PAUL SCHERRER INSTITUT
PSI

ETH zürich

swissnuclear

Arnau Albà, R. Boiger, D. Rochman, A. Adelmann :: Paul Scherrer Institut, Switzerland

Lasso Monte Carlo, a Novel Method for High Dimensional UQ

SIAM CSE 23, 27th February 2023

Contact: arnau.albajacas@psi.ch

Pre-print available:

arXiv > stat > arXiv:2210.03634

Statistics > Computation

[Submitted on 7 Oct 2022]

Lasso Monte Carlo, a Novel Method for High Dimensional Uncertainty Quantification

Arnau Albà, Romana Boiger, Dimitri Rochman, Andreas Adelmann

Currently under review for SIAM UQ journal.

See paper for full proofs, details of algorithm, and citations.

Definition of Uncertainty Quantification (UQ)

Let $f \in L^2(\mathbb{R}^d)$ be a **computationally expensive** model with

$$f: \begin{array}{ccc} \mathbb{R}^d & \rightarrow & \mathbb{R} \\ \boldsymbol{x} & \mapsto & f(\boldsymbol{x}) \, . \end{array}$$

Let $\boldsymbol{x} = (x_1, x_2, ..., x_d)$ be an input with uncertainty $\Delta \boldsymbol{x}$.

Uncertainty in $f(\boldsymbol{x})$?

Definition of Uncertainty Quantification (UQ)

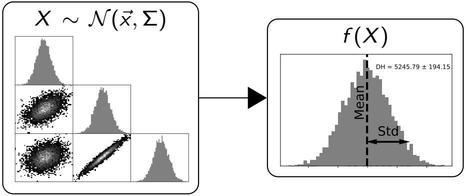
Let $f \in L^2(\mathbb{R}^d)$ be a **computationally expensive** model with

$$f: \begin{array}{ccc} \mathbb{R}^d & \to & \mathbb{R} \\ \boldsymbol{x} & \mapsto & f(\boldsymbol{x})\,. \end{array}$$

Let $\boldsymbol{x} = (x_1, x_2, ..., x_d)$ be an input with uncertainty $\Delta \boldsymbol{x}$.

Uncertainty in $f(\boldsymbol{x})$?

Model input as random variable $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{x}, \Sigma)$, with $\Sigma \in \mathbb{R}^{d \times d}$:

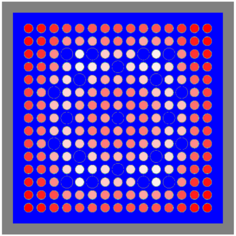


Goal: estimate **mean** and **variance** of output, with small computational effort:

$$f(\boldsymbol{x}) = \mu \pm \sigma\,.$$

Motivation for **high-dimensional** UQ: SNF Characterisation

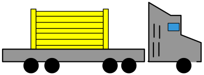
Nuclear codes are used to simulate and characterise spent nuclear fuel:



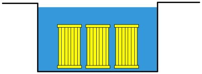
2D profile of spent nuclear fuel.
Red indicates U^{235} concentration.

$$f : \underbrace{\text{Nuclear Data}}_{\substack{\sim\mathcal{N}(x,\Sigma),\\ \text{with } x\in\mathbb{R}^{10^4}}} \rightarrow \begin{cases} \text{Decay Heat} \\ \text{Isotopic Content} \\ \text{Criticality} \\ \text{etc...} \end{cases}$$

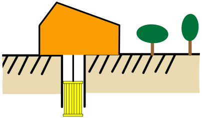
Accurate UQ reduces risks and costs during:



Transport



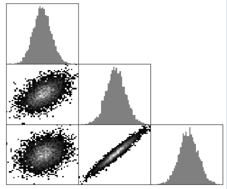
Storage



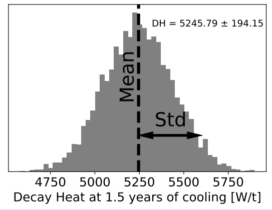
Disposal

Simple Monte Carlo UQ

1. Sample inputs $\vec{x}_1, ..., \vec{x}_N \sim \mathcal{N}(\vec{x}, \Sigma)$



→ Run $f(\vec{x}_i)$ N times → Compute sample mean and variance

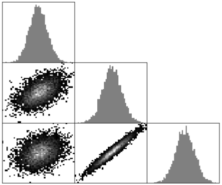


2. Compute sample mean and variance

$$\mu_N = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}_i), \quad \sigma_N^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(f(\boldsymbol{x}_i) - \sum_{j=1}^{N}\frac{f(\boldsymbol{x}_j)}{N}\right)^2 .$$

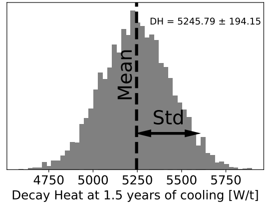
Simple Monte Carlo UQ

1. Sample inputs $\vec{x}_1, \ldots, \vec{x}_N \sim \mathcal{N}(\vec{x}, \Sigma)$



Run $f(\vec{x}_i)$ N times

Compute sample mean and variance



2. Compute sample mean and variance

$$\mu_N = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}_i)\,, \quad \sigma_N^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(f(\boldsymbol{x}_i) - \sum_{j=1}^{N}\frac{f(\boldsymbol{x}_j)}{N}\right)^2 .$$

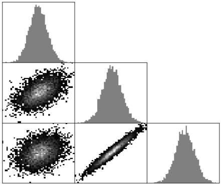
Simple MC is unbiased, but slow:

$$\lim_{N\to\infty} \mu_N = \mathbb{E}[f]\,, \quad \text{since MSE}\left(\mu_N - \mathbb{E}[f]\right) = \frac{\mathrm{Var}[f]}{N}\,,$$

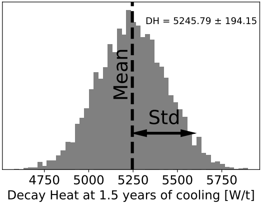
$$\lim_{N\to\infty} \sigma_N^2 = \mathrm{Var}[f]\,, \quad \text{since MSE}\left(\sigma_N^2 - \mathrm{Var}[f]\right) = \frac{1}{N}\left(m_4[f] - \frac{N-3}{N-1}\mathrm{Var}^2[f]\right) .$$

