# On the Stochastic/Deterministic Numerical Solution of Composite Deterministic Elliptic PDE Problems*

George Sarailidis[1] and Manolis Vavalis[2]

*Abstract*— **We consider stochastic numerical solvers for deterministic elliptic Partial Differential Equation (PDE) problems. We concentrate on those that are characterized by their multi-domain or/and multi-physics nature. In particular we consider either plain random walk on spheres methods or synergies of conventional deterministic PDE solving methods and traditional probabilistic Monte Carlo approaches. Our main objectives are two. One is to clearly define the context and the practical approach concerning the use of deterministic components that lead to effective numerical solvers for linear deterministic PDEs. The other is the design and implementation of a proof-of-concept computational framework that allows experimentation in order to elucidate the capabilities and identify the emerging computational characteristics of the proposed approaches. A class of model problems in two and three space dimensions are first considered and experimental results are presented and discussed.**

Keywords – Numerical solution of PDEs, MultidDomain MultiPhysics problems, Monte Carlo methods, Random Walk on Spheres.

## I. INTRODUCTION

The Monte Carlo method has the capability to provide approximate solutions to a variety of mathematical problems, not necessarily with probabilistic content or structure, by performing statistical sampling experiments. About a century has been passed since the discovery of methods which based on the Monte Carlo concept provide numerical approximations to Partial Differential Equation (PDE) problems. These methods generate random numbers and by observing certain of their characteristics and behavior are capable of calculating approximations to the solutions. Specifically, it was [35] who first considered the relationship between stochastic processes and parabolic differential equations followed by [9] who proposed numerical procedures for elliptic PDEs while [26] were the fist to dignify this stochastic approach with a name referring to the gambling facilities available at the Monte Carlo city and propose it as a generic term for numerical methods that use sampling of random numbers.

Since then Monte Carlo methods have been commonly, and in fact heavily, used and still are for many important problems. For example the U.S.A. Department of Energy claimed that Monte Carlo simulations have consistently consumed up to a half of their high-performance cycles since the beginning of its super-computing facilities. Nevertheless, they have not be much utilized for linear PDE-based applications. They are generally considered as methods of last resort ideally suitable only for problems either in high dimensions or very complex geometries [23]. It is interesting to point out that the Monte Carlo pioneer Mark Kac's say *"You use Monte Carlo methods until you understand the problem"* several years ago describes accurately how most of us currently view Monte Carlo methods.

PDE problems have been related to Monte Carlo in several ways (see [20] for a recent survey). The famous Feynman-Kac formula for example, establishes an interesting link between PDEs and stochastic processes. Monte Carlo methods has been, and to a great extent still remains, the only computational choice for several non-linear problems while it has been recognized as a good choice for many other computationally difficult non-linear problems. In addition they seem to be a natural choice for any differential equation in which one or more of the terms is a stochastic process, thus resulting in a solution which is itself a stochastic process. This is clearly depicted by the plethora of very recent Monte Carlo based research efforts devoted to numerical solution of such equations commonly known as stochastic differential equations (see for example [37], [1] for time depended problems and [3], [25], [37], [2], [46] for elliptic problems).

As already mentioned even fundamental linear PDEs are strongly related to stochasticity. For example, it is known that diffusion is in fact a form of Brownian motion at microscopic scale. This provided enough motivation to the several attempts to develop and promote Monte Carlo based numerical solvers for time depended PDEs (e.g. [19], [8], [17], [14] and in particular [20]). Linear non-stochastic elliptic boundary value problems are also strongly connected to probability (rigorous measure theory). For example, integrals with respect to certain measure have been recognized as solutions of certain parabolic or elliptic differential equations [9]. It is worth to mention that there are several recent research efforts concerning probabilistic interpretations of harmonicity and of fundamental elliptic PDEs using Brownian motion and stochastic calculus (see [34] and reference therein).

In this paper we restrict our investigation on the effectiveness of Monte Carlo methods for the numerical solution of linear elliptic PDEs and we concentrate on the Poisson equation. It has to be pointed out that although there has been, and currently exist, significant research activity on

*This research has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program Education and Lifelong Learning of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thales. Investing in knowledge society through the European Social Fund.

[2]G. Sarailidis is a free lancer for Computational and Informational Science and Engineering, Athens, GREECE waxplain at gmail.com

[4]M. Vavalis is with the Department of Electrical & Computer Engineering, University of Thessaly, Gklavani 37, 38221 Volos, GREECE mav at uth.gr

this subject, the proposed methods have not attracted so far the expected attention. Furthermore, one can find very few software components[1] that are publicly available and appropriate to support the experimentation which is much needed for elucidating the characteristics and idiosyncrasy of the proposed methods and convincing both researchers and practitioners that can be effectively used for real-world problems as for example the one considered in[41].

In this paper we restrict ourselves on rectangular multi-domains in two and three dimensions and we focus on the implementation of a computational framework that allows easy experimentation with hybrid methods consisting of a combination of mainly two steps:

- **Stochastic pre-processing:** A Monte Carlo-based walk on spheres approach is utilized to decouple the original PDE problem into a set of independent PDE sub-problems.
- **Deterministic solving:** Any of the resulting sub-problems is numerical solved independently by means of selected finite element schemes.

