A Study of Biased and Unbiased Stochastic Algorithms for Solving Integral Equations

Ivan Tomov Dimov, Sylvain Maire, Rayna Georgieva

A Study of Biased and Unbiased Stochastic Algorithms for Solving Integral Equations

I.T. Dimov[1], S. Maire[2], and R. Georgieva[1]

[1] Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Acad. G. Bonchev 25A, 1113 Sofia, Bulgaria
ivdimov@bas.bg, rayna@parallel.bas.bg

[2] Laboratoire LSIS, UMR 7296 Equipe Signal et Image, Université de Toulon, AV. Georges Pompidou BP 56 83162 La Valette du Var Cedex, France
maire@univ-tln.fr

Abstract. In this paper we propose and analyse a new unbiased stochastic method for solving a class of integral equations, namely the second kind Fredholm integral equations. We study and compare three possible approaches to compute linear functionals of the integral under consideration: i) biased Monte Carlo method based on evaluation of truncated Liouville-Neumann series, ii) transformation of this problem into the problem of computing a finite number of integrals, and iii) unbiased stochastic approach. Five Monte Carlo algorithms for numerical integration have been applied for approach (ii). Error balancing of both stochastic and systematic errors has been discussed and applied during the numerical implementation of the biased algorithms. Extensive numerical experiments have been performed to support the theoretical studies regarding the convergence rate of Monte Carlo methods for numerical integration done in our previous studies. We compare the results obtained by some of the best biased stochastic approaches with the results obtained by the proposed unbiased approach. Conclusions about the applicability and efficiency of the algorithms have been drawn.

1 Introduction

Integral equations are of high applicability in different areas of applied mathematics, physics, and engineering. In particular, they are widely used in mechanics, geophysics, electricity and magnetism, kinetic theory of gases, quantum mechanics, mathematical economics, and queuing theory. That is why it is reasonable to develop and study efficient and reliable approaches to solve integral equations.

An important advantage of Monte Carlo (MC) methods is that they allow to find directly an unknown linear functional of the solution of integral equations with a number of operations necessary to calculate the solution of an integral equation only at one point of the domain [15]. The existing MC methods for integral equations (MCM-int eq.) are based on probabilistic representations of the Liouville-Neumann series for the second kind Fredholm integral equation [2, 4].

The possible unbiased approaches deal with infinite series, while the biased MC approaches use probabilistic representations of truncated Liouville-Neumann series. A well known and widely used biased method is the Markov chain MC (see, for example [11]). Usually, the Markov chain stops after a fixed number of steps.

The unbiased stochastic algorithm proposed in this work is actually based on a probabilistic representation of the infinite series introducing some probability for absorbtion. The idea of the construction of the algorithm is similar to the one used in [8] for solving linear systems.

A possible approach to deal with the problem of approximation of linear functionals of the solution of an integral equation is to transform it into approximate evaluation of finite number of integrals (FNI) (linear functionals of iterative functions) [3, 4]. We have already applied a variance reduction approach called *importance separation method* for each of integrals and extended the study of its properties for solving integral equations (see [10]). Here we extend the study of the properties to five different MC algorithms for multidimensional numerical integration for solving integral equations. These algorithms are: Crude MC algorithm [15], based on SIMD-oriented fast Mersenne Twister pseudo-random number generator [13, 18], quasi-Monte Carlo (qMC) algorithm based on $\Lambda\Pi_\tau$ Sobol quasirandom sequences, MC algorithms (MCA-MSS-1 and MCA-MSS-2) based on modified Sobol quasirandom sequences [5–7] and a stratified symmetrised MC algorithm (MCA-MSS-2-S) [5].

The paper is organized as follows. The problem setting is given in Section 2. A description of biased stochastic approaches applied to solve the problem under consideration is presented in Section 3. In this section some error analysis based on the concept of balancing both stochastic and systematic errors is done. The new unbiased stochastic approach is presented in Section 4. Numerical results are described and analysed in Section 5. Some final conclusions are drawn in Section 6.

2 Formulation of the Problem

Consider the second kind Fredholm integral equation:

$$u(\mathrm{x}) = \int_\Omega k(\mathrm{x}, \mathrm{x}')u(\mathrm{x}')\mathrm{d}\mathrm{x}' + f(\mathrm{x}) \tag{1}$$

or

$$u = \mathcal{K}u + f \ \ (\mathcal{K} \text{ is an integral operator}), \quad \text{where}$$

$k(\mathrm{x}, \mathrm{x}') \in L_2(\Omega \times \Omega), f(\mathrm{x}) \in L_2(\Omega)$ are given functions and $u(\mathrm{x}) \in L_2(\Omega)$ is an unknown function, $\mathrm{x}, \mathrm{x}' \in \Omega \subset \mathbb{R}^n$ (Ω is a bounded domain).

We are interested in Monte Carlo methods for evaluation of linear functionals of the solution of the following type:

$$J(u) = \int \varphi(\mathrm{x})u(\mathrm{x})\mathrm{d}\mathrm{x} = (\varphi, u), \tag{2}$$

where $\varphi(\mathrm{x}) \in L_2(\Omega)$ is a given function. We can apply successive approximation method for solving integral equations:

$$u^{(i)} = \sum_{j=0}^{i} \mathcal{K}^{(j)} f = f + \mathcal{K} f + \ldots + \mathcal{K}^{(i-1)} f + \mathcal{K}^{(i)} f, \quad i = 1, 2, \ldots \tag{3}$$

where $u^{(0)}(\mathrm{x}) \equiv f(\mathrm{x})$. It is known that the condition $\|\mathcal{K}\|_{L_2} < 1$ is a sufficient condition for convergence of the Liouville-Neumann series [2, 4]. Thus, when this condition is satisfied, the following statement holds:

$$u^{(i)} \longrightarrow u \quad \text{as} \quad i \to \infty.$$

Therefore,

$$J(u) = (\varphi, u) = \lim_{i\to\infty} (\varphi, u^{(i)}) = \lim_{i\to\infty} \left(\varphi, \sum_{j=0}^{i} \mathcal{K}^{(j)} f \right) = \lim_{i\to\infty} \sum_{j=0}^{i} \left(\varphi, \mathcal{K}^{(j)} f \right).$$

Usually it is assumed that $\varphi(\mathrm{x}) \in L_2(\Omega)$, because $k(\mathrm{x}, \mathrm{x}') \in L_2(\Omega \times \Omega)$, $f(\mathrm{x}) \in L_2(\Omega)$. In a more general setting $k(\mathrm{x}, \mathrm{x}') \in \mathbf{X}(\Omega \times \Omega)$, $f(\mathrm{x}) \in \mathbf{X}(\Omega)$, where $\mathbf{X}$ is a Banach space. Then the given function $\varphi(\mathrm{x})$ should belong to the adjoint Banach space $\mathbf{X}^*$, and the problem under consideration may by formulated in an alternative way:

$$v = \mathcal{K}^* v + \varphi, \tag{4}$$

where $v, \varphi \in \mathbf{X}^*(\Omega)$, and $\mathcal{K}^*(\Omega \times \Omega) \in [\mathbf{X}^* \to \mathbf{X}^*]$. In such a way one may compute the value $J(v) = \int f(\mathrm{x}) v(\mathrm{x}) \mathrm{dx} = (f, v)$ instead of (2). An important case for practical computations is the case when $\mathbf{X} \equiv L_1$, where L_1 is defined in a standard way:

$$\| f \|_{L_1} = \int_{\Omega} |f(\mathrm{x})| dx; \quad \| \mathcal{K} \|_{L_1} \leq \sup_{\mathrm{x}} \int_{\Omega} |k(\mathrm{x}, \mathrm{x}')| dx'.$$

In this case the function $\varphi(\mathrm{x})$ from the functional (2) belongs to L_∞, i.e. $\varphi(\mathrm{x}) \in L_\infty$ since $L_1^* \equiv L_\infty$. It is also easy to see that $(\mathcal{K}^* v, u) = (v, \mathcal{K} u)$, and also $(\varphi, u) = (f, v)$. This fact is important for practical computations since often the computational complexity for solving the adjoint problem is smaller than the complexity for solving the original one. The above consideration shows that if $\varphi(\mathrm{x})$ is a Dirac δ–function $\delta(\mathrm{x} - \mathrm{x}_0)$, then the functional $J(u) = (\varphi(\mathrm{x}), u(\mathrm{x})) = (\delta(\mathrm{x} - \mathrm{x}_0), u(\mathrm{x})) = u(\mathrm{x}_0)$ is the solution of the integral equation at the point x_0, if $u \in L_\infty$.

