GPU accelerated Monte Carlo sampling for SPDEs

Nikolay Shegunov, Peter Armianov, Atanas Semerdjiev, Oleg Iliev

Faculty of Mathematics and Informatics, Sofia University, Bulgaria
ITWM Fraunhofer, Kaiserslautern, Germany

{nshegunov, parmianov, asemerdjiev}@fmi.uni-sofia.bg
{oleg.iliev}@itwm.fraunhofer.de

Abstract

Monte Carlo sampling methods is a broad class of computational algorithms, that rely on repeated random sampling to obtain numerical results. The idea of such algorithms is to introduce randomness to solve problems, even in the deterministic case. Such algorithms are often used in physical and mathematical problems and are most useful when it is difficult or impossible to use other approaches due to limitations, such as cost of performing experiment or inability to take direct measures. The problems typically require solving a stochastic partial differential equations (SDPEs), where an uncertainty is incorporated in to the model. For example: as an input parameter, or initial boundary condition. Extensive efforts have been devoted to the development of accurate numerical algorithms, so that simulation predictions are reliable in since that the numerical errors are well understood and under control for practical problems. Multilevel Monte Carlo (MLMC) is a novel idea. Instead of sampling from the true solution, a sampling is done at different levels. Such approach is beneficial in terms of convergence rate. However for practical simulations a large number of problems has to be solved, with huge number of unknowns. Such computational restrictions naturally leads to challenging parallel algorithms. To overcome some of the limitations, here we consider a parallel implementation of MLMC algorithm for a model SPDE, that uses GPU acceleration for the permeability generation.

1 Introduction

Many mathematical models describing industrial problems are subject to uncertainty due to some limitation. Incorporating the uncertainty typically leads to more accurate representation of the world. However it is in the cost of solving a statistical problem, for example one can consider saturated flow in subsurface, or heat condition in Metal Matrix Composites. Such stochastic models require enormous computational effort, thereby requiring new fast algorithms to facilitate that need. In this paper we consider a scalar elliptic SPDE describing single phase flow throughout heterogeneous porous media. Although the approach here is not limited to this problem, we aim at computing the mean flux throughout saturated porous media with prescribed pressure drop and known distribution of the random coefficients.

One of the preferred and powerful methods for solving SPDE is Multilevel Monte Carlo algorithm (MLMC). The algorithm exploits a combination of fewer expensive computations with a plenty of cheap ones to compute expected values at significantly lower computational cost then standard Monte Carlo. A key component is the selection of the different levels of combination. Many different approaches exist. In [1] authors use the number of therms of Karhunen-Loewe expansion in order to define coarser levels. In [4], similarly to our approach, the coarser levels are defined on coarser grids via averaging of the coefficients in the PDE. Here we construct the levels by renormalization. For details we reefer to [5, 3]. For generation of permeability (random field) we use circulant embedding algorithm [7].

Due to the rapidly advancing area of computer science, methods based on Monte Carlo sampling are of great interest. Such methods are suitable for parallelization and can compute the expected value in reasonable time. For realistic simulations usually a HPC implementation is need. Here we investigate the possible benefits of generating permeability on GPUs and study the reduction of time in overall computation of MLMC.

2 Model problem

In order to test the overall performance of MLMC in combination with CUDA generated permeability field, we consider simple model problem in a unit cube domain, steady state single phase flow in random porous media. This problem, illustrates well the challenges in solving stochastic PDEs.

$$-\nabla \cdot [k(x,\omega)\nabla p(x,\omega)] = 0 \text{ for } x \in D = (0,1)^d, \omega \in \Omega \tag{1}$$

Subject to boundary conditions:

$$\begin{aligned} p_{x=0} &= 1 \\ p_{x=1} &= 0 \\ \partial_n p &= 0 \text{ on other boundaries,} \end{aligned} \tag{2}$$

with dimension $d = \{2, 3\}$, pressure p, scalar permeability k, and random vector ω. The quantify of interest is the mean (expected value) of the total flux through the unit cube:

$$Q(x, \omega) := \int_{x=0} k(x, \omega) \partial_n p(x, \omega) dx \tag{3}$$

Both the coefficient $k(x, \omega)$ and the solution $p(x, \omega)$ are subject to uncertainty, characterized by the random vector ω in a properly defined space.

