}_2[1:n_3]^T, \ldots, \mathbf{y}_{L-1}[1:n_L]^T)^T$.

Parameter Estimation

- The model parameters are estimated by maximum likelihood.
- The closed-form estimates include

$$\hat{\alpha} = \frac{Y_{-1}^T \mathbf{K}^{-1} Y_{-1}}{\mathbf{1}_{N_{-1}}^T \mathbf{K}^{-1} \mathbf{1}_{N_{-1}}},$$

and

$$\hat{\tau}^2 = \frac{(Y_{-1} - \hat{\alpha}\mathbf{1}_{N_{-1}})^T \mathbf{K}^{-1} (Y_{-1} - \hat{\alpha}\mathbf{1}_{N_{-1}})}{N_{-1}}.$$

- Kernel parameters are estimated by maximizing the profile likelihood.
- Closed-form gradients improve optimization stability and speed.

Kernel Functions: Lowest-Fidelity Layer

- For the lowest-fidelity layer, K_1 can be chosen from standard GP kernels, such as the squared exponential or Matérn kernel.
- For example, the squared exponential kernel is

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

- This kernel models the spatial dependence of the lowest-fidelity simulator output.

Kernel Functions: Transition Layer

- For the transition function $W(t, \mathbf{x}, y)$, we use a nonseparable kernel:

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

Kernel Functions: Transition Layer

- For the transition function $W(t, \mathbf{x}, y)$, we use a nonseparable kernel:

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

- The parameter $0 \leq \beta \leq 1$ controls the interaction between the fidelity parameter t and the augmented input $(\mathbf{x}, y)$.

- When $\beta = 0$, the kernel becomes separable.

Posterior Prediction: The Challenge

- Let $f_l(\mathbf{x}) := f(t_l, \mathbf{x})$.

Posterior Prediction: The Challenge

- Let $f_l(\mathbf{x}) := f(t_l, \mathbf{x})$.
- Conditional on the previous fidelity output, the GP model gives

$$f_l(\mathbf{x}) \mid Y_N, f_{l-1}(\mathbf{x}) \sim \mathcal{N}\left(\mu_l(\mathbf{x}, f_{l-1}(\mathbf{x})), \sigma_l^2(\mathbf{x}, f_{l-1}(\mathbf{x}))\right),$$

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

Posterior Prediction: The Challenge

- Let $f_l(\mathbf{x}) := f(t_l, \mathbf{x})$.
- Conditional on the previous fidelity output, the GP model gives

$$f_l(\mathbf{x}) \mid Y_N, f_{l-1}(\mathbf{x}) \sim \mathcal{N}\left(\mu_l(\mathbf{x}, f_{l-1}(\mathbf{x})), \sigma_l^2(\mathbf{x}, f_{l-1}(\mathbf{x}))\right),$$

  for $l = 2, \ldots, L$.
- To predict the exact solution, for $0 \leq t < t_L$,

$$f(t, \mathbf{x}) \mid Y_N, f_L(\mathbf{x}) \sim \mathcal{N}\left(\mu_{L+1}(t, \mathbf{x}, f_L(\mathbf{x})), \sigma_{L+1}^2(t, \mathbf{x}, f_L(\mathbf{x}))\right).$$

Posterior Prediction: The Challenge

- Let $f_l(\mathbf{x}) := f(t_l, \mathbf{x})$.
- Conditional on the previous fidelity output, the GP model gives

$$f_l(\mathbf{x}) \mid Y_N, f_{l-1}(\mathbf{x}) \sim \mathcal{N}\left(\mu_l(\mathbf{x}, f_{l-1}(\mathbf{x})), \sigma_l^2(\mathbf{x}, f_{l-1}(\mathbf{x}))\right),$$

  for $l = 2, \ldots, L$.
- To predict the exact solution, for $0 \leq t < t_L$,

$$f(t, \mathbf{x}) \mid Y_N, f_L(\mathbf{x}) \sim \mathcal{N}\left(\mu_{L+1}(t, \mathbf{x}, f_L(\mathbf{x})), \sigma_{L+1}^2(t, \mathbf{x}, f_L(\mathbf{x}))\right).$$

- The marginal posterior $p(f(t, \mathbf{x}) \mid Y_N)$ requires integrating over the unknown intermediate outputs.