An approximation of the unknown value (φ, u) can be obtained using a truncated Liouville-Neumann series (3) for a sufficiently large i:

$$(\varphi, u^{(i)}) = (\varphi, f) + (\varphi, \mathcal{K} f) + \ldots + (\varphi, \mathcal{K}^{(i-1)} f) + (\varphi, \mathcal{K}^{(i)} f).$$

So, we transform the problem for solving integral equations into a problem for approximate evaluation of a finite number of multidimensional integrals. We will

use the following denotation $(\varphi, \mathcal{K}^{(j)} f) = I(j)$, where $I(j)$ is a value, obtained after integration over $\Omega^{j+1} = \Omega \times \ldots \times \Omega$, $j = 0, \ldots, i$. It is obvious that the calculation of the estimate $(\varphi, u^{(i)})$ can be replaced by evaluation of a sum of linear functionals of iterative functions of the following type $(\varphi, \mathcal{K}^{(j)} f), j = 0, \ldots, i$, which can be presented as:

$$\begin{aligned}(\varphi, \mathcal{K}^{(j)} f) &= \int_{\Omega} \varphi(\mathrm{t}_0) \mathcal{K}^{(j)} f(\mathrm{t}_0) \mathrm{dt}_0 = \\ &= \int_{G} \varphi(\mathrm{t}_0) k(\mathrm{t}_0, t_1) \ldots k(\mathrm{t}_{j-1}, \mathrm{t}_j) f(\mathrm{t}_j) \mathrm{dt}_0 \ldots \mathrm{dt}_j,\end{aligned} \tag{5}$$

where $\mathrm{t} = (\mathrm{t}_0, \ldots, \mathrm{t}_j) \in G \equiv \Omega^{j+1} \subset \mathbb{R}^{n(j+1)}$. If we denote by $F_j(\mathrm{t})$ the integrand function

$$F(\mathrm{t}) = \varphi(t_0) k(t_0, t_1) \ldots k(\mathrm{t}_{j-1}, \mathrm{t}_j) f(\mathrm{t}_j), \quad t \in \Omega^{j+1},$$

then we will obtain the following expression for (5):

$$I(j) = (\varphi, \mathcal{K}^{(j)} f) = \int_{G} F_j(\mathrm{t}) \mathrm{dt}, \quad \mathrm{t} \in G \subset \mathbb{R}^{n(j+1)}. \tag{6}$$

Thus, we will consider the problem for approximate calculation of multiple integrals of the type (6).

The above consideration shows that there are two classes of possible stochastic approaches. The first one is the so-called biased approach when one is looking for a random variable which mathematical expectation is equal to the approximation of the solution problem by a truncated Liouville-Neumann series (3) for a sufficiently large i. An unbiased approach assumes that the formulated random variable is such that its mean value approaches the true solution of the problem. Obviously, in the first class of approaches there are two errors: a systematic one (a truncation error) R_{sys} and a stochastic (a probabilistic) one, namely R_N, which depends on the number N of values of the random variable,or the number of chains used in the estimate. In the case of unbiased stochastic methods one should analyse the only probabilistic error. In the case of biased stochastic methods more careful error analysis is needed: balancing of both systematic and stochastic error should be done in order to minimize the computational complexity of the methods (for more details, see [4]).

Let us now present the simplest stochastic approach, known also as Crude Monte Carlo method (CMCM) for evaluating integrals. It is well-known that CMCM reduces problem to the approximate calculation of mathematical expectation which coincides with the unknown functional defined by (2) (see, for example, [4]). The Monte Carlo quadrature formula is based on the probabilistic interpretation of an integral. If $\{\mathrm{x}_l\}, l = 1, \ldots, N$ is a uniformly distributed sequence in G, then the Crude Monte Carlo approximation to the integral $I(j)$ is

$$I(j) \approx I_N(j) = \frac{\mathrm{V}(G)}{N} \sum_{l=1}^{N} F_j(\mathrm{x}_l)$$

with integration error $\varepsilon_N(j) = |I(j) - I_N(j)| \approx \sqrt{\frac{Var(F)}{N}}$, where V($G$) is the volume of G and $Var(F_j)$ is the variance of the corresponding random variable whose mathematical expectation coincides with the unknown functional. This random variable depends on the integrand $F(t)$.

3 Biased Stochastic Approaches

In this section, a brief description of the biased stochastic approaches for solving the problem (1)-(2) presented in Section 2 is given.

3.1 Monte Carlo Method for Integral Equations

It is known (see [15, 4]) that the approximate calculation of linear functionals of the solution of an integral equation (φ, u) brings to the calculation of a finite sum of linear functionals of iterative functions $(\varphi, \mathcal{K}^j f), j = 0, \ldots, i$. First, we construct a random trajectory (Markov chain) T_i of length i starting from state x_0 in the domain Ω:

$$T_i : \mathrm{x}_0 \longrightarrow \mathrm{x}_1 \longrightarrow \ldots \longrightarrow \mathrm{x}_i$$

according to the initial $\pi(\mathrm{x})$ and transition $p(\mathrm{x}, \mathrm{x}')$ probabilities. The functions $\pi(\mathrm{x})$ and $p(\mathrm{x}, \mathrm{x}')$ satisfy the requirements for non-negativeness, to be acceptable to function $\varphi(\mathrm{x})$ and the kernel $k(\mathrm{x}, \mathrm{x}')$ respectively and $\int_\Omega \pi(\mathrm{x})\mathrm{d}\mathrm{x} = 1$, $\int_\Omega p(\mathrm{x}, \mathrm{x}')\mathrm{d}\mathrm{x}' = 1$ for any $\mathrm{x} \in \Omega \subset \mathbb{R}^n$. From the above supposition on the kernel $k(\mathrm{x}, \mathrm{x}')$ and the well-known facts that

$$\mathbf{E}\theta_i[\varphi] = (\varphi, u^{(i)}), \quad \text{where} \quad \theta_i[\varphi] = \frac{\varphi(\mathrm{x}_0)}{\pi(\mathrm{x}_0)} \sum_{j=0}^{i} W_j f(\mathrm{x}_j),$$

$$\text{and} \quad W_0 = 1, \quad W_j = W_{j-1} \frac{k(\mathrm{x}_{j-1}, \mathrm{x}_j)}{p(\mathrm{x}_{j-1}, \mathrm{x}_j)}, \quad j = 1, \ldots, i$$

it follows that the corresponding Monte Carlo estimation of $(\varphi, u^{(i)})$ can be presented in the following form:

$$(\varphi, u^{(i)}) \approx \frac{1}{N} \sum_{l=1}^{N} \theta_i[\varphi]_l.$$

Therefore, the random variable $\theta_i[\varphi]$ can be considered as an biased estimate of the desired value (φ, u) for i sufficiently large with a statistical error of order $\mathcal{O}(N^{-1/2})$, where N is the number of chains and $\theta_i[\varphi]_l$ is the value of $\theta_i[\varphi]$ taken over the l-th chain.

It is very important that the same trajectories of the type T_i can be used for a biased approximate evaluation of $(\varphi, u^{(i)})$ for various functions $\varphi(\mathrm{x})$. Furthermore, they can be used for various integral equations with the same kernel $k(\mathrm{x}, \mathrm{x}')$, but with different right-hand sides $f(\mathrm{x})$.

