PAUL SCHERRER INSTITUT

PSI

ETH zürich

swissnuclear

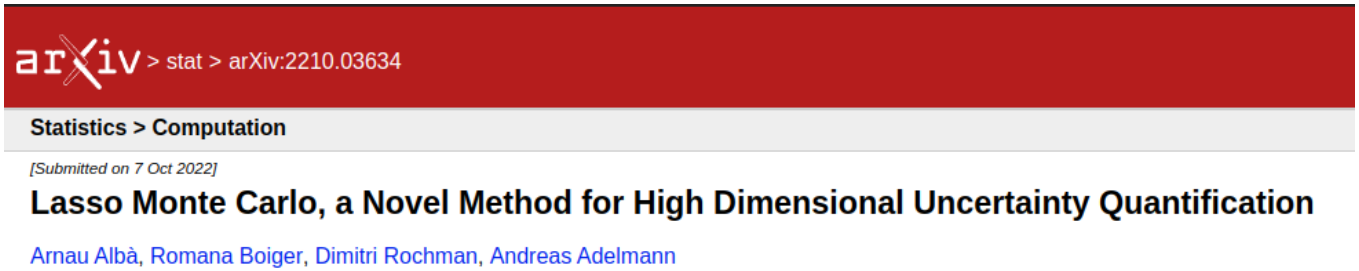
Arnau Albà, R. Boiger, D. Rochman, A. Adelmann :: AMAS Group, LSM

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

ML Lunch, 18th January 2023

Contact: arnau.albajacas@psi.ch

Pre-print available:



Currently under review for SIAM UQ journal.

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

Overview

Definition of Uncertainty Quantification (UQ)

Uncertainty Quantification (UQ) aims to calculate the effect of unknown or uncertain system parameters on the outcome of an experiment or computation.

Definition of Uncertainty Quantification (UQ)

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

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

Let $\boldsymbol{x} = (x_1, x_2, ..., x_d)$ be an input with uncertainty $\Delta\boldsymbol{x}$.
What is the uncertainty in $f(\boldsymbol{x})$?

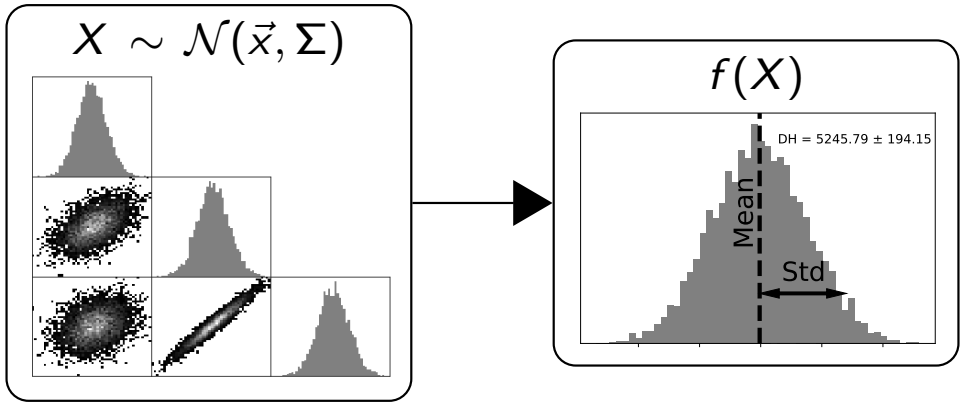
Definition of Uncertainty Quantification (UQ)

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

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

Let $\boldsymbol{x} = (x_1, x_2, ..., x_d)$ be an input with uncertainty $\Delta\boldsymbol{x}$.
What is the uncertainty in $f(\boldsymbol{x})$?

Common approach, model input as random variable $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{x}, \Sigma)$, with $\Sigma \in \mathbb{R}^{d\times d}$ the covariance matrix (uncertainties and correlations):



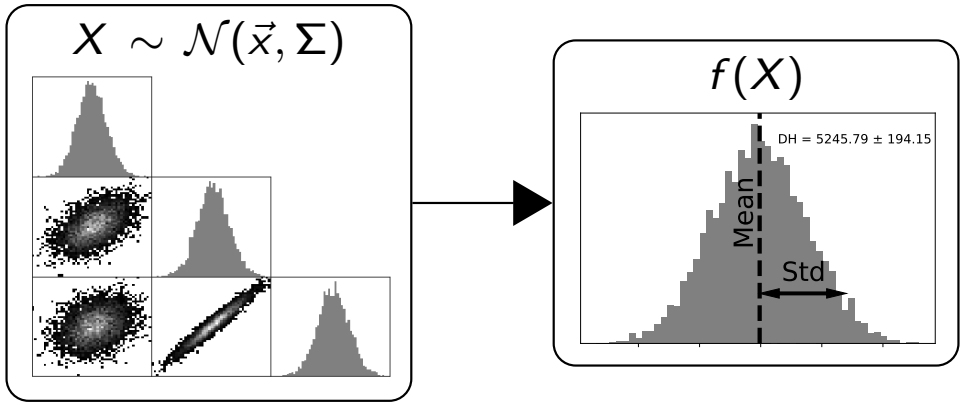
Definition of Uncertainty Quantification (UQ)

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

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

Let $\boldsymbol{x} = (x_1, x_2, ..., x_d)$ be an input with uncertainty $\Delta\boldsymbol{x}$.
What is the uncertainty in $f(\boldsymbol{x})$?

Common approach, model input as random variable $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{x}, \Sigma)$, with $\Sigma \in \mathbb{R}^{d\times d}$ the covariance matrix (uncertainties and correlations):



Concentrate on *Response Variability Methods*: estimate mean and variance of output

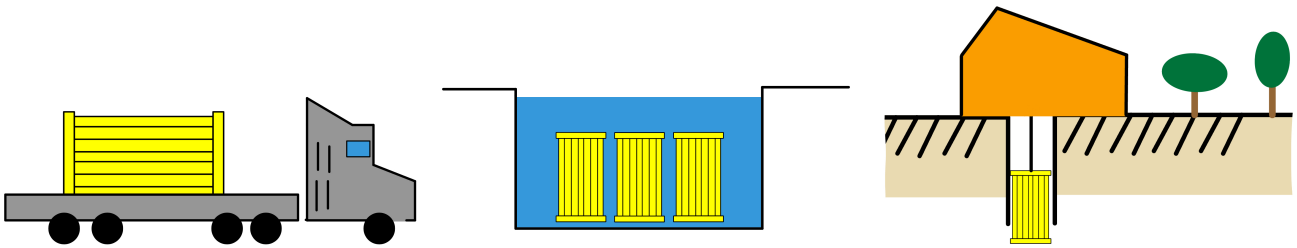
$$f(\boldsymbol{x}) = \mu \pm \sigma\,.$$

Motivation: SNF Characterisation

Nuclear burnup simulations are used to characterise spent nuclear fuel:

$$f : \underbrace{\begin{pmatrix} \texttt{Fresh fuel parameters} \\ \texttt{Irradiation history} \\ \texttt{Reactor parameters} \\ \texttt{Nuclear Data} \end{pmatrix}}_{\text{Input with known uncertainty}} \rightarrow \underbrace{\begin{pmatrix} \texttt{Decay Heat} \\ \texttt{Isotopic Content} \\ \texttt{Criticality} \end{pmatrix}}_{\text{Output with unknown uncertainty}}$$

Any uncertainty in outputs will increase the risks and costs of



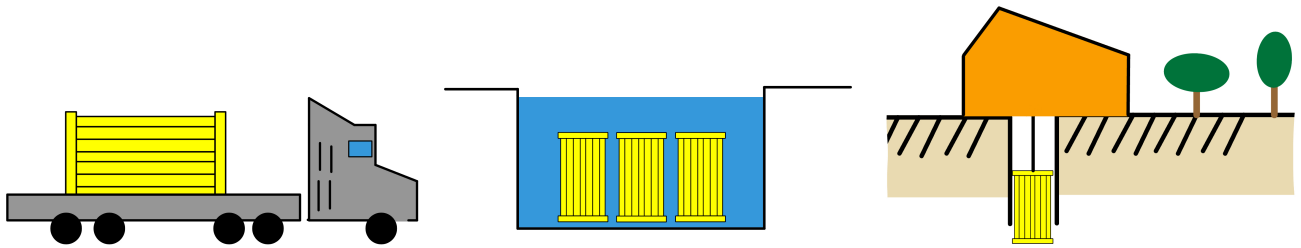
Transport Storage Disposal

Motivation: SNF Characterisation

Nuclear burnup simulations are used to characterise spent nuclear fuel:

$$f : \underbrace{\begin{pmatrix} \texttt{Fresh fuel parameters} \\ \texttt{Irradiation history} \\ \texttt{Reactor parameters} \\ \texttt{Nuclear Data} \end{pmatrix}}_{\text{Input with known uncertainty}} \rightarrow \underbrace{\begin{pmatrix} \texttt{Decay Heat} \\ \texttt{Isotopic Content} \\ \texttt{Criticality} \end{pmatrix}}_{\text{Output with unknown uncertainty}}$$

Any uncertainty in outputs will increase the risks and costs of



Transport Storage Disposal

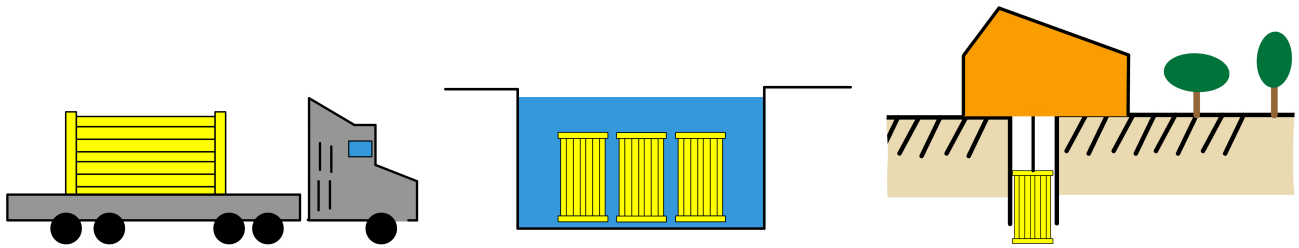
Accurate estimation of uncertainty saves money and reduces risks.

