MC and RQMC on GPU

Giray Ökten
Department of Mathematics
Florida State University
(Based on research with Linlin Xu)

July 14, 2012
Estimate

\[ I = \int_{[0,1)^s} f(x) \, dx \]

using sums of the form

\[ I_N = \frac{1}{N} \sum_{n=1}^{N} f(q^{(n)}) \]

- Monte Carlo \( q^{(n)} \) is a (pseudo)random vector from \( U(0,1)^s \). Convergence rate is \( O(N^{-1/2}) \)
Estimate

\[ I = \int_{[0,1)^s} f(x) \, dx \]

using sums of the form

\[ I_N = \frac{1}{N} \sum_{n=1}^{N} f(q^{(n)}) \]

- **Monte Carlo \( q^{(n)} \)** is a (pseudo)random vector from \( U(0, 1)^s \). Convergence rate is \( O(N^{-1/2}) \)
- **Quasi-Monte Carlo \( q^{(n)} \)** is the \( n \)th term of an \( s \)-dimensional low-discrepancy sequence (Halton, Sobol’, Faure, Niederreiter). Convergence rate is \( O(N^{-1} \log^s N) \)
How do QMC sequences look?

QMC

MC
Error analysis

- Monte Carlo
  Descriptive statistics for $I_N^{(1)}, I_N^{(2)}, \ldots, I_N^{(m)}$
Error analysis

- Monte Carlo
  Descriptive statistics for $I_N^{(1)}, I_N^{(2)}, \ldots, I_N^{(m)}$

- Quasi-Monte Carlo
  Koksma-Hlawka inequality:

$$|I_N - I| \leq V_{HK}(f) D^*_N(q_n)$$

where

$$D^*_N(q_n) = \sup_{\alpha \in [0,1)^s} \left| \frac{A_N([0,\alpha))}{N} - \alpha_1 \cdot \ldots \cdot \alpha_s \right|$$

and $\alpha = (\alpha_1, \ldots, \alpha_s)$, $A_N([0,\alpha))$ is the number of $q_n$, $n = 1, \ldots, N$ that belong to the interval $[0,\alpha) = [0,\alpha_1) \times \ldots \times [0,\alpha_s)$. 
Randomized quasi-Monte Carlo Methods

Estimate

\[ I = \int_{[0,1]^s} f(x) \, dx \]

using sums of the form

\[ Q(q_u) = \frac{1}{N} \sum_{n=1}^{N} f(q_u^{(n)}) \]

where \( q_u \) is a family of \( s \)-dimensional low-discrepancy sequences indexed by the random parameter \( u \).

**Important properties:**

1. \( E[Q(q_u)] = I \)
Randomized quasi-Monte Carlo Methods

Estimate

\[ I = \int_{[0,1]^s} f(x) \, dx \]

using sums of the form

\[ Q(q_u) = \frac{1}{N} \sum_{n=1}^{N} f(q_u^{(n)}) \]

where \( q_u \) is a family of \( s \)-dimensional low-discrepancy sequences indexed by the random parameter \( u \).

**Important properties:**

1. \( E[Q(q_u)] = I \)
2. \( \text{Var}(Q(q_u)) = O\left(N^{-2}(\log N)^{2s}\right) \) (or, better, for certain integrands and RQMC methods)
Randomized quasi-Monte Carlo Methods

Estimate

\[ I = \int_{[0,1]^s} f(x) \, dx \]

using sums of the form

\[ Q(q_u) = \frac{1}{N} \sum_{n=1}^{N} f(q_u^{(n)}) \]

where \( q_u \) is a family of \( s \)-dimensional low-discrepancy sequences indexed by the random parameter \( u \).

**Important properties:**

1. \( E[Q(q_u)] = I \)
2. \( \text{Var}(Q(q_u)) = O \left( N^{-2}(\log N)^{2s} \right) \) (or, better, for certain integrands and RQMC methods)
3. \( |Q(q_u) - I| \leq V(f)D_N^*(q_u) \)
Current methods

- Scrambled \((t, m, s)\)-nets and \((t, s)\)-sequences (Owen; Hickernell)
- Alternative scrambling methods for \((t, m, s)\)-nets and \((t, s)\)-sequences, such as *random digit scrambling*, and *random linear scrambling* (Matoušek)
- Random shifting (Cranley & Patterson; Tuffin)
- Random-start Halton sequences (Wang & Hickernell; Ökten)
The \( n \)th term of the van der Corput sequence, \( \phi_b(n) \), in base \( b \), is defined as follows: First, write the base \( b \) expansion of \( n \):

\[
n = (a_k \cdots a_1 a_0)_b = a_0 + a_1 b + \ldots + a_k b^k,
\]

then compute

\[
\phi_b(n) = (.a_0 a_1 \cdots a_k)_b = \frac{a_0}{b} + \frac{a_1}{b^2} + \ldots + \frac{a_k}{b^{k+1}}. \tag{1}
\]

The Halton sequence in the bases \( b_1, \ldots, b_s \) is \( (\phi_{b_1}(n), \ldots, \phi_{b_s}(n))_{n=1}^{\infty} \).

