MAP 5611 Intro to Computational Finance HW5

Student: Sen Zhang

I. Executive Summary

Firstly, in this report the routine of random number generator was tested to be correctly. The mean and variance of the generated normally distributed random numbers is around 0 and 1. Also a graph of p.d.f. with respect to those numbers is plot the same as that of normally distribution. Then it is tested that the random number sequence changes when the seed is changed.

Secondly, from the linear relationship of error and Δt, we can see that the implementation works correctly for solving a ODE IVP for the Black-Scholes model. And the weak convergence rate of Euler-Maruyama is around 1.

At last, the price of European call option is around $98.6.

II. Statement of Problem

To integrate a stochastic ODE initial value problem for the Black-Scholes model

$$\frac{dS}{S} = rdt + \sigma\phi\sqrt{dt}$$

Where ϕ is a random variable chosen from N(0, 1).

1. Write and test a procedure to compute normally distributed random numbers.
2. Implement the Euler-Maruyuma method to the integrate the SDE and perform following 2 tests:
   a) For σ =0 and your choice of r to show that the implementation works correctly
   b) For $\sigma \neq 0$ and your choice of r to determine the weak convergence rate of the approximation and compare to the expected.
3. The price of a European call option is the discounted expected value of the payoff function.

$$C(S_0,T) = e^{-rT}E[\max(S(T)-K,0)]$$

With parameters:

T	t	r	σ	K	S_0
54/365	0	0.0675	0.135	3425.0	3441.0

III. Description of The Mathematics

Normally distributed random number

The p.d.f of Gaussian distribution is $p(x,y) = \frac{1}{2\pi} e^{-\frac{x^2+y^2}{2}}$

Convert to polar form

$$p(r) = \frac{1}{2\pi} e^{-\frac{r^2}{2}}$$

For Poisson distribution, $P(x) = ae^{-ax}$ we have poisson distributed random number:

$$x = -\frac{1}{a}\ln(1-u)$$

with uniform random number u

Thus $r = \sqrt{-2\ln(1-u_1)}$

So

$$x = r\cos(2\pi u_2)$$
$$y = r\sin(2\pi u_2)$$

For $1-u_1$ and u_1 are both uniformly distributed random variables.

Then we can write

$$x = \sqrt{-2\ln(u_1)}\cos(2\pi u_2)$$
$$y = \sqrt{-2\ln(u_1)}\sin(2\pi u_2)$$

Euler-Maruyama

$$dS = rSdt + \sigma SdB$$

$$S_{n+1} - S_n = \int_{t_n}^{t_{n+1}} rSdt + \int_{t_n}^{t_{n+1}} \sigma SdB$$
$$\approx rS\Delta t + \sigma S\Delta B$$

B is geometric Brownian Motion

$$\Delta B \sim N(0, \Delta t)$$

$$\phi \sim \sqrt{\Delta t} N(0,1)$$

$$S_{n+1} = S_n + rS_n\Delta t + \sigma S_n \phi\sqrt{\Delta t}$$

IV. Description of The Algorithm

Algorithm for Normally Distributed Random Numbers:

$i = time(NULL)$
$set\ seed(i)$
$for\ n = 0\ to\ Nstep - 1$
$if\ (i \bmod 2 = 0)$
$u_1 = rand() / RAND\_MAX$
$u_2 = rand() / RAND\_MAX$
$end\ \text{if}$
$random[i] = \sqrt{-2\ln(u_1)}\cos(2\pi u_2)$
$random[i+1] = \sqrt{-2\ln(u_1)}\sin(2\pi u_2)$
$next\ n$

Algorithm for Euler-Maruyuma:

$set\ seed$
$for\ k = 1\ to\ Ntrial$
$for\ k = 1\ to\ Nstep - 1$
$\phi = random$
$S_{n+1} = S_n + rS_n\Delta t + \sigma S_n \phi\sqrt{\Delta t}$
$next\ n$
$sum = sum + S$
$next\ n$

V. Results

1.

a)

Problem 1:

In the first algorithm, $u_1 \sim$ uniform(0, 1)

then $P(\ln(u_1) = 0) > 0$

Thus there exists that some random numbers are INF.

Problem 2:

Since RAND_MAX = 32767, the normally random numbers are included in the interval (-4.56, 4.56).