Motivation: SNF Characterisation

$$f : \mathbb{R}^{15000} \rightarrow \mathbb{R}$$

Nuclear Data $\rightarrow$ Decay Heat

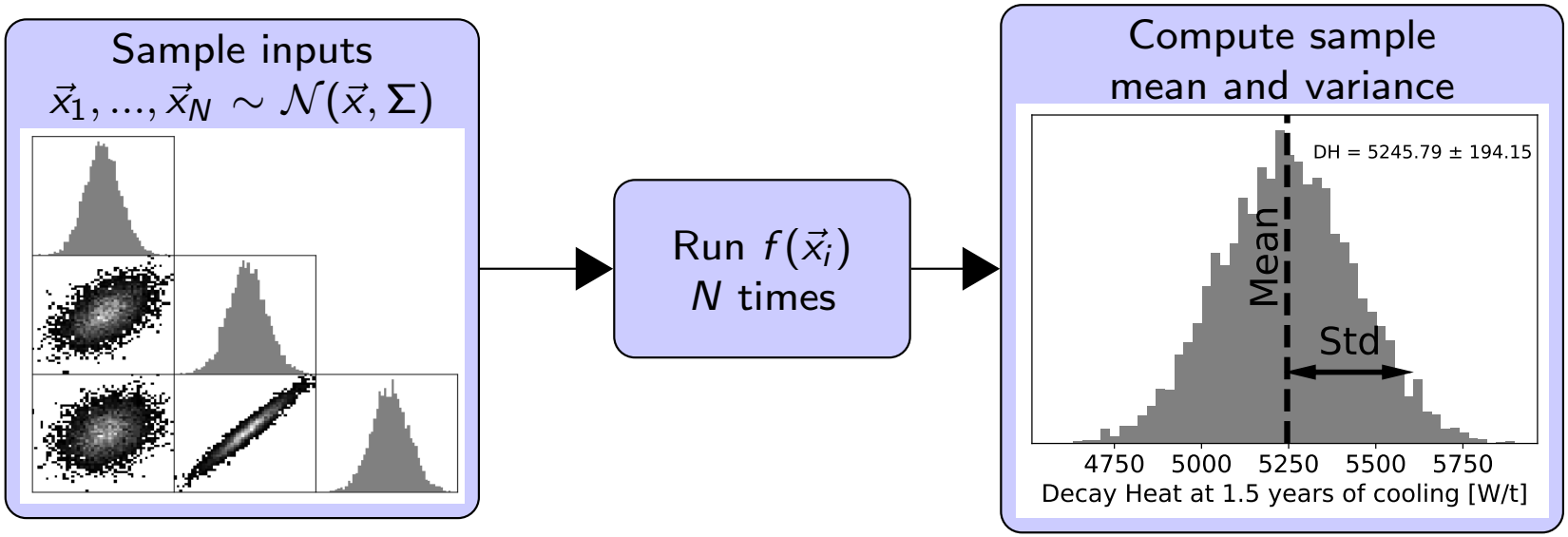
Any uncertainty in outputs will increase the risks and costs of



Transport Storage Disposal

Accurate estimation of uncertainty saves money and reduces risks.

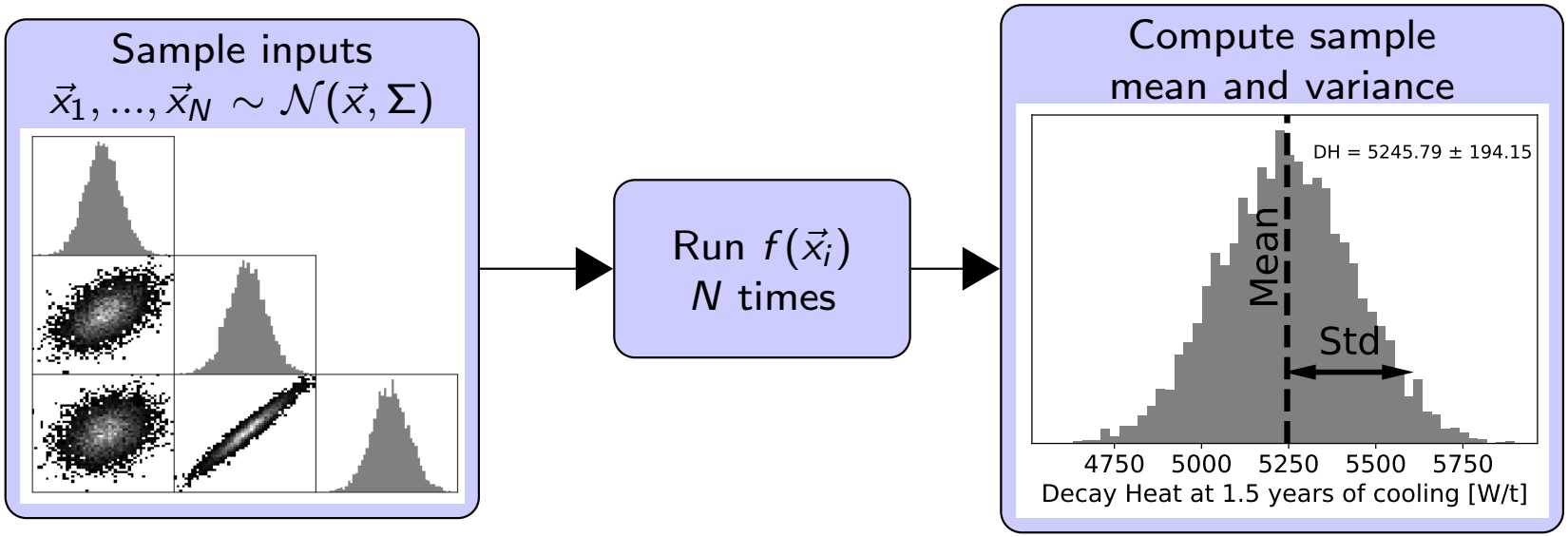
Simple Monte Carlo UQ

1. 

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. 

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 (error$= \sqrt{\text{MSE}} = \mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$):

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

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

Simple Monte Carlo UQ

Simple MC is the current approach used for nuclear data propagation:

- MC converges as $\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$, i.e. many simulations required!
- E.g. for SNF characterisation $N \sim 1000$, with each simulation lasting a few hours.
- Expecting > 12000 fuel assemblies in Switzerland.
- $\Rightarrow$ millions of CPU hours $\Rightarrow$ **MC UQ is too slow!**

UQ with Surrogate Models

A more modern approach: Surrogate models (e.g. PCE [1], NNs [2, 3]):

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 [1], NNs [2, 3]):

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 [1], NNs [2, 3]):

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 $\widetilde{\sigma}^2_M$.

- Converges very fast, since M can be large
- Training $\widetilde{f}$ requires a big training set, at least $N_{tr} > d$ (generally much more, see *curse of dimensionality*) (e.g. nuclear data has $d = 15\,000$).
- Estimates are biased since

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

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

In summary: simple MC and surrogate models are inadequate for high-dimensional UQ.

Lasso Monte Carlo

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

- Multilevel Monte Carlo (MLMC) [4, 5]
- Lasso regression [6]

MLMC

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

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

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

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 [7, 5], MLMC can be used for higher order moments.

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 estimators 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 estimators 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 than simple MC μ_N, σ^2_N, if and only if following conditions are satisfied

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

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

Two-level MC

Common usage of MLMC:

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. Evaluate $\widetilde{f}$ $N + M$ times to obtain $\widetilde{f}(\boldsymbol{x}_1), \widetilde{f}(\boldsymbol{x}_2), ..., \widetilde{f}(\boldsymbol{x}_N)$, and $\widetilde{f}(\boldsymbol{z}_1), \widetilde{f}(\boldsymbol{z}_2), ..., \widetilde{f}(\boldsymbol{z}_M)$.
4. Evaluate f N times, to obtain $f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_N)$.
5. Compute 2-level estimators $\mu_{N,M}, \sigma^2_{N,M}$.

Two-level MC

Common usage of MLMC:

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. Evaluate $\widetilde{f}$ $N + M$ times to obtain $\widetilde{f}(\boldsymbol{x}_1), \widetilde{f}(\boldsymbol{x}_2), ..., \widetilde{f}(\boldsymbol{x}_N)$, and $\widetilde{f}(\boldsymbol{z}_1), \widetilde{f}(\boldsymbol{z}_2), ..., \widetilde{f}(\boldsymbol{z}_M)$.
4. Evaluate f N times, to obtain $f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_N)$.
5. Compute 2-level estimators $\mu_{N,M}, \sigma^2_{N,M}$.

- Unbiased.
- More accurate than simple MC for a given N (if conditions (1, 2)).
- However, bottleneck is still generating the training set N_{tr} (especially in high-dimensional cases).

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 $N_{tr} \ll d$?

Lasso regression technique fits a 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 $N_{tr} \ll d$?

Lasso regression technique fits a 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.
- Does it satisfy the convergence conditions (1, 2)?

