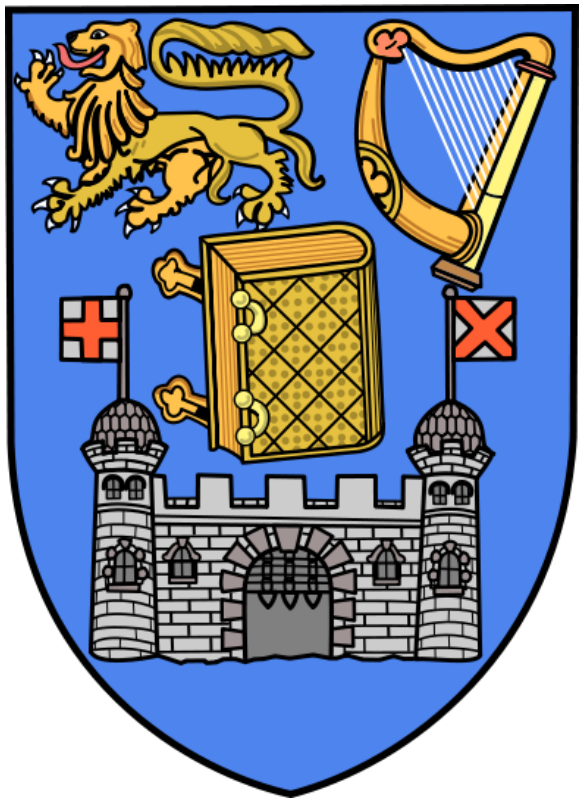
UNIVERSITY OF DUBLIN

TRINITY COLLEGE

SCHOOL OF MATHEMATICS



Course: MA 5637
Stochastic Differential Equations in Finance Report

Supervisor: Dr. Darach Golden

Luo Qiyuan
Student Number: 12330466

5 June, 2014

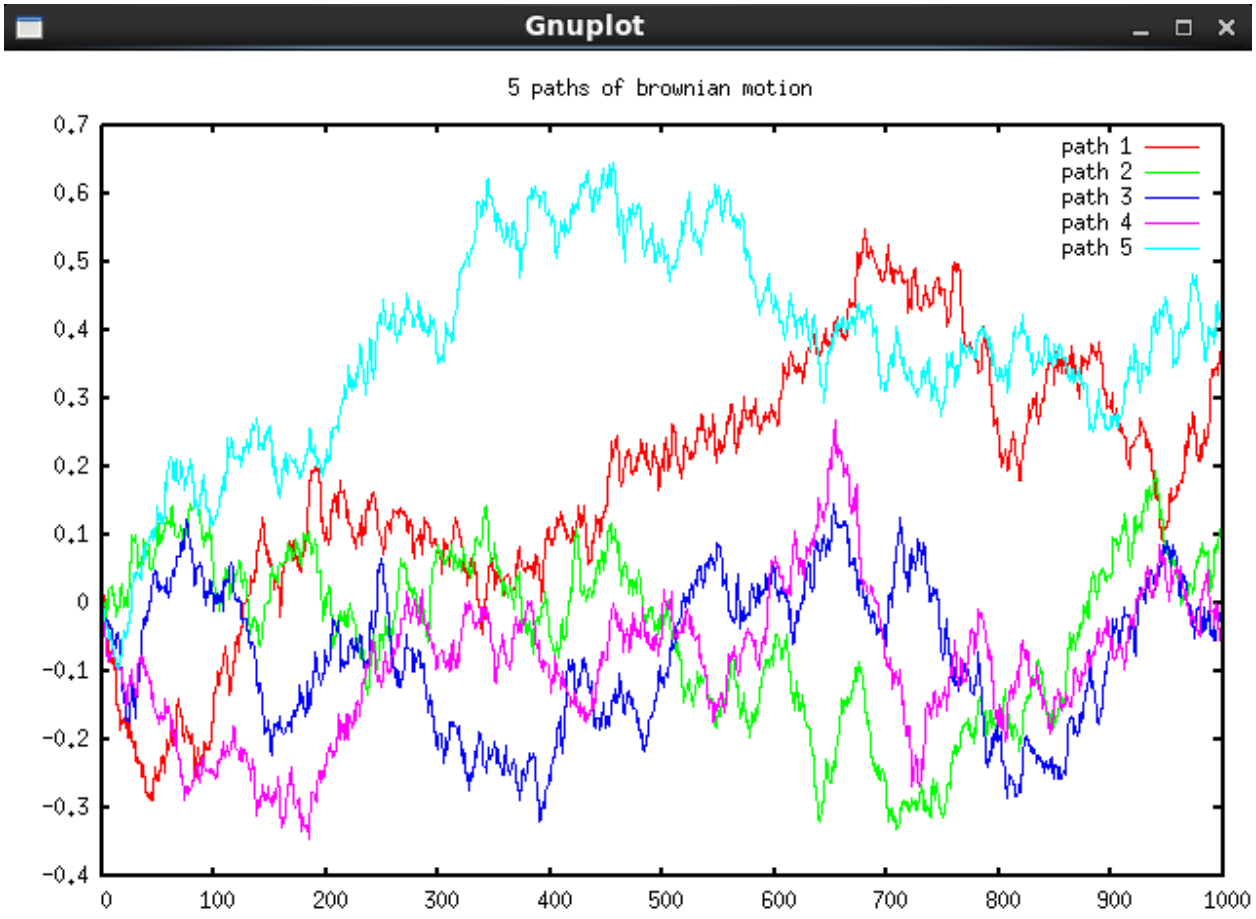
Stochastic Differential Equations in Finance Report

Part 1: Implement random number generator to simulate brownian motion

In this part, I used the gnu_ran_gaussian() from gsl library to generate gaussian distributed numbers.

The following picture is a sample of 5 brownian motions plotted by gnuplot with data generated from my cpp code:

Graph 1



To run the code, we can come into the directory long/1RNG and input $ make test. The graph above is the running result of $ make test.

