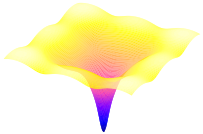
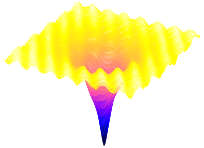
Fast Gaussian Process Regression

Aleksei G Sorokin[12], Pieterjan M Robbe[2], Fred J Hickernell[1]
IIT, Department of Applied Math[1]. Sandia National Lab[2].

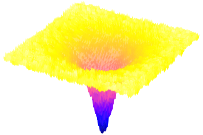
SE grid
L_2 relative error = 3.71e-2
time = 2.24e0
average CI width = 2.45e-3
CI captured 0.1%



SI lattice
L_2 relative error = 2.85e-2
time = 1.92e-3
average CI width = 3.71e0
CI captured 92.5%



DSI digital net
L_2 relative error = 2.62e-2
time = 4.54e-3
average CI width = 2.74e0
CI captured 98.8%



FastGPs Software

`pip install fastgps` or visit alegresor.github.io/fastgps/

FastGPs Software

`pip install fastgps` or visit alegresor.github.io/fastgps/

A scalable Python software for fast Gaussian process regression (GPR) requiring only

- $\mathcal{O}(N \log N)$ computations (compared to typically $\mathcal{O}(N^3)$ cost), and
- $\mathcal{O}(N)$ memory (compare to classic $\mathcal{O}(N^2)$ memory requirements).

FastGPs Software

`pip install fastgps` or visit alegresor.github.io/fastgps/

A scalable Python software for fast Gaussian process regression (GPR) requiring only

- $\mathcal{O}(N \log N)$ computations (compared to typically $\mathcal{O}(N^3)$ cost), and
- $\mathcal{O}(N)$ memory (compare to classic $\mathcal{O}(N^2)$ memory requirements).

Fast GPR exploits Gram matrix structure present when pairing

- certain low discrepancy point sets (either lattices or digital nets) to
- certain shift invariant (SI) kernels (digitally SI (DSI) kernels for digital nets).

FastGPs Software

`pip install fastgps` or visit alegresor.github.io/fastgps/

A scalable Python software for fast Gaussian process regression (GPR) requiring only

- $\mathcal{O}(N \log N)$ computations (compared to typically $\mathcal{O}(N^3)$ cost), and
- $\mathcal{O}(N)$ memory (compare to classic $\mathcal{O}(N^2)$ memory requirements).

Fast GPR exploits Gram matrix structure present when pairing

- certain low discrepancy point sets (either lattices or digital nets) to
- certain shift invariant (SI) kernels (digitally SI (DSI) kernels for digital nets).

This requires

- control over the design of experiments (to use low discrepancy designs), and
- using uncommon kernels (periodic SI kernels or discontinuous DSI kernels).

FastGPs Software Features

`pip install fastgps` or visit alegresor.github.io/fastgps/

FastGPs Software Features

`pip install fastgps` or visit alegresor.github.io/fastgps/

1. **Kernel hyperparameter optimization** of marginal log likelihood (MLL), cross validation (CV), or generalized cross validation (GCV) loss.

FastGPs Software Features

`pip install fastgps` or visit alegresor.github.io/fastgps/

1. **Kernel hyperparameter optimization** of marginal log likelihood (MLL), cross validation (CV), or generalized cross validation (GCV) loss.
2. **Fast Bayesian cubature** for uncertainty quantification in Quasi-Monte Carlo.

FastGPs Software Features

`pip install fastgps` or visit alegresor.github.io/fastgps/

1. **Kernel hyperparameter optimization** of marginal log likelihood (MLL), cross validation (CV), or generalized cross validation (GCV) loss.
2. **Fast Bayesian cubature** for uncertainty quantification in Quasi-Monte Carlo.
3. **Fast multi-task GPR** with support for different sample sizes for each task. Potentially useful for multi-fidelity simulations and Multilevel Monte Carlo.

FastGPs Software Features

`pip install fastgps` or visit alegresor.github.io/fastgps/

1. **Kernel hyperparameter optimization** of marginal log likelihood (MLL), cross validation (CV), or generalized cross validation (GCV) loss.
2. **Fast Bayesian cubature** for uncertainty quantification in Quasi-Monte Carlo.
3. **Fast multi-task GPR** with support for different sample sizes for each task. Potentially useful for multi-fidelity simulations and Multilevel Monte Carlo.
4. **Batched GPR** for simultaneously modeling vector-output simulations.

FastGPs Software Features

`pip install fastgps` or visit alegresor.github.io/fastgps/

1. **Kernel hyperparameter optimization** of marginal log likelihood (MLL), cross validation (CV), or generalized cross validation (GCV) loss.
2. **Fast Bayesian cubature** for uncertainty quantification in Quasi-Monte Carlo.
3. **Fast multi-task GPR** with support for different sample sizes for each task. Potentially useful for multi-fidelity simulations and Multilevel Monte Carlo.
4. **Batched GPR** for simultaneously modeling vector-output simulations.
5. **GPU support** enabled by the `PyTorch` stack .

FastGPs Software Features

`pip install fastgps` or visit alegresor.github.io/fastgps/

1. **Kernel hyperparameter optimization** of marginal log likelihood (MLL), cross validation (CV), or generalized cross validation (GCV) loss.
2. **Fast Bayesian cubature** for uncertainty quantification in Quasi-Monte Carlo.
3. **Fast multi-task GPR** with support for different sample sizes for each task. Potentially useful for multi-fidelity simulations and Multilevel Monte Carlo.
4. **Batched GPR** for simultaneously modeling vector-output simulations.
5. **GPU support** enabled by the `PyTorch` stack .
6. **Flexible LD sequences and SI/DSI kernels** from the Quasi-Monte Carlo Python package `QMCPy` qmcsoftware.github.io/QMCSoftware/.

FastGPs Software Features

`pip install fastgps` or visit alegresor.github.io/fastgps/

1. **Kernel hyperparameter optimization** of marginal log likelihood (MLL), cross validation (CV), or generalized cross validation (GCV) loss.
2. **Fast Bayesian cubature** for uncertainty quantification in Quasi-Monte Carlo.
3. **Fast multi-task GPR** with support for different sample sizes for each task. Potentially useful for multi-fidelity simulations and Multilevel Monte Carlo.
4. **Batched GPR** for simultaneously modeling vector-output simulations.
5. **GPU support** enabled by the `PyTorch` stack .
6. **Flexible LD sequences and SI/DSI kernels** from the Quasi-Monte Carlo Python package `QMCPy` qmcsoftware.github.io/QMCSoftware/.
7. **Derivative-informed GPR** for simulations coupled with automatic differentiation.

FastGPs Software Features

`pip install fastgps` or visit alegresor.github.io/fastgps/

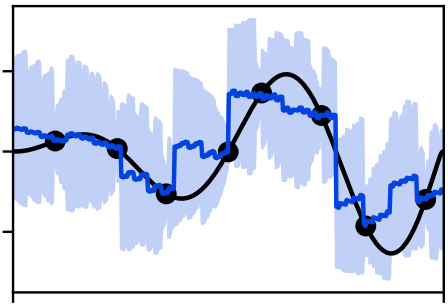
1. **Kernel hyperparameter optimization** of marginal log likelihood (MLL), cross validation (CV), or generalized cross validation (GCV) loss.
2. **Fast Bayesian cubature** for uncertainty quantification in Quasi-Monte Carlo.
3. **Fast multi-task GPR** with support for different sample sizes for each task. Potentially useful for multi-fidelity simulations and Multilevel Monte Carlo.
4. **Batched GPR** for simultaneously modeling vector-output simulations.
5. **GPU support** enabled by the `PyTorch` stack .
6. **Flexible LD sequences and SI/DSI kernels** from the Quasi-Monte Carlo Python package `QMCPy` qmcsoftware.github.io/QMCSoftware/.
7. **Derivative-informed GPR** for simulations coupled with automatic differentiation.
8. **Efficient variance projections** for non-greedy Bayesian optimization in MLMC.