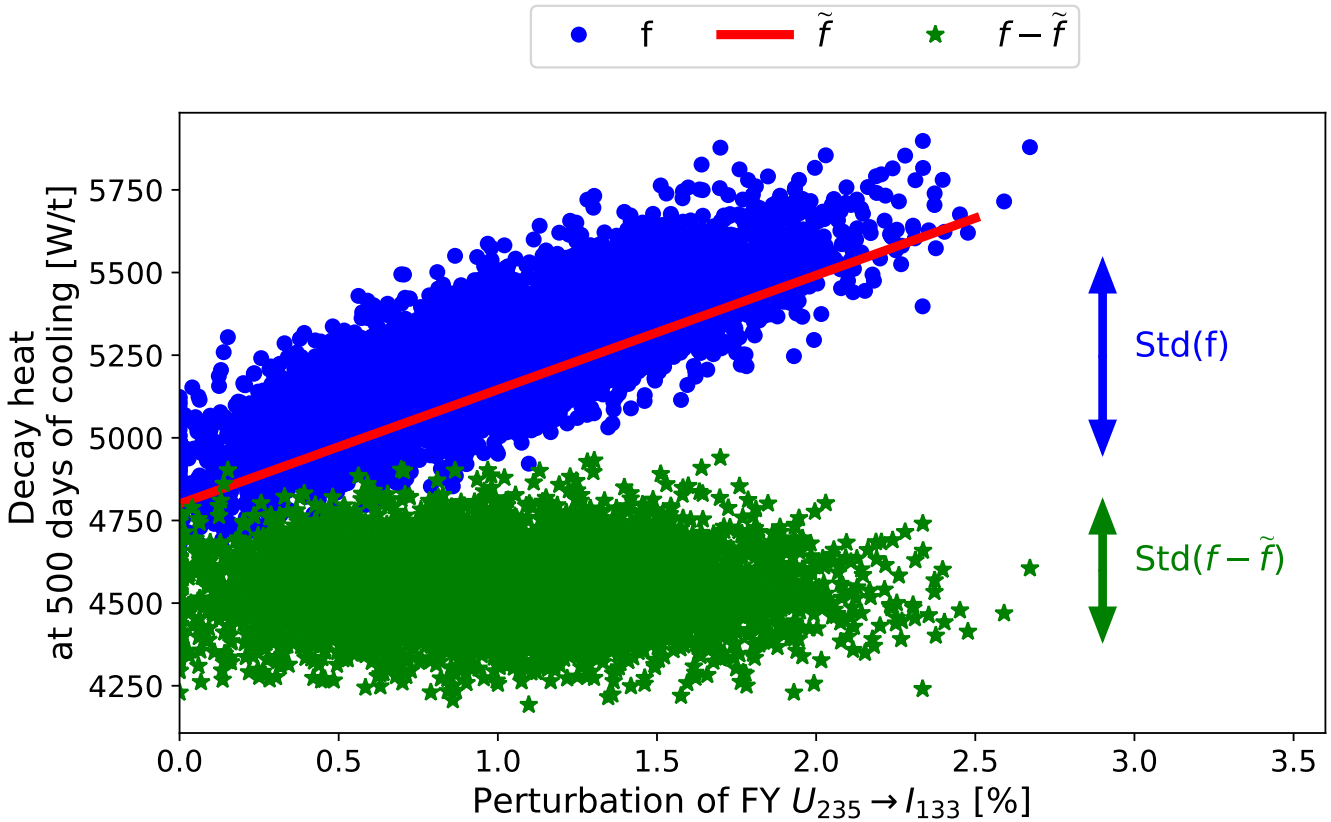
Choosing Surrogate Model

Does Lasso $\widetilde{f}$ satisfy the convergence conditions (1, 2)?

Condition (1) $\mathrm{Var}[f - \widetilde{f}] \leq \mathrm{Var}[f]$, is always satisfied! (as long as λ is chosen correctly)

Choosing Surrogate Model

Does Lasso $\widetilde{f}$ satisfy the convergence conditions (1, 2)?
Condition (1) $\mathrm{Var}[f - \widetilde{f}] \leq \mathrm{Var}[f]$, 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**.

Choosing Surrogate Model

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

Choosing Surrogate Model

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:

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

Choosing Surrogate Model

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.

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 $\boldsymbol{x}_1, f(\boldsymbol{x}_1), \boldsymbol{x}_2, f(\boldsymbol{x}_2), \ldots, \boldsymbol{x}_{N_{tr}}, f(\boldsymbol{x}_{N_{tr}})$.
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 $\mu_{N,M}, \sigma^2_{N,M}$

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: this version of LMC omits some steps (splitting and averaging), see full algorithm in paper.

Lasso Monte Carlo

An actual code example:

```
>>> from sklearn.linear_model import LassoCV, Lasso
>>> from LMC.classLMC import LassoMC

>>> lmc = LassoMC(regressor = Lasso(lambda = 0.02),
                  random_state = seed, verbose = True,
                  validation_method = '5Fold')
>>> N = 150; M = 6000
>>> Xs = get_inputs(N)
>>> ys = [my_simulation(x) for x in Xs]
>>> Zs = get_inputs(M)
>>> lmc.get_single_estimate(Xtrain = Xs,
                            ytrain = ys,
                            Xtest = Zs)
Ntr = 150 labelled samples, Ntest = 6000 unlabelled samples
MC estimates: 5234.470666666667 +- 174.65316984757996
LMC estimates: 5246.745253371253 +- 192.6719429998857
```

Lasso Monte Carlo

An actual code example:

```
>>> from sklearn.linear_model import LassoCV, Lasso
>>> from LMC.classLMC import LassoMC

>>> lmc = LassoMC(regressor = Lasso(lambda = 0.02),
                  random_state = seed, verbose = True,
                  validation_method = '5Fold')
>>> N = 150; M = 6000
>>> Xs = get_inputs(N)
>>> ys = [my_simulation(x) for x in Xs]
>>> Zs = get_inputs(M)
>>> lmc.get_single_estimate(Xtrain = Xs,
                            ytrain = ys,
                            Xtest = Zs)
Ntr = 150 labelled samples, Ntest = 6000 unlabelled samples
MC estimates: 5234.470666666667 +- 174.65316984757996
LMC estimates: 5246.745253371253 +- 192.6719429998857
```

Most expensive step of the algorithm

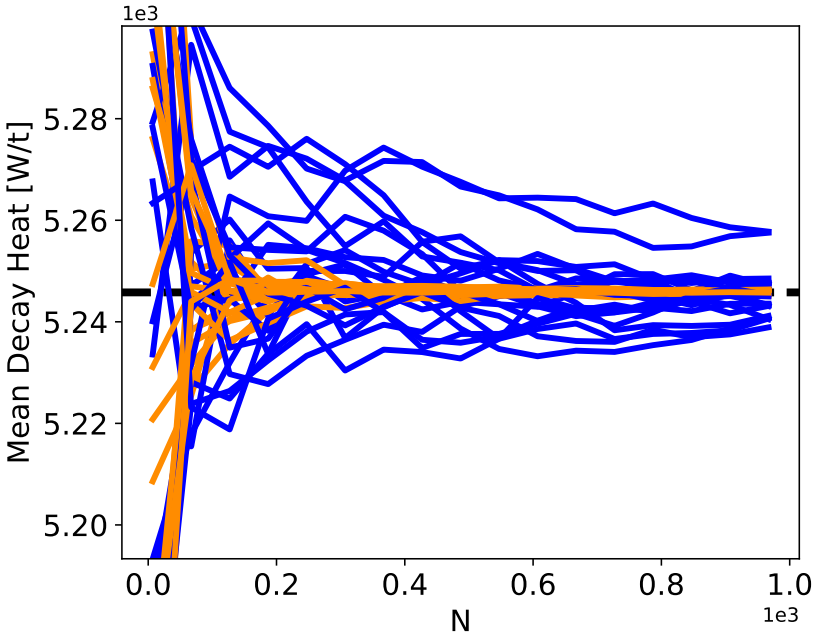
1. Motivation for High-Dimensional UQ: Example from Nuclear Physics
   - Current Methods and Shortcomings
     - Simple MC
     - Surrogate Models
2. New method: Lasso Monte Carlo
   - Multilevel Monte Carlo
   - Lasso Regression
3. Benchmarks
4. Conclusion

