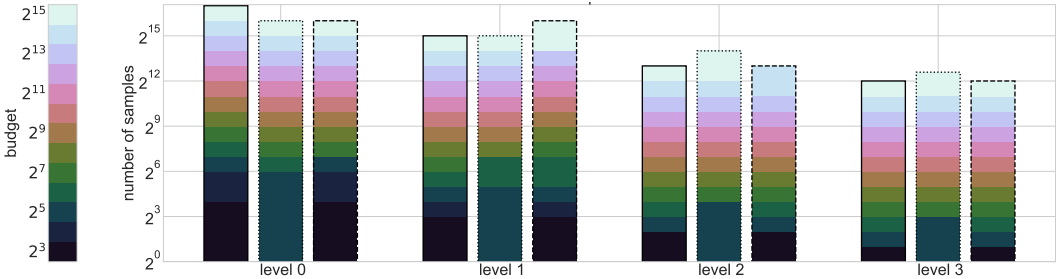
Fast Bayesian Multilevel Quasi-Monte Carlo

Aleksei G. Sorokin[1,2], Pieterjan Robbe[2], Gianluca Geraci[2]
Michael S. Eldred[2], Fred J. Hickernell[1],

[1]Illinois Institute of Technology, Department of Applied Mathematics.
[2] Sandia National Lab.

[1] https://arxiv.org/abs/2510.24604



SIAM PD 2025: Kernel-Based Surrogate Models for Solving PDEs Part I of II

Motivating Example

$2D$ Nonlinear Darcy flow PDE with random forcing term g and conductivity e^a

$$\begin{cases} -\nabla \cdot (e^{a(\boldsymbol{u},\boldsymbol{X})}.\nabla q(\boldsymbol{u},\boldsymbol{X})) + (q(\boldsymbol{u},\boldsymbol{X}))^3 = g(\boldsymbol{u},\boldsymbol{X}), & \boldsymbol{u} \in [0,1]^2 \\ q(\boldsymbol{u},\boldsymbol{X}) = 0, & \boldsymbol{u} \in \partial[0,1]^2. \end{cases}$$

Coefficients (g,a) are independent Gaussian processes with stochasticity $\boldsymbol{X} \sim \mathcal{U}[0,1]^d$.
$q_\ell(u,\boldsymbol{X})$ a numerical PDE solution on a computational grid with $\mathcal{O}(2^{2\ell})$ nodes

- level $\ell \in \{1,\ldots,L\}$ controls solution accuracy: larger $\ell \longrightarrow$ finer mesh
- Numerical PDE solution from a Levenberg–Marquardt iteration [2, 3]
- Stochastic dimension $d = \mathcal{O}(2^{2L})$, nodes in the finest mesh (the largest level)

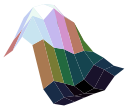
Quantity of interest

$$\mathbb{E}[Q_L(\boldsymbol{X})] := \mathbb{E}\left[\max_{\boldsymbol{u}\in[0,1]^2} q_L(\boldsymbol{u},\boldsymbol{X})\right], \qquad \boldsymbol{X} \sim \mathcal{U}[0,1]^d$$

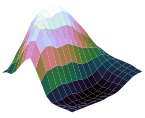
$2D$ Nonlinear Darcy Flow PDE

$$\mathbb{E}[Q_L(\boldsymbol{X})] := \mathbb{E}\left[\max_{\boldsymbol{u}\in[0,1]^2} q_L(\boldsymbol{u}, \boldsymbol{X})\right], \qquad \boldsymbol{X} \sim \mathcal{U}[0,1]^d$$

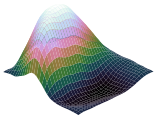
solution $q_0(\boldsymbol{u}, \boldsymbol{x})$
$Q_0(\boldsymbol{x}) = 0.0288$



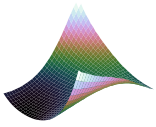
solution $q_1(\boldsymbol{u}, \boldsymbol{x})$
$Q_1(\boldsymbol{x}) = 0.0374$



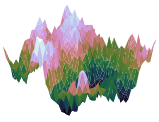
solution $q_2(\boldsymbol{u}, \boldsymbol{x})$
$Q_2(\boldsymbol{x}) = 0.0369$



conductivity $e^{a(\boldsymbol{u}, \boldsymbol{x})}$

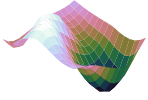


forcing term $g(\boldsymbol{u}, \boldsymbol{x})$

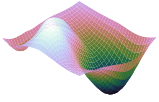


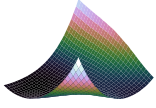
$Q_0(\boldsymbol{x}) = 0.0302$

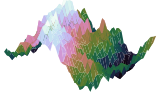
$Q_1(\boldsymbol{x}) = 0.0281$



$Q_2(\boldsymbol{x}) = 0.0349$







The Multilevel Monte Carlo Problem

Can we accelerate high-fidelity UQ by exploiting cheaper low-fidelity surrogates?

$Q_\ell : [0,1]^d \to \mathbb{R}$ simulates at (fidelity) level $\ell \in \{1, \ldots, L\}$ with cost C_ℓ increasing in ℓ.

- e.g., Q_ℓ a (possibly nonlinear) functional of numerical PDE solution q_ℓ

Stochasticity identified with $\boldsymbol{X} \sim \mathcal{U}[0,1]^d$ (after a possible transform)

- e.g., $\boldsymbol{X}$ used to generate random PDE coefficients
- d may equal number of nodes in the computational grid
- d may equal number of coefficients in truncated Karhunen–Loève expansion
- d may change with level, e.g., d_ℓ the number of monitoring times in option pricing

Quantity of interest is the expectation (mean) at the highest-level (highest-fidelity)

$$\nu = \mathbb{E}[Q_L(\boldsymbol{X})] = \sum_{\ell=1}^{L} \mathbb{E}[Q_\ell(\boldsymbol{X}) - Q_{\ell-1}(\boldsymbol{X})] = \sum_{\ell=1}^{L} \mathbb{E}[Y_\ell(\boldsymbol{X})] = \sum_{\ell=1}^{L} \mu_\ell$$

where $Y_\ell := Q_\ell - Q_{\ell-1}$, $\mu_\ell := \mathbb{E}[Y_\ell]$, and we set $Q_0 = 0$ so $Y_1 = Q_1$.

Multilevel Monte Carlo (MLMC) with IID Points [4]

$$\nu = \mathbb{E}[Q_L(\boldsymbol{X})] = \sum_{\ell=1}^{L} \mathbb{E}[Q_\ell(\boldsymbol{X}) - Q_{\ell-1}(\boldsymbol{X})] = \sum_{\ell=1}^{L} \mathbb{E}[Y_\ell(\boldsymbol{X})] = \sum_{\ell=1}^{L} \mu_\ell, \qquad \boldsymbol{X} \sim \mathcal{U}[0,1]^d.$$

Approximate ν by the sum of n_ℓ-sample Monte Carlo approximations

$$\widehat{\nu} = \sum_{\ell=1}^{L} \widehat{\mu}_\ell \qquad \text{where} \qquad \widehat{\mu}_\ell = \frac{1}{n_\ell} \sum_{i=0}^{n_\ell - 1} Y_\ell(\boldsymbol{x}_{\ell,i}),$$

using IID points $\boldsymbol{x}_{1,0}, \ldots, \boldsymbol{x}_{1,n_1-1}, \ldots, \boldsymbol{x}_{L,0}, \ldots, \boldsymbol{x}_{L,n_L-1} \overset{\text{IID}}{\sim} \mathcal{U}[0,1]^d$.
$\therefore$ $\mathbb{V}[\widehat{\nu}] = \sum_{\ell=1}^{L} V_\ell / n_\ell$ for variance $V_\ell = \mathbb{V}[Y_\ell(\boldsymbol{X})]$.
$\therefore$ for a fixed error tolerance, optimal allocation $n_\ell \propto \sqrt{V_\ell / C_\ell}$.

