Optimal Monte Carlo Algorithms

Ivan T. Dimov
Institute for Parallel Processing
Department of Parallel Algorithms
Bulgarian Academy of Sciences
Acad. G. Bonchev St., 25 A
1113 Sofia, Bulgaria
and
ACET Centre
University of Reading
Whiteknights, PO Box 217, Reading, RG6 6AH, UK
E-mail: I.T.Dimov@reading.ac.uk; ivdimov@bas.bg
Web site: http://www.personal.rdg.ac.uk/ sis04itd/

Abstract

The question "what Monte Carlo can do and cannot do efficiently" is discussed for some functional spaces that define the regularity of the input data. Important for practical computations data classes are considered: classes of functions with bounded derivatives and Hölder type conditions.

Theoretical performance analysis of some algorithms with unimprovable rate of convergence is given. Estimates of complexity of two classes of algorithms – deterministic and randomized for the solution of a class of integral equations are presented.

Keywords: Monte Carlo algorithms, deterministic algorithms, integral equations, unimprovable rate of convergence.

1 Introduction: definitions and basic notations

The Monte Carlo method is a powerful tool in many fields of mathematics, physics and engineering. It is known that the algorithms based on this method give statistical estimates for any linear functional of the solution by performing random sampling of a certain random variable (r.v.) whose mathematical expectation is the desired functional. The Monte Carlo method is a method for solving problems using random variables. Usually Monte Carlo methods reduce problems to the approximate calculation of mathematical expectations. Let the variable J be the desired solution of the problem or some desired linear functional of the solution. A r.v. ξ with mathematical expectation equal to J must be constructed: $E\xi = J$. Using n independent values (realizations) of $\xi : \xi_1, \xi_2, \ldots, \xi_n$, an approximation to J

$$J \approx \frac{1}{n}(\xi_1 + \ldots + \xi_n), \tag{1}$$

can then be computed.

Monte Carlo numerical algorithms are usually used for solving *deterministic* problems by modeling random variables or random fields. It is clear, that in this case a statistical error appear. The error estimates are important issue in studying Monte Carlo algorithms. It should be mentioned here that one can only state that a certain Monte Carlo algorithm can produce the result with a given probability error. If the mean value of n realizations of the r.v. ξ is denoted by $\overline{\xi}_n = \frac{1}{n}\sum_{i=1}^{n} \xi_i$, then the following definition of the probability error can be given:

Definition 1.1 *If J is the exact solution of the problem, then the probability error is the least possible real number R_n, for which:*

$$P = Pr\left\{|\overline{\xi}_n - J| \leq R_n\right\}, \tag{2}$$

where $0 < P < 1$. If $P = 1/2$, then the probability error is call probable error denoted by r_n.

[1]Institute for Parallel Processing, Department of Parallel Algorithms, Bulgarian Academy of Sciences, Acad. G. Bonchev St., 25 A, 1113 Sofia, Bulgaria, and ACET Centre, University of Reading, Whiteknights, PO Box 217, Reading, RG6 6AH, UK E-mail: I.T.Dimov@reading.ac.uk; ivdimov@bas.bg, Web site: http://www.personal.rdg.ac.uk/ sis04itd/

So, dealing with randomized algorithms one has to accept that the result of the computation can be true only with a certain (even high) probability.

Definition 1.2 *A finite discrete Markov chain T_i is defined as a finite set of states $\{k_1, k_2, ..., k_i\}$.*

At each of the sequence of times $t = 0, 1, \ldots, i, \ldots$ the system T_i is in one of the following states k_j. The state k_j determines a set of conditional probabilities p_{jl}, such that p_{jl} is the probability that the system will be in the state k_l at the $(\tau + 1)^{th}$ time given that it was in state k_j at time τ. Thus, p_{jl} is the probability of the transition $k_j \Rightarrow k_l$. The set of all conditional probabilities p_{jl} defines a transition probability matrix $P = \{p_{jl}\}_{j,l=1}^{i}$, which completely characterizes the given chain T_i. In the general case, iterative Monte Carlo algorithms can be defined as *terminated Markov chains*:

$$T = k_{t_0} \to k_{t_1} \to k_{t_2} \to \ldots \to k_{t_i}, \tag{3}$$

where k_{t_q}, $(q = 1, \ldots, i)$ is one of the absorbing states. This determines the value of some function $F(T) = J(u)$, which depends on the sequence (3). The function $F(T)$ is a random variable. After the value of $F(T)$ has been calculated, the system is restarted at its initial state k_{t_0} and the transitions are begun anew. A number of n independent runs are performed through the Markov chain starting from the state s_{t_0} to any of the absorbing states. The average

$$\frac{1}{n} \sum_{T} F(T) \tag{4}$$

is taken over all actual sequences of transitions (3). The value in (4) approximates $E\{F(T)\}$, which is the required linear form of the solution.

