Qijie Li 526 Homework ID: A20355279

1. INTRODUCTION

This homework is to write three functions to implement Monte Carlo Methods for option pricing.

- In question 1, we need to use one step method to calculate European Call option price
- In question 2, we need to use Black-Schole SDE, Euler, Milstein schemes to simulate underlying asset's prices.
- In question 3, we need to write **MCOptionPrices** function to combine what techniques we used in previous tow questions. Then we need to compare convergence properties of each scheme as M and N vary.

2. FUNCTION USED

2.1 Required functions. The assignment specications require the following four functions:

- **BSMonteCarlo(S0, K, r, T, sigma, checkpoints, rateCurve, samples==None).** This function simulates the European Call option price using Monte Carlo Methods in one step.
- **MCStockPrices(S0, sigma, rateCurve, t, samples, integrator).** This function simulates underlying assets' price using multiple steps integrator methods
- **MCOptionPrices(S0, K, rateCurce, sigma, t, checkpoints, samples, integrator).** This function simulates European Call option price using multiple steps integrator methods

2.2. In this homework, we are provided with **rateCurce**. So we need to use linear interpolation method to find interest rate. I set **tenors =[1., 3., 6., 12., 24., 36., 60.]**, then interest rate **r=np.interp(T, tenors, rateCurve)**

- **T** is the maturity time of the European Call option.

3. TEST SUITE

To test whether these function can be called successfully and work correctly. I wrote a test suite below each function using **if __name ==’__main__’** code.

After running each test suite, I got appropriate results.

4. VALUE INPUTS

To test three functions. In each functions, I assign **S0=50, sigma=0.1, T=10(monthly), rateCurve=[0.0017, 0.03, 0.0049, 0.0061, 0.0079,0.0093, 0.012]**

The rateCurves are obtained from daily treasury yield curve rates in 09/ 20/ 2016.

(1) Question 1:

- **K=55, checkpoints=[1000, 2000, 5000, 10,000, 20,000, 50,000, 100,000, 200,000,500,000, 1,000,000], M=1,000,000, N=1**
    - Samples are generated by the code: **samples=randn(N,M)**

(2) Question 2:

- **M=10000, N=365, samples= rand(N,M)**
    - For the purpose of quick function test, I set M a not too big number.
- **t=array[T/N]*N, T= sum(t)**
    - Since **t** is a list of dt, to find interest rate, we need to calculate the maturity time **T** to find interest rate using linear interpolation

(3) Question 3:

1) Function test

- **checkpoints=[1000, 2000, 5000, 10,000, 20,000, 50,000], M=checkpoints[-1], N=100**
    - Set M, and N not too big number for the purpose of quick function test.

- **t=array[T/N]*N, T= sum(t)**
    - Since **t** is a list of dt, to find interest rate, we need to calculate the maturity time **T** to find interest rate using linear interpolation