Posterior Prediction: The Challenge

- Let $f_l(\mathbf{x}) := f(t_l, \mathbf{x})$.
- Conditional on the previous fidelity output, the GP model gives

$$f_l(\mathbf{x}) \mid Y_N, f_{l-1}(\mathbf{x}) \sim \mathcal{N}\left(\mu_l(\mathbf{x}, f_{l-1}(\mathbf{x})), \sigma_l^2(\mathbf{x}, f_{l-1}(\mathbf{x}))\right),$$

  for $l = 2, \ldots, L$.
- To predict the exact solution, for $0 \leq t < t_L$,

$$f(t, \mathbf{x}) \mid Y_N, f_L(\mathbf{x}) \sim \mathcal{N}\left(\mu_{L+1}(t, \mathbf{x}, f_L(\mathbf{x})), \sigma_{L+1}^2(t, \mathbf{x}, f_L(\mathbf{x}))\right).$$

- The marginal posterior $p(f(t, \mathbf{x}) \mid Y_N)$ requires integrating over the unknown intermediate outputs.
- A direct Monte Carlo approximation is possible, but can be computationally demanding.

Closed-Form Recursive Prediction

Key Result

Suppose the design points are nested. For $0 \leq t < t_L$, the posterior mean and variance of the exact-solution emulator can be computed recursively:

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

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

For $l = 2, \ldots, L$,

$$\mu_l^*(\mathbf{x}) = h_1\left(\mathbf{x} \mid t_l, \mu_{l-1}^*(\mathbf{x}), \sigma_{l-1}^{*2}(\mathbf{x})\right),$$

$$\sigma_l^{*2}(\mathbf{x}) = h_2\left(\mathbf{x} \mid t_l, \mu_{l-1}^*(\mathbf{x}), \sigma_{l-1}^{*2}(\mathbf{x})\right).$$

h_1 and h_2 have closed-form expressions, avoiding expensive MC integration.

