MPI
Pricing Call Option
Pseudo and Sobol
Monte Carlo

HPCFall2012
Florencia Ortiz
12/18/2012

Monte Carlo Method

- Pricing a Call Option

- $I_N[f] = \frac{1}{N}\sum_{k=1}^{N} f(sequence\ of\ random\ points, K)$ $=e^{-rT}[\frac{1}{N}\sum_{k=1}^{N} \max({S_T}^k$-K,0)]
- $I[f] = Black\ Scholes\ formula$

$$\varepsilon = |I[f] - I_N[f]|$$

  - Slow convergence: $\varepsilon_N = \frac{\sigma(f)}{N^{1/2}}$
    - $f(\sigma)$ can be decreased by : antithetic variables or control variates.
    - But the rate the converge remains: $\varepsilon_N \sim \frac{1}{N^{1/2}}$

Monte Carlo Method

- MC with Sobol
  - Improve convergence in Monte Carlo $\varepsilon_N = \frac{O(\ln N)^n}{N}$
  - Discrepancy is a measure of deviation of uniformity.
  - Best uniformity of distribution for as N goes to infinity.
  - Parallelization can be achieved by changing parameters in the "Preprocessing" part of the code.
  - No inter-processor communication is needed.

Monte Carlo Method

- MC with Sobol
  - Improve convergence in Monte Carlo $\varepsilon_N = \frac{O(\ln N)^n}{N}$
  - If N= $2^k, k\ is\ integer$
  - Discrepancy is a measure of deviation of uniformity.
  - Best uniformity of distribution for as N goes to infinity.
  - Parallelization can be achieved by changing parameters in the "Preprocessing" part of the code.
  - No inter-processor communication is needed.

Monte Carlo Method

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

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

Monte Carlo Method

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

Random/ Processors Number	Seconds	Exact Price	MC Price	nPaths	N	Absolute Error
Sobol/4	12.831499	19.620613	19.731652	2	1000000	1.110e-01
Pseudo/4	12.721947	19.620613	25.817706	2	1000000	6.197e+00
Sobol /4	18.478064	19.620613	19.446532	3	1000000	1.741e-01
Pseudo/4	18.931408	19.620613	29.742907	3	1000000	1.012e+01

Monte Carlo Method

Precision Floating	Real Number $y = (1 + x)2^r$ $\mathbf{0 < x < 1}$	The most significant	Binary representation exponent part	Binary representation of the mantissa part
Single	$y\ is\ d_0 d_1 \ldots d_{31}$	d_0= 0	$d_1 \ldots d_8$ r+127 r is the floating point exponent -126 < r < 127	$d_9 \ldots d_{31}$ $x2^{23}$
Double	$y\ is\ d_0 d_1 \ldots d_{63}$	$d_0 = 0$	$d_1 \ldots d_{11}$ r is the floating point exponent -1022 < r < 1023	$d_{12} \ldots d_{63}$ $x2^{52}$

Monte Carlo Method

Atanassov's Algorithm Sobol Generator

Avoids the multiplication and conversion from integer to floating point

Single	Double
1. X ~[0,1) 2. Y is the mantissa 3. Y is stored as a 32 bit integer 4. If one xor's 001111111 to the nine most-significant bits of y 5. Remains in memory is the floating-point representation of (1+x).	4. If one xor's 001111111111 to the twelve most-significant bits of y

1. Input initial data:
   - if the precision is single, set the number of bits b to 32, and the maximal power of two p to 23, otherwise set b to 64 and p to 52;
   - dimension s;
   - direction vectors $\{a_{ij}\}$, $\quad i = 0, p$, $j = 1, \ldots, s$ representing the matrices $A_1, \ldots, A_d$ (always $a_{ij} < 2^{i+1}$);
   - scrambling terms $d_1, \ldots, d_s$ - arbitrary integers less than 2^p, if all of them are equal to zero, then no scrambling is used;
   - index of the first term to be generated - n;
   - scaling factor m, so the program should generate elements with indices $2^m j + n, \quad j = 0, 1, \ldots$.
2. Allocate memory for $s * l$ b-bit integers (or floating point numbers in the respective precision) $y_1, \ldots, y_s$..
3. Preprocessing: calculate the twisted direction numbers $v_{ij}, \quad i = 0, \ldots, p-1$, $j = 0, \ldots, s$:
   - for all j from 1 to s do
   - for $i = 0$ to $p - 1$ do
   - if i=0, then $v_{ij} = a_{ij} 2^{p-m}$, else

$$v_{ij} = v_{i-1j} \mathbf{xor}(a_{i+m,j} * (2^{p-i-m}));$$

4. Calculate the coordinates of the $n^{\text{-th}}$ term of the Sobol' sequence (with the scrambling applied) using any known algorithm (this operation is performed only once). Add +1 to all of them and store the results as floating point numbers in the respective precision in the array y.
5. Set the counter N to $\lfloor \frac{n}{2^m} \rfloor$.
6. Generate the next point of the sequence:
   - When a new point is required, the user supplies a buffer x with enough space to hold the result.
   - The array y is considered as holding floating point numbers in the respective precision, and the result of subtracting 1. from all of them is placed in the array x.
   - Add 1 to the counter N;
   - Determine the first nonzero binary digit k of N so that $N = (2M + 1)2^k$ (on the average this is achieved in 2 iterations);
   - consider the array y as an array of b-bit integers and updated it by using the k^{th} row of twisted direction numbers:
     for $i = 1$ to d do
   - $y_i = y_i xor v_{ki}$.
   - return the control to the user. When a new point is needed, go to 6.

Monte Carlo Method

- Conslusion
  - Monte Carlo Convergence with Sobol's sequences can be generated with speeds comparable or superior to those of pseudo random generators.
  - MPI is straightforward to be implemented to do Monte Carlo with Sobol's sequences.
  - Pricing Exotic Options can be easily implemented.

Monte Carlo Method

- Acknowledgments
  - The author would like to thanks Prof. Andreas Kloeckner and Prof. Marsha Berger for their considerable time.

- References
  - Atanassov E. I.: A new Efficient Algorithm for Generating the Scrambled Sobol's Sequence. (2003)
  - URL: http://parmac1.bas.bg/emanouil/sequence.html.