3.2 Monte Carlo Algorithms based on Modified Sobol $\Lambda\Pi_\tau$ Sequences

$\Lambda\Pi_\tau$ sequences are *uniformly distributed sequences* (u.d.s.). The term *u.d.s.* was introduced by Hermann Weyl [17]. For practical purposes an u.d.s. should satisfy the following requirements [15, 16]: (i) the best asymptote as $N \to \infty$, (ii) well distributed points for small N, and (iii) a computationally inexpensive algorithm. Suitable distributions such as $\Lambda\Pi_\tau$ sequences are also called (t, m, s)-nets and (t, s)-sequences in base $b \geq 2$ [12]. I. M. Sobol [15] defines his Π_τ-meshes and $\Lambda\Pi_\tau$ sequences, which are (t, m, s)-nets and (t, s)-sequences in base 2, respectively. Subroutines to compute these points can be found in [1], and with more details in [14]. Here we consider two randomized quasi-Monte Carlo algorithms based on Sobol $\Lambda\Pi_\tau$ sequences, namely MCA-MSS-1 and MCA-MSS-2.

3.2.1 Summary of the MCM-MSS-1. One of the algorithms (based on a procedure of *shaking*) was proposed recently in [6]. The idea is that we take a Sobol $\Lambda\Pi_\tau$ d-dimensional point (vector) x. Then x is considered as a centrum of a sphere with a radius $\rho << 1$. A random point $\xi \in U^d$ uniformly distributed on the sphere is taken. Consider a random variable θ defined as a value of the integrand at that random point, i.e., $\theta_j = F_j(\xi)$. Consider random points $\xi^{(l)}(\rho) \in U^d, l = 1, \ldots, N$. Assume $\xi^{(l)}(\rho) = \mathrm{x}^{(l)} + \rho\omega^{(l)}$, where $\omega^{(l)}$ is a unique uniformly distributed vector in U^d. The radius ρ is relatively small $\rho \ll \dfrac{1}{2^{d_j}}$, such that $\xi^{(l)}(\rho)$ is still in the same elementary l^{th} interval $E_l^d = \prod_{\nu=1}^{d} \left[\dfrac{a_j^{(l)}}{2^{d_\nu}}, \dfrac{a_\nu^{(l)}+1}{2^{d_\nu}}\right]$, where the pattern $\Lambda\Pi_\tau$ point $\mathrm{x}^{(l)}$ is. We use a subscript i in E_i^d to indicate that the l-th $\Lambda\Pi_\tau$ point $\mathrm{x}^{(i)}$ is in it. So, we assume that if $\mathrm{x}^{(l)} \in E_l^d$, then $\xi^{(l)}(\rho) \in E_l^d$ too. It was proven in [6] that the mathematical expectation of the random variable $\theta_j = F_j(\xi)$ is equal to the value of the integral (6), that is $\mathbf{E}\theta_j = \int_{U^d} F_j(\mathrm{t})d\mathrm{t}$. This result allows to define a randomized algorithm called Monte Carlo Algorithm based on Modified Sobol Sequences (MCA-MSS-1).

In [7] the probability error of the algorithm MCA-MSS-1 is analysed. It is proved that for integrands with continuous and bounded first derivatives, i.e. $F_j \in \mathbf{W}^1(L; U^d)$, where $L = \|F_j\|$, it holds $err(F_j, d) \leq c_d^{'} \|F_j\| N^{-\frac{1}{2}-\frac{1}{d}}$ and $r(F_j, d) \leq c_d^{''} \|F_j\| N^{-\frac{1}{2}-\frac{1}{d}}$, where the constants $c_d^{'}$ and $c_d^{''}$ do not depend on N.

3.2.2 Summary of the MCM-MSS-2. The second algorithm called MCA-MSS-2 is a modification of MCA-MSS-1 algorithm. It is proposed and analysed in [5]. It is assumed that $n = m^d$, $m \geq 1$. The unit cube U^d is divided into m^d disjoint sub-domains, such that they coincide with the elementary d-dimensional subintervals $U^d = \bigcup_{l=1}^{m^d} K_l$, where $K_l = \prod_{\nu=1}^{d} [a_\nu^{(l)}, b_\nu^{(l)})$, with $b_\nu^{(l)} - a_\nu^{(l)} = \dfrac{1}{m}$ for all $l = 1, \ldots, d$. In such a way in each d-dimensional sub-domain K_l there is exactly one $\Lambda\Pi_\tau$ point $\mathrm{x}^{(l)}$. Assuming that after shaking, the random point stays inside K_l, i.e., $\xi^{(l)}(\rho) = \mathrm{x}^{(l)} + \rho\omega^{(l)} \in K_l$ one may try to exploit the smoothness

of the integrand in case if the integrand F_j belongs to $\mathbf{W}^2(L;U^d)$. In each sub-domain K_j the central point is denoted by $\mathrm{s}^{(j)}$, where $\mathrm{s}^{(j)} = (s_1^{(j)}, s_2^{(j)}, \ldots, s_d^{(j)})$.

Suppose two random points $\xi^{(l)}$ and $\xi^{(l)'}$ are chosen, such that $\xi^{(l)}$ is selected during our procedure used in MCA-MSS-1. The second point $\xi^{(l)'}$ is chosen to be symmetric to $\xi^{(l)}$ according to the central point $s^{(l)}$ in each cube K_l. In such a way the number of random points is $2m^d$. One may calculate all function values $f(\xi^{(l)})$ and $f(\xi^{(l)'})$, for $l = 1, \ldots, m^d$ and approximate the value of the integral in the following way:

$$I(f) \approx \frac{1}{2m^d} \sum_{l=1}^{2N} \left[f(\xi^{(l)}) + f(\xi^{(l)'}) \right]. \tag{7}$$

This estimate corresponds to MCA-MSS-2.

In [5] it is proved that the Monte Carlo algorithm MCA-MSS-2 has an optimal rate of convergence for functions with continuous and bounded second derivative [4]. This means that if $F_j \in \mathbf{W}^2(L;U^d)$, where $L = \|F_j\|$, it holds $err(F_j, d) \leq \widetilde{c}_d' \, \|F_j\| \, N^{-\frac{1}{2}-\frac{2}{d}}$ and $r(F_j, d) \leq \widetilde{c}_d'' \, \|F_j\| \, N^{-\frac{1}{2}-\frac{2}{d}}$, where the constants c_d' and c_d'' do not depend on N.

Remark 31 *Note that both MCA-MSS-1 and MCA-MSS-2 have one control parameter, that is the radius ρ of the sphere of shaking. At the same time, to be able to efficiently use this control parameter one should increase the computational complexity. It happens because after shaking the random point may leave the multidimensional sub-domain, and one needs to check if the random point is still in the same sub-domain. It is clear that the procedure of checking if a random point is inside the given domain is a computationally expensive procedure when one has to deal with a large number of points.*

3.3 Stratified Symmetrised Monte Carlo Approach

The main idea of the stratified symmetrised Monte Carlo approach is to generate a random point $\xi^{(l)} \in K_l$ uniformly distributed inside K_l and after that take the symmetric point $\xi^{(l)'}$ according to the central point $s^{(l)}$. Such a completely randomized approach simulates the concept of MCA-MSS-2 algorithm, but the *shaking* is with different radiuses ρ in each sub-domain. This algorithm is called MCA-MSS-2-S, because this approach looks like the stratified symmetrised Monte Carlo. Obviously, MCA-MSS-2-S is less expensive than MCA-MSS-2, but the drawback is that there is not such a control parameter like the radius ρ, which can be considered as a parameter randomly chosen in each sub-domain K_j.

It is important to notice that all three above mentioned algorithms MCA-MSS-1, MCA-MSS-2 and MCA-MSS-2-S have optimal (unimprovable) rate of convergence for the corresponding functional classes, that is MCA-MSS-1 is optimal in $\mathbf{W}^1(L;U^d)$ and both MCA-MSS-2 and MCA-MSS-2-S are optimal in $\mathbf{W}^2(L;U^d)$.