Simple Monte Carlo UQ

1. Sample inputs $\vec{x}_1, \ldots, \vec{x}_N \sim \mathcal{N}(\vec{x}, \Sigma)$



→ Run $f(\vec{x}_i)$ N times → Compute sample mean and variance



2. Compute sample mean and variance

$$\mu_N = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}_i)\,, \quad \sigma_N^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(f(\boldsymbol{x}_i) - \sum_{j=1}^{N}\frac{f(\boldsymbol{x}_j)}{N}\right)^2 .$$

Simple MC is unbiased, but slow:

$$\lim_{N\to\infty} \mu_N = \mathbb{E}[f]\,, \quad \text{since MSE}\left(\mu_N - \mathbb{E}[f]\right) = \frac{\mathrm{Var}[f]}{N}\,,$$

$$\lim_{N\to\infty} \sigma_N^2 = \mathrm{Var}[f]\,, \quad \text{since MSE}\left(\sigma_N^2 - \mathrm{Var}[f]\right) = \frac{1}{N}\left(m_4[f] - \frac{N-3}{N-1}\mathrm{Var}^2[f]\right) .$$

E.g. for SNF, $\sim$ 1000 simulations of 3 hours each are required for a 1% error!

UQ with Surrogate Models

A more modern approach: Surrogate models (e.g. PCE [Frey and Adelmann, 2021], NNs [Solans et al., 2021]):

1. Gather a training set $\boldsymbol{x}_1, f(\boldsymbol{x}_1)$, $\boldsymbol{x}_2, f(\boldsymbol{x}_2)$, ..., $\boldsymbol{x}_{N_{tr}}, f(\boldsymbol{x}_{N_{tr}})$.
2. Train a surrogate model $\widetilde{f} \sim f$, that is **fast to evaluate**.
3. Run surrogate M times to obtain samples $\widetilde{f}(\boldsymbol{z}_1), \widetilde{f}(\boldsymbol{z}_2), ..., \widetilde{f}(\boldsymbol{z}_M)$, with $Z \sim \mathcal{N}(\boldsymbol{x}, \Sigma)$.
4. Compute sample mean $\widetilde{\mu}_M$ and variance $\widetilde{\sigma}^2_M$.

UQ with Surrogate Models

A more modern approach: Surrogate models (e.g. PCE [Frey and Adelmann, 2021], NNs [Solans et al., 2021]):

1. Gather a training set $\boldsymbol{x}_1, f(\boldsymbol{x}_1)$, $\boldsymbol{x}_2, f(\boldsymbol{x}_2)$, ..., $\boldsymbol{x}_{N_{tr}}, f(\boldsymbol{x}_{N_{tr}})$.
2. Train a surrogate model $\widetilde{f} \sim f$, that is **fast to evaluate**.
3. Run surrogate M times to obtain samples $\widetilde{f}(\boldsymbol{z}_1), \widetilde{f}(\boldsymbol{z}_2), \dots, \widetilde{f}(\boldsymbol{z}_M)$, with $Z \sim \mathcal{N}(\boldsymbol{x}, \Sigma)$.
4. Compute sample mean $\widetilde{\mu}_M$ and variance $\widetilde{\sigma}^2_M$.

- Fast convergence, since M can be large
- Training $\widetilde{f}$ requires a big training set, at least $N_{tr} > d$ (generally much more, *curse of dimensionality*) (e.g. nuclear data has $d > 10^4$).
- Estimates are biased since

$$\text{MSE}\left(\widetilde{\mu}_M - \mathbb{E}[f]\right) = \mathbb{E}^2\left[\widetilde{f} - f\right] + \frac{\mathrm{Var}\left[\widetilde{f}\right]}{M},$$

$$\text{MSE}\left(\widetilde{\sigma}^2_M - \mathrm{Var}[f]\right) = \left(\mathrm{Var}[f] - \mathrm{Var}[\widetilde{f}]\right)^2 + \frac{1}{M}\left(m_4[\widetilde{f}] - \frac{M-3}{M-1}\mathrm{Var}^2[\widetilde{f}]\right).$$

Lasso Monte Carlo

Lasso Monte Carlo (LMC) is a new technique that combines two existing methods:

- "Two-level estimators", based on Multilevel Monte Carlo (MLMC) [Giles, 2008]
- Lasso regression [Tibshirani, 1996]

Two-level MC

Let

- f be the true, expensive model, that we evaluate N times: $f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_N)$.
- $\widetilde{f}$ a cheap, biased, surrogate model, that we evaluate $N + M$ times, with $M \gg N$: $\widetilde{f}(\boldsymbol{x}_1), \widetilde{f}(\boldsymbol{x}_2), ..., \widetilde{f}(\boldsymbol{x}_N)$, and $\widetilde{f}(\boldsymbol{z}_1), \widetilde{f}(\boldsymbol{z}_2), ..., \widetilde{f}(\boldsymbol{z}_M)$.

Two-level MC

Let

- f be the true, expensive model, that we evaluate N times: $f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_N)$.
- $\widetilde{f}$ a cheap, biased, surrogate model, that we evaluate $N+M$ times, with $M \gg N$: $\widetilde{f}(\boldsymbol{x}_1), \widetilde{f}(\boldsymbol{x}_2), ..., \widetilde{f}(\boldsymbol{x}_N)$, and $\widetilde{f}(\boldsymbol{z}_1), \widetilde{f}(\boldsymbol{z}_2), ..., \widetilde{f}(\boldsymbol{z}_M)$.