Solving this equation can be broken into three sub-problems: **generating permeability (random field), solving the deterministic problem** and **reducing the variance with MLMC method**. We briefly discuss each of them.

Generating permeability field

Generating permeability fields is essential problem in solving the SPDE. Here we consider a practical covariance proposed by [8]:

$$C(x, y) = \sigma^2 exp(-||x - y||_p / \lambda),\ p = 2 \tag{4}$$

Where $|| \cdot ||_p$ denotes the l_p norm. in R^d.
which satisfies:

$$\begin{aligned} &E[K(x, .)] = 0, \\ &E[K(x, .), K(y, .)] = C(x - y) = C(y - x) \\ &\text{for } x, y \in D \text{ and } K(x, \omega) = log(k(x, \omega)) \end{aligned}$$

Several approaches has been developed of generating random permeability fields applicable to flow simulations. Here we use an algorithm based on forward and inverse Fourier transform over circulant covariance matrix to generate permeability. Realization of such field is governed by two parameters: standard deviation σ and correlation length λ. More details can be found in the papers [2] and [7].

Solving the deterministic problem

Literature provides different numerical schemes for solving PDEs. In [2], an Multi-scale Finite element methods is used. Here for solving the elliptic PDEs corresponding to each realization of permeability field, we use *finite volume method* on a cell centered grid. This method is mass conservative. More details can be found in [1, 3].

Variance reduction

We shortly recall the idea proposed in [1].
Let $\{M_l : l = 0 \dots L\} \in N$ be increasing sequence of numbers called levels, with corresponding quantities $\{Q_{M_l}\}_{l=0}^{L}$, and $s \geq 2$ be coarsening factor, such that we have $M_l = sM_{l-1}$, for $l = 0 \dots L$. Defining $Y_l = Q_{M_l} - Q_{M_{l-1}}$ and setting $Y_0 = Q_{M_0}$, we can use telescopic sum to write the following identitity for the expected value E

$$E[Q_M] = E[Q_{M_0}] + \sum_{l=1}^{L} E[Q_{M_l} - Q_{M_{l-1}}] = \sum_{l=0}^{L} E[Y_l] \tag{5}$$

The expectation on the finest level is equal to the expectation on the coarsest level plus sum of corrections, i.e. differences in the expectations on each pair of consecutive levels. The terms in (5) are approximated using standard MC independent estimators, with N_l samples. For the mean square error we have:

$$e(\widehat{Q}_{M,N}^{ML})^2 = E[(\widehat{Q}_{M,N}^{ML} - E[Q])^2] = \sum_{l=0}^{L} N_l^{-1} V[Y_l] + (E[Q_M - E[Q])^2 \tag{6}$$

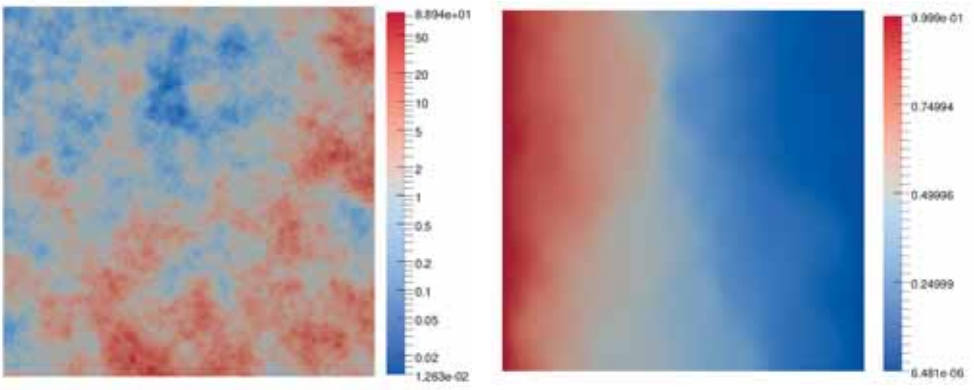
Our goal is to have:

$$e(\widehat{Q}_{M,N}^{ML})^2 = \sum_{l=0}^{L} N_l^{-1} V[Y_l] + (E[Q_M - E[Q])^2 \leq 2\varepsilon^2 \tag{7}$$