Gaussian Process Regression (GPR)

$f \sim \mathrm{GP}(0, K)$ with posterior mean and covariance

$$\mathbb{E}[f(\boldsymbol{x})|\mathsf{X}, \boldsymbol{f}] = \boldsymbol{K}_{\mathsf{X}}(\boldsymbol{x})\mathsf{K}^{-1}\boldsymbol{f}$$
$$\mathbb{V}[f(\boldsymbol{x})|\mathsf{X}, \boldsymbol{f}] = K(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{K}_{\mathsf{X}}(\boldsymbol{x})^T\mathsf{K}^{-1}\boldsymbol{K}_{\mathsf{X}}(\boldsymbol{x})$$

- SPD $K : [0,1]^d \times [0,1]^d \to \mathbb{R}$
- Sampling locations $\mathsf{X} = \{\boldsymbol{x}_i\}_{i=0}^{N-1}$
- Sample values $\boldsymbol{f} = \{f(\boldsymbol{x}_i)\}_{i=0}^{N-1}$
- Kernel vector $\boldsymbol{K}_{\mathsf{X}}(\boldsymbol{x}) = \{K(\boldsymbol{x}, \boldsymbol{x}_i)\}_{i=0}^{N-1}$
- Gram matrix $\mathsf{K} = \{K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})\}_{i,i'=0}^{N-1}$



Typically requires $\mathcal{O}(N^2)$ storage and $\mathcal{O}(N^3)$ computations

Fast GPs Pairing LD Points with Special Kernels

Recently explored in the context of fast Bayesian cubature [Rathinavel, 2019]

Fast GPs Pairing LD Points with Special Kernels

Recently explored in the context of fast Bayesian cubature [Rathinavel, 2019]

1. **LD Lattices + Shift Invariant (SI) Kernels**
   - Give circulant Gram matrices $\mathsf{K} = \{K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})\}_{i,i'=0}^{N-1}$
   - $\therefore$ Eigendecomp $\mathsf{K} = \mathsf{V}\Lambda\overline{\mathsf{V}}$ where $\overline{\mathsf{V}}$ is the DFT matrix $\rightarrow$ FFT in $\mathcal{O}(N \log N)$
   - [Rathinavel and Hickernell, 2019]

Fast GPs Pairing LD Points with Special Kernels

Recently explored in the context of fast Bayesian cubature [Rathinavel, 2019]

1. **LD Lattices + Shift Invariant (SI) Kernels**
   - Give circulant Gram matrices $\mathsf{K} = \{K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})\}_{i,i'=0}^{N-1}$
   - $\therefore$ Eigendecomp $\mathsf{K} = \mathsf{V}\Lambda\overline{\mathsf{V}}$ where $\overline{\mathsf{V}}$ is the DFT matrix $\rightarrow$ FFT in $\mathcal{O}(N \log N)$
   - [Rathinavel and Hickernell, 2019]
2. **LD Digital Nets + Digitally Shift Invariant (DSI) Kernels**
   - Give Recursive Symmetric Block Toeplitz (RSBT) Gram matrices K
   - $\therefore$ Eigendecomp $\mathsf{K} = \mathsf{V}\Lambda\overline{\mathsf{V}}$ where $\overline{\mathsf{V}}$ is the Hadamard matrix $\rightarrow$ FWHT in $\mathcal{O}(N \log N)$
   - [Rathinavel and Hickernell, 2022]

Fast GPs Pairing LD Points with Special Kernels

Recently explored in the context of fast Bayesian cubature [Rathinavel, 2019]

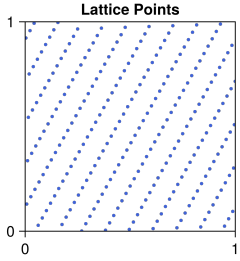
1. **LD Lattices + Shift Invariant (SI) Kernels**
   - Give circulant Gram matrices $\mathsf{K} = \{K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})\}_{i,i'=0}^{N-1}$
   - $\therefore$ Eigendecomp $\mathsf{K} = \mathsf{V}\Lambda\overline{\mathsf{V}}$ where $\overline{\mathsf{V}}$ is the DFT matrix $\rightarrow$ FFT in $\mathcal{O}(N \log N)$
   - [Rathinavel and Hickernell, 2019]
2. **LD Digital Nets + Digitally Shift Invariant (DSI) Kernels**
   - Give Recursive Symmetric Block Toeplitz (RSBT) Gram matrices K
   - $\therefore$ Eigendecomp $\mathsf{K} = \mathsf{V}\Lambda\overline{\mathsf{V}}$ where $\overline{\mathsf{V}}$ is the Hadamard matrix $\rightarrow$ FWHT in $\mathcal{O}(N \log N)$
   - [Rathinavel and Hickernell, 2022]

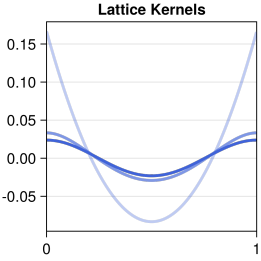
Let $\boldsymbol{v}_1 = \mathbf{1}/\sqrt{N}$ and $\boldsymbol{k}_1$ be the first columns of $\overline{\boldsymbol{V}}$ and K respectively:

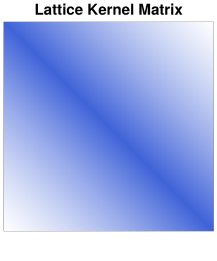
$$\boldsymbol{\lambda} := \Lambda\mathbf{1} = \sqrt{N}\Lambda\overline{\boldsymbol{v}}_1 = \sqrt{N}\overline{\mathsf{V}}\mathsf{V}\Lambda\overline{\boldsymbol{v}}_1 = \sqrt{N}\overline{\mathsf{V}}\boldsymbol{k}_1$$

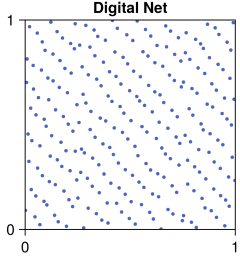
- $\mathsf{K}\boldsymbol{a}$, $\mathsf{K}^{-1}\boldsymbol{a}$, and $|\mathsf{K}|$ can all be computed in $\mathcal{O}(N \log N)$ computations
- Only requires evaluating and storing the first column of K

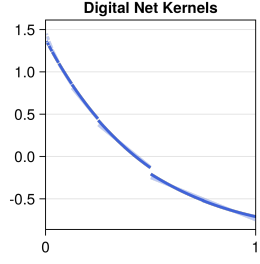
Lattices + SI K = Circulant K and Digital Nets + DSI K = RSBT K