And

$$P(-\infty < x < -4.56 \ or \ 4.56 < x < \infty) = 5.11531 \times 10^{-6}$$

Which means its probability is small enough to ignore it.

b)

Set seed(1)

Table 1 Means and Vars of Different Number of Random Numbers

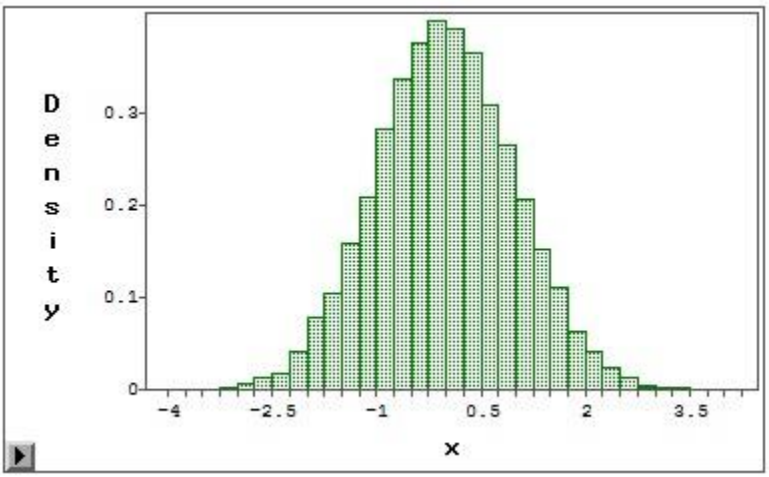
Number	Mean	Variance

100	0.00580708	1.08216
1000	-0.0226552	1.0693
5000	-0.0142078	1.00951
10000	-0.0125921	1.00789
20000	-0.0156921	0.996598
30000	-0.0116631	0.995044

From table 1, we can see that the mean and var are nearly to 0 and 1.

Then plot the pdf of 10000 random numbers with seed(1) in SAS

Graph 1 Probability Density Function of Random Numbers



From graph 1, we can see that the random numbers generated is normal distribution.

When the seed is changed, the random numbers will be different.

2.

a)

Set S(0)=1, T=10, r=0.1

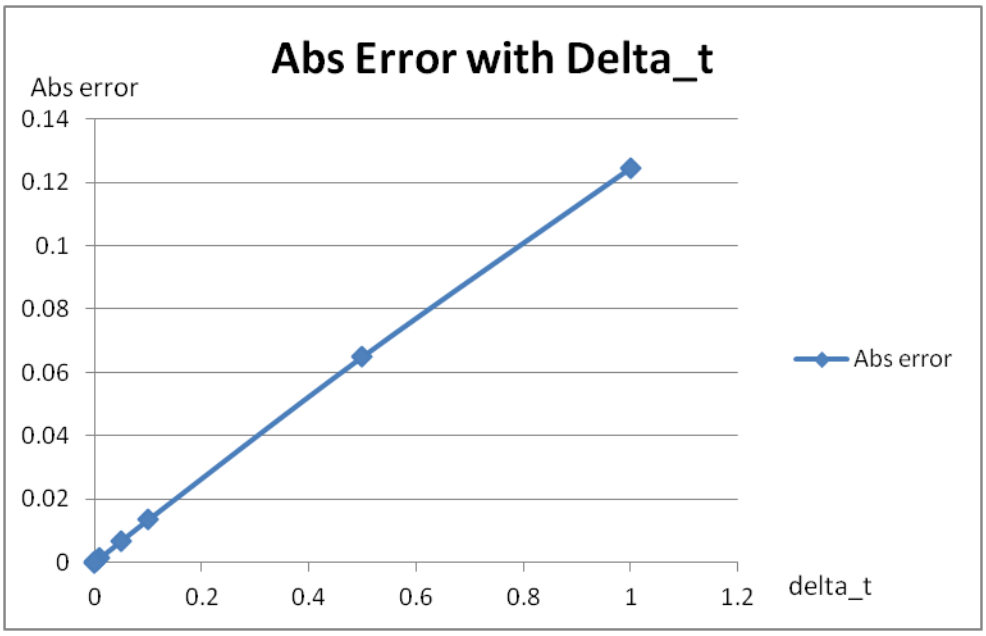
Table 2 Approximation of S(T) with Different Δt

Δt	Approx of S(T)	Real value of S(T)	Abs error

1	2.59374	2.71828	0.124539
0.5	2.6533	2.71828	0.0649841
0.1	2.70481	2.71828	0.013468
0.05	2.71152	2.71828	0.00676471
0.01	2.71692	2.71828	0.0013579
0.005	2.7176	2.71828	0.000679259
0.001	2.71815	2.71828	0.000135902
0.0005	2.71821	2.71828	6.79593e-5

Then plot the graph of absolute error with Δt

Graph 2 Absolute Error with Δt



From graph 2 we can see that absolute error of approximation is decreasing linearly with decreasing Δt.

