IMPLEMENTING HYBRID PDE SOLVERS

Manolis Vavalis and George Sarailidis

http://dx.doi.org/10.6084/m9.figshare.1134520

Abstract We investigate the possibility that we effectively combine both conventional deterministic PDE solving methods and traditional probabilistic Monte Carlo approaches for solving linear Elliptic Partial Differential Equations. Our objective is to provide a robust and easy to use implementation that allows further experimentation on this new type of PDE solvers in order to elucidate their capabilities and computational characteristics. We first present the general formulation of the algorithm, then describe its implementation in C++ for a class of model problems in two and three space dimensions, we analyze its performance and we finally discuss possible extensions.

1 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 behaviour are capable of calculating approximations to the solutions. Specifically, it was [43] who first considered the raltionship between stohastic processes and parabolic differential equations followed by [14] who proposed numerical procedures for elliptic PDEs while [34] were the fist to dignify this stohastic approach with a name refering 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.

Manolis Vavalis · George Sarailidis
Department of Electrical & Computer Engineering, University of Thessaly, Gklavani 37, 38221 Volos, GREECE e-mail: mav@uth.gr, e-mail: gesarel@uth.gr

Since then Monte Carlo methods have been commonly, and in fact heavily[1], used and still are for many important problems, but not much for linear PDE-based applications. They are generally considered as methods of last resort idealy suitable only for problems either in high dimensions or very complex geometries [31]. 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 curently view Monte Carlo methods.

PDE problems have been related to Monte Carlo in several ways (see [27] 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 stil 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 [45, 7] for time depented problems and [9, 33, 45, 8, 55] for elliptic problems).

As already mentioned even fundamental linear PDEs are strongly related to stohasticity. 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. [25, 13, 23, 19] and in particular [27]). Linear non-stohastic elliptic boundary value problems are also strongly connected to probability (regorous measure theory). For example, integrals with respect to certain measure have been recognized as solutions of certain parabolic or elliptic differential equations [14]. It is worth to mention that there are several recent research efforts concerning probabilistic interpretations of harmonicity and of fundamental ellipric PDEs using Brownian motion and stochastic calculus (see [42] and reference therein).

In this paper we restrict our investigation on the effectivness of Monte Carlo methods for the numerical solution of linear elliptic PDEs and we concentrade on the Poisson equation. It has to be pointed out that although there has been, and curently exist, significant research activity on this subject, the proposed methods have not attracted so far the expected attention. Furthermore, one can find very few software components[2] that are publically available and appropriate to support the experimentation which is much needed for elucidating the characteristics and idiosygrasism of the proposed methods and convining both researchers and practitioners that can be effectivelly used for real-world problems (see for example [48].

[1] 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 begining of its supercomputing facilities.

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

In this paper we restrict ourselves on rectangular multidomains in two and three dimentions and we consider the implementation of a computational framework that allows easy experimentation with hybrid methods consisting of a combination of mainly two steps:

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

It is our believe the proposed and implemented framework promotes an interesting new concept in solving 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 2 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 adressed in section 3 which are coupled with instalation and usage details. A summary of the numerical experiments performed can be found in section 4. Our concluding remarks together with our vision for future enhasments and research prospects are given in section 5.

2 Mathematical Background

2.1 Monte Carlo methods for linear elliptic PDES

(we mention Courtant in the intro. is it?)

First was [39] which is based on the then classified work of [34].

They actually modivated people to build a special purpose machine [47] also [46] applications too

[53] deals with a Monte Carlo method for the numerical solution of the linear system that arices 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.

What methods do exist late [39, 18] recent [54, 11, 15, 35, 36, 16] in particular those in the past decade [29, 32, 23, 20, 38, 37, 17, 40, 24, 12, 45, 50, 41, 51, 48, 53] This recent excellent work have so far received minimal attention from our scientific community (e.g. according to citations at scopus.)

(Check if [26, 45, 48] is for stochastic PDES of just plain PDEs CLASIFY [21, 52])

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 pausible why the multicore available systems have not attarcted Monte Carlo methods at least as much as expected[3].

