Home Work 4

Author: Yuxuan Xia

Tool: CUDA 8.0

Condition

rate = 0.02

volatility = 0.05

stockPrice = 20

strickPrice = 20

timeToMature = 1 (year)

n_iters = 1000 (number of steps)

n_trials = 100000 (number of trials)

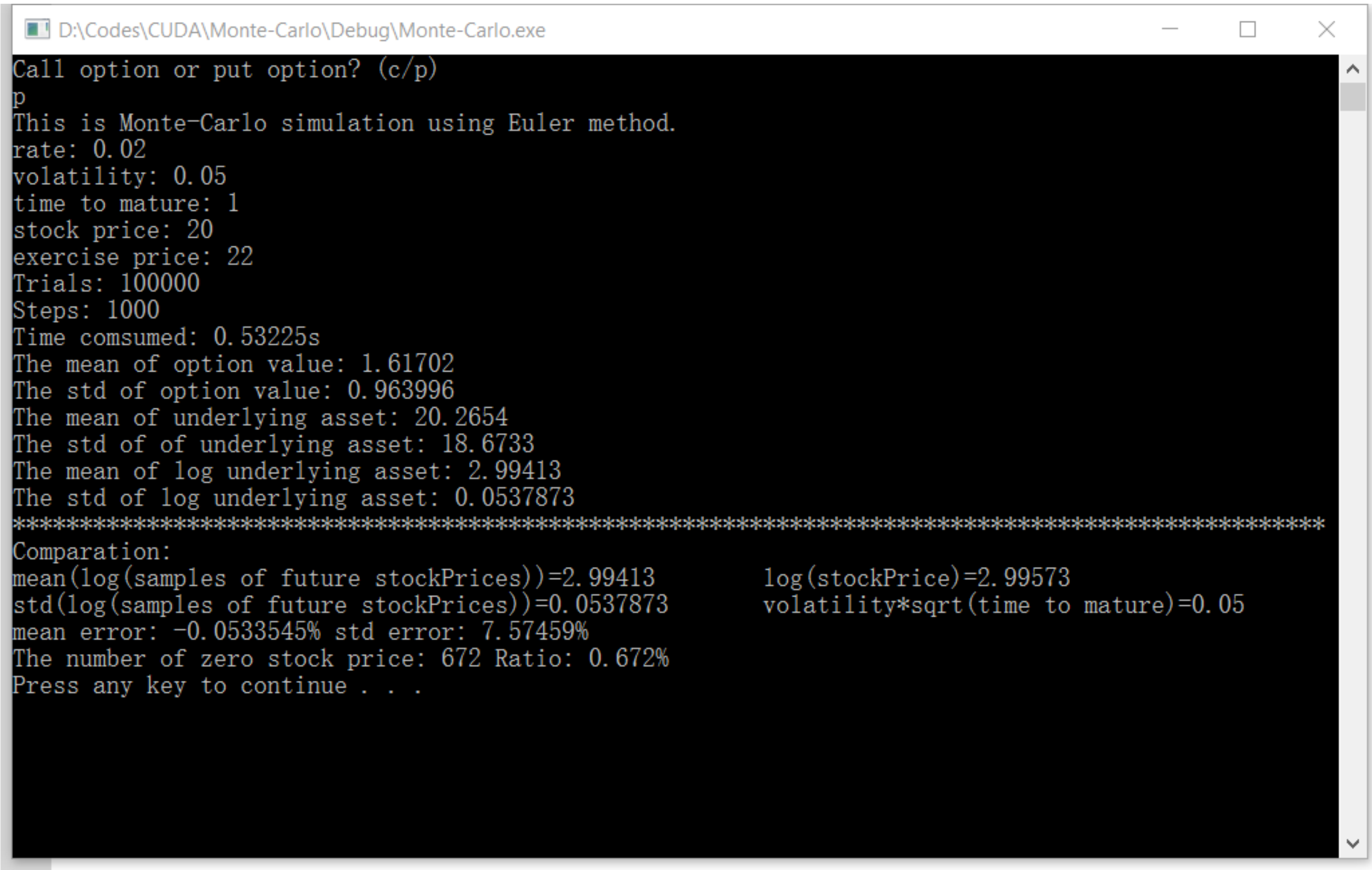
Result

Call option

D:\Codes\CUDA\Monte-Carlo\Debug\Monte-Carlo.exe

```
Call option or put option? (c/p)
c
This is Monte-Carlo simulation using Euler method.
rate: 0.02
volatility: 0.05
time to mature: 1
stock price: 20
exercise price: 22
Trials: 100000
Steps: 1000
Time comsumed: 0.531237s
The mean of option value: 0.0303226
The std of option value: 0.160685
The mean of underlying asset: 20.2706
The std of of underlying asset: 20.2379
The mean of log underlying asset: 2.99439
The std of log underlying asset: 0.0537101
****************************************************************************************************
Comparation:
mean(log(samples of future stockPrices))=2.99439        log(stockPrice)=2.99573
std(log(samples of future stockPrices))=0.0537101       volatility*sqrt(time to mature)=0.05
mean error: -0.0447274% std error: 7.42018%
The number of zero stock price: 672 Ratio: 0.672%
Press any key to continue . . .
```