We also will be interested in *performance analysis of algorithms*. The performance analysis deals with *computational cost* of the algorithms (see [12, 9]).

Definition 1.3 *Computational cost of a randomized iterative algorithm A^R is defined by*

$$cost(A, x, \omega) = nE(q)t_0,$$

where $E(q)$ is the mathematical expectation of the number of transitions in the sequence (3) and t_0 is the mean time needed for value of one transition.

The question: *what Monte Carlo can do and cannot do efficiently?* frequently arises among people dealing with numerical methods, scientific computing and applications of mathematics in theoretical and applied sciences. Often if there are two algorithms people are interested which one is *better*. What means *better* in the case you cannot get the exact solution of the problem and you are happy to have an ε-approximation to the true solution? Obviously, you will call *better* the algorithm, which produces the ε-approximation to the solution faster, or with a smaller number of operations.

The mathematical expectation of the r.v. θ is denoted by $E(\theta)$ (sometimes abbreviated to $E\theta$); the variance by $D(\theta)$ (or $D\theta$) and the standard deviation by $\sigma(\theta)$ (or $\sigma\theta$). We shall let γ denote the random number, that is a uniformly distributed r.v. in $[0, 1]$ with $E(\gamma) = 1/2$ and $D(\gamma) = 1/12)$. We shall further denote the values of the random point ξ or θ by ξ_i, $\theta_i (i = 1, 2, \ldots, n)$ respectively. If ξ_i is a d-dimensional random point, then usually it is constructed using d random numbers γ, i.e., $\xi_i \equiv (\gamma_i^{(1)}, \ldots, \gamma_i^{(d)})$. The density (frequency) function will be denoted by $p(x)$ and the transition density function by $p(x, y)$. $F(x)$ will denote the distribution function. Finally the mean value of n values of the r.v. ξ will be denoted by $\overline{\xi}_n = \frac{1}{n} \sum_{i=1}^{n} \xi_i$. Normally, after finding the mean value that approximates the exact solution one has to estimate the probability error R_n (see Definition 1.1).

The analysis, studying and finding the number of operations (or the computational cost) we call *performance analysis*. So, the performance analysis deals with the computational cost of algorithms. Here we consider the computational cost and complexity of solving integral equations. In Section 2 we define what we mean by *solving integral equations*. In fact we consider algorithms for computing bilinear forms of the solution of the Fredholm integral equations of second kind. In Section 3 the functional spaces, which define the regularity of the *input data* are defined and error estimates are presented. Some results of ε-complexity of the problem under consideration are presented in Section 4. Here we also define the class of *almost optimal randomized algorithms*. Results of the computational cost of a grid-free almost optimal randomized algorithm are given in section 5.

2 Formulation of the problem of solving integral equations

Let us consider the following problem: Compute the functional

$$(h, u) = \int_G h(x)u(x)dx, \tag{5}$$

where $h(x)$ is a given function and $u(x)$ is the solution of the Fredholm integral equation of second kind:

$$u(x) = \int_G l(x, y)u(y)dy + f(x), \tag{6}$$

or in an operator form $u = Lu + f$, where $L : C(G) \to C(G)$ denotes an integral operator. We should note here that such a formulation of the problem is very often used in

theoretical and applied sciences. The meaning of the above formulated functional is given in the Introduction. It could be the mean value of the velocity of the particles (the first integral moment of the velocity) or the energy (the second integral moment of the velocity) in statistical physics problems, or effect of given pollution levels $u(x)$ (satisfying integral transport equation) on the life matter ($h(x)$ is sensitivity to a given pollutant).

The solution operator for the above formulated problem can be written in the following form:

$$S_{Eq}(l,f) = (h,u) = \left((I-L)^{-1} f, h \right).$$

Sometimes, the adjoint equation

$$v = L^* v + h \tag{7}$$

is used.