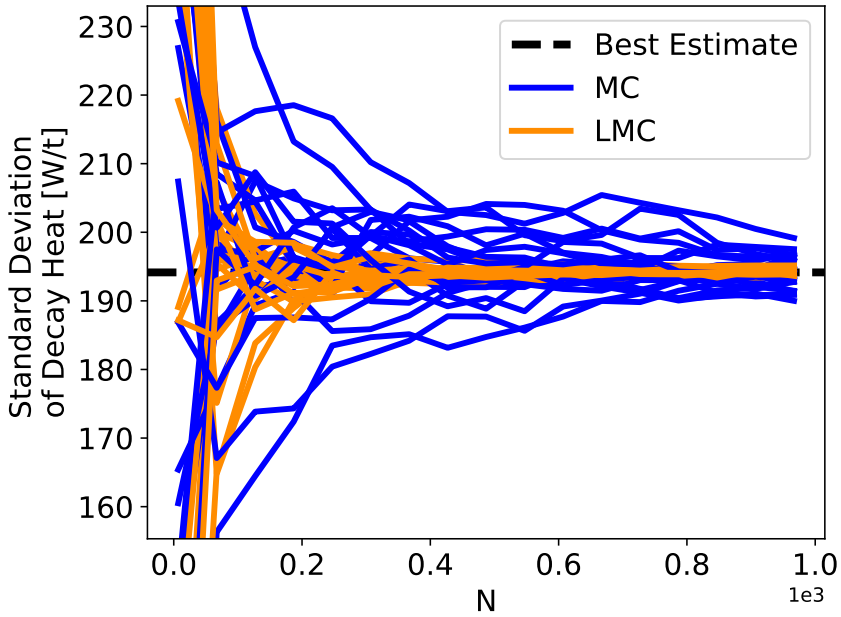
SNF Characterisation

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

$$\text{Nuclear Data} \mapsto \text{Decay Heat}$$

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

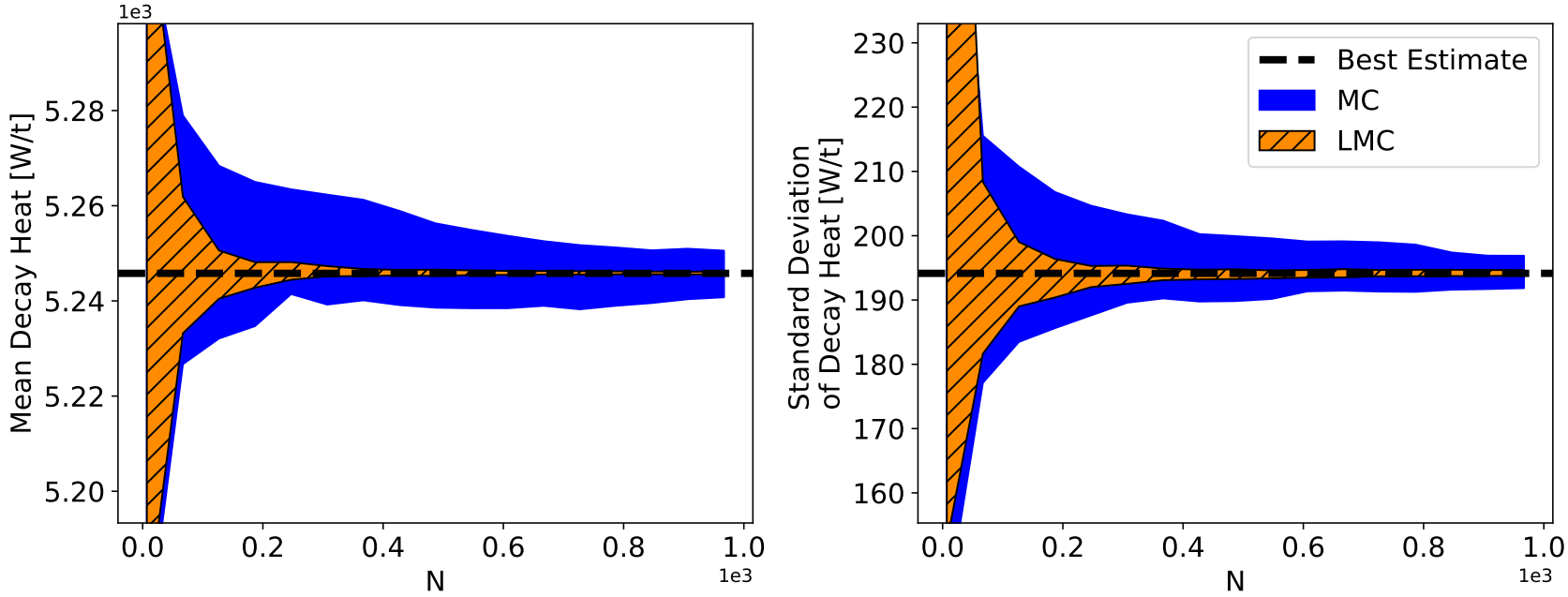




SNF Characterisation

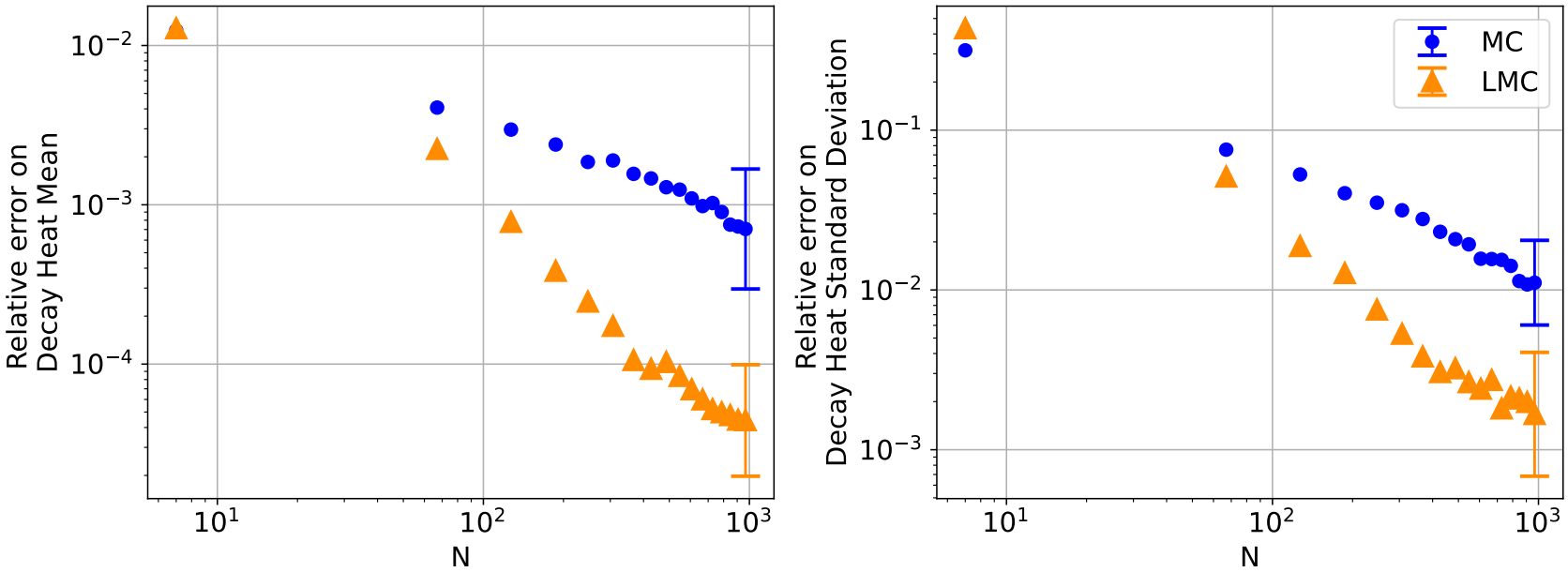
$$f : \mathbb{R}^{15\,557} \rightarrow \mathbb{R}$$

Nuclear Data $\mapsto$ 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**.

