

mimclib

Abdul-Lateef Haji-Ali

King Abdullah University of Science and Technology (KAUST), Saudi Arabia.
`stochastic_numerics.kaust.edu.sa`

September 7, 2016



Outline

Library Overview

Primer

- Problem

- Monte Carlo (MC)

- Multilevel Monte Carlo (MLMC)

Installation

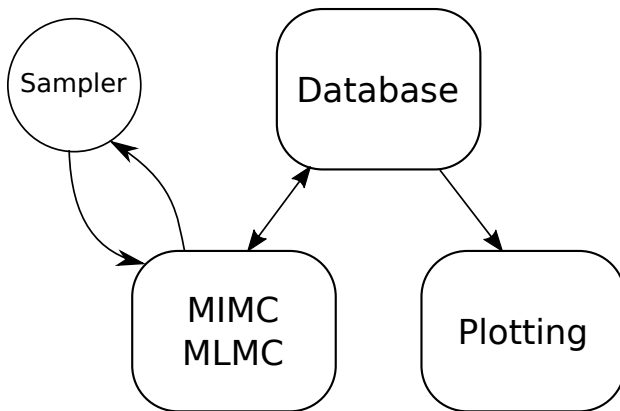
GBM Example

Vision (the ambitious version)

- Provide an **easy to use**, **customizable** and **extendable** open source library for UQ problems, both forward and inverse.
- Multilevel and Multi-index versions of Monte Carlo, Quasi Monte Carlo, Stochastic collocation, Least square projection among others.
- Support parallel computation whenever possible.
- Provide easy to use storage facility.
- Provide easy to customize plotting facility (for common plots).
- Provide easy to run test cases.
- Use Python for easier implementation of most parts of code and use object code (C++ or FORTRAN) for computationally expensive parts.

What has been done, mimclib 0.2.0.dev0

- Multilevel and Multi-index versions of Monte Carlo.
- Provide easy to use storage facility in MySQL.
- Provide easy to customize plotting facility (for common plots).
- Provide easy to run test cases.
- Documentation is still in progress (these slides are part of it).
- Interface is written with the other features in mind.



Why?

- Python
 - Open source. No need for licensing
 - An easy to use programming language. Familiar to MATLAB users (Especially with `numpy` and `matplotlib`)
 - Can call object code for computationally expensive parts, e.g., samplers.
- MySQL
 - Relatively easy data modifications and querying.
 - Allows asynchronous access which is ideal for parallel computation.
 - Allows remote access.
 - Optimal storage and data querying.

The central question

Given an \mathbb{R}^n -valued random variable, \mathbf{X} with PDF $f(\mathbf{x}) : \mathbb{R}^n \rightarrow [0, \infty)$ and a function, $g : \mathbb{R}^n \rightarrow \mathbb{R}$, assume that we are interested in computing the quantity

$$\mathbb{E}[g(\mathbf{X})] = \int_{\mathbb{R}^n} g(\mathbf{x})f(\mathbf{x})d\mathbf{x}.$$

The central question

Given an \mathbb{R}^n -valued random variable, \mathbf{X} with PDF $f(\mathbf{x}) : \mathbb{R}^n \rightarrow [0, \infty)$ and a function, $g : \mathbb{R}^n \rightarrow \mathbb{R}$, assume that we are interested in computing the quantity

$$\mathbb{E}[g(\mathbf{X})] = \int_{\mathbb{R}^n} g(\mathbf{x})f(\mathbf{x})d\mathbf{x}.$$

Possible difficulties:

- PDF, f , is inaccessible, only (approximate) samples of \mathbf{X} can be generated.
 - E.g., \mathbf{X} is the solution of an SDE.
- Dimension, n , is large or even infinite, hence the integral is expensive to approximate.
 - E.g., expansion of a random field.

Setup

Our objective is to build an estimator $\mathcal{A} \approx \mathbb{E}[g(\mathbf{X})]$ with **minimal work** where

$$P(|\mathcal{A} - \mathbb{E}[g(\mathbf{X})]| \leq \text{TOL}) \geq \epsilon$$

for a given accuracy TOL and a given confidence level determined by $0 < \epsilon < 1$.

Setup

Our objective is to build an estimator $\mathcal{A} \approx \mathbb{E}[g(\mathbf{X})]$ with **minimal work** where

$$P(|\mathcal{A} - \mathbb{E}[g(\mathbf{X})]| \leq \text{TOL}) \geq \epsilon$$

for a given accuracy TOL and a given confidence level determined by $0 < \epsilon < 1$.