In (7) $v, h \in \mathbf{X}^*$, $L^* \in [\mathbf{X}^* \to \mathbf{X}^*], \mathbf{X}^*$ is the dual functional space to $\mathbf{X}$ and L^* is an adjoint operator. For some important applications $\mathbf{X} = \mathbf{L_1}$ and $\| f \|_{\mathbf{L}_1} = \int\limits_G | f(x) | dx$. In this case $h(x) \in \mathbf{L}_\infty$, hence ${\mathbf{L}_1}^* \equiv \mathbf{L}_\infty$ and $\| h \|_{\mathbf{L}_\infty} = \sup |h(x)|, \ \ x \in G$. Obviously, if $u \in \mathbf{L}_1$ and $h \in \mathbf{L}_\infty$ the inner product (5) will be bounded.

For many applications $\mathbf{X} = \mathbf{X}^* = \mathbf{L}_2$. $\mathbf{L_2}$ norms are defined as follows:

$$\| f \|_{\mathbf{L}_2} = \left(\int\limits_G (f(x))^2 dx \right)^{\frac{1}{2}};$$

$$\| L \|_{\mathbf{L}_2} \leq \sup_x \left(\int\limits_G (l(x,x'))^2 dx' \right)^{\frac{1}{2}}.$$

Note also, that if $h(x), u(x) \in \mathbf{L}_2$ then the inner product (5) is finite. One can see, that if $u(x) \in \mathbf{L}_2$ and $l(x,x') \in \mathbf{L}_2(G \times G)$ then $Lu(x) \in \mathbf{L}_2$. It is trivial to show that $L^2 u(x), \ldots, L^i u(x), \ldots$ also belong to $\mathbf{L}_2(G)$.

For simplicity and concreteness we assume that $\mathbf{X} = \mathbf{X}^* = \mathbf{L}_2$. If it is also assumed that $\| L^m \| < 1$, where m is any natural number, then the Neumann series $u = \sum_{i=0}^{\infty} L^i f$ converges.

The condition $\| L^m \| < 1$ is not very strong, since, as it was shown by K. Sabelfeld [10], it is possible to construct a Monte Carlo algorithm for which the Neumann series does not converge. Analytically extending the resolvent by a change of the spectral parameter gives a possibility to obtain a convergent algorithm when Neumann series for the original problem does not converge or to accelerate the convergence when it converges slowly. It is easy to show that $J = (h,u) = (f,v)$. This equality means that the solution of the adjoint problem is equivalent to the solution of the original one. The last fact is important for performance analysis studies, because in practice very often the solution of the adjoint problem is easier than the solution of the original one.

Let us consider the Monte Carlo algorithm for evaluating the functional (5). It can be seen that when $l(x,x') \equiv 0$ evaluation of the integrals can pose a problem. Consider a random point $\xi \in G$ with a density $p(x)$ and let there be n values of the random point $\xi_i (i = 1, 2, ..., n)$. Let a random variable $\theta(\xi)$ be defined in G, such that $E\theta(\xi) = J$.

Then the computational problem becomes one of calculating repeated values of θ and of combining them into an appropriate statistical estimator of J. The nature of the every process realization of θ is a Markov process. We will consider *discrete Markov processes with a finite set of states*, the so called *Markov chains* (see, Definition 1.2 given in the Introduction).

An approximate value of the linear functional J, defined by (5) is $J \approx \frac{1}{n} \sum_{s=1}^{n} \{\theta\}_s = \hat{\theta}_n$, where $(\theta)_s$ is the s-th value of the random variable θ.

The r.v. whose mathematical expectation is equal to $J(u)$ is given by the following expression

$$\theta[h] = \frac{h(\xi_0)}{p(\xi_0)} \sum_{j=0}^{\infty} Q_j f(\xi_j),$$

where $Q_0 = 1$; $Q_j = Q_{j-1} \dfrac{l(\xi_{j-1}, \xi_j)}{p(\xi_{j-1}, \xi_j)}, j = 1, 2, \ldots,$ and $\xi_0, \xi_1, \ldots$ is a Markov chain in G with initial density function $p(x)$ and transition density function $p(x,y)$.