Algorithm 1 MC: MLMC with IID Points

Require: $N, \quad C_\ell, \quad n_\ell^{\text{next}}$ ▷ *budget, costs, initial samples*

$n_\ell \leftarrow 0$ for $\ell \in \{1, \dots, L\}$ and $\mathcal{L} \leftarrow \{1, \dots, L\}$

while true **do**

Generate $\boldsymbol{x}_{\ell,i} \sim \mathcal{U}[0,1]^d$ for $\ell \in \mathcal{L}$ and $n_\ell \leqslant i < n_\ell^{\text{next}}$ all independently

Evaluate $Y_\ell(\boldsymbol{x}_{\ell,i})$ for $\ell \in \mathcal{L}$ and $n_\ell \leqslant i < n_\ell^{\text{next}}$

$\widehat{\mu}_\ell \leftarrow \frac{1}{n_\ell} \sum_{i=0}^{n_\ell - 1} Y_\ell(\boldsymbol{x}_{\ell,i})$ for $\ell \in \mathcal{L}$ ▷ *estimate of* μ_ℓ

$\sigma_\ell^2 \leftarrow \frac{1}{n_\ell - 1} \sum_{i=0}^{n_\ell - 1} (Y_\ell(\boldsymbol{x}_{\ell,i}) - \widehat{\mu}_\ell)^2$ for $\ell \in \mathcal{L}$ ▷ *estimate of* V_ℓ

$\mathcal{L}_{\text{feasible}} \leftarrow \{\ell \in \{1, \dots, L\} : \sum_{\ell'=1}^{L} C_{\ell'} n_{\ell'} + C_\ell n_\ell \leqslant N\}$

if $\mathcal{L}_{\text{feasible}} = \varnothing$ **then** break ▷ *exit loop before going over-budget*

$\overset{\star}{\ell} \leftarrow \operatorname{argmax}_{\ell \in \mathcal{L}_{\text{feasible}}} \frac{\sigma_\ell^2}{n_\ell C_\ell}$ ▷ *chose the level of maximum utility*

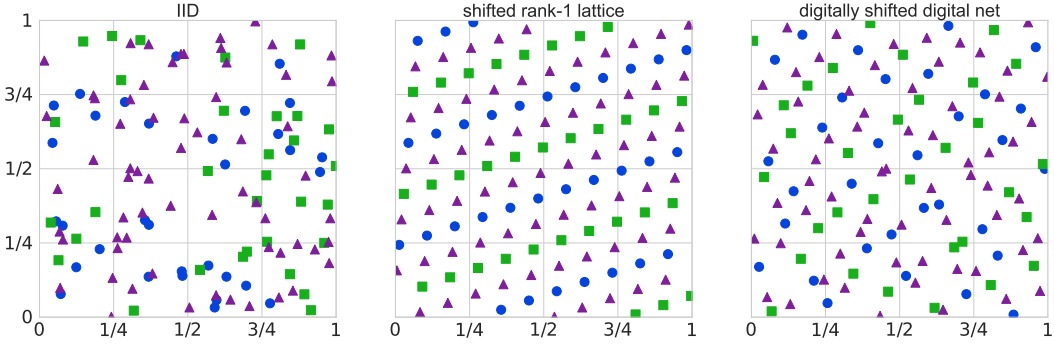
$\mathcal{L} \leftarrow \{\overset{\star}{\ell}\}$ and $n_{\overset{\star}{\ell}} \leftarrow n_{\overset{\star}{\ell}}^{\text{next}}$ and $n_{\overset{\star}{\ell}}^{\text{next}} \leftarrow 2 n_{\overset{\star}{\ell}}$ ▷ *double samples on chosen level*

$\widehat{\nu} \leftarrow \sum_{\ell=1}^{L} \widehat{\mu}_\ell$ and $\sigma^2 \leftarrow \sum_{\ell=1}^{L} \widehat{\sigma}_\ell^2 / n_\ell$ ▷ *estimates of* ν *and* $\mathbb{V}[\widehat{\nu}]$ *respectively*

return $\widehat{\nu}, \sigma, \{n_\ell\}_{\ell=1}^{L}$ ▷ *the estimate, its standard error, and samples per level*

Quasi-Monte Carlo (QMC) Methods

QMC uses highly-uniform low-discrepancy (LD) sequences instead of IID points to accelerate Monte Carlo

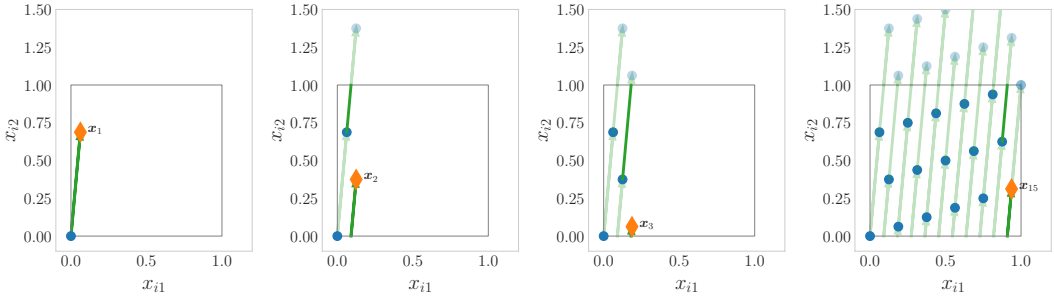


IID Monte Carlo has error like $\mathcal{O}(1/\sqrt{n})$. IID points have gaps and clusters.
QMC has errors like $\mathcal{O}(1/n)$. LD lattices and digital nets evenly cover the unit cube.

Unrandomized LD Lattice Construction for QMC

Generating vector $\boldsymbol{g} \in \mathbb{N}^d$ defines unrandomized lattice $(\boldsymbol{z}_i)_{i=0}^{n-1} \subset [0,1)^d$ with

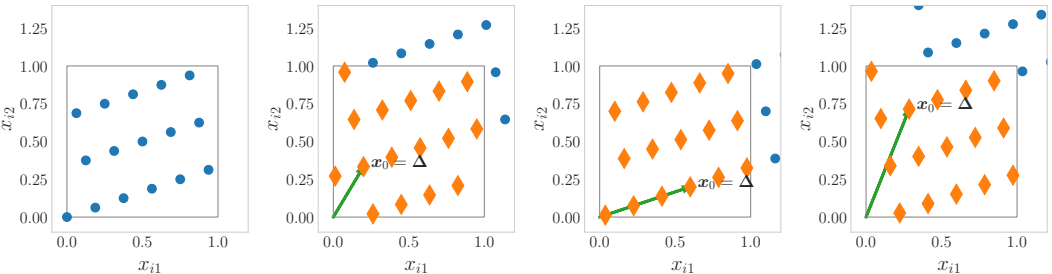
$$\boldsymbol{z}_i = (i\boldsymbol{g}/n) \mod \mathbf{1}, \qquad i = 0, \ldots, n-1.$$



Randomized LD Lattice Construction for QMC

A single random shift $\boldsymbol{\Delta} \in [0,1)^d$ defines the randomly shifted lattice $(\boldsymbol{x}_i)_{i=0}^{n-1} \subset [0,1)^d$

$$\boldsymbol{x}_i = \boldsymbol{z}_i \oplus \boldsymbol{\Delta} := (\boldsymbol{z}_i + \boldsymbol{\Delta}) \mod 1, \qquad i = 0, \ldots, n-1.$$



$R > 1$ independently randomized LD sequences gives R independent QMC estimates [5]

Multilevel QMC (MLQMC) with Replications [6]

Existing MLQMC methods use $R > 1$ independently randomized (replications) of an LD sequence

$$\nu = \mathbb{E}[Q_L(\boldsymbol{X})] = \sum_{\ell=1}^{L} \mathbb{E}[Q_\ell(\boldsymbol{X}) - Q_{\ell-1}(\boldsymbol{X})] = \sum_{\ell=1}^{L} \mathbb{E}[Y_\ell(\boldsymbol{X})] = \sum_{\ell=1}^{L} \mu_\ell, \qquad \boldsymbol{X} \sim \mathcal{U}[0,1]^d.$$

$R > 1$ independent shifts on each level gives $(\boldsymbol{x}_{\ell,r,i})_{i=0}^{n_\ell - 1}$ for $1 \leqslant \ell \leqslant L$ and $1 \leqslant r \leqslant R$:

$$\boldsymbol{x}_{\ell,r,i} = \boldsymbol{z}_i \oplus \boldsymbol{\Delta}_{\ell,r}, \qquad \boldsymbol{\Delta}_{1,1}, \ldots, \boldsymbol{\Delta}_{1,R}, \ldots, \boldsymbol{\Delta}_{L,1}, \ldots, \boldsymbol{\Delta}_{L,r} \overset{\text{IID}}{\sim} \mathcal{U}[0,1)^d.$$

