CISC 5352 Financial Data Analytics Quiz (2) [1]

[1]Please turn in your workable codes and corresponding running results.

Problem set (1) Hart's algorithm to compute cumulative standard normal distribution (10 points)

The cumulative standard normal distribution

$$N(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-u^2/2} du$$

has no an analytic solution. It is widely used in option pricing. The classic method to conduct numerical integration is called Hart algorithm invented in 1968.

- Hart gave the following approximation for this cumulative distribution function (CDF) $N(x)$

$$N(x) = \left\{ \begin{array}{c} e^{-t^2/2}\frac{A}{B}, \text{when } t < t_1 \\ e^{-t^2/2}\frac{1}{t_3 C}\text{when } t_1 \le t \le t_2 \\ 0, when t > t_2 \end{array} \right)$$

where $t = |x|$ and $t_1 = 7.07106781186547$, $t_2 = 37$, $t_3 = 2.506628274631$

- We only need to consider when $x < 0$ for convenience of computing. When $x > 0$, we use basic property of a normal distribution and get $N(x) = 1 - N(x)$

  A = $((((((a_1t+a_2)t+a_3)t+a_4)t+a_5)t+a_6)t+a_7)$[2]
  B = $(((((((b_1t+b_2)t+b_3)t + b_4)t + b_5)t + b_6)t + b_7)t + b_8)$
  C = t+1/(t + 2/(t + 3/(t + 4/(t + 0.65))))

  a and b are the following arrays

[2]Such addition format is good for computer to implement!

a_1 = 0.0352624965998911
a_2 = 0.700383064443688
a_3 = 6.37396220353165
a_4 = 33.912866078383
a_5 = 112.079291497871
a_6 = 221.213596169931
a_7 = 220.206867912376

b_1 = 0.0883883476483184
b_2 = 1.75566716318264
b_3 = 16.064177579207
b_4 = 86.7807322029461
b_5 = 296.564248779674
b_6 = 637.333633378831
b_7 = 793.826512519948
b_8 = 440.413735824752

Write a python function with the following signature and a corresponding main method to test your function.

```
caculateStandardNormalCDF(x)
```

- You need to verify your method *as least* to calculate, $N(-0.02)$, N(0.02), N(-0.8), N(0.182)
- Compare your results with stats.norm.cdf( ) for these inputs

Problem set (2) Polish Monte Carlo simulations (25 points)

1. Write a function with the following signature such that it can handle both call and put options.

```
mc_pricing(S, K, T, r, sigma, option_type, no_trial)
```

- You are required to use the following call and examples to test your function
  - An European call with continuous dividend yield: $S = 50$,$K = 80$, $r = 0.1$,$T = 5/12$, $\sigma = 0.35$
  - An European put option on stock indexes with a cost-of-carry: $S = 80$,$K = 75$, $r = 0.1$,$T = 5/12$, $\sigma = 0.20$
  - The no_trial: should be at least 10^7
  - Plot your simulation results for the first 5 trials (You can 'relax' the huge trial number a little bit for the sake of visualization)

Python multithreading programming for Monte Carlo simulations

- Multithreading programming is an important skill in data analytics.
- Go through the following sample multithreading python demo codes to understand multithreading programming basics.
- Convert your Monte Carlo simulation results in to a multithreading version

```
################################################################
## demoThread2.py
## This is a sample file for CISC5352: Financial Data analytics
## Created by Henry Han
## Last update: 09/15/2016
#################################################################
```

```
import threading
import time

def addHarmonicSeries(n):
sum=0.0
for i in range(1,n):

     sum=sum + 1.0/i
     print('{:5d} {:12.6f}'.format(i, sum))

start_time = time.clock()
no_thread = 100
for i in range(no_thread):

     t = threading.Thread(target=addHarmonicSeries, args=(i,))
     t.start()
     print("\n-->"+t.getName() + "\n")
     time.sleep(1)


print("\n The main stread stops after {}".format(time.clock()-start_time)\
+ " seconds")
```

Problem set (3) 'Polish' Black-Scholes (15 points)

1. Write a function with the following signature such that it can handle both call and put options.

```
bs_pricing(S, K, T, r, sigma, option_type)
```

- You are required to use the put call examples in lecture note (2) to test your function.
- Your output should format output

2. Write a function with the following signature such that it can handle both call and put options for BSM model

```
bsm_pricing(S, K, T, r, sigma, option_type)
```

- You are required to use the following call and examples to test your function
  - An European call with continuous dividend yield: $S = 50$,$K = 80$, $r = 0.1$,$T = 5/12$, $\sigma = 0.35$, $q = 0.05$
  - An European put option on stock indexes with a cost-of-carry: $S = 80$,$K = 75$, $r = 0.1$,$T = 5/12$, $\sigma = 0.20$, $q = 0.07$