A Monte Carlo Method Based on $\Lambda \Pi_T$ Sequences

Ivan Dimov
(in collaboration with Rayna Georgieva)

Bulgarian Academy of Sciences (BAS)
e-mail: ivdimov@bas.bg

National Science Fund Grant DTK 02/44
"Efficient Monte Carlo Methods for Large-Scale Scientific Problems"

Eighth IMACS Seminar on
“Monte Carlo Methods”
Introduction
- deterministic algorithms
- randomized (Monte Carlo) algorithms

Problem setting

Mathematical background
- Uniformly distributed sequences
- \((t, m, s)\)-nets and \((t, s)\)-sequences in base \(b\)
- \(\Lambda\Pi_\tau\) sequences
- Randomized Quasi-Monte Carlo (RQMC)

Description of the Monte Carlo algorithm

Numerical experiments

Discussion and applicability

Concluding remarks
Consider the following problem of integration:

\[ S(f) := I = \int_{E^s} f(x) \, dx, \quad \text{where} \]

\[ E^s \equiv [0, 1]^s, \quad x \equiv (x_1, \ldots, x_s) \in E^s \subset \mathbb{R}^s, \quad f \in C(E^s). \]

Assume for a given r.v. \( \theta \) one can prove that \( E \theta = I \).

Monte Carlo approximation to the solution: \( \bar{\theta}_n = \frac{1}{n} \sum_{i=1}^{n} \theta(i) \approx I. \)

**Definition.**

If \( I \) is the exact solution of the problem, then the probability error is the least possible real number \( R_n \), for which \( P = Pr \{ |\bar{\theta}_n - I| \leq R_n \} \), where \( 0 < P < 1 \).
The computational problem is to approximate $S(f)$:

$$S(f) : f \rightarrow \mathbb{R}, \quad \text{where}$$

$$S(f) = \int_{E^s} f(x) dx \quad \text{and} \quad f \in F_0 \subset C(E^s).$$

- **Quadrature formula** $A = \sum_{i=1}^{n} c_i f(x^{(i)})$.

- **Randomized quadrature formula** $A^R = \sum_{i=1}^{n} \sigma_i f(\xi^{(i)})$.

**Definition.**

Consider the set $\mathcal{A}$ of algorithms $\mathcal{A}$: $\mathcal{A} = \{A : \Pr(R_n \leq \varepsilon) \geq c\}$ that solve a given problem with a probability error $R_n$ such that the probability that $R_n$ is less than *a priori* given constant $\varepsilon$ is bigger than a constant $c < 1$. 
Definition (H. Weyl, 1916).

The sequence \( x_1, x_2, \ldots \) is called an **uniformly distributed sequence** (u.d.s.) if, for an arbitrary region \( \Omega \subset E^s \),

\[
\lim_{n \to \infty} \left[ \frac{S_n(\Omega)}{n} \right] = V(\Omega),
\]

where \( S_n(\Omega) \) is the number of points with \( 1 \leq i \leq n \) that lie inside \( \Omega \) and \( V(\Omega) \) is the \( s \)-dimensional volume of \( \Omega \).


The relation

\[
\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} f(\xi_j) = \int_{E^s} f(x) dx
\]

holds for all Riemann integrable functions \( f \) if and only if the sequence \( x_1, x_2, \ldots \) is u.d.s.
Definition (H. Weyl, 1916).

The sequence \( x_1, x_2, \ldots \) is called an uniformly distributed sequence (u.d.s.) if, for an arbitrary region \( \Omega \subset E^s \),

\[
\lim_{n \to \infty} \frac{S_n(\Omega)}{n} = V(\Omega),
\]

where \( S_n(\Omega) \) is the number of points with \( 1 \leq i \leq n \) that lie inside \( \Omega \) and \( V(\Omega) \) is the \( s \)-dimensional volume of \( \Omega \).