Then the 2-level estimators estimators [Krumscheid et al., 2020] are

$$\mu_{N,M} = \frac{1}{M}\sum_{i=1}^{M}\widetilde{f}(\boldsymbol{z}_i) + \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}_i) - \widetilde{f}(\boldsymbol{x}_i) = \widetilde{\mu}_M + \mu_N - \widetilde{\mu}_N \,,$$

$$\sigma^2_{N,M} = \widetilde{\sigma}^2_M + \sigma^2_N - \widetilde{\sigma}^2_N \,.$$

Two-level MC

Let

- f be the true, expensive model, that we evaluate N times: $f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_N)$.
- $\widetilde{f}$ a cheap, biased, surrogate model, that we evaluate $N+M$ times, with $M \gg N$: $\widetilde{f}(\boldsymbol{x}_1), \widetilde{f}(\boldsymbol{x}_2), ..., \widetilde{f}(\boldsymbol{x}_N)$, and $\widetilde{f}(\boldsymbol{z}_1), \widetilde{f}(\boldsymbol{z}_2), ..., \widetilde{f}(\boldsymbol{z}_M)$.

Then the 2-level estimators estimators [Krumscheid et al., 2020] are

$$\mu_{N,M} = \frac{1}{M}\sum_{i=1}^{M}\widetilde{f}(\boldsymbol{z}_i) + \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}_i) - \widetilde{f}(\boldsymbol{x}_i) = \widetilde{\mu}_M + \mu_N - \widetilde{\mu}_N \,,$$

$$\sigma^2_{N,M} = \widetilde{\sigma}^2_M + \sigma^2_N - \widetilde{\sigma}^2_N \,.$$

- Estimators are unbiased $\lim_{\substack{N\to\infty \\ M\to\infty}} \mu_{N,M} = \mathbb{E}[f]\,, \quad \lim_{\substack{N\to\infty \\ M\to\infty}} \sigma^2_{N,M} = \mathrm{Var}[f]\,.$
- More accurate (i.e. faster) than simple MC μ_N, σ^2_N, if and only if following conditions are satisfied

$$\mathrm{Var}[f-\widetilde{f}] \leq \mathrm{Var}[f]\,, \tag{1}$$

$$m_{2,2}\left[f+\widetilde{f}, f-\widetilde{f}\right] + \tfrac{1}{N-1}\mathrm{Var}[f+\widetilde{f}]\,\mathrm{Var}[f-\widetilde{f}] - \tfrac{N-2}{N-1}\left(\mathrm{Var}[f]-\mathrm{Var}[\widetilde{f}]\right)^2 \leq m_4[f] - \tfrac{N-3}{N-1}\mathrm{Var}^2[f]\,. \tag{2}$$

- However, bottleneck is still generating the training set N_{tr}.

Choosing Surrogate Model

How to choose a surrogate model that satisfies convergence conditions, and can be trained with a small training set $N_{tr} \ll d$?

Choosing Surrogate Model

How to choose a surrogate model that satisfies convergence conditions, and can be trained with a small training set $N_{tr} \ll d$?

Use a regularised model, e.g. Lasso (sparse linear model):

$$\widetilde{f}(\boldsymbol{x}) = \boldsymbol{\beta} \cdot \boldsymbol{x}, \quad \text{with } \boldsymbol{\beta} \text{ sparse,}$$

by minimising loss function

$$\mathcal{L}(\boldsymbol{\beta}) = \underbrace{\frac{1}{2}\sum_{i=1}^{N_{tr}} \left(f(\boldsymbol{x}_i) - \boldsymbol{\beta} \cdot \boldsymbol{x}_i\right)^2}_{\text{OLS loss}} + \underbrace{\lambda ||\boldsymbol{\beta}||_1}_{\text{Regularisation term}},$$

with $\lambda > 0$ a chosen regularisation constant.

Choosing Surrogate Model

How to choose a surrogate model that satisfies convergence conditions, and can be trained with a small training set $N_{tr} \ll d$?

Use a regularised model, e.g. Lasso (sparse linear model):

$$\widetilde{f}(\boldsymbol{x}) = \boldsymbol{\beta} \cdot \boldsymbol{x}, \quad \text{with } \boldsymbol{\beta} \text{ sparse,}$$

by minimising loss function

$$\mathcal{L}(\boldsymbol{\beta}) = \underbrace{\frac{1}{2} \sum_{i=1}^{N_{tr}} \left(f(\boldsymbol{x}_i) - \boldsymbol{\beta} \cdot \boldsymbol{x}_i\right)^2}_{\text{OLS loss}} + \underbrace{\lambda ||\boldsymbol{\beta}||_1}_{\text{Regularisation term}},$$

with $\lambda > 0$ a chosen regularisation constant.

- Lasso can be trained for small training sets, without overfitting.
- Satisfies the convergence conditions (1, 2), under correct choice of λ and some conditions on f (see paper).

Choosing Surrogate Model

How to choose a surrogate model that satisfies convergence conditions, and can be trained with a small training set $N_{tr} \ll d$?

Use a regularised model, e.g. Lasso (sparse linear model):

$$\widetilde{f}(\boldsymbol{x}) = \boldsymbol{\beta} \cdot \boldsymbol{x}, \quad \text{with } \boldsymbol{\beta} \text{ sparse},$$

by minimising loss function

$$\mathcal{L}(\boldsymbol{\beta}) = \underbrace{\frac{1}{2} \sum_{i=1}^{N_{tr}} \left(f(\boldsymbol{x}_i) - \boldsymbol{\beta} \cdot \boldsymbol{x}_i\right)^2}_{\text{OLS loss}} + \underbrace{\lambda ||\boldsymbol{\beta}||_1}_{\text{Regularisation term}},$$

with $\lambda > 0$ a chosen regularisation constant.

- Lasso can be trained for small training sets, without overfitting.
- Satisfies the convergence conditions (1, 2), under correct choice of λ and some conditions on f (see paper).

I.e. two-level estimators with Lasso
will converge faster or equally than simple MC.