2.2 Stochastic/deterministic elliptic PDE solvers

We consider the following elliptic boundary value problem

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

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

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 $\mathscr{D}$, 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 $C_2(\mathscr{D} cap \partial\mathscr{D})$ of problem (1)–(2). We furthermore assume that the donain $\mathscr{D}$ consists of (or can be splitted into) $\mathscr{N}_{\mathscr{D}}$ subdomains, i.e.

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

and that L_μ and f_μ are the restrictions of L and f on $\mathscr{D}_\mu$ while B_μ and g_μ are the restrictions of B and g on $\partial\mathscr{D}_\mu \cap \partial\mathscr{D}$. We finally define the interface between the two subdomains $\mathscr{D}_\mu$ and $\mathscr{D}_\nu$ as

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

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

Obviously we consider only those interfaces that $\mathscr{I}_{\mu,\nu} \neq \emptyset$.

```
Data: $i_1, i_2, \dots, i_N$: the ids of the subdomains in which we wish to compute the
        solution.
Result: $\tilde{u}_\mu$, $\mu = i_1, \dots, i_N$: computer approximations of the restrictions of the
          exact solution $u$ in the subdomains $\mathscr{D}_\mu$, $\mu = i_1, \dots, i_N$.
// PHASE I: Estimate solution on the interfaces
;
while $\mathscr{I}_{\mu,\nu} \subset \cup_{j=1}^{N} \partial \mathscr{D}_{i_j}$ do
    Select control points $x_i \in \mathscr{I}_{\mu,\nu}, \ \ i = 1, 2, \dots, M_{\mu,\nu}$;
    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 $j = 1, 2, \dots, N$ do
    Solve the PDE problem:;
        $L_{i_j} u_{i_j}(x) = f_{i_j}(x) \ \ x \in \mathscr{D}_{i_j}$ ;
        $B_{i_j} u_{i_j}(x) = g_{i_j}(x) \ \ x \in \partial \mathscr{D}_{i_j} \cap \partial \mathscr{D}$ ;
        $L_{i_j} u_{i_j}(x) = h_{i_j}(x) \ \ x \in \mathscr{D}_{i_j}$ ; // $h_{i_j}(x)$ constructed using the
        $u^I_{\mu,\nu}$s
end
```

Algorithm 1: The Generic Algorithm.

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 Ω is complex and can be simplified if decomposed in subdomains .... In such cases it is very important that one selects the most appropiate 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 only on selected subdomains that are of particular importance to us.

Finally we should note that PARALLEL and in particular on distributed heterogeneous systems. Besides the inherent to the Monte Carlo method parallelism the MCDD enjoys several other parallel processing characteristics. Nice ratio communication computation. Preliminary numerical data support our claims while a systematic experimental verification of the above mentioned advances of MCDD is under way and will be presented elsewhere.

2.3 Related Work

Model Carlo based stochastic-deterministic hybrid methods are not new in general. Original idea maybe in [39] IS IT SO?. For non-stochastic linear PDEs though, it was only very recently that such methods have been proposed. To the best of our knowledge the idea first appeared at [21, 1] and furthermore considered in [1, 2, 3, 5, 6, 4]Elements of the above discribed algorithm have been considered previously. Specifically, ... Similar to us A Hybrid Stohastic/Deterministic Method See PDE model (and beyond) from lecture slides of Mascari

$$Lu \equiv -u''(x) = f(x), \quad x \in \Omega \equiv [a,b] \tag{5}$$

with $a, b, \gamma \in R$, subject to folowing generic boundary conditions

$$Bu(x) = g(x), \quad x \in \partial\Omega \equiv [a,b] \tag{6}$$

which, for simplicity in the presentation of the method are taken to be Dirichlet.