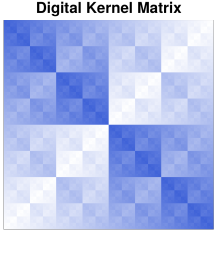




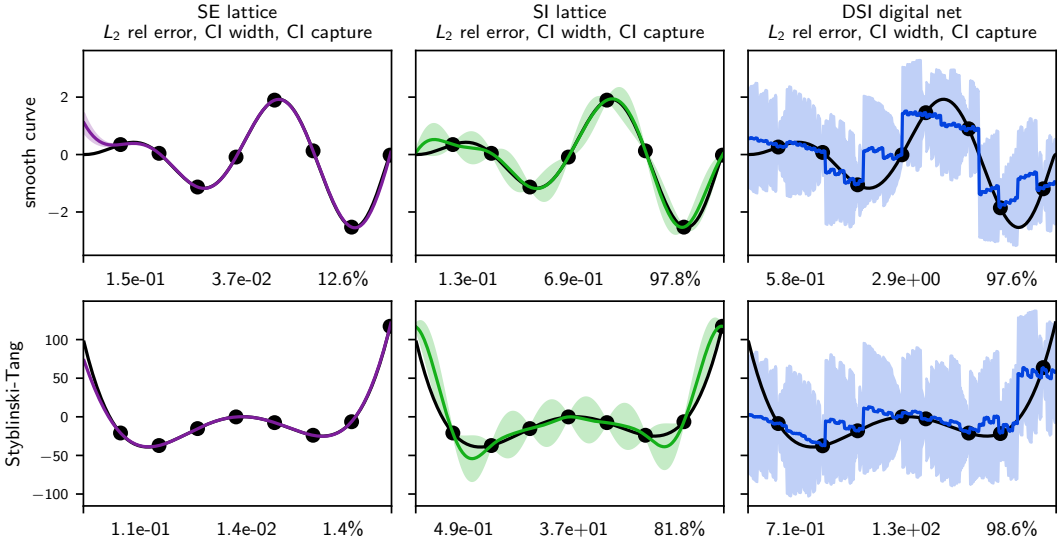




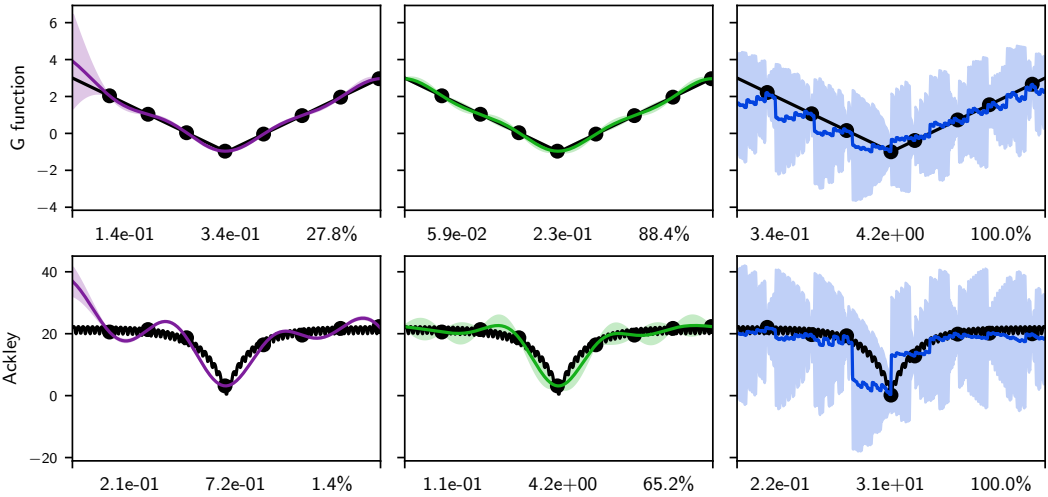




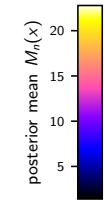
Examples in One Dimension [Surjanovic and Bingham], $N = 8$



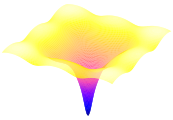
Examples in One Dimension [Surjanovic and Bingham], $N = 8$



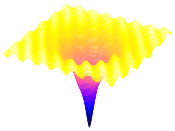
Ackley Function in Two Dimensions [Surjanovic and Bingham], $N = 4096$



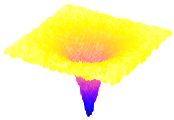
SE grid
L_2 relative error = 3.71e-2
time = 2.24e0
average CI width = 2.45e-3
CI captured 0.1%

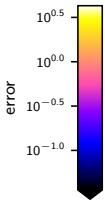


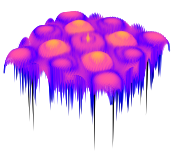
SI lattice
L_2 relative error = 2.85e-2
time = 1.92e-3
average CI width = 3.71e0
CI captured 92.5%

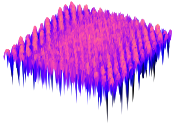


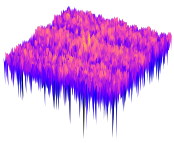
DSI digital net
L_2 relative error = 2.62e-2
time = 4.54e-3
average CI width = 2.74e0
CI captured 98.8%











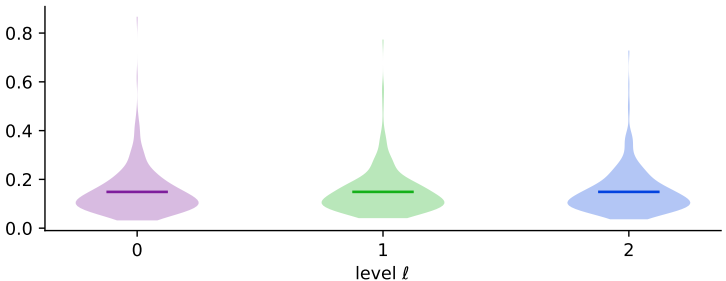
Multilevel (Multi-Task) Modeling

Given a multilevel simulation

$$f : \{1, \dots, L\} \times [0,1]^d \to \mathbb{R},$$

we want to model $f(L, \cdot)$, the true (maximum-fidelity) simulation.

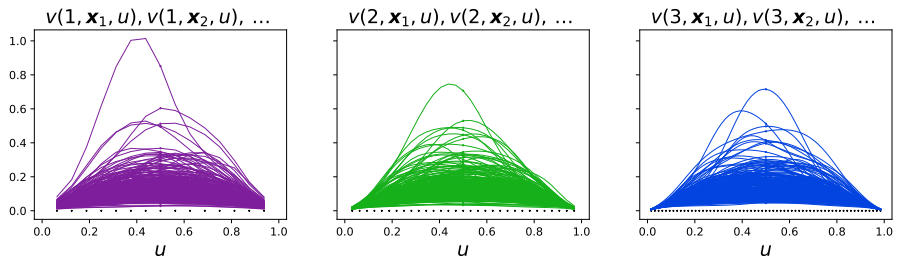
- $f(\ell, \boldsymbol{x})$ simulates at level $\ell \in \{1, \dots, L\}$ and with parameters $\boldsymbol{x} \in [0,1]^d$
- Cost C_ℓ typically greater on higher levels
- $f(1, \cdot), f(2, \cdot), \dots, f(L, \cdot)$ are typically highly correlated



Numerical Solutions of PDEs with Random Coefficients

$$f(\ell, \boldsymbol{x}) = \mathcal{F}(v(\ell, \boldsymbol{x}, \cdot))$$

- $v : \{1, \dots, L\} \times [0,1]^d \times \Omega$ is the numerical solution to the PDE
- $\boldsymbol{x}$ represent random coefficients, e.g. coefficients in a Karhunen–Loève expansion
- ℓ controls the fidelity of the numerical solver e.g. the mesh width is $2^{-\ell}$
- $\mathcal{F}$ is a (possibly non-linear) functional of the PDE solution, e.g.,
  - $\mathcal{F}(v(\ell, \boldsymbol{x}, \cdot)) = \mathbb{E}[v(\ell, \boldsymbol{x}, \boldsymbol{U})]$ where $\boldsymbol{U} \sim \mathcal{U}(\Omega)$, or
  - $\mathcal{F}(v(\ell, \boldsymbol{x}, \cdot)) = v(\ell, \boldsymbol{x}, u)$ for some $u \in \Omega$, e.g., $u = 1/2$ shown below



Multilevel (Quasi-)Monte Carlo without Replications

$$\mathbb{E}[f(L, \boldsymbol{X})] = \sum_{\ell=1}^{L} \mathbb{E}[\underbrace{f(\ell, \boldsymbol{X}) - f(\ell-1, \boldsymbol{X})}_{\Delta_\ell(\boldsymbol{X})}], \quad \boldsymbol{X} \sim \mathcal{U}[0,1]^d, \qquad f(0, \cdot) = 0$$