Iterative randomized algorithms are characterized by two types of errors:

- *systematic* error $r_i, \ \ i \geq 1$ (obtained from the truncating of the Markov chain) which depends on the number of iterations i of the used iterative process:

$$|r_i| \leq \frac{\| L \|_{\mathbf{L_2}}^{i+1} \|f\|_{\mathbf{L_2}}}{1 - \| L \|_{\mathbf{L_2}}},$$

  and

- *statistical* error R_n, which depends on the number of samples n of Markov chain:

$$R_n = c_\beta \ \sigma^2(\theta[h]) n^{-1/2}, \qquad 0 < \beta < 1, \ \beta \in I\!R.$$

The constant c_β (and therefore also the complexity estimates of algorithms) depends on the confidence level β. Probable error r_n is often used, which corresponds to a $1/2$ confidence level. Such a level is often acceptable for practical computations.

The problem to achieve a good balance between the systematic and statistical error has a great practical importance. To ensure a statistical error ε, it is necessary to perform i

transitions in the Markov process, where i is chosen from the inequality

$$i > \ln^{-1}\alpha\left[\ln\varepsilon + \ln(1-\alpha) - \ln\|f\|_{\mathbf{L_2}}\right] - 1$$

$$(\text{assuming}\,\|f\|_{\mathbf{L_2}} > \varepsilon(1-\alpha)),$$

where $\alpha = \|L\|_2$ and the initial approximation is chosen to be the right-hand side f. To achieve a probable error ε, it is necessary to perform n samples depending on the following inequality:

$$n > \left[c_{0.5}\frac{\sigma(\theta)}{\varepsilon}\right]^2, \qquad c_{0.5} \approx 0.6745,$$

where θ is the random variable, whose mathematical expectation coincides with the desired linear functional (5).

3 Error analysis results

For the error $r(A_{Eq})$ associated with an algorithm A_{Eq} the following theorem is proved:

Theorem 3.1 *(Emelyanov and Il'in [7]) For the problem S_{Eq} of solving d-dimensional Fredholm integral equations of second kind with p smooth data*

$$r(A_{Eq}) \le c' n^{-\frac{p}{2d}} \tag{8}$$

for the deterministic algorithms $\mathcal{A}$ and

$$r(A^R_{Eq}) \le c'' n^{-\frac{p}{2d}-\frac{1}{2}} \tag{9}$$

for the randomized algorithms $\mathcal{A}^{\mathcal{R}}$.

This result can be slightly improved if one assumes that the "input" data (l, f) satisfy a kind of Hölder conditions. Let us define the class $\hat{H}^p_\lambda(\alpha, E^d)$ for $p, d \in \mathcal{N}$, $\beta, \gamma > 0$, $0 < \delta < 1$:

$$\begin{aligned}
X = \quad & W^p(E^{2d}) \times W^p(E^d) \\
\hat{H}^p_\lambda(\alpha, E^d) = \quad & \Big\{(l,f) \in X : \|\, f \,\|_{W^p(E^d)} \le \gamma, \\
& \|\, l \,\|_{W^p(E^{2d})} \le \beta,\ \|\, l \,\|_{C^0(E^{2d})} \le \delta, \\
& |D^p f(y_1,\dots,y_d) - D^p f(z_1,\dots,z_d)| \\
& \le \alpha \sum_{j=1}^{d} |y_j - z_j|^\lambda, \\
& \left| l^{(p)}_{y_1^{r_1}\dots y_d^{r_d}}(x_1,\dots,x_d;u_1,\dots,u_d) \right. \\
& \left. - l^{(p)}_{y_1^{r_1}\dots y_d^{r_d}}(x_1,\dots,x_d;v_1,\dots,v_d) \right| \\
& \le \alpha(x_1,\dots,x_d) \sum_{j=1}^{d} |u_j - v_j|^\lambda \Big\}.
\end{aligned}$$

We should comment here that the condition $\|\, l \,\|_{C^0(E^{2d})} \le \delta$ ensures the existence and uniqueness of the solution of the integral equation under consideration.