We assume that the regularity conditions for the closed domain and the given functions $b(x)$, c(x) 0, (x) and (x) are satisfied. These conditions guarantee the existence and uniqueness of the solution u(x) in C2 () C() of problem (1),

Assume that Ω is decomposed into the p non-overlapping subdomains $\Omega_i \equiv [x_{i-1}, x_i]$, $i = 1, \ldots, p$ with $x_0 = a$, $x_p = b$ and $x_{i-1} < x_i \in \Omega$ for $i = 1, \ldots, p-1$. We denote the size of a subdomain Ω_i by $\ell_i = x_i - x_{i-1}$ and the restrictions of L, f and γ in Ω_i by L_i, f_i, γ_i, respectively. We further assume that $\gamma(x) = \gamma_i$ for $x \in \Omega_i$, $i = 1, \ldots, p$, where the γ_i's are real constants.

Our implementation can solve Poisson's equation with Dirichlet boundary conditions. ([note: The following sentence is wrong - it's there just to be under consideration] What restricts this method of solving other problems too, is the need for the Green's function that suits the problem.)

It supports an arbitrary number of threads. For the multithreading we use the pthreads library (note: the program's form allow for easy migration to MPI).

Moreover, the domains that are supported are 2D and 3D hyperrectanges. Those can be decomposed to an arbitrary M x N grid of smaller subdomains (hyperrectangles).

Roughly, we accomplish probabilistic domain decomposition as follows:

compute solution at interface points using the Monte Carlo method described in [15] (we implement it ourselves – this part can be easily detached in order to be used in another application), we estimate local solutions along the boundaries of the new subdomains,

provide solution on interfaces we interpolate the estimated local solutions (we use Sintef's Multilevel B-spline (MBA) [28] library for the 3D and Burkardt's splines library for the 2D), and thus we form the boundaries of the new subdomains,

compute local solutions now that the boundaries of the subdomains are known, we solve the problem for each subdomain using a deterministic method (specifically ?finite elements?/?conjugate gradient?) (we use the deal.II library).

3 Implementation and Usage

The above described algorithm has been implemented utilizing three different technological frameworks.

1. Basic Implementation at https://github.com/mvavalis/Hybrid-numerical-PDE-solvers,
2. CPU/GPU implementation and
3. Web services implementation.

We note that the basic imlementation may be compined with either the CPU/GPU or with the web services or a compination of them. In the rest of this section and for the simplicity in the presentation, we discribe each implementation separetly.

3.1 Basic implementation

We use quasi only once for parallelism but see [30] This invited review of parallel quasi-Monte Carlo methods provides an overview of the subject and some new results for single eigenvalue computations.

external libraries include ???? lisence

The problem (i.e. the right hand side of the Poisson's equation and the boundary functions) is specified in file `Problem.h`. [note: mention `test_u()` also (more in the comments in main.cpp)]

`main()` creates a Pdd object, and calls `Pdd::pdd()` which takes care of the whole process.

`Pdd::pdd()` goes through the following steps:

1. it sets the coordinates of the nodes,
2. it creates a MCDriver object and calls `MCDriver::monte_carlo()`, which gets the coordinates of the nodes as input, and outputs the estimation for each node,
3. if we are solving a 3D problem, it creates an `Inter3DDriver` object and calls `Inter3DDriver::interpolation()`, which gets the coordinates of the nodes and their respective estimations as input, and outputs the new boundaries (2D planes); else it skips this step (when solving the 2D ploblem, interpolation is set to use directly through the finite element solver),
4. lastly, it creates a `LaplaceDriver` object and calls `LaplaceDriver::laplace_driver()`, in order to solve the problem in each subdomain and output the results.

`MCDriver::monte\_carlo()` creates one job for each node and hands the jobs to the available threads. It returns when all threads are finished. The estimation for each node is computed by `MCDriver::solve()` which uses the method of][15]. (described at the next section). `Inter3DDriver::interpolation()` creates one job for each node and hands the jobs to the available threads. It returns when all threads are finished. Each thread creates an MBA object and calls

`MBA::mba.MBAalg()` for the 2D interpolation. Sintef's Multilevel B-splines Library (MBA[4]) library is used. In particular we [28] ..