Approximate ν by the resulting LR independent sample means $\widetilde{\mu}_{\ell,r}$,

$$\widehat{\nu} = \sum_{\ell=1}^{L} \widehat{\mu}_\ell \qquad \text{where} \qquad \widehat{\mu}_\ell = \frac{1}{R} \sum_{r=1}^{R} \widetilde{\mu}_{\ell,r} \qquad \text{and} \qquad \widetilde{\mu}_{\ell,r} = \frac{1}{n_\ell} \sum_{i=0}^{n_\ell - 1} Y_\ell(\boldsymbol{x}_{\ell,r,i}).$$

$\therefore\ \mathbb{V}[\widehat{\nu}] = \sum_{\ell=1}^{L} \widetilde{V}_{\ell,n_\ell}/R$ for $\widetilde{V}_{\ell,n_\ell} := \mathbb{V}[\frac{1}{n_\ell} \sum_{i=0}^{n_\ell - 1} Y_\ell(\boldsymbol{z} \oplus \boldsymbol{\Delta})] \xrightarrow{n_\ell \to \infty} 0$ for $\boldsymbol{\Delta} \sim \mathcal{U}[0,1)^d$.

Algorithm 2 RQMC: MLQMC with Replications

Require: $N, \quad C_\ell, \quad n_\ell^{\text{next}}, \quad R$ ▷ *budget, costs, initial samples, replications*

Generate $\boldsymbol{\Delta}_{1,1},\dots,\boldsymbol{\Delta}_{1,R},\dots,\boldsymbol{\Delta}_{L,1},\dots,\boldsymbol{\Delta}_{L,r} \overset{\text{IID}}{\sim} \mathcal{U}[0,1)^d$

$n_\ell \leftarrow 0$ for $\ell \in \{1,\dots,L\}$ and $\mathcal{L} \leftarrow \{1,\dots,L\}$

while true **do**

Generate $x_{\ell,r,i} = \boldsymbol{z}_i \oplus \boldsymbol{\Delta}_{\ell,r}$ for $\ell \in \mathcal{L}$ and $1 \leqslant r \leqslant R$ and $n_\ell \leqslant i < n_\ell^{\text{next}}$

Evaluate $Y_\ell(\boldsymbol{x}_{\ell,r,i})$ for $\ell \in \mathcal{L}$ and $1 \leqslant r \leqslant R$ and $n_\ell \leqslant i < n_\ell^{\text{next}}$

$\widetilde{\mu}_{\ell,r} \leftarrow \frac{1}{n_\ell}\sum_{i=0}^{n_\ell-1} Y_\ell(\boldsymbol{x}_{\ell,r,i})$ for $\ell \in \mathcal{L}$ and $1 \leqslant r \leqslant R$ ▷ *per-randomization estimate of μ_ℓ*

$\widehat{\mu}_\ell \leftarrow \frac{1}{R}\sum_{r=1}^{R} \widetilde{\mu}_{\ell,r}$ for $\ell \in \mathcal{L}$ ▷ *aggregate estimate of mean μ_ℓ*

$\widetilde{\sigma}_\ell^2 \leftarrow \frac{1}{R-1}\sum_{r=1}^{R} (\widetilde{\mu}_{\ell,r} - \widehat{\mu}_\ell)^2$ for $\ell \in \mathcal{L}$ ▷ *estimate of variance $\widetilde{V}_{\ell,n_\ell}$*

$\mathcal{L}_{\text{feasible}} \leftarrow \{\ell \in \{1,\dots,L\} : \sum_{\ell'=1}^{L} RC_{\ell'}n_{\ell'} + RC_\ell n_\ell \leqslant N\}$

if $\mathcal{L}_{\text{feasible}} = \varnothing$ **then** break ▷ *exit loop before going over budget*

$\overset{\star}{\ell} \leftarrow \operatorname{argmax}_{\ell \in \mathcal{L}_{\text{feasible}}} \frac{\widetilde{\sigma}_\ell^2}{Rn_\ell C_\ell}$ ▷ *choose the level of maximum utility*

$\mathcal{L} \leftarrow \{\overset{\star}{\ell}\}$ and $n_{\overset{\star}{\ell}} \leftarrow n_{\overset{\star}{\ell}}^{\text{next}}$ and $n_{\overset{\star}{\ell}}^{\text{next}} \leftarrow 2n_{\overset{\star}{\ell}}$ ▷ *double samples on chosen level*

$\widehat{\nu} \leftarrow \sum_{\ell=1}^{L} \widehat{\mu}_\ell$ and $\widetilde{\sigma}^2 \leftarrow \sum_{\ell=1}^{L} \widetilde{\sigma}_\ell^2/R$ ▷ *estimate for ν and $\mathbb{V}[\widehat{\nu}]$ respectively*

return $\widehat{\nu}, \widetilde{\sigma}, \{Rn_\ell\}_{\ell=1}^{L}$ ▷ *the estimate, its standard error, and samples per level*

QMC with Multiple Replications is Inefficient

Bayesian cubature enables QMC error estimation without inefficient replications

- Most QMC methods use replications only for error estimation
- Fewer replications R gives a more accurate QMC estimate $\widehat{\mu}_\ell$, but a worse estimate for the predicted error $\widetilde{\sigma}_\ell$
- Replications R trades off accuracy of the approximation for accuracy of the error estimate
- MLQMC needs both accurate approximations and estimates

Can we avoid replications while still getting an error estimate?
Yes! Using Bayesian cubature with LD sampling locations

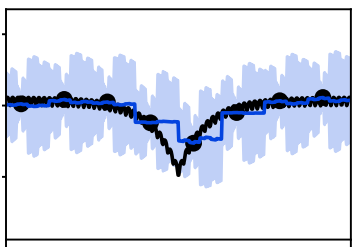
- A single LD sequence is used to construct a Gaussian process (GP).
- Error estimation using GPs posterior variance.
- Bonus: GPs with matching kernels and LD points cost only $\mathcal{O}(n \log n)$ to fit

Gaussian Process Regression [7]

Gaussian process $Y \sim \mathrm{GP}(\tau, K)$ has posterior mean and covariance

$$\mathbb{E}[Y(\boldsymbol{x})|\mathsf{X}, \boldsymbol{Y}] = \tau + \boldsymbol{K}^\intercal(\boldsymbol{x})\mathsf{K}^{-1}(\boldsymbol{Y} - \tau\mathbf{1}) \qquad \text{and}$$
$$\mathrm{Cov}[Y(\boldsymbol{x}), Y(\boldsymbol{x}')|\mathsf{X}] = K(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{K}^\intercal(\boldsymbol{x})\mathsf{K}^{-1}\boldsymbol{K}(\boldsymbol{x}')$$

- Constant prior mean $\tau \in \mathbb{R}$
- Nodes $\mathsf{X} := (\boldsymbol{x}_i)_{i=0}^{n-1}$
- Evaluations $\boldsymbol{Y} := (Y(\boldsymbol{x}_i))_{i=0}^{n-1} \in \mathbb{R}^n$
- Kernel vector $\boldsymbol{K}(\boldsymbol{x}) = (K(\boldsymbol{x}, \boldsymbol{x}_i))_{i=0}^{n-1} \in \mathbb{R}^n$
- SPD Gram matrix $\mathsf{K} = (K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}))_{i,i'=0}^{n-1} \in \mathbb{R}^{n\times n}$



Kernel hyperparameters $\boldsymbol{\theta}$ often optimized with NMLL loss

$$L(\boldsymbol{\theta}) = \log|\mathsf{K}| + (\boldsymbol{Y} - \tau\mathbf{1})^\intercal\mathsf{K}^{-1}(\boldsymbol{Y} - \tau\mathbf{1})$$

GP regression typically requires $\mathcal{O}(n^2)$ storage and $\mathcal{O}(n^3 + n^2 d)$ computations.