It is our believe that the proposed and implemented framework promotes an interesting new concept in solving deterministic PDEs and not only supports experimentation but it has the potential to become a practical tool too.

The rest of this paper is organized as follows. In section II we present a review of existing approaches for the numerical solution of linear elliptic PDEs using Monte Carlo based methods. We also present the mathematical background and the associated generic algorithm for our stochastic/deterministic solving framework and system and briefly comment on its characteristics. Implementation issues are addressed in section III which are coupled with installation and usage details. A summary of the numerical experiments performed can be found in section IV. Our concluding remarks together with our vision for future research actions are given in section V.

## II. Mathematical Background

### A. Monte Carlo methods for linear elliptic PDES

Model Carlo based stochastic-deterministic hybrid methods are not new. As mentioned above, Currant [9] was the first to mention the use of Monte Carlo methods for solving non-stochastic PDE problems. Nevertheless, it was Muller [31] who based on the, then classified, work of Metropolis [26] first proposed a particular associated numerical scheme. His work followed by others (see for example [13]) has actually modivated researcher to build a special purpose machines [39] and also consider particular applications [38].

Related recent work have recently appeared in the following papers [45], [5], [10], [27], [28], [11] and in particular those in the past decade [22], [24], [17], [15], [30], [29], [12], [32], [18], [7], [37], [42], [33], [43], [41], [44]. This recent excellent work have so far received minimal attention from our scientific community. As it has been observed (e.g. from citation count at scopus) it has not been accepted as widely as it should be according to our believe.

It is interesting to mention that in our study we consider the integration of stochastic solvers in the numerical solution of PDEs only at continuous level. Specifically, we do not consider studies like the one found in [44] which deals with a Monte Carlo method for the numerical solution of the linear system that arises from the discretization of the Poisson equation on a 2-dimensional rectangular domain using the 5-point-star finite difference scheme with uniform discretization step. We believe that approaches may have their value but do not fit into the meta-computing type of solvers we envision in our study. We should mention that (excluding just a few exceptions) most of the related work mentioned so far does not focus on the efficient implementation of Monte Carlo solvers in general and on modern parallel computing systems in particular. It is plausible in general why the multicore available systems have not attracted Monte Carlo methods at least as much as expected[2].

### B. Stochastic/deterministic elliptic PDE solvers

We consider the following elliptic boundary value problem

$$Lu(x) = f(x) \quad x \in \mathcal{D} \subset \mathbb{R}^d, \tag{1}$$

$$Bu(x) = g(x) \quad x \in \partial\mathcal{D}, \tag{2}$$

where $L$ is an elliptic differential operator, $B$ a boundary operator and $d \in \mathbb{N}$. We assume that the regularity conditions for the closed domain $\mathcal{D}$, the operators $L$ and $B$ and the given functions $f(x)$ and $g(x)$ are satisfied. These conditions guarantee the existence and uniqueness of the solution $u(x)$ in $C_2(\mathcal{D} \cap \partial\mathcal{D})$ of problem (1)–(2). We furthermore assume that the domain $\mathcal{D}$ consists of (or can be splitted into) $\mathcal{N}_{\mathcal{D}}$ subdomains, i.e.

$$\mathcal{D} = \cup_{\mu=1}^{\mathcal{N}_{\mathcal{D}}} \mathcal{D}_\mu \tag{3}$$

and that $L_\mu$ and $f_\mu$ are the restrictions of $L$ and $f$ on $\mathcal{D}_\mu$ while $B_\mu$ and $g_\mu$ are the restrictions of $B$ and $g$ on $\partial\mathcal{D}_\mu \cap \partial\mathcal{D}$. We finally define the interface between the two subdomains $\mathcal{D}_\mu$ and $\mathcal{D}_\nu$ as

$$\mathcal{I}_{\mu,\nu} = \partial\mathcal{D}_\mu \cap (\partial\mathcal{D}_\nu \cup \mathcal{D}_\nu) \subset \mathbb{R}^{d-1}, \quad \mu, \nu = 1, \ldots, \mathcal{N}_{\mathcal{D}}. \tag{4}$$

Assuming $\mu \neq \nu$.

Obviously we consider only those interfaces for which we have that $\mathcal{I}_{\mu,\nu} \neq \emptyset$.

It is important to point out that the above generic methodology becomes particularly attractive in several real-world configurations, for example when the restrictions of the elliptic operator $L$ is not the same in all subdomains, when there exist singularity points in some subdomains, when the PDE domain $\Omega$ is complex and can be simplified if decomposed in subdomains ... . In such cases it is very important that one selects the most appropriate local solver tailored to each particular subdomain and the restrictions of the operators and functions on it. Furthermore, the above scheme offers us the possibility of computing the solution

[1] Searching, for example, with "Monte Carlo" as keyword in TOMS BibTeX bibliography results with just 10 items.