Multilevel (Quasi-)Monte Carlo without Replications

$$\mathbb{E}[f(L,\boldsymbol{X})] = \sum_{\ell=1}^{L} \mathbb{E}[\underbrace{f(\ell,\boldsymbol{X}) - f(\ell-1,\boldsymbol{X})}_{\Delta_\ell(\boldsymbol{X})}], \quad \boldsymbol{X} \sim \mathcal{U}[0,1]^d, \qquad f(0,\cdot) = 0$$

MLMC $\mathbb{E}[\Delta_\ell(\boldsymbol{X})] \approx \frac{1}{N_\ell} \sum_{i=0}^{N_\ell - 1} \Delta_\ell(\boldsymbol{x}_i^\ell)$ for IID $\boldsymbol{x}_0^\ell, \ldots, \boldsymbol{x}_{N_\ell-1}^\ell \sim \mathcal{U}[0,1]^d$

- $N_\ell \propto \sqrt{\mathbb{V}[\Delta_\ell(\boldsymbol{X})]/C_\ell}$ chosen to optimally minimize error [Giles, 2008]

Multilevel (Quasi-)Monte Carlo without Replications

$$\mathbb{E}[f(L,\boldsymbol{X})] = \sum_{\ell=1}^{L} \mathbb{E}[\underbrace{f(\ell,\boldsymbol{X}) - f(\ell-1,\boldsymbol{X})}_{\Delta_\ell(\boldsymbol{X})}], \quad \boldsymbol{X} \sim \mathcal{U}[0,1]^d, \qquad f(0,\cdot) = 0$$

MLMC $\mathbb{E}[\Delta_\ell(\boldsymbol{X})] \approx \frac{1}{N_\ell}\sum_{i=0}^{N_\ell-1} \Delta_\ell(\boldsymbol{x}_i^\ell)$ for IID $\boldsymbol{x}_0^\ell, \ldots, \boldsymbol{x}_{N_\ell-1}^\ell \sim \mathcal{U}[0,1]^d$

- $N_\ell \propto \sqrt{\mathbb{V}[\Delta_\ell(\boldsymbol{X})]/C_\ell}$ chosen to optimally minimize error [Giles, 2008]

R-MLQMC $\mathbb{E}[\Delta_\ell(\boldsymbol{X})] \approx \frac{1}{R}\sum_{r=1}^{R} \frac{1}{N_\ell}\sum_{i=0}^{N_\ell-1} \Delta_\ell(\boldsymbol{x}_{ri}^\ell)$ [Giles and Waterhouse, 2009]

- IID randomizations of low discrepancy pointsets $\{\boldsymbol{x}_{1i}^\ell\}_{i=0}^{N_\ell-1}, \ldots, \{\boldsymbol{x}_{Ri}^\ell\}_{i=0}^{N_\ell-1}$.
- N_ℓ greedily doubled on level where $\mathbb{V}[\Delta_\ell(\boldsymbol{X})]/(N_\ell C_\ell)$ the largest
- Variance approximated by sample variance across R sub-means

Multilevel (Quasi-)Monte Carlo without Replications

$$\mathbb{E}[f(L,\boldsymbol{X})] = \sum_{\ell=1}^{L} \mathbb{E}[\underbrace{f(\ell,\boldsymbol{X}) - f(\ell-1,\boldsymbol{X})}_{\Delta_\ell(\boldsymbol{X})}], \quad \boldsymbol{X} \sim \mathcal{U}[0,1]^d, \qquad f(0,\cdot) = 0$$

MLMC $\mathbb{E}[\Delta_\ell(\boldsymbol{X})] \approx \frac{1}{N_\ell}\sum_{i=0}^{N_\ell-1} \Delta_\ell(\boldsymbol{x}_i^\ell)$ for IID $\boldsymbol{x}_0^\ell, \dots, \boldsymbol{x}_{N_\ell-1}^\ell \sim \mathcal{U}[0,1]^d$

- $N_\ell \propto \sqrt{\mathbb{V}[\Delta_\ell(\boldsymbol{X})]/C_\ell}$ chosen to optimally minimize error [Giles, 2008]

R-MLQMC $\mathbb{E}[\Delta_\ell(\boldsymbol{X})] \approx \frac{1}{R}\sum_{r=1}^{R} \frac{1}{N_\ell}\sum_{i=0}^{N_\ell-1} \Delta_\ell(\boldsymbol{x}_{ri}^\ell)$ [Giles and Waterhouse, 2009]

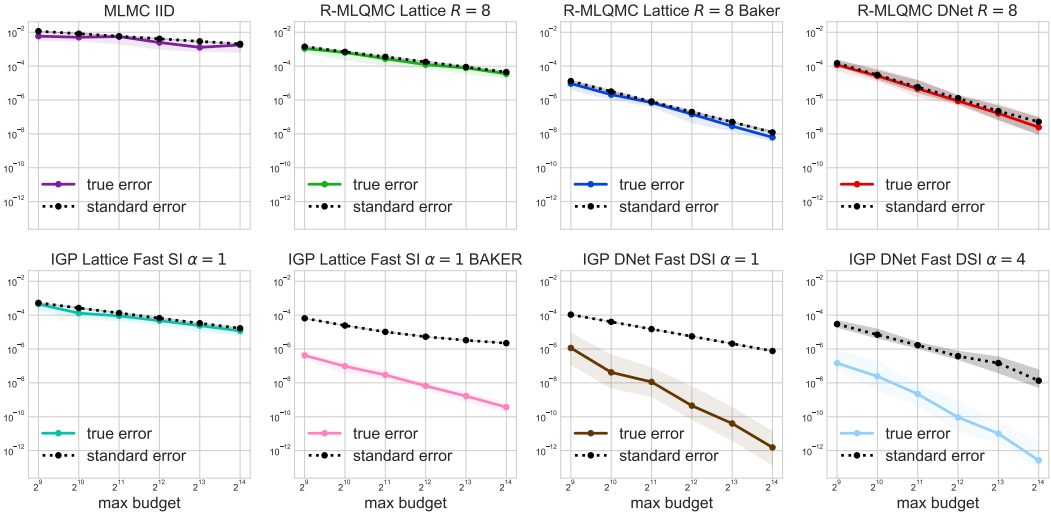
- IID randomizations of low discrepancy pointsets $\{\boldsymbol{x}_{1i}^\ell\}_{i=0}^{N_\ell-1}, \dots, \{\boldsymbol{x}_{Ri}^\ell\}_{i=0}^{N_\ell-1}$.
- N_ℓ greedily doubled on level where $\mathbb{V}[\Delta_\ell(\boldsymbol{X})]/(N_\ell C_\ell)$ the largest
- Variance approximated by sample variance across R sub-means

(IGP) Bayesian MLQMC without replications $\mathbb{E}[\Delta_\ell(\boldsymbol{X})] \approx \frac{1}{N_\ell}\sum_{i=0}^{N_\ell-1} \Delta_\ell(\boldsymbol{x}_i^\ell)$