Instead, we impose the following, more restrictive, two constraints:

Bias constraint: $|\mathbb{E}[\mathcal{A} - g(\mathbf{X})]| \leq (1 - \theta)\text{TOL},$

Statistical constraint: $P(|\mathcal{A} - \mathbb{E}[\mathcal{A}]| \leq \theta\text{TOL}) \geq \epsilon.$

For some tolerance splitting, $\theta \in (0, 1)$.

Setup

Our objective is to build an estimator $\mathcal{A} \approx \mathbb{E}[g(\mathbf{X})]$ with **minimal work** where

$$P(|\mathcal{A} - \mathbb{E}[g(\mathbf{X})]| \leq \text{TOL}) \geq \epsilon$$

for a given accuracy TOL and a given confidence level determined by $0 < \epsilon < 1$.

Instead, we impose the following, more restrictive, two constraints:

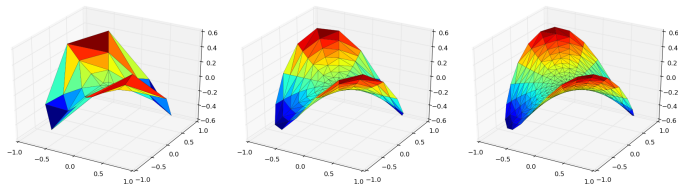
Bias constraint: $|\mathbb{E}[\mathcal{A} - g(\mathbf{X})]| \leq (1 - \theta)\text{TOL},$

Statistical constraint: $\text{Var}[\mathcal{A}] \leq \left(\frac{\theta \text{TOL}}{\Phi^{-1}\left(\frac{\epsilon+1}{2}\right)} \right)^2.$

For some tolerance splitting, $\theta \in (0, 1)$. Assuming (at least asymptotic) normality of the estimator, \mathcal{A} . Here, Φ^{-1} is the inverse of the standard normal CDF.

Numerical approximation

Notation: g_ℓ for $\ell \in \mathbb{N}$ is the approximation of g calculated based on discretization parameters $\mathbf{h}_\ell = (h_{\ell,i})_{i=1}^d$.



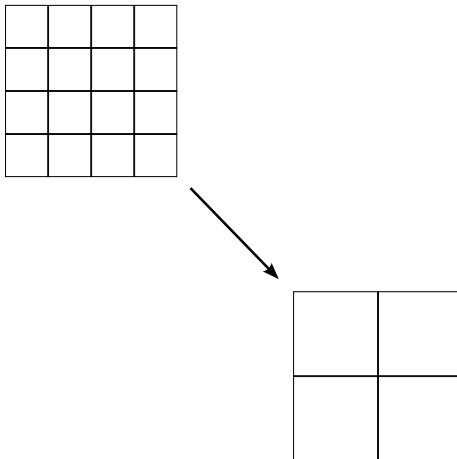
Monte Carlo

The simplest (and most popular) estimator is the Monte Carlo estimator

$$\mathcal{A}_{\text{MC}}[g_L; M] = \frac{1}{M} \sum_{m=1}^M g_L(\mathbf{x}^{(m)}),$$

for a given level, L , and number of samples, M , that we can choose to satisfy the error constraints and minimize the work.

MLMC main idea: Variance reduction



Multilevel Monte Carlo (MLMC)

(Heinrich, 1998) and (Giles, 2008)

Observe the telescopic identity

$$\mathbb{E}[g] \approx \mathbb{E}[g_L] = \mathbb{E}[g_0] + \sum_{\ell=1}^L \mathbb{E}[g_\ell - g_{\ell-1}]$$

Multilevel Monte Carlo (MLMC)

(Heinrich, 1998) and (Giles, 2008)

Observe the telescopic identity

$$\mathbb{E}[g] \approx \mathbb{E}[g_L] = \mathbb{E}[g_0] + \sum_{\ell=1}^L \mathbb{E}[g_\ell - g_{\ell-1}] = \sum_{\ell=0}^L \mathbb{E}[\Delta_\ell g],$$

where

$$\Delta_\ell g = \begin{cases} g_0 & \text{if } \ell = 0, \\ g_\ell - g_{\ell-1} & \text{if } \ell > 0. \end{cases}$$

Multilevel Monte Carlo (MLMC)

(Heinrich, 1998) and (Giles, 2008)

Observe the telescopic identity

$$\mathbb{E}[g] \approx \mathbb{E}[g_L] = \mathbb{E}[g_0] + \sum_{\ell=1}^L \mathbb{E}[g_\ell - g_{\ell-1}] = \sum_{\ell=0}^L \mathbb{E}[\Delta_\ell g],$$

where

$$\Delta_\ell g = \begin{cases} g_0 & \text{if } \ell = 0, \\ g_\ell - g_{\ell-1} & \text{if } \ell > 0. \end{cases}$$

Then, using Monte Carlo to approximate each difference independently, the MLMC estimator can be written as

$$\mathcal{A}_{\text{MLMC}}[g; L] = \sum_{\ell=0}^L \mathcal{A}_{\text{MC}}[\Delta g_\ell; M_\ell].$$

Main idea: variance reduction using cheaper approximations.