Denote $v_l = V[Y_l]$, and let t_l be the mean time computing difference Y_l once, and $T = \sum_{l=0}^{L} n_l t_l$ be the total time for the computation. Minimizing T under the above constraint, with Lagrangian multipliers and turning it to integer value gives us:

$$n_l = \lceil \alpha \sqrt{(v_l/t_l)} \rceil \text{ with Lagrangian multiplier } \alpha = \frac{1}{\epsilon^2} \sum_{l=0}^{L} \sqrt{(v_l/t_l)} \tag{8}$$

To define the levels, the resolution of the discretization is used, such that the number of square cells in $D = 2$ and cubic cells in $D = 3$ are exact power of 2. Thus on the finer level we have 4 times more cells than on the coarser level for $D = 2$, and 8 times more for $D = 3$. To approximate the random field, we employ heuristic technique, witch combines each 4 cells in $2D$ into one by combination of simple arithmetic, geometric and harmonic average. For details we refer to [3].



(a) Permeability

(b) Solution

Figure 1: Generated permeability and corresponding solution

3 Simulation results

The computational time used by the setup of the generation of 100 permeability fields with $\sigma = 2.0$ and $\lambda = 0.2$ is shown on figure 2. Our underling grid is in 2D with size, approximately $1.7 * 10^7$. As it can be observed, the speed up that is achieved using only CPU cores for computation at all of the steps is significant up to around 8 cores. At this point it is approximately up to 6.755 times faster than calculation on a single core. Using more than 8 cores efficiency gradually decreases and with 24 cores the speed up achieved is approximately 15 times. The GPU generation is much faster: using single GPU, generation time is comparable to 11 CPU cores, and on 2 GPUs - approximately 22 CPU cores. This is expected since permeability generation is mainly a forward Fourier transform over a matrix, followed by multiplication with random numbers for each element and then inverse Fourier transform.

Table 1 and 2 show the comparison of 3 level MLMC implementation using only CPU versus implementation, that uses GPU for permeability generation. The presented results are averaged over 10 runs of the algorithm.

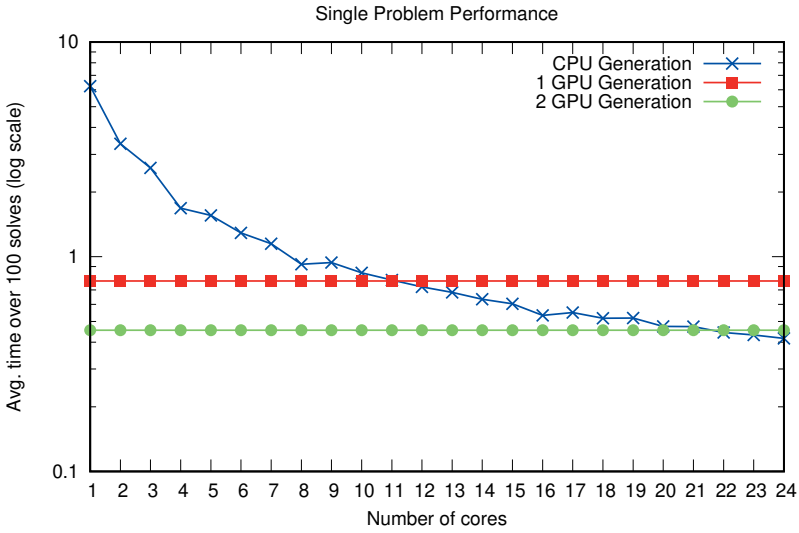


Figure 2: Average times for generation single permeability field over 100 samples.

The fine grid is of size $2^{10} * 2^{10}$ and permeability generation parameters are $\sigma = 1.5$, $\lambda = 0.1$. Each problem is computed by own CPU core and all use shared GPU, thus in a given moment of the execution 12 or 24 concurrent calls to the GPUs may exist. This is more taxing situation for the GPU compared to a distribution where a group of processes solves a single problem. In table 1 one can observe that MLMC calculated on single GPU and 12 CPU cores is faster than same algorithm using only CPU cores and the execution of the generation step on the different levels is approximately 2 times faster while the execution of the same MLMC implementation on single GPU and 24 CPU cores shows that the overall time is slower than execution only on the same number of CPU cores. The performance is significantly improved with introduction of a second GPU. The generation times are notably lower than the case with a single GPU.