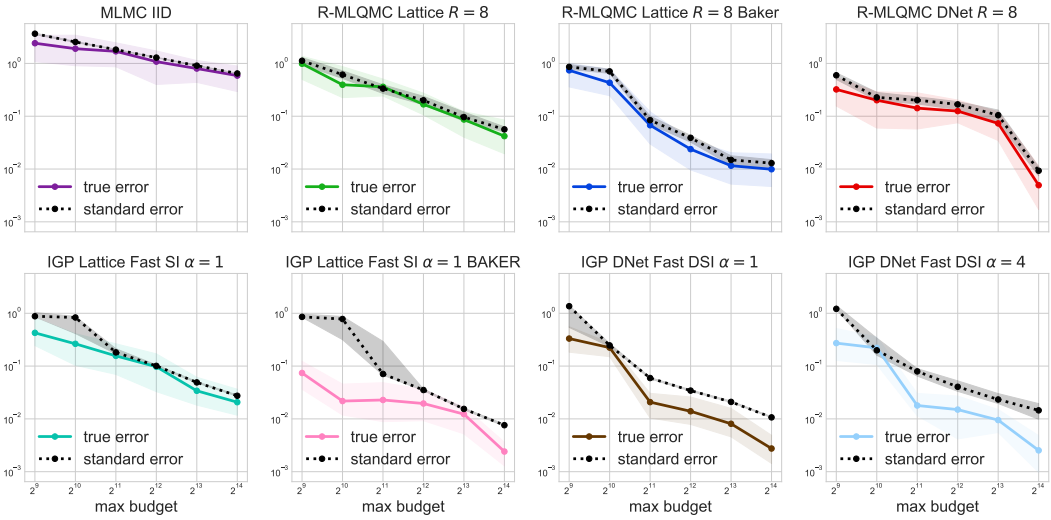
- $\{\boldsymbol{x}_i^\ell\}_{i=0}^{N_\ell-1}$ a single randomized low discrepancy pointset
- N_ℓ greedily doubled on level where $\mathbb{V}[\Delta_\ell(\boldsymbol{X})]/(N_\ell C_\ell)$ the largest
- Variance taken to be posterior variance with $\Delta_\ell \sim \mathrm{GP}(0, K_\ell)$ (independent GPs)

$$f(\ell,(x_1,x_2)) = \sin(x_1) + 2^{-\ell}\cos(x_2)$$

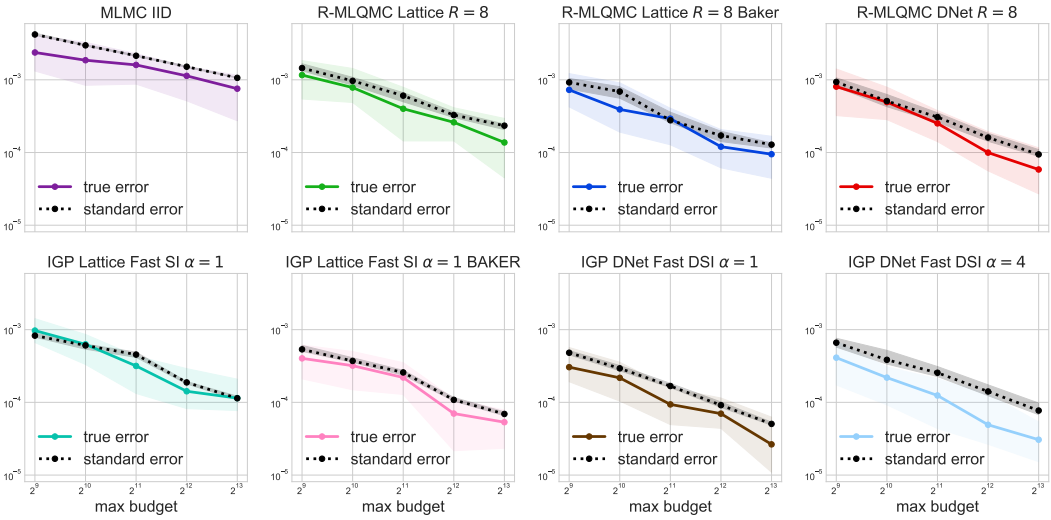
analytic problem with $d = 2$ dimensions and $L = 4$ levels with $T = 50$ trials



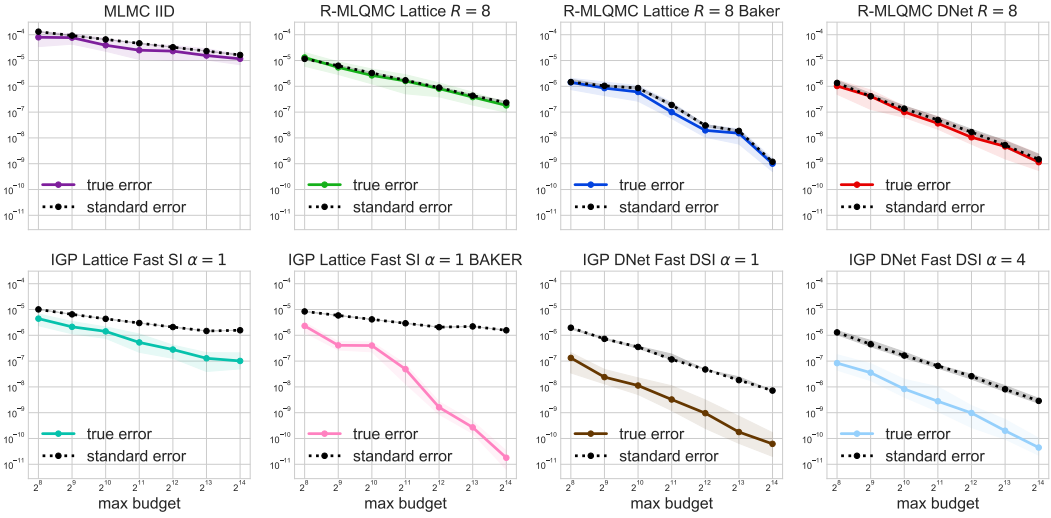
borehole problem with $d = 8$ dimensions and $L = 2$ levels with $T = 50$ trials



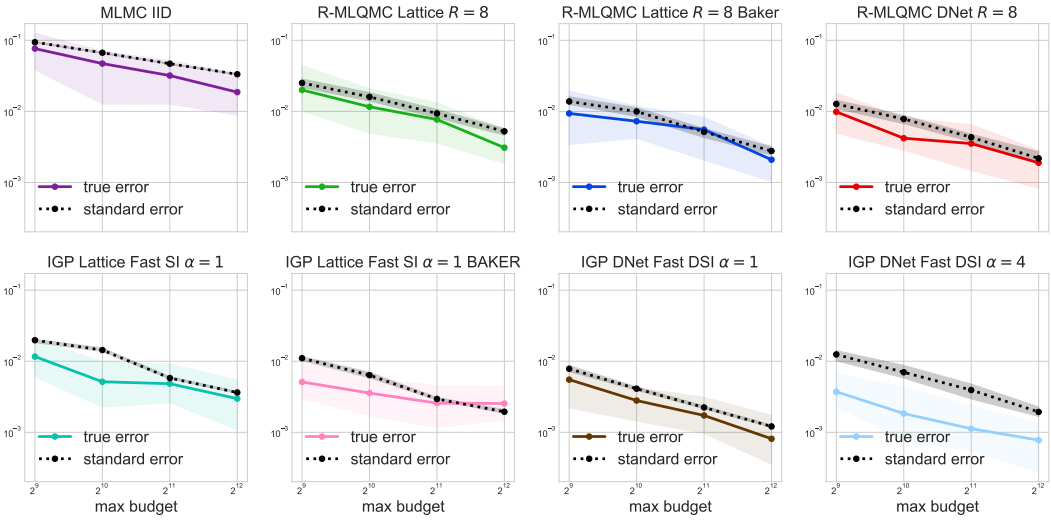
elliptic problem with $d = 8$ dimensions and $L = 4$ levels with $T = 50$ trials



steady state diffusion 1d problem with $d = 9$ dimensions and $L = 5$ levels with $T = 50$ trials



Asian option problem with $d = 16$ dimensions and $L = 8$ levels with $T = 50$ trials