3.4 Specific Modification of the Algorithms for Solving Integral Equations

In the current versions of MCA-MSS-1 and MCA-MSS-2 algorithms the following modifications of the algorithms have been made:

- When the function from the desired linear functional is the δ-function, the dimension of the corresponding quasirandom and pseudorandom sequences coincides with the number of transitions in Liouville-Neumann series. For any other function, the dimension is larger than the number of transitions by one.
- Minimal distance between quasirandom points is calculated to compute each of the integrals with a different dimension. This procedure during the preprocessing approximation of the integrals increases the total computational time for MCA-MSS-1 and MCA-MSS-2 algorithms.
- We do one additional check: it is verified not only if the generated pseudorandom point lies inside the corresponding integration domain, but also if it lies inside the corresponding elementary multidimensional interval. Usually some points are rejected and that is why the total number of samples corresponding to MCA-MSS-2 is smaller than the number of samples for MCA-MSS-1 for some prescribed fixed values of coefficient ratio.
- One important feature of MCA-MSS-2 is that it is efficient when the number of divisions is the same for all directions, i.e. the integration domain is divided into multidimensional subcubes.

3.5 Error Analysis of Biased Stochastic Approaches

So far we have considered only biased stochastic approaches. We analyse the problem for approximate calculation of linear functional of the solution of integral equation (φ, u). Let us denote the i-th iterative approximation of $u\,(i \geq 0)$ by $u^{(i)}$ and the Monte Carlo approximation by $\tilde{u}$. It is known that two errors systematic one (a truncation error) R_{sys} and stochastic (a probabilistic) one, R_N appear. Actually, we approximate the truncated Liouville-Neumann series. If ε is a given sufficiently small positive parameter then

$$|(\varphi, u) - (\varphi, \tilde{u})| \leq |(\varphi, u) - (\varphi, u^{(i)})| + |(\varphi, u^{(i)}) - (\varphi, \tilde{u})| = \varepsilon_i + \varepsilon_N < \varepsilon,$$

where ε_i is the truncation error, ε_N is the probability error. We can obtain a lower bound for the number i of iterations using [2]:

$$i > \frac{\ln \dfrac{\varepsilon}{2|(\varphi, u^{(0)}) - (\varphi, u)|}}{\ln \|\mathcal{K}\|_{L_2}},$$

where $u^{(0)}$ is the initial approximation of u. Using the Cauchy-Bunyakowski inequality it is easy to show that

$$|(\varphi, u^{(0)}) - (\varphi, u)| \leq \|\varphi\|_{L_2(\Omega)} \|u^{(0)} - u\|_{L_2(\Omega)}.$$

The second multiplier can be estimated:

$$\|u^{(0)} - u\|_{L_2(\Omega)} \leq (\|I\|_{L_2} + \|\mathcal{K}\|_{L_2} + \ldots + \|\mathcal{K}\|^i_{L_2} + \ldots)\|r^{(0)}\|_{L_2(\Omega)},$$

$$\text{where} \qquad r^{(j)} = f - u^{(j)} - \mathcal{K}u^{(j)}, \;\; j = 0, 1, \ldots.$$

Taking the limit for $i \to \infty$ one can obtain

$$\|u^{(0)} - u\|_{L_2} \leq \frac{\|r^{(0)}\|_{L_2(\Omega)}}{1 - \|\mathcal{K}\|_{L_2}}.$$

It holds if the condition $\|\mathcal{K}\|_{L_2} < 1$ is fulfilled.

4 Unbiased Stochastic Approach

In this section an Unbiased Stochastic Approach (USA) will be discussed and analysed. The unbiased approach is based on the presentation of the functional under consideration as an infinite series. We are following a similar strategy like in our previous work [8] accept that now the state space is continuous and the transition density function, as well as the probability of absorbtion is no more discrete. In this case the random trajectory (Markov chain) T_i introduced in Section 3, subsection 3.1 is infinite. The USA method is a generalization to integral equations of the method developed in [8] for solving systems of linear algebraic equations. The discrete Markov chain used in [8] is replaced by a continuous one with transition probabilities depending on the kernel $k(\mathrm{x}, \mathrm{x}')$. In subsection 4.1 we describe the case $0 \leq k(\mathrm{x}, \mathrm{x}') \leq 1$, where $1 - k(\mathrm{x}, \mathrm{x}')$ is an absorbtion rate. In subsection 4.2 we develop the USA algorithm for more general kernels, where the above condition may not be fulfilled.

4.1 Unbiased approach for a simplified problem

We start with a simple case assuming that $0 \leq k(\mathrm{x}, \mathrm{x}') \leq 1$ for any $\mathrm{x}, \mathrm{x}' \in \Omega \equiv [0, 1]^d$, $\mathrm{x} = (x_1, \ldots, x_d)$, $\mathrm{x}' = (x'_1, \ldots, x'_d)$.

Assume that $Y \equiv U[0, 1]^d$ is a uniformly distributed random variable in $[0, 1]^d$. Then we use the following stochastic presentation:

$$u(\mathrm{x}) = E\{k(\mathrm{x}, Y)u(Y) + (1 - k(\mathrm{x}, Y))0\} + f(\mathrm{x}),$$

which suggests a representation for the function $u(\mathrm{x})$ based on the same ideas used in [8] for solving linear systems of equations. The probability for absorbtion is $1 - k(\mathrm{x}, Y)$, the score is the function $f(\mathrm{x})$, and if the trajectory is not absorbed then it continues at point Y. We consider a random trajectory (Markov chain) T_i of length i starting from state $X = \mathrm{x} = \mathrm{x}_0$ in the domain Ω assuming that $\partial \equiv \mathbb{R}^d \setminus \Omega$, i.e.:

$$T_i : \mathrm{x}_0 \longrightarrow \mathrm{x}_1 \longrightarrow \ldots \longrightarrow \mathrm{x}_i,$$

where

$$P(\mathrm{x}_{i+1} \in [a, b]/\mathrm{x}_i \neq \partial) = k(\mathrm{x}, U_i)P(U_i \in [a, b]),$$

$$P(\mathrm{x}_{i+1} = \partial / \mathrm{x}_i \neq \partial) = 1 - k(\mathrm{x}, U_i)$$

and

$$P(\mathrm{x}_{i+1} = \partial / \mathrm{x}_i = \partial) = 1, \quad \text{and} \quad \mathrm{f}(\partial) = 0.$$

Then we have:

$$u(\mathrm{x}) = E\left(\sum_{i=0}^{\infty} f(\mathrm{x}_i)/\mathrm{x}_0 = \mathrm{x}\right) = E\left(\sum_{i=0}^{\tau} f(\mathrm{x}_i)/\mathrm{x}_0 = \mathrm{x}\right), \text{ where } \tau = \inf_{\mathrm{i} \geq 0}(\mathrm{x_i}) = \partial.$$

Obviously, one can write

$$\begin{aligned} u(\mathrm{x}) = E\left(\sum_{i=0}^{\infty} f(\mathrm{x}_{i+1})/\mathrm{x}_0 = \mathrm{x}\right) + f(\mathrm{x}) &= E\left(E\left(\sum_{i=0}^{\infty} f(\mathrm{x}_{i+1})/\mathrm{x}_1\right)\right) + f(\mathrm{x}) \\ &= \int_{\Omega} k(\mathrm{x}, \mathrm{y}) u(\mathrm{y}) d\mathrm{y} + f(\mathrm{x}) \end{aligned} \tag{8}$$

The algorithm in this simplified case may be described as follows.