`LaplaceDriver::laplace\_driver()` creates one job (corresponding to a certain subdomain) for each node and hands the jobs to the available threads. It returns when all threads are finished. The solution for each subdomain is computed by `LaplaceSolve::run()`.

The numerical solution of the partial differential equations in each subdomain defined is computed by `LaplaceSolve::run()` which properly utilizes the state of the art C++ program library deal.II[5]. This recently developed and already widely used library [10] offers adaptive finite element solvers of high quality for the numerical solution of partial differential equations. Specifficaly the 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. [note: more on this later]

The walk-on-spheres algorithm and our implementation (`MCDriver::solve()`):

`MCDriver::solve()` is based on the walk-on-spheres method of [15] They describe the algorithm (one walk) as follows (say, we want to estimate $u(x_0)$): [additional definitions: s is the current solution estimation; $B(x)$ is the largest ball in the domain centered at point x; $q(y)$ is the right hand side of the problem; a(d) is a function associated with Green's function for the problem, which takes as input the radius of the B(x)] 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 many times, and the mean of the estimations at the end of each process is used as the final estimation.

`MCDriver::solve()` takes as input the coordinates of the node (argument x, which is a vector of 2 or 3 dimensions) and the number of walks to do (argument `nof_walks);` and outputs the estimation of the value of the function which we want to find at point x. Note that the first step in each walk is accomplished using a quasi-random sequence; [conjecture] 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 helps a lot to make a more uniform sampling, which in turn results in faster convergence [fysika 8a mporouse na einai ola quasi, alla auto einai zoriko...]. The code describing its function follows:

Listing 1 Test
```
double MCDriver<DIMS>::solve(int nof_walks, const double *x)
{
        double msol_est = .0; //mean of computed solutions
        for (i=0; i<nof_walks; i++) {
                double sol_est = .0; //current computed solution
                if ((d = calc_sphere_rad(x)) > btol) {
                        quasi_update_y(x, y, d);
```

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

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

```
                        sol_est += a(d)*Prob.q(y);

                        quasi_update_x(x, d);
                }

                while ((d = calc_sphere_rad(x)) > btol) {
                        rand_update_y(x, y, d);
                        sol_est += a(d)*Prob.q(y);

                        rand_update_x(x, d);
                }
                sol_est += Prob.f(x);

                msol_est += sol_est/nof_walks;
        }

        return msol_est;
}
```

Additional variables and functions used: d: the radius of the current ball (calculated by `calc_sphere_rad(x)`, which receives as input the center of the ball x) `btol`: the boundary tolerance `Prob.f(x)`: the values on the boundary `Prob.q(y)`: the right hand side of the problem `rand_update_y(x, y, d)`: find randomly a point inside B(x), with respect to the density of B(x) `rand_update_x(x, d)`: find randomly a point on the surface of B(x) `quasi_update_y(x, y, d)`: same as `rand_update_y(x, y, d)`, however, using a quasi-random sequence `quasi_update_x(x, d)`: same as `rand_update_x(x, d)`, however, using a quasi-random sequence `a(d)`: a function associated with Green's function (as described in [15])

More on how we find randomly a point inside $B(x)$, with respect to the density of $B(x)$ (`rand_update_y(x, y, d)`): [for the two dimensional case]

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 [44].

[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.

[$f(x) = max(PDF')$ (this makes the implementation even simpler (and faster)): $(d/dx)(PDF') = (4/d^2)*(lnd - lnr - 1) = 0 <=> lnr = lnd - lne <=> r = d/e$, that is $f(x) = max(PDF') = max(2*pi*\rho(r,\theta)) = 2*pi*\rho(d/e,\theta) = 4/(e*d)$

What we gonna do: 1) choose uniformly a random x1 in (0, d), and a random x2 in (0, 4/(e*d)) 2) check if (x1, x2) is below the PDF' curve, that is if $2 * pi * \rho(x1, \theta) < x2$. If it is, we found our radius x1, else go to step 1.]

