# Robust Monte Carlo Algorithms

Ivan Dimov

Centre for Advanced Computing and Emerging Technologies
The University of Reading

Reading, 16 November 2005

# Outline

- General overview:
  - Definition
  - Historical Remarks
  - Areas of Application in Science and Technologies
- Robust Monte Carlo Algorithms
  - Basic Linear Algebra Problems
  - Evaluation of Matrix Polynomials
  - Robust Monte Carlo Algorithms
  - Interpolation Algorithms
- Conclusion

# Definition

- Monte Carlo method is a numerical method using random variables for solving numerical problems.

**Definition 1. [Yu.A.Shreider, 66]** *The Monte Carlo method consists of solving various problems of computational mathematics by means of the construction of some random process for each such problem, with the parameters of the process equal to the required quantities of the problem.*

There are two classes of Monte Carlo algorithms:

- Monte Carlo $Simulations$: the problem is described by a $random$ process and the mathematical model is probabilistic.

- Monte Carlo $Numerical$ Algorithms: the problem is $deterministic$ (an artificial random process is introduced).

## Historical Overview

- **1777**: G. Compte de Buffon: computing the probability $P = 2L/\pi d$;
- **1886**: Marquis Pierre-Simon de Laplace: method of computing $\pi$;
- **1949**: "official birthday" of Monte Carlo - **Manhatton project** (John von Neumann, E. Fermi, G. Kahn, N. Metropolis and S. Ulam), Los Alamos (USA);
- The end of **20th century**: the development of **modern computers**, and particularly parallel computing systems, provided fast and **specialized generators of random numbers**.

## Areas of application of Monte Carlo algorithms

The randomized algorithms are currently widely used for those problems for which the deterministic algorithms hopelessly break down:

- high-dimensional integration,
- LA problems for large-scale systems,
- boundary-value problems for differential equations in domains with complicated boundaries,
- simulation of turbulent flows,
- studying of chaotic structures.

## Advantages of Monte Carlo

- Direct determination of an unknown functional of the solution,
- efficient parallel implementation,
- efficient vectorization.

In statistical physics and environmental sciences: computing linear functionals of the solution of the equations for

- density distribution function (such as Schroedinger equation), i.e., probability of finding a particle at a given $(x, t)$ (integral of the solution),
- mean value of the velocity of the particles (the first integral moment of the velocity) or
- energy (the second integral moment of the velocity) and, so on.

# Error estimation

**Definition 2.** *If $J$ is the exact solution of the problem, then the probability error is the least possible real number $R_N$, for which:*

$$P = Pr\left\{|\overline{\xi}_N - J| \leq R_N\right\}, \tag{1}$$

*where $0 < P < 1$. If $P = 1/2$, then the probability error is call $probable\ error$.*

# A weak point of Monte Carlo

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

- The probabilistic setting of the problem of error estimation may not be acceptable if one needs a guaranteed accuracy or strictly reliable results. But in the most cases it is reasonable to accept an error estimate with a probability smaller than $1$.

In fact, this is the price paid by randomized algorithms to increase their convergence rate. It is important to note here that the value of the probability $P \quad (0 < P < 1)$ in $(1)$ does not reflect on the rate of convergence of the probability error $R_n$. It reflects only on the constant. That's why the choice of the value of $P$ is not important for the convergence rate (respectively, for the rate of algorithmic complexity).

# Motivation

Criteria for choice of a $better$ algorithm:

- the algorithm, which produces the $\varepsilon$-approximation of the solution faster or with a smaller number of steps

To increase the $efficiency$ of the algorithms one should:

- decrease the $probability\ error$, or
- find a better $parallelization\ strategy$ if $clusters\ or\ grids$ are used.

## Iterative Monte Carlo

Define an iteration of *order* $i$:

$$u^{(k+1)} = F_k(A, b, u^{(k)}, u^{(k-1)}, \ldots, u^{(k-i+1)}),$$

where $u^{(k)}$ is obtained from the $k$-th iteration. It is desired that

$$u^{(k)} \to u = A^{-1}b \ \text{ as } \ k \to \infty.$$

The algorithm is called *stationary* if $F_k = F$ for all $k$, that is, $F_k$ is independent of $k$. The iterative process is called *linear* if $F_k$ is a linear function of $u^{(k)}, \ldots, u^{(k-i+1)}$. Here we will be interested in *stationary linear iterative Monte Carlo* algorithms.