For

$$S_{n+1} = S_n + rS_n\Delta t \;\; e \sim \mathrm{O}(\Delta t)$$

From graph 2, we can see

$$\boxed{abserror \sim \mathrm{O}(\Delta t)}$$

Thus the implementation works correctly.

b)

For the weak convergence rate, we have formula

$$E[S(T)] - E[S_{Nstep}] \le K\Delta t^{\gamma}$$

$$\log(|\, E[S(T)] - E[S_{Nstep}]\,|) \le \gamma \log \Delta t + \log K$$

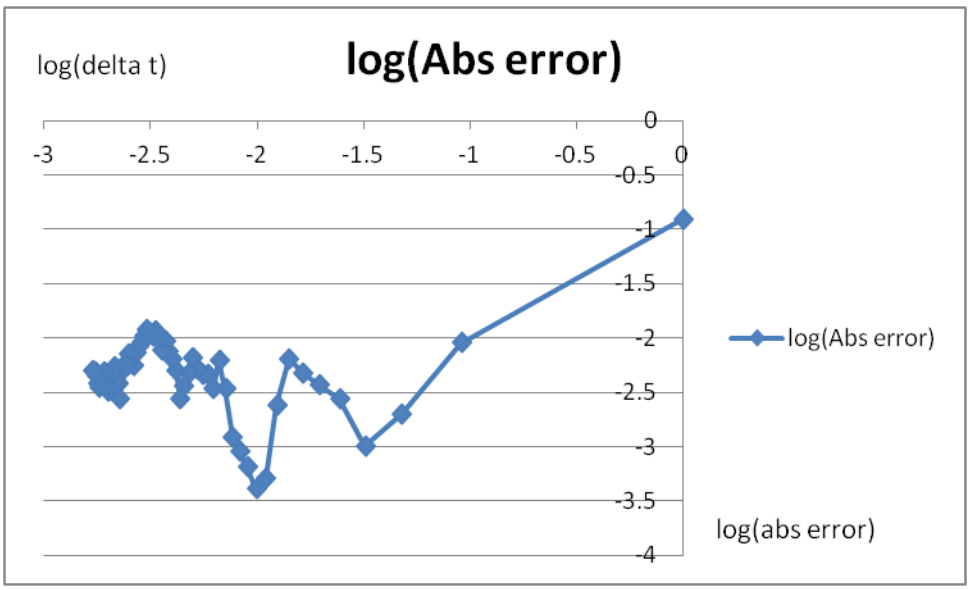
Then set S(0)=1, T=10, r=0.1, the absolute errors are as following:

Table 2 Approximation of S(T) with Different Δt

Δt	Approx of S(T)	Real value of S(T)	Abs error
1	2.5945	2.71828	0.123782
0.5	2.66364	2.71828	0.0546381
0.1	2.70994	2.71828	0.00834503
0.05	2.71654	2.71828	0.00174233
0.01	2.7174	2.71828	0.000881109
0.005	2.71245	2.71828	0.0058325
0.001	2.72159	2.71828	0.00331075
0.0005	2.73028	2.71828	0.0119964

Plot the graph of log(error) with respect to log(Δt) in excel

Graph 3 log(Absolute Error) with log(Δt)



From graph 3, it can be seen that for the right side of the curve

$$\log(\text{abs error}) \le \log \Delta t + \log K$$

Since there exists rounding error when Δt is very small, we can ignore the left side of the curve.

Then the weak convergence rate is around 1.

3.

From graph 3 we can see the optional Δt is included in (0.01, 0.011), which means

$optional\ nstep \sim (990,1000)$

With the value

T	t	r	σ	K	S_0
54/365	0	0.0675	0.135	3425.0	3441.0

The price of European call option is as following:

nstep	Call price
995	98.6683
996	98.6733

997	98.6025
998	98.6317
999	98.6344
1000	98.6488

From this table we can conclude that the price should be around $98.6.

Comparing the result 98.5677 from last homework, this result has at least 2 significant digits.

VI. Conclusions

1. The routine of random number generator was tested to be correctly. The mean and variance of the random numbers is around 0 and 1. And the random number sequence changes when the seed is changed.