- **samples= rand(N,M)**

2) Convergence properties test

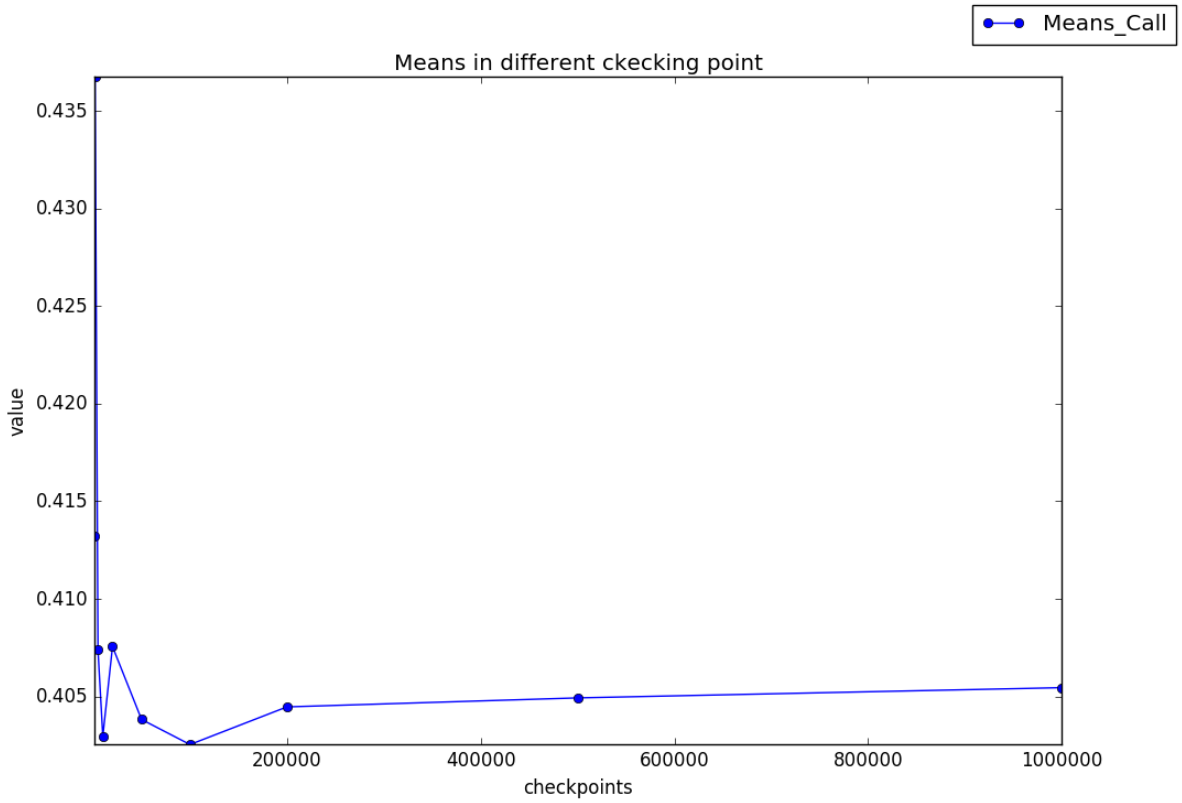
- **M=range(100,5,010,100), N=range(1,101,1)**
  - In the convergence properties test, **M** and **N** are list so that I can test the European option price in dynamic **M** and **N**. Besides, other input value are as same as Function test
  - In the code, I reset **M=range(100,2,010,100)** for the purpose quick code check.