Fast GPs Pairing LD Points with Special Kernels [8, 9]

Accelerate GP regression by inducing nice structure into the Gram matrix K

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 $\mathcal{O}(n \log n)$
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 $\mathcal{O}(n \log n)$

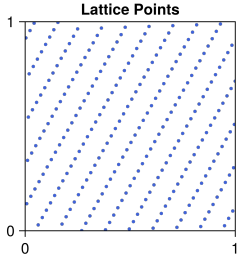
Let $\boldsymbol{v}_1 = \mathbf{1}/\sqrt{n}$ and $\boldsymbol{k}_1$ be the first columns of $\overline{\boldsymbol{V}}$ and K respectively:

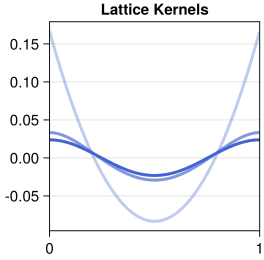
$$\boldsymbol{\lambda} := \Lambda\mathbf{1} = \sqrt{n}\Lambda\overline{\boldsymbol{v}}_1 = \sqrt{n}\overline{\mathsf{V}}\mathsf{V}\Lambda\overline{\boldsymbol{v}}_1 = \sqrt{n}\overline{\mathsf{V}}\boldsymbol{k}_1$$

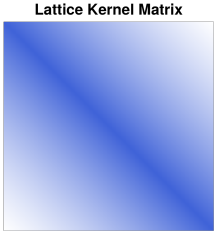
- $\mathsf{K}\boldsymbol{a}$, $\mathsf{K}^{-1}\boldsymbol{a}$, and $|\mathsf{K}|$ can all be computed in $\mathcal{O}(n \log n)$ computations
- Only requires evaluating and storing the first column of K

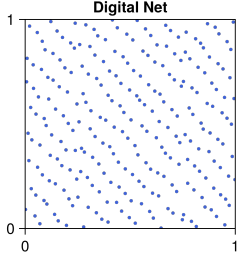
Fast GP regression only requires $\mathcal{O}(n)$ storage and $\mathcal{O}(n \log n + nd)$ computations.

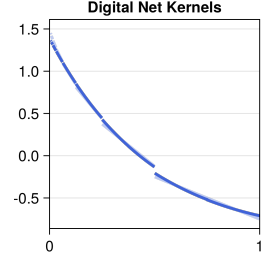
Lattices + SI K = Circulant K and Digital Nets + DSI K = RSBT K

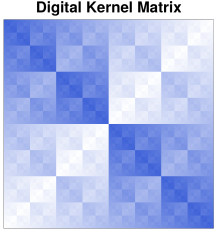












Fast Bayesian Cubature [10, 11, 12]

$$\mu = \mathbb{E}_{\boldsymbol{X}}[Y(\boldsymbol{X})] \sim \mathcal{N}(\widehat{\mu}, \widehat{V}_n), \qquad \boldsymbol{X} \sim \mathcal{U}[0,1]^d$$

with respective posterior cubature mean and variance

$$\widehat{\mu} := \mathbb{E}[\mu|\mathsf{X}, \boldsymbol{Y}] = \tau + \left(\int \boldsymbol{K}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\right)^{\mathsf{T}} \mathsf{K}^{-1}(\boldsymbol{Y} - \tau\boldsymbol{1}) \qquad \text{and}$$

$$\widehat{V}_n := \mathbb{V}[\mu|\mathsf{X}] = \int\int K(\boldsymbol{x}, \boldsymbol{x}')\mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{x}' - \left(\int \boldsymbol{K}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\right)^{\mathsf{T}} \mathsf{K}^{-1} \left(\int \boldsymbol{K}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\right).$$

Certain fast GP constructions make the GP prediction $\widehat{\mu}$ equal the QMC sample mean:

$$\widehat{\mu} = \tau = \frac{1}{n}\sum_{i=0}^{n-1} Y(\boldsymbol{x}_i)$$

Fast Bayesian MLQMC without Replications

$$\mu_\ell = \mathbb{E}_{\boldsymbol{X}}[Y_\ell(\boldsymbol{X})] \sim \mathcal{N}(\widehat{\mu}_\ell, \widehat{V}_{\ell,n_\ell}), \qquad \boldsymbol{X} \sim \mathcal{U}[0,1]^d$$

Posterior cubature mean

$$\widehat{\nu} := \mathbb{E}[\nu|(\mathsf{X}_\ell, \boldsymbol{Y}_\ell)_{\ell=1}^L] = \sum_{\ell=1}^{\ell} \widehat{\mu}_\ell \qquad \text{where} \qquad \widehat{\mu}_\ell := \mathbb{E}[\mu_\ell|\mathsf{X}_\ell, \boldsymbol{Y}_\ell] = \frac{1}{n_\ell} \sum_{i=0}^{n_\ell-1} Y_\ell(\boldsymbol{x}_{\ell,i})$$

and computable posterior cubature variance (assuming independent GPs)

$$\widehat{V}_{n_1,\dots,n_L} := \mathbb{V}[\nu|(\mathsf{X}_\ell)_{\ell=1}^L] = \sum_{\ell=1}^{L} \widehat{V}_{\ell,n_\ell} \quad \text{where} \quad \widehat{V}_{\ell,n_\ell} \text{ only depends on } \mathsf{X}_\ell \qquad (1)$$

Use projected variance, $\widehat{V}_{\ell,\widehat{n}_\ell}$ with $\widehat{n}_\ell \geqslant n_\ell$, to decide on which level to double samples

Algorithm 3 `BQMC`: Fast Bayesian MLQMC Without Replications (Independent GPs)

Require: N, C_ℓ, n_ℓ^{next}, $\boldsymbol{\theta}_\ell$ ▷ *budget, costs, initial samples, initial kernel parameters*

$\boldsymbol{\Delta}_1, \ldots, \boldsymbol{\Delta}_L \overset{\text{IID}}{\sim} \mathcal{U}[0,1)^d$ ▷ *random shifts, one per level*

$n_\ell \leftarrow 0$ for $\ell \in \{1, \ldots, L\}$ and $\mathcal{L} \leftarrow \{1, \ldots, L\}$

while true **do**

- Generate $x_{\ell,i} = \boldsymbol{z}_i \oplus \boldsymbol{\Delta}_\ell$ for $\ell \in \mathcal{L}$ and $n_\ell \leqslant i < n_\ell^{\text{next}}$
- Evaluate $Y_\ell(\boldsymbol{x}_{\ell,i})$ for $\ell \in \mathcal{L}$ and $n_\ell \leqslant i < n_\ell^{\text{next}}$
- Update $\boldsymbol{\theta}_\ell$ to optimize the NMLL for $\ell \in \mathcal{L}$
- $\widehat{\mu}_\ell \leftarrow \frac{1}{n_\ell} \sum_{i=0}^{n_\ell - 1} Y_\ell(\boldsymbol{x}_{\ell,i})$ for $\ell \in \mathcal{L}$ ▷ *per-level posterior cubature mean*
- Compute $\widehat{V}_{\ell,n_\ell}$ for $\ell \in \mathcal{L}$ ▷ *per-level posterior cubature variance*
- $\mathcal{L}_{\text{feasible}} \leftarrow \{\ell \in \{1, \ldots, L\} : \sum_{\ell'=1}^{L} C_{\ell'} n_{\ell'} + C_\ell n_\ell \leqslant N\}$
- **if** $\mathcal{L}_{\text{feasible}} = \varnothing$ **then** break ▷ *exit loop before going over budget*
- $\overset{\star}{\ell} \leftarrow$ `level_select_BQMC`$\big(\mathcal{L}_{\text{feasible}}, (C_\ell, \boldsymbol{\theta}_\ell, n_\ell)_{\ell=1}^{L}\big)$ ▷ *choose level with Algorithm 4*
- $\mathcal{L} \leftarrow \{\overset{\star}{\ell}\}$ and $n_{\overset{\star}{\ell}} \leftarrow n_{\overset{\star}{\ell}}^{\text{next}}$ and $n_{\overset{\star}{\ell}}^{\text{next}} \leftarrow 2n_{\overset{\star}{\ell}}$ ▷ *double samples on chosen level*

$\widehat{\nu} \leftarrow \sum_{\ell=1}^{L} \widehat{\mu}_\ell$ and $\widehat{V}_{n_1,\ldots,n_L} \leftarrow \sum_{\ell=1}^{L} \widehat{V}_{\ell,n_\ell}$ ▷ *posterior cubature mean and variance respectively*

return $\widehat{\nu}, \sqrt{\widehat{V}_{n_1,\ldots,n_L}}, \{n_\ell\}_{\ell=1}^{L}$ ▷ *the estimate, its standard error, and samples per level*

Numerical Experiments: Single-Level Functions in $d = 32$ Dimensions

Ridge functions $Y(\boldsymbol{x}) = g(u(\boldsymbol{x}))$ use $u(\boldsymbol{x}) = \sum_{j=1}^{d} c_j \Phi^{-1}(x_j)$ with weights $c_j = 2^{-j}/\sqrt{\sum_{j'=1}^{d} 2^{-2j'}}$ and Φ the normal CDF.