Part 2: Write software to solve the differential equation

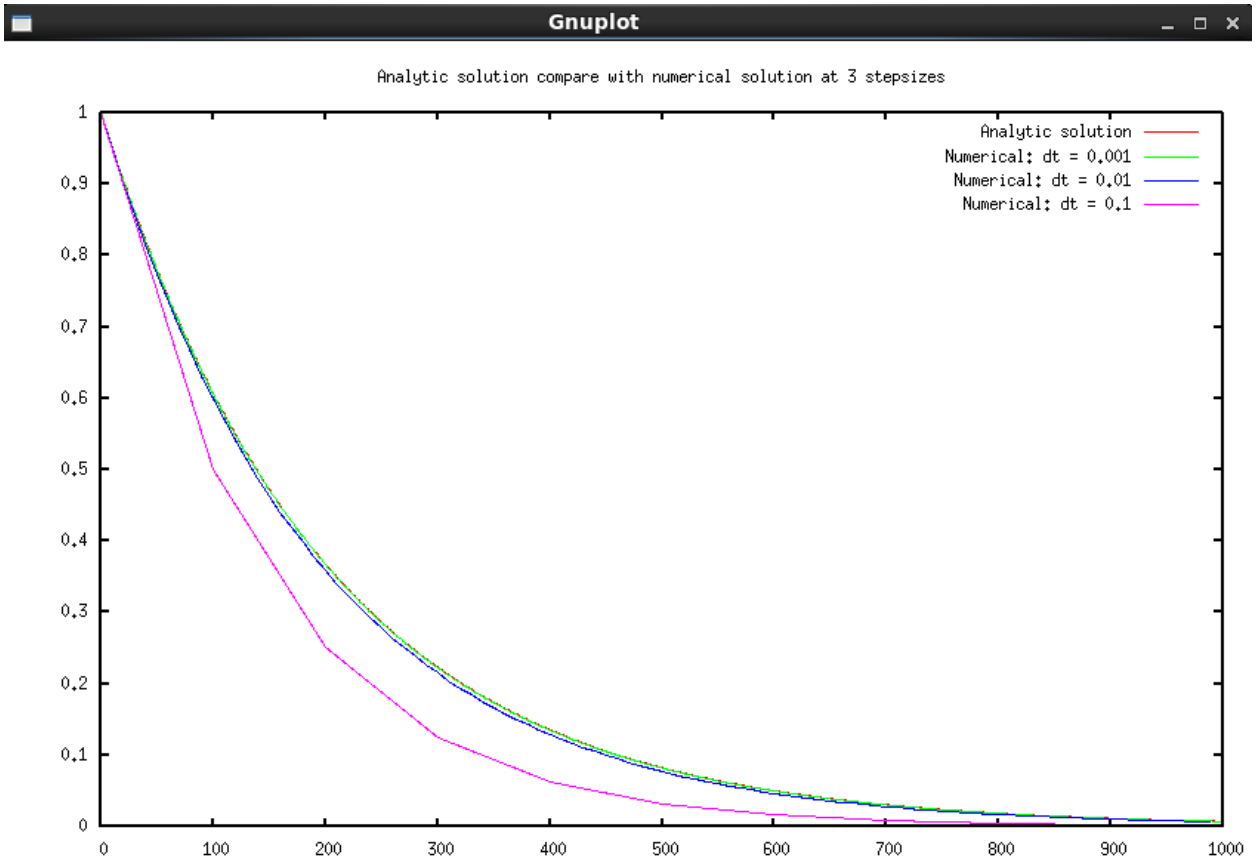
The equation given is:
dx = -5x dt, x(0) = 1,

The analytic solution x = e^(-5t) in c++ language should be written as x = exp(-5 * t).

Solving this equation by euler method, Xn+1 = Xn + h*(f(tn, Xn)).

I implemented euler method with 3 different step sizes inside [0,1], red line is the analytic solution, green line is the numerical solution with dt=0.001, blue line is the numerical solution with dt=0.01, purple line is the numerical solution with dt=0.1. We can see from the following graph that green line is very close to red line which means when dt=0.001 the numerical solution from Euler method is pretty close to analytic solution.

Graph 2



To run the code, we can come into the directory long/2ODE and input $ make test. The graph above is the running result of $ make test.

Part 3: Solution of one dimensional SDEs

In this part I wrote the most complex codes. All codes in part 3 are object oriented and the classes are defined in file class.h. SDE simulation code in part 3.a is contained in EandMoop.cc whilst in part 3.b is separated in to function?.h and EandMoop.cc. The function?.h files contains the functions of milstein method, euler method and analytic solution. function0.h is same as 3.a which was used for test, function1.h is about question 3.b.i, function2.h is about 3.b.ii.

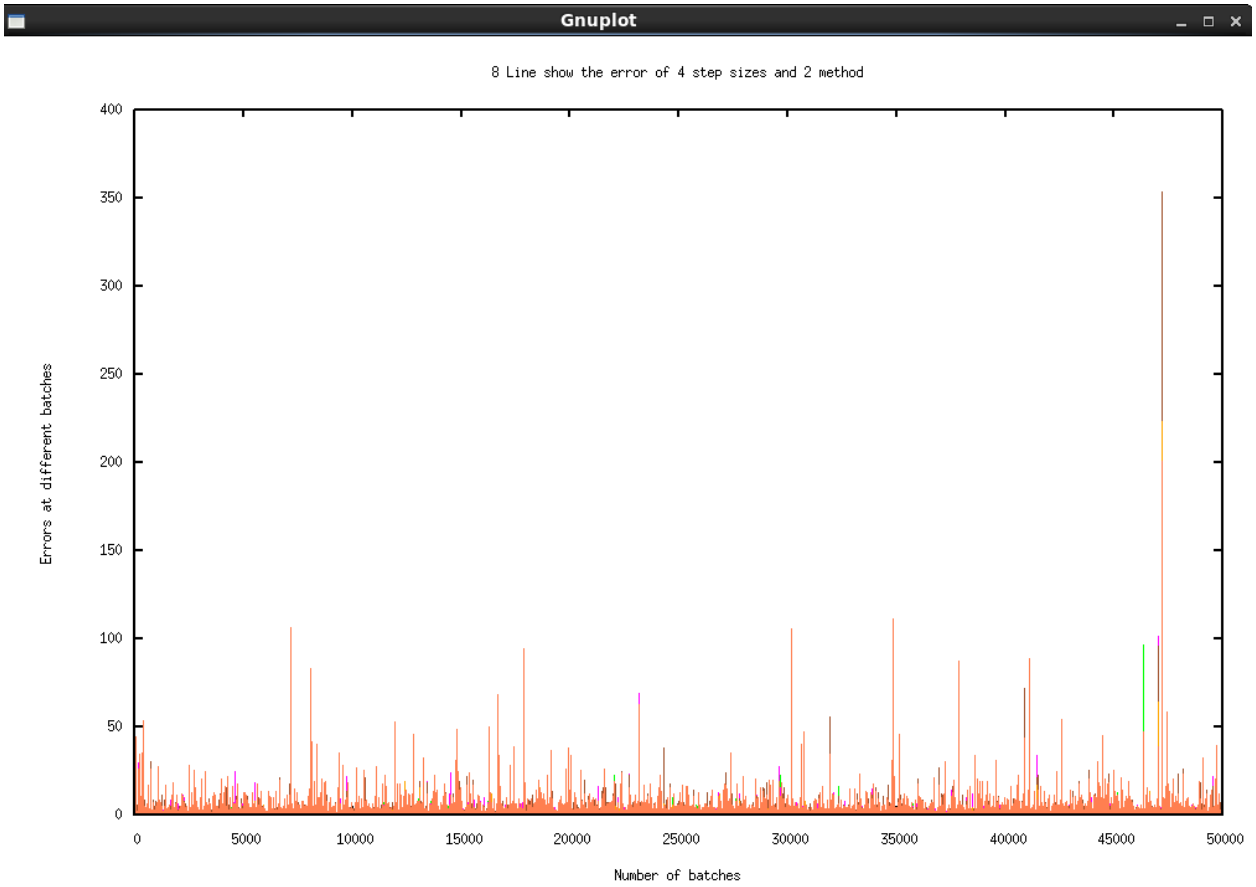
3.a:

A very basic SDE problem should be solved in this part: $dX = aXdt + bXdW$. I set $a=2$, $b=2$ and $X(0)=1$.