Lasso Monte Carlo

Two-level MC + Lasso:

1. Gather small set $\boldsymbol{x}_1, f(\boldsymbol{x}_1)$, $\boldsymbol{x}_2, f(\boldsymbol{x}_2)$, ..., $\boldsymbol{x}_{N_{tr}}, f(\boldsymbol{x}_{N_{tr}})$.
2. Train a Lasso model $\widetilde{f} \sim f$.
3. Evaluate $\widetilde{f}$ $N + M$ times, with $M \gg N$.
4. Evaluate f N times.
5. Compute 2-level estimators $\mu_{N,M}, \sigma^2_{N,M}$

Lasso Monte Carlo

Two-level MC + Lasso:

1. Gather small set $\boldsymbol{x}_1, f(\boldsymbol{x}_1)$, $\boldsymbol{x}_2, f(\boldsymbol{x}_2)$, ..., $\boldsymbol{x}_{N_{tr}}, f(\boldsymbol{x}_{N_{tr}})$.
2. Train a Lasso model $\widetilde{f} \sim f$.
3. Evaluate $\widetilde{f}$ $N + M$ times, with $M \gg N$.
4. Evaluate f N times.
5. Compute 2-level estimators $\mu_{N,M}, \sigma^2_{N,M}$

Reuse the same set for training!

Lasso Monte Carlo

LMC algorithm:

1. Evaluate f N times: $\boldsymbol{x}_1, f(\boldsymbol{x}_1)$, $\boldsymbol{x}_2, f(\boldsymbol{x}_2)$, ..., $\boldsymbol{x}_N, f(\boldsymbol{x}_N)$.
2. Train a Lasso model $\widetilde{f} \sim f$.
3. Evaluate $\widetilde{f}$ $N + M$ times, with $M \gg N$.
4. Compute 2-level estimators $\mu_{N,M}, \sigma^2_{N,M}$

- Unbiased.
- Faster (or equal) convergence than simple MC for a given N.
- Surrogate model trained *for free* (no extra simulations required).
- Note: see full algorithm in paper.

LMC Code Example

```
>>> N = 150; M = 6000
>>> Xs = get_inputs(N)
>>> ys = [my_simulation(x) for x in Xs]
>>> Zs = get_inputs(M)
```

LMC Code Example

```
>>> N = 150; M = 6000
>>> Xs = get_inputs(N)
>>> ys = [my_simulation(x) for x in Xs]
>>> Zs = get_inputs(M)
```

```
>>> import LassoMonteCarlo as LMC
>>> lmc = LMC()
>>> lmc.regressor = Lasso(lambda = 0.02)
>>> lmc.get_estimate(Xtrain = Xs, ytrain = ys, Xtest = Zs)
```

LMC Code Example

```
>>> N = 150; M = 6000
>>> Xs = get_inputs(N)
>>> ys = [my_simulation(x) for x in Xs]
>>> Zs = get_inputs(M)
```

```
>>> import LassoMonteCarlo as LMC
>>> lmc = LMC()
>>> lmc.regressor = Lasso(lambda = 0.02)
>>> lmc.get_estimate(Xtrain = Xs, ytrain = ys, Xtest = Zs)
```

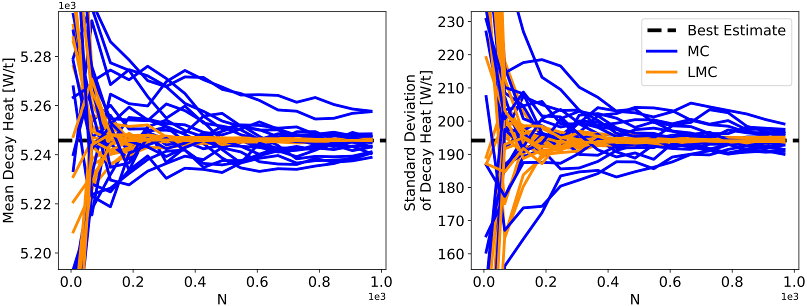
Output:

```
Ntr = 150 labelled samples, Ntest = 6000 unlabelled samples
MC estimates: 5234.47 +- 174.65
LMC estimates: 5246.75 +- 192.67
```

SNF Characterisation

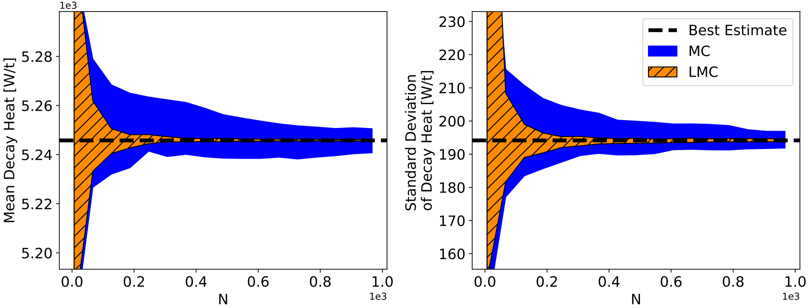
$$f : \mathbb{R}^{15\,557} \rightarrow \mathbb{R}$$
$$\text{Nuclear Data} \mapsto \text{Decay Heat}$$

Plots show increasing N, and fixed $M = 6000$.



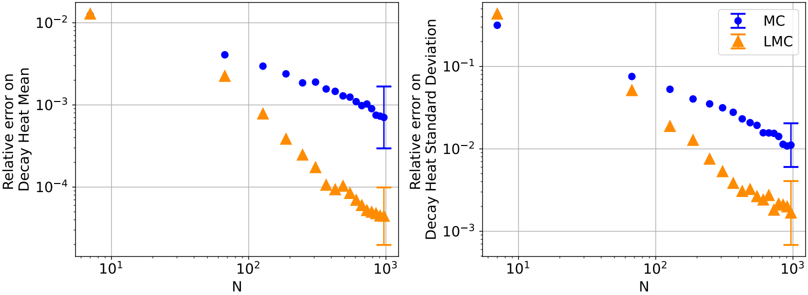
SNF Characterisation

$$f : \mathbb{R}^{15\,557} \rightarrow \mathbb{R}$$
$$\text{Nuclear Data} \mapsto \text{Decay Heat}$$



SNF Characterisation

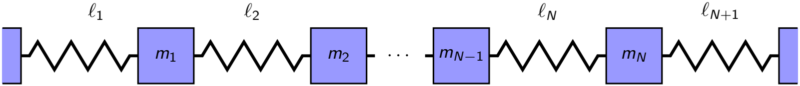
$$f : \mathbb{R}^{15\,557} \rightarrow \mathbb{R}$$
$$\text{Nuclear Data} \mapsto \text{Decay Heat}$$