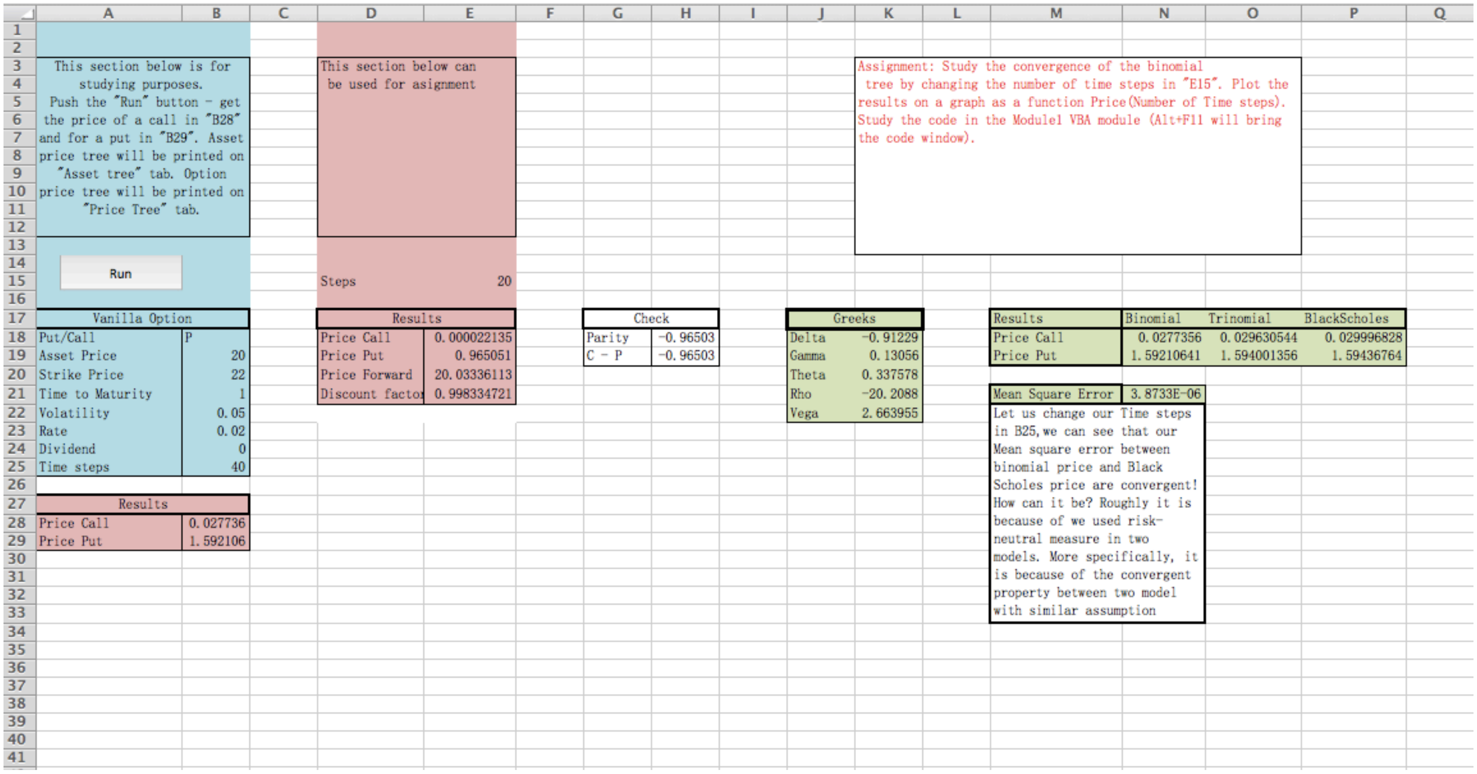
Put option



D:\Codes\CUDA\Monte-Carlo\Debug\Monte-Carlo.exe

```
Call option or put option? (c/p)
p
This is Monte-Carlo simulation using Euler method.
rate: 0.02
volatility: 0.05
time to mature: 1
stock price: 20
exercise price: 22
Trials: 100000
Steps: 1000
Time comsumed: 0.53225s
The mean of option value: 1.61702
The std of option value: 0.963996
The mean of underlying asset: 20.2654
The std of of underlying asset: 18.6733
The mean of log underlying asset: 2.99413
The std of log underlying asset: 0.0537873
*************************************************************************************************
Comparation:
mean(log(samples of future stockPrices))=2.99413        log(stockPrice)=2.99573
std(log(samples of future stockPrices))=0.0537873       volatility*sqrt(time to mature)=0.05
mean error: -0.0533545% std error: 7.57459%
The number of zero stock price: 672 Ratio: 0.672%
Press any key to continue . . .
```

Black Scholes and Binomial tree



Analysis

Environment

Language: CUDA 8.0 in C++

IDE: Visual Studio 2015

Operation System: Windows 10

Efficiency

Simulations: 1e5 trials * 1e3 steps

Time comsumed: ~0.5s

One can see the super power of GPU multi-threads programming!

Stock price simulation

1. Geometic Brownian Motion Property: log-normal

   Using Euler discretization in order to simulate geometric brownian motion.

In result, one can see that the mean of log(samples of future stock price) is approximate equial to log(stock price today) which means log(stock price) it is a martingale.

Moreover the std of log(samples of future stock price) is approximate equal to volatility × sqrt(time to mature).

In conclusion, the future stock price simulated by Euler discretization method agree with geometric brownian motion whose distribution of a fixed t is log-normaled.

2. Trueness Error & Stopped Process

   However, when I start to calculate the statics property of log(samples of future stock prices) I find that there are some samples whose value equal to zero, which is undefined in log function. Fortunately, the ratio of these bad values is about only 0.67%. So I ignore these points to obtain the previous results.

   To be precise, the bad value result from the computer's trueness error. As the exponential of any finite number is bigger than 0, one can never obtain 0 stock price in Euler method theoretically. However, if the stock price at any time equal to 0, it will always be 0 as stockPrice *= exp(Euler method here).

   In conclusion, the stock price is not a pure geometric Brownian motion. It is a stopped process. And trueness error combined with stopping time account for the error between Black Scholes and Euler Discretization method.

option value simulation

European option value = max(0,future stockprice - strike price)

All statical property or errors can be gathered by future stock price. It's neither log normal nor normal. Then, not repeat here.