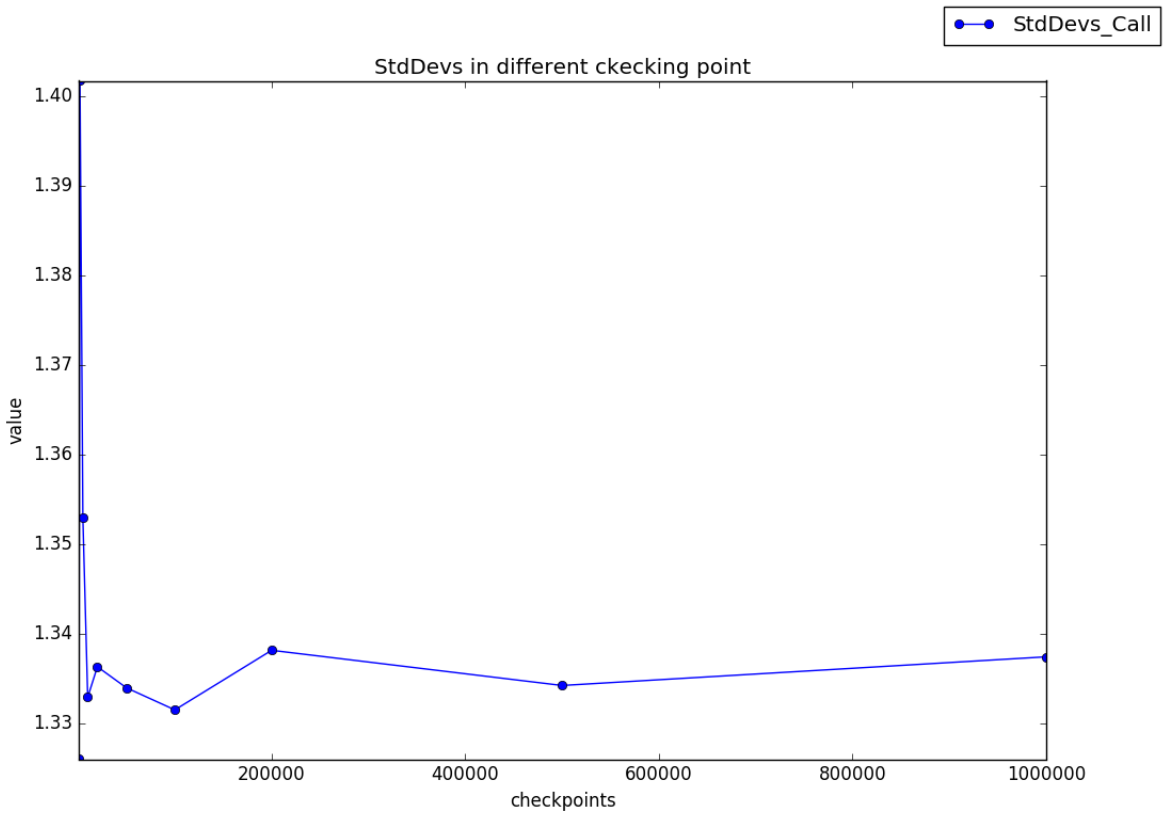
5. PRPGRAM RESULT

Below are the result from my codes

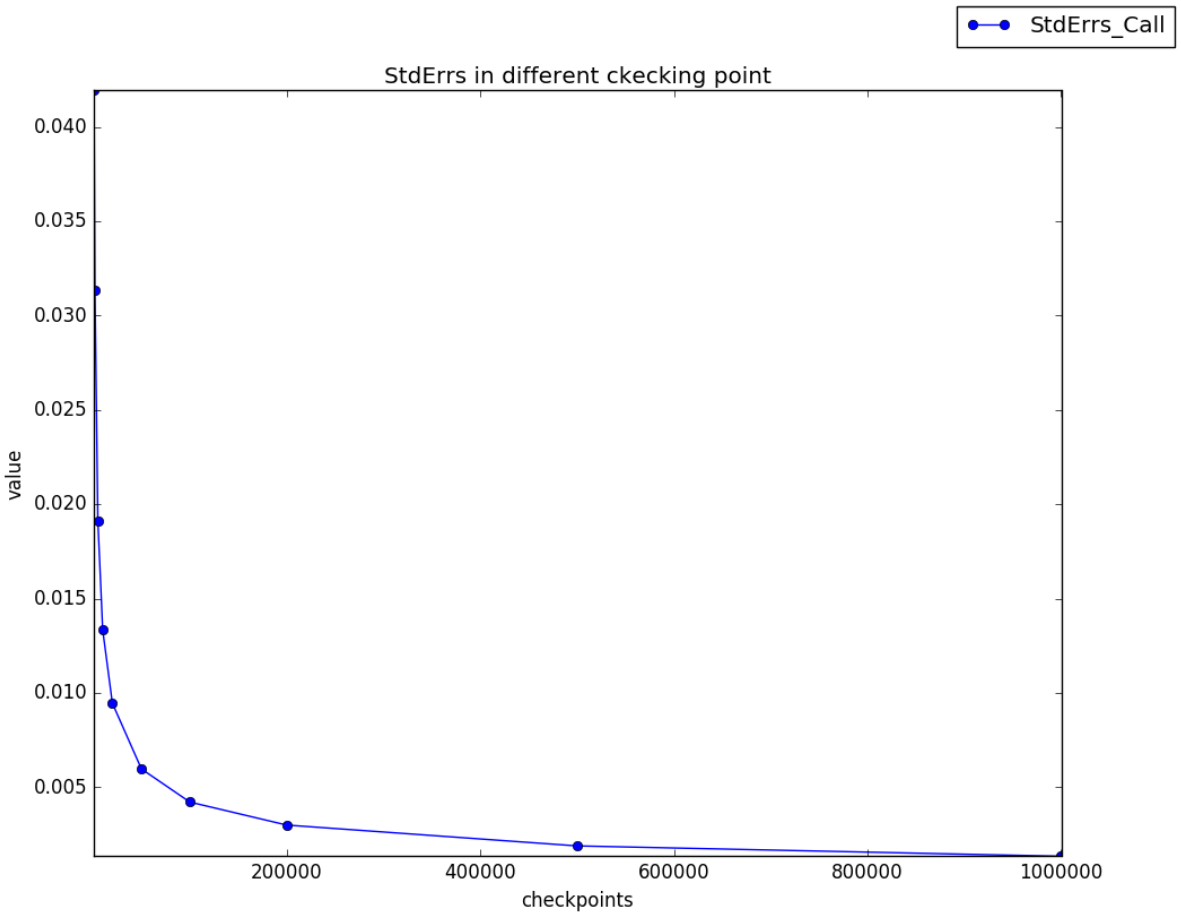
(1) Question 1:

('Call option TV is 0.405', 'Put option TV is5.144')




StdDevs_Call
StdDevs in different ckecking point
1.40
1.39
1.38
1.37
1.36
1.35
1.34
1.33
value
200000
400000
600000
800000
1000000
checkpoints

G-2



G-3

(2) Question 2:

(array([[50.31610454, 49.99089004, 50.29082313, ..., 49.97966882,

49.99208693, 49.57485835],

[50.28816257, 49.95387088, 50.25448587, ..., 50.12583325,

50.04078896, 49.80393928],

[50.16872986, 50.17904107, 50.39891917, ..., 50.29820259,

49.97699146, 49.62350686],

...,

[51.10487882, 63.42871353, 54.46288453, ..., 52.34185376,

```
        44.44934853,  50.84123908],
       [ 51.25100136,  63.46543245,  54.65724121, ...,  51.97513475,
        44.01319511,  50.4661104 ],
       [ 51.41104234,  63.90704311,  54.30957943, ...,  52.15444448,
        43.78252704,  50.4201274 ]]), (365L, 10000L))
(array([[ 50.31568029,  49.99145999,  50.29055139, ...,  49.98023546,
        49.99265708,  49.57361136],
       [ 50.28830517,  49.95499737,  50.25477528, ...,  50.12675878,
        50.04190665,  49.8027248 ],
       [ 50.16930414,  50.18023692,  50.39957595, ...,  50.29940784,
        49.97863828,  49.62253767],
       ...,
       [ 51.08239573,  63.39280425,  54.47764758, ...,  52.3332202 ,
        44.47882272,  50.86214514],
       [ 51.22882871,  63.43021543,  54.67233277, ...,  51.96586862,
        44.04073246,  50.48605147],
       [ 51.389136  ,  63.87077664,  54.3240884 , ...,  52.14543106,
        43.80981587,  50.44060565]]), (365L, 10000L))
(array([[ 50.31568029,  49.99145999,  50.29055139, ...,  49.98023546,
        49.99265708,  49.57361136],
       [ 50.28830517,  49.95499737,  50.25477528, ...,  50.12675878,
        50.04190665,  49.8027248 ],
       [ 50.16930414,  50.18023692,  50.39957595, ...,  50.29940784,
```

49.97863828, 49.62253767],

...,

[51.08239573, 63.39280425, 54.47764758, ..., 52.3332202 ,

44.47882272, 50.86214514],

[51.22882871, 63.43021543, 54.67233277, ..., 51.96586862,

44.04073246, 50.48605147],

[51.389136 , 63.87077664, 54.3240884 , ..., 52.14543106,

43.80981587, 50.44060565]]), (365L, 10000L))

(3) Question 3:

1) Function test

Using standard method, Call option TV is 0.40981

('Means are', array([[0.3877856 , 5.14583998],

[0.4431912 , 5.04791751],

[0.42259522, 5.05246346],

[0.41784352, 5.07361865],

[0.40835352, 5.1037282],

[0.40980575, 5.12143199]]))

Using euler method, Call option TV is 0.40910

('Means are', array([[0.38733163, 5.14579632],

[0.44207327, 5.04824683],

[0.4216207 , 5.05273656],

[0.4171414 , 5.07323865],

[0.40760978, 5.10329668],

[0.4091027 , 5.12095782]]))