To run the code, we can we can come into the directory long/3SDE/part-a/ and input $ make test. The graphs above are the running result of $ make test.

In the following graph, X axis is the number of batches and Y axis is the value of error of the batch. 8 lines in the graph stand for 4 step sizes (1/512, 1/256, 1/128, 1/64) and 2 methods (Milstein method and Euler-Maruyama method).

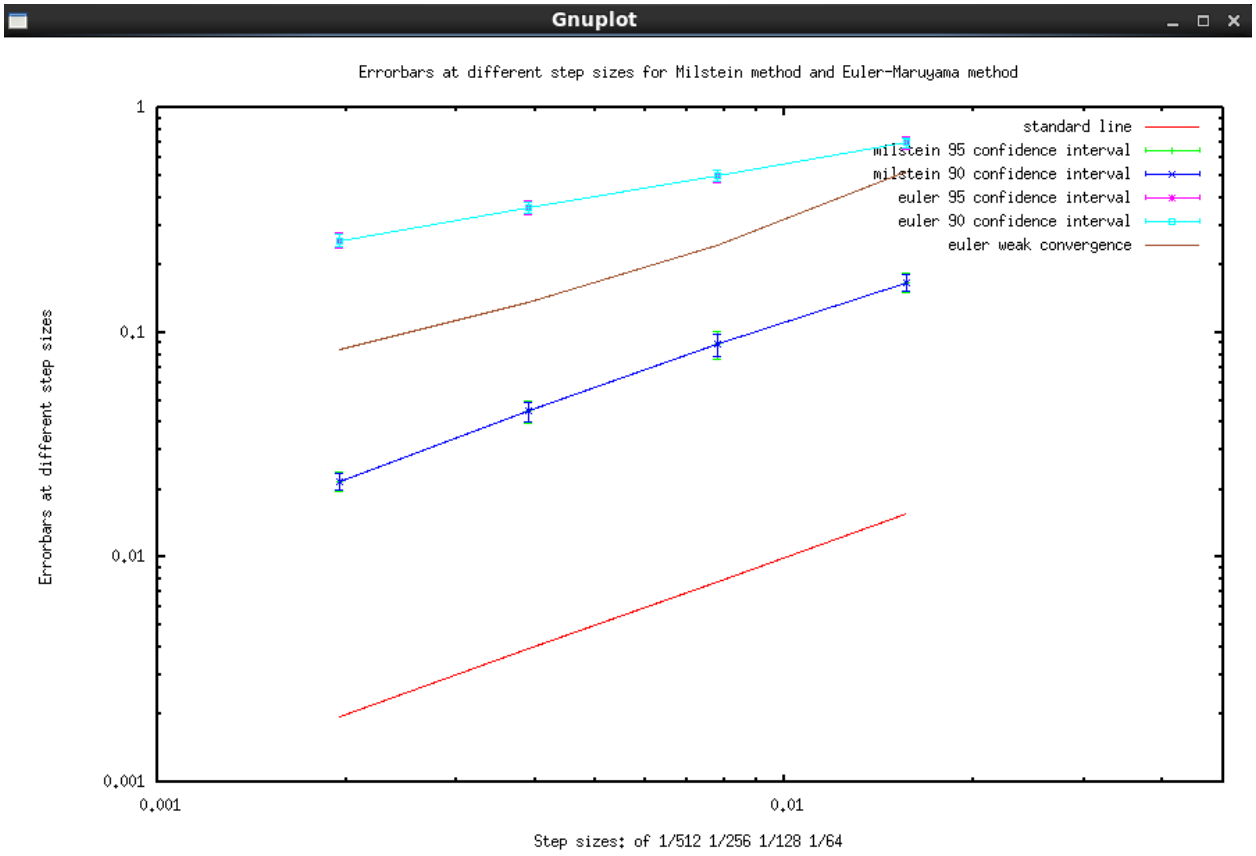
Graph 3



Red line: dt = 1/512 for Milstein method.
Green line: dt = 1/512 for Euler-Maruyama method.
Dark blue line: dt = 1/256 for Milstein method.
Purple line: dt = 1/256 for Euler-Maruyama method.
Light blue line: dt = 1/128 for Milstein method.
Brown line: dt = 1/128 for Euler-Maruyama method.
Yellow line: dt = 1/64 for Milstein method.
Light Brown line: dt = 1/64 for Euler-Maruyama method.

In the graph below, X axis shows the step sizes of the numerical solutions of the SDE and Y axis is the error of each step size and respect error bars. The light blue line shows the strong convergence error of Euler-Maruyama method. The dark blue line shows the strong convergence error of Milstein method. The brown line shows the weak convergence error of Euler-Maruyama method. Red line at bottom is the standard comparing line defined by four points (1/512, 1/512), (1/256, 1/256), (1/128, 1/128), (1/64, 1/64).