1. sumxex $Y(\boldsymbol{x}) = -d + \sum_{j=1}^{d} x_j \exp(x_j)$. Smooth and additive. Easy for QMC.
2. ridge PL sparse $Y(\boldsymbol{x}) = g(u(\boldsymbol{x}))$ with $g(u) = \max(u-1, 0) - \phi(1) + \Phi(-1)$ and ϕ the density of $\mathcal{N}(0,1)$. Continuous piecewise linear (PL) with a kink.
3. ridge JSU sparse $Y(\boldsymbol{x}) = g(u(\boldsymbol{x}))$ with $g(u) = -\eta + F^{-1}(\Phi(v))$ where F is the CDF of a Johnson's SU distribution [13]. Heavy tailed.
4. Genz corner-peak 2 $Y(\boldsymbol{x}) = \left(1 + \sum_{j=1}^{d} c_j x_j\right)^{-(d+1)}$ with coefficients of the second kind $c_j = j^{-2}/(4\sum_{j'=1}^{d}(j')^{-2})$. A Genz function [14] with moderate anisotropy and effective dimension.

Numerical Experiments: Multilevel Problems

Decay of the means μ_ℓ and standard deviations $\sqrt{V_\ell}$

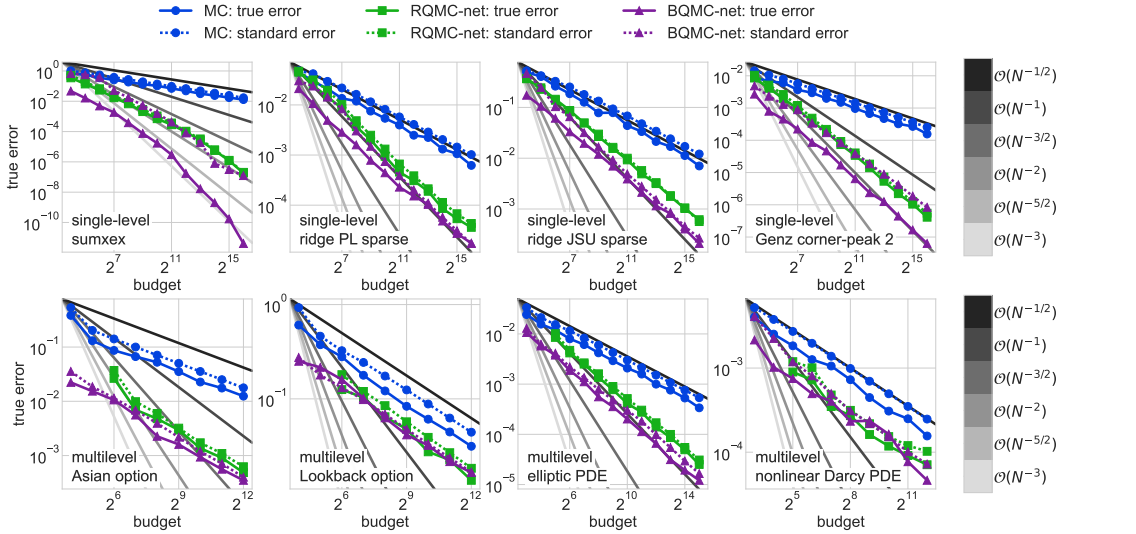
mean $\mu_\ell = \mathbb{E}[Y_\ell(\boldsymbol{X})] = \mathbb{E}[Q_\ell(\boldsymbol{X}) - Q_{\ell-1}(\boldsymbol{X})]$ with $\boldsymbol{X} \sim \mathcal{U}[0,1]^d$

problem/ℓ	1	2	3	4	5	6	7	8
Asian option	6.3e0	-3.0e-1	-1.5e-1	-7.2e-2	-3.7e-2	-1.8e-2	-9.3e-3	-4.8e-3
Lookback option	1.3e1	1.4e0	8.9e-1	5.8e-1	4.1e-1	2.8e-1	1.8e-1	1.3e-1
elliptic PDE	1.6e-1	-1.1e-2	2.5e-3	1.5e-3				
nonlinear Darcy PDE	4.1e-2	8.5e-4	3.3e-4					

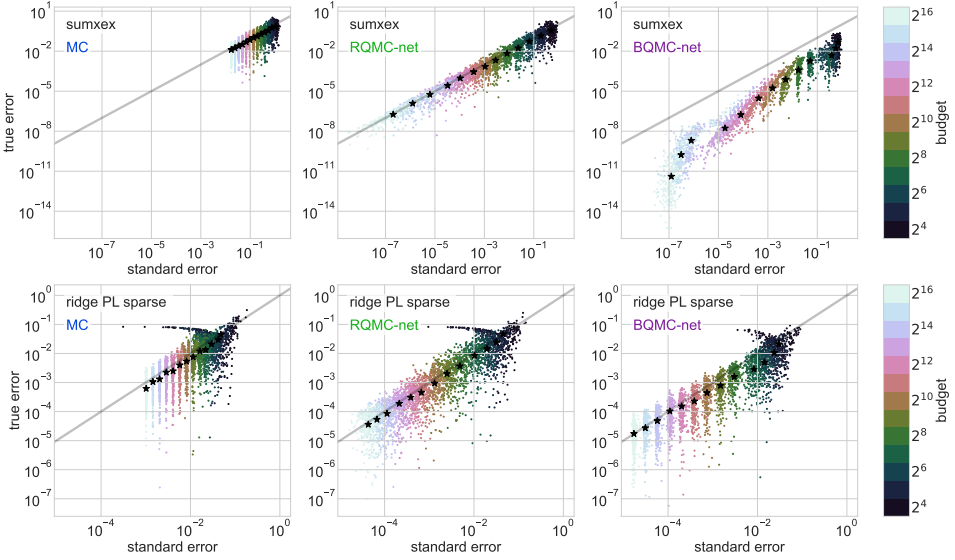
standard deviation $\sqrt{V_\ell} = \sqrt{\mathbb{V}[Y_\ell(\boldsymbol{X})]} = \sqrt{\mathbb{V}[Q_\ell(\boldsymbol{X}) - Q_{\ell-1}(\boldsymbol{X})]}$ with $\boldsymbol{X} \sim \mathcal{U}[0,1]^d$

problem/ℓ	1	2	3	4	5	6	7	8
Asian option	8.7e0	6.8e-1	3.5e-1	1.7e-1	8.4e-2	4.3e-2	2.1e-2	1.1e-2
Lookback option	1.3e1	2.1e0	1.4e0	9.0e-1	6.3e-1	4.2e-1	2.9e-1	2.0e-1
elliptic PDE	1.4e-1	6.2e-2	1.0e-2	3.5e-3				
nonlinear Darcy PDE	7.8e-2	6.7e-3	1.9e-3					

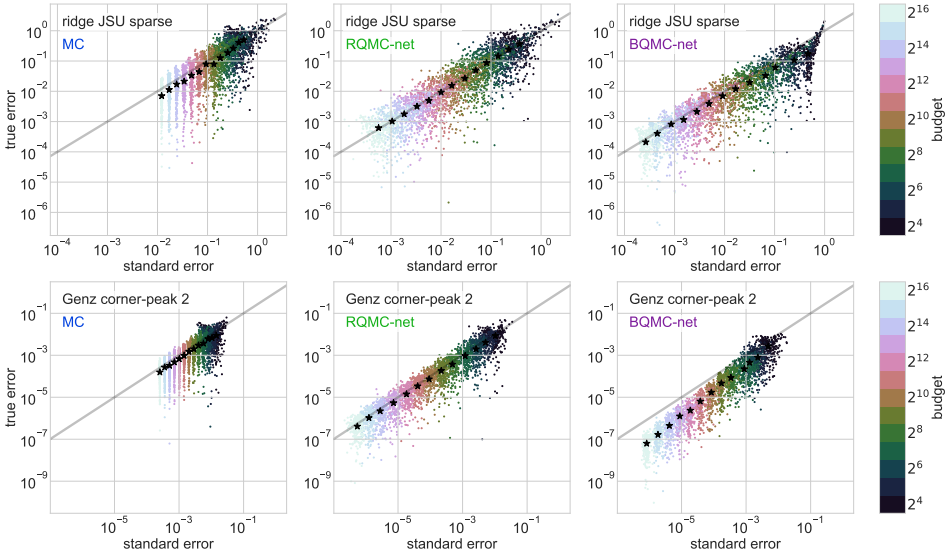
Convergence with Increasing Budgets: LD Digital Nets



