

PAUL SCHERRER INSTITUT



**ETH** zürich

swiss**nuclear**



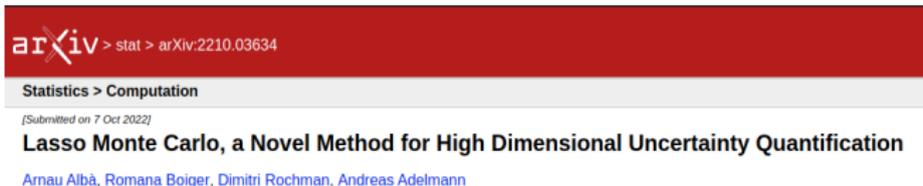
**Arnau Albà**, R. Boiger, D. Rochman, A. Adelmann :: AMAS Group, PSI

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

Zurich PhD Seminar, 27th January 2023

Contact: [arnau.albajacas@psi.ch](mailto:arnau.albajacas@psi.ch)

Pre-print available:



Currently under review for SIAM UQ journal.

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

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

# 1. Motivation for High-Dimensional UQ: Characterisation of Spent Nuclear Fuel

## Current Methods and Shortcomings

Simple MC

Surrogate Models

## 2. New method: Lasso Monte Carlo

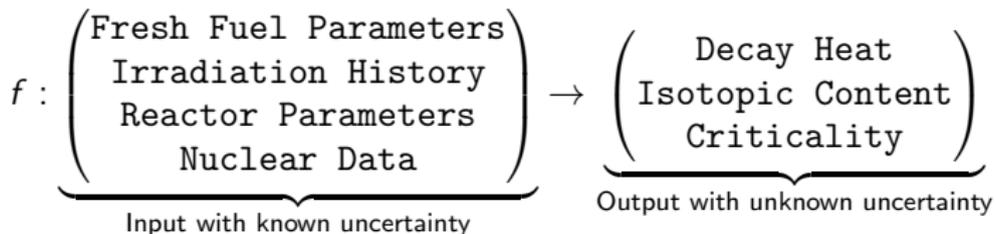
Multilevel Monte Carlo

Lasso Regression

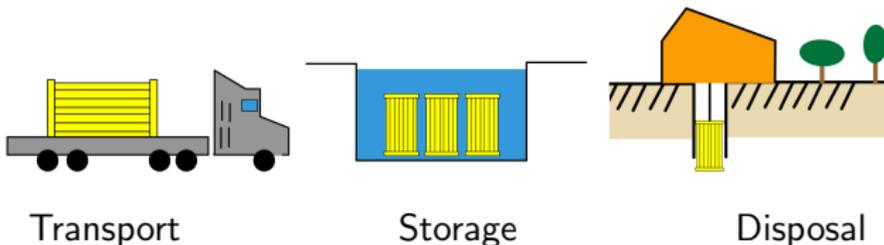
## 3. Benchmarks

## 4. Conclusion

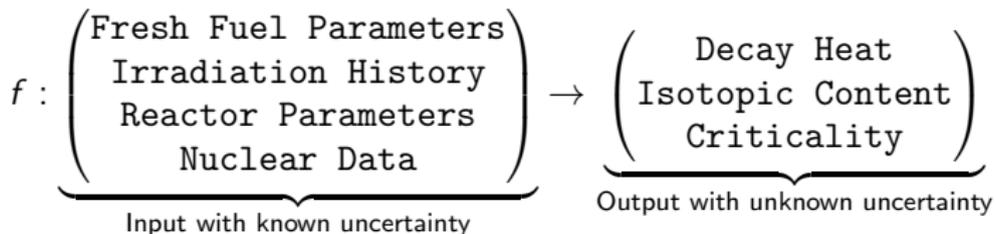
Nuclear burnup and criticality simulations are used to characterise spent nuclear fuel:



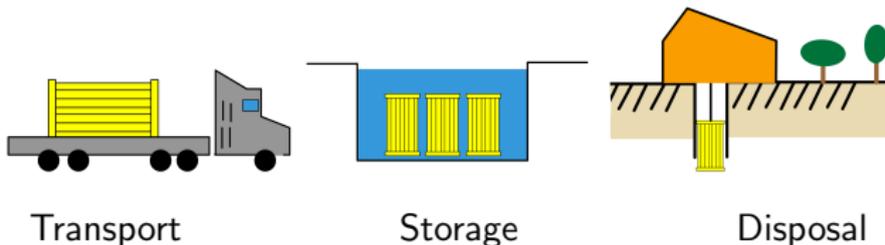
Any uncertainty in outputs will increase the risks and costs of



Nuclear burnup and criticality simulations are used to characterise spent nuclear fuel:



Any uncertainty in outputs will increase the risks and costs of



**Accurate estimation of uncertainty saves money and reduces risks.**

More specifically, we are dealing with **high-dimensional** UQ, due to Nuclear Data:

$$f : \underbrace{\text{Nuclear Data}}_{\in \mathbb{R}^{15\,557}} \rightarrow \begin{pmatrix} \text{Decay Heat} \\ \text{Isotopic Content} \\ \text{Criticality} \end{pmatrix}$$

Nuclear data includes:

- Cross-sections
- Fission spectrum
- Fission yield
- Neutron multiplicity

For every isotope and energy.

## 1. Motivation for High-Dimensional UQ: Characterisation of Spent Nuclear Fuel

### Current Methods and Shortcomings

Simple MC

Surrogate Models

## 2. New method: Lasso Monte Carlo

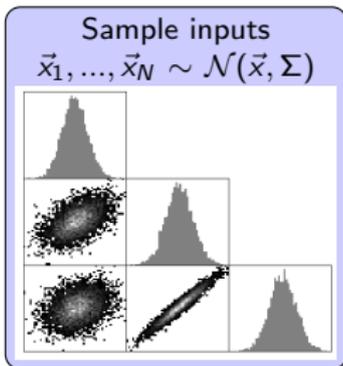
Multilevel Monte Carlo

Lasso Regression

## 3. Benchmarks

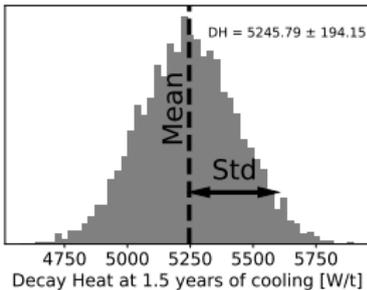
## 4. Conclusion

1.