Consider the sequence $u_1, u_2, ...,$:

$$u_k = L(u_{k-1}) + f, \ \ k = 1, 2, \ldots.$$

$$u_k = f + L(f) + \cdots + L^{k-1}(f) + L^k(u_0), \qquad k > 0,$$

When the infinite series converges, the sum is an element $u$ which satisfies the equation

$$u = L(u) + f.$$

The truncation error is $u_k - u = L^k(u_0 - u)$. Let $J(u_k)$ be a functional that is to be calculated. Consider the spaces $\mathbf{T}_{i+1} = \underbrace{\mathbb{R}^d \times \mathbb{R}^d \times \cdots \times \mathbb{R}^d}_{i \text{ times}}, \qquad i = 1, 2, \ldots, k, .$

$\theta_i,\ i = 0, 1, \ldots, k$ are defined on the respective product spaces $\mathbf{T}_{i+1}$ and have conditional mathematical expectations:

$$E\theta_0 = J(u_0),\ E(\theta_1/\theta_0) = J(u_1), \ldots, E(\theta_k/\theta_0) = J(u_k),$$

where $J(u)$ is a functional of $u$.

## Performance analysis

**Definition 3.** *Computational cost of a randomized iterative algorithm $A^R$ is defined by*

$$cost(A^R, x, \omega) = nE(k)t_0,$$

*where $E(k)$ is the mathematical expectation of the number of transitions in the sequence and $t_0$ is the mean time needed to compute the value of one transition.*

# Linear Algebra Problems - the Power method

$A \in C^{n \times n}$; $X^{-1}AX = diag(\lambda_1, \ldots, \lambda_n)$, $X = (x_1, \ldots, x_n)^T$, and $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$. Given $f^{(0)} \in C^n$, the *power method* produces a sequence of vectors $f^{(k)}$:

$$z^{(k)} = Af^{(k-1)},$$

$$f^{(k)} = z^{(k)}/||z^{(k)}||_2,$$

$$\lambda^{(k)} = [f^{(k)}]^H A f^{(k)}, \quad k = 1, 2, \ldots.$$

The computational complexity:

$$T_1(PM) \approx \tau l_A k(4n^2 + 3n - 2) = O(kn^2),$$
$$T_1(PM - MC) \approx (c_1 + c_2 d)kN = O(kdN).$$

# Monte Carlo evaluating matrix polynomials

$$p(A) = \sum_{i=0}^{s} b_i A^i$$

MC evaluates the real values:

$$(h, p(A)f).$$

Consider MC method for evaluating the real values:

$$\lambda = \frac{(h, Ap(A)f)}{(h, p(A)f)}.$$

# Monte Carlo algorithms

## The Power Monte Carlo algorithm: $p(A) = A^i$

Iteration process:

$$\lambda_{max} = lim_{i \to \infty} \frac{(h, A^i f)}{(h, A^{i-1} f)},$$

$$(h, A^i f) = E\{W_i f_{k_i}\}, \;\; i = 1, 2, \ldots. \tag{2}$$

$$\lambda_{max} \approx \frac{E\{W_i f_{k_i}\}}{E\{W_{i-1} f_{k_{i-1}}\}}.$$

# Inverse shifted MC algorithm (Resolvent MC)

$$p(A) = \sum_{i=0}^{\infty} q^i C_{m+i-1}^i A^i$$

If $|qA| < 1$, then

$$p(A) = \sum_{i=0}^{\infty} q^i C_{m+i-1}^i A^i = [I - qA]^{-m} = R_q^m$$

$R$ is the resolvent matrix.

$$\lambda = \frac{(h, Ap(A)f)}{(h, p(A)f)} = \frac{(h, AR_q^m f)}{(h, R_q^m f)}$$