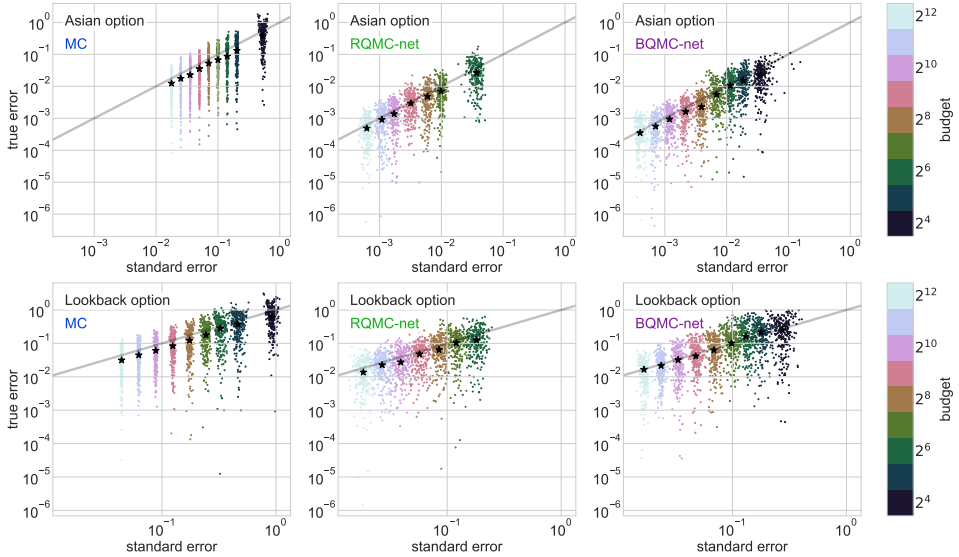
Single-Level Error Estimate Accuracy: LD Digital Nets



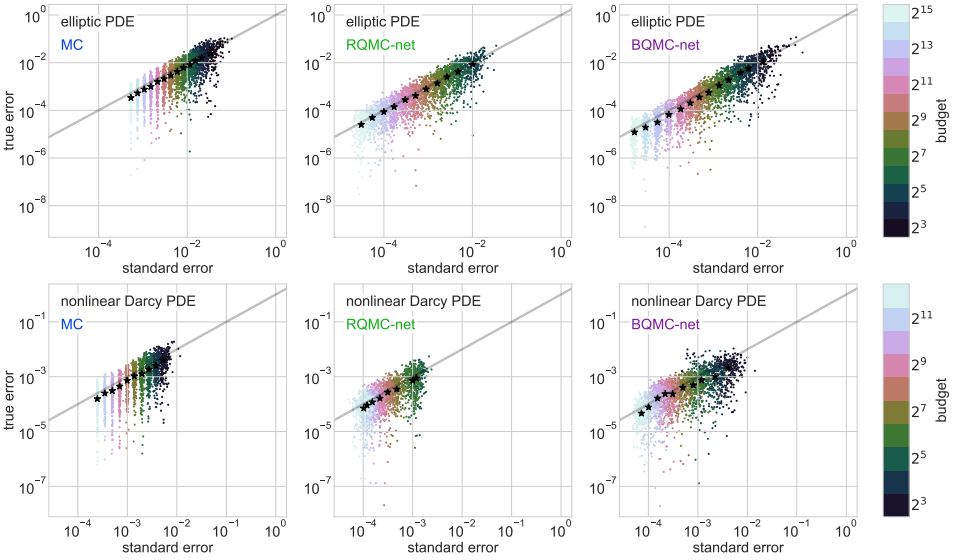
Single-Level Error Estimate Accuracy: LD Digital Nets



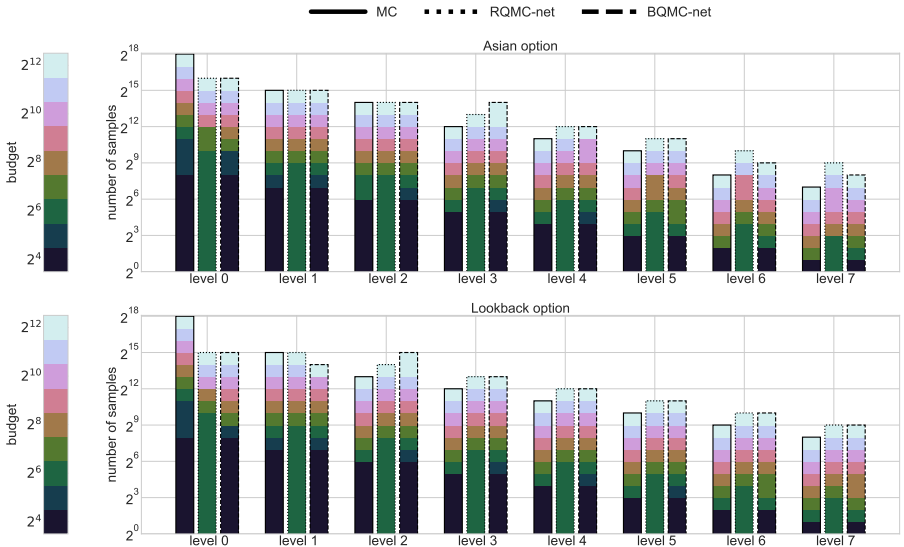
Multilevel Error Estimate Accuracy: LD Digital Nets



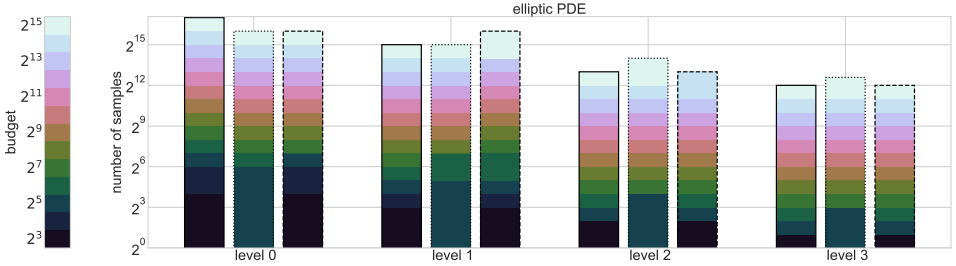
Multilevel Error Estimate Accuracy: LD Digital Nets

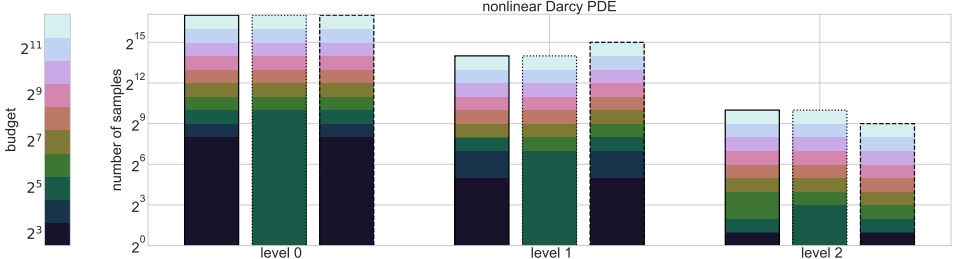


Multilevel Sample Allocation: LD Digital Nets



Multilevel Sample Allocation: LD Digital Nets





Summary and Software

[1] https://arxiv.org/abs/2510.24604

Algorithms

1. Multilevel Monte Carlo with IID points (MLMC)
2. Multilevel Quasi-Monte Carlo with Replications (RMLQMC)
3. Fast Bayesian Multilevel Quasi-Monte Carlo without Replications (BMLQMC)

Python Software

1. `FastGPs`: fast $\mathcal{O}(n \log n)$ GP regression using matching kernels and LD points
   https://alegresor.github.io/fastgps/
2. `QMCPy`: QMC tools for LD points, transforms, stopping criteria, ... [15, 16, 17, 18]
   https://qmcsoftware.github.io/QMCSoftware/

Thank you for listening!