[2] http://www.oxford-man.ox.ac.uk/gpuss

**Data**: $i_1, i_2, \ldots, i_N$: the ids of the subdomains in which we wish to compute the solution.
**Result**: $\tilde{u}_\mu$, $\mu = i_1, \ldots, i_N$: computer approximations of the restrictions of the exact solution $u$ in the subdomains $\mathcal{D}_\mu$, $\mu = i_1, \ldots, i_N$.

// PHASE I: Estimate solution on the interfaces
;
**while** $\mathcal{I}_{\mu,\nu} \subset \cup_{j=1}^{N} \partial\mathcal{D}_{i_j}$ **do**
  Select control points $x_i \in \mathcal{I}_{\mu,\nu}$, $i = 1, 2, \ldots, M_{\mu,\nu}$;
  Estimate the solution $u$ at the control points $x_i$ using a Monte Carlo method;
  Calculate the interpolant $u^I_{\mu,\nu}$ of $u\mu, \nu$ using the control points $x_i$;
**end**
// PHASE II: Estimate solution in the subdomains
;
**for** $j = 1, 2, \ldots, N$ **do**
  Solve the PDE problem:;
  $L_{i_j} u_{i_j}(x) = f_{i_j}(x)$ $x \in \mathcal{D}_{i_j}$ ;
  $B_{i_j} u_{i_j}(x) = g_{i_j}(x)$ $x \in \partial\mathcal{D}_{i_j} \cap \partial\mathcal{D}$ ;
  $L_{i_j} u_{i_j}(x) = h_{i_j}(x)$ $x \in \mathcal{D}_{i_j}$ ; // $h_{i_j}(x)$ constructed using the $u^I_{\mu,\nu}$s
**end**

**Algorithm 1:** The Generic Algorithm.

only on selected subdomains that are of particular importance to us.

## III. Implementation and Usage

We start this section with a presentation of our basic implementation[3] of the algorithm described above. We note that this basic implementation may be combined with either the CPU/GPU or with the web services or a combination of them. In the rest of this paper and for the simplicity in the presentation and due to lack of space we only describe the basic implementation.

Initially, the user needs to specify the problem (by programming it in the source code), i.e., the right hand side of the Poisson's equation and the boundary functions, the domain, and the desired partitioning of the domain (used for the parallelization).

Then, the user needs to specify the resolution of the Monte Carlo estimations, i.e., the number of points along an interface between two subdomains whereat the solver should estimate the solution using the walk-on-spheres method, the number of (independent) walks that should be used and ultimately be averaged so as to get an estimation, and the boundary tolerance, i.e., the distance that signifies whether a point is close enough to the boundary so that the solver can assume that the value of the solution on that point is the same as that on the boundary.

[3]Can be found at https://github.com/mvavalis/Hybrid-numerical-PDE-solvers.

The solver runs in parallel the computations for each of the points on the interface.[4]

Our implementation is based on the walk-on-spheres method and follows the approach and the basic idea found in [10] (Section II, pp. 126) and can be summarized as follows. Assume that we want to estimate $u(x_0)$. Let $s$ be the current estimation of the solution, $B(x)$ is the largest ball in the domain centered at point $x$, $q(y)$ is the right hand side of the problem, and $a(d)$ is a function associated with Green's function for the problem, which takes as input the radius of the $B(x)$. One walk requires the following computations.

**step i:** assign $x_0$ to $x$; assign 0 to $s$;
**step ii:** if $x$ is close enough to the boundary, go to step $v$;
**step iii:** find randomly a point $y$ inside $B(x)$, with respect to the density of $B(x)$ (more on this later); assign to s, the sum of the previous value of $s$, plus the product of $q(y)$ multiplied by $a(d)$;
**step iv:** find randomly a point on the surface of $B(x)$, assign this point to $x$; go to step ii;
**step v:** return $s$;

This process is repeated for enough many times, and the mean of the estimations at the end of each process is used as the final estimation. A more detailed exposition is seen in the associated listing in the appendix.

Note that the first step in each walk is accomplished using a quasi-random sequence (not evident in the listing). We believe that the first step determines considerably more than the rest of the steps, the region where the walk takes place; therefore, using a quasi-random sequence for the first step significantly helps us to make a more uniform sampling, which in turn results in faster convergence.

Let us now consider how we find randomly a point $y$ inside $B(x)$, with respect to the density of $B(x)$ (`rand_update_y(x, y, d)`).[5] To calculate the new $y$ we need to calculate a new radius and angle of the vector to add at the vector corresponding to the point $x$.

The probability density function (PDF) of the radius and the angle is: $\rho(r, \theta) = \frac{2r}{\pi d^2} \ln \frac{d}{r}$, and because it is independent of the angle, we can choose an angle uniformly. Now we have to find a new PDF (let's say $\rho(r)$) for the radius: $\rho(r) = \int_0^{2*pi} \rho(r, \theta) d\theta = 2\pi\rho(r, \theta) = \frac{4r}{d^2} \ln \frac{d}{r}$

We can choose a radius using the quantile function of $\rho(r)$, i.e., the inverse of its cumulative distribution function). However, we cannot compute the quantile function analytically, therefore we use the rejection method [36]. [6]

Note that $max_x(PDF(x)) = 4/(e * d)$[7] The rejection method will work as follows.