2. a) From the linear relationship of error and Δt, we can see that the implementation works correctly.

b) the weak convergence rate is around 1.

3. The price of European call option is around $98.6.

VII. Program Listing

SDE.h

```
#define FM float

#pragma once

class EulerM

{public:

    FM  *Box_Muller(int Nstep);

};
```

SDE.cpp

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <cstdlib>
#define _USE_MATH_DEFINES
#include "math.h"
#include "SDE.h"
#include "windows.h"
using namespace std;

FM*  EulerM::Box_Muller(int Nstep)
{
	//Sleep(1000);
	FM *random=new FM[Nstep+1];

	FM u1, u2;
	for (int i=0; i<Nstep; i++)
	{
		if (i % 2 == 0)
		{
			FM temp = (FM) rand();
			if (temp == 0.0)
			{
				temp = (FM) rand();
			}
			else
```

```
                {
                        u1 = temp/ FM(RAND_MAX);
                }

                temp = (FM) rand();
                if (temp == 0.0)
                {
                        temp = (FM) rand();
                }
                else
                {
                        u2 = temp/ FM(RAND_MAX);
                }

        random[i] = FM( sqrt(-2*log(u1))*cos(2*M_PI*u2));
        random[i+1] = FM( sqrt(-2*log(u1))*sin(2*M_PI*u2));
        }
    }

    FM *point_random;
    point_random = random;

    return point_random;
    delete []random;
}
```

```
int main()
{
	EulerM E;

	int Nstep = 996;

	//FM random[Nstep+1];
	//FM *random = E.Box_Muller(Nstep);
  /*for (int i=0; i<Nstep; i++)
	{
		random[i] = point_random[i];
	}*/
	FM sum;
	/*FM mean, var;
	sum = 0.0;
	for (int i=0; i<Nstep; i++)
	{
		sum = sum + random[i];
		//cout<<random[i]<<endl;
	}
	mean = sum/Nstep;

	sum = 0.0;
	for (int i=0; i<Nstep; i++)
	{
		sum += pow((random[i]-mean), 2);
```

```
                //sum += random[i]*random[i];
        }
        //var = sum/(Nstep-1)-Nstep/(Nstep-1)*mean;
        var = sum/Nstep;

        /*ofstream myFile;
myFile.open( "random.txt");
for (int i=0; i<Nstep;i++)
{
   myFile << random[i] << endl;
}
myFile.close();*/

        /*cout<<"The mean of random numbers is "<<mean<<endl;
        cout<<"The mean of random numbers is "<<var<<endl;*/

        //// Question 2 part a

        //FM S0 = 1.0;
        //FM S;
        //FM r = 0.1;
        //FM T = 10.0;
        //FM sigma = 0.1;

FM S0 = 3441.0;
        FM S;
```

```
        FM r = 0.0675;
        FM T = 54.0/356.0;
        FM sigma = 0.135;
        FM K = 3425.0;
        FM temp;
        FM C;

//FM Error[100];
//      FM Delta_t[100];
//      int ntemp = 10;
//      for(int k=0; k<ntemp; k++){
//              Nstep = 10+100*(k+50);

        FM delta = T/Nstep;
        int Ntrial = 20000;
        sum = 0.0;

        for (int i=0; i<Ntrial; i++)
        {
                srand((unsigned int) i);

          FM *random = E.Box_Muller(Nstep);
    /*for (int i=0; i<Nstep; i++)
          {
                        random[i] = point_random[i];
                }*/
```

```
		S = S0;
		for (int i=0; i<Nstep; i++)
		{
			S = S * (1.0 + r*delta + sigma*sqrt(delta)*random[i]);
		}

		if(S-K>0)
		{
			temp = S-K;
		}
		else
		{
			temp = 0;
		}
		sum = sum + temp;
		//sum = sum + S;
}
S = sum / Ntrial;
C = exp(-r*T)*S;
cout<<"Call price is "<<C<<endl;
//FM error = abs(S0*exp(r*T)-S);
//Error[k] = error;
//Delta_t[k] = delta;
//}
//ofstream myFile;
```

```
//   myFile.open( "error.txt");
//   for (int i=0; i<ntemp;i++)
//   {
//       myFile << Error[i] << "\t" << Delta_t[i] << endl;
//   }
//   myFile.close();


            //cout<<"The error is "<<error<<" S is "<<S<<" Real value is "<<S0*exp(r*T)<<endl;


            return 0;
}
```