A u.d.s. should satisfy three additional requirements:

(i) the best asymptote as \( n \to \infty \);

(ii) well distributed points for small \( n \);

(iii) a computationally inexpensive algorithm.
An elementary $s$-interval in base $b$ is a subset of $E^s$ of the form

$$\prod_{j=0}^{s} \left[ \frac{a_j}{b^{d_j}}, \frac{a_j + 1}{b^{d_j}} \right],$$

where $a_j$, $d_j$ are integers and $a_j < d_j$ for all $j \in \{1, \ldots, s\}$. 

Uniformly distributed sequences
$(t, m, s)$-nets and $(t, s)$-sequences in base $b$
$\Lambda\Pi_{\tau}$ sequences
Randomized Quasi-Monte Carlo (RQMC)
Given two integers $0 \leq t \leq m$, a $(t, m, s)$-net in base $b$ is a sequence $x_n$ of $b^m$ points of $E^s$ such that $\text{Card } P \cap \{x_1, \ldots, x_{b^m}\} = b^t$ for any elementary interval $P$ in base $b$ of hypervolume $\lambda(P) = b^{t-m}$. 

**Definition (H. Niederreiter, 1988).**
Definition (H. Niederreiter, 1988).

Given two integers $0 \leq t \leq m$, a $(t, m, s)$-net in base $b$ is a sequence $x_n$ of $b^m$ points of $E^s$ such that $\text{Card } P \cap \{x_1, \ldots, x_{b^m}\} = b^t$ for any elementary interval $P$ in base $b$ of hypervolume $\lambda(P) = b^{t-m}$.

Definition (H. Niederreiter, 1988).

Given a non-negative integer $t$, a $(t, s)$-sequence in base $b$ is an infinite sequence of points $x_n$ such that for all integers $k \geq 0$, $m \geq t$, the sequence $\{x^{kb^m}, \ldots, x^{(k+1)b^m-1}\}$ is a $(t, m, s)$-net in base $b$. 
Choose a primitive polynomial of degree $s_j$ over the Galois field

$$P_j = x^{s_j} + a_{1,j}x^{s_j-1} + a_{2,j}x^{s_j-2} + \ldots + a_{s_j-1,j}x + 1,$$

where the coefficients $a_{1,j}, \ldots, a_{s_j-1,j} \in \{0, 1\}$.

A sequence of positive integers $\{m_{1,j}, m_{2,j}, \ldots\}$ is defined by

$$m_{k,j} = 2a_{1,j}m_{k-1,j} \oplus 2^2a_{2,j}m_{k-2,j} \oplus \ldots \oplus 2^{s_j}m_{k-s_j,j} \oplus m_{k-s_j,j},$$

where the initial values $m_{k,j}, 1 \leq k \leq s_j$ are odd and less than $2^k$.

The direction numbers $\{v_{1,j}, v_{2,j}, \ldots\}$: $v_{k,j} = \frac{m_{k,j}}{2^k}$.

The $j$-th component of the $i$-th point in a Sobol’ sequence

$$x_{i,j} = i_1 v_{1,j} \oplus i_2 v_{2,j} \oplus \ldots,$$

where $i_k$ is the $k$-th binary digit of $i = (\ldots i_3 i_2 i_1)_2$. 

† I. Sobol’ (1979), P. Bradley, B. Fox (1988)
Randomized Quasi-Monte Carlo turns QMC into a variance reduction method by carefully randomizing well distributed points $x_i \equiv (x_{i,1}, x_{i,2} \ldots x_{i,s})$.

Examples of RQMC point sets include
- randomly shifted lattice rules,
- scrambled digital nets,
- digital nets with a random digital shift,
- a Latin hypercube sample or a stratified sample followed by a random permutation of the points.

Array-RQMC.

If

\[ x_i = (x_{i,1}, x_{i,2}, \ldots x_{i,s}) \in \mathbb{E}^s \]

- the \( i \)-th \( \Lambda \Pi_\tau \) point,

then

the \( i \)-th random point \( \xi_i(\rho) \)

with a p.d.f. \( p(x) \):

\[ \xi_i(\rho) = x_i + \rho \omega_i, \]

where \( \omega_i \) is a unique uniformly distributed vector in \( \mathbb{E}^s \) and \( \rho \) is the "shaking radius".