Multi-Task Gaussian Processes

$$f(\ell, \boldsymbol{x}) \sim \mathrm{GP}(0, K)$$

$N = N_1 + \cdots + N_L$ sampling locations $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_L$ where $\mathcal{D}_\ell = \left\{ \left(\ell, \boldsymbol{x}_i^\ell\right) \right\}_{i=1}^{N_\ell}$.
Posterior mean and covariance

$$\mathbb{E}[f(\ell, \boldsymbol{x})] = \boldsymbol{K}^T(\ell, \boldsymbol{x}) \mathsf{K}^{-1} \boldsymbol{f}$$
$$\mathbb{V}[f(\ell, \boldsymbol{x})] = K((\ell, \boldsymbol{x}), (\ell, \boldsymbol{x})) - \boldsymbol{K}^T(\ell', \boldsymbol{x}') \mathsf{K}^{-1} \boldsymbol{K}(\ell, \boldsymbol{x})$$

- $\boldsymbol{K}(\ell, \boldsymbol{x}) = K(\mathcal{D}, (\ell, \boldsymbol{x}))$ and $\boldsymbol{f} = f(\mathcal{D})$ are length N vectors
- $\mathsf{K} = K(\mathcal{D}, \mathcal{D}^T)$ is the $N \times N$ Gram matrix

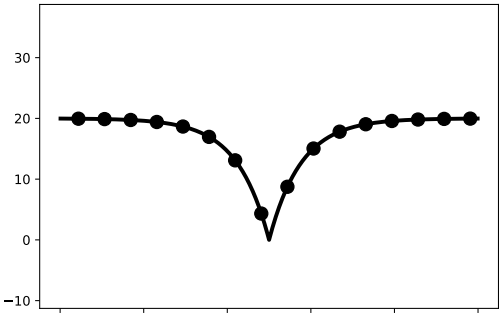
Kernel K depends on hyperparameters $\boldsymbol{\theta}$ e.g. global scale, lengthscale, etc.
Hyperparameters $\boldsymbol{\theta}$ often chosen to minimize negative marginal log likelihood (NMLL)

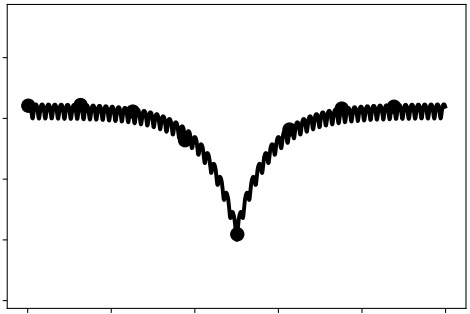
$$\mathrm{NMLL} \propto \boldsymbol{f}^T \mathsf{K}^{-1} \boldsymbol{f} + \log|\mathsf{K}| + \log(2\pi) N$$

$\therefore$ Multi-Task GPs require computing $\mathsf{K}^{-1}\boldsymbol{f}$ and $\log|\mathsf{K}|$ $\implies$ standard cost $\mathcal{O}(N^3)$

Independent GPs vs a Multi-Task GP Example

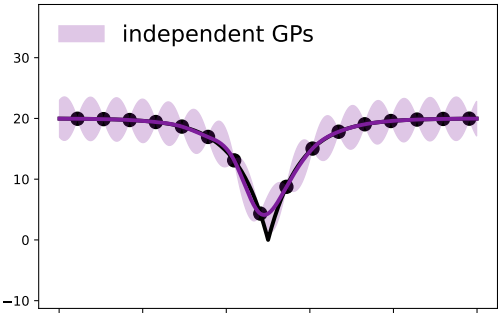
Low Fidelity Left, High Fidelity Right

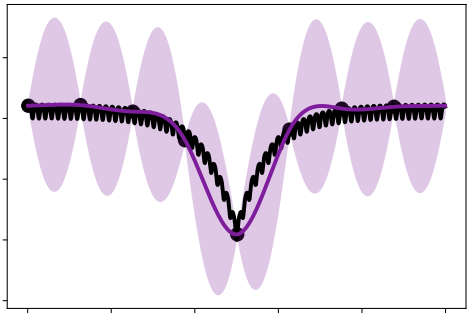




Independent GPs vs a Multi-Task GP Example

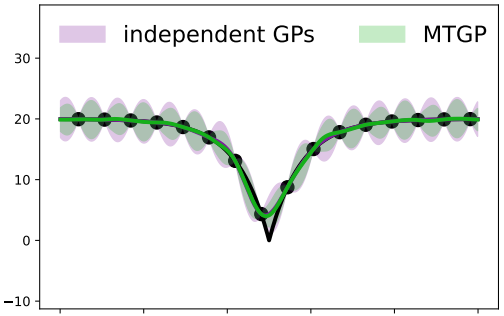
Low Fidelity Left, High Fidelity Right

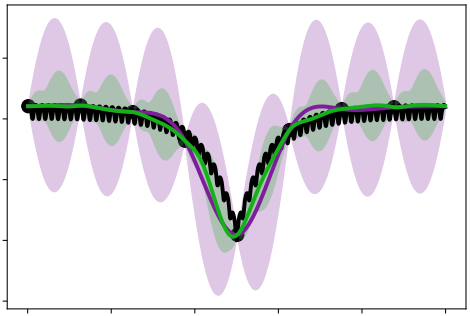




Independent GPs vs a Multi-Task GP Example

Low Fidelity Left, High Fidelity Right





Product Kernels for Multi-Task GPs

Common to assume

$$K((\ell, \boldsymbol{x}), (\ell', \boldsymbol{x}')) = R(\ell, \ell') Q(\boldsymbol{x}, \boldsymbol{x}')$$

- $R : \{1, \dots, L\} \times \{1, \dots, L\} \to \mathbb{R}$ an SPD kernel over levels e.g.

$$\mathsf{R} = \left\{R(\ell, \ell')\right\}_{\ell, \ell'=1}^{L} = \mathsf{B}\mathsf{B}^T + \operatorname{diag}(\boldsymbol{\nu}), \quad \boldsymbol{\nu} \in \mathbb{R}_+^L, \quad \mathsf{B} \in \mathbb{R}^{L \times r}, \quad \text{rank } r \leqslant L$$

- $Q : [0,1]^d \times [0,1]^d \to \mathbb{R}$ an SPD kernel over parameters

$$\mathsf{K} = \begin{pmatrix} \mathsf{K}_{11} & \cdots & \mathsf{K}_{1L} \\ \vdots & \ddots & \vdots \\ \mathsf{K}_{1L} & \cdots & \mathsf{K}_{LL} \end{pmatrix} = \begin{pmatrix} R_{11}\mathsf{Q}_{11} & \cdots & R_{1L}\mathsf{Q}_{1L} \\ \vdots & \ddots & \vdots \\ R_{1L}\mathsf{Q}_{1L} & \cdots & R_{LL}\mathsf{Q}_{LL} \end{pmatrix}$$

- $R_{\ell\ell'} = R(\ell, \ell')$ is a scalar
- $\mathsf{Q}_{\ell\ell'} = Q(\mathcal{D}_\ell, \mathcal{D}_{\ell'}^T)$ is an $N_\ell \times N_{\ell'}$ Gram matrix

Fast Multi-Task GPs

$$\mathsf{K} = \begin{pmatrix} R_{11}\mathsf{Q}_{11} & \cdots & R_{1L}\mathsf{Q}_{1L} \\ \vdots & \ddots & \vdots \\ \overline{R_{1L}\mathsf{Q}_{1L}} & \cdots & R_{LL}\mathsf{Q}_{LL} \end{pmatrix}$$

Idea: Force "nice" structure in $\mathsf{Q}_{\ell\ell'}$ through special pairings of $\mathsf{X}_\ell = \{\boldsymbol{x}_i^\ell\}_{i=1}^{N_\ell}$ and Q