Algorithm 41 *:*

Unbiased stochastic algorithm for $0 \leq k(\mathrm{x}, y) \leq 1, \quad \mathrm{x}, \mathrm{y} \in [0,1]^d$

1. *Initialization* ***Input*** *initial data: the kernel* $k(\mathrm{x}, \mathrm{y})$*, the function* $f(\mathrm{x})$*, and the number of random trajectories* N*.*
2. *Calculations:*
   - *2.1.* ***Set*** *score=0*
   - *2.2.* ***Set*** $\mathrm{x}_0 = \mathrm{x}, test = 1$

```
Do j = 1, N
  Do While (test ≠ 0)
    score = score + f(x_0)
    U = rand[0,1]^d,   V = rand[0,1]
    If k(x_0, U) < V then
      test = 0 else
      x_0 = U
    Endif
  Endwhile
Enddo
```

3. *Compute the solution:*
   $u(\mathrm{x}) = \frac{score}{N}$

4.2 Unbiased approach for the general problem

Now we may describe the method for the case when the kernel $k(\mathrm{x}, \mathrm{y})$ may be negative and also we allow that $|k(\mathrm{x}, \mathrm{y})| \geq 1$, but still $k(\mathrm{x}, \mathrm{y}) \in L_2(\Omega \times \Omega)$. If $-1 \leq k(\mathrm{x}, Y) \leq 0$, then $k(\mathrm{x}, Y)$ is not any more the probability to stop. Instead, $|k(\mathrm{x}, Y)|$ is used as a probability to stop. But now one needs to compute $-u(Y)$

instead of $u(Y)$. In this case, the score is $-f(\mathrm{x})$ instead of $f(\mathrm{x})$. If $|k(\mathrm{x},Y)| \geq 1$, then the random walk continues and the score is $k(\mathrm{x},Y)f(\mathrm{x})$. The above described procedure leads to the following method, which is a modification of the previous approach. Assume we deal with the Markov chain T_i of length i starting from state $X = \mathrm{x} = \mathrm{x}_0$ in the domain Ω assuming that $\delta \equiv \mathbb{R}^d - \Omega$:

$$T_i : \mathrm{x}_0 \longrightarrow \mathrm{x}_1 \longrightarrow \ldots \longrightarrow \mathrm{x}_i,$$

where in this case

$$P(\mathrm{x}_{i+1} \in [a,b]/x_i \neq \partial) = \min\left(|k(\mathrm{x}_i,\mathrm{x}_{i+1})|, 1\right),$$
$$P(\mathrm{x}_{i+1} = \partial/x_i \neq \partial) = 1 - \min\left(|k(\mathrm{x}_i,\mathrm{x}_{i+1})|, 1\right),$$

and

$$P(\mathrm{x}_{i+1} = \partial/\mathrm{x}_i = \partial) = 1.$$

Then we have:

$$u(\mathrm{x}) = f(\mathrm{x}) + E\left(\sum_{k=1}^{\infty}\prod_{i=1}^{k}\hat{k}(\mathrm{x}_{i-1},\mathrm{x}_i)f(\mathrm{x}_{i-1})/\mathrm{x}_0 = \mathrm{x}\right),$$

where $\hat{k}(\mathrm{x},\mathrm{x}')$ is defined in the following way:

$$\hat{k}(\mathrm{x},\mathrm{x}') = \begin{cases} k(\mathrm{x},\mathrm{x}') & \text{if} \quad |k(\mathrm{x},\mathrm{x}')| \geq 1 \\ 1 & \text{if} \quad 0 \leq k(\mathrm{x},\mathrm{x}') \leq 1 \\ -1 & \text{if} \quad -1 \leq k(\mathrm{x},\mathrm{x}') \leq 0 \end{cases}$$

Now, we may present the algorithm in this more general case:

Algorithm 42 *:*

Unbiased stochastic algorithm for the general case:

1. *Initialization* ***Input*** *initial data: the kernel* $k(\mathrm{x},\mathrm{y})$*, the function* $f(\mathrm{x})$*, and the number of random trajectories* N*.*
2. *Calculations:*
   - *2.1.* ***Set*** *score=0*
   - *2.2.* ***Do*** $j = 1, N$
   - *2.3.* ***Set*** $\mathrm{x}_0 = \mathrm{x}, test = 1$*, prod=1*

```
Do While (test ≠ 0)
score = score + f(x_0) * prod
U = rand(0,1)^d
  If |k(x_0,U)| > 1 then
    prod = prod * k(x_0,U)
  else x_0 = U
  V = rand[0,1]^d
  If |k(x_0,U)| < V then
    test = 0 else
    prod = prod * sign (k(x_0,U)),   x_0 = U
  Endif
Endwhile
Enddo
```

3. Compute the solution:
$u(\mathrm{x}) = \frac{score}{N}$

4.3 Some extensions of the unbiased approach

4.3.1 Weak approximation In many situations, like for instance in sensitivity analysis [5], one is not necessary interested in pointwise value of the solution but in mean values $\int_\Omega \varphi(\mathrm{x})u(\mathrm{x})d\mathrm{x}$. In some applications the given function $\varphi(\mathrm{x})$ may be considered as a weight function. Often the support of $\varphi(\mathrm{x})$ is a relatively *small* subdomain of Ω.

We can still use a double randomization technique for computing such a quantity. If $\varphi(\mathrm{x})$ is a probability density function defined on Ω, then the integral under consideration can be presented as a mean value of $u(\mathrm{x})$, i.e., $\int_\Omega u(\mathrm{x})\varphi(\mathrm{x})d\mathrm{x} = E\left(u(Y)\right)$, where Y is a random variable which admits $\varphi(\mathrm{x})$ as a density function. This will only lead to a slight modification of the algorithm. More precisely, the starting point x_0 is the random variable Y. It is not always easy to sample from φ, more importantly, φ is not always a density function. There is another possibility that can be used if φ is bounded. One can pick the starting point x_0 uniformly distributed at random and multiply the sum of the values of the random variable (the score) by $\varphi(\mathrm{x}_0)$.

4.3.2 Global approximation Another important extension is the possibility to compute the solution of the Fredholm integral equation of second kind in all points of the domain Ω using the results of the same unbiased USA algorithm. One should take into account that during the random walk many points of the domain are visited. The information obtained during this process can be used (after a proper statistical treatment) to obtain an approximation of the solution at all point of the domain. The following procedure may be used to do that:

(i) The starting point x_0 is picked up at random uniformly.

(ii) Because of that, all the visited points among all the random trajectories are also uniformly distributed. One may state that for each visited point we obtain a score, which mean value is an unbiased estimate of the solution at this point.

(iii) We may express the approximation of the solution at any point x by using the Taylor expansion. We assume that we have values of the solution in a neighborhood of x of distance h at points $\mathrm{x}_i,\ i = 1, \ldots, N_h$ and $\mathrm{h} \in \mathbf{R}^d$ is a d-dimensional vector, such that $\mathrm{h} = \mathrm{x}_i - \mathrm{x}$:

$$\begin{aligned} u(\mathrm{x}_i) = u(\mathrm{x} + \mathrm{h}_i) &+ \nabla u(\mathrm{x})\ \mathrm{h}_i \\ &+ \tfrac{1}{2}(\mathrm{h})^T\ [D^2 u(\mathrm{x})]\mathrm{h} + O(h^3), \end{aligned}$$

where $\nabla u(\mathrm{x}) = \left[\dfrac{\partial u(\mathrm{x})}{\partial x_1}, \ldots, \dfrac{\partial f(\mathrm{x})}{\partial x_d}\right]$ is the gradient, $[D^2 u(\mathrm{x})] = \left[\dfrac{\partial^2 u(\mathrm{x})}{\partial x_i \partial x_k}\right]_{i,k=1}^{d}$ is the Hessian matrix, and $(\mathrm{h})^T$ means transposed vector h. Obviously, the mean

value of the solution at points x_i may be expressed in the following way:

$$\frac{1}{N_h}\sum_{i=1}^{N_h} u(\mathrm{x}_i) = u(\mathrm{x}) + \frac{1}{N_h}\sum_{i=1}^{N_h} h_i \nabla u(\mathrm{x}) + \frac{1}{N_h}\sum_{i=1}^{N_h} \frac{h_i^2}{2}[D^2 u(\mathrm{x})] + O(h^3), \qquad (9)$$

where the Hessian matrix may be estimated from above by the maximum of all second derivatives at point x.