Diffusion Representation

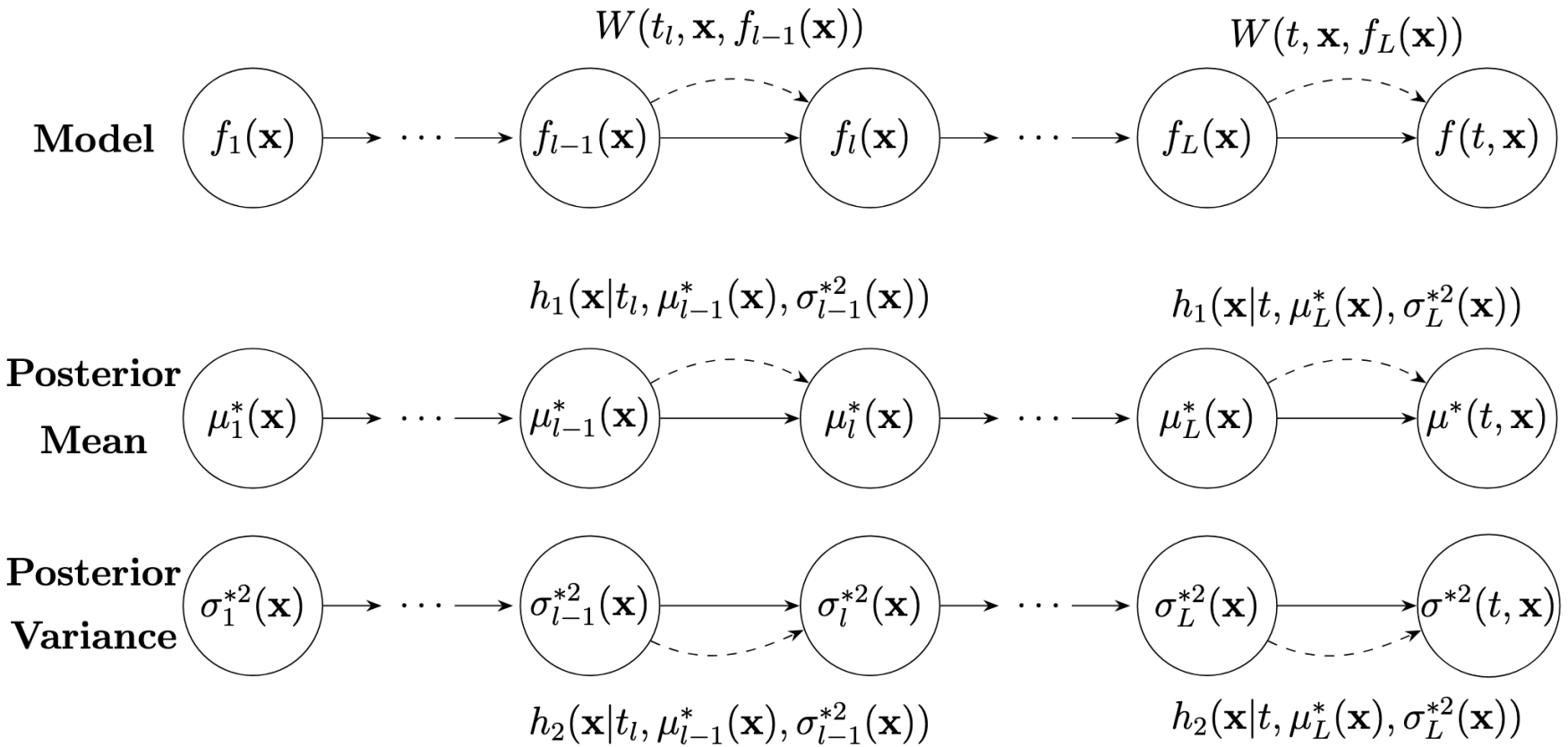
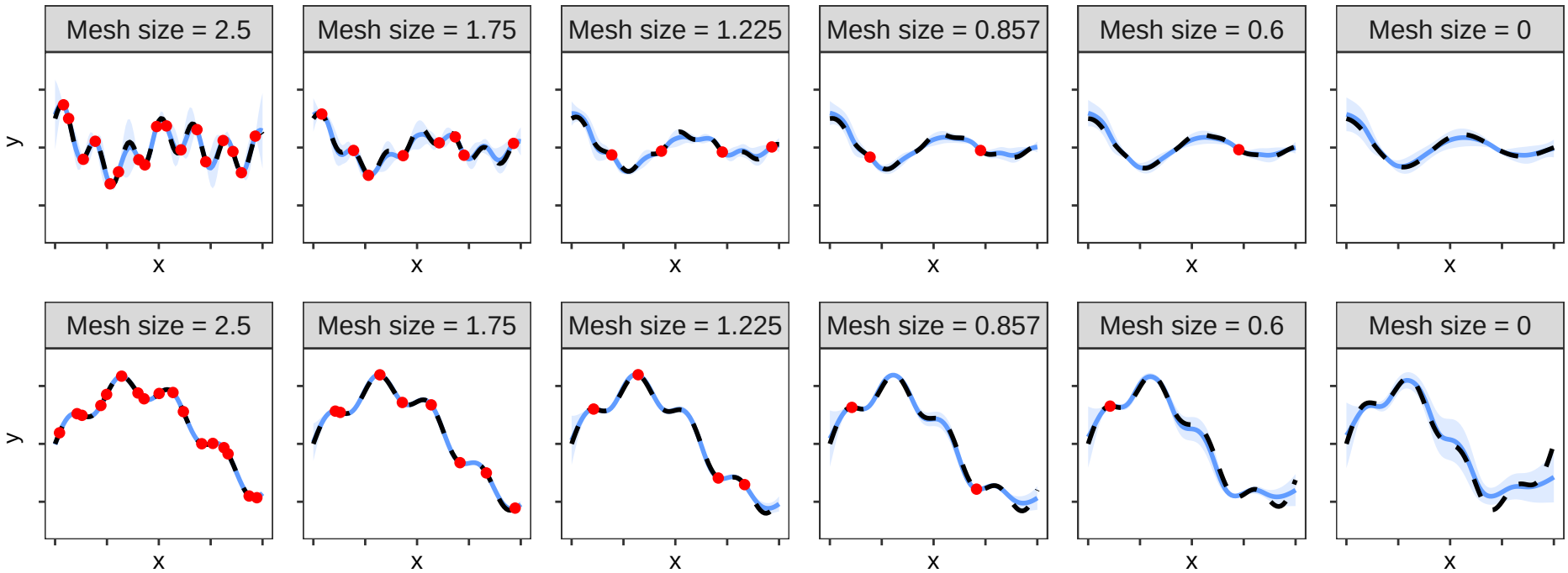


Figure: The Markov chain representation of the hierarchical modeling structure.