**Example**

van der Corput sequence in base 2

\[
\begin{align*}
1 & 1 & 3 & 1 & 5 & 3 & 7 \\
\bar{2} & \bar{4} & \bar{4} & \bar{8} & \bar{8} & \bar{8} & \bar{8} & \ldots
\end{align*}
\]
Permuted van der Corput and Halton sequences

The permuted van der Corput sequence generalizes (1) as

$$\phi_b(n) = \frac{\sigma(a_0)}{b} + \frac{\sigma(a_1)}{b^2} + \ldots + \frac{\sigma(a_k)}{b^{k+1}}$$

where $\sigma$ is a permutation on the digit set $\{0, \ldots, b-1\}$. The permuted Halton sequence is obtained from scrambled van der Corput sequences in the usual way.
von Neumann-Kakutani transformation

An ergodic and measure-preserving transformation $T : [0, 1) \rightarrow [0, 1)$, constructed inductively, by a *splitting and stacking* process.

Splitting & stacking process

$T : [0, 1) \rightarrow [0, 1)$

- The orbit of 0 under $T$ is the van der Corput sequence
von Neumann-Kakutani transformation

An ergodic and measure-preserving transformation \( T : [0, 1) \rightarrow [0, 1) \), constructed inductively, by a \textit{splitting and stacking} process.

\[ T : [0, 1) \rightarrow [0, 1) \]

- The orbit of 0 under \( T \) is the \textit{van der Corput sequence}
- The orbit of any \( x \in (0, 1) \) is a QMC sequence. An independent realization of \textit{random start Halton sequence} is obtained by choosing \( x \in U(0, 1) \)
- Nodes: $p_1, p_2, ...$
- Want $M$ estimates $\theta_m = \frac{1}{N} \sum_{i=1}^{N} f(q_i^m)$, $m = 1, \ldots, M$.
- Sequential computing versus "counter-based" computing.

<table>
<thead>
<tr>
<th></th>
<th>$p_1$</th>
<th>$\ldots$</th>
<th>$p_m$</th>
<th>$\ldots$</th>
<th>$p_M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\times$</td>
<td></td>
<td></td>
<td>$q_1^m$</td>
<td>$\times$</td>
</tr>
<tr>
<td>2</td>
<td>$\times$</td>
<td>$q_2^m$</td>
<td>$\times$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N$</td>
<td>$\times$</td>
<td>$q_N^m$</td>
<td>$\times$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parallel environment

<table>
<thead>
<tr>
<th></th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$\ldots$</th>
<th>$p_i$</th>
<th>$\ldots$</th>
<th>$p_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>$\ldots$</td>
<td></td>
<td></td>
<td>$\times$</td>
<td>$\ldots$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$m$</td>
<td>$q_1^m$</td>
<td>$q_2^m$</td>
<td>$q_i^m$</td>
<td>$q_N^m$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\ldots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
</tbody>
</table>

Massively parallel environment
Numerical results

Problems:

- Pricing collateralized mortgage obligations (CMO)
- Caplet pricing with LIBOR market model

Computing environment:

- CPU: Intel i7-2630QM
- GPU: Nvidia GeForce GT 540M

Sequences used:

- Rasrap, Philox, Twister, Sobol’
Figure: Relation between $N$ and sample variance in pricing MBS
Figure: Relation between $N$ and time in pricing MBS
Figure: Relation between $N$ and efficiency in pricing MBS
Figure: Relation between $N$ and variance in pricing caplet
Figure: Relation between $N$ and running time in pricing caplet
Figure: Relation between $N$ and efficiency in pricing caplet
**Figure:** Speed-up when pricing mortgage-backed securities on GPU
Figure: Speed-up when pricing caplet on GPU
Conclusions

- Rasrap has the best efficiency among all sequences.
- The observed convergence rate for Monte Carlo sequences is between $O(N^{-0.49})$ and $O(N^{-0.52})$.
- The observed convergence rate for Rasrap is between $O(N^{-0.63})$ and $O(N^{-0.85})$.
- Sobol’ sequence in the CURAND library exhibits strange behavior.