**step i:** choose uniformly a random $x_1$ in $(0, d)$, and a random $x_2$ in $(0, 4/(e * d))$.

[4]We could parallelize further by running all walks in parallel, but the speed up would not be significant, especially if we take into account the computation costs of the finite-element solver that are to follow.

[5]The details correspond to the two-dimensional case.

[6][MATLAB script (to see the $\rho(r)$): $d = 1; r = 0 : .001 : d; y = (4/d^2) * (r * log(d) - r. * log(r)); plot(r, y);$] The bell like form of $\rho(r)$ means that the rejection method is going to be efficient.

[7]We have $(d/dx)(PDF(x)) = (4/d^2) * (lnd - lnr - 1) = 0 \Leftrightarrow lnr = lnd - lne \Leftrightarrow r = d/e$, that is $max_x(PDF(x)) = max(2 * pi * \rho(r, \theta)) = 2 * pi * \rho(d/e, \theta) = 4/(e * d)$.

**step ii:** check if $(x_1, x_2)$ is below the curve of PDF, that is if $2 * pi * \rho(x1, \theta) < x2$. If it is, we found our radius $x_1$, else go to step i.

*1) Interpolation:* There is a significant difference in our implementation as regards to the interpolation between the 2-dimensional and the 3-dimensional cases. In case of the three dimensions we need 2-dimensional interpolation. As the latter is relatively complex in terms of computation we precompute the interpolants and feed them to the finite-element solver. We use Sintef's Multilevel B-splines Library (MBA[8]) library is used. In particular we [21]. Here, too, the computations for each interpolant are executed in parallel.

In the two dimensional case the computations are relatively simple and we have chosen not to pre-compute the interpolants. Instead, we supply the finite-element solver with the information needed to compute the required interpolants on spot. For the 1-dimensional interpolation we use John Burkardt's SPLINE C++ library.

Clearly, in both cases, the points used for the interpolation are the ones computed in the previous step of the Monte Carlo estimations.

*2) Finite Elements:* The final step in the process is the solution of each sub-problem (corresponding to each sub-domain) using a finite-elements solver. Clearly, the computations for each sub-problem are executed in parallel. We use the state-of-the-art C++ library deal.II[9]. This recently developed and already widely used library [4] offers adaptive finite element solvers of high quality for the numerical solution of partial differential equations.[10]

## IV. Numerical Experiments

### A. 2-dimensional Experiments

We start by considering the rectangular domain $\Omega \equiv [-1, 1] \times [-1, 1]$ and the Poisson equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y), \ \forall (x, y) \in \Omega, \tag{5}$$

where

$$f(x, y) = (1-\pi^2)\left(\sin(\pi x)\sinh(y) + 4\cosh(2x)\cos(2\pi y)\right), \tag{6}$$

subject to the following Dirichlet boundary conditions

$$\begin{array}{ll} u(\pm 1, y) = & \cosh(\pm 2)\cos(2\pi y) \\ u(x, \pm 1) = & \sin(\pi x)\sinh(\pm 1) + \cosh(2x), \end{array} \quad \forall (x, y) \in \partial\Omega. \tag{7}$$

The exact solution of the above problem is given by

$$u(x, y) = \sin(\pi x)\sinh(y) + \cosh(2x)\cos(2\pi y). \tag{8}$$

and as depicted in figure IV-A has rather strong variations along both axis allowing us to qualitative examine the effectiveness of our system. For this, we decompose the PDE domain $\Omega$ into the eight non-overlapping subdomains defined by interface lines drawn at $x_1 = 0$ and $y_1 = -0.5$, $y_2 = 0$ and $y_3 = 0.75$, we solve only the PDE subproblems defined by subdomains $\Omega_{1,0}$, $\Omega_{0,1}$ and $\Omega_{2,1}$.

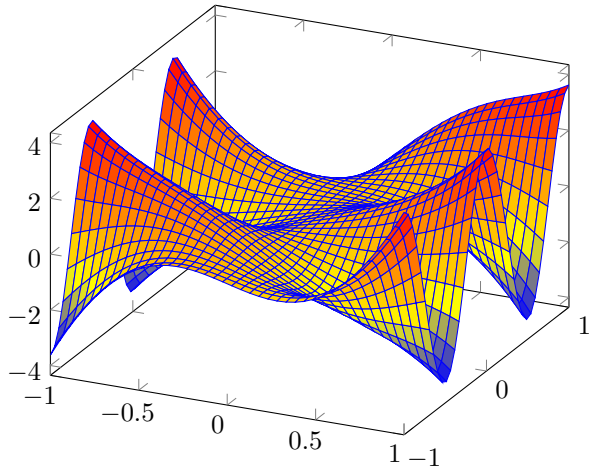


Fig. 1. True solution of the PDE problem defined by (5)–(7).

We note here that besides the functions that define the PDE problem (the right hand sides of the elliptic operator and of the boundary conditions, and perhaps the true solution if it is known) the user needs to give other parameters as those are depicted in the following listing 1 which is in accordance to a graphical user interface we develop. The detailed description of this interface is beyond the scope of this paper.