A Toy Example

Example 1: $f_1(x,t) = \exp(-1.4x)\cos(3.5\pi x) + \dfrac{t^2}{10}\sin(40x), \quad x \in [0,1],$

Example 2: $f_2(x,t) = \sin\left(\dfrac{10\pi x}{5+t}\right) + 0.2\sin(8\pi x), \quad x \in [0,1].$



Red dots: observed runs; **dashed black**: true function; **blue curve**: posterior mean.

Theoretical Question

- The DNA model recursively propagates information across fidelity levels:

$$f_1(\mathbf{x}) \longrightarrow f_2(\mathbf{x}) \longrightarrow \cdots \longrightarrow f_L(\mathbf{x}) \longrightarrow f(0, \mathbf{x}).$$

Theoretical Question

- The DNA model recursively propagates information across fidelity levels:

$$f_1(\mathbf{x}) \longrightarrow f_2(\mathbf{x}) \longrightarrow \cdots \longrightarrow f_L(\mathbf{x}) \longrightarrow f(0, \mathbf{x}).$$

- A natural question is:

How does the prediction error depend on the design and the fidelity levels?

Theoretical Question

- The DNA model recursively propagates information across fidelity levels:

$$f_1(\mathbf{x}) \longrightarrow f_2(\mathbf{x}) \longrightarrow \cdots \longrightarrow f_L(\mathbf{x}) \longrightarrow f(0, \mathbf{x}).$$

- A natural question is:

  How does the prediction error depend on the design and the fidelity levels?

- The theory decomposes the error into two parts:
  - extrapolation error: due to predicting from t_L to $t = 0$;
  - interpolation error: due to finite samples at each fidelity level.

Main Error Bound

Error decomposition

For all $\mathbf{x} \in \Omega$,

$$|f(0,\mathbf{x}) - \hat{f}(0,\mathbf{x})| \leq \underbrace{(B_t + \Lambda_{L+1})t_L}_{\text{extrapolation error}} + \underbrace{\sum_{l=2}^{L} A_l n_l^{-\eta/d} + A_1 n_1^{-\eta/d}}_{\text{interpolation errors across fidelity levels}},$$

where A_l depends on the kernel, Lipschitz constants, and RKHS norms.

- Smaller t_L reduces the extrapolation error.
- Larger n_l reduces the interpolation error at fidelity level l.
- The rate depends on dimension d and kernel smoothness η.

From Error Bound to Sample Allocation

- Suppose the computational cost per run at fidelity level l is C_l.

From Error Bound to Sample Allocation

- Suppose the computational cost per run at fidelity level l is C_l.
- The error bound suggests the optimization problem:

$$\min_{n_1,\ldots,n_L} \sum_{l=1}^{L} \lambda_{\max}^{L-l+1} n_l^{-\eta/d} \quad \text{subject to} \quad \sum_{l=1}^{L} C_l n_l \leq C_{\text{total}}.$$

From Error Bound to Sample Allocation

- Suppose the computational cost per run at fidelity level l is C_l.
- The error bound suggests the optimization problem:

$$\min_{n_1,\ldots,n_L} \sum_{l=1}^{L} \lambda_{\max}^{L-l+1} n_l^{-\eta/d} \quad \text{subject to} \quad \sum_{l=1}^{L} C_l n_l \leq C_{\text{total}}.$$

- Solving the first-order optimality conditions gives

$$n_l = \frac{C_{\text{total}}}{\sum_{k=1}^{L} \lambda_{\max}^{\frac{(L-k+1)d}{\eta+d}} C_k^{\frac{\eta}{\eta+d}}} \left(\frac{\lambda_{\max}^{L-l+1}}{C_l} \right)^{\frac{d}{\eta+d}}, \qquad l = 1, \ldots, L.$$