References I

[1] Aleksei G. Sorokin, Pieterjan Robbe, Gianluca Geraci, Michael S. Eldred, and Fred J. Hickernell. Fast Bayesian multilevel quasi-Monte Carlo. *ArXiv preprint*, abs/2510.24604, 2025. URL `https://arxiv.org/abs/2510.24604`.

[2] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[3] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2): 431–441, 1963.

[4] Michael B. Giles. Multilevel Monte Carlo path simulation. *Operations research*, 56(3):607–617, 2008.

[5] Pierre L'Ecuyer, Marvin K. Nakayama, Art B. Owen, and Bruno Tuffin. Confidence intervals for randomized quasi-Monte Carlo estimators. In *Proceedings of the Winter Simulation Conference 2023*, pages 445–456. IEEE, 2023.

References II

[6] 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.

[7] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. MIT Press, Cambridge, Massachusetts, 2006. URL `http://www.gaussianprocess.org/gpml/`.

[8] Xiaoyan Zeng, Peter Kritzer, and Fred J. Hickernell. Spline methods using integration lattices and digital nets. *Constructive Approximation*, 30:529–555, 2009.

[9] Xiaoyan Zeng, King-Tai Leung, and Fred J. Hickernell. *Error analysis of splines for periodic problems using lattice designs*. Springer, 2006.

References III

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

[11] 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.

[12] Jagadeeswaran Rathinavel. *Fast automatic Bayesian cubature using matching kernels and designs*. PhD thesis, Illinois Institute of Technology, 2019.

[13] Norman L. Johnson. Systems of frequency curves generated by methods of translation. *Biometrika*, 36(1/2):149–176, 1949.

[14] Alan Genz. Comparison of methods for the computation of multivariate normal probabilities. *Computing Science and Statistics*, pages 400–400, 1993.

References IV

[15] Sou-Cheng T. Choi, Fred J. Hickernell, Jagadeeswaran Rathinavel, Michael J. McCourt, and Aleksei G. Sorokin. Quasi-Monte Carlo software. In Alexander Keller, editor, *Monte Carlo and Quasi-Monte Carlo Methods 2020*, pages 23–47, Cham, 2022. Springer International Publishing. ISBN 978-3-030-98319-2.

[16] Aleksei G. Sorokin and Jagadeeswaran Rathinavel. On bounding and approximating functions of multiple expectations using quasi-Monte Carlo. In Aicke Hinrichs, Peter Kritzer, and Friedrich Pillichshammer, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2022*, pages 583–599. Springer, 2022.

[17] Sou-Cheng T. Choi, Yuhan Ding, Fred J. Hickernell, Jagadeeswaran Rathinavel, and Aleksei G. Sorokin. Challenges in developing great quasi-Monte Carlo software. In Aicke Hinrichs, Peter Kritzer, and Friedrich Pillichshammer, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2022*, pages 209–222. Springer, 2022.

References V

[18] Fred J. Hickernell, Nathan Kirk, and Aleksei G. Sorokin. Quasi-Monte Carlo methods: what, why, and how? *ArXiv preprint*, abs/2502.03644, 2025. URL https://arxiv.org/abs/2502.03644.

Algorithm 4 `level_select_BQMC`: Level selection for fast Bayesian MLQMC

Require: $\mathcal{L}_{\text{feasible}} \subseteq \{1, \dots, L\}$ with $\widetilde{L} := |\mathcal{L}_{\text{feasible}}| > 0$ elements ▷ *levels to consider*

Require: $C_\ell, \quad \boldsymbol{\theta}_\ell, \quad n_\ell$ ▷ *costs, GP hyperparameters, current samples*

Set unique $l_1, \dots, l_{\widetilde{L}} \in \mathcal{L}_{\text{feasible}}$ so that $n_{l_1} C_{l_1} \geqslant \cdots \geqslant n_{l_{\widetilde{L}}} C_{l_{\widetilde{L}}}$
▷ *order feasible levels by non-increasing cost-of-doubling*

$\ell \leftarrow l_1$ ▷ *initialize the selected level*

for $k = 2, \dots, \widetilde{L}$ **do**

$\ell' \leftarrow \ell_k$ ▷ *will compare levels ℓ and ℓ' with $n_\ell C_\ell \geqslant n_{\ell'} C_{\ell'}$*

$\widehat{n}_{\ell'} \leftarrow n_\ell C_\ell / C_{\ell'} + n_{\ell'}$ ▷ *gives equal costs for increasing sample sizes*

$p \leftarrow \lfloor \log_2(\widehat{n}_{\ell'}) \rfloor$ ▷ *implies $2^p \leqslant \widehat{n}_{\ell'} < 2^{p+1}$*

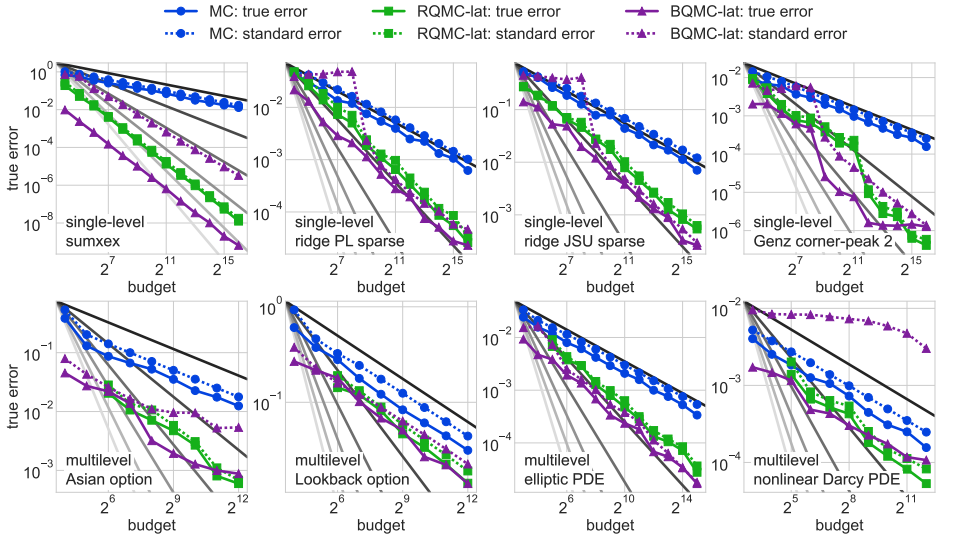
$\widehat{V}_{\ell',\widehat{n}_{\ell'}} \leftarrow (\widehat{V}^{p+1}_{\ell',2^p} / \widehat{V}^{p}_{\ell',2^{p+1}})^{\widehat{n}_{\ell'}^{\log_2\left(\widehat{V}_{\ell',2^{p+1}} / \widehat{V}_{\ell',2^p}\right)}}$ ▷ *log-log interpolation*

if $\widehat{V}_{\ell',n_{\ell'}} - \widehat{V}_{\ell',\widehat{n}_{\ell'}} \geqslant \widehat{V}_{\ell,n_\ell} - \widehat{V}_{\ell,2n_\ell}$ **then** ▷ *a greater projected variance decrease*