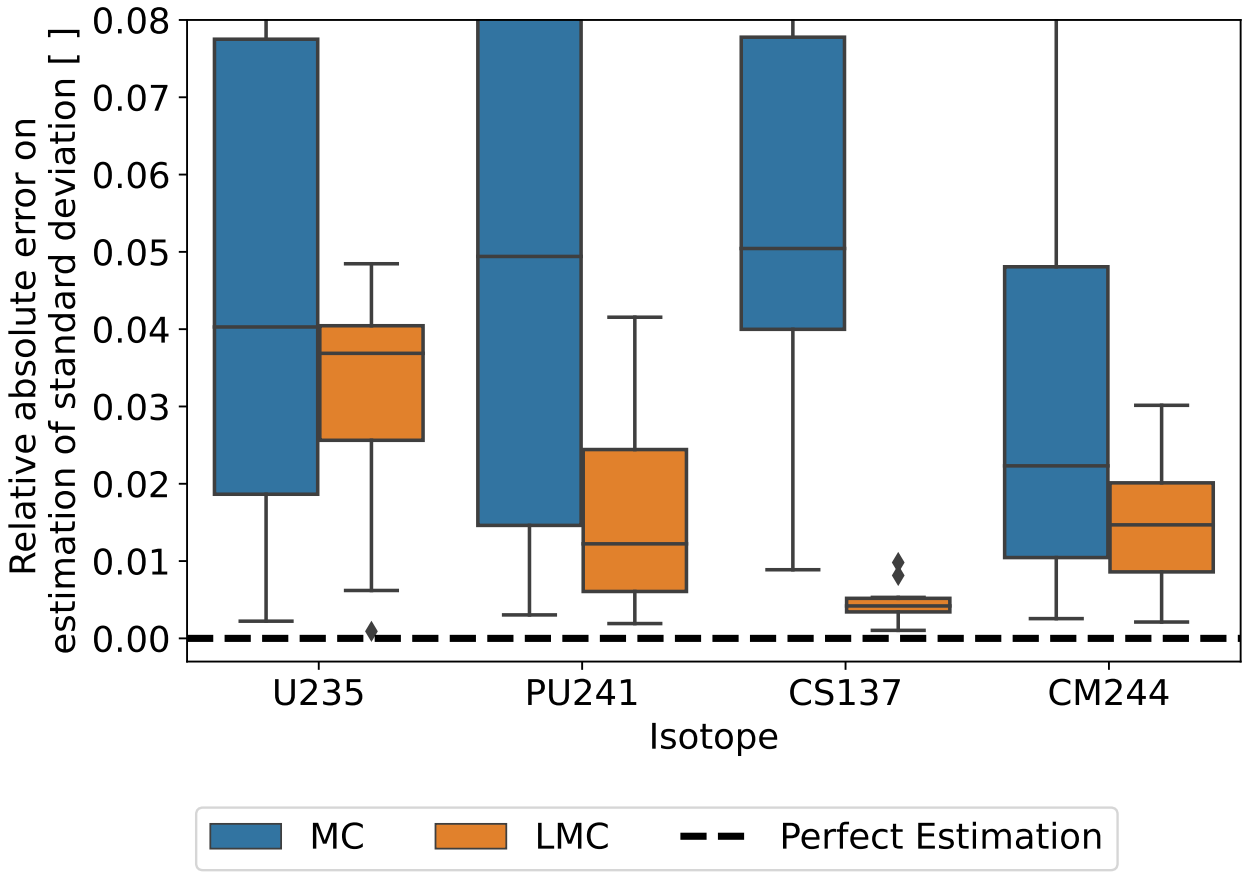
SNF Characterisation

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

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

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

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

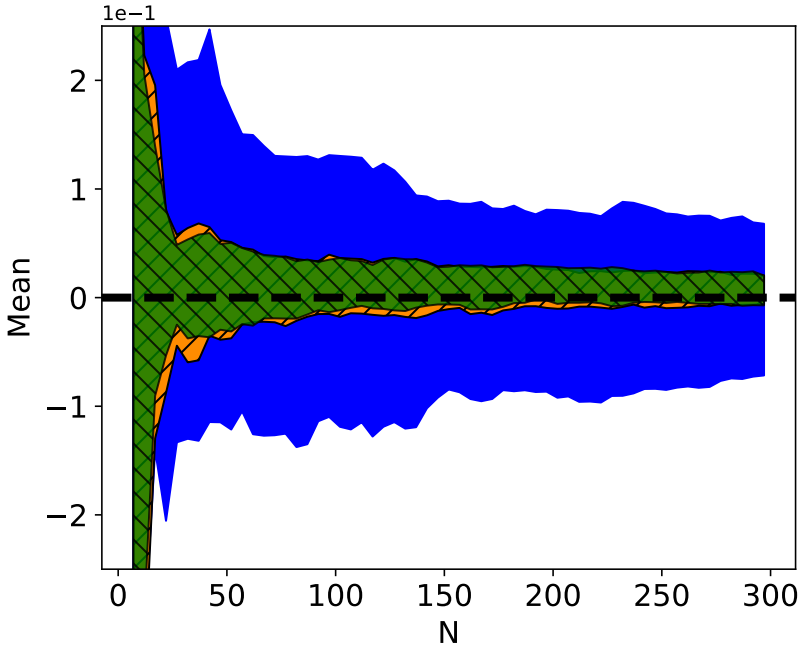


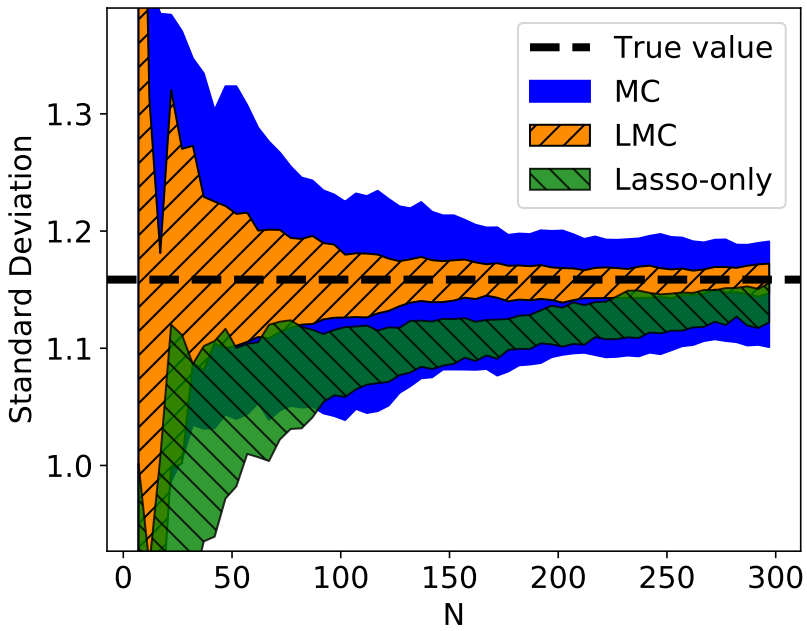
Linear Function

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 $X \sim \mathcal{N}(0, I_d)$.



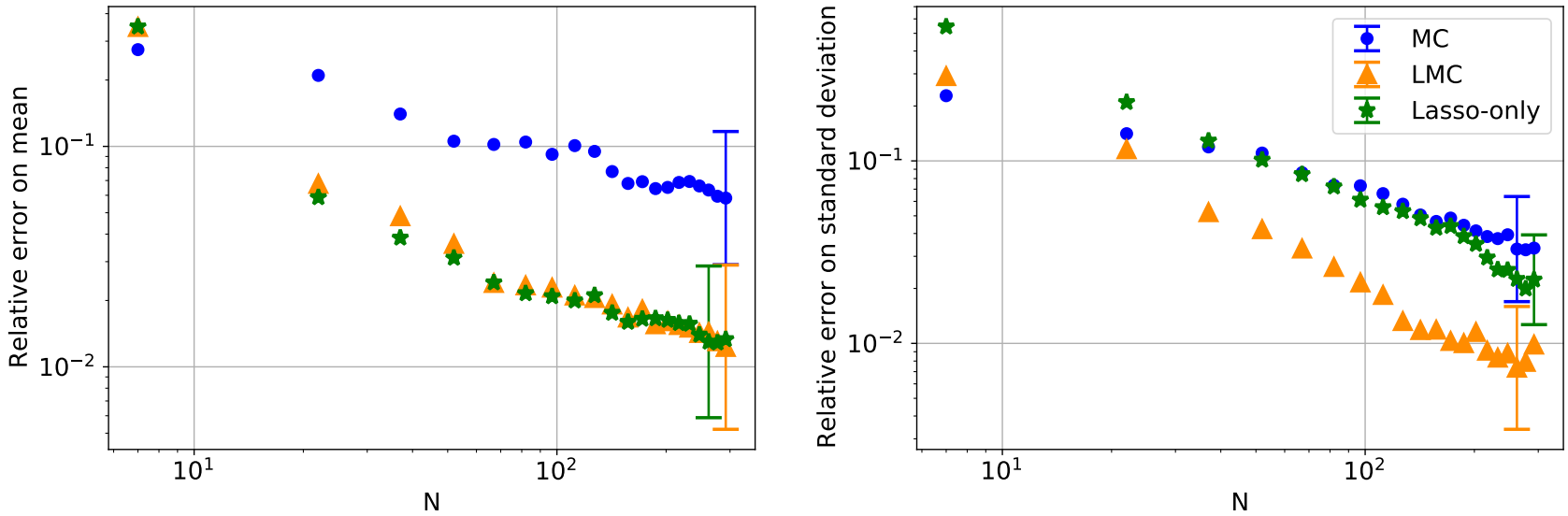


Linear Function

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 $X \sim \mathcal{N}(0, I_d)$.

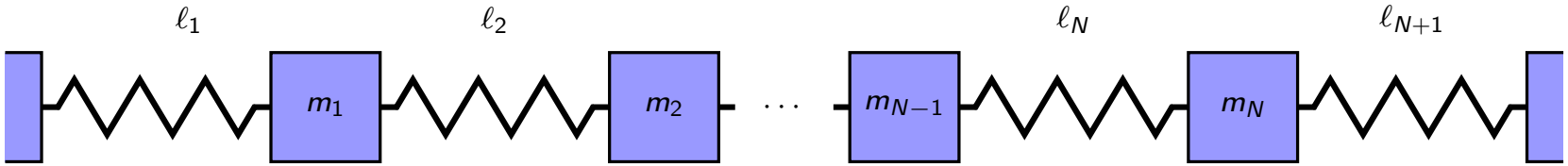


FPUT Oscillator

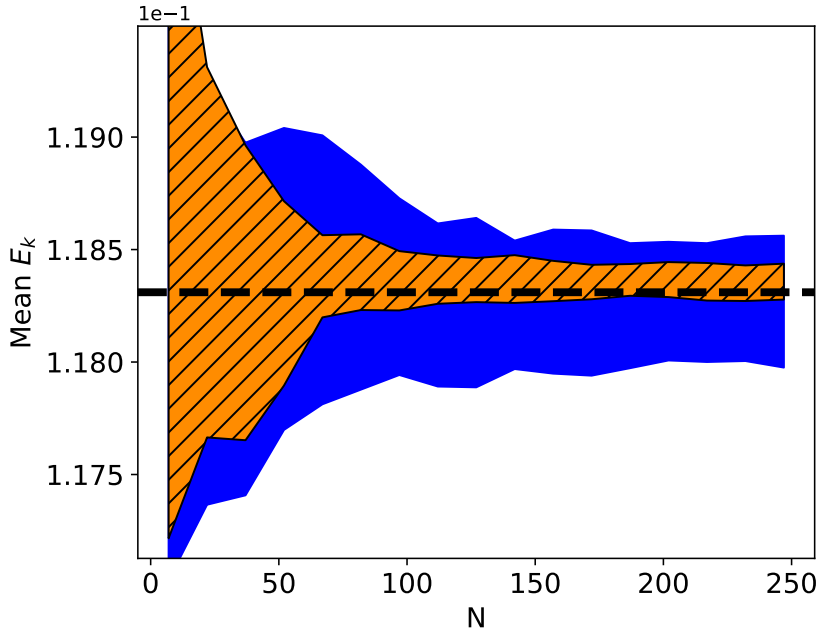
Let there be a chain of nonlinear oscillators

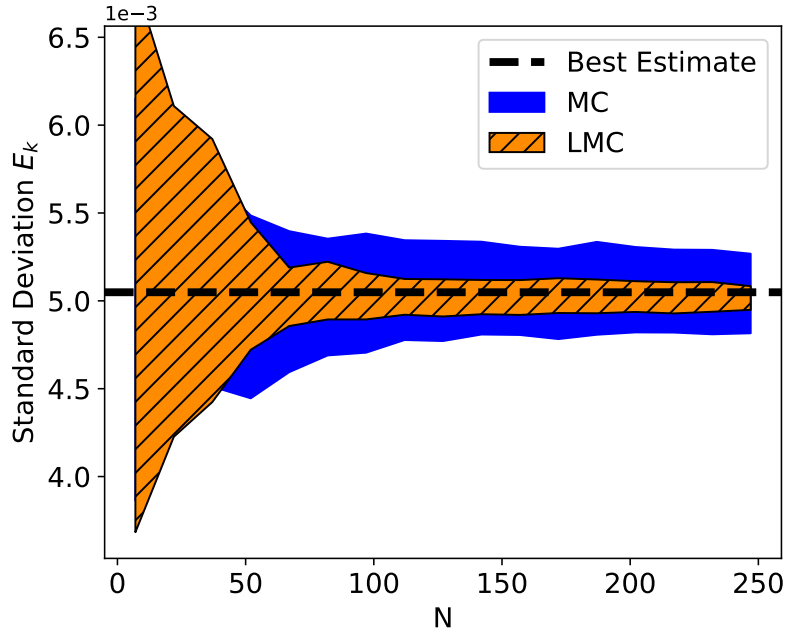
$$\ddot{x}_j = k_j \left(\ell_{j+1} - \ell_j\right) + \alpha k_j (\ell_{j+1}^2 - \ell_j^2)\,, \quad \forall j = 1, 2, ..., N\,,$$

with appropriate boundary conditions.



Consider an uncertainty in the spring constants $k_1, k_2, ..., k_N$, and nonlinear term α, with $N = 40$. What is the uncertainty in E_{kin}?