Obviously, it is important to chose the value of h, the size of the d-dimensional domain $A^N_{h,\mathrm{x}}$ around the point x in which we want to compute the solution. The value h should be small enough, such that the second order term in the Taylor expansion is relatively small. At the same time, the number of points lying in $A^N_{h,\mathrm{x}}$ should not be too small, since than the variance of the first order term is too big. The mathematical expectation of the first order term in (9) is zero, and its variance depends on the number of point as $O(\frac{1}{N_h})$. We can now estimate the value $u(\mathrm{x})$ by the mean value $\frac{1}{N_h}\sum_{i=1}^{N_h} u(\mathrm{x}_i)$.

5 Numerical Results and Discussion

Numerical results are presented and discussed within two subsections. In the first one we are focussing on the comparison of the three classes of methods described above, namely Markov chain MC for integral equations, computing finite number of integrals, and the newly proposed unbiased stochastic algorithm. In the second subsection we do further investigation of the USA algorithm.

5.1 Comparison of the different algorithms

The numerical algorithms are tested on the following example [9]:

$$u(x) = \int_{\Omega} k(x,x')u(x')\mathrm{d}x' + f(x), \qquad \Omega \equiv [0;1] \qquad (10)$$

where

$$k(x,x') = x^2 \mathrm{e}^{x'(x-1)}, \qquad (11)$$

$$f(x) = x + (1-x)\mathrm{e}^x, \qquad (12)$$

$$\varphi(x) = \delta(x - x_0). \qquad (13)$$

The solution of this test problem is $u(x) = e^x$. We are interested in an approximate calculation of (φ, u), where $\varphi(x) = \delta(\mathrm{x} - \mathrm{x}_0),\ x_0 = 0.5$.

5.1.1 Numerical experiments We have performed the following biased stochastic algorithms: MC algorithm and qMC based on Sobol sequences for integral equations and several algorithms for integrals into which the problem of solving integral equations is transformed. For the above set of algorithms we use Crude MCA, quasi-Monte Carlo algorithm based on $\Lambda\Pi_\tau$ Sobol quasirandom sequences, MCA-MSS-1, MCA-MSS-2, and MCA-MSS-2-S.

The algorithms have been studied after 10 runs to average the final approximation and for various sample sizes chosen according to proper sample size for Sobol quasirandom sequences. The samples number is defined by the requirement for error balancing following the approach given in Section 3, subsection 3.5. The number of iterations i/d is fixed, but it is chosen according to the L_2-norm of the kernel $k(\mathrm{x}, \mathrm{x}')$. For approximate computation of any integral $I(j),\ j = 0, \ldots, i$ different number of samples are chosen to satisfy the error balancing requirements.

Table 1. Relative errors and computational time for computing $u(x_0)$ at point $x_0 = 0.5$ using MCM for integral equations.

i	ε	N	Crude MC		Sobol seq.	
			Rel. err.	Time (s)	Rel. err.	Time (s)
1	0.4	128	0.05960	< 0.0001	0.05916	< 0.0001
2	0.14	1024	0.00566	< 0.0001	0.00670	< 0.0001
2	0.08	4 096	0.00294	0.01	0.00712	< 0.0001
3	0.02	32 768	0.00092	0.17	0.00204	0.01
4	0.018	65 536	0.00076	0.49	0.00096	0.03

Table 2. Relative errors and computational time for computing $u(\mathrm{x}_0)$ as a finite number (i) of integrals from the Liouville-Neumann series at $x_0 = 0.5$ using Crude MCM.

i	N	Integral equation				FNI	
		Sobol seq.		Crude MC		Sobol seq.	Crude MC
		Rel. err.	Time (s)	Rel. err.	Time (s)	Rel. R_N	Rel. R_N
1	128	0.05158	< 0.0001	0.05197	< 0.0001	0.00024	0.00016
2	1024	0.01439	< 0.0001	0.01405	< 0.0001	2e-05	0.00033
2	4 096	0.01438	< 0.0001	0.01422	< 0.0001	5e-06	0.00015
3	32 768	0.00399	0.01	0.00404	0.11	1e-06	4e-05
4	65 536	0.00111	0.03	0.00108	0.63	1e-06	3e-05

Because of the bias of the first two classes of algorithms there is a reason to present the relative errors, as well as, the approximation of the corresponding Liouville-Neumann series with a fixed length in Table 2, Table 3, and Table 4,

Table 3. Relative errors and computational time for computing $u(\mathrm{x}_0)$ as a finite number of integrals from the Liouville-Neumann series at $x_0 = 0.5$ using MCA-MSS-1 and MCA-MSS-2 algorithms for integrals.

i	N	ρ	MCA-MSS-1					MCA-MSS-2		
			Int. eq.			FNI		Int. eq.		FNI
			Sobol	CMCM	Time	Sobol	CMCM	CMCM		
			Rel. err.	Rel. err.	(s)	Rel. R_N	Rel. R_N	Rel. R_N	Time (s)	Rel. R_N
1	128	2e-03	0.0516	0.0515	<1e-04	0.0003	0.0004	0.0514	<1e-04	0.0005
2	1017	2e-04	0.0144	0.0144	0.02	2e-06	2e-06	0.0144	0.02	2e-05
2	4 081	6e-05	0.0144	0.0144	0.14	2e-06	3e-06	0.0144	0.16	7e-06
3	32 749	8e-06	0.0040	0.0040	10.4	8e-07	8e-07	0.0040	10.73	1e-06
4	65 521	4e-06	0.0011	0.0011	54.9	1e-07	1e-07	0.0011	55.4	2e-07

as well. We have used symbolic computations to determine the values of the systematic error R_{sys} for number of iterations up to 3. In Table 6 we present the computed values for the systematic error R_{sys} at point $x_0 = 0.1$ and $x_0 = 0.5$ just to have an idea about the magnitude of this kind of error. One can clearly see that the systematic error decreases when the number of iterations increase. At the same time, for such small number of iteration i the systematic error dominates in almost all cases. The reason for that is that the analysis of balancing of stochastic and systematic errors are done following the assumption that Crude MC method is used. Actually, the applied algorithms, especially randomized quasi-Monte Carlo algorithms MCA-MSS-1, MCA-MSS-2 based on *shaking* of Sobol points are much higher quality and their stochastic error R_N is much smaller than the systematic error R_{sys}. Some typical values of R_N are, say 10^{-6} for relatively small number of random trajectories. These algorithms are very efficient when the norm of the kernel is relatively small. Then one may increase (but not too much) the number of iterations i, which will allow for a small number of N (and low computational complexity) to have a relatively high accuracy.

One should take into account that the choice of sample size to compute integrals defined on subdomains (IDSs) for the algorithm MCA-MSS-2-S is a crucial question. The IDSs have different dimensionality and this fact affects the corresponding sample size. The following strategy is applied here: first, the number N_i is chosen to be approximately equal to 2^k (to give a possibility for a fair comparison with results obtained using Sobol quasirandom sequences) and to be close to the i-th power of an integer. We need this because the integration domain is the i-th dimensional unit cube. Then, $N_1 := N_i$, $N_l < N_i$, $l = 2, 3, \ldots, i-1$, and N_l is the l-th power of an integer that is the number of sunintervals on each direction. Finally, we have chosen to compute each IDS with the same number of samples. It defines serious restrictions about the sample size

Table 4. Relative errors and computational time for computing $u(\mathrm{x}_0)$ as a finite number (i) of integrals from the Liouville-Neumann series at $x_0 = 0.5$ using MCA-MSS-2-S algorithm for integrals.

i	N	Integral equation		Finite number of int, FNI
		Crude MC		Crude MC
		Rel. err.	Time (s)	Rel. R_N
1	128	0.05182	< 0.0001	4e-09
2	1 024	0.01437	< 0.0001	6e-08
2	4 096	0.01437	0.01	2e-08
3	46 656	0.00399	0.31	2e-08
4	4 096	0.00111	0.04	1e-06
4	531 441	0.00111	5.97	7e-09

applied during the numerical experiments. Nevertheless, we succeeded to achieve the goal of studying properties of MCA-MSS-2-S for solving integral equations based on the computation of a finite number of integrals.