Example:

\[ \omega_i = \{\cos \phi_i, \sin \phi_i\} \in \mathbb{E}^2 \]

Figure: Generation of a random point \( \xi_i \in \mathbb{E}^2 \).
Theorem.

The mathematical expectation of the random variable $\theta = f(\xi)$ is equal to the value of the integral, that is

$$E\{\theta\} = S(f) = \int_{E^s} f(x)dx.$$ 

Sketch of proof.

Assume $\xi(\rho) = x + \rho \omega \in E^s$, where $\rho$ is relatively small $\rho << \left[ \frac{a_j}{2^d}, \frac{a_j+1}{2^d} \right]$. $\xi_i(\rho)$ is still in the same elementary $s$-interval $E^s_i = \prod_{j=0}^{s} \left[ \frac{a_j}{2^d}, \frac{a_j+1}{2^d} \right]$, where the pattern $\Lambda \Pi_\tau$ point $x_i$ is. Since $\int_{E^s} p(x)dx = 1$; $p(x) = 1$ for $x \in E^s$. That’s why for the mathematical expectation of $\theta = f(\xi)$ we have

$$E\{\theta\} = \int_{E^s} f(x)p(x)dx = \int_{E^s} f(x)dx.$$
Example of a non-smooth integrand:

\[ f_1(x_1, x_2, x_3, x_4) = \sum_{i=1}^{4} |(x_i - 0.5)^{-1/3}|, \]

\[ S(f_1) \approx 7.55953. \]

Figure: The integrand function in two-dimensional case.
**Table**: Radius $\rho$ of spheres of the random points (*radius coefficient* $\kappa = \rho/\delta$).

<table>
<thead>
<tr>
<th>$n$</th>
<th>Min. dist., $\delta$</th>
<th>$\kappa$</th>
<th>$\rho$</th>
<th>$\kappa$</th>
<th>$\rho$</th>
<th>$\kappa$</th>
<th>$\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.43301</td>
<td>0.001</td>
<td>0.00043</td>
<td>0.09</td>
<td>0.03897</td>
<td>0.4</td>
<td>0.17321</td>
</tr>
<tr>
<td>$10^2$</td>
<td>0.13166</td>
<td>0.001</td>
<td>0.00013</td>
<td>0.09</td>
<td>0.01185</td>
<td>0.4</td>
<td>0.05266</td>
</tr>
<tr>
<td>$10^3$</td>
<td>0.06392</td>
<td>0.001</td>
<td>0.00006</td>
<td>0.09</td>
<td>0.00575</td>
<td>0.4</td>
<td>0.02557</td>
</tr>
<tr>
<td>$10^4$</td>
<td>0.02812</td>
<td>0.001</td>
<td>0.00003</td>
<td>0.09</td>
<td>0.00253</td>
<td>0.4</td>
<td>0.01125</td>
</tr>
<tr>
<td>$50.10^3$</td>
<td>0.01400</td>
<td>0.001</td>
<td>0.00001</td>
<td>0.09</td>
<td>0.00126</td>
<td>0.4</td>
<td>0.00560</td>
</tr>
</tbody>
</table>
### Relative error and computational time for numerical integration.

<table>
<thead>
<tr>
<th>$n$</th>
<th>SFMT</th>
<th>Sobol'</th>
<th>MCA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rel. err.</td>
<td>Time (s)</td>
<td>Rel. err.</td>
</tr>
<tr>
<td>10</td>
<td>0.0001</td>
<td>$&lt; 0.01$</td>
<td>0.2813</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^2$</td>
<td>0.0114</td>
<td>0.01</td>
<td>0.0565</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^3$</td>
<td>0.0023</td>
<td>0.06</td>
<td>0.0114</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^4$</td>
<td>0.0006</td>
<td>0.53</td>
<td>0.0023</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$30 \cdot 10^3$</td>
<td>0.0002</td>
<td>1.63</td>
<td>0.0011</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$50 \cdot 10^3$</td>
<td>0.0009</td>
<td>2.67</td>
<td>0.0008</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Calculations have been carried out on a PC with Intel(R) Pentium(R) 4 Processor.
Figure: Relative error according to the "shaking radius".
**Table:** Difference of relative errors for Sobol’ algorithm and the proposed Monte Carlo algorithm.