Optimal number of samples

Given the following estimates

$$V_\ell = \begin{cases} \text{Var}[g_0] & \ell = 0, \\ \text{Var}[g_\ell - g_{\ell-1}] & \ell \geq 1, \end{cases}$$

$$W_\ell = \begin{cases} \text{Work of a single sample of } g_0 & \ell = 0, \\ \text{Work of a single sample of } (g_\ell - g_{\ell-1}) & \ell \geq 1. \end{cases}$$

Then, using simple Lagrangian optimization of the total work subject to the variance constraint then, we can obtain the optimal number of samples

$$M_\ell \approx \left(\frac{\theta \text{TOL}}{\Phi^{-1}\left(\frac{\epsilon+1}{2}\right)} \right)^{-2} \sqrt{\frac{V_\ell}{W_\ell}} \left(\sum_{i=0}^L \sqrt{V_i W_i} \right), \quad \text{for } 0 \leq \ell \leq L.$$

Assumptions

Assumption MC1 (Bias): $|E[g - g_\ell]| \lesssim \exp(-w\ell),$

Assumption MC2 (Work): $\text{Work}[g_\ell] \lesssim \exp(d\gamma\ell),$

Assumption MLMC3 (Variance): $\text{Var}[g_\ell - g_{\ell-1}] \lesssim \exp(-s\ell)$

for all $\ell \in \mathbb{N}$ and positive constants γ, w, s .

Assumptions

Assumption MC1 (Bias): $|E[g - g_\ell]| \lesssim \exp(-w\ell),$

Assumption MC2 (Work): $\text{Work}[g_\ell] \lesssim \exp(d\gamma\ell),$

Assumption MLMC3 (Variance): $\text{Var}[g_\ell - g_{\ell-1}] \lesssim \exp(-s\ell)$

for all $\ell \in \mathbb{N}$ and positive constants γ, w, s .

The optimal work of MLMC is

$$\begin{cases} \mathcal{O}(\text{TOL}^{-2}) & s > d\gamma \\ \mathcal{O}(\text{TOL}^{-2}) (\log(\text{TOL}^{-1}))^2 & s = d\gamma \\ \mathcal{O}\left(\text{TOL}^{-2 - \frac{d\gamma - s}{w}}\right) & s < d\gamma \end{cases}$$

Assumptions

Assumption MC1 (Bias): $|E[g - g_\ell]| \lesssim \exp(-w\ell),$

Assumption MC2 (Work): $\text{Work}[g_\ell] \lesssim \exp(\textcolor{red}{d}\gamma\ell),$

Assumption MLMC3 (Variance): $\text{Var}[g_\ell - g_{\ell-1}] \lesssim \exp(-s\ell)$

for all $\ell \in \mathbb{N}$ and positive constants γ, w, s .

Recall: Optimal cost of Monte Carlo is $\mathcal{O}\left(\text{TOL}^{-2-\frac{\textcolor{red}{d}\gamma}{w}}\right)$

The optimal work of MLMC is

$$\begin{cases} \mathcal{O}(\text{TOL}^{-2}) & s > \textcolor{red}{d}\gamma \\ \mathcal{O}(\text{TOL}^{-2}) (\log(\text{TOL}^{-1}))^2 & s = \textcolor{red}{d}\gamma \\ \mathcal{O}\left(\text{TOL}^{-2-\frac{\textcolor{red}{d}\gamma-s}{w}}\right) & s < \textcolor{red}{d}\gamma \end{cases}$$

Notice: the cost exponentially increases with increasing $\textcolor{red}{d}$.

Continuation MLMC

- To compute M_ℓ we need to find L and find estimates of V_ℓ .

Continuation MLMC

- To compute M_ℓ we need to find L and find estimates of V_ℓ .
- Instead of running for the small required TOL, CMLMC runs a sequence of MLMC realizations, for decreasing tolerances, ending with the required TOL.
- In each step, estimates of V_ℓ are generated using a Bayesian setting which uses **Assumption MLMC3** coupled with the generated samples to produce good estimates even with a small number of samples.
- The value of L is also chosen in each step to minimize the work. This allows choosing a better splitting parameter, θ .
- CMLMC does not have to reuse samples between iterations, ensuring an unbiased estimator for level L approximation.

mimclib showcase

- Code required for basic MLMC run.
- Show single run of `mimclib`.
- Show plots in pdf.
- Show database in mysql.