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

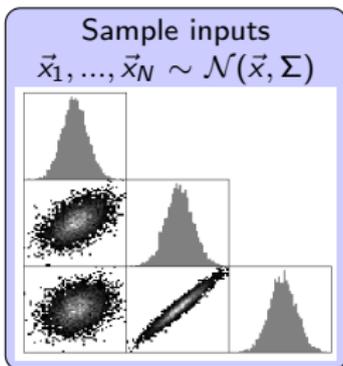
Compute sample  
 mean and variance



2. Compute sample mean and variance

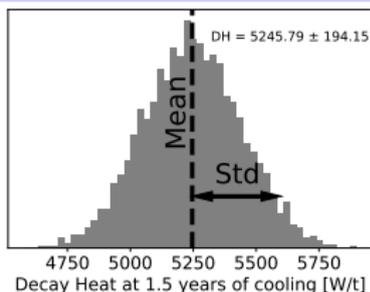
$$\mu_N = \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}_i), \quad \sigma_N^2 = \frac{1}{N-1} \sum_{i=1}^N \left( f(\mathbf{x}_i) - \sum_{j=1}^N \frac{f(\mathbf{x}_j)}{N} \right)^2.$$

1.



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

Compute sample  
 mean and variance



2. Compute sample mean and variance

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

Simple MC is **unbiased**, but **slow** ( $\text{MSE} = \mathcal{O}(\frac{1}{N})$ ):

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

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

Simple MC is slow!

- $\text{MSE} = \mathcal{O}\left(\frac{1}{N}\right)$ , i.e. many simulations required!
- For SNF characterisation  $N \sim 1000$ .
- Each simulation lasts a few hours.
- Expecting  $> 12000$  fuel assemblies in Switzerland.
- $\Rightarrow$  millions of CPU hours  $\Rightarrow$  **MC UQ is too slow!**

A more modern approach: Surrogate models (e.g. PCE [Frey and Adelman, 2021], NNs [Fiorina et al., 2020, Solans et al., 2021]):

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

A more modern approach: Surrogate models (e.g. PCE [Frey and Adelman, 2021], NNs [Fiorina et al., 2020, Solans et al., 2021]):

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

A more modern approach: Surrogate models (e.g. PCE [Frey and Adelman, 2021], NNs [Fiorina et al., 2020, Solans et al., 2021]):

1. Gather a training set  $\mathbf{x}_1, f(\mathbf{x}_1), \mathbf{x}_2, f(\mathbf{x}_2), \dots, \mathbf{x}_{N_{tr}}, f(\mathbf{x}_{N_{tr}})$ .
  2. Train a surrogate model  $\tilde{f} \sim f$ , that is **fast to evaluate**.
  3. Run surrogate  $M$  times to obtain samples  $\tilde{f}(\mathbf{z}_1), \tilde{f}(\mathbf{z}_2), \dots, \tilde{f}(\mathbf{z}_M)$ , with  $Z \sim \mathcal{N}(\mathbf{x}, \Sigma)$ .
  4. Compute sample mean  $\tilde{\mu}_M$  and variance  $\tilde{\sigma}_M^2$ .
- Converges very fast, since  $M$  can be large
  - Training  $\tilde{f}$  requires a big training set, at least  $N_{tr} > d$  (*curse of dimensionality*).
  - Estimates are biased since

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

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

A more modern approach: Surrogate models (e.g. PCE [Frey and Adelman, 2021], NNs [Fiorina et al., 2020, Solans et al., 2021]):

1. Gather a training set  $\mathbf{x}_1, f(\mathbf{x}_1), \mathbf{x}_2, f(\mathbf{x}_2), \dots, \mathbf{x}_{N_{tr}}, f(\mathbf{x}_{N_{tr}})$ .
  2. Train a surrogate model  $\tilde{f} \sim f$ , that is **fast to evaluate**.
  3. Run surrogate  $M$  times to obtain samples  $\tilde{f}(\mathbf{z}_1), \tilde{f}(\mathbf{z}_2), \dots, \tilde{f}(\mathbf{z}_M)$ , with  $\mathbf{Z} \sim \mathcal{N}(\mathbf{x}, \Sigma)$ .
  4. Compute sample mean  $\tilde{\mu}_M$  and variance  $\tilde{\sigma}_M^2$ .
- Converges very fast, since  $M$  can be large
  - Training  $\tilde{f}$  requires a big training set, at least  $N_{tr} > d$  (*curse of dimensionality*).
  - Estimates are biased since

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

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

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

## 1. Motivation for High-Dimensional UQ: Characterisation of Spent Nuclear Fuel

Current Methods and Shortcomings

Simple MC

Surrogate Models

## 2. New method: Lasso Monte Carlo

Multilevel Monte Carlo

Lasso Regression

## 3. Benchmarks

## 4. Conclusion

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

- Multilevel Monte Carlo (MLMC) [Giles, 2008, Krumscheid et al., 2020]
- Lasso regression [Tibshirani, 1996]

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

- Multilevel Monte Carlo (MLMC) [Giles, 2008, Krumscheid et al., 2020]  $\Rightarrow$  Combine models of different fidelities into an **unbiased estimator**.
- Lasso regression [Tibshirani, 1996]  $\Rightarrow$  Train surrogate models with **small training sets**.

Let

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

Let

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

Then the estimators are

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

$$\sigma_{N,M}^2 = \tilde{\sigma}_M^2 + \sigma_N^2 - \tilde{\sigma}_N^2.$$

Let

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

Then the estimators are

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

$$\sigma_{N,M}^2 = \tilde{\sigma}_M^2 + \sigma_N^2 - \tilde{\sigma}_N^2.$$

- Estimators are unbiased
- More accurate than simple MC  $\mu_N, \sigma_N^2$  (under certain conditions of  $f$ ).