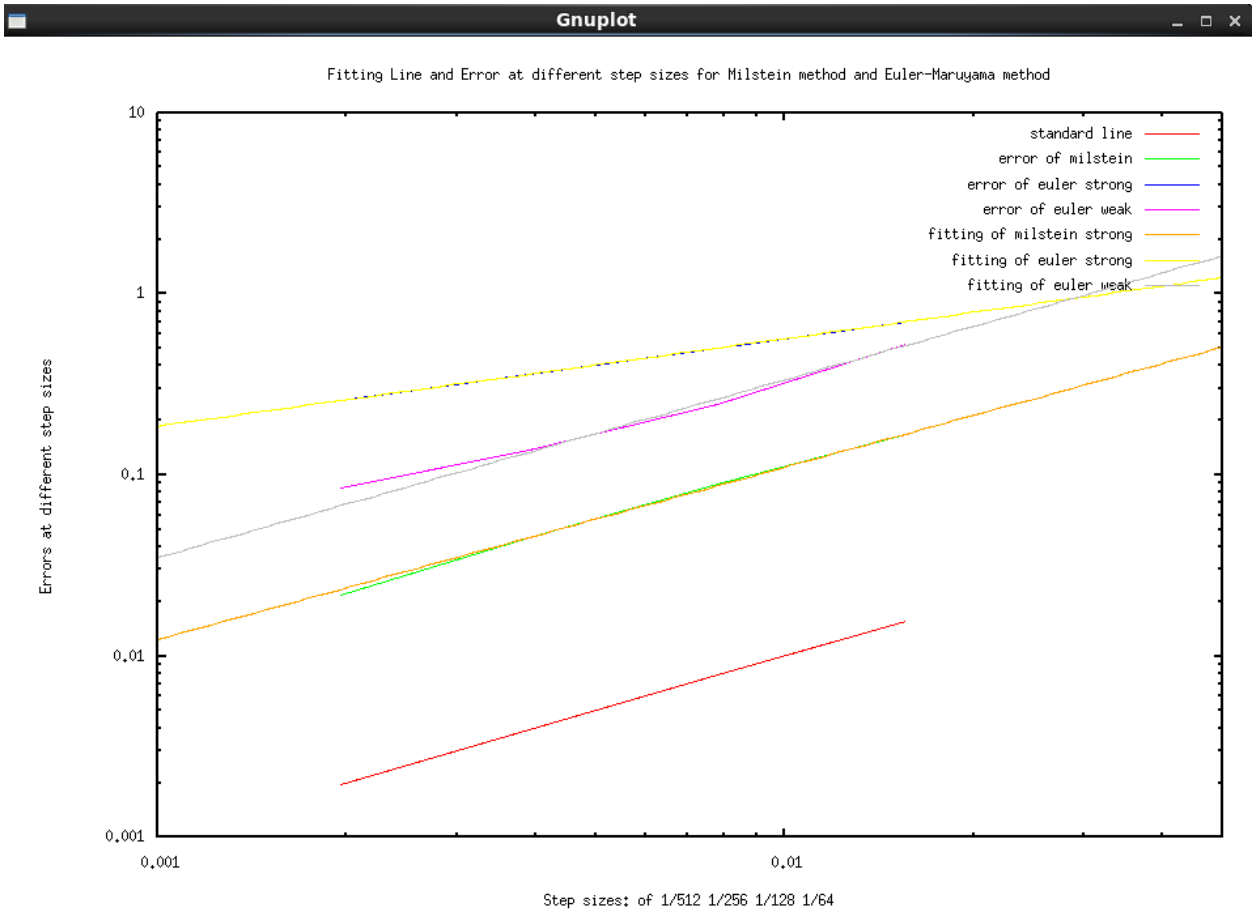
Graph 4



The 95% confidence interval error bars in the graph above is not very clear because it is just tiny larger than 90% confidence interval error bars.

In the graph below, X axis shows the step sizes of the numerical solutions of the SDE and Y axis shows the error of each step sizes and respect fitting lines. The lines' colour and function could be found in the graph.

Graph 5



The test shown above has a perfect test result. All 3 pairs of fitting lines got very good result.

The orange line is the fitting line of milstein strong method, its slope is 0.950794. The slope is close to the expected value 1. The fitting result is shown in the graph below:

Graph 6

```
 Iteration 14
 WSSR        : 5.07314e-06       delta(WSSR)/WSSR   : -1.56999e-07
 delta(WSSR) : -7.96478e-13      limit for stopping : 1e-05
 lambda   : 8.40892e-08

resultant parameter values

a1              = 8.77989
b1              = 0.950794

After 14 iterations the fit converged.
final sum of squares of residuals : 5.07314e-06
rel. change during last iteration : -1.56999e-07

degrees of freedom    (FIT_NDF)                       : 2
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.00159266
variance of residuals (reduced chisquare) = WSSR/ndf   : 2.53657e-06

Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a1              = 8.77989          +/- 0.7064       (8.046%)
b1              = 0.950794         +/- 0.01824      (1.918%)
```

The grey line is the fitting line of euler weak method, its slope is 0.980635. The slope is close to the expected value 1 and the fitting result is shown in the graph below:

Graph 7

```
 Iteration 34
 WSSR        : 0.000649038       delta(WSSR)/WSSR   : -0.0287243
 delta(WSSR) : -1.86431e-05      limit for stopping : 1e-05
 lambda   : 1.39891e-06

resultant parameter values

a3              = 30.6054
b3              = 0.980607
/

 Iteration 35
 WSSR        : 0.000649038       delta(WSSR)/WSSR   : -3.69288e-07
 delta(WSSR) : -2.39682e-10      limit for stopping : 1e-05
 lambda   : 1.39891e-07

resultant parameter values

a3              = 30.61
b3              = 0.980635

After 35 iterations the fit converged.
final sum of squares of residuals : 0.000649038
rel. change during last iteration : -3.69288e-07

degrees of freedom    (FIT_NDF)                       : 2
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.0180144
variance of residuals (reduced chisquare) = WSSR/ndf   : 0.000324519

Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a3              = 30.61            +/- 9.325        (30.47%)
b3              = 0.980635         +/- 0.06927      (7.064%)
```

The yellow line is the fitting line of euler strong method, its slope is 0.482512. The slope is close to the expected value 0.5 and the fitting result is shown in the graph below:

Graph 8

```
 Iteration 7
 WSSR        : 1.0324e-05       delta(WSSR)/WSSR   : -3.85576e-07
 delta(WSSR) : -3.98069e-12     limit for stopping : 1e-05
 lambda   : 1.39891e-06

resultant parameter values

a2              = 5.21579
b2              = 0.482512

After 7 iterations the fit converged.
final sum of squares of residuals : 1.0324e-05
rel. change during last iteration : -3.85576e-07

degrees of freedom    (FIT_NDF)                        : 2
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.002272
variance of residuals (reduced chisquare) = WSSR/ndf   : 5.162e-06

Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a2              = 5.21579          +/- 0.0889       (1.704%)
b2              = 0.482512         +/- 0.00361      (0.7482%)
```

3.b.i:

The SDE given in this question is very similar to Langevin Equation which is usually used to describe the time evolution of a subset of the degrees of freedom.