Experimental Design Construction

- In practice, the coarsest and finest tuning parameters are often prespecified:

$$t_1 \quad \text{and} \quad t_L.$$

Experimental Design Construction

- In practice, the coarsest and finest tuning parameters are often prespecified:

$$t_1 \quad \text{and} \quad t_L.$$

- Intermediate tuning parameters are chosen geometrically:

$$t_l = t_1 T^{-(l-1)}, \qquad T = \left(\frac{t_1}{t_L}\right)^{1/(L-1)}.$$

Experimental Design Construction

- In practice, the coarsest and finest tuning parameters are often prespecified:

$$t_1 \quad \text{and} \quad t_L.$$

- Intermediate tuning parameters are chosen geometrically:

$$t_l = t_1 T^{-(l-1)}, \qquad T = \left(\frac{t_1}{t_L}\right)^{1/(L-1)}.$$

- The cost model is

$$C_l = c_1 t_l^{-\gamma}.$$

Experimental Design Construction

- In practice, the coarsest and finest tuning parameters are often prespecified:

$$t_1 \quad \text{and} \quad t_L.$$

- Intermediate tuning parameters are chosen geometrically:

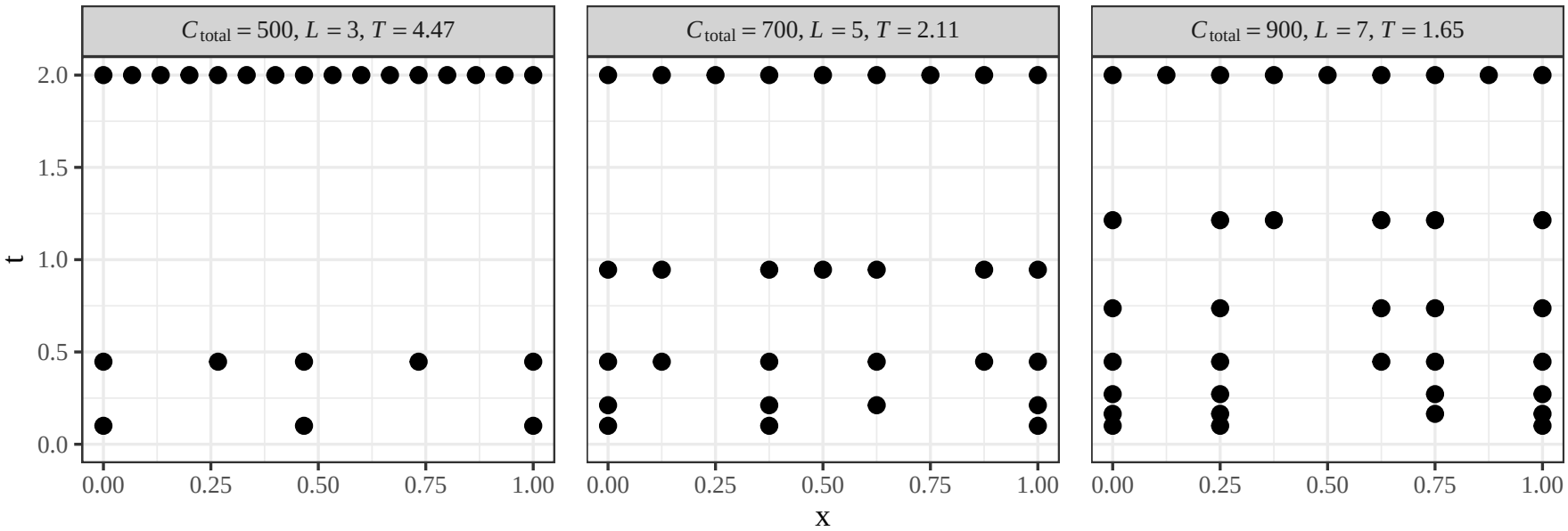
$$t_l = t_1 T^{-(l-1)}, \qquad T = \left(\frac{t_1}{t_L}\right)^{1/(L-1)}.$$

- The cost model is

$$C_l = c_1 t_l^{-\gamma}.$$

- Given C_{total}, we choose both:
  - the number of fidelity levels L;
  - the sample sizes $n_1, \ldots, n_L$.