Installation

- Prerequisites: gcc (supporting c++11), python2.7, pip, mysql-server, mysql-client.
- First step:

```
> git clone \
    https://github.com/StochasticNumerics/mimclib.git
> cd mimclib
> make pip
```
- Note: to update to latest version

```
> git pull
```
- Done! Sort of.

Setting up the database

- Make sure mysql-server, mysql-client, and libmysqlclient-dev are installed on your data server.

- Create mimclib database on data server

```
> python -c 'from mimclib.db import MIMCDatabase ; \
    print MIMCDatabase().DBCreationScript();' \
| mysql -u root -p
```

- Create database user for mimclib (hint: you can use local username)

```
> mysql -u root -p
mysql> CREATE USER 'USER'@'%' \
    IDENTIFIED BY 'password';
mysql> GRANT ALL PRIVILEGES ON mimc.* TO \
    'USER'@'%' WITH GRANT OPTION;
```

Other useful MySQL commands

- Change password

```
> mysql -u USER -p
mysql> SET PASSWORD FOR 'USER'@'%' \
      = PASSWORD('newpassword');
```

MIMC Database

tbl_runs		
run_id	int	[PK]
creation_date	DATETIME	
TOL	REAL	
done_flag	int	
dim	int	
tag	VARCHAR(128)	
params	mediumblob	
comment	TEXT	

tbl_data		
data_id	int	[PK]
run_id	int	[U, FK]
TOL	REAL	
bias	REAL	
stat_error	REAL	
creation_date	DATETIME	
totalTime	REAL	
Qparams	mediumblob	
userdata	mediumblob	
iteration_idx	int	[U]

tbl_lvls		
data_id	int	[U, FK]
lvl	text	
lvl_hash	varchar(35)	[U]
E1	REAL	
V1	REAL	
W1	REAL	
T1	REAL	
M1	int	
psums_delta	mediumblob	
psums_fine	mediumblob	

Legend

[FK] Foreign Key

[PK] Primary key

[U] Unique constraint

Created by SQL::Translator 0.11021

Installation for Debian/Ubuntu

- ```
> git clone \
 https://github.com/StochasticNumerics/mimclib.git
> cd mimclib
> ./mimc_install.sh
```
- Updates all packages on the system
- Installs all prerequisites
- Clones and installs the library
- Creates a database with the current user and no password.
- Standard GBM Example is ready to run.

## Problem setup: Geometric Brownian Motion

Given  $S(0)$ ,  $\mu$  and  $\sigma$ , define

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t)$$

Euler-Maruyama discretization

$$\begin{aligned} S_{n+1} &= S_n + \mu S_n \Delta t + \sigma S_n \Delta W_n \\ &= S_n (1 + \mu \Delta t + \sigma \Delta W_n) \end{aligned}$$

We can find (using Itô integral and a log transformation)

$$\begin{aligned} \mathbb{E}[S(t)] &= S(0)e^{\mu t} \\ \text{Var}[S(t)] &= (S(0))^2 e^{2\mu t} (e^{\sigma^2 t} - 1) \end{aligned}$$

## Numerical ex.: PDE with random coeffs., [H-ANTT, 2015]

We solve a 3D elliptic PDE with random coefficient and forcing term

$$\begin{aligned} -\nabla \cdot (e^{\kappa(\mathbf{x}, \mathbf{y})} \nabla u(\mathbf{x}, \mathbf{y})) &= 1 \quad \text{in } \mathcal{D} = [0, 1]^3 \\ u(\mathbf{x}, \mathbf{y}) &= 0 \quad \text{on } \partial\mathcal{D}, \end{aligned}$$

where

$$\kappa(\mathbf{x}, \mathbf{y}) = \sum_{k=0}^n y_k \sqrt{3} \exp(-k) \Psi_k(\mathbf{x}).$$

Here  $\{y_k\}_{k=1}^n$  are i.i.d.  $\mathcal{U}[-1, 1]$ . Moreover,  $\Psi_k$  is a tensorizable cosine/sine basis. Quantity of interest is:

$$g = \left(2\pi\sigma^2\right)^{-\frac{3}{2}} \int_{\mathcal{D}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_0\|_2^2}{2\sigma^2}\right) u(\mathbf{x}) d\mathbf{x},$$

for  $\mathbf{x}_0 \in \mathcal{D}$  and  $\sigma > 0$ .

Using 2<sup>nd</sup> order Finite Difference Method with GMRES linear solver, for this problem we have  $\gamma \approx 1$ ,  $w = 2$  (isotropic case).

# The end

- Slides can be found on GitHub under “docs”  
<https://github.com/StochasticNumerics/mimclib>.
- You can post questions in the “Issues” page.
- Next time
  - Next, next time: Multi-index Monte Carlo applied on PDEs in 3D.