Computational Finance Project 3

Kai Chen, Rutgers University 02/27/2018

Introduction

This paper reports the main work of Project 3 for Computational Finance (Math 16:642:623). In this project, some classical random number generators are implemented in a C++ configuration. The generated random numbers are first used for option pricing. Then they are used to generate another sets of normal random numbers, which are compared with the true probability distribution for a standard normal variable.

Preliminary

In this project, we assume that the stock price is geometric Brownian motion and all the needed variables are constant:

- Volatility $\sigma = 0.3$
- Initial asset price $S(0) = 100$
- Risk-free interest rate $r = 0.05$
- Strike $K = 110$
- Maturity $T = 1$ year

Closed-form Formula

The arbitrage-free call option price implied by Black-Scholes-Merton model is:

$$C(S_t, t) = e^{-r(T-t)}[S_t e^{r(T-t)} N(d_1) - KN(d_2)]$$

where,

$$d_1 = \frac{1}{\sigma\sqrt{T-t}}[\ln(\frac{S_t}{K}) + (r + \frac{1}{2}\sigma^2)(T-t)]$$

$$d_2 = d_1 - \sigma\sqrt{T-t}$$

The prices of this European call option calculated by our program is:

$$c = 10.0201$$

Benchmark

This result is identical to the benchmark that we use, as is shown in the following screen-shots:

Black-Scholes 1973 options on non-dividend paying stocks

Implementation By Espen Gaarder Haug Copyright 2006

Time in :	Years	Long	Call
Stock price (S)		100.00	
Strike price (X)		110.00	
Time to maturity (T)		1.0000	
Risk-free rate (r)		5.00%	Continuou
Volatility (σ)		30.00%	
Forward price		105.1271	
Option Value		10.0201	

Random Number Generation

Problem 2.1 (a) L'Ecuyer Generator

L'Ecuyer generator is a combined recursive multiple random number generator. The algorithm is given in the following picture (from Glasserman):

```
#define m1 2147483647
#define m2 2145483479
#define a12 63308
#define a13 -183326
#define a21 86098
#define a23 -539608
#define q12 33921
#define q13 11714
#define q21 24919
#define q23 3976
#define r12 12979
#define r13 2883
#define r21 7417
#define r23 2071
#define Invmp1 4.656612873077393e-10;
int x10, x11, x12, x20, x21, x22;

int    Random()
       {
       int h, p12, p13, p21, p23;
       /* Component 1 */
       h = x10/q13; p13 = -a13*(x10-h*q13)-h*r13;
       h = x11/q12; p12 = a12*(x11-h*q12)-h*r12;
       if(p13<0) p13 = p13+m1; if(p12<0) p12 = p12+m1;
       x10 = x11; x11 = x12; x12 = p12-p13; if(x12<0) x12 = x12+m1;
       /* Component 2 */
       h = x20/q23; p23 = -a23*(x20-h*q23)-h*r23;
       h = x22/q21; p21 = a21*(x22-h*q21)-h*r21;
       if(p23<0) p23 = p23+m2; if(p21<0) p21 = p21+m2;
       /* Combination */
       if (x12<x22) return (x12-x22+m1); else return (x12-x22);
       }

double Uniform01()
       {
       int Z;
       Z=Random(); if(Z==0) Z=m1; return (Z*Invmp1);
       }
```

Problem 2.1 (b) Fishman Acceptance-Rejection Algorithm

This method generates normals from double exponential. Fishman illustrates the use of the acceptance-rejection method by generating half-normal samples from the exponential distribution. Fishman also notes that the method can be used to generate normal random variables.

The double exponential density is $g(x) = exp(-|x|)/2$ and the normal density is $f(x) = exp(-x^2/2)/\sqrt{2\pi}$. Then the ratio is:

$$\frac{f(x)}{g(x)} = \sqrt{\frac{2}{\pi}}e^{-\frac{1}{2}x^2+|x|} \leq \sqrt{\frac{2e}{\pi}} = c$$

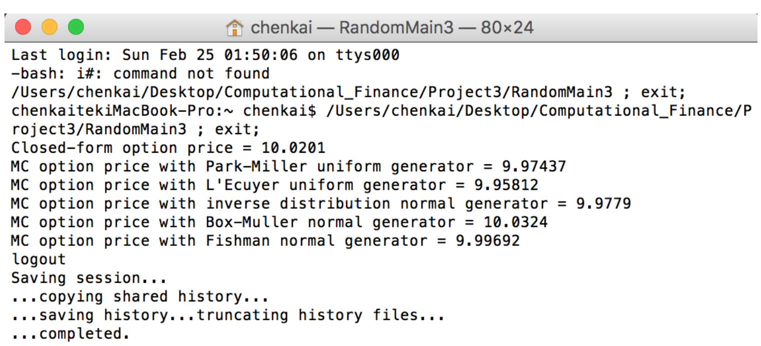
Thus, the normal density is dominated by the scaled double exponential density $cg(x)$. Then the rejection test can be implemented as:

$$u > exp(-\frac{1}{2}(|x|-1)^2)$$

In summery, the combined steps are as follows:

1. generate U_1, U_2, U_3 from Unif[0,1]
2. $X = -log(U_1)$
3. if $U_2 > exp(-0.5(X-1)^2)$
   go to Step 1
4. if $U_3 \leq 0.5$
   $X = -X$
5. return X

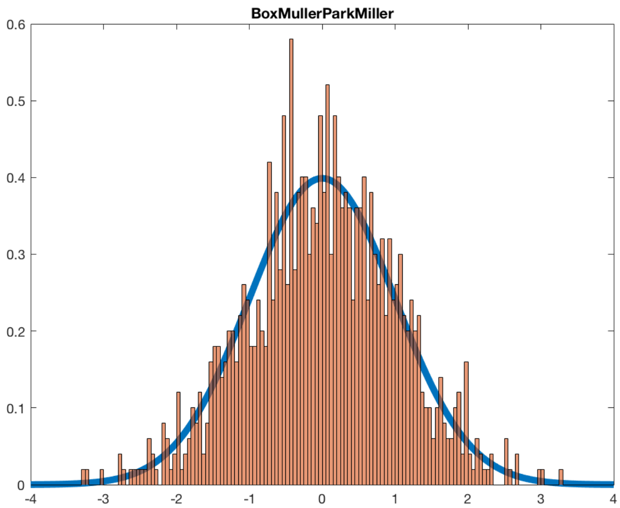
Result for Problem 2.1

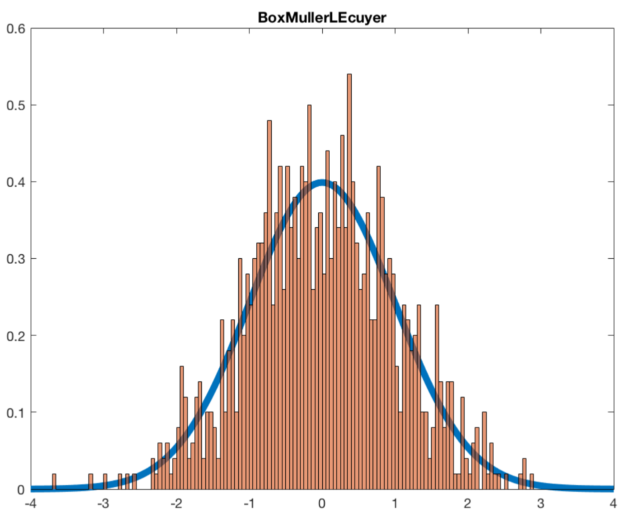


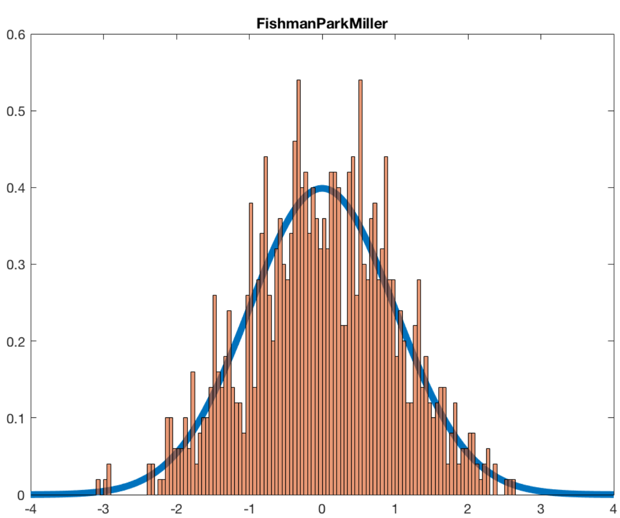
```
chenkai — RandomMain3 — 80×24
Last login: Sun Feb 25 01:50:06 on ttys000
-bash: i#: command not found
/Users/chenkai/Desktop/Computational_Finance/Project3/RandomMain3 ; exit;
chenkaitekiMacBook-Pro:~ chenkai$ /Users/chenkai/Desktop/Computational_Finance/P
roject3/RandomMain3 ; exit;
Closed-form option price = 10.0201
MC option price with Park-Miller uniform generator = 9.97437
MC option price with L'Ecuyer uniform generator = 9.95812
MC option price with inverse distribution normal generator = 9.9779
MC option price with Box-Muller normal generator = 10.0324
MC option price with Fishman normal generator = 9.99692
logout
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.
```