Listing 1. Listing of the configuration file for the 2D model problem.

```
# ============================

# General Parameters
2  # Number of dimensions
4  # Maximum number of threads

1  # Dimension X length
1  # Dimension Y length

2  # Number of subdomains along dimension X
2  # Number of subdomains along dimension Y

# ============================

# Monte Carlo Parameters
1000          # Number of walks
.0000000001   # Boundary tolerance

# ============================

# Interpolation Parameters
3  # Number of nodes along dimension X
3  # Number of nodes along dimension Y

# ============================

# Conjugate Gradient Parameters
6  # laplace grid refine times
   # the parameters of solver_control()
```

For the numerical solution of the above mentioned sub-problems we have utilized finite element methods from the deal.II library. Different adaptive refinement strategies (h, h and hp) based on local error indicators and error estimators are supported and utilized. In Table I we present various characteristics of these modules. Figure IV-A presents the

[8] http://www.sintef.no/upload/IKT/9011/geometri/MBA/mba-1.1.tgz

[9] http://www.dealii.org/

[10] Our C++ class `LaplaceSolve` is based on class `LaplaceProblem`, implemented in the 4th step of the tutorial, in the documentation of library's version 6.1.0.

quality of the computed solution and the effect of the refinement mechanism. We only present data for a particular sub-problem. The data associated with the other subproblems are similar. Specifically, we present on the y-axis the L2 norm of the error and on the x-axis the refinement level.

TABLE I
THE NUMERICAL CHARACTERISTICS OF THE FINITE ELEMENT MODULES USED AND THE ASSOCIATED NUMBER OF EQUATIONS REQUIRED FOR CONVERGENCE BY THE CONJUGATE GRADIENT ITERATION SCHEME.

| | cycle | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|
| | cells | 4 | 16 | 64 | 256 | 1024 |
| Q1 elements | dofs | 9 | 25 | 81 | 289 | 1089 |
| | CG iterations | 1 | 6 | 23 | 51 | 103 |
| Q2 elements | dofs | 25 | 81 | 289 | 1089 | 4225 |
| | CG iterations | 7 | 31 | 70 | 142 | 286 |

### B. 3-dimensional Experiments

We consider a slightly modified problem than the PDE problem presented in the previous section as follows.

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2} = x+y+z \;\; \forall (x,y) \in \Omega \equiv [-1,1]^3, \quad (9)$$

subject to the Dirichlet boundary conditions

$$u(x,y,z) = g(x,y,z) \;\; \forall (x,y) \in \partial\Omega, \quad (10)$$

where the right hand side function $g$ is selected so that the exact solution of the above problem (9)–(10) is given by the equation

$$u(x,y,z) = \exp(\sqrt{2}\pi x)\sin(\pi(y+z)) + \frac{1}{6}\left(x^3+y^3+z^3\right). \quad (11)$$

and depicted in figure 3.

In addition to the functions defining the problem itself we need as in the 2D case to specify various parameters. The associated configuration file for our experiment is given in listing 2 below. It is worth to point out the similarity of the two configuration files for the 2D and the 3D cases (listings 1 and 2 respectively) allowing to write our programs almost dimension independent.

Listing 2. Listing of the configuration file for the 3D model problem.

```
# ============================

# General Parameters
3  # Number of dimensions
4  # Maximum number of threads

1  # Dimension X length
1  # Dimension Y length
1  # Dimension Z length

2  # Number of subdomains along dimension X
2  # Number of subdomains along dimension Y
2  # Number of subdomains along dimension Z

# ============================

# Monte Carlo Parameters
1000          # Number of walks
.0000000001   # Boundary tolerance
```