From the book Numerical Solution of Stochastic Differential Equations I found the analytic solution of

$$dX_t = -aX_t dt + bdW_t \quad (3.1)$$

is

$$X_t = e^{-at}X_0 + e^{-at}\int_0^t e^{as}bdW_s \quad (3.2)$$

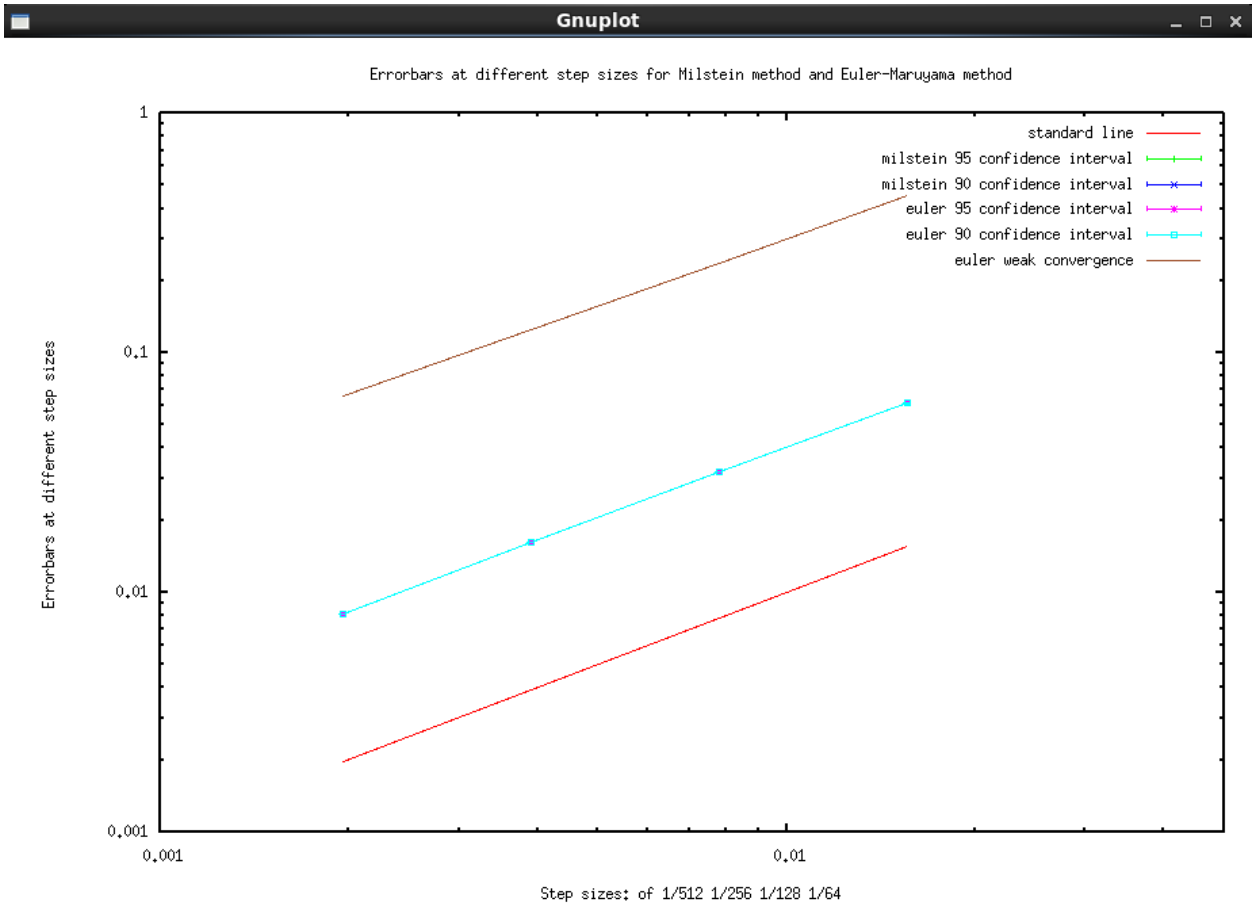
(page 105 of [NUMERICAL])

The integral part of function 3.2 was the difficulty part during the implementation.

To run the code, we can come into the directory long/3SDE/part-b/, open the file EandMopp.cc, uncomment line 12, comment line 11 and 13, save and quit file editing. Then we can input $ make test. The graphs below are the running result of $ make test.

The following graph is the error bar graph of this question.

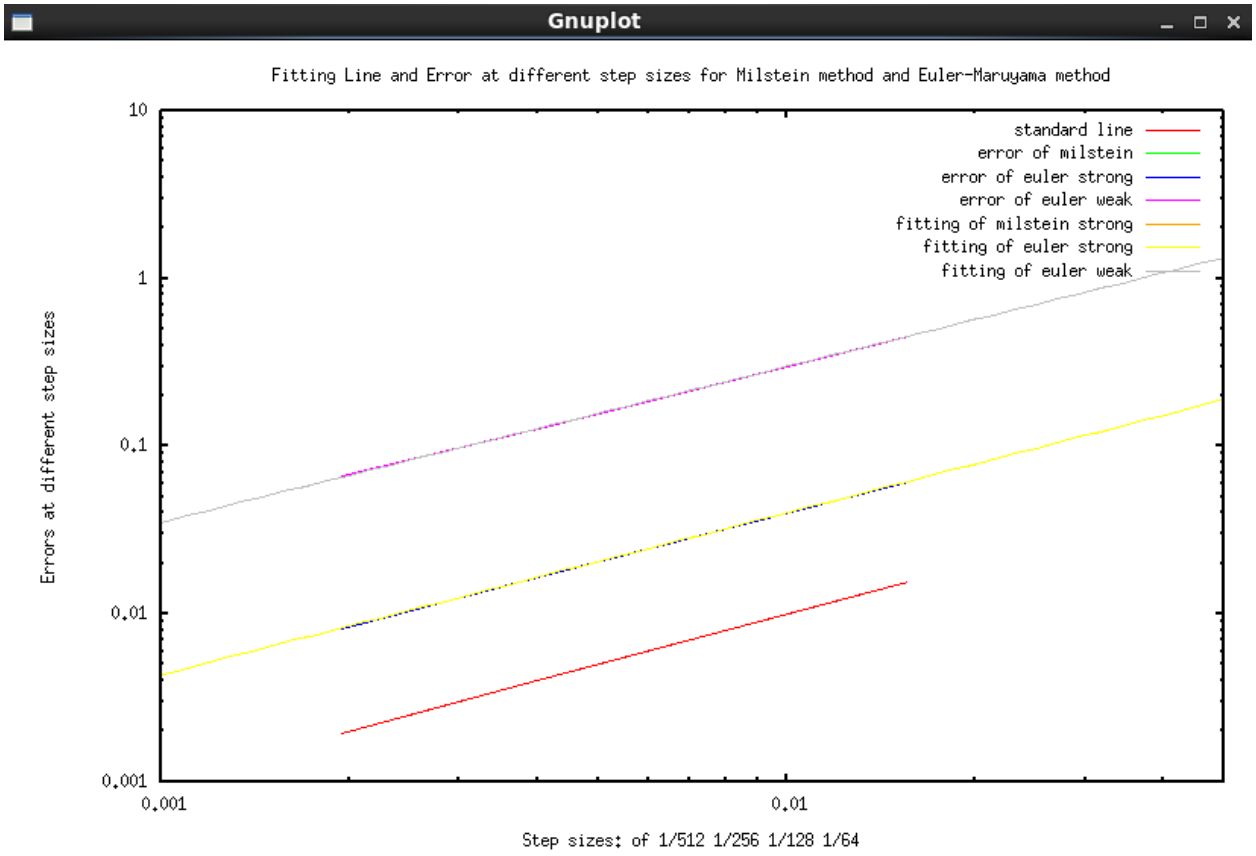
Graph 9



Since in this question the result of milstein simulation is same with euler simulation, the error line of milstein method and strong euler method are coincided together, so is the error bars of milstein method and strong euler method.