# Formulation of the MC Algorithm

Suppose we have a Markov chain

$$T = \alpha_0 \to \alpha_1 \to \alpha_2 \to \ldots \to \alpha_k \to \ldots$$

with $n$ states. The random trajectory (chain) $T_k$ of length $k$ is defined as follows:

$$T_k = \alpha_0 \to \alpha_1 \to \ldots \to \alpha_j \to \ldots \to \alpha_k. \tag{3}$$

Assume that

$$P(\alpha_0 = \alpha) = p_\alpha, \quad P(\alpha_j = \beta | \alpha_{j-1} = \alpha) = p_{\alpha\beta}. \tag{4}$$

Probabilities $p_{\alpha\beta}$ define a transition matrix $P$. We require that

$$\sum_{\alpha=1}^{n} p_\alpha = 1 \quad \text{and} \quad \sum_{\beta=1}^{n} p_{\alpha\beta} = 1, \quad \text{for any} \quad \alpha = 1, 2, ..., n. \tag{5}$$

# MAO density distributions

**Definition 4.** *The distribution* $(p_{\alpha_1}, ..., p_{\alpha_n})$ *is tolerant to vector* $v$*, if*

$$\begin{cases} p_{\alpha_s} > 0 & \textit{when } v_{\alpha_s} \neq 0 \\ p_{\alpha_s} \geq 0 & \textit{when } v_{\alpha_s} = 0. \end{cases} \tag{6}$$

*Similarly, the distribution* $p_{\alpha_{s-1},\alpha_s}$ *is tolerant to matrix* $A$*, if*

$$\begin{cases} p_{\alpha_{s-1},\alpha_s} > 0 \text{ when } a_{\alpha_{s-1},\alpha_s} \neq 0 \\ p_{\alpha_{s-1},\alpha_s} \geq 0 \text{ when } a_{\alpha_{s-1},\alpha_s} = 0. \end{cases} \tag{7}$$

We will consider a special choice of tolerant density distributions $p_i$ and $p_{i,j}$ defined as follows:

$$p_i = \frac{|v_i|}{\| v \|}, \quad p_{ij} = \frac{|a_{ij}|}{\| a_i \|}. \tag{8}$$

# MC algorithm for computing bilinear forms of matrix powers $(v, A^k h)$

The pair of density distributions (8) defines a finite chain of vector and matrix entrances:

$$v_{\alpha_0} \to a_{\alpha_0 \alpha_1} \to \ldots \to a_{\alpha_{k-1} \alpha_k}. \tag{9}$$

The latter chain induces the following product of matrix/vector entrances and norms:

$$A_v^k = v_{\alpha_0} \prod_{s=1}^{k} a_{\alpha_{s-1} \alpha_s}$$

$$\| A_v^k \| = \| v \| \times \prod_{s=1}^{k} \| a_{\alpha_{s-1}} \| .$$

The rule for creating the value of $\| A_v^k \|$ is following: the norm of the initial vector $v$, as well as norms of all row-vectors of matrix $A$ visited by the chain (9) defined by densities (8), are included. For such a choice of densities $p_i$ and $p_{ij}$ we can prove the following Lemma.

**Lemma 1.**

$$E\{h_{\alpha_k}\} = \frac{sign\{A_v^k\}}{\parallel A_v^k \parallel} \left(v, A^k h\right).$$

Obviously, the standard deviation $\sigma\{h_{\alpha_k}\}$ is finite. Since we proved that the random variable $\theta^{(k)} = sign\{A_v^k\} \parallel A_v^k \parallel h_{\alpha_k}$ is a unbiased estimate of the form $(v, A^k h)$, Lemma 1 can be used to construct a MC algorithm.

Let us consider $N$ realizations of the Markov chain $T_k$ (3) defined by the pair of density distributions (8). Denote by $\theta_i^{(k)}$ the $i^{th}$ realization of the random variable $\theta^{(k)}$. Then the value

$$\bar{\theta}^{(k)} = \sum_{i=1}^{N} \theta_i^{(k)} = sign\{A_v^k\} \parallel A_v^k \parallel \sum_{i=1}^{N} \{h_{\alpha_k}\}_i \tag{10}$$

can be considered as a MC approximation of the form $(v, A^k h)$.