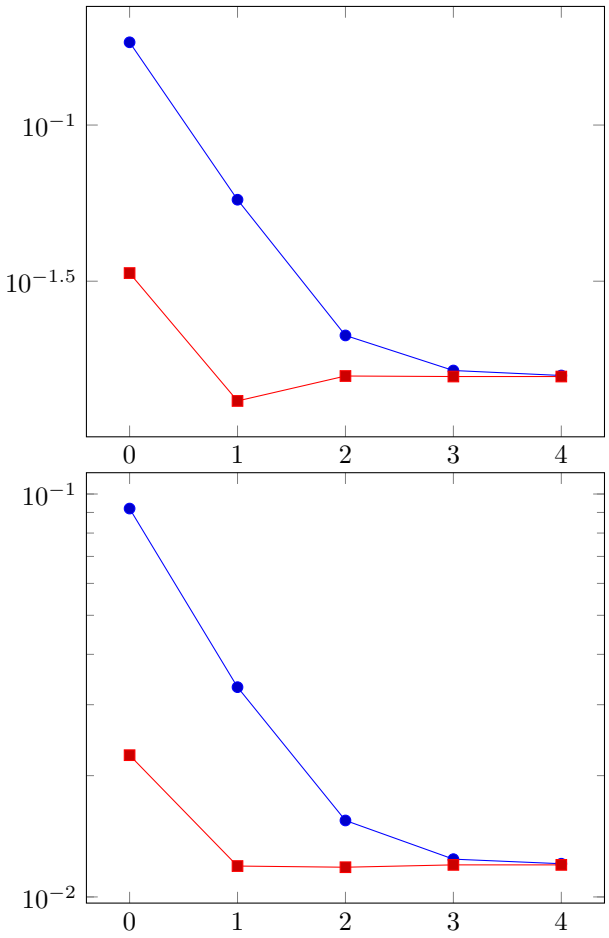


Fig. 2. L2 error in the numerical solution computed by the FE methods on domain $\Omega_{0,1}$ as a function of refinement levels (x-axis) for the Q1 and Q2 elements (top and bottom figures respectively).

```
# ============================

# Interpolation Parameters
2  # Number of nodes along dimension Y
   #      (on the YZ planes)
2  # Number of nodes along dimension Z
   #      (on the YZ planes)

2  # Number of nodes along dimension X
   #      (on the XZ planes)
2  # Number of nodes along dimension Z
   #      (on the XZ planes)

2  # Number of nodes along dimension X
   #      (on the XY planes)
2  # Number of nodes along dimension Y
   #      (on the XY planes)

7  # Number of levels in the hierarchical
   # construction (see MBA::MBAalg() of
   # Multilevel B-spline (MBA) library)

# ============================

# Conjugate Gradient Parameters
6  # laplace grid refine times
   # the params of solver_control()
```

In a similar to the 2-dimensional case we decompose the domain $\Omega$ into 16 non-overlapping subdomains defined by

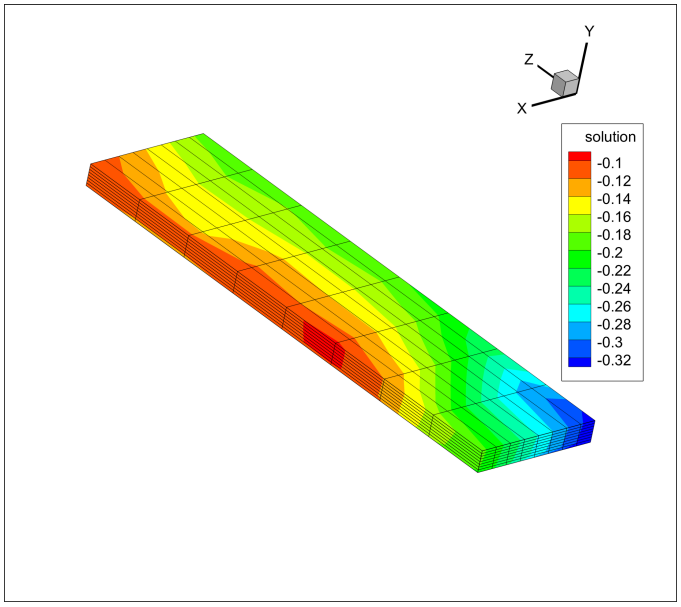


Fig. 3. The computed solution of the global PDE problem defined by (9)–(10).

the interface planes $x_1 = 0$ and $y_1 = -0.5$, $y_2 = 0$, $y_3 = 0.75$ and $z_1 = -0.2$. The computed solution of the global problem is depicted in Figure 3. In general the computational behavior of our method on the 3D problem is similar to the 2D case one. Details presentation of the computational characteristics of our method is beyond the scope of this paper.

Here it is worth to mention that we have also developed basic procedures that automatically combine the local solution into a global one and properly display it using several state-of-the-art scientific visualization tools like the ones found in `http://www.tecplot.com` and which has been used for the Figure 3.

## V. Conclusions and Prospects

Monolithic and such

The objective of our study is to increase our intuition about the proposed algorithm rather than to attempt to prove new results or even provide a computational tool for real world problems. We have developed a system that realizes a meta-computing environment that allow straightforward mixing of diverse deterministic and stochastic modules. This allows us to depart from the traditional monolithic numerical solver development for multi-domain and/or multi-physics PDE problems. Monolithic, hard to developed solvers are replaced by smaller and more flexible cooperative numerical modules (local PDE solvers, interpolants, random walk estimators, etc.) that are highly tuned to the particular characteristics of the subproblems involved.

Our study is by no means complete. In fact we consider it as a first exposition of a wide effort that requires much further investigations. Among the directions one could move is to extended these hybrid methods to more general operators [45] possibly with singularities [15], to operators of higher order [16], [7] to problems with Neumman or mixed boundary conditions [40], [42], [43] to non-rectangular domains and virtually to any problem with known Green's function.

An extension of our work to time depended problems seems not difficult. Our optimism arises from related efforts that already appeared in the literature (see [8], [41], [14]) or others that are currently emerging [6].

We should note that the inherent parallelism in general and on multilevel distributed heterogeneous systems in particular, is of particular importance. Besides the inherent to the Monte Carlo method parallelism the proposed hybrid method enjoys several other parallel processing characteristics. These include, multilevel parallelism and low communication computation ratio. Preliminary numerical data support our above claims while a systematic experimental verification of the above mentioned advances of our methods is under way and will be presented elsewhere.