3.2 Parallel implementation on Central Processing Units (CPUs) and Graphics Proccesing Units (GPUs)

OpenCL (Open Computing Language) presents a framework for developing programs which execute across heterogeneous platforms consisting of CPUs, GPUs, and other processors. OpenCL 1.0 out late 2008 Vision: write one portable application and execute in any processor or collection of processors. Strong industry support and drivers out for NVIDIA, Intel, AMD/ATI, IBM (Cell) chipsets etc. CUDA? Since 2013, OpenCL is supported by ARM, Altera, Intel etc. and became an industry standard. CUDA only for NVIDIA but it is simpler to implement.

3.3 Web Implementation through web services

4 Numerical Experiments

4.1 2-dimensional Experiments

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

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = (1-\pi^2)\left(\sin(\pi x)\sinh(y) + 4\cosh(2x)\cos(2\pi y)\right), \ \forall (x,y) \in \Omega, \quad (7)$$

subject to the following Dirichlet boundary conditions

$$\begin{array}{l} u(\pm 1, y) = \cosh(\pm 2)\cos(2\pi y) \\ u(x, \pm 1) = \sin(\pi x)\sinh(\pm 1) + \cosh(2x), \end{array} \ \forall (x,y) \in \partial\Omega. \quad (8)$$

The exact solution of the above problem is given by

$$u(x,y) = \sin(\pi x)\sinh(y) + \cosh(2x)\cos(2\pi y). \quad (9)$$

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

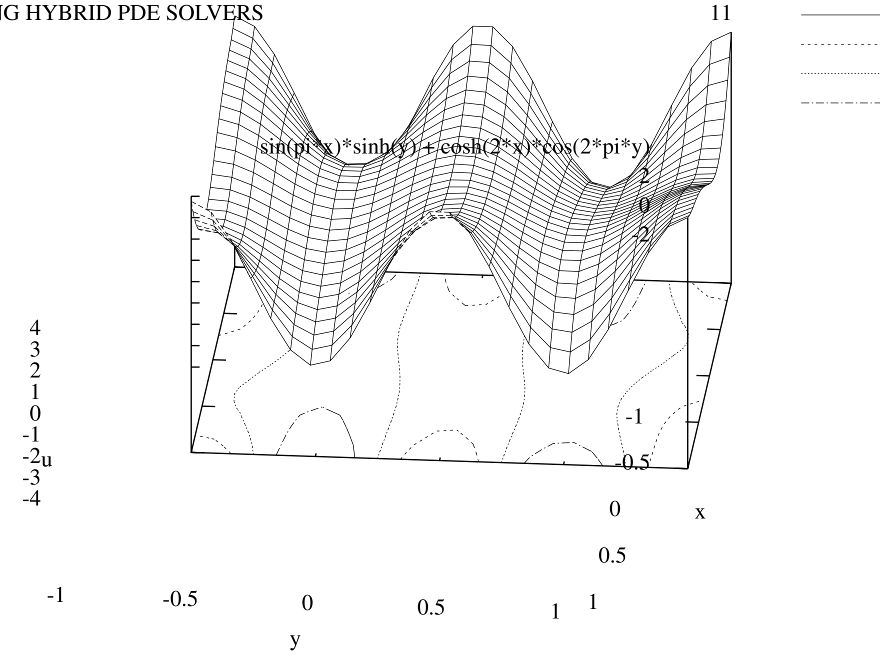

Fig. 1 True solution of the PDE problem defined by (7)–(8).

4.2 3-dimensional Experiments

We consider a slitely modified the PDE problem considered in the previous section as follows.

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\frac{\partial^2 u}{\partial z^2}=x+y+z \;\; \forall(x,y)\in\Omega\equiv[-1,1]^3, \qquad (10)$$

subject to the Dirichlet boundary conditions

$$u(x,y,z)=g(x,y,z) \;\; \forall(x,y)\in\partial\Omega, \qquad (11)$$

where the right hand side function g is selected so that the exact solution of the above problem (10)–(11) is given by the equation

$$u(x,y,z)=\exp(\sqrt{2}\pi x)\sin(\pi(y+z))+\frac{1}{6}\left(x^3+y^3+z^3\right). \qquad (12)$$

and depicted in figure 2.