Designs Under Increasing Budget



As C_{total} increases, the design selects more fidelity levels and places tuning parameters more densely between t_1 and t_L.

Real Applications

- We evaluate the proposed DNA model on three real simulation studies.

Real Applications

- We evaluate the proposed DNA model on three real simulation studies.
- In each case, the exact solution or very high-fidelity solution is difficult or expensive to obtain.

Real Applications

- We evaluate the proposed DNA model on three real simulation studies.
- In each case, the exact solution or very high-fidelity solution is difficult or expensive to obtain.
- Simulations are conducted at five fidelity levels:

$$t_1 > t_2 > \cdots > t_5 > 0.$$

Real Applications

- We evaluate the proposed DNA model on three real simulation studies.
- In each case, the exact solution or very high-fidelity solution is difficult or expensive to obtain.
- Simulations are conducted at five fidelity levels:

$$t_1 > t_2 > \cdots > t_5 > 0.$$

- The design points are generated using nested space-filling designs.

Real Applications

- We evaluate the proposed DNA model on three real simulation studies.
- In each case, the exact solution or very high-fidelity solution is difficult or expensive to obtain.
- Simulations are conducted at five fidelity levels:

$$t_1 > t_2 > \cdots > t_5 > 0.$$

- The design points are generated using nested space-filling designs.
- Performance is evaluated using RMSE and CRPS over 100 test points and 100 repetitions.

Competing Methods

We compare `DNAmf` with five existing approaches.

Methods targeting the highest observed fidelity $f(t_L, \mathbf{x})$

- `CoKriging`: auto-regressive co-kriging model.
- `NARGP`: nonlinear auto-regressive GP model.
- `RNAmf`: recursive non-additive emulator.

Methods targeting the exact solution $f(0, \mathbf{x})$

- `BM`: nonstationary GP with Brownian motion kernel.
- `FBM`: extension with fractional Brownian motion kernel.
- `DNAmf`: proposed diffusion non-additive model.

Three Case Studies

- Poisson's Equation

  A two-dimensional elliptic PDE on $\Omega = [0, 1]^2$ with one input parameter $x \in [-1, 1]$.

  Quantity of interest: maximum of the PDE solution over the domain.

Three Case Studies

- Poisson's Equation

  A two-dimensional elliptic PDE on $\Omega = [0, 1]^2$ with one input parameter $x \in [-1, 1]$.

  Quantity of interest: maximum of the PDE solution over the domain.

- Vibration of a Square Plate

  Finite-element simulation of the fourth natural frequency of a square elastic plate.

  Inputs: Young's modulus, Poisson's ratio, and mass density.

Three Case Studies

- Poisson's Equation

  A two-dimensional elliptic PDE on $\Omega = [0,1]^2$ with one input parameter $x \in [-1,1]$.

  Quantity of interest: maximum of the PDE solution over the domain.

- Vibration of a Square Plate

  Finite-element simulation of the fourth natural frequency of a square elastic plate.

  Inputs: Young's modulus, Poisson's ratio, and mass density.

- Heat Equation

  A one-dimensional time-dependent parabolic PDE.

  The tuning parameter is $t = S - s$, measuring temporal distance from the target time.

