The objective

I want to parallelize the algorithm to compute the price of a call option using Pseudo and Sobol Monte Carlo Simulation with MPI. The results of both approximations are compared by doing different trials in two computers: 32 bits, IMac and 64 bits, Sony.

Introduction

The Monte Carlo Simulation helps to approximate the following expectation

$$e^{-rT} \mathbb{E}(f(S_T, K)_+),$$

by using the law of the large numbers, which tell us if that $S_k$ are sequences of identically distributed independent random variables, then with probability 1 the sequence

$$I_N[f] = \frac{1}{N} \sum_{k=1}^{N} f(\text{sequence of random points}, K) = e^{-rT}[\frac{1}{N} \sum_{k=1}^{N} \max(S_k^T - K, 0)],$$

converges to $e^{-rT} \mathbb{E}(f(S_T, K)_+)$. 

The random numbers are extracted from a uniform distribution in the interval from $[0,1)$, the Box-Muller algorithm is used to get a $N(0,1)$ distribution and compute
\[ f \left( S_0 e^{(r-\frac{1}{2}\sigma^2)T + \sigma \sqrt{T} \, N(0,1) - k} \right)_+ , \]

I do this many times (N) and take the average. I multiply this average by \( e^{-rT} \) to get present value of this payoff.

**Previous Research on Sobol’ sequences.**

The evolution of the algorithm to generate Sobol’s sequences has been directed through Owen [2]. He proposed scrambling the sequence to maintain its low discrepancy. However, many have pointed out that scrambling is difficult to implement and time consuming.

Emanouil I. Atanassov is the author of a new generation algorithm that allows consecutives terms of the scrambled Sobol’ sequence to be obtained with essentially two operations per coordinate: one floating point addition and one bit-wise xor operation. This omits operations that are needed only once per tuple.

His approach is based in floating-point representation, which avoids the multiplication and conversion from integer to floating point.

Another interesting feature of his algorithm is that Quasi-Monte Carlo calculations can be performed in parallel. In \( n \) - processors machine there are essentially two approaches that are used: blocking and leapfrogging. I am using blocking where the processor \( p_i \) computes and uses the terms of the sequence with indexes from \( x - coordinate \ i \) up to \( x - coordinate \ (i + 1) \) (but not included). Also, the difference from sequential version is only in the parameters passed to the initialization routine. No further inter-processor communication is required.

**Algorithms**

**Sobol’ sequences algorithm.**

1. Store the terms of the sequence with one added to them in a floating point array, \( y_j^k, k = 1, \ldots, s \).
2. Store the twisted direction numbers in an array with columns indexed by dimensions. It is computed once for all in the initialization.
3. The coordinate of the next term of sequence \( y^k_{j+1} \) follows from \( y^k_j \) through a single \( \text{xor} \) operation with appropriated twisted direction numbers, what remains in memory is the floating-point representation of \( 1 + x \) and subtract one to get \( x \). This rounded to zero.

4. Return the control to the user when one more term of the sequence is needed.

A reference of the implementation of \textit{single} – and \textit{double} – \textit{precision} written in C by Atanassov is given the link in [1].

\textbf{Call Option Price with Pseudo and Sobol Monte Carlo Algorithm.}

1. I have some warning cast from pointer to integer that I don’t know how to fix them. I showed them to Andreas. They are coming from the file “sobolgen.c” by Atanassov.

2. The first step was to define a model for the risk-neutral stock price \( S_T^k \) evolution,

\[
dS_T = rS_Td_T + \sigma S_TdW_T
\]

3. The equation is solved by passing to the log and using Ito’s lemma

\[
d\log S_T = (r - \frac{1}{2}\sigma^2)d_T + \sigma dW_T
\]

4. The algorithm use two generators of Uniform \([0,1]\), pseudo (1) and Sobol (2).

5. Box-Muller is used to get Normal \((0,1)\) numbers from Uniform\([0,1]\).

6. As this process is constant-coefficient, it has the solution and \( W_T \) is a Brownian motion, \( W_T \) is distributed as a Gaussian with mean zero and variance \( T \), so we can write

\[
W_T = \sqrt{T}N(0,1),
\]

\[
S_T = S_0e^{((r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}N(0,1))}
\]
7. The price of the call option is therefore

\[ e^{-rT} \mathbb{E} \left( f \left( S_0 e^{\left( \frac{r-1}{2} \sigma^2 \right) T + \sigma \sqrt{T} N(0,1) - k \right) \right) \right) \]

8. The expectation is computed by for loop from 0 to N (In Sobol case is the number of terms from the sequence and in the other case is the number of pseudo numbers). MPI_Reduce is taking the average from each processor and compute total sum to be divided by N and multiply by \( e^{-rT} \).