In a similar to the 2-dimensional case we decompose the domain Ω into 16 non-overlapping subdomains defined by the intreface planes $x_1=0$ and $y_1=-0.5$, $y_2=0$, $y_3=0.75$ and $z_1=-0.2$. ...

Fig. 2 True solution of the PDE problem defined by (10)–(11).

5 Conclusions and Prospects

Our objective 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.

extended to more general operators [54] possibly with singularities [20] operators of higher order [22, 12] deal with Neumman or mixed boundary conditions [49, 50, 51] non-rectangular domains [**?**] and virtualy to any problem with known Green's function. to time depended problems [13, 48, 19]

Furthermore, the implementation of high performance Monte Carlo solvers on modern architectures and emerging computational platforms (e.g. many-core systems, GPUs and streaming computing.

Above all our study is based on a new line of reasoning that provides new intuition about the dynamics of Monte Carlo simualtions.

Monolithic and such

Acknowledgements 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.

References

1. Acebrón, J., Busico, M., Lanucara, P., Spigler, R.: Probabilistically induced domain decomposition methods for elliptic boundary-value problems. Journal of Computational Physics **210**(2), 421–438 (2005)
2. Acebrón, J., Busico, M., Lanucara, P., Spigler, R.: Domain decomposition solution of elliptic boundary-value problems via Monte Carlo and quasi-Monte Carlo methods. SIAM Journal of Scientific Computing **27**(2), 440–457 (2006)
3. Acebrón, J., Durán, R., Rico, R., Spigler, R.: A new domain decomposition approach suited for grid computing. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **4699 LNCS**, 744–753 (2007). URL http://www.scopus.com/inward/record.url?eid=2-s2.0-38049032039&partnerID=40
4. Acebrón, J., Rodríguez-Rozas, A., R., S.: Domain decomposition solution of nonlinear two-dimensional parabolic problems by random trees. Journal of Computational Physics **228**(15), 5574–5591 (2009). URL http://www.scopus.com/inward/record.url?eid=2-s2.0-67349224351&partnerID=40
5. Acebrón, J., Spigler, J.: A fully scalable parallel algorithm for solving elliptic partial differential equations. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **4641 LNCS**, 727–736 (2007). URL http://www.scopus.com/inward/record.url?eid=2-s2.0-38049187637&partnerID=40
6. Acebrón, J., Spigler, R.: Scalability and performance analysis of a probabilistic domain decomposition method. Lecture Notes in Computer Science (including subseries Lecture Notes in