Simulation Setup

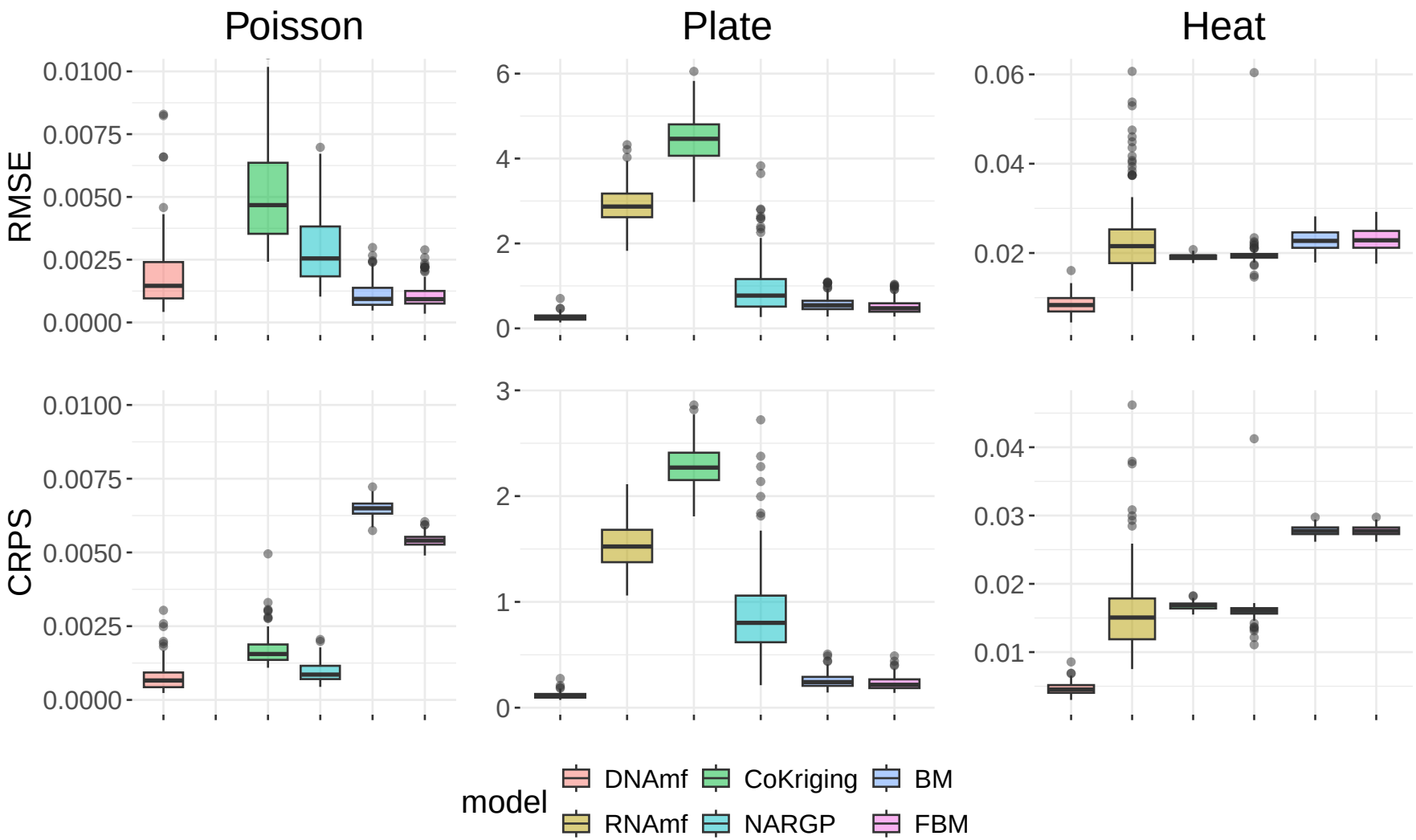
Case	n_1	n_2	n_3	n_4	n_5	d	t_1	T^{-1}
Poisson	17	11	7	5	3	1	0.1	0.65
Plate	28	21	15	11	8	3	0.55	0.9
Heat	25	15	9	5	3	1	0.5	0.7

- The tuning parameters follow

$$t_l = T^{-1} t_{l-1}.$$

- Smaller t_l corresponds to higher fidelity.
- The sample size decreases as fidelity increases.

Predictive Performance



Comparison of RMSE and CRPS across 100 repetitions for the three real case studies.

Conclusion

- Developed the DNA model for continuous-fidelity emulation and extrapolation to the exact solution $f(0, \mathbf{x})$.
- Introduced a flexible non-additive transition model across fidelity levels.
- Derived closed-form recursive prediction for posterior mean and uncertainty.
- Established error bounds and cost-aware sample allocation strategies.

Thank You!

Thank NSF DMS 2338018 for supporting this work.

MICHIGAN STATE
UNIVERSITY

Boutelet, R. and Sung, C.-L. (2025). Active learning for finite element simulations with adaptive non-stationary kernel function. *arXiv preprint arXiv:2503.23158*.

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.

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.

Tuo, R., Wu, C. F. J., and Yu, D. (2014). Surrogate modeling of computer experiments with different mesh densities. *Technometrics*, 56(3):372–380.