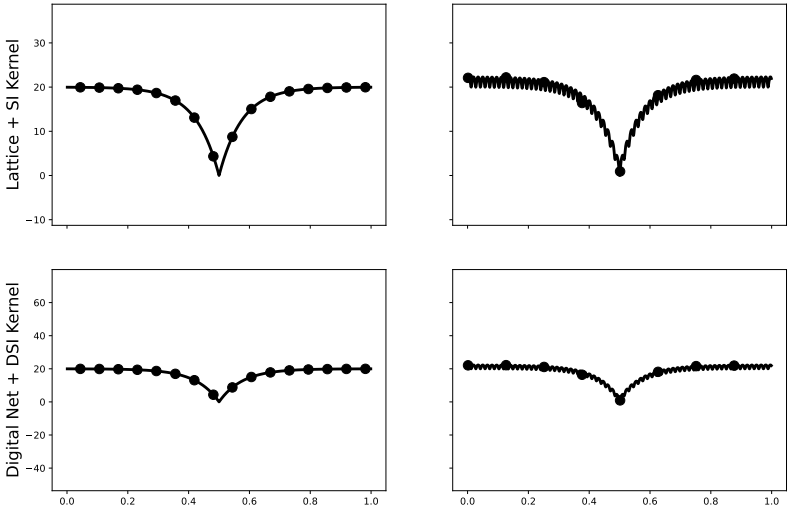
1. X_ℓ a lattice and Q a shift invariant (SI) kernel $\implies$ $\mathsf{Q}_{\ell\ell'}$ block circulant
2. X_ℓ a (base 2) digital net and Q a digitally SI (DSI) kernel $\implies$ $\mathsf{Q}_{\ell\ell}$ block RSBT

Technicalities

- Lattices and digital nets require sample sizes $N_\ell = 2^{m_\ell}$
- Lattices $\mathsf{X}_1, \ldots, \mathsf{X}_L$: same generating vector, possibly different random shifts
- Circulant matrices diagonalizable by FFT
- Digital nets $\mathsf{X}_1, \ldots, \mathsf{X}_L$: same generating matrices, possibly different digital shifts
- RSBT matrices diagonalizable by FWHT

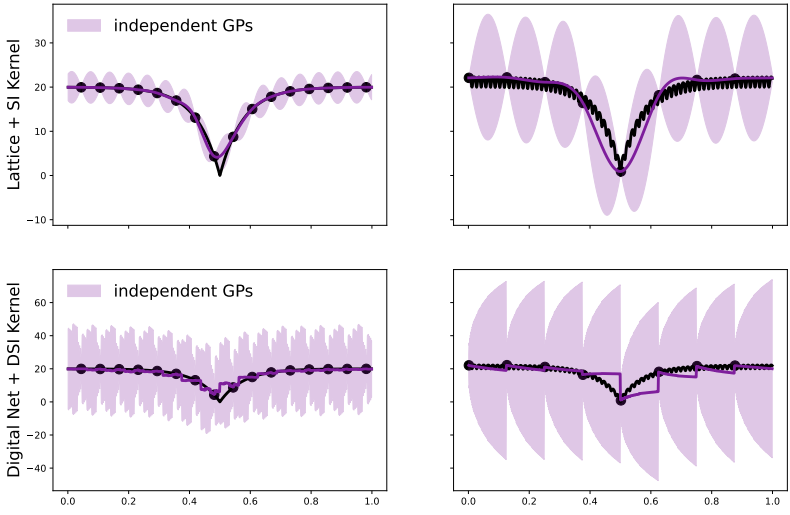
Independent Fast GPs vs Fast Multi-Task GPs Example

Low Fidelity Left, High Fidelity Right



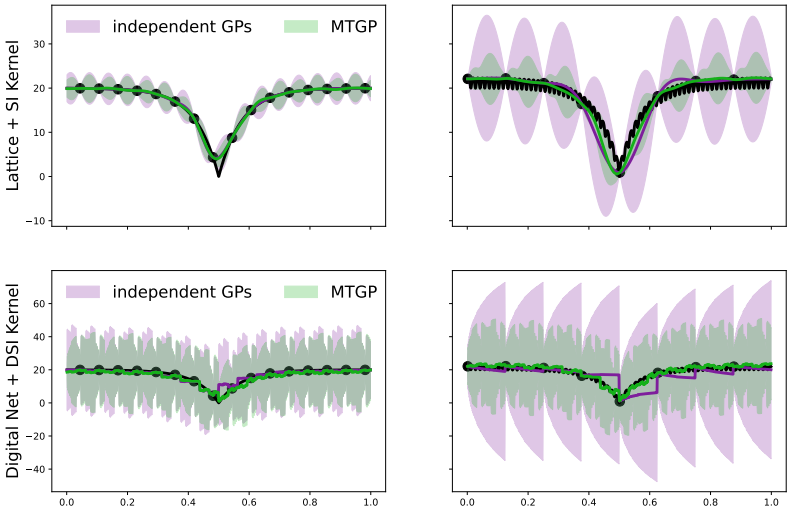
Independent Fast GPs vs Fast Multi-Task GPs Example

Low Fidelity Left, High Fidelity Right



Independent Fast GPs vs Fast Multi-Task GPs Example

Low Fidelity Left, High Fidelity Right



Fast Multi-Task GPs Continued

$$\mathsf{K}_{\ell\ell'} = R_{\ell\ell'}\mathsf{Q}_{\ell\ell'} = \mathsf{V}_{m_\ell}\Sigma_{\ell\ell'}\overline{\mathsf{V}_{m_{\ell'}}}$$

- $\overline{\mathsf{V}_m}$ a $2^m \times 2^m$ *fast transform matrix*
  1. Lattice X_ℓ with SI Q makes $\overline{\mathsf{V}_{m_\ell}}$ the Fast Fourier Transform
  2. Digital Net X_ℓ with DSI Q makes $\overline{\mathsf{V}_{m_\ell}}$ the Fast Walsh Hadamard Transform
- $\mathsf{V}_m\boldsymbol{a}$ and $\overline{\mathsf{V}_m}\boldsymbol{a}$ both cost only $\mathcal{O}(m2^m)$ to compute
- The first column of $\overline{\mathsf{V}_m}$ is $\mathbf{1}_m/\sqrt{2^m}$
- $\Sigma_{\ell\ell'}$ a diagonal block matrix characterized by

$$\boldsymbol{\sigma}_{\ell\ell'} = \Sigma_{\ell\ell'}\mathbf{1}_{m_{\ell'}} = \sqrt{2^{m_{\ell'}}}\overline{\mathsf{V}_{m_\ell}}\boldsymbol{k}_{\ell\ell',1}$$

where $\boldsymbol{k}_{\ell\ell',1}$ is the first column of $\mathsf{K}_{\ell\ell'}$ and we assume $m_\ell \geqslant m_{\ell'}$

Fast Multi-Task GPs NMLL

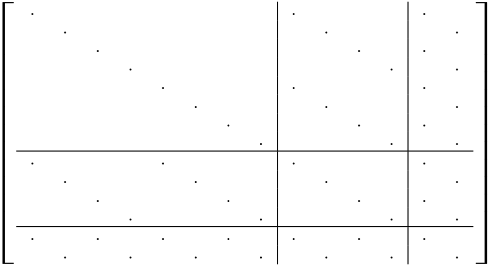
$$\mathsf{K} = \begin{pmatrix} \mathsf{V}_{m_1} & & \\ & \ddots & \\ & & \mathsf{V}_{m_L} \end{pmatrix} \begin{pmatrix} \Sigma_{11} & \cdots & \Sigma_{1L} \\ \vdots & \ddots & \vdots \\ \overline{\Sigma_{1L}} & \cdots & \Sigma_{LL} \end{pmatrix} \begin{pmatrix} \overline{\mathsf{V}_{m_1}} & & \\ & \ddots & \\ & & \overline{\mathsf{V}_{m_L}} \end{pmatrix} =: \mathsf{V}\Sigma\overline{\mathsf{V}}$$

$$\mathrm{NMLL} \propto \widehat{\boldsymbol{f}}^T \Sigma^{-1} \widehat{\boldsymbol{f}} + \log|\Sigma| + \log(2\pi)N, \quad \widehat{\boldsymbol{f}} = \overline{\mathsf{V}}\boldsymbol{f}$$