The following graph is the fitting graph of this question.

Graph 10



Since the error line of milstein method and strong euler method are coincided together, the fitting lines of milstein method and strong euler method are also coincided. The yellow line above is the fitting line of both milstein method and euler strong method.

The slope of the yellow line is 0.928943 and of grey line is 0.966704, both not far from the expected value 1.

Graph 11

```
 Iteration 7
 WSSR        : 5.02437e-08       delta(WSSR)/WSSR   : -4.77021e-08
 delta(WSSR) : -2.39673e-15      limit for stopping : 1e-05
 lambda   : 1.39891e-08

resultant parameter values

a2              = 3.43808
b2              = 0.966704

After 7 iterations the fit converged.
final sum of squares of residuals : 5.02437e-08
rel. change during last iteration : -4.77021e-08

degrees of freedom    (FIT_NDF)                       : 2
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.000158499
variance of residuals (reduced chisquare) = WSSR/ndf   : 2.51218e-08

Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a2              = 3.43808          +/- 0.07634      (2.22%)
b2              = 0.966704         +/- 0.005042     (0.5215%)
```

Graph 12

```
 Iteration 19
 WSSR        : 1.37445e-06       delta(WSSR)/WSSR   : -5.88658e-06
 delta(WSSR) : -8.0908e-12       limit for stopping : 1e-05
 lambda   : 1.39891e-07

resultant parameter values

a3              = 21.5252
b3              = 0.928943

After 19 iterations the fit converged.
final sum of squares of residuals : 1.37445e-06
rel. change during last iteration : -5.88658e-06

degrees of freedom    (FIT_NDF)                       : 2
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.00082899
variance of residuals (reduced chisquare) = WSSR/ndf   : 6.87224e-07

Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a3              = 21.5252          +/- 0.3283       (1.525%)
b3              = 0.928943         +/- 0.00345      (0.3714%)
```

3.b.ii:

In this part, the difficulties of this SDE given are how could we deal the negative value of X(t) and how to calculate the analytic solution.

One possible method to avoid negative value under sqrt() is define X(t) as complex value.

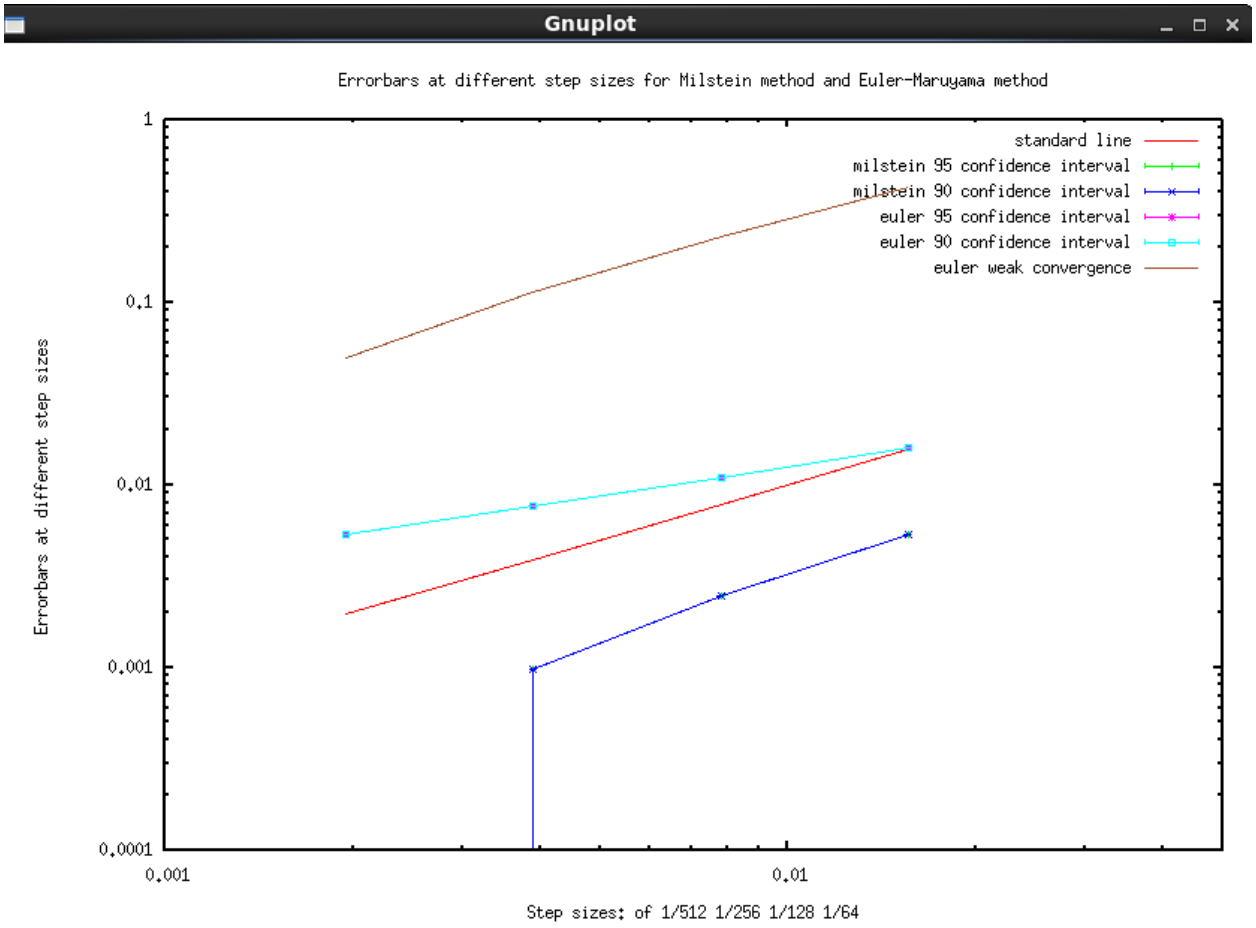
This method will make the code much more complex and one more dimension will be added in.