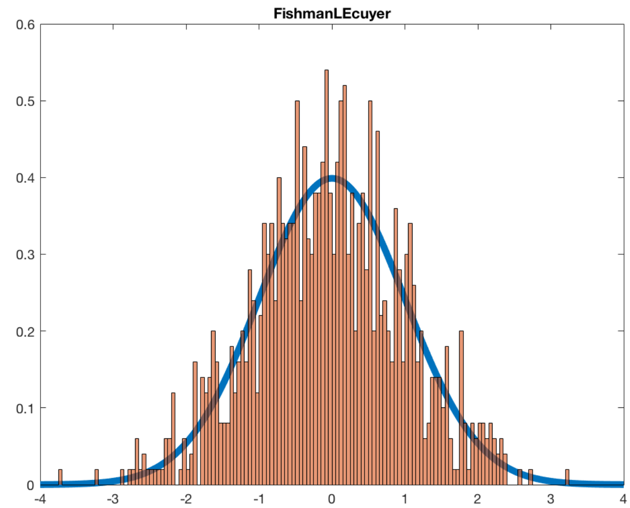
Problem 2.2

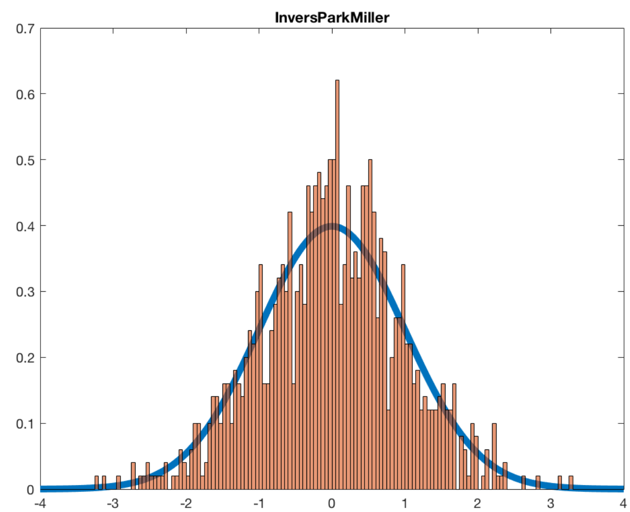
This question asks us to first generate uniform random numbers using Park Miller or L'Ecuyer methods. Then apply Box Muller, inverse transform, or Fishman algorithms to obtain 1,000 normal random numbers. Each result is compared with the graph of the standard normal probability density function. MATLAB is used to plot the results.

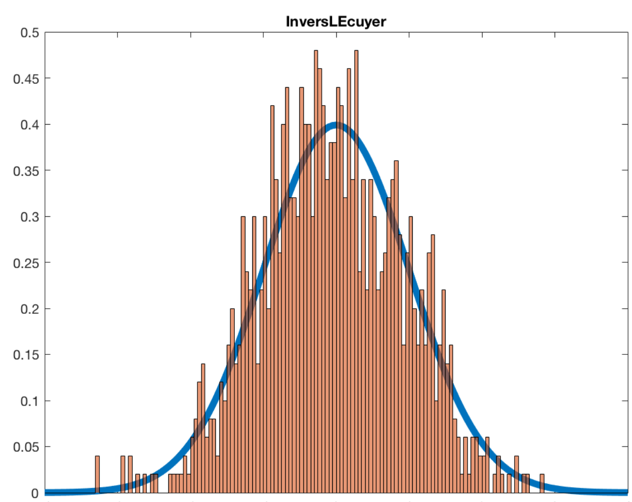

BoxMullerParkMiller
0.6
0.5
0.4
0.3
0.2
0.1
0
-4
-3
-2
-1
0
1
2
3
4


BoxMullerLEcuyer
0.6
0.5
0.4
0.3
0.2
0.1
0
-4
-3
-2
-1
0
1
2
3
4


FishmanParkMiller
0.6
0.5
0.4
0.3
0.2
0.1
0
-4
-3
-2
-1
0
1
2
3
4


FishmanLEcuyer
0.6
0.5
0.4
0.3
0.2
0.1
0
-4
-3
-2
-1
0
1
2
3
4


InversParkMiller
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-4
-3
-2
-1
0
1
2
3
4


InversLEcuyer
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-4
-3
-2
-1
0
1
2
3
4