<table>
<thead>
<tr>
<th>( \kappa )</th>
<th>0.009</th>
<th>0.03</th>
<th>0.2</th>
<th>0.45</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.07709</td>
<td>0.23746</td>
<td>0.20639</td>
<td>0.23037</td>
</tr>
<tr>
<td>( 10^2 )</td>
<td>0.03594</td>
<td>0.05277</td>
<td>0.05214</td>
<td>0.05155</td>
</tr>
<tr>
<td>( 10^3 )</td>
<td>0.01014</td>
<td>0.00976</td>
<td>0.00940</td>
<td>0.01099</td>
</tr>
<tr>
<td>( 10^4 )</td>
<td>0.00197</td>
<td>0.00225</td>
<td>0.00228</td>
<td>0.00212</td>
</tr>
<tr>
<td>( 30.10^3 )</td>
<td>0.00102</td>
<td>0.00094</td>
<td>0.00084</td>
<td>0.00079</td>
</tr>
<tr>
<td>( 50.10^3 )</td>
<td>0.00077</td>
<td>0.00062</td>
<td>0.00077</td>
<td>0.00078</td>
</tr>
</tbody>
</table>
Example of a smooth integrand:

\[ f_2(x_1, x_2, x_3, x_4) = e^{x_1 + 2x_2} \frac{\cos(x_3)}{1 + x_2 + x_3 + x_4}, \]

\[ S(f_2) \approx 1.83690. \]

Figure: The integrand function in two-dimensional case.
**Table:** Relative error and computational time for numerical integration.

<table>
<thead>
<tr>
<th>$n$</th>
<th>SFMT Rel. err.</th>
<th>SFMT Time (s)</th>
<th>Sobol’ Rel. err.</th>
<th>Sobol’ Time (s)</th>
<th>MCA $\delta$</th>
<th>MCA $\kappa$</th>
<th>MCA $\rho \times 10^3$</th>
<th>MCA Rel. err.</th>
<th>MCA Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^2$</td>
<td>0.0350</td>
<td>$&lt; 0.01$</td>
<td>0.0155</td>
<td>$&lt; 0.01$</td>
<td>0.132</td>
<td>0.03</td>
<td>3.9</td>
<td>0.0160</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.45</td>
<td>59</td>
<td>0.0264</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>$10^3$</td>
<td>0.0045</td>
<td>0.01</td>
<td>0.0023</td>
<td>$&lt; 0.01$</td>
<td>0.064</td>
<td>0.03</td>
<td>1.9</td>
<td>0.0025</td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.45</td>
<td>29</td>
<td>0.0058</td>
<td>0.06</td>
<td></td>
</tr>
<tr>
<td>$10^4$</td>
<td>0.0016</td>
<td>0.10</td>
<td>0.0002</td>
<td>0.02</td>
<td>0.028</td>
<td>0.03</td>
<td>0.8</td>
<td>0.0003</td>
<td>3.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.45</td>
<td>12.7</td>
<td>0.0016</td>
<td>3.28</td>
<td></td>
</tr>
<tr>
<td>$30 \times 10^3$</td>
<td>0.0006</td>
<td>0.28</td>
<td>0.0001</td>
<td>0.04</td>
<td>0.019</td>
<td>0.03</td>
<td>0.6</td>
<td>0.0002</td>
<td>28.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.45</td>
<td>8.3</td>
<td>0.0011</td>
<td>28.4</td>
<td></td>
</tr>
<tr>
<td>$50 \times 10^3$</td>
<td>0.0004</td>
<td>0.46</td>
<td>$6\times 10^{-5}$</td>
<td>0.07</td>
<td>0.014</td>
<td>0.03</td>
<td>0.4</td>
<td>0.0001</td>
<td>76.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.45</td>
<td>6.3</td>
<td>0.0008</td>
<td>76.1</td>
<td></td>
</tr>
</tbody>
</table>

Calculations have been carried out on a PC with Intel(R) Pentium(R) 4 Processor.
The proposed algorithm improves the error estimates for non-smooth integrands when the radius $\rho$ is smaller than the minimal distance between $\Lambda\Pi_T$ points $\delta$. Strongly speaking the proposed approach is applicable if $\rho$ is much smaller than $\delta$. The implementation of the algorithm shows that this requirement is not very strong. Even for relatively large radiiuses $\rho$ the results are good. The reason is that centers of spheres are very well uniformly distributed by definition. So that, even for large values of radiiuses of _shaking_ the generated random points continue to be well distributed.