9. The Monte Carlo result is compared with the exact Black-Scholes formula.

**Results**

Sony Intel(R) Core (TM) i7-3612QM **CPU@2.10GHz** RAM 8.00GB 64 bit OS

<table>
<thead>
<tr>
<th>Random/Processors Number</th>
<th>Seconds</th>
<th>Exact Price</th>
<th>MC Price</th>
<th>NPaths</th>
<th>N</th>
<th>Absolute Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sobol/8</td>
<td>6.340026</td>
<td>19.620613</td>
<td>19.561750</td>
<td>2</td>
<td>1000000</td>
<td>5.886e-02</td>
</tr>
<tr>
<td>Pseudo/8</td>
<td>6.575098</td>
<td>19.620613</td>
<td>21.795881</td>
<td>2</td>
<td>1000000</td>
<td>2.175e+00</td>
</tr>
<tr>
<td>Sobol/4</td>
<td>3.427575</td>
<td>19.620613</td>
<td>19.417058</td>
<td>1</td>
<td>1000000</td>
<td>2.036e-01</td>
</tr>
<tr>
<td>Pseudo/4</td>
<td>3.466962</td>
<td>19.620613</td>
<td>21.053336</td>
<td>1</td>
<td>1000000</td>
<td>1.433e+00</td>
</tr>
</tbody>
</table>

iMac Intel Core 2 Duo **CPU@3.06GHz** RAM 4.00GB 32 bit OS

<table>
<thead>
<tr>
<th>Random/Processors Number</th>
<th>Seconds</th>
<th>Exact Price</th>
<th>MC Price</th>
<th>NPaths</th>
<th>N</th>
<th>Absolute Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sobol/4</td>
<td>12.831499</td>
<td>19.620613</td>
<td>19.731652</td>
<td>2</td>
<td>1000000</td>
<td>1.110e-01</td>
</tr>
<tr>
<td>Pseudo/4</td>
<td>12.721947</td>
<td>19.620613</td>
<td>25.817066</td>
<td>2</td>
<td>1000000</td>
<td>6.197e+00</td>
</tr>
<tr>
<td>Sobol/4</td>
<td>18.478064</td>
<td>19.620613</td>
<td>19.446532</td>
<td>3</td>
<td>1000000</td>
<td>1.741e-01</td>
</tr>
<tr>
<td>Pseudo/4</td>
<td>18.931408</td>
<td>19.620613</td>
<td>29.742907</td>
<td>3</td>
<td>1000000</td>
<td>1.012e+01</td>
</tr>
</tbody>
</table>
**Makefile**

There is a Makefile that can be used to tailor the program based on your hardware and optimization. I took from Atanassov's file. Also, I have attached his files.

**Running in mpiexec**

There are the following to be defined:

1. "number of paths"
2. “from"
3. "scale"
4. "nterms"

I am not using number of paths but definitely can be very useful to improve the results. For pricing exotic options can be excellent. The second argument is the beginning of the sequence. The third argument is the scale. The fourth argument nterms is the N number of Monte Carlo. The fifth argument is the type of random number.

I did different trials and the following examples gave me good approximations:

**Sobol:**

```bash
mpiexec -n 8 MCSobol.out 2 0 0 1000000 2
```

**Pseudo:**

```bash
mpiexec -n 4 MCSobol.out 2 0 0 1000000 1
```

**Conclusions**

1. Atanassov says that if you need less than $2^{23}$ terms of the sequence, single precision may be enough for you, since the terms of the Sobol sequence will be represented without loss of precision. However, this is not true if scrambling is applied.
2. The Monte Carlo convergence is better in double than single precision. I am not sure, if it is related with the number of processors too.
3. Monte Carlo convergence using Sobol’s sequences can be generated with speeds comparable or superior those of pseudo random generators.
4. Monte Carlo using Sobol’ sequences can be implemented very straightforward.
5. Pricing exotic option will be the next step to do.

Acknowledgments
I want to thanks Prof. Andreas Kloeckner and Prof. Marsha Berger for their considerable time. Also, I want to mention that Kangping Zhu helps me to catch up with MPI. Michael Lewis gave me the contact with Kangping and Eduarod Corona helps me to review my fish file.

Reference