Finally, we believe that our study is based on a new line of reasoning that provides new intuition about the dynamics of Monte Carlo simulations.

## ACKNOWLEDGMENT

The authors are grateful to Professor George Zouraris for his general discussions on generic Monte Carlo approaches, and Professors Martin Simon and Sylvain Maire for the discussions on random walks on spheres.

## References

[1] C. Alves and A. Cruzeiro. Monte Carlo simulation of stochastic differential systems - a geometrical approach. *Stochastic Processes and their Applications*, 118(3):346–367, 2008.

[2] I. Babuska, F. Nobile, and R. Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 45(3):1005–1034, 2007.

[3] I. Babuska, R. Tempone, and G. E. Zouraris. Solving elliptic boundary value problems with uncertain coefficients by the finite element method: The stochastic formulation. *Computer Methods in Applied Mechanics and Engineering*, 194(12-16):1251–1294, 2005.

[4] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II—A general-purpose object-oriented finite element library. *ACM Trans. Math. Softw.*, 33(4):24–??, 2007.

[5] A. Bignami and E. Cupini. Monte Carlo method for finite difference equations of elliptic type in a multiregion domain. *J. Comput. Appl. Math.*, 8(2):87–92, 1982.

[6] Alexander Bihlo, Ronald D. Haynes, and Emily J. Walsh. Stochastic domain decomposition for time dependent adaptive mesh generation. *J. Math. Study*, 48(2):106–124, 2015.

[7] F. M. Buchmann and W. P. Petersen. An exit probability approach to solving high dimensional dirichlet problems. *SIAM Journal of Scientific Computing*, 28(3):1153–1166, 2006.

[8] J. Carlsson. A backward Monte Carlo method for solving parabolic partial differential equations. *arXiv:math/0010118v1*, 2000.

[9] R. Courant, K. Friedrichs, and H. Lewy. *Über* die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100:32–74, 1928.

[10] J. M. DeLaurentis and L. A. Romero. A Monte Carlo method for Poisson's equation. *J. Comput. Phys.*, 90(1):123–140, 1990.

[11] I. T. Dimov and T. V. Gurov. Estimates of the computational complexity of iterative Monte Carlo algorithm based on Green's function approach. *Mathematics and Computers in Simulation*, 47(2-5):183–199, 1998.

[12] I. T. Dimov and R. Y. Papancheva. Green's function Monte Carlo algorithms for elliptic problems. *Mathematics and Computers in Simulation*, 63(6):587–604, 2003.

[13] K. Edwards and A. Hogg. Methods of improving analogue Monte Carlo solutions of elliptic partial differential equations. 1:664–673, 1967.

[14] R. Farnoosh and M. Ebrahimi. Monte Carlo method via a numerical algorithm to solve a parabolic problem. *Applied Mathematics and Computation*, 190(2):1593–1601, 2007.

[15] J. Given and C. Hwang. Edge distribution method for solving elliptic boundary value problems with boundary singularities. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 68(4 2):461281–461286, 2003.

[16] K. Gopalsamy and B. Aggarwala. Monte Carlo methods for some fourth order partial differential equations. *Z Angew Math Mech*, 50(12):759–767, 1970.

[17] M. Griebel and M. A. Schweitzer. A particle-partition of unity method for the solution of elliptic, parabolic, and hyperbolic PDEs. *SIAM Journal of Scientific Computing*, 22(3):853–890, 2001.

[18] S. Heinrich. The randomized information complexity of elliptic PDE. *Journal of Complexity*, 22(2):220–249, 2006.

[19] S. Hoshino and K. Ichida. Solution of partial differential equations by a modified random walk. *Numerische Mathematik*, 18(1):61–72, 1971.

[20] B. Lapeyre, E. Pardoux, and R. Sentis. *Introduction to Monte Carlo Methods for Transport and Diffusion Equations*. 0xford University Press, 2003. Translated by A. Craig and F. Craig.

[21] S. Lee, G. Wolberg, and S. Shin. Scattered data interpolation with multilevel B-splines. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):228–244, 1997. cited By (since 1996) 175.

[22] R. N. Makarov. Solution of boundary value problems for nonlinear elliptic equations by the Monte Carlo method. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 14(5):453–467, 1999.

[23] M. Mascagni and C. Hwang. $\epsilon$-Shell error analysis for "Walk On Spheres" algorithms. *Mathematics and Computers in Simulation*, 63(2):93–104, 2003.

[24] M. Mascagni, A. Karaivanova, and Y. Li. A quasi-Monte Carlo method for elliptic partial differential equations. *Monte Carlo Methods and Applications*, 7:283–294, 2001.

[25] H. G. Matthies and A. Keese. Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 194(12-16):1295–1331, 2005.

[26] N. Metropolis and S. Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44:335–341, 1949.

[27] G. A. Mikhailov. Recurrent formulae and the Bellman principle in the Monte Carlo method. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 9(3):281–289, 1994.