In fact, (10) together with the rules (8) defines a MC algorithm. The expression (10) gives a MC approximation of the form $(v, A^k h)$ with a probability error $R_N^{(k)}$. Obviously, the quality of the MC algorithm depends on the behavior of the standard deviation $\sigma\{\theta^{(k)}\}$. So, there is a reason to consider a special class of $robust\ MC\ algorithms$.

# Robust MC algorithms

**Definition 5.** *MC algorithm for which the standard deviation does not increase with increasing of matrix power $k$ is called robust MC algorithm.*

**Lemma 2.** *If MC algorithm is robust, then there exist a constant $M$ such that*

$$\lim_{k \to \infty} \sigma\{\theta^{(k)}\} \leq M.$$

It is interesting to answer the question:

- *How small could be the probability error? and*
- *Is it possible to construct MC algorithms with zero probability error?*

To answer the first question one has to analyze the structure of the variance. Then it will be possible to answer the second question concerning the existence of algorithms with zero probability error.

# Interpolation MC algorithms

**Definition 6.** *MC algorithm for which the probability error is zero is called interpolation MC algorithm.*

The next theorem gives the structure of the variance for MAO algorithm. Let us introduce the following notations:

$$\hat{h} = \{h_i^2\}_{i=1}^n,$$
$$\bar{v} = \{|v_i|\}_{i=1}^n,$$
$$\bar{A} = \{|a_{ij}|\}_{i,j=1}^n.$$

**Theorem 1.**

$$D\{\theta^{(k)}\} = \| A_v^k \| \left(\bar{v}, \bar{A}^k \hat{h}\right) - (v, A^k h)^2.$$

We can formulate an important corollary that gives a sufficient condition for constructing an interpolation MC algorithm.

**Corollary 1.** *Let* $h = (1, \ldots, 1)$, $v = (\frac{1}{n}, \ldots, \frac{1}{n})$ *and* $A = \begin{pmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & & \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{pmatrix}$.

*Then MC algorithm defined by density distributions (8) is an interpolation MC algorithm.*

# Error balancing technique

## There are two types of errors

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

$$r_k \leq \frac{\| A \|_2^{k+1} \|f\|_2}{1 - \| A \|_2},$$

and

- *statistical* error $R_N$, which depends on the number of samples $N$ of Markov chain:

$$R_N = c_\beta \; \sigma^2(\theta[h]) N^{-1/2}, \qquad 0 < \beta < 1, \; \beta \in \mathbb{R}.$$

# Computational Cost and Speed-up

$$cost_1(A^{(RMC)}, x, \omega) \approx 2\tau \left[ (k + \gamma_A) l_A + \frac{1}{2} d l_L \right] kN + 2\tau n(1 + d)$$
$$\approx (c_1 + c_2 d) kN. \tag{11}$$

To get an optimal (by rate) algorithm we should have $k = O(1)$.

$$S_p(RMC) \approx \frac{(c_1 + c_2 d) kN}{(c_1 + c_2 d) k \frac{N}{p} (1 + \varepsilon)}.$$

$$S_p(RMC) \approx \frac{p}{1 + \varepsilon} \geq 1.$$

$$\frac{1}{1 + \varepsilon} \leq E_p(RMC) \leq 1.$$

# Applicability and acceleration analysis

The *systematic error* is

$$O\left[\left(\frac{2\lambda_1 + \lambda_n}{2\lambda_1 + \lambda_{n-1}}\right)^k\right], \tag{12}$$

where $k$ is the power (or the number of iterations).

- The best convergence in case when all eigenvalues are positive or all are negative is $O[(2/3)^k]$.
- When $\lambda_n \approx -2\lambda_1$ the convergence is very good and the complexity is *close* to the **optimal complexity**.

# Conclusion

- A common approach for solving a *class of LA problems* is formulated.
- A necessary condition for constructing a *robust Monte Carlo algorithm* is obtained.
- It is shown that *interpolation Monte Carlo algorithms* (i.e., algorithms with zero probability error) exist.
- *Balancing of errors* is an important issue for constructing efficient algorithms.

# Questions?

http://www.personal.rdg.ac.uk/~sis04itd/