$\therefore$ Fast Multi-Task GP fitting requires computing $\Sigma^{-1}\widehat{\boldsymbol{f}}$ and $\log|\Sigma|$

For example, if $N_1 = 8$, $N_2 = 4$, and $N_3 = 2$ then Σ has the following structure

$\Sigma =$



Fast Multi-Task GPs Storage and Costs for Σ^{-1}, $\log|\Sigma|$

- Assume evaluating $f(\ell, \boldsymbol{x})$ costs C_ℓ for any $\boldsymbol{x} \in [0,1]^d$
- Assume evaluating $\mathsf{K}((\ell, \boldsymbol{x}), (\ell', \boldsymbol{x}'))$ costs $\mathcal{O}(d)$
- Assume $m_1 \geqslant \cdots \geqslant m_L$ i.e. less samples on higher levels (or reorder levels)

$$\text{NMLL} \propto \widehat{\boldsymbol{f}}^T \Sigma^{-1} \widehat{\boldsymbol{f}} + \log|\Sigma| + \log(2\pi)N$$

- Evaluate $\boldsymbol{f} = f(\mathcal{D})$ at cost $\mathcal{O}\left(\sum_{\ell=1}^L C_\ell 2^{m_\ell}\right)$
- Evaluate $\widehat{\boldsymbol{f}} = \overline{\mathsf{V}}\boldsymbol{f}$ at cost $\mathcal{O}\left(\sum_{\ell=1}^L m_\ell 2^{m_\ell}\right)$
- Evaluate only first columns of $\mathsf{K}_{\ell\ell'}$ at total cost $\mathcal{O}\left(d\sum_{\ell=1}^L (L-\ell+1)2^{m_\ell}\right)$
- Evaluate Σ at cost $\mathcal{O}\left(\sum_{\ell=1}^L (L-\ell+1)m_\ell 2^{m_\ell}\right)$
- Store Σ in $\mathcal{O}\left(\sum_{\ell=1}^L (L-\ell+1)2^{m_\ell}\right)$ (possibly complex) floats
- Evaluate $\Sigma^{-1}\widehat{\boldsymbol{f}}$ and $\log|\Sigma|$ at cost $\mathcal{O}\left(\sum_{\ell'=1}^L 2^{-m_{\ell'}}\left[\sum_{\ell=1}^{\ell'-1} 2^{m_\ell}\right]^2\right)$ ▸ algorithm

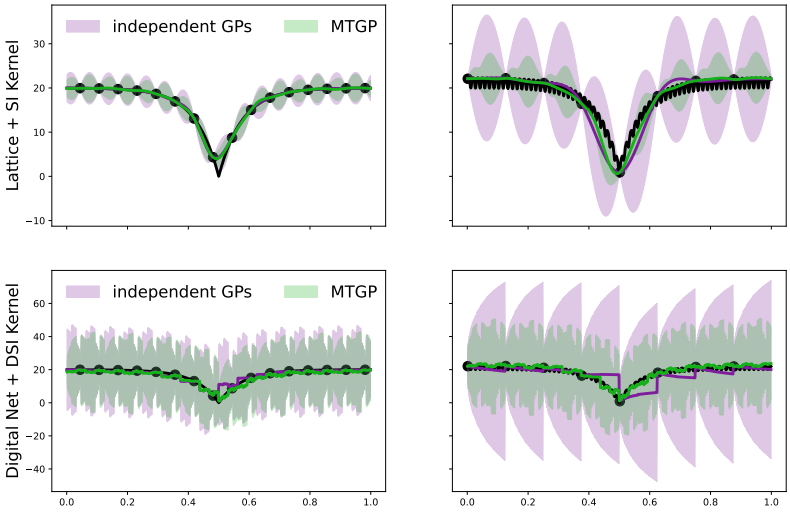
`pip install fastgps`: alegresor.github.io/fastgps/

- Single task and multi-task fast GPs (and baseline standard GPs)
- Kernel hyperparameter optimization (MLL, CV, GCV)
- Fast Bayesian cubature (including for multi-task GPs)
- Efficient batched GPR and GPU support

`pip install qmcpy`: qmcsoftware.github.io/QMCSoftware/

- Quasi-Monte Carlo Python software
- LD sequences with randomizations
  - lattices (random shifts)
  - digital nets (digital shifts, LMS, Owen scrambling (NUS), higher order nets)
  - Halton points (digital shifts, permutation scrambles, NUS)
- SI/DSI kernels of varying smoothness
- Automatic transforms (rewrite problem into $f : [0,1]^d \to \mathbb{R}$)
- Adaptive stopping criteria (choosing N to meet user-specified error tolerance ε)
- Diverse use cases (financial options, parameterized PDEs, sensitivity indices, ...)

Thank you for listening!



References I

Michael Giles. Multilevel monte carlo path simulation. *Operations Research*, 56: 607–617, 06 2008. doi: 10.1287/opre.1070.0496.

Michael B Giles and Benjamin J Waterhouse. Multilevel quasi-monte carlo path simulation. *Advanced Financial Modelling, Radon Series on Computational and Applied Mathematics*, 8:165–181, 2009.

Jagadeeswaran Rathinavel. *Fast automatic Bayesian cubature using matching kernels and designs*. Illinois Institute of Technology, 2019.

Jagadeeswaran Rathinavel and Fred J. Hickernell. Fast automatic bayesian cubature using lattice sampling. *Statistics and Computing*, 29(6):1215–1229, Sep 2019. ISSN 1573-1375. doi: 10.1007/s11222-019-09895-9. URL http://dx.doi.org/10.1007/s11222-019-09895-9.

Jagadeeswaran Rathinavel and Fred J Hickernell. Fast automatic bayesian cubature using sobol sampling. In *Advances in Modeling and Simulation: Festschrift for Pierre L'Ecuyer*, pages 301–318. Springer, 2022.

References II

S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved April 9, 2025, from http://www.sfu.ca/~ssurjano.

Return to slide on **Fast Multi-Task GPs Storage and Costs**

Algorithm 1 Inverse and Determinant of Σ

Require: Σ diagonal block matrix with $m_1 \geqslant \cdots \geqslant m_L$.

$\mathsf{A} \leftarrow \Sigma_{11}^{-1}$ diagonal

$\rho \leftarrow |\Sigma_{11}|$

$\ell \leftarrow 2$

while $\ell \leqslant L$ **do**

$\mathsf{D} \leftarrow \Sigma_{\ell\ell}$ diagonal

$\mathsf{C} \leftarrow \Sigma_{1:\ell-1,\ell}$ diagonal blocks

$\mathsf{S}_\ell \leftarrow \mathsf{D} - \overline{\mathsf{C}}\mathsf{A}^{-1}\mathsf{C}$ diagonal Schur complement

$$\mathsf{A} \leftarrow \begin{pmatrix} \mathsf{A}^{-1} + \mathsf{A}^{-1}\mathsf{C}\mathsf{S}_\ell^{-1}\overline{\mathsf{C}}\mathsf{A}^{-1} & -\mathsf{A}^{-1}\mathsf{C}\mathsf{S}_\ell^{-1} \\ -\mathsf{S}_\ell^{-1}\overline{\mathsf{C}}\mathsf{A}^{-1} & \mathsf{S}_\ell^{-1} \end{pmatrix}$$

$\rho \leftarrow \rho|\mathsf{S}_\ell|$

$\ell \leftarrow \ell + 1$

end while

return $\mathsf{A} = \Sigma^{-1}$ and $\rho = |\Sigma|$