Artificial Intelligence and Lecture Notes in Bioinformatics) **4967 LNCS**, 1257–1264 (2008). URL http://www.scopus.com/inward/record.url?eid=2-s2.0-45449108802&partnerID=40
7. Alves, C., Cruzeiro, A.: Monte Carlo simulation of stochastic differential systems - a geometrical approach. Stochastic Processes and their Applications **118**(3), 346–367 (2008)
8. Babuska, I., Nobile, F., Tempone, R.: A stochastic collocation method for elliptic partial differential equations with random input data. SIAM Journal on Numerical Analysis **45**(3), 1005–1034 (2007)
9. Babuska, I., Tempone, R., Zouraris, G.E.: 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)
10. Bangerth, W., Hartmann, R., Kanschat, G.: deal.II—A general-purpose object-oriented finite element library. ACM Trans. Math. Softw. **33**(4), 24–?? (2007). DOI http://doi.acm.org/10.1145/1268776.1268779
11. Bignami, A., Cupini, E.: Monte Carlo method for finite difference equations of elliptic type in a multiregion domain. J. Comput. Appl. Math. **8**(2), 87–92 (1982)
12. Buchmann, F.M., Petersen, W.P.: An exit probability approach to solving high dimensional dirichlet problems. SIAM Journal of Scientific Computing **28**(3), 1153–1166 (2006)
13. Carlsson, J.: A backward Monte Carlo method for solving parabolic partial differential equations. arXiv:math/0010118v1 (2000)
14. Courant, R., Friedrichs, K., Lewy, H.: *Ü*ber die partiellen differenzengleichungen der mathematischen physik. Mathematische Annalen **100**, 32–74 (1928)
15. DeLaurentis, J.M., Romero, L.A.: A Monte Carlo method for Poisson's equation. J. Comput. Phys. **90**(1), 123–140 (1990). DOI http://dx.doi.org/10.1016/0021-9991(90)90199-B
16. Dimov, I.T., Gurov, T.V.: 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)
17. Dimov, I.T., Papancheva, R.Y.: Green's function Monte Carlo algorithms for elliptic problems. Mathematics and Computers in Simulation **63**(6), 587–604 (2003)
18. Edwards, K., Hogg, A.: Methods of improving analogue Monte Carlo solutions of elliptic partial differential equations **1**, 664–673 (1967)
19. Farnoosh, R., Ebrahimi, M.: Monte Carlo method via a numerical algorithm to solve a parabolic problem. Applied Mathematics and Computation **190**(2), 1593–1601 (2007)
20. Given, J., Hwang, C.: Edge distribution method for solving elliptic boundary value problems with boundary singularities. Physical Review E - Statistical, Nonlinear, and Soft Matter Physics **68**(4 2), 461,281–461,286 (2003)
21. Gobet, E., Maire, S.: Sequential Monte Carlo domain decomposition for the Poisson equation. In: 17th IMACS World Congress, Scientific Computation, Applied Mathematics and Simulation (2005). URL http://maire.univ-tln.fr/fichiersweb/T1-I-62-0995%20(1).pdf
22. Gopalsamy, K., Aggarwala, B.: Monte Carlo methods for some fourth order partial differential equations. Z Angew Math Mech **50**(12), 759–767 (1970)
23. Griebel, M., Schweitzer, M.A.: 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)
24. Heinrich, S.: The randomized information complexity of elliptic PDE. Journal of Complexity **22**(2), 220–249 (2006)
25. Hoshino, S., Ichida, K.: Solution of partial differential equations by a modified random walk. Numerische Mathematik **18**(1), 61–72 (1971)
26. Karaivanova, A., Hongmei, C., Gurov, T.: Quasi-random walks on balls using c.u.d. sequences. pp. 165–172. Springer (2007)
27. Lapeyre, B., Pardoux, E., Sentis, R.: Introduction to Monte Carlo Methods for Transport and Diffusion Equations. 0xford University Press (2003). Translated by A. Craig and F. Craig
28. Lee, S., Wolberg, G., Shin, S.: Scattered data interpolation with multilevel B-splines. IEEE Transactions on Visualization and Computer Graphics **3**(3), 228–244 (1997). URL http://www.scopus.com/inward/record.url?eid=2-s2.0-0031190350&partnerID=40. Cited By (since 1996) 175