## Common usage of MLMC:

1. Gather a training set  $\mathbf{x}_1, f(\mathbf{x}_1), \mathbf{x}_2, f(\mathbf{x}_2), \dots, \mathbf{x}_{N_{tr}}, f(\mathbf{x}_{N_{tr}})$ .
2. Train a surrogate model  $\tilde{f} \sim f$ , that is **fast to evaluate**.
3. Evaluate  $\tilde{f}$   $N + M$  times to obtain  $\tilde{f}(\mathbf{x}_1), \tilde{f}(\mathbf{x}_2), \dots, \tilde{f}(\mathbf{x}_N)$ , and  $\tilde{f}(\mathbf{z}_1), \tilde{f}(\mathbf{z}_2), \dots, \tilde{f}(\mathbf{z}_M)$ .
4. Evaluate  $f$   $N$  times, to obtain  $f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_N)$ .
5. Compute 2-level estimators  $\mu_{N,M}, \sigma_{N,M}^2$ .

Common usage of MLMC:

1. Gather a training set  $\mathbf{x}_1, f(\mathbf{x}_1), \mathbf{x}_2, f(\mathbf{x}_2), \dots, \mathbf{x}_{N_{tr}}, f(\mathbf{x}_{N_{tr}})$ .
2. Train a surrogate model  $\tilde{f} \sim f$ , that is **fast to evaluate**.
3. Evaluate  $\tilde{f}$   $N + M$  times to obtain  $\tilde{f}(\mathbf{x}_1), \tilde{f}(\mathbf{x}_2), \dots, \tilde{f}(\mathbf{x}_N)$ , and  $\tilde{f}(\mathbf{z}_1), \tilde{f}(\mathbf{z}_2), \dots, \tilde{f}(\mathbf{z}_M)$ .
4. Evaluate  $f$   $N$  times, to obtain  $f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_N)$ .
5. Compute 2-level estimators  $\mu_{N,M}, \sigma_{N,M}^2$ .
  - Unbiased.
  - More accurate than simple MC for a given  $N$ .
  - However, bottleneck is still generating the training set  $N_{tr}$  (especially in high-dimensional cases).

Surrogate model that can be trained with a small training set  $N_{tr} \ll d$ ?

Surrogate model that can be trained with a small training set  $N_{tr} \ll d$ ?

Lasso regression technique fits a sparse linear model:

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

by minimising loss function

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

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

Surrogate model that can be trained with a small training set  $N_{tr} \ll d$ ?

Lasso regression technique fits a sparse linear model:

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

by minimising loss function

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

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

- Lasso can be trained for small training sets, without overfitting.

Two-level MC + Lasso:

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

Two-level MC + Lasso:

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

Reuse the same set for training!

LMC algorithm:

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

LMC algorithm:

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

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
```

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)
```

Most expensive  
step of the algorithm

```
Ntr = 150 labelled samples, Ntest = 6000 unlabelled samples
MC estimates: 5234.470666666667 +- 174.65316984757996
LMC estimates: 5246.745253371253 +- 192.6719429998857
```

## 1. Motivation for High-Dimensional UQ: Characterisation of Spent Nuclear Fuel

### Current Methods and Shortcomings

Simple MC

Surrogate Models

## 2. New method: Lasso Monte Carlo

Multilevel Monte Carlo

Lasso Regression

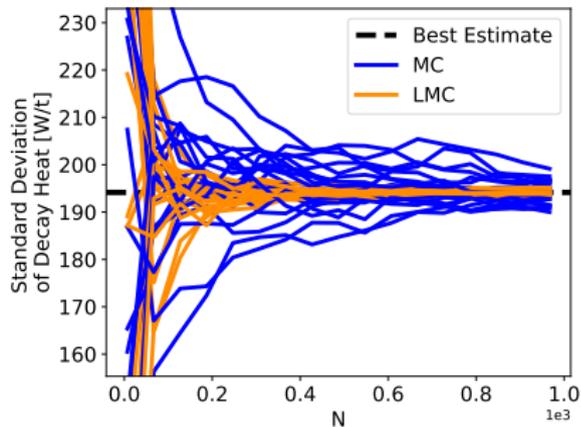
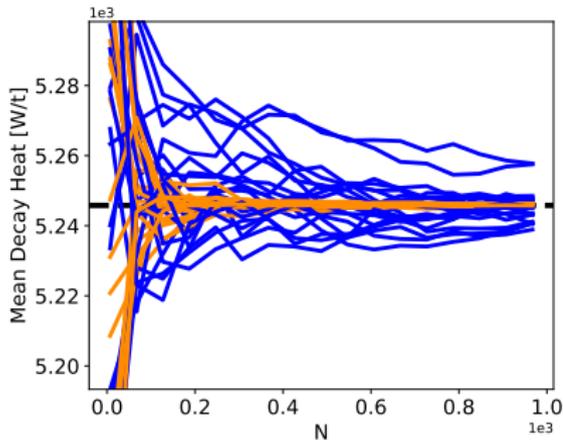
## 3. Benchmarks

## 4. Conclusion

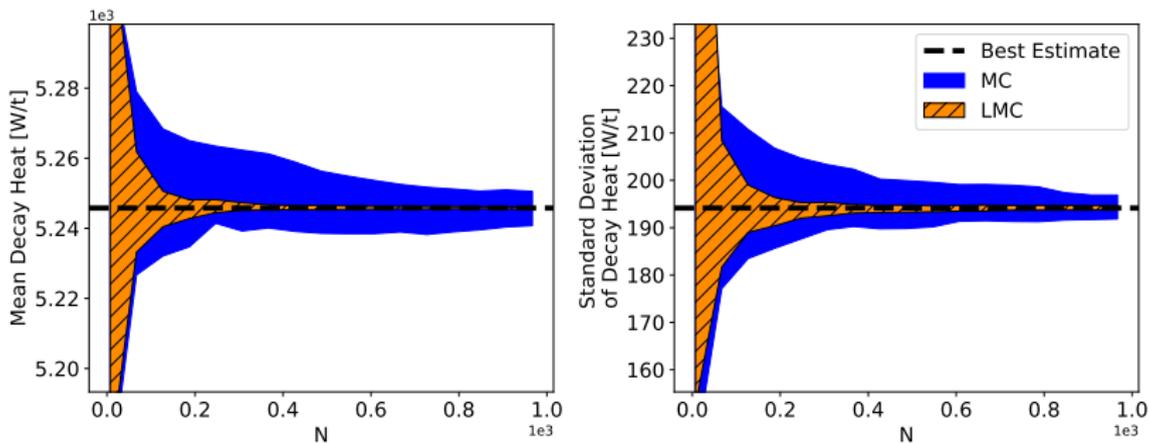
$$f: \mathbb{R}^{15557} \rightarrow \mathbb{R}$$