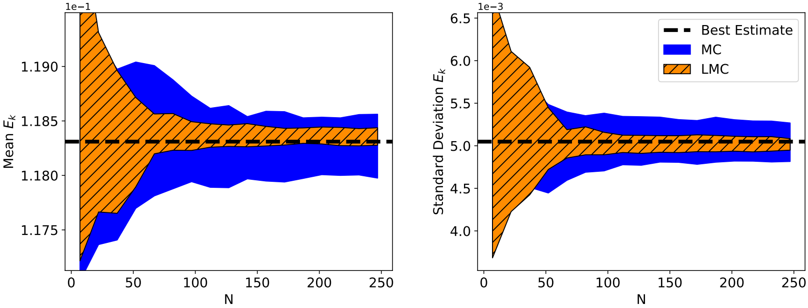
To obtain a 1% error in estimations, simple MC requires $N = 1000$ expensive simulations f, while LMC requires $N = 200$. I.e. **5 times speedup thanks to LMC**.

FPUT Oscillator

Chain of nonlinear oscillators

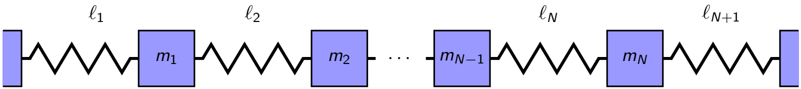


Consider an uncertainty in the spring constants $k_1, k_2, ..., k_N$, and nonlinear term α. What is the uncertainty in E_{kin}?

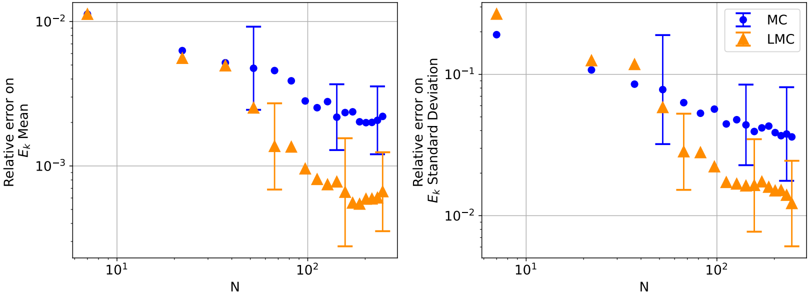


FPUT Oscillator

Chain of nonlinear oscillators



Consider an uncertainty in the spring constants $k_1, k_2, \ldots, k_N$, and nonlinear term α. What is the uncertainty in E_{kin}?



Conclusions and Further Work

- LMC converges faster or equally than simple MC.
- Up to **5 times faster than simple MC** for nuclear simulations!
- Given a set of N simulations, LMC can immediately be applied without extra work.

- The speedup is not constant, it's very dependent on f.
- Theoretical guarantees of faster convergence conditioned on choice of λ (chosen empirically so far).

Thank you for your attention.

Questions?

References

M. Frey and A. Adelmann. Global sensitivity analysis on numerical solver parameters of Particle-In-Cell models in particle accelerator systems. *Computer Physics Communications*, 258: 107577, 2021.

V. Solans, D. Rochman, Ch. Brazell, A. Vasiliev, H. Ferroukhi, and A. Pautz. Optimisation of used nuclear fuel canister loading using a neural network and genetic algorithm. *Neural Computing and Applications*, 33(23):16627–16639, December 2021. ISSN 1433-3058. doi: 10.1007/s00521-021-06258-2. URL https://doi.org/10.1007/s00521-021-06258-2.

M. B. Giles. Multilevel Monte Carlo Path Simulation. *Operations Research*, 56(3):607–617, June 2008. ISSN 0030-364X. doi: 10.1287/opre.1070.0496. URL https://pubsonline.informs.org/doi/abs/10.1287/opre.1070.0496.

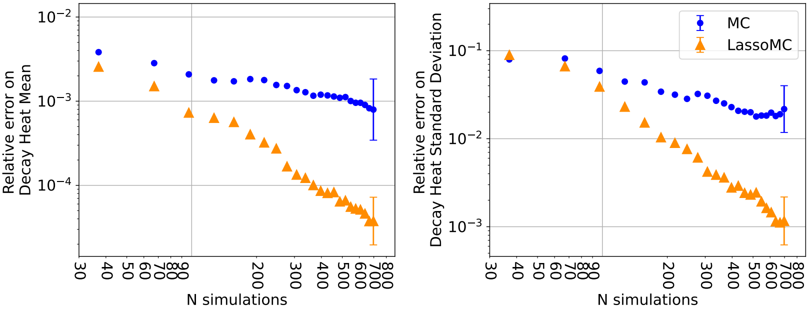
R. Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. ISSN 0035-9246. URL https://www.jstor.org/stable/2346178.

S. Krumscheid, F. Nobile, and M. Pisaroni. Quantifying uncertain system outputs via the multilevel Monte Carlo method — Part I: Central moment estimation. *Journal of Computational Physics*, 414, August 2020. ISSN 0021-9991.

Extra Slides

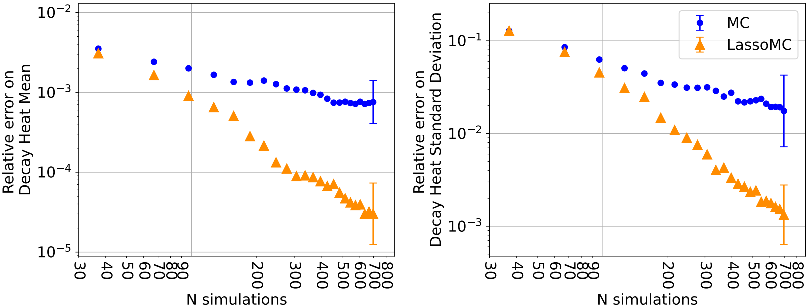
Results

Decay heat prediction at 30 years of cooling:



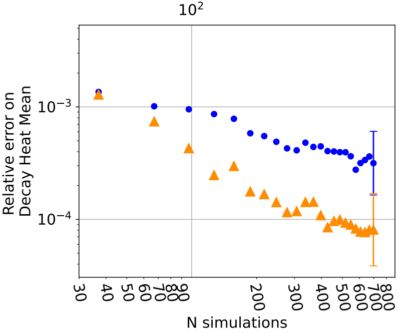
Results

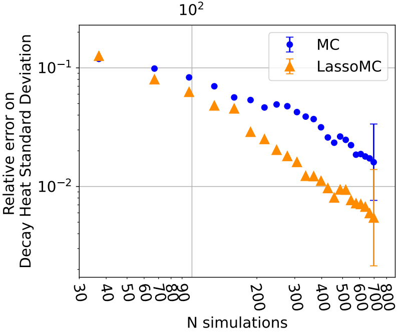
Decay heat prediction at 50 years of cooling:



Results

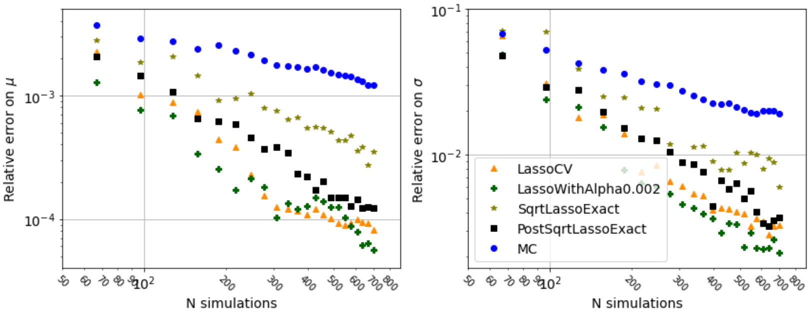
U235 concentration at discharge





Lasso versions

Other versions of Lasso regression



Full LMC Algorithm

Require: the probability distribution of the input of $f(\boldsymbol{x})$, the training sets $\{\boldsymbol{x}_1, ...\boldsymbol{x}_N\}$ and $\{\boldsymbol{z}_1, ...\boldsymbol{z}_M\}$
Ensure: $N \ll M$
1: Compute the labels $f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_N)$ from the training set.
2: Compute the simple MC estimates μ_N, σ_N^2 with the labelled training set, using the simple MC estimators.
3: Do an S-fold split on the training set to obtain S smaller training sets $T_1, T_2, ..., T_S$ of size $N\frac{S-1}{S}$ each, and S correction sets $C_1, C_2, ..., C_S$ of size $n := \frac{N}{S}$ each. Each training set T_i does not overlap with its corresponding correction set C_i.
4: **for** $\mathrm{s} = 1 \dots S$ **do**
5: Fit a Lasso model $\widetilde{f}_\mathrm{s}$ on training set T_s.
6: Use the surrogate model to compute the labels of the surrogate set $\widetilde{f}_\mathrm{s}(\boldsymbol{z}_1), \widetilde{f}_\mathrm{s}(\boldsymbol{z}_2), ..., \widetilde{f}_\mathrm{s}(\boldsymbol{z}_M)$, and the C_s correction set $\widetilde{f}_\mathrm{s}\left(\boldsymbol{x}_{n(\mathrm{s}-1)+1}\right), f_\mathrm{s}\left(\boldsymbol{x}_{n(\mathrm{s}-1)+2}\right), ..., \widetilde{f}_\mathrm{s}\left(\boldsymbol{x}_{n\mathrm{s}}\right)$.
7: Combine the n labels from the correction set and the M from the surrogate set to compute the two-level estimators $(\mu_{n,M})_\mathrm{s}$ and $(\sigma_{n,M}^2)_\mathrm{s}$.
8: **end for**
9: Compute the LMC mean and variance, by averaging out the estimations of each split

$$M_{N,M} = \frac{1}{S}\sum_{\mathrm{s}=1}^{S}\left(\mu_{n,M}\right)_\mathrm{s}, \quad \text{and} \quad \Sigma_{N,M}^2 = \frac{1}{S}\sum_{\mathrm{s}=1}^{S}\left(\sigma_{n,M}^2\right)_\mathrm{s}.$$

MLMC Explanation

MLMC combines models of different levels of fidelity.

Let X be a random variable, and $f_1, f_2, ..., f_L$ be models of increasing accuracy, and increasing computational cost. Then

$$\mathbb{E}[f_L(X)] = \mathbb{E}[f_1(X)] + \mathbb{E}[f_2(X) - f_1(X)] + \mathbb{E}[f_3(X) - f_2(X)] + ... + \mathbb{E}[f_{L-1}(X) - f_L(X)]$$

MLMC Explanation

MLMC combines models of different levels of fidelity.

Let X be a random variable, and $f_1, f_2, ..., f_L$ be models of increasing accuracy, and increasing computational cost. Then

$$\mathbb{E}[f_L(X)] = \mathbb{E}[f_1(X)] + \mathbb{E}[f_2(X) - f_1(X)] + \mathbb{E}[f_3(X) - f_2(X)] + ... + \mathbb{E}[f_{L-1}(X) - f_L(X)]$$

Terms computed with

$$\mathbb{E}[f_\ell(X) - f_{\ell-1}(X)] = \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} \{f_\ell(x_i) - f_{\ell-1}(x_i)\},$$

will converge as $\mathcal{O}\left(\frac{\mathrm{Var}[f_\ell - f_{\ell-1}]}{\sqrt{N_L}}\right)$.

MLMC Explanation

MLMC combines models of different levels of fidelity.

Let X be a random variable, and $f_1, f_2, ..., f_L$ be models of increasing accuracy, and increasing computational cost. Then

$$\mathbb{E}[f_L(X)] = \mathbb{E}[f_1(X)] + \mathbb{E}[f_2(X) - f_1(X)] + \mathbb{E}[f_3(X) - f_2(X)] + ... + \mathbb{E}[f_{L-1}(X) - f_L(X)]$$

Terms computed with

$$\mathbb{E}[f_\ell(X) - f_{\ell-1}(X)] = \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} \{f_\ell(x_i) - f_{\ell-1}(x_i)\},$$

will converge as $\mathcal{O}\left(\frac{\mathrm{Var}[f_\ell - f_{\ell-1}]}{\sqrt{N_L}}\right)$.