Theorem 3.2 *For the problem S_{Eq} of solving d-dimensional Fredholm integral equations of second kind with p smooth data, which is Hölder with a rate of λ, e.i., $(l, f) \in \hat{H}^p_\lambda(\alpha, E^d)$ we have:*

$$r(A_{Eq}) \le c' n^{-\frac{p+\lambda}{2d}} \tag{10}$$

for the deterministic algorithms $\mathcal{A}$ and

$$r\left(A^R_{Eq}\right) \le c'' n^{-\frac{p+\lambda}{2d}-\frac{1}{2}} \tag{11}$$

for the randomized algorithms $\mathcal{A}^{\mathcal{R}}$.

4 Complexity of the problem of computing functionals of Fredholm integral equations of second kind

Theorem 4.1 *For $X_0 \equiv \hat{H}^p_\lambda(\alpha, E^d)$ the ε-complexity of solving d-dimensional Fredholm integral equation of second kind S_{Eq} is*

$$comp_\varepsilon(S_{Eq}) = k\left(c'(d, p+\lambda)\right)^{\frac{2d}{p+\lambda}} \left(\frac{1}{\varepsilon}\right)^{\frac{2d}{p+\lambda}}$$

for the class of deterministic algorithms $\mathcal{A}$, and

$$comp_\varepsilon(S_{Eq}) = k^R\left(c''(d, p+\lambda)\right)^{\frac{2d}{p+\lambda+d}} \left(\frac{1}{\varepsilon}\right)^{\frac{2d}{p+\lambda+d}}$$

for the class of randomized algorithms $\mathcal{A}^{\mathcal{R}}$.

Corollary 4.1 *If there is not edditional regularity, i.e., $p + \lambda = 0$ the deterministic algorithms are not feasible while the randomized algorithms are feasible with a rate of ε-complexity of order*

$$\left(\frac{1}{\varepsilon}\right)^2.$$

It is reasonable to consider a class of randomized algorithms with a slightly higher ε- complexity.

Definition 4.1 *Randomized algorithms with ε- complexity of order:*

$$\left(\frac{1}{\varepsilon}\right)^2 \log\varepsilon$$

will be called almost optimal randomized algorithms.

In the next section an almost optimal Monte Carlo algorithm with a rate of ε- complexity of order $\left(\frac{1}{\varepsilon}\right)^2 \log\varepsilon$ for evaluation functionals of solution of Fredholm integral equation of second kind in case $p + \lambda = 0$ will be presented.

5 Computational cost of a grid-free randomized algorithm for evaluating functionals of the solution of Fredholm integral equations of second kind

In this section we consider a grid-free Monte Carlo algorithm called (in the simplest case) *spherical process* for computing of the bilinear forms of the solution of Fredholm integral equations of second kind. We will denote this algorithm by A^R_{GF}. As a first step of this algorithm a Δ-strip ∂G_Δ of the boundary ∂G is chosen (on the supposition that the solution is known on the boundary) to ensure the convergence of the constructed iterative process. The following number of operations is necessary for one random walk:

- generation of d (this number depends on initial probability π) random numbers to determine the initial point in the Markov chain: $d(k_A + k_L)$ operations (k_A and k_L are the arithmetic and logical operations necessary for the generation of one random number) **or** modelling of an isotropic vector that needs a number of operations of order $R * d(k_A + k_L)$ (the constant R depends on the efficiency of the modelling method and transition probability);
- calculating the coordinates of the initial **or** next point: p_{next} (depends on the modelling method and the dimension d of the domain $B(x)$);
- calculating one value of functions: $p_f; p_\pi, p_\varphi$ **or** p_k, p_P;
- calculating one sample of the random variable (it needs less than 3 arithmetic operations);
- calculating the distance from the current point to the boundary ∂G: γ_A arithmetic and γ_L logical operations (depends on the dimension d of the domain G);
- verification if the current point belongs to the chosen Δ-strip ∂G_Δ.

The following logarithmic estimate for the average number $E\{i\}$ of spheres on a single trajectory holds for a wide class of boundaries [11]:

$$E\{i\} \le const\, |\ln \Delta|, \qquad const > 0, \tag{12}$$

where $const$ depends on the boundary ∂G.