Nuclear Data  $\mapsto$  Decay Heat

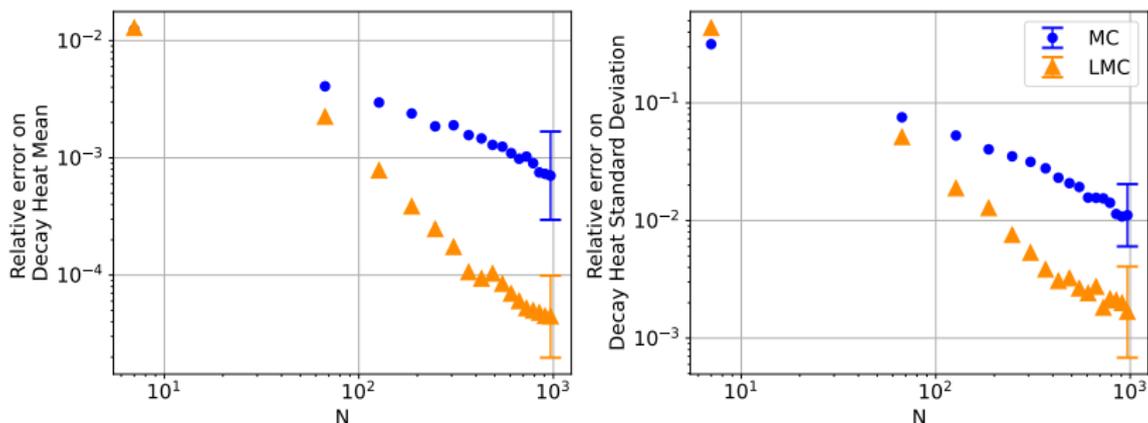
Plots show increasing  $N$ , and fixed  $M = 6000$ .



$f: \mathbb{R}^{1557} \rightarrow \mathbb{R}$   
 Nuclear Data  $\mapsto$  Decay Heat



$f: \mathbb{R}^{1557} \rightarrow \mathbb{R}$   
 Nuclear Data  $\mapsto$  Decay Heat



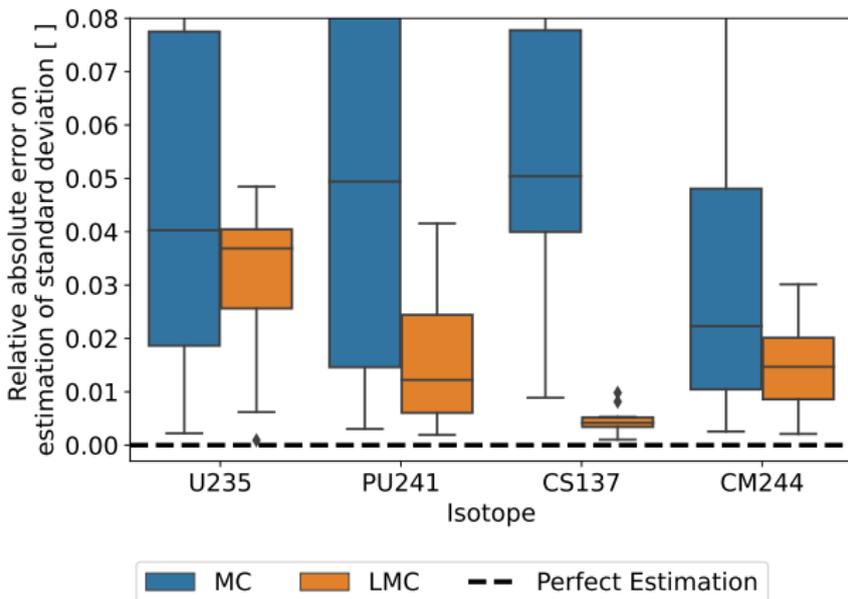
To obtain a 1% error in estimations, simple MC requires  $N = 1000$  expensive simulations  $f$ , while LMC requires  $N = 200$ . I.e. **5 times speedup thanks to LMC**.

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

$$f: \mathbb{R}^{1557} \rightarrow \mathbb{R}$$

Nuclear Data  $\mapsto$  Isotopic Content

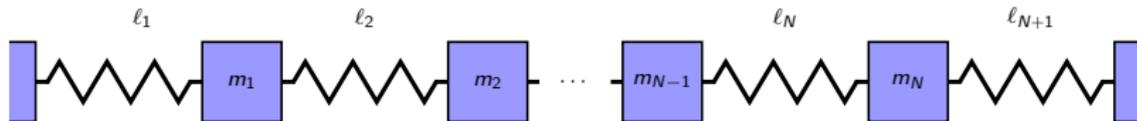
Predicting different quantities gives different improvements. But **LMC is always equal or better than simple MC**.