So if we have

$$\mathrm{Var}(f_1) > \mathrm{Var}(f_2 - f_1) > \mathrm{Var}(f_3 - f_2) > ... > \mathrm{Var}(f_L - f_{L-1}),$$

we require

$$N_1 > N_2 > ... > N_L.$$

Overall computational cost is reduced if N_ℓ are correctly chosen!

MLMC Explanation

MLMC combines models of different levels of fidelity.

Let X be a random variable, and $f_1, f_2, ..., f_L$ be models of increasing accuracy, and increasing computational cost. Then

$$\mathbb{E}[f_L(X)] = \mathbb{E}[f_1(X)] + \mathbb{E}[f_2(X) - f_1(X)] + \mathbb{E}[f_3(X) - f_2(X)] + ... + \mathbb{E}[f_{L-1}(X) - f_L(X)]$$

Terms computed with

$$\mathbb{E}[f_\ell(X) - f_{\ell-1}(X)] = \frac{1}{N_\ell}\sum_{i=1}^{N_\ell}\{f_\ell(x_i) - f_{\ell-1}(x_i)\},$$

will converge as $\mathcal{O}\left(\frac{\mathrm{Var}[f_\ell - f_{\ell-1}]}{\sqrt{N_L}}\right)$.
So if we have

$$\mathrm{Var}(f_1) > \mathrm{Var}(f_2 - f_1) > \mathrm{Var}(f_3 - f_2) > ... > \mathrm{Var}(f_L - f_{L-1}),$$

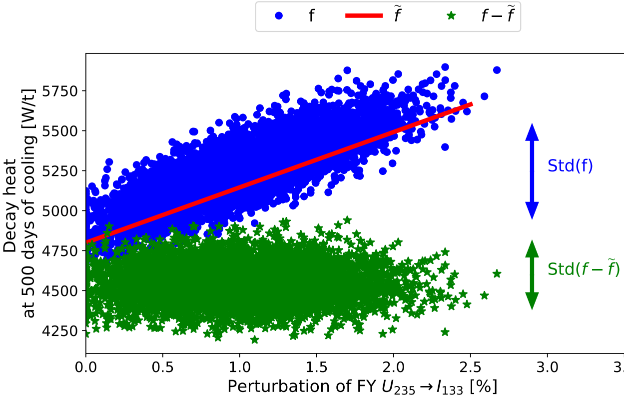
we require

$$N_1 > N_2 > ... > N_L.$$

Overall computational cost is reduced if N_ℓ are correctly chosen!
Thanks to more recent papers

Convergence condition 1:

Does Lasso $\widetilde{f}$ satisfy the convergence conditions (1, 2)?
Condition (1) is always satisfied! (as long as λ is chosen correctly)



I.e. the two-level estimator $\mu_{N,M}$ with Lasso, **is guaranteed to converge equally or faster than simple MC**.

Convergence condition 2:

Does Lasso $\widetilde{f}$ satisfy the convergence conditions (1, 2)?
Condition (2), is unfortunately not guaranteed.

Convergence condition 2:

Does Lasso $\widetilde{f}$ satisfy the convergence conditions (1, 2)?
Condition (2), is unfortunately not guaranteed.

However, if f is a *noisy linear function*

$$f(\boldsymbol{x}) = \boldsymbol{\alpha} \cdot \boldsymbol{x} + \mathcal{E}\,, \quad \text{with}\, \mathcal{E} \sim \mathcal{N}(0, \varepsilon)$$

then condition (2) is guaranteed! This is true to first order for any f:

$$f(\boldsymbol{x} + \delta\boldsymbol{x}) = f(\boldsymbol{x}_0) + \delta\boldsymbol{x} \cdot \boldsymbol{\nabla} f(\boldsymbol{x}_0) + \mathcal{O}\left(||\delta\boldsymbol{x}||^2\right).$$

Convergence condition 2:

Does Lasso $\widetilde{f}$ satisfy the convergence conditions (1, 2)?
Condition (2), is unfortunately not guaranteed.

However, if f is a *noisy linear function*

$$f(\boldsymbol{x}) = \boldsymbol{\alpha} \cdot \boldsymbol{x} + \mathcal{E}\,, \quad \text{with}\, \mathcal{E} \sim \mathcal{N}(0, \varepsilon)$$

then condition (2) is guaranteed! This is true to first order for any f:

$$f(\boldsymbol{x} + \delta\boldsymbol{x}) = f(\boldsymbol{x}_0) + \delta\boldsymbol{x} \cdot \boldsymbol{\nabla} f(\boldsymbol{x}_0) + \mathcal{O}\left(||\delta\boldsymbol{x}||^2\right).$$

I.e. the two-level estimator $\sigma^2_{N,M}$ with Lasso, will often converge faster than simple MC, and is guaranteed to do so under certain conditions on f.

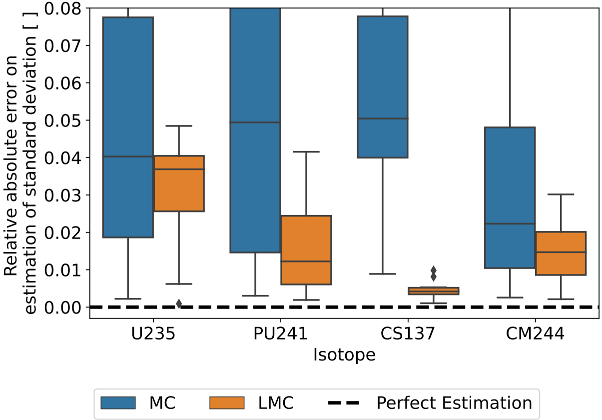
SNF Characterisation

Fixed $N = 150$ and $M = 6000$.

$$f : \mathbb{R}^{15\,557} \to \mathbb{R}$$

$$\text{Nuclear Data} \mapsto \text{Isotopic Content}$$

Predicting different quantities gives different improvements. But **LMC is always equal or better than simple MC**.