Using milstein method, Call option TV is 0.40975

('Means are', array([[0.38773598, 5.14578432],

[0.44313346, 5.04786739],

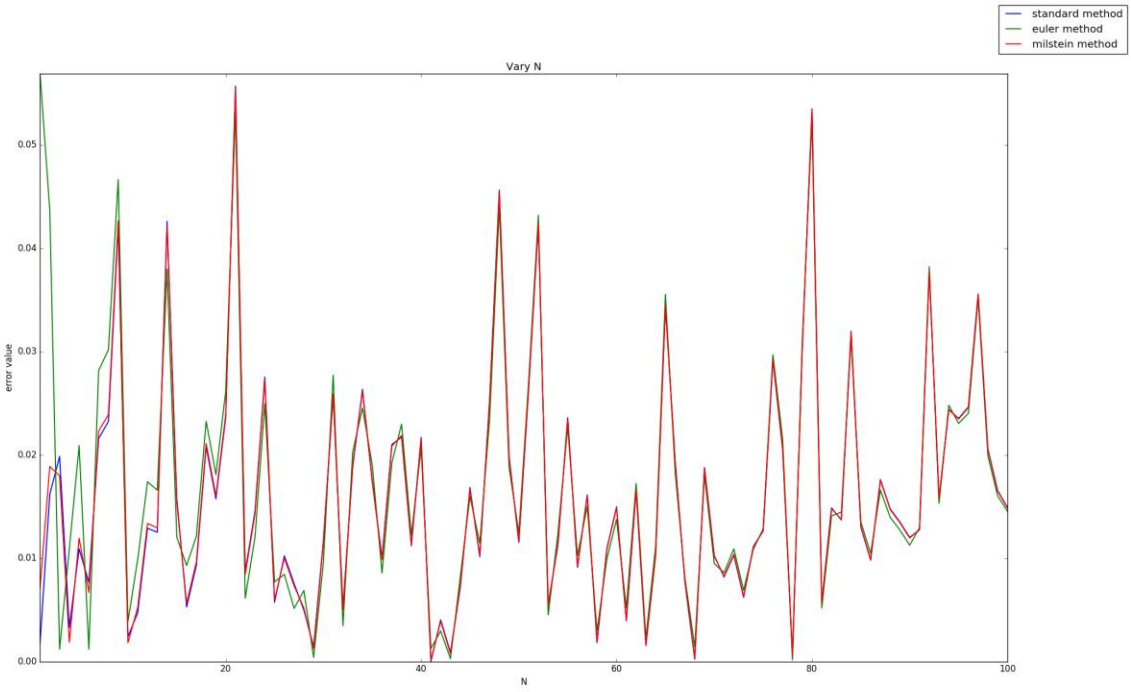
[0.42253694, 5.05241551],

[0.41778624, 5.07357296],

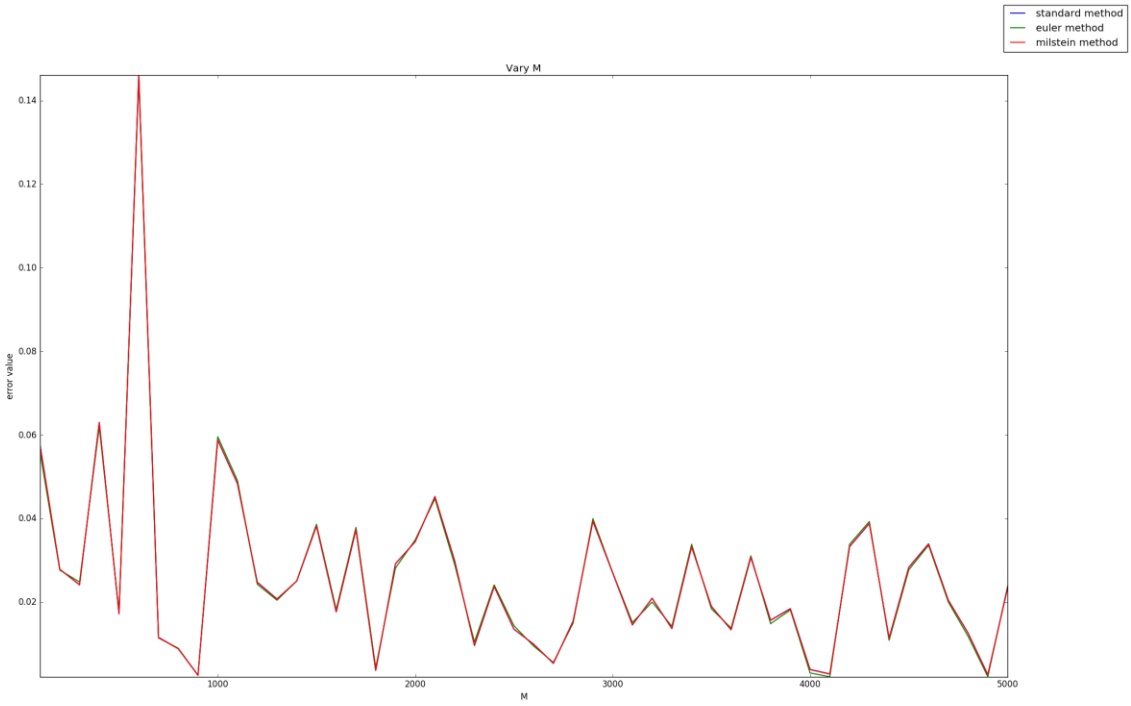
[0.40829667, 5.10368067],

[0.40974872, 5.12138288]]))

2) Convergence properties test



G-4



G-5

6. ANALYSIS OF RESULT

(1) Question 1:

The simulation result is close the theory result. What's more , with the number of paths increase, Means and StdDevs are more stable, and StdErrs decrease gradually.

(2) Question 2:

As the results shown above, the function works properly.

(3) Question 3:

1) Function test

   As the results shown above, the function works properly in different integrator schemes.

2) Convergence properties test

   For three different integrator schemes, when the path increase, error is decreasing, and when the number of iteration increase, error show unpredictable movement.

7. CONCLUSION

- Generally, the simulation results of Monte Carlo Methods are close to theory result. In some degree, we can use it as the approximation answer. But this will produce error.
- When the simulation number increase, the error will decrease.