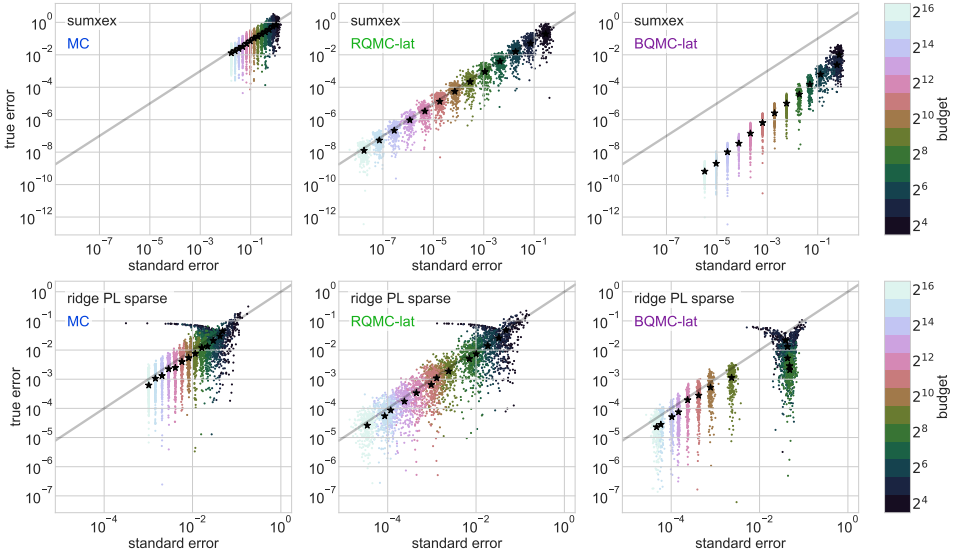
$\ell \leftarrow \ell'$ ▷ *update selected level*

return ℓ ▷ *the selected level*

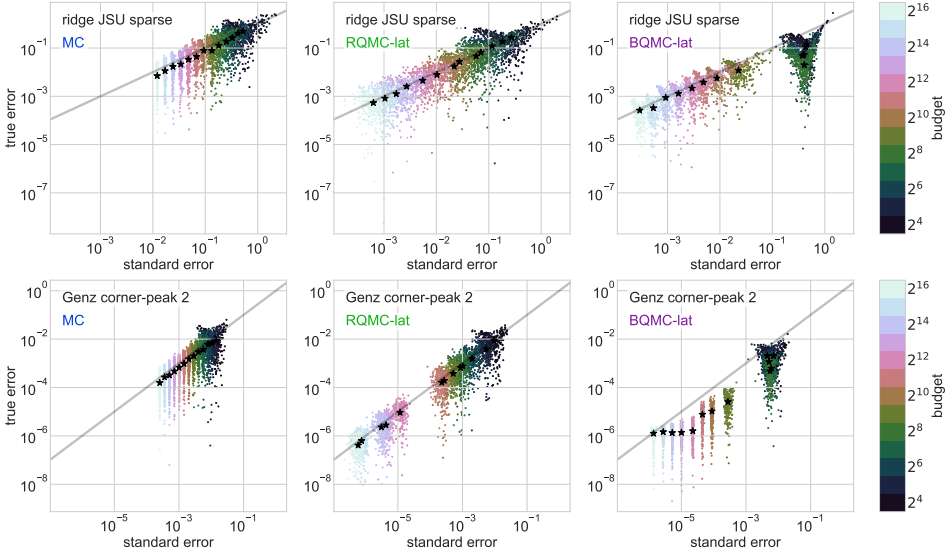
Convergence with Increasing Budgets: LD Lattices



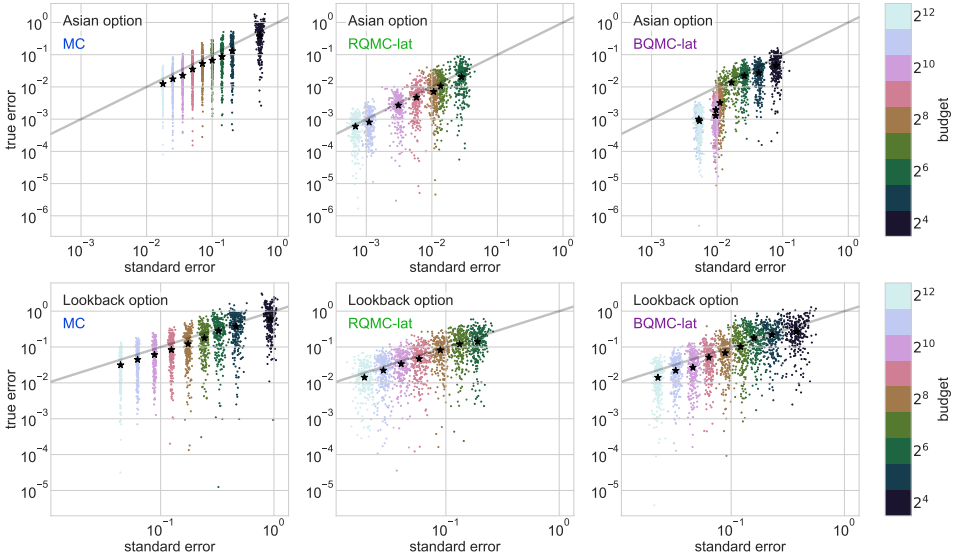
Single-Level Error Estimate Accuracy: LD Lattices



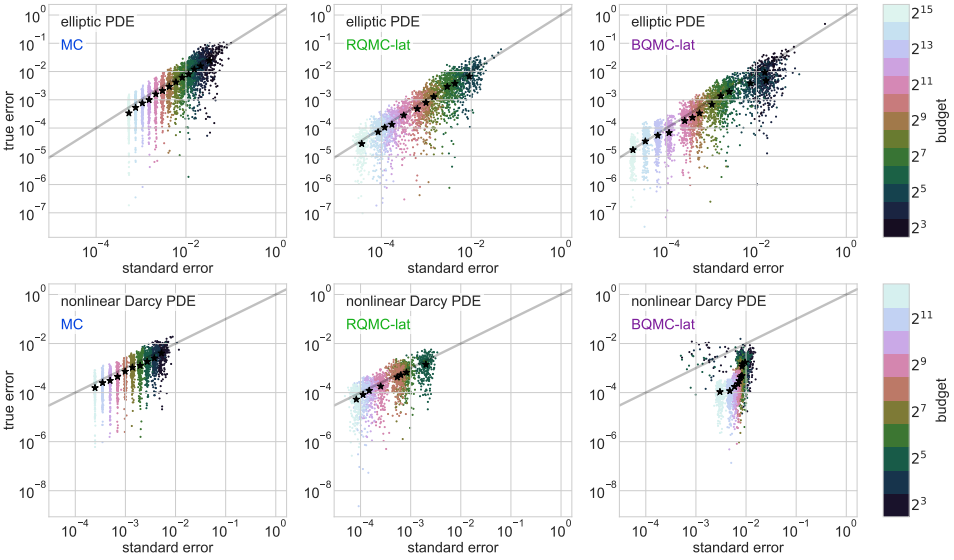
Single-Level Error Estimate Accuracy: LD Lattices



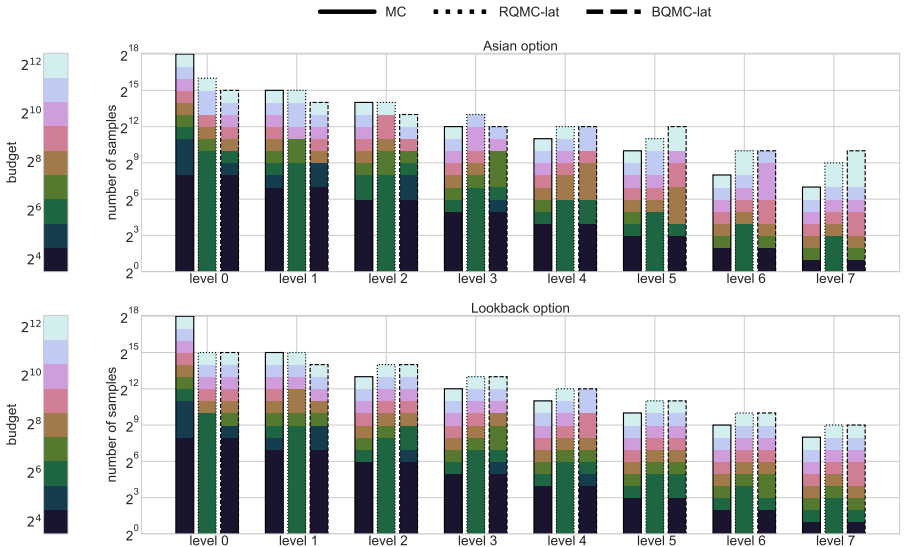
Multilevel Error Estimate Accuracy: LD Lattices



Multilevel Error Estimate Accuracy: LD Lattices



Multilevel Sample Allocation: LD Lattices



Multilevel Sample Allocation: LD Lattices