Calculating the linear functional with a preliminary given accuracy ε and attainment of a good balance between the *statistical* and the *systematic* error is an important issue in performance analysis studies. To ensure a statistical error ε, it is necessary to perform i transitions in the Markov process, where i is chosen from the inequality:

$$i \ge \ln^{-1}\alpha\,(\ln\varepsilon + \ln(1-\alpha) - \ln F^{(0)}) - 1 \tag{13}$$

$$(\text{assuming } F^{(0)} > \varepsilon\,(1-\alpha)),$$

where $\alpha = V_{B(x)}\, L, \quad L = \max\limits_{x,t} |l(x,t)|$ and the initial approximation is chosen to be the right-hand side $f(x)$. On the other hand, the estimate (12) can be used to chose the value of the parameter Δ of the boundary strip. The value of Δ explicitly depends on the number of transitions i of the Markov process according to (13), that is $\Delta \approx \exp(-i/const)$.

Therefore, the following estimate holds for the mathematical expectation of the time required to obtain an approximation with accuracy ε using the considered grid-free Monte Carlo algorithm:

$$\begin{aligned}
&cost(A^R_{GF}, x, \omega)\\
&\approx \tau\, n[(d\,k_A + p_{next} + p_f + p_\pi + p_\varphi + \gamma_A + 4)\, l_A +\\
&(d\,k_L + \gamma_L + 1)\, l_L +\\
&((R\,d\,k_A + p_{next} + p_f + p_k + p_P + 4 + \gamma_A)\, l_A\\
&+(R\,d\,k_L + \gamma_L + 1)\, l_L) \times\\
&\times \frac{(\ln\varepsilon + \ln(1-\alpha) - \ln^3 F^{(0)}}{\ln^3 \alpha}] \ \frac{\{c_\beta\, \sigma(\Theta[h])\}^2}{\varepsilon^2}.
\end{aligned}$$

Remark 5.1 *From the avobe estimate it is easy to see that the computational cost of the grid-free randomized algorithm A^R_{GF} for evaluating bilinear forms of Fredholm integral equations of second kind has a rate of ε- complexity of order $\left(\frac{1}{\varepsilon}\right)^2 \log\varepsilon$. Since not additional regularity is required this algorithm has almost optimal rate.*

Remark 5.2 *If some additional regularity is assumed, then it is possible that some deterministic algorithm or grid Monte Carlo algorithm could be applied (see, for instance, [6]).*

The bounds for the ε- complexity for randomized and deterministic algorithms are given in Theorem 4.1. But it is interesting to compare two randomized algorithms - the above described grid-free Monte Carlo algorithm A^R_{GF} with the grid Monte Carlo algorithm A^R_G. The description of the grid Monte Carlo algorithm is given in [5]. This algorithm is based on the approximation of the integral equation under consideration by a system of linear algebraic equations. This transformation represents the initial step of the considered class of grid randomized algorithms. The linear system is obtained using some approximate cubature rule (cubature method, Nystrom method, [1, 8]). The next step is to apply the resolvent Monte Carlo algorithm [2] for solving linear systems of equations. We should mention here that (according to my knowledge) it is still not proved that the grid

Monte Carlo algorithm presented in [4] is almost optimal in sense of reaching the correspondent rate of complexity, as it was shown for the grid-free algorithm A^R_{GF}. Nevertheless, we can mention some conditions under which the grid Monte Carlo algorithm could be competitive with the grid-free Monte Carlo algorithm:

- the *input data* functions for the integral equation $l(x,t), f(x), h(x)$ should have comparatively small maximum norm in the corresponding domain and it should be a possibility to calculate their values with a relatively low complexity;
- the initial and transition probability used in the grid-free algorithm are complicated for modeling (acceptance-rejection method);
- the dimension d of the integration domain is large.

It has to be noted that the grid Monte Carlo algorithms are admissible only for integral equations with smooth functions, but some techniques of avoiding singularities of this kind exist (see [11]).

6 Concluding remarks

Consider the summary results of the work presented on Table 1 and compared with the results for multidimensional integration.