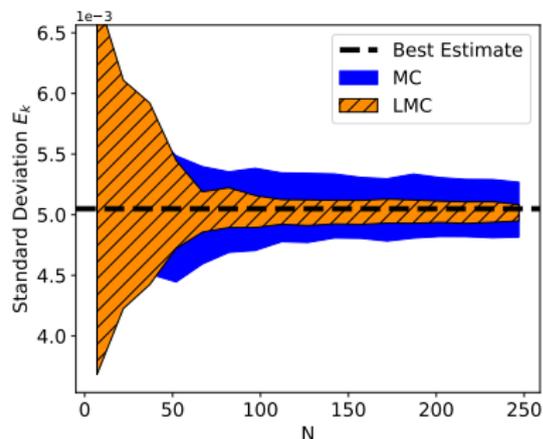
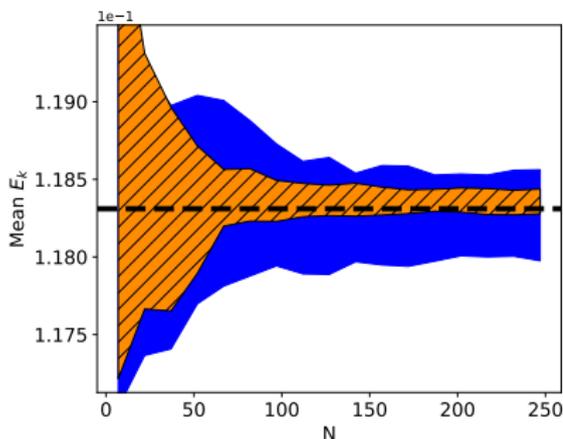
Let there be a chain of nonlinear oscillators

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

with appropriate boundary conditions.



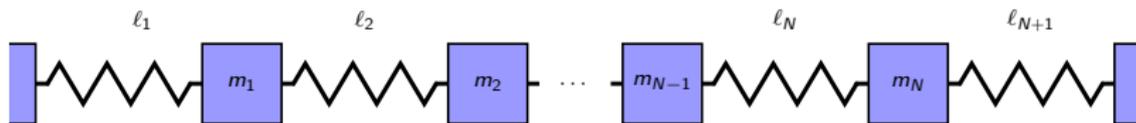
Consider an uncertainty in the spring constants  $k_1, k_2, \dots, k_N$ , and nonlinear term  $\alpha$ , with  $N = 40$ . What is the uncertainty in  $E_{kin}$ ?



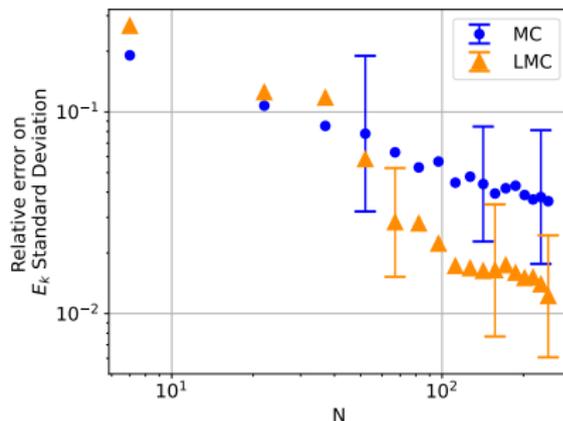
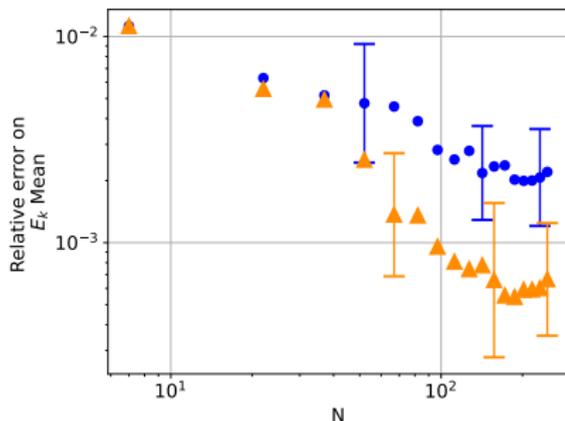
Let there be a chain of nonlinear oscillators

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

with appropriate boundary conditions.



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

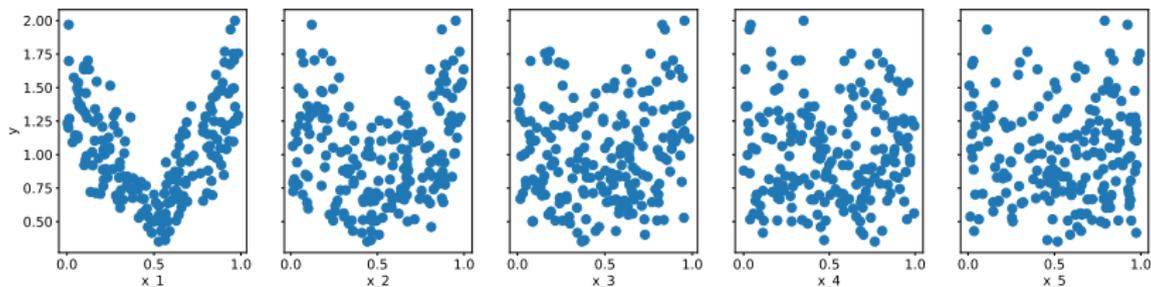


$$\begin{cases} f(\mathbf{x}) = \prod_{i=1}^d \frac{|4x_i - 2| + c_i}{1 + c_i}, \\ \text{with } \mathbf{c} = (1, 2, 5, 10, 20, 50, 100, 200, 500, 500, \dots, 500), \end{cases}$$

with  $d = 400$ , and  $\mathbf{X} \sim U[0, 1]^d$ .

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

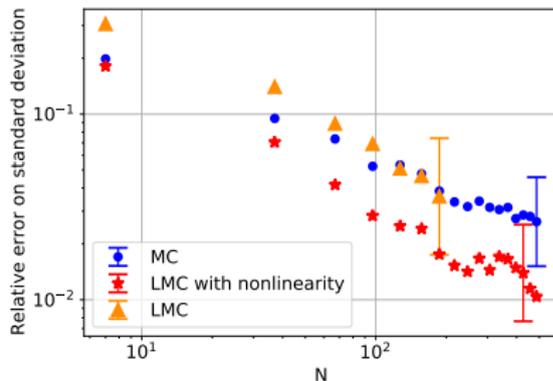
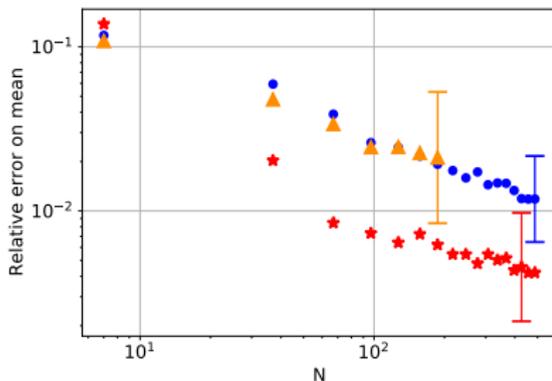


$$\begin{cases} f(\mathbf{x}) = \prod_{i=1}^d \frac{|4x_i - 2| + c_i}{1 + c_i}, \\ \text{with } \mathbf{c} = (1, 2, 5, 10, 20, 50, 100, 200, 500, 500, \dots, 500), \end{cases}$$

with  $d = 400$ , and  $\mathbf{X} \sim U[0, 1]^d$ .