[28] G. A. Mikhailov. Solving the Dirichlet problem for nonlinear elliptic equations by the Monte Carlo method. *Siberian Mathematical Journal*, 35(5):967–975, 1994.

[29] G. A. Mikhailov and V. L. Lukinov. Probability representations and the Monte Carlo method for solving equations with powers of elliptic operators. *Doklady Mathematics*, 67(3):423–425, 2003.

[30] G. Milstein and M. Tretyakov. The simplest random walks for the Dirichlet problem. *Theory of Probability and its Applications*, 47(1):53–68, 2003.

[31] M.E. Muller. Some continuous Monte Carlo methods for the Dirichlet problem. *The Annals of Mathematical Statistics*, 27(3):569–589, 1956.

[32] R. J. Papancheva, I. T. Dimov, and T. V. Gurov. *A new class of grid-free Monte Carlo algorithms for elliptic boundary value problems*, volume 2542, pages 132–139. 2003.

[33] R. Y. Papancheva. *Parallel realization of grid-free Monte Carlo algorithm for boundary value problems*, volume 3743 of *Lecture Notes in Computer Science*, pages 181–188. Springer, 2006.

[34] N. Privault. Potential theory in classical probability. In U. Franz and M. Schurmann, editors, *Quantum Potential Theory*, volume 4310 of *Lecture Notes in Mathematics*, pages 3–59. Springer, 2008.

[35] L.J.W.S. Rayleigh. On James Bernoullis theorem in probabilities. *Philos. Mag.*, 47:246–251, 1899.

[36] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[37] L. Roman and M. Sarkis. Stochastic Galerkin method for elliptic SPDEs: A white noise approach. *Discrete and Continuous Dynamical Systems - Series B*, 6(4):941–955, 2006.

[38] J. Sacadura and A. Al-Abed. Analysis of time-dependent cylindrical problems using Monte Carlo. *ASME, Transactions, Journal of Heat Transfer (ISSN 0022-1481)*, 105:931–933, 1983.

[39] E. Sadeh and M.A. Franklin. Monte Carlo solution of partial differential equations by special purpose digital computer. *IEEE Transactions on Computers*, 23(4):389–397, 1974.

[40] M. Sadiku and K. Gu. New Monte Carlo method for -Neumann problems. pages 92–95, 1996.

[41] M.N.O. Sadiku, C.M. Akujuobi, S.M. Musa, and S.R. Nelatury. Analysis of time-dependent cylindrical problems using Monte Carlo. *Microwave and Optical Technology Letters*, 49(10):2571–2573, 2007.

[42] N. Simonov. Monte Carlo methods for solving elliptic equations with boundary conditions containing the normal derivative. *Doklady Mathematics*, 74(2):656–659, 2006.

[43] N. Simonov. Random walks for solving boundary-value problems with flux conditions. volume 4310 of *Lecture Notes in Computer Science*, pages 181–188. Springer, 2007.

[44] B. Vajargah and K. Vajargah. Monte Carlo method for finding the solution of Dirichlet partial differential equations. *Applied Mathematical Sciences*, 1(10):453–462, 2007.

[45] J. Vrbik. Monte Carlo simulation of the general elliptic operator. *J. Phys. A: Math. Gen.*, 20(3):2693–2697, 1987.

[46] X. Wan and G. Karniadakis. Solving elliptic problems with non-Gaussian spatially-dependent random coefficients. *Computer Methods in Applied Mechanics and Engineering*, 198(21–26):1985–1995, 2009. Advances in Simulation-Based Engineering Sciences - Honoring J. Tinsley Oden.

## APPENDIX

Listing 3. Random walks on spheres implementation

```
/**
 * @nof_walks: this first argument of the function corresponds to the number
 *             of walks we are willing to make.
 * @x_start: this second argument is the point whereat we want the value of
 *           the solution function.
 * The algorithm estimates the value of the solution function at @x_start by
 * averaging the estimates of the @nof_walks different walks on spheres.
 *
 * Some additional functions are used in the following:
 *   calc_sphere_rad(x) returns the radius of the largest sphere, in the domain,
 *                      that is centered at @x
 *   f(x) returns the value of the solution function on @x close to the boundary
 *   q(x) returns the value of the right hand side on @x
 */
double mc_estimate(int nof_walks, double *x_start)
{
  int i, j;
  double msol_est; //mean estimate of the solution

  msol_est = .0; //initialize msol_est
  for (i=0; i<nof_walks; i++) {
    double x[2]; //randomly chosen point on the boundary (random distribution: uniform)
    double y[2]; //randomly chosen point inside the sphere (random distribution: dictated by the Green function)
    double sol_est; //current estimate of the solution
    double d; //radius of the sphere at hand

    x[0] = x_start[0]; //initialize x
    x[1] = x_start[1];
    sol_est = .0; //initialize sol_est

    while ((d = calc_sphere_rad(x)) > btol) { //i.e., x is not close to the boundary
      rand_update_y(x, y, d); //update y
      sol_est += (d*d/4.)*q(y);

      rand_update_x(x, d); //update x
    }

    sol_est += f(x);

    msol_est += sol_est/nof_walks;
  }

  return msol_est;
}
```