5.1.2 Discussion and applicability The tendencies concerning the behavior of biased MC algorithms here are similar to the tendencies observed in [5]. The following observations can be drawn based on the results of our computations:

- MCA-MSS-2 algorithm gives a slightly larger relative stochastic error R_N in comparison with MCA-MSS-1 related to the linear functional under consideration, and smaller relative stochastic error (almost 10 times smaller for the most cases) related to the finite number of integrals.
- MCA-MSS-2-S algorithm gives the smallest relative stochastic error and the shortest computational time for a fixed number of steps in Liouville-Neumann series in comparison with MCA-MSS-1 and MCA-MSS-2, but serious restrictions about the choice of sample size exist for its implementation.
- The main disadvantage of MCA-MSS-1 and MCA-MSS-2 algorithms is the high computational complexity due to computing the minimal distance between the generated original Sobol sequences.
- MCM-int. eq. algorithm gives the smallest relative errors and corresponding to the problem of approximation of a linear functional of the integral equation (10) in comparison with other biased MC algorithms and similar computing time to the Crude MC and MCA-MSS-2-S algorithms.
- MCM-int. eq. has smaller relative errors in comparison with unbiased USA algorithm especially for smaller sample sizes $(512, 1024)$ or similar relative errors for larger ones.

Table 5. Relative errors and computational time for computing $u(\mathrm{x}_0)$ at different initial points using unbiased USA for integral equations.

N	$x_0 = 0.5$			$x_0 = 0.99$		
	i	Rel. err.	Time (s)	i	Rel. err.	Time (s)
128	1.3	0.02618	0.001	2.38	0.00237	0.001
512	1.28	0.01160	0.002	2.39	0.00714	0.002
1 024	1.28	0.01020	0.001	2.38	0.00199	0.005
2 048	1.27	0.00422	0.006	2.38	0.00213	0.005
16 384	1.27	0.00142	0.024	2.38	9**e** -06	0.044
65 536	1.27	0.00075	0.096	2.37	0.00041	0.142
131 072	1.27	0.00037	0.154	2.37	0.00024	0.281
262 144	1.27	0.00014	0.305	2.37	0.00026	0.558
4 194 304	1.27	0.00003	4.888	2.37	0.00011	8.802

Table 6. Relative systematic error R_{sys} computed for three number of iterations $i = 1, 2, 3$ at two points x_0

x_0	$i = 1$	$i = 2$	$i = 3$
$x = 0.1$	2.28 e-03	0.635 e-03	0.176 e-03
$x = 0.5$	51.8 e-03	14.3 e-03	3.99 e-03

- USA algorithm is the fastest and the most reliable for larger sample sizes than all other algorithms under consideration. The reason for that could be the fact that the algorithm is unbiased and also the probability for absorbtion of the random trajectory is chosen to depend on the problem data, namely, $k(\mathrm{x}, \mathrm{x}')$ and $f(\mathrm{x})$.
- The average number of the steps of the unbiased USA algorithm is smaller than 2 due to the fast convergence because i) the initial point at each chain is 0.5, and ii) the most of the kernel values are ranged in $(0,1; 0,3)$ in this case. Just to make a comparison: for MC algorithms $2, 3, 4$ jumps are necessary to reach the prescribed accuracy (the L_2 norm of the kernel is approximately equal to 0.3, i.e. one can expect a fast convergent iterative process).
- USA algorithm behavior has been studied for other initial points, especially for $x_0 = 0.99$; the average number of iteration steps for USA algorithm increased and was approximately equal to 2 (more precisely, it is around 2.4) for all sample sizes considered.

5.2 Further investigation of the USA algorithm

There are two parts in this subsection. In the first one we study a new example, to show how to handle the different difficulties described in Subsection 4.3. In the second part we compute a global approximation of the solution of the first example (10).

5.2.1 General USA algorithm In order to validate the general version of the unbiased algorithm we consider a *second numerical example*, where the kernel can take negative values and also its absolute value can be greater than 1:

$$u(x) = \int_0^1 k(x, x')u(x')\mathrm{d}x' + f(x), \tag{14}$$

where

$$k(x, x') = \beta(2x' - 1)x^2 \mathrm{e}^{x'(x-1)}, \tag{15}$$

$$f(x) = \mathrm{e}^x + \beta(2\mathrm{e}^x - 2 - x\mathrm{e}^x - x). \tag{16}$$

$$\varphi(x) = \delta(x - x_0), \tag{17}$$

where x_0 take values 0.1, 0.5, and 0.99 in different numerical experiments. The solution of the above equation is still $u(x) = \mathrm{e}^x$. We use special parameter β to be able to deal with various difficulties, namely, with negative kernels and values of the kernel greater then one in absolute value. If the values of the parameter β increases, then we approach instability barrier when the variance is getting infinite.

Following the idea in subsection 4.3, Section 4 we also compute two quantities:

$$I_1 = \int_0^1 u(x)dx, \tag{18}$$

as well as

$$I_2 = \int_0^1 2xu(x)dx \tag{19}$$

to illustrate our new unbiased approach.

For the first quantity the initial value is just picked up uniformly, while for the second one we have two possibilities:

(i) pick the starting point x_0 according to the density $2x$; this leads to a random variable distributed as $\sqrt{U}$ (U is a uniformly distributed random variable) using the well-known inversion method;

(ii) one can also start uniformly at random from the point x_0 and multiply the sum of values of the random variable (score) by $2x_0$.

Some results for the numerical solution of the second example (14)–(17) are presented in Table 7. All simulations are done for a fixed number of random trajectories $N = 10^6$. One can see, that for each point $x_0 = 0.1,\ 0.5,\ 0.99$ the standard deviation and the error increases with β. This result is in full agreement with the theory, since when the parameter β increases the norm of the kernel

Table 7. Relative error (Err.), standard deviation (σ) and mean value of steps of absorbtion i for the USA approach for various values of β; $N = 10^6$.

	$\beta = 1$			$\beta = 2$			$\beta = 4$		
	Err.	σ	i	Err.	σ	i	Err.	σ	i
$x = 0.1$	7e-05	0.105	1.003	1e-05	0.17	1.009	9e-04	0.62	1.02
$x = 0.5$	2e-04	0.600	1.116	1e-04	0.97	1.27	0.004	8.43	1.63
$x = 0.99$	5e-04	1.388	1.588	3e-04	2.76	2.05	0.008	32.9	2.48
$\bar{U}$ meth.1	6e-04	0.900	1.178	8e-04	1.39	1.38	0.012	11.65	1.70
I_2 meth.(i)	1e-03	1.95	1.176	1e-03	2.63	1.38	0.015	20.5	1.70
I_2 meth.(ii)	2e-04	1.03	1.27	8e-04	1.70	1.58	0.013	14	2

$k(\mathrm{x}, \mathrm{x}')$ is getting closer and closer to 1. That is why for high values of β we can observe some numerical instability.

In the second part of Table 7 the results obtained for another functional of the solution, namely I_2 (see, (19)) by applying different techniques of sampling: the first example is when we sample uniformly $\bar{U}$, the next example is when we pick the starting point x_0 according to the density $2x$ (see, approach (i) described above), and the last example is when we start uniformly at random from the point x_0 and multiply the sum of values of the random variable (score) by $2x_0$ (see, approach (ii) described above). One can see from the results presented in the table that approach (ii) has outperformed approach (i). These results give a sort of hint how to chose the sampling technique. The results also show that with the increasing of β when we approach to the limit of the convergency the relative error and standard deviation are getting bigger and bigger.