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**.

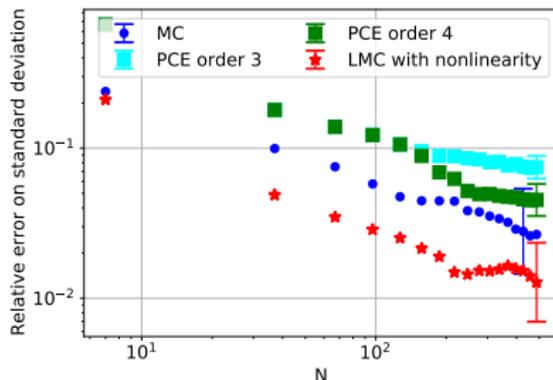
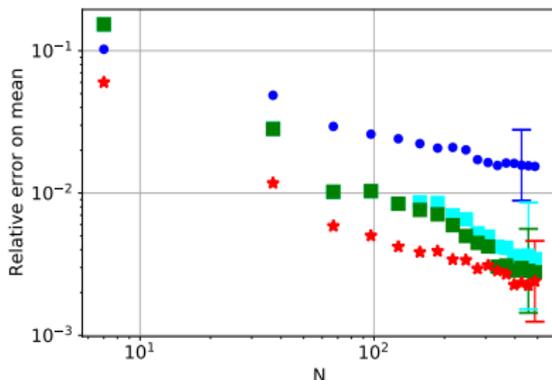
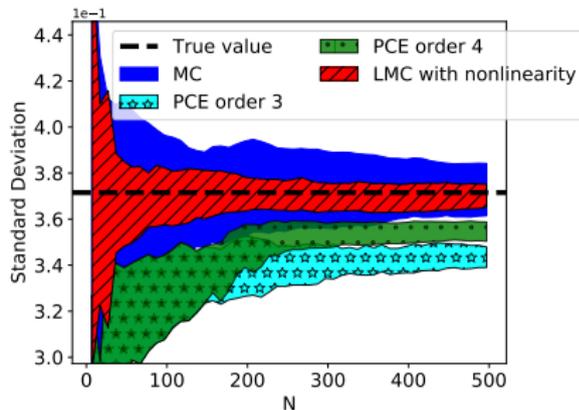
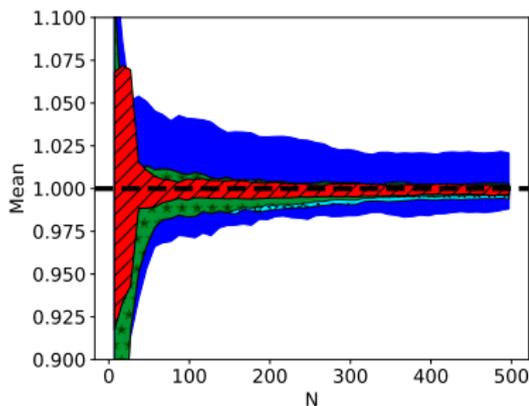
However we can instead fit a modified Lasso model  $\tilde{f}(\mathbf{x}) = \beta \cdot \phi(\mathbf{x})$ , with  $\phi(\mathbf{x}) = |x - 0.5|$ .



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

## Comparison to PCE

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



## 1. Motivation for High-Dimensional UQ: Characterisation of Spent Nuclear Fuel

### Current Methods and Shortcomings

Simple MC

Surrogate Models

## 2. New method: Lasso Monte Carlo

Multilevel Monte Carlo

Lasso Regression

## 3. Benchmarks

## 4. Conclusion

- LMC converges up to **5 times faster than simple MC!** I.e. same results with 20% of the computing resources.
- LMC is advantageous for a variety of nonlinear problems, **including UQ for spent nuclear fuel.**
- Given a set of simulations  $\mathbf{x}_1, f(\mathbf{x}_1), \mathbf{x}_2, f(\mathbf{x}_2), \dots, \mathbf{x}_N, f(\mathbf{x}_N)$ , 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:
  - mild conditions on  $f$
  - an optimal choice of regularisation parameter  $\lambda$  (chosen empirically so far).

An aerial photograph of a university campus, likely the University of Zurich, situated along a river. The campus features several large, modern buildings with flat roofs and a prominent circular structure. The surrounding landscape is lush green with rolling hills and fields, some of which are yellow, possibly indicating rapeseed. In the background, there are mountains under a clear blue sky with some light clouds. A semi-transparent white box is overlaid on the left side of the image, containing the text "Thank you for your attention." and "Questions?".

Thank you for your attention.

Questions?