#GPU	E[Q]	Time[s]	RMS	$t_{l_0}^{gen}$[s]	$t_{l_1}^{gen}$[s]	$t_{l_2}^{gen}$[s]
0	1.041	1451.830	0.0028	0.430	0.433	0.518
1	1.039	1370.275	0.0029	0.266	0.163	0.229
2	1.040	1205.470	0.0027	0.042	0.060	0.093

Table 1: MLMC simulation, on 12 cores

#GPU	E[Q]	Time[s]	RMS	$t_{l_0}^{gen}$[s]	$t_{l_1}^{gen}$[s]	$t_{l_2}^{gen}$[s]
0	1.040	792.177	0.0029	0.440	0.537	0.530
1	1.039	836.920	0.0029	0.564	0.421	0.584
2	1.038	633.000	0.0029	0.0758	0.163	0.253

Table 2: MLMC simulation, on 24 cores

The implementation of the algorithm, used for that test is written in C++. The method used for solving the PDE conjugate gradient preconditioned with AMG if provided by DUNE library [9]. For implementation of circulant embedding algorithm the fftw library is used for generation on CPU, and cufft library provided by NVIDIA is used for generation on GPU. The Multilevel Monte Carlo algorithm is implemented with pure MPI. All tests are performed on **HybriLIT**-education and testing cluster, Dubna Russia, on a GPU node with two NVIDIA TESLA K80 GPUs and Intel Xeon E5-2695 v2 processors.

4 Conclusions

Generating random filed using modern GPU accelerated computing is promising possibility, witch can decrease the computational time of the generation step of the MLMC algorithm if the task is small enough to fit in the GPU memory. When the task is larger, however, the execution time using GPU can be larger than using only CPU cores. The idea of GPU acceleration is fairly new in the area of scientific computing and there are not many libraries implemented, using GPU accelerated approach. Further more, HPC systems with GPU enabled nodes are rather expensive and the number of large scientific computing clusters with GPU capabilities is relatively small. On the other hand, in the recent years, some large commercial clusters for GPU computing were built, inspired by the block-chain and neural networks trend. A further investigation on solving large scientific problems on such clusters is an interesting study.

References

[1] Cliffe, K.A., Giles, M.B., Scheichl R., Teckentrup A.L.: *Multilevel Monte Carlo Methods and Applications to Elliptic PDEs with Random Coefficients.* Computing and Visualization in Science **14**(1) (pp. 3-15). Springer(2011)

[2] Mohring, J., Milk, R., Ngo, A., Klein, O., Iliev, O., Ohlberger, M. and Bastian, P.: *Uncertainty Quantification for Porous Media Flow Using Multilevel Monte Carlo.* In International Conference on Large-Scale Scientific Computing (pp. 145-152). Springer(2015)

[3] Iliev O., Mohring J., Shegunov N.: *Renormalization Based MLMC Method for Scalar Elliptic SPDE.* In: International Conference on Large-Scale Scientific Computing (pp. 145-152). Springer(2017)

[4] Blaheta, R., Béreš, M., Domesová, S.: *A study of stochastic FEM method for porous media flow problem.* In: Bris, R., Dao, P. (eds.) Applied Mathematics in Engineering and Reliability (pp. 281-289). 2016

[5] Lunati, I., Bernard, D., Giudici, M., Parravicini, G., Ponzini, G.: *A numerical comparison between two upscaling techniques: non-local inverse based scaling and simplified renormalization.* Advances in Water Resources 24 (pp. 913-929). Elsevier(2001)

[6] Renard, P., De Marsily, G.: *Calculating equivalent permeability: a review.* Advances in Water Resources 20(5) (pp. 253–278). Elsevier(1997)

[7] Graham, G., Kuo F.Y., Nuyens, D., Scheichl, R., Sloan, I.H. *Quasi-Monte Carlo methods for elliptic PDEs with random coefficients and applications.* Journal of Computational Physics (230) (pp. 3668-3694). Elsevier(2011)

[8] Hoeksema, R.J., Kitanidis, P.K.: *Analysis of the spatial structure of properties of selected aquifers.* Water Resources Research 21(4) (pp. 563–572). (1985)

[9] Bastian, P., Blatt, M., Dedner, A., Engwer, C., Klofkorn, R., Ohlberger, M., Sander, O.: *A Generic Grid Interface for Parallel and Adaptive Scientific Computing. Part I: Abstract Framework.* Computing 82(2-3) (pp. 103–119). (2008)