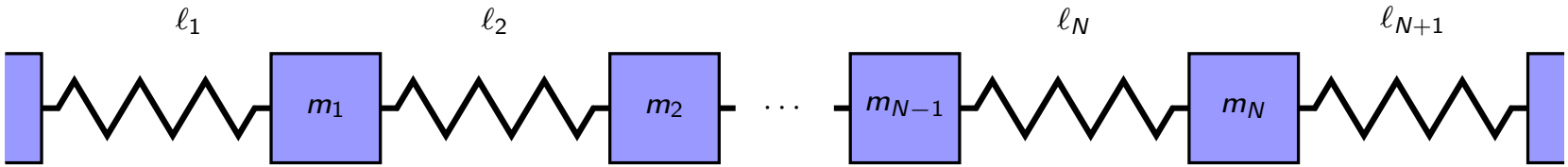


FPUT Oscillator

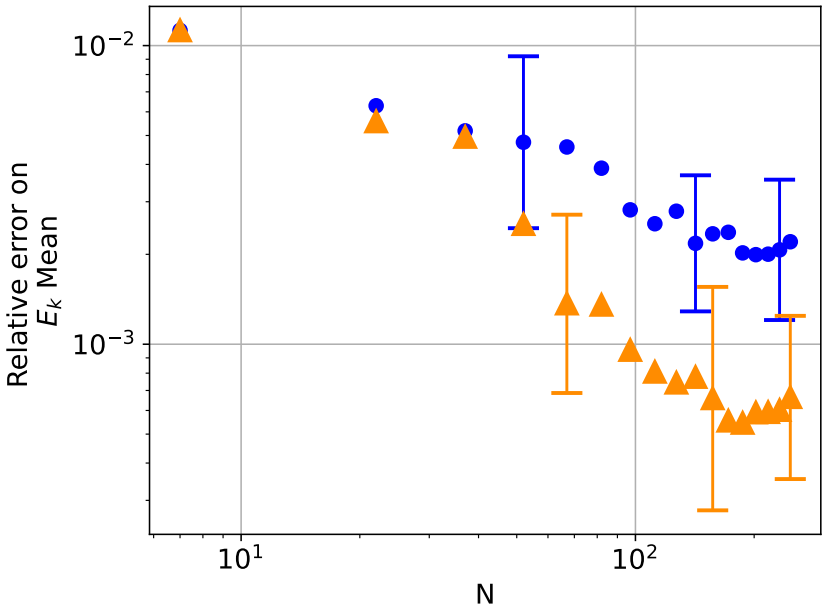
Let there be a chain of nonlinear oscillators

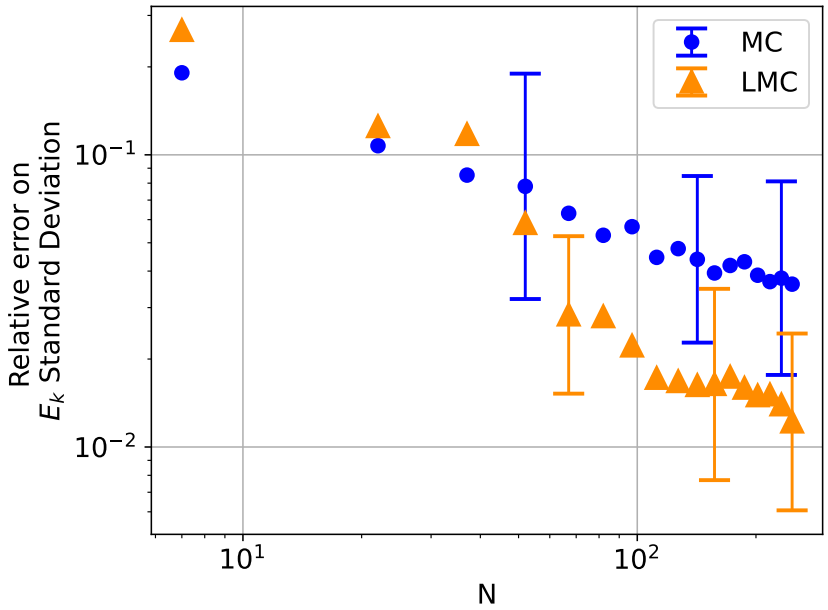
$$\ddot{x}_j = k_j \left(\ell_{j+1} - \ell_j\right) + \alpha k_j (\ell_{j+1}^2 - \ell_j^2)\,, \quad \forall j = 1, 2, ..., N\,,$$

with appropriate boundary conditions.



Consider an uncertainty in the spring constants $k_1, k_2, ..., k_N$, and nonlinear term α, with $N = 40$. What is the uncertainty in E_{kin}?



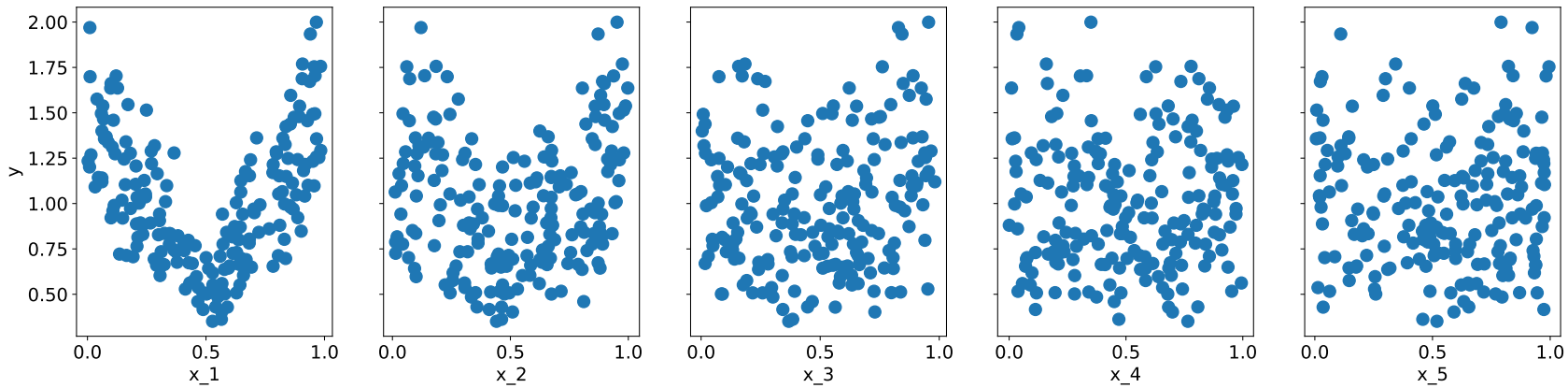


Sobol Function

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

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 $\widetilde{f}(\boldsymbol{x}) = \boldsymbol{\beta} \cdot \boldsymbol{\phi}(\boldsymbol{x})$, with $\boldsymbol{\phi}(\boldsymbol{x}) = |x - 0.5|$.

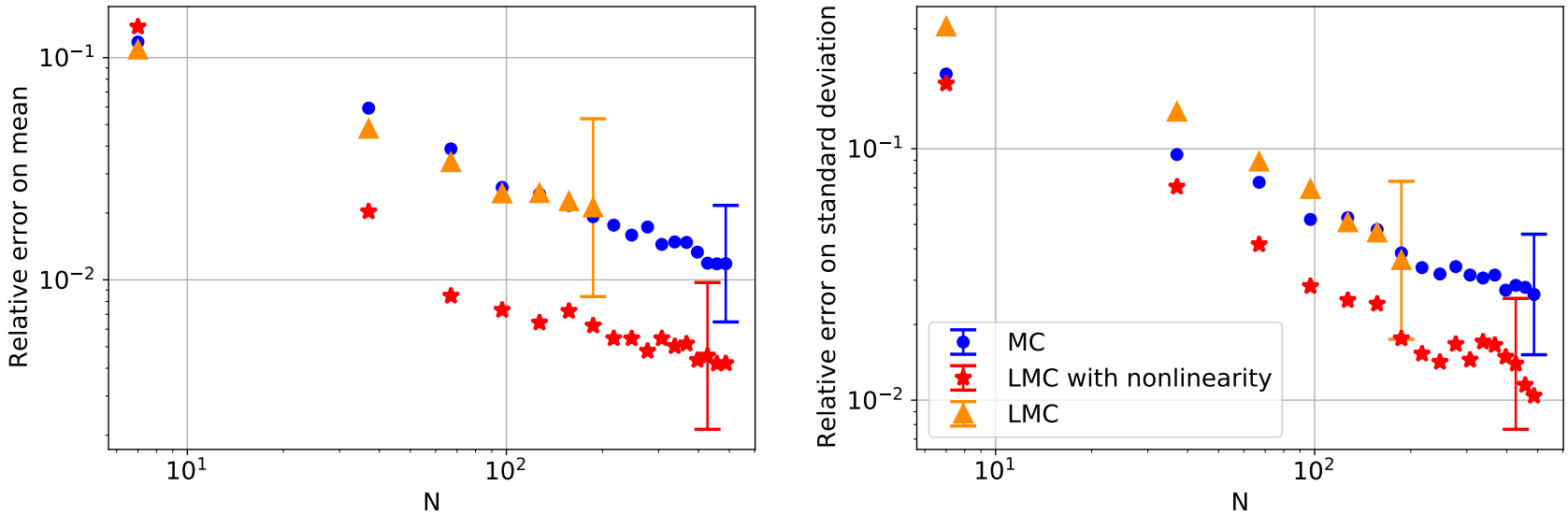


Sobol Function

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

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**.

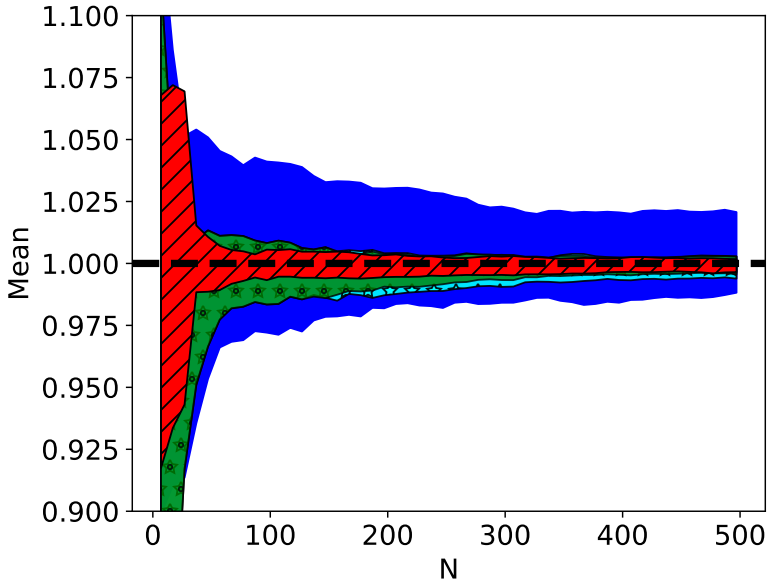
However we can instead fit a modified Lasso model $\widetilde{f}(\boldsymbol{x}) = \boldsymbol{\beta} \cdot \boldsymbol{\phi}(\boldsymbol{x})$, with $\boldsymbol{\phi}(\boldsymbol{x}) = |x - 0.5|$.