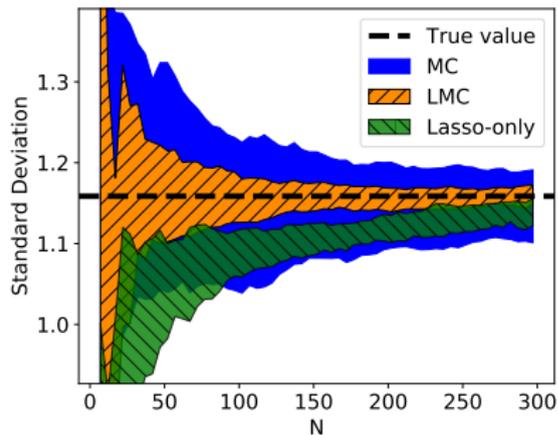
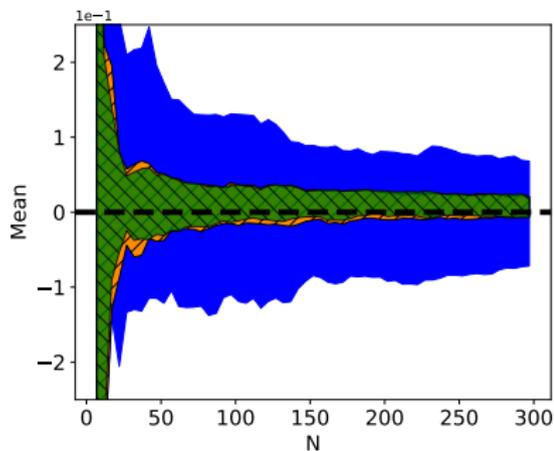
- M. Frey and A. Adelman. Global sensitivity analysis on numerical solver parameters of Particle-In-Cell models in particle accelerator systems. *Computer Physics Communications*, 258: 107577, 2021.
- C. Fiorina, A. Scolaro, D. Siefman, M. Hursin, and A. Pautz. Artificial Neural Networks as Surrogate Models for Uncertainty Quantification and Data Assimilation in 2-D/3-D Fuel Performance Studies. *Journal of Nuclear Engineering*, 1 (1), 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. ISSN 1433-3058. doi: 10.1007/s00521-021-06258-2. URL <https://doi.org/10.1007/s00521-021-06258-2>.
- Michael 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>.
- 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.
- Robert 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>.

## Extra Slides

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

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

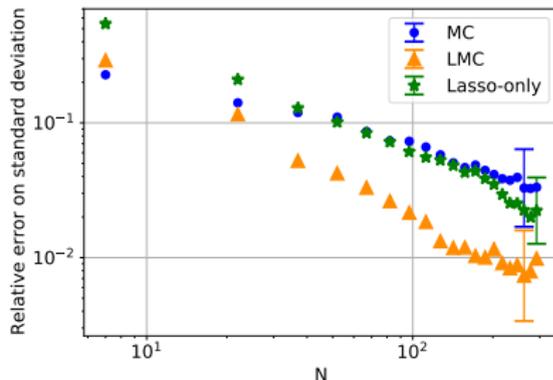
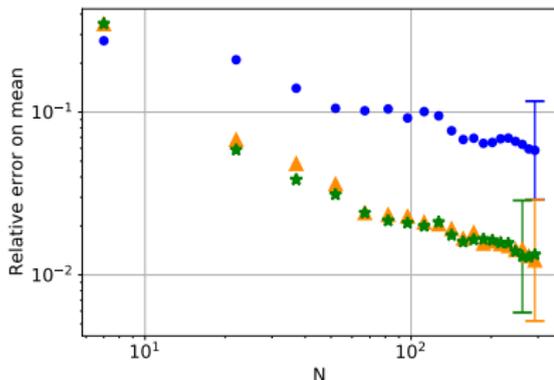
with  $\dim(\boldsymbol{\alpha}) = 400$  and with a normally distributed input  $X \sim \mathcal{N}(0, I_d)$ .



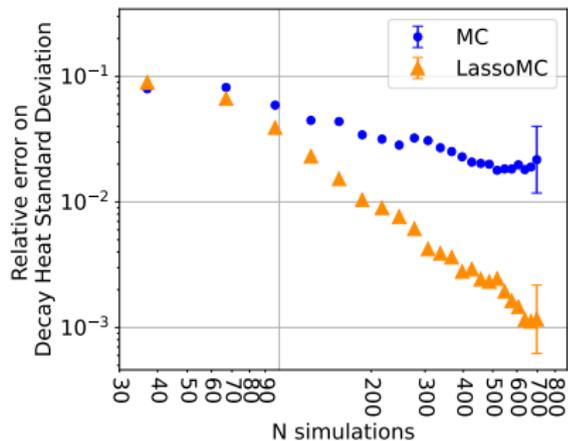
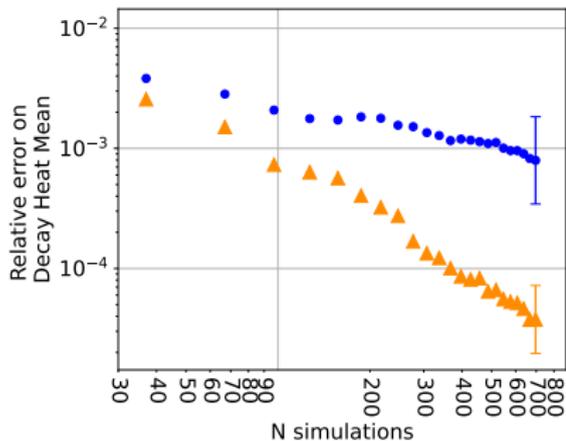
Let  $f$  be a linear function with a large input dimension  $d = 400$ :

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

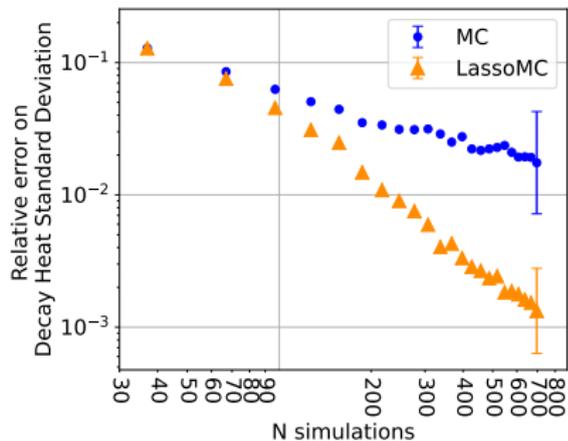
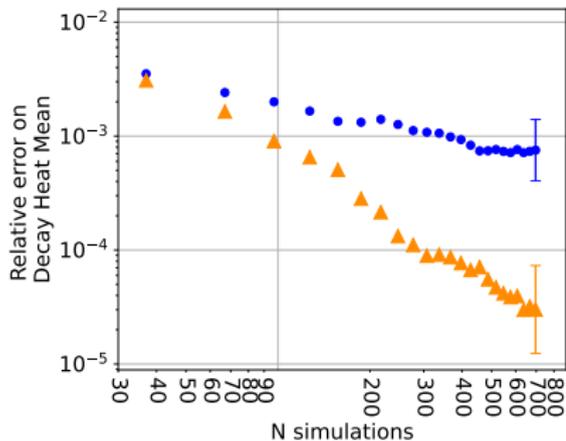
with  $\dim(\boldsymbol{\alpha}) = 400$  and with a normally distributed input  $X \sim \mathcal{N}(0, I_d)$ .