5.2.2 Global approximation We are now dealing with the algorithm described in Subsection 4.3 of Section 4 to compute an approximation in all the points of the domain. It is also important to see how the accuracy improves when the number of random trajectories N increases if we apply our approach for computing the solution in all the points of the domain. Results for the relative error, standard deviation σ and mean value of number of steps i in each random trajectory are presented in Table 8.

The numerical results show that the relative error decreases with the increasing the number of random trajectories N according to the expectations. When the unbiased USA algorithm is used for computing the values of the solution in all the points of the domain, the following observations can be drown:

- when the solution is computed at any point x by using the Taylor expansion the accuracy is satisfactory; the relative error increases as expected, but it is still small enough;

Table 8. Relative error, standard deviation and mean value of steps of absorbtion i for the USA approach for various values of number of random trajectories N.

	σ	i	$N = 10^4$	$N = 10^5$	$N = 10^6$	$N = 10^7$
$x = 0.1$	0.14	1.009	3e-04	6e-04	5e-05	4e-05
$x = 0.5$	0.76	1.27	3e-03	1e-03	7e-04	1e-05
$x = 0.99$	0.92	2.37	2e-03	1e-03	8e-04	8e-05
$h = 0.2$	0.91	1.41	5e-03	6e-03	7e-03	6e-03
$h = 0.1$			8e-03	2e-03	2e-03	1.5e-03
$h = 0.05$			1e-02	4e-04	8e-04	7e-05

- if the parameter h is too big ($h = 0.2$), then the procedure *smoothes* the information obtained from the points where we have an unbiased estimation for the solution; that is why the increase of the number of random trajectories does not improve the accuracy; it means that the information which is used is too *integrated*;
- for relatively small values of h, say $h = 0.05$, when N is small we have very small number of points in which we have unbiased estimates of the solution; the information is fragmentized, and that is why the relative error is big; with increasing the number of random trajectories, the information used is getting more *integrated*, and the error decreases significantly.

6 Conclusions and Future Plans

A new unbiased stochastic method for solving Fredholm integral equations of second kind was proposed and analysed. We have studied three possible approaches to compute linear functionals of the solution of integral equation under consideration:

- biased Monte Carlo method based on evaluation of truncated Liouville-Neumann series,
- transformation of this problem into the problem of computing a finite number of integrals, and
- unbiased stochastic approach.

Five Monte Carlo algorithms for numerical integration have been applied in the second case, namely

- Crude Monte Carlo method based on a high quality SIMD-oriented Mersenne Twister pseudorandom number generator,
- Quasi-Monte Carlo based on $\Lambda\Pi_\tau$ Sobol quasirandom sequences,
- a stratified symmetrised MC algorithm MCA-MSS-2-S,

– a randomized Quasi-Monte Carlo based on a special procedure of *shaking* $\Lambda\Pi_\tau$ Sobol quasirandom points with a convergence rate $O\left(N^{-\frac{1}{2}-\frac{1}{d}}\right)$, and
– a randomized Quasi-Monte Carlo with a convergence rate $O\left(N^{-\frac{1}{2}-\frac{2}{d}}\right)$.

We have shown that the procedure of balancing of both stochastic R_N and systematic R_{sys} errors is very important for the quality of the biased algorithms.

In almost all numerical experiments performed, the unbiased USA algorithm performs with relatively high accuracy and low computational complexity in comparison with the best available biased algorithms. The numerical examples studied in this work are one-dimensional since we are comparing Monte Carlo algorithms only. It is obvious that the same algorithms can be used in any fixed dimension. However, the algorithms based on evaluation of finite number of integrals will suffer more from the effect of high dimensionality, because they are based on quadrature points.

As a further step, we would like to focus on the analysis of the variance of the unbiased USA algorithm, especially to study how the *unbalancing* of the kernel may reflect to the accuracy of the results. It is also important to study the computational complexity of USA in terms of number of floating point operations, and how the complexity depends on the problem's parameters. Some comparison of the stochastic approaches under consideration with known deterministic algorithms could be also considered.

Acknowledgment

This work has been supported by the EC FP7 Project AComIn (FP7-REGPOT-2012-2013-1), by ISITV, Université de Sud Toulon-Var, as well as by the Bulgarian NSF Grants DMU 03/61/2011 and DCVP 02/1/2010. We would like to thank Dr. Mehmet Ersoy who helped us with the symbolic computations.

References

1. P. Bradley, B. Fox, Algorithm 659: Implementing Sobol's quasi random sequence generator, ACM Trans. Math. Software 14(1) (1988) 88-100.
2. J. H. Curtiss. Monte Carlo methods for the iteration of linear operators, J. Math Phys., vol. 32, 209-232, 1954.
3. I. Dimov. Efficient and overconvergent Monte Carlo Methods. Parallel algorithms., *Advances in Parallel Algorithms, I.Dimov, O.Tonev (Eds.), Amsterdam, IOS Press*, 100–111, 1994.
4. I. Dimov. Monte Carlo methods for applied scientists., *World Scientific*, New Jersey, London, Singapore, World Scientific, 2008, 291p., ISBN-10 981-02-2329-3 (monograph).
5. I. T. Dimov, R. Georgieva. Multidimensional Sensitivity Analysis of Large-scale Mathematical Models. O.P. Iliev et al. (eds.), Numerical Solution of Partial Differential Equations: Theory, Algorithms, and Their Applications, *Springer Proceedings in Mathematics and Statistics*, 45, Springer Science+Business Media, New York, 137–156, 2013. ISBN: 978-1-4614-7171-4 (book chapter).

6. I.T. Dimov, R. Georgieva, Monte Carlo method for numerical integration based on Sobol' sequences, in: LNCS 6046, Springer, 2011, 50-59.
7. I. T. Dimov, R. Georgieva, Tz. Ostromsky, Z. Zlatev, Advanced algorithms for multidimensional sensitivity studies of large-scale air pollution models based on Sobol sequences, Computers and Mathematics with Applications 65 (3), "Efficient Numerical Methods for Scientific Applications", Elsevier, 2013, pp. 338-351. ISSN: 0898-1221
8. I.T. Dimov, S. Maire, J.M. Sellier, A New Walk on Equations Monte Carlo Method for Linear Algebraic Problems, preprint hal-00979044, 2014.
9. R. Farnoosh, M. Ebrahimi. Monte Carlo Method for Solving Fredholm Integral Equations of the Second Kind, *Applied Mathematics and Computation*, 195 309–315, 2008.
10. R. Georgieva, S. Ivanovska. Importance Separation for Solving Integral Equations. - In: Proceedings of Large-Scale Scientific Computing 2003 (I. Lirkov, S. Margenov, J. Wasniewski, and P. Yalamov - Eds.), LNCS 2907, Springer, 144–152, 2004.
11. Kalos, Malvin H.; Whitlock, Paula A. (2008). Monte Carlo Methods. Wiley-VCH. ISBN 978-3-527-40760-6.
12. H. Niederreiter, Low-discrepancy and low-dispersion sequences, Journal of Number Theory 30 (1988) 51-70.
13. M. Saito, M. Matsumoto, SIMD-oriented fast Mersenne Twister: a 128-bit pseudorandom number generator, in: Keller, A., Heinrich, S., Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer (2008) 607-622.
14. I. Sobol, D. Asotsky, A. Kreinin, S. Kucherenko, Construction and comparison of high-dimensional Sobol' generators, Wilmott Journal (2011) 67-79.
15. I. M. Sobol. *Monte Carlo Numerical Methods*, Nauka, Moscow, 1973, (in Russian).
16. I. M. Sobol, On quadratic formulas for functions of several variables satisfying a general Lipschitz condition, USSR Comput. Math. and Math. Phys. 29(6) (1989) 936-941.
17. H. Weyl, Ueber die Gleichverteilung von Zahlen mod Eins. Math. Ann. 77(3) (1916) 313-352.
18. www.math.sci.hiroshima-u.ac.jp/∼ m-mat/MT/SFMT/index.html.