Table 1. ε-complexity of problems for the classes of deterministic and randomized algorithms for $H^p_\lambda(\alpha, E^d)$ and $H^p_\lambda(\alpha, E^{2d})$ spaces

Problem	ε-complexity
Determ. Integration	$k\left(c'_A(d,p+\lambda)\alpha\right)^{\frac{d}{p+\lambda}}\left(\frac{1}{\varepsilon}\right)^{\frac{d}{p+\lambda}}$
Random. Integration	$k^R\left(c''_A(d,p+\lambda)\alpha\right)^{\frac{d}{p+\lambda+d/2}}\left(\frac{1}{\varepsilon}\right)^{\frac{d}{p+\lambda+d/2}}$
Determ. Int. eq.	$k\left(c'(d,p+\lambda)\right)^{\frac{2d}{p+\lambda}}\left(\frac{1}{\varepsilon}\right)^{\frac{2d}{p+\lambda}}$
Random. Int. eq.	$k^R\left(c''(d,p+\lambda)\right)^{\frac{2d}{p+\lambda+d}}\left(\frac{1}{\varepsilon}\right)^{\frac{2d}{p+\lambda+d}}$

One can conclude that as smaller is the regularity as simpler randomized algorithm should be used. Even for small dimensions $(d = 1, 2)$ Monte Carlo is a right choice if the functional class has no smoothness. In this case you have to accept that the error estimate will be obtained with a given probability. If the computational problem you are treating allows such an interpretation, then Monte Carlo will be the best choice. Here we will note that this is a kind of game: to win in the rate of convergence you have to lose in the reliability. It means that the increased rate of convergence is paid by accepting uncertainty in the answer.

As a general remark it should be emphasized that for both problems: numerical integration and evaluation linear functionals of integral equations the randomized algorithms have better convergence rate for the same regularity of the input data.

But one should be careful because

- the better convergence rate for randomized algorithms is reached with a given probability less than 1, so the advantage of Monte Carlo algorithms is a matter of definition of the probability error.
- If the nature if the problem under consideration do not allow to use the probability error for estimates or the answer should be given with a guaranteed error then the higher convergence order randomized algorithms are not acceptable.
- An important obvious advantage of randomized algorithms is the case of *bad functions*, i.e., functions that do not satisfy some additional conditions of regularity. The main problem with the deterministic algorithms is that normally they need some additional approximation procedure that require additional regularity. The randomized algorithms do not need such procedures.

References

[1] Bachvalov, N.S., Zhidkov, N.P., Kobelkov, G.M., *Numerical Methods*, **Nauka**, Moscow (1987).

[2] Dimov, I.T., Alexandrov, V.N., Karaivanova, A.N., *Parallel resolvent Monte Carlo algorithms for linear algebra problems*, **Mathematics and Computers in Simulation**, **Vol. 55** (2001), 25–35.

[3] Dimov, I.T., Atanassov, E., *Exact Error Estimates and Optimal Randomized Algorithms for Integration*, **Mathematics and Computers in Simulation**, **Vol. 55** (2006), 25–35.

[4] I. Dimov, R. Georgieva, *Complexity of a Class of Monte Carlo Algorithms for Integral Equations*, Proc. of the International Conference on MC&QMC Methods, June 7-10, 2004 Juan-les-Pins, Côte d'Azur, France, 2004.

[5] I. Dimov, A. Karaivanova, H. Kuchen, H. Stoltze, *Monte Carlo Algorithms for Elliptic Differential*

Equations. Data Parallel Functional Approach, **Journal of Parallel Algorithms and Applications, Vol. 9** (1996), pp. 39–65.

[6] I. Dimov, O. Tonev, *Monte Carlo algorithms: performance analysis for some computer architectures*, **J. of Computational and Applied Mathematics, Vol. 48** (1993), pp. 253–277.

[7] K.V. Emelyanov and A.M. Il'in, *On the number of arithmetic operations nessesary for the approximate solution of Fredholm integral equations of second kind*, **J. Vychisl. Math. i Math. Phys., Vol. 7** (1967), pp. 905–910 (in Russian).

[8] Kantorovich, L.V., Krylov, V.I., *Approximate Methods of Higher Analysis*, **Physical and Mathematical State Publishing House**, Leningrad, 1962.

[9] E. Novak, *Deterministic and stochastic error bound in Numerical Analysis*, **Lecture Notes in Mathematics, 1349**, Springer, Berlin, Heidelberg, New York, London, Paris and Tokio, 1988.

[10] K. Sabelfeld, *Algorithms of the Method Monte Carlo for Solving Boundary Value Problems*, **Nauka**, Moscow, 1989.

[11] I.M. Sobol, *Monte Carlo numerical methods*, **Nauka**, Moscow, 1973.

[12] J.F. Traub, G.W. Wasilkowski and H. Woźniakowski, *Information-based compexity*, **Academic Press**, New York, 1988.