29. Makarov, R.N.: 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)
30. Mascagni, M.: Deterministic Monte Carlo methods and parallelism. pp. 249–258 (2003)
31. Mascagni, M., Hwang, C.: ε-Shell error analysis for "Walk On Spheres" algorithms. Mathematics and Computers in Simulation **63**(2), 93–104 (2003)
32. Mascagni, M., Karaivanova, A., Li, Y.: A quasi-Monte Carlo method for elliptic partial differential equations. Monte Carlo Methods and Applications **7**, 283–294 (2001)
33. Matthies, H.G., Keese, A.: Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations. Computer Methods in Applied Mechanics and Engineering **194**(12-16), 1295–1331 (2005)
34. Metropolis, N., Ulam, S.: The Monte Carlo method. Journal of the American Statistical Association **44**, 335–341 (1949)
35. Mikhailov, G.A.: Recurrent formulae and the Bellman principle in the Monte Carlo method. Russian Journal of Numerical Analysis and Mathematical Modelling **9**(3), 281–289 (1994)
36. Mikhailov, G.A.: Solving the Dirichlet problem for nonlinear elliptic equations by the Monte Carlo method. Siberian Mathematical Journal **35**(5), 967–975 (1994)
37. Mikhailov, G.A., Lukinov, V.L.: Probability representations and the Monte Carlo method for solving equations with powers of elliptic operators. Doklady Mathematics **67**(3), 423–425 (2003)
38. Milstein, G., Tretyakov, M.: The simplest random walks for the Dirichlet problem. Theory of Probability and its Applications **47**(1), 53–68 (2003)
39. Muller, M.: Some continuous Monte Carlo methods for the Dirichlet problem. The Annals of Mathematical Statistics **27**(3), 569–589 (1956)
40. Papancheva, R.J., Dimov, I.T., Gurov, T.V.: A new class of grid-free Monte Carlo algorithms for elliptic boundary value problems, vol. 2542, pp. 132–139 (2003)
41. Papancheva, R.Y.: Parallel realization of grid-free Monte Carlo algorithm for boundary value problems, *Lecture Notes in Computer Science*, vol. 3743, pp. 181–188. Springer (2006)
42. Privault, N.: Potential theory in classical probability. In: U. Franz, M. Schurmann (eds.) Quantum Potential Theory, *Lecture Notes in Mathematics*, vol. 4310, pp. 3–59. Springer (2008)
43. Rayleigh, L.: On James Bernoullis theorem in probabilities. Philos. Mag. **47**, 246–251 (1899)
44. Robert, C.P., Casella, G.: Monte Carlo Statistical Methods. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2005)
45. Roman, L., Sarkis, M.: Stochastic Galerkin method for elliptic SPDEs: A white noise approach. Discrete and Continuous Dynamical Systems - Series B **6**(4), 941–955 (2006)
46. Sacadura, J., Al-Abed, A.: Analysis of time-dependent cylindrical problems using Monte Carlo. ASME, Transactions, Journal of Heat Transfer (ISSN 0022-1481) **105**, 931–933 (1983)
47. Sadeh, E., Franklin, M.: Monte Carlo solution of partial differential equations by special purpose digital computer. IEEE Transactions on Computers **23**(4), 389–397 (1974). DOI http://doi.ieeecomputersociety.org/10.1109/T-C.1974.223954
48. Sadiku, M., Akujuobi, C., Musa, S., Nelatury, S.: Analysis of time-dependent cylindrical problems using Monte Carlo. Microwave and Optical Technology Letters **49**(10), 2571–2573 (2007)
49. Sadiku, M., Gu, K.: New Monte Carlo method for -Neumann problems pp. 92–95 (1996)
50. Simonov, N.: Monte Carlo methods for solving elliptic equations with boundary conditions containing the normal derivative. Doklady Mathematics **74**(2), 656–659 (2006)
51. Simonov, N.: Random walks for solving boundary-value problems with flux conditions. pp. 181–188. Springer (2007)
52. Tsai, Y., Wang, C., Chang, C., Cheng, Y.: Tunable bounding volumes for Monte Carlo applications, *Lecture Notes in Computer Science*, vol. 3980, pp. 171–180. Springer (2006)
53. Vajargah, B., Vajargah, K.: Monte Carlo method for finding the solution of Dirichlet partial differential equations. Applied Mathematical Sciences **1**(10), 453–462 (2007)
54. Vrbik, J.: Monte Carlo simulation of the general elliptic operator. J. Phys. A: Math. Gen. **20**(3), 2693–2697 (1987)

55. Wan, X., Karniadakis, G.: Solving elliptic problems with non-Gaussian spatially-dependent random coefficients. Computer Methods in Applied Mechanics and Engineering **198**(21–26), 1985–1995 (2009). DOI DOI:10.1016/j.cma.2008.12.039. URL http://www.sciencedirect.com/science/article/B6V29-4VFK7SW-2/2/97c813d79f0e5f6bec99789f23409016. Advances in Simulation-Based Engineering Sciences - Honoring J. Tinsley Oden