Lab3_sol

September 20, 2017

1 Lab 3 sample solutions

First we should import libraries what we need Allways we import scipy

```
In [20]: import numpy as np
```

In this lab we also need to draw graphs, so let us import plot and show from pylab

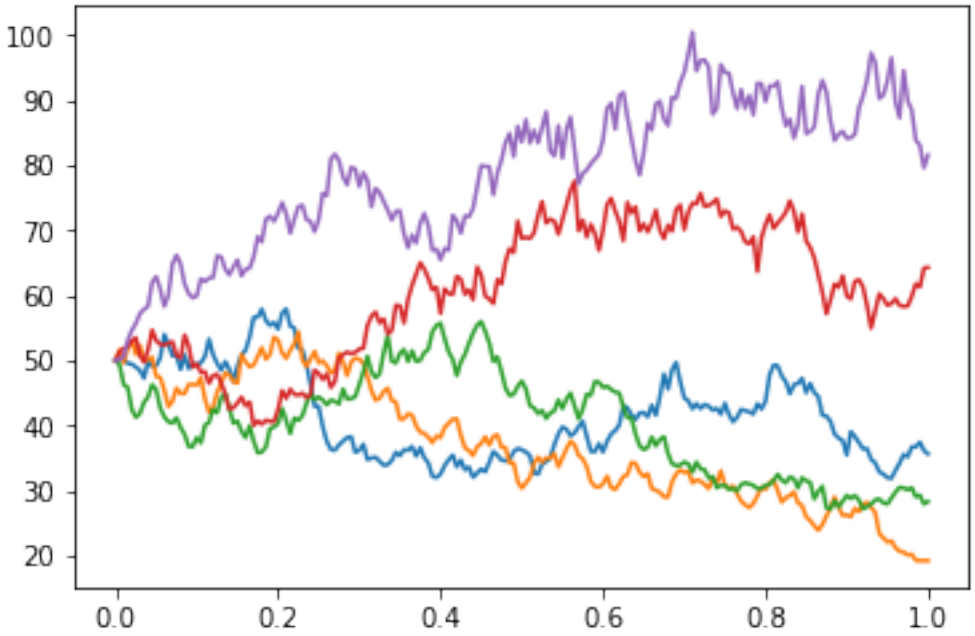
```
In [21]: import pylab as pl
```

1.1 Exercise 1

Write a function `BSgraph(S0,n,m,mu,sigma,T)` that plots the graph of n trajectories of the stock price on the interval $[0,T]$, corresponding to the Black-Scholes market model with constant parameters μ and σ. For computing the values of the stock prices divide the interval $[0,T]$ into m equal subintervals (ie. use the time points $t_i = \frac{i \cdot T}{m}$, $i = 0,1,\ldots,m$) and use the Euler's method.

Solution

```
In [22]: def BSgraph(S0,n,m,mu,sigma,T):
             hi=T/m
             S=np.zeros(shape=(m+1,n))
             S[0,:]=S0
             for i in np.arange(1,m+1):
                 Xi=np.random.randn(n)
                 S[i,:]=S[i-1,:]*(1+mu*hi+sigma*np.sqrt(hi)*Xi)
             t=np.linspace(0,T,m+1)
             pl.plot(t,S)
             pl.show()
         BSgraph(S0=50,n=5,m=200,mu=0.1,sigma=0.5,T=1)
```



1.2 Exercise 2

In the case of pricing European options by Monte-Carlo methods, we do not want to look at the trajectories of the stock prices but need only to generate values of $S(T)$. Define a function `ST(S0,n,m,mu,sigma,T)` that returns a vector of n randomly generated values of $S(T)$ according to BS market model with constant μ and non-constant volatility given by a function $\sigma(s,t)$. Compute the mean value and standard deviation of 100000 generated stock prices at time $t = T$ in the case $\mu = 0.05$, $m = 100$, $S_0 = 10$, $T = 1$ and $\sigma(s,t) = \frac{e^{-0.1\cdot t}}{1+0.005s^2}$. (For checking answers: mean and standard deviation should be approximately 10.5 and 6.1)

Solution

```
In [23]: def ST(S0,n,m,mu,sigma,T):
             """
             generator of final stock prices S(T) in the case of BS market model
             sigma should be a function of two variables
             """
             hi=T/m
             S=np.zeros(shape=(m+1,n))
             S[0,:]=S0
             t=np.linspace(0,T,m+1)
             for i in range(1,m+1):
                 Xi=np.random.randn(n)
                 S[i,:]=S[i-1,:]*(1+mu*hi+sigma(S[i-1,:],t[i-1])*np.sqrt(hi)*Xi)
             return S[m,:]
```

```
        def sigma_ex2(s,t):
            return np.exp(-0.1*t)/(1+0.005*s**2)
        result=ST(n=100000,S0=10,mu=0.05,m=100,T=1,sigma=sigma_ex2)
        print("mean:",np.mean(result))
        print("standard deviation",np.std(result))
```

```
mean: 10.505523167
standard deviation 6.07615963614
```

1.3 Exercise 3