Another method which is also what I used in this part is using the smallest step size (1/512) milstein numerical solution instead of analytic solution and replacing all negative X(t) values by 0. All these could be found in function2.h.

To run the code, we can come into the directory long/3SDE/part-b/, open the file EandMopp.cc, uncomment line 13, comment line 12 and 13, save and quit file editing. Then we can input $ make test. The graphs below are the running result of $ make test.

The following graph is the error bar graph of this question.

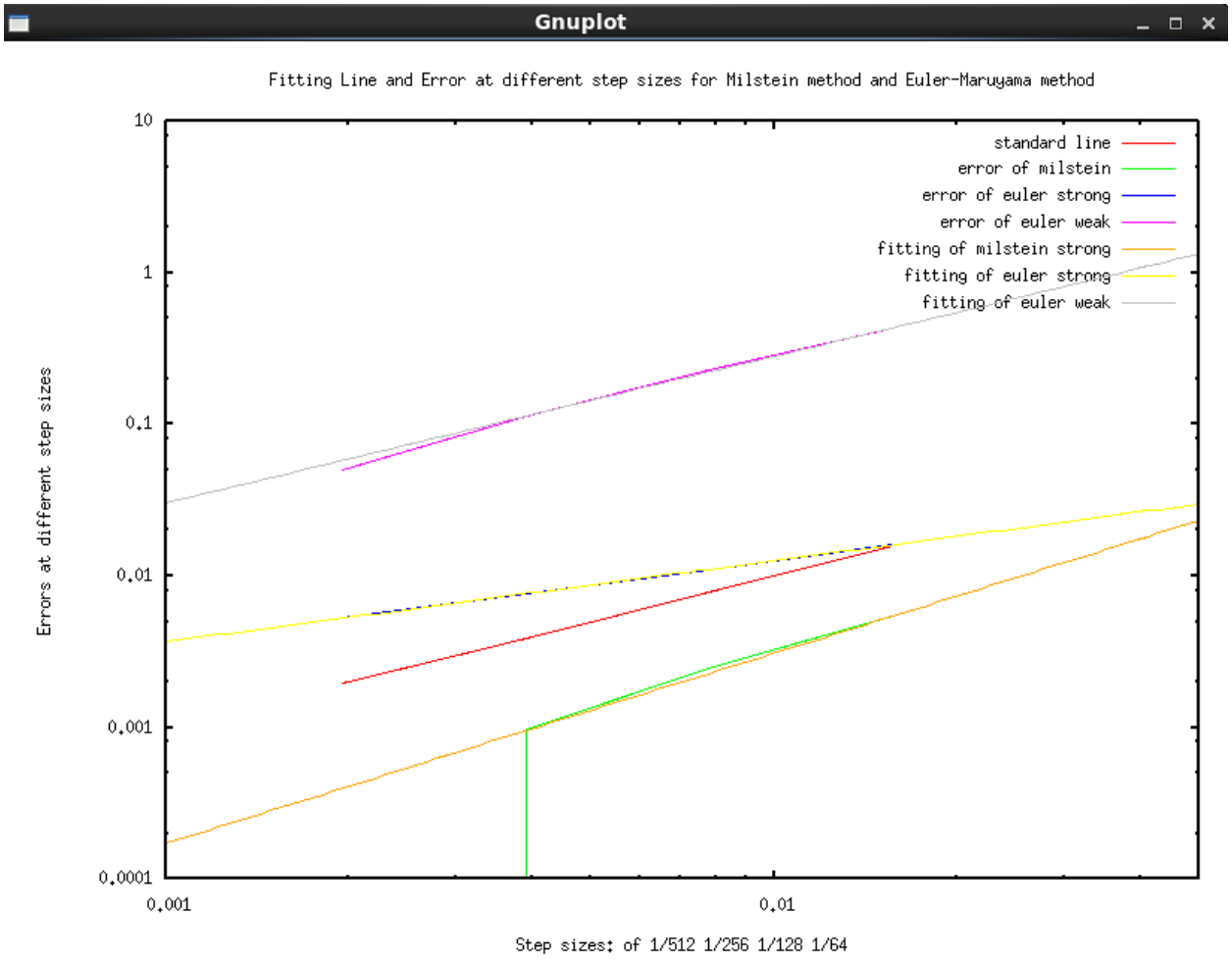
Graph 12



Since in this question I used the milstein numerical solution when dt=1/512 as the actual solution, there is no error shown at dt=1/512 on the blue line.

The following graph is the fitting graph of this question.

Graph 14



Similar with the error bar graph, error is 0 when dt=1/512 for milstein method.

The slope of the orange line is 1.25376 and of grey line is 0.966464, both not far from the expected value 1.

Graph 15

```
 Iteration 26
 WSSR        : 1.97885e-07       delta(WSSR)/WSSR   : -6.32627e-06
 delta(WSSR) : -1.25188e-12      limit for stopping : 1e-05
 lambda   : 8.40892e-08

resultant parameter values

a1              = 0.994955
b1              = 1.25376

After 26 iterations the fit converged.
final sum of squares of residuals : 1.97885e-07
rel. change during last iteration : -6.32627e-06

degrees of freedom    (FIT_NDF)                        : 2
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.000314552
variance of residuals (reduced chisquare) = WSSR/ndf   : 9.89427e-08

Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a1              = 0.994955         +/- 0.6651       (66.85%)
b1              = 1.25376          +/- 0.1553       (12.38%)
```

Graph 16

```
 Iteration 23
 WSSR        : 0.000114105       delta(WSSR)/WSSR   : -1.03913e-06
 delta(WSSR) : -1.18569e-10      limit for stopping : 1e-05
 lambda   : 1.39891e-07

resultant parameter values

a3              = 23.9843
b3              = 0.966464

After 23 iterations the fit converged.
final sum of squares of residuals : 0.000114105
rel. change during last iteration : -1.03913e-06

degrees of freedom    (FIT_NDF)                        : 2
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.0075533
variance of residuals (reduced chisquare) = WSSR/ndf   : 5.70524e-05

Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a3              = 23.9843          +/- 3.633        (15.15%)
b3              = 0.966464         +/- 0.0344       (3.559%)
```

The slope of the yellow line is 0.534063, also not far from the expected value 0.5.