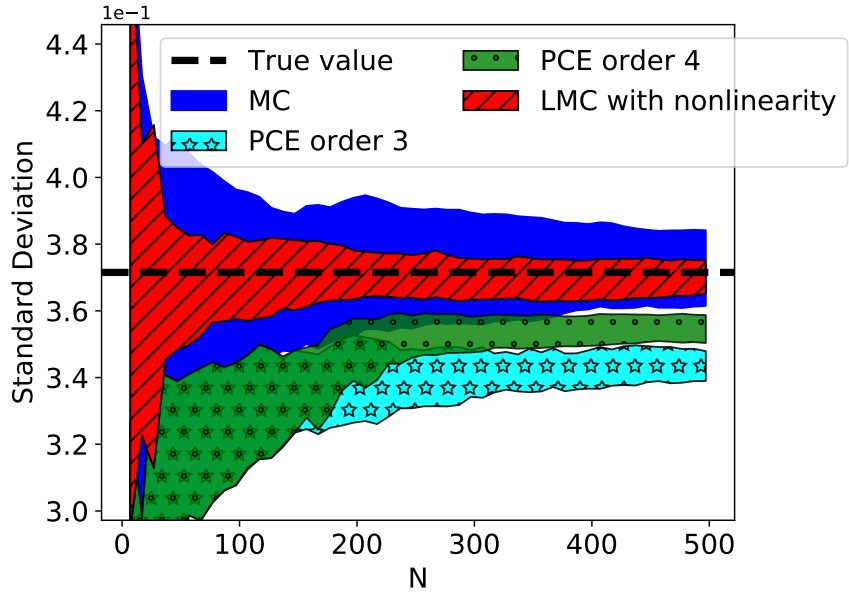


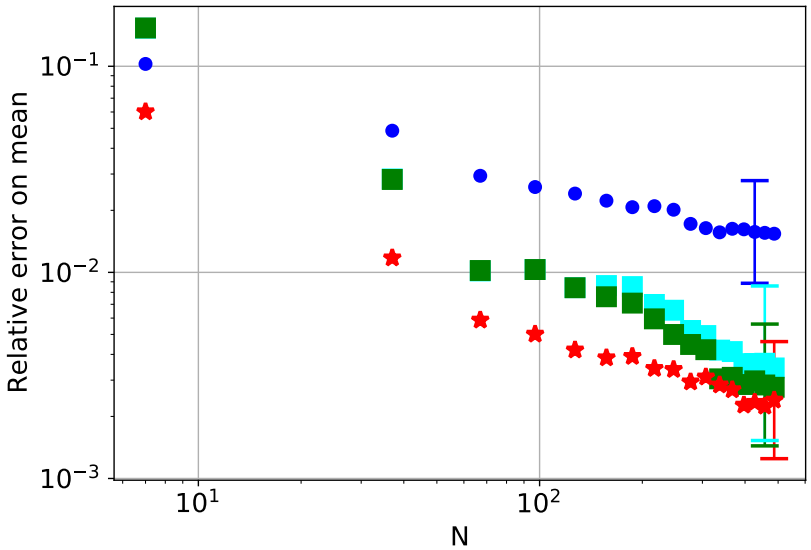
Any kind of surrogate could be used in LMC, **as long as it is strongly regularised**.

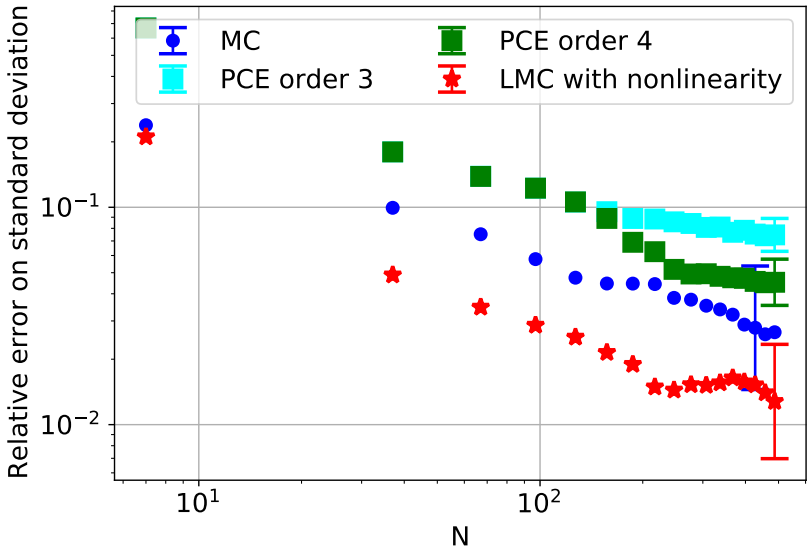
Comparison to PCE

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









Conclusions and Further Work

- LMC converges up to **5 times faster than simple MC**! I.e. same results with 20% of the computing resources.
- LMC is often advantageous over simple MC and surrogate models in high-dimensional settings.
- Given a set of simulations $\boldsymbol{x}_1, f(\boldsymbol{x}_1)$, $\boldsymbol{x}_2, f(\boldsymbol{x}_2)$, ..., $\boldsymbol{x}_N, f(\boldsymbol{x}_N)$, the LMC estimates can be obtained without any extra simulations.
- Could in principle be used with any surrogate model, as long as it is regularised

- The speedup is not constant, it's very dependent on f.
- Unfortunately, theoretical guarantee of faster convergence is conditioned on:
  - f being close to *a noisy linear function*.
  - an optimal choice of regularisation parameter 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, January 2021.

C. Fiorina, A. Scolaro, D. Siefman, M. Hursin, and A. Pautz.

Artificial Neural Networks as Surrogate Models for Uncertainty Quantification and Data Assimilation in 2-D/3-D Fuel Performance Studies.

Journal of Nuclear Engineering, 1(1):54–62, December 2020.

Virginie Solans, Dimitri Rochman, Christian Brazell, Alexander Vasiliev, Hakim Ferroukhi, and Andreas 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.

Michael B. Giles.

Multilevel Monte Carlo Path Simulation.

Operations Research, 56(3):607–617, June 2008.

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, 414:109466, August 2020.

Robert Tibshirani.

Regression Shrinkage and Selection via the Lasso.

Journal of the Royal Statistical Society. Series B (Methodological), 58(1):267–288, 1996.

Claudio Bierig and Alexey Chernov.

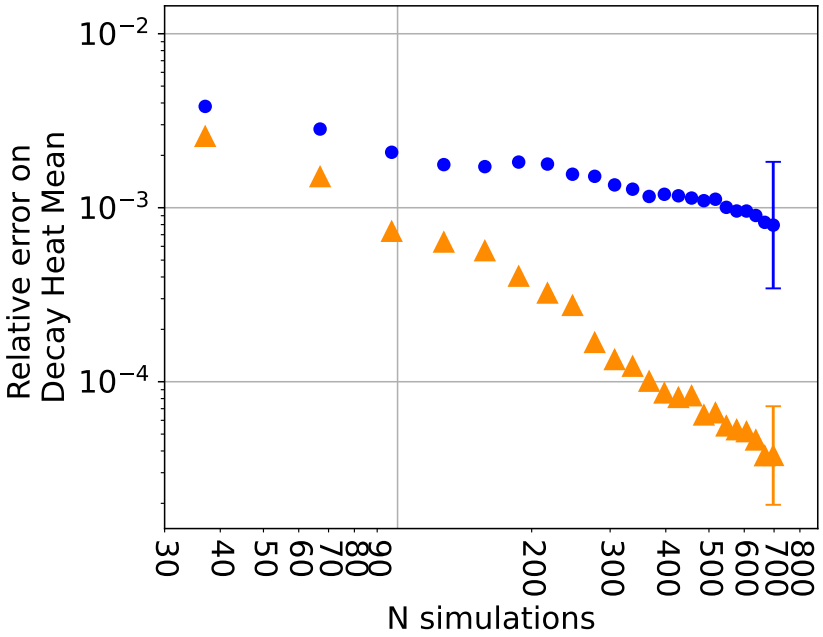
Estimation of arbitrary order central statistical moments by the multilevel Monte Carlo method.

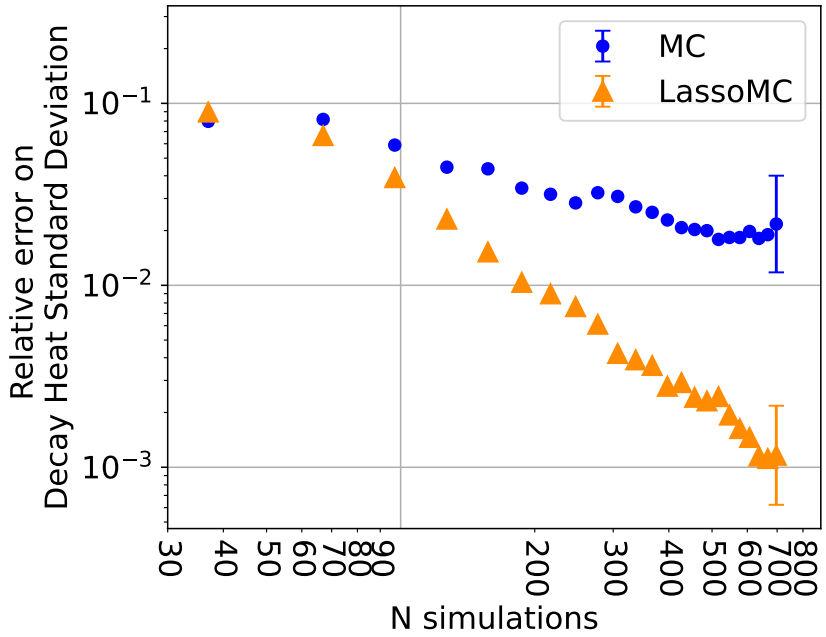
Stochastics and Partial Differential Equations Analysis and Computations, 4(1):3–40, March 2016.