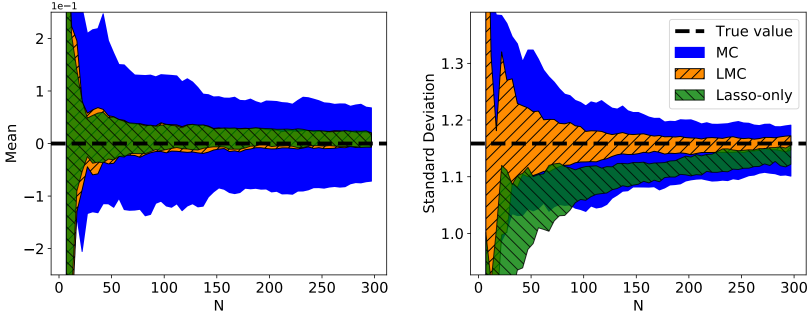


Linear Function

Let f be a linear function with a large input dimension $d = 400$:

$$\begin{cases} f(\boldsymbol{x}) = \boldsymbol{\alpha} \cdot \boldsymbol{x}, \\ \text{with } \boldsymbol{\alpha} = \left(1, \frac{1}{2}, \frac{1}{5}, \frac{1}{10}, \frac{1}{20}, \frac{1}{50}, \frac{1}{100}, \frac{1}{100}, \ldots, \frac{1}{100}\right), \end{cases}$$

with $\dim(\boldsymbol{\alpha}) = 400$ and with a normally distributed input $X \sim \mathcal{N}(0, I_d)$.

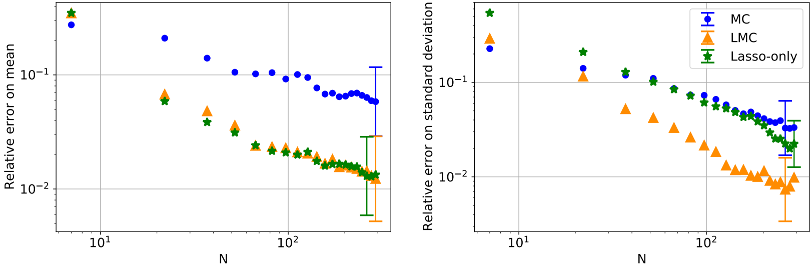


Linear Function

Let f be a linear function with a large input dimension $d = 400$:

$$\begin{cases} f(\boldsymbol{x}) = \boldsymbol{\alpha} \cdot \boldsymbol{x}, \\ \text{with } \boldsymbol{\alpha} = \left(1, \frac{1}{2}, \frac{1}{5}, \frac{1}{10}, \frac{1}{20}, \frac{1}{50}, \frac{1}{100}, \frac{1}{100}, \ldots, \frac{1}{100}\right), \end{cases}$$

with $\dim(\boldsymbol{\alpha}) = 400$ and with a normally distributed input $X \sim \mathcal{N}(0, I_d)$.

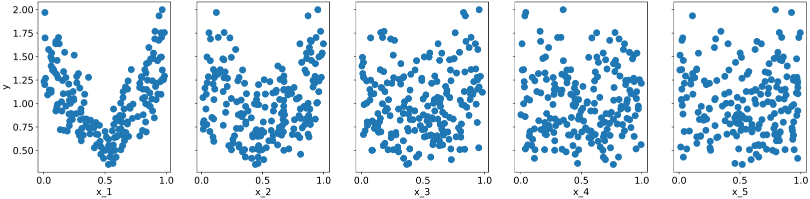


Sobol Function

$$\begin{cases} f(\boldsymbol{x}) = \prod_{i=1}^{d} \frac{|4x_i - 2| + c_i}{1 + c_i}, \\ \text{with } \boldsymbol{c} = (1, 2, 5, 10, 20, 50, 100, 200, 500, 500, ..., 500), \end{cases}$$

with $d = 400$, and $\boldsymbol{X} \sim U[0,1]^d$.
Function is symmetric around $x = 0.5$, so a Lasso fit model will be flat, **i.e. worst-case scenario, LMC will be equally accurate as simple MC**.

However we can instead fit a modified Lasso model $\widetilde{f}(\boldsymbol{x}) = \boldsymbol{\beta} \cdot \phi(\boldsymbol{x})$, with $\phi(\boldsymbol{x}) = |x - 0.5|$.

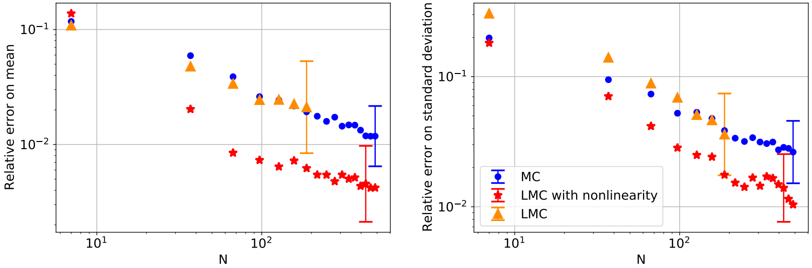


Sobol Function

$$\begin{cases} f(\boldsymbol{x}) = \prod_{i=1}^{d} \frac{|4x_i - 2| + c_i}{1 + c_i}, \\ \text{with } \boldsymbol{c} = (1, 2, 5, 10, 20, 50, 100, 200, 500, 500, \ldots, 500), \end{cases}$$

with $d = 400$, and $\boldsymbol{X} \sim U[0,1]^d$.
Function is symmetric around $x = 0.5$, so a Lasso fit model will be flat, **i.e. worst-case scenario, LMC will be equally accurate as simple MC**.

However we can instead fit a modified Lasso model $\widetilde{f}(\boldsymbol{x}) = \boldsymbol{\beta} \cdot \phi(\boldsymbol{x})$, with $\phi(\boldsymbol{x}) = |x - 0.5|$.



Any kind of surrogate could be used in LMC, **as long as it is strongly regularised**.

Comparison to PCE

Use the Sobol function, with input dimension $d = 8$ (higher dimensions are too slow to handle with Chaospy library).