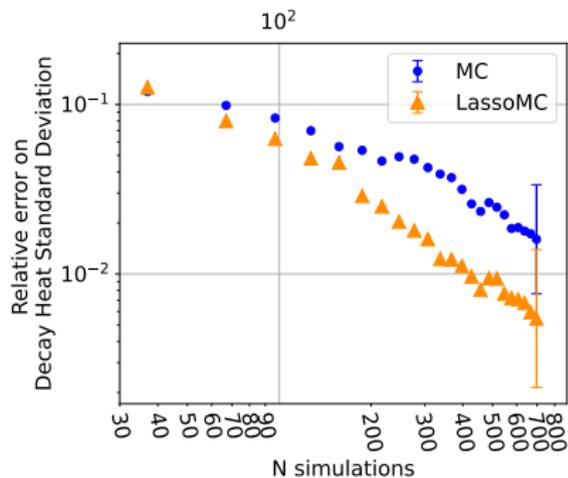
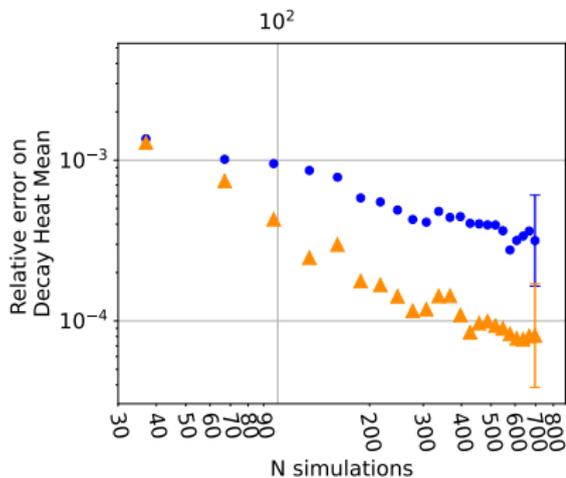
Decay heat prediction at 30 years of cooling:



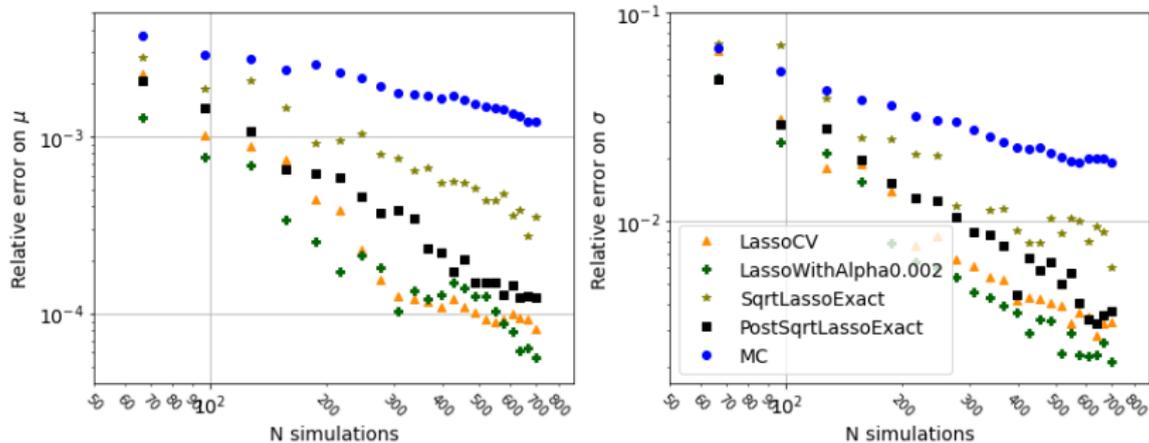
Decay heat prediction at 50 years of cooling:



## U235 concentration at discharge



## Other versions of Lasso regression



**Require:** the probability distribution of the input of  $f(\mathbf{x})$ , the training sets  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  and  $\{\mathbf{z}_1, \dots, \mathbf{z}_M\}$

**Ensure:**  $N \ll M$

- 1: Compute the labels  $f(\mathbf{x}_1), \dots, f(\mathbf{x}_N)$  from the training set.
- 2: Compute the simple MC estimates  $\mu_N, \sigma_N^2$  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  $T_1, T_2, \dots, T_S$  of size  $N \frac{S-1}{S}$  each, and  $S$  correction sets  $C_1, C_2, \dots, C_S$  of size  $n := \frac{N}{S}$  each. Each training set  $T_i$  does not overlap with its corresponding correction set  $C_i$ .
- 4: **for**  $s = 1 \dots S$  **do**
- 5:     Fit a Lasso model  $\tilde{f}_s$  on training set  $T_s$ .
- 6:     Use the surrogate model to compute the labels of the surrogate set  $\tilde{f}_s(\mathbf{z}_1), \tilde{f}_s(\mathbf{z}_2), \dots, \tilde{f}_s(\mathbf{z}_M)$ , and the  $C_s$  correction set  $\tilde{f}_s(\mathbf{x}_{n(s-1)+1}), \tilde{f}_s(\mathbf{x}_{n(s-1)+2}), \dots, \tilde{f}_s(\mathbf{x}_{ns})$ .
- 7:     Combine the  $n$  labels from the correction set and the  $M$  from the surrogate set to compute the two-level estimators  $(\mu_{n,M})_s$  and  $(\sigma_{n,M}^2)_s$ .
- 8: **end for**
- 9: Compute the LMC mean and variance, by averaging out the estimations of each split

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