Extra Slides

Results

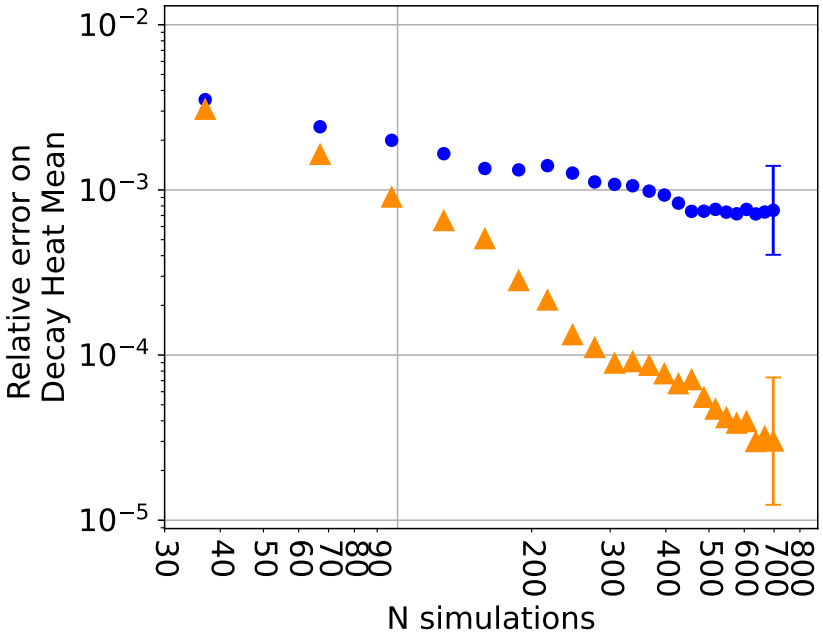
Decay heat prediction at 30 years of cooling:

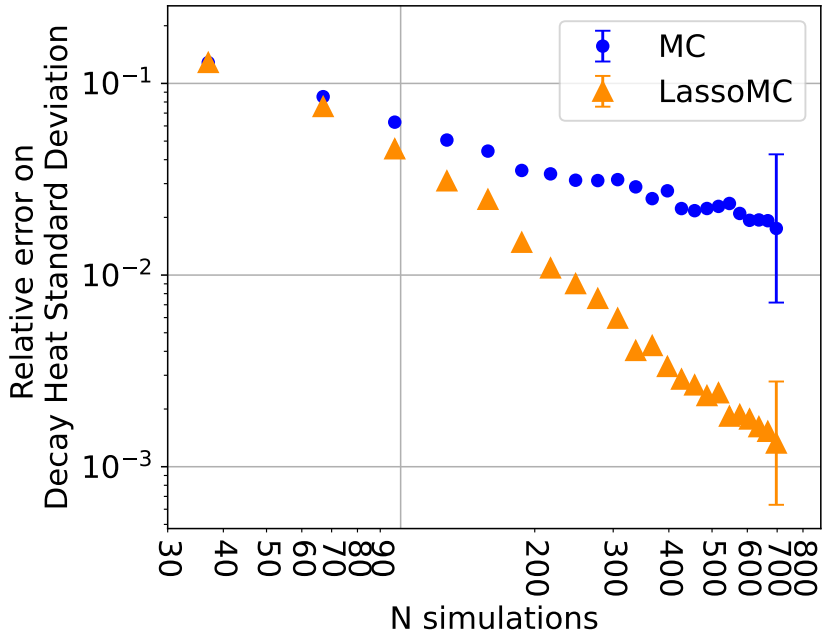




Results

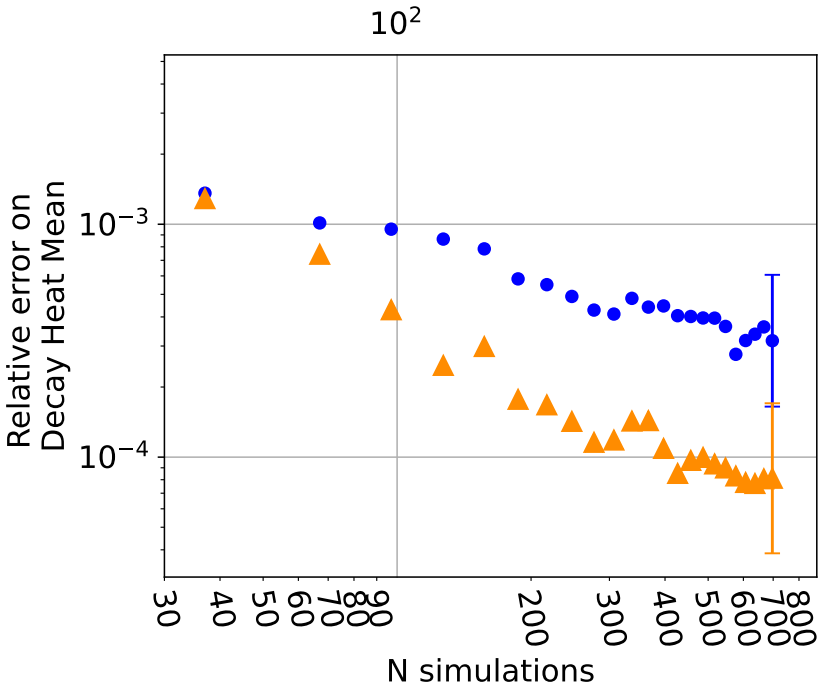
Decay heat prediction at 50 years of cooling:

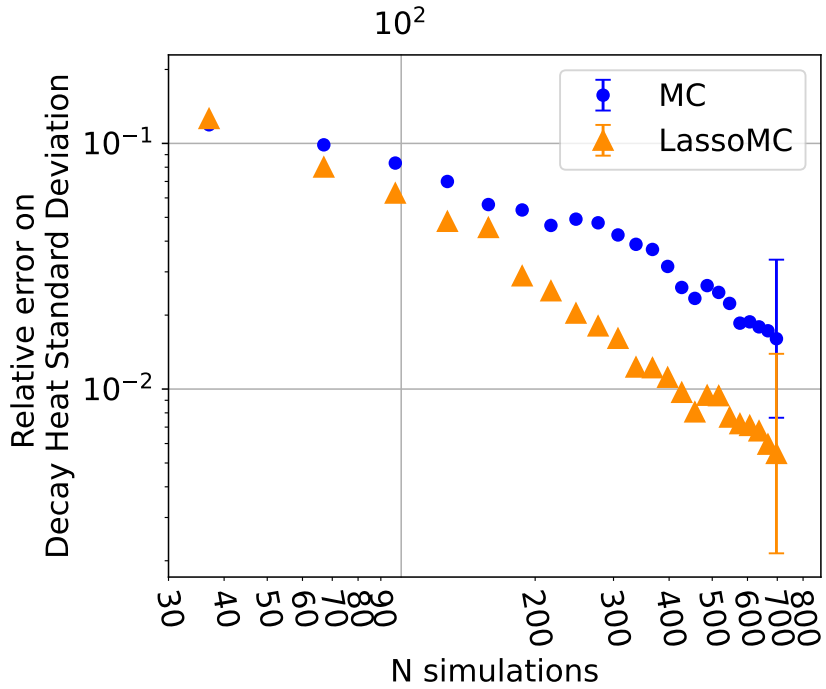




Results

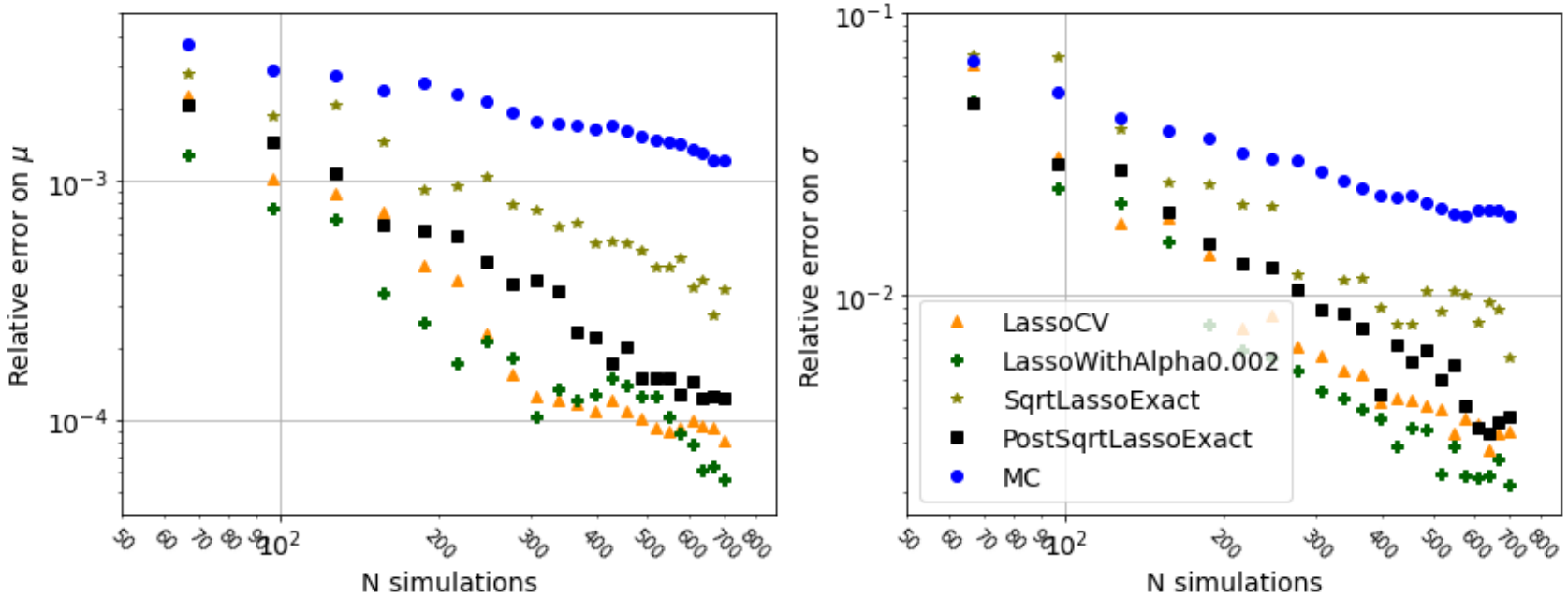
U235 concentration at discharge





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 $(\sigma^2_{n,M})_\mathrm{s}$.
8: **end for**
9: Compute the LMC mean and variance, by averaging out the estimations of each split

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