Graph 17

```
 Iteration 51
 WSSR        : 9.47761e-09       delta(WSSR)/WSSR   : -3.39626e-07
 delta(WSSR) : -3.21884e-15      limit for stopping : 1e-05
 lambda   : 1.39891e-06

resultant parameter values

a2              = 0.147301
b2              = 0.534063

After 51 iterations the fit converged.
final sum of squares of residuals : 9.47761e-09
rel. change during last iteration : -3.39626e-07

degrees of freedom    (FIT_NDF)                        : 2
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 6.8839e-05
variance of residuals (reduced chisquare) = WSSR/ndf   : 4.73881e-09

Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a2              = 0.147301         +/- 0.003515     (2.386%)
b2              = 0.534063         +/- 0.005103     (0.9555%)
```

Part 4: Two dimensional SDE

In this part I implemented Euler method to solve the 2-dimensional SDE. Before coding, I studied the Chapter 12.2 Implicit Strong Taylor Schemes of Numerical Solution of Stochastic Differential Equations by Peter E. Kloeden and Eckhard Platen.

$$X_t = P\begin{bmatrix} exp(\rho^{+}(t)) & 0 \\ 0 & exp(\rho^{-}(t)) \end{bmatrix} P^{-1} X_0 \quad (4.1)$$

where

$$\rho^{\pm}(t) = (-a - \tfrac{1}{2}b^2 \pm a)t + bW_t \quad (4.2)$$

and

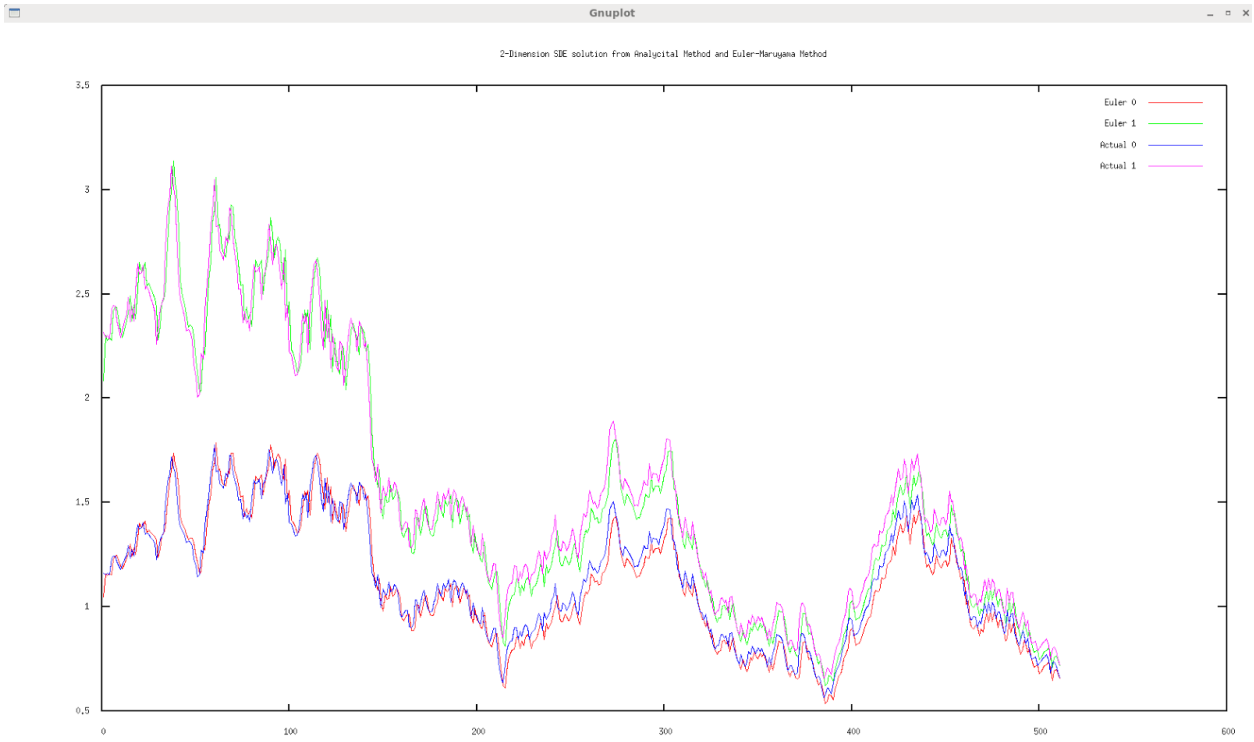
$$P = \tfrac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \; with \; P^{-1} = P \quad (4.3)$$

(page 397 of [NUMERICAL])

The initial matrices are $A = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and I set vector $X_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$. Both actual and Euler solution are stored in file plot.dat and will be plotted by gnuplot later. The a and b in function 4.2 could be found in Matrices A and B. The up-right value of A equals to the value of a and the up-left value of B equals to the value of b.

To run the code, we can come into the directory long/4SDE-2D and input $ make test. The graph below is the running result of $ make test.

Graph 18



In the graph above, red and green lines show the two values of vector Xt by Euler simulation. Blue and purple lines show the actual two values of vector Xt.

We can see that the euler simulation lines fit the actual solution line very well and all four solutions converged at t=1.

Part 5: Heston stochastic volatility model

In this part I implemented both Euler and Milstein method to price in Heston model under the given initial conditions: S(0)=100, V(0)=0.05, r=5%, kappa=1.2, theta=0.04, epsilon=0.3. Both Euler and Milstein method do not need input parameters. If uncomment line 37 and comment line 36, the rho will be set to 0 which means W1 and W2 are not correlated. The monte carlo test number in both methods are 10,000.

One problem of this part is the value of V(t). Value of V(t) may become negative and then be calculated in sqrt(V(t)). In order to avoid negative value under sqrt(), I take absolute value of V(t) at every step.

To run the code, we can come into the directory long/5HESTON and input $ make test. The graph below is the running result of $ make test.

Graph 19

```
[luoq@lonsdale01 5HESTON]$ make test
g++ -Wall euler-heston.cc -O2 -o euler-heston -lgsl -lgslcblas
g++ -Wall milstein-heston.cc -O2 -o milstein-heston -lgsl -lgslcblas
./milstein-heston
This is a Call Option at 100$
Option price is 9.79188
./euler-heston
This is a Call Option at 100$
Option price is 10.1406
```

Part 6: Pricing model

In this part I implemented only Euler method to price the european call option under the given initial conditions: S(0)=1, V(0)=0.01. The code of this part is similar to part 5. The monte carlo test number is 10,000.

Same as part 5, value of V(t) may become negative and then be calculated in sqrt(V(t)). In order to avoid negative value under sqrt(), I made the negative value to be 0 with function max().

To run the code, we can come into the directory long/6PRICING and input $ make test. The graph below is the running result of $ make test.

Graph 20

```
[luoq@lonsdale01 6PRICING]$ make test
g++ -Wall euler-pricing.cc -O2 -o euler-pricing -lgsl -lgslcblas
./euler-pricing
This is a Call Option at 1$
Option price is 0.0496378
```

Reference:

[NUMERICAL]: "Numerical Solution of Stochastic Differential Equations", Peter E. Kloeden and Eckhard Platen, Springer-Verlag Berlin Heidelberg, Second Corrected Printing (1995)