For relatively low number of points ($< 1000$) the proposed algorithm gives results with a high accuracy. The relative error is approximately equal to 0.0038 for $n = 100$. For the same sample size the Sobol’ algorithm gives more than 10 times higher error. For $n = 1000$ our algorithm gives relative error $0.0004 - 0.0016$ depending on the parameter $\kappa$ while the Sobol’ algorithm gives 0.0114. This is an important fact because one has a possibility to estimate the value of the integral with a relatively high accuracy using a small number of random points.
The proposed algorithm improves the error estimates for non-smooth integrands when the radius $\rho$ is smaller than the minimal distance between $\Lambda\Pi_{\tau}$ points $\delta$. Strongly speaking the proposed approach is applicable if $\rho$ is much smaller than $\delta$. The implementation of the algorithm shows that this requirement is not very strong. Even for relatively large radiiuses $\rho$ the results are good. The reason is that centers of spheres are very well uniformly distributed by definition. So that, even for large values of radiiuses of shaking the generated random points continue to be well distributed.

For relatively low number of points ($< 1000$) the proposed algorithm gives results with a high accuracy. The relative error is approximately equal to 0.0038 for $n = 100$. For the same sample size the Sobol’ algorithm gives more than 10 times higher error. For $n = 1000$ our algorithm gives relative error $0.0004 - 0.0016$ depending on the parameter $\kappa$ while the Sobol’ algorithm gives 0.0114. This is an important fact because one has a possibility to estimate the value of the integral with a relatively high accuracy using a small number of random points.
The proposed algorithm combines properties of *two of the best available approaches* - Sobol’s quasi-Monte Carlo integration and a high quality pseudorandom number SIMD-oriented Fast Mersenne Twister (SFMT) generator.

The Monte Carlo algorithm has advantages against quasi-Monte Carlo and SFMT for non-smooth integrands. *For relatively small number of points the proposed approach gives much better results than Sobol’s quasi-Monte Carlo integration.*

*In case of smooth functions the proposed algorithm has significant advantage against plain Monte Carlo that uses SFMT generator with respect to the relative error.*
The proposed algorithm combines properties of two of the best available approaches - Sobol’s quasi-Monte Carlo integration and a high quality pseudorandom number SIMD-oriented Fast Mersenne Twister (SFMT) generator.

The Monte Carlo algorithm has advantages against quasi-Monte Carlo and SFMT for non-smooth integrands. For relatively small number of points the proposed approach gives much better results than Sobol’s quasi-Monte Carlo integration.

In case of smooth functions the proposed algorithm has significant advantage against plain Monte Carlo that uses SFMT generator with respect to the relative error.
The proposed algorithm combines properties of *two of the best available approaches* - Sobol’s quasi-Monte Carlo integration and a high quality pseudorandom number SIMD-oriented Fast Mersenne Twister (SFMT) generator.

The Monte Carlo algorithm has advantages against quasi-Monte Carlo and SFMT for non-smooth integrands. *For relatively small number of points the proposed approach gives much better results than Sobol’s quasi-Monte Carlo integration.*

*In case of smooth functions the proposed algorithm has significant advantage against plain Monte Carlo that uses SFMT generator with respect to the relative error.*
**Figure:** Poor projections of $\Lambda \Pi_\tau$ sequences for high dimensions
A Monte Carlo Method Based on ΛΠτ Sequences


More information about **SFMT** generator:
http://www.math.sci.hiroshima-u.ac.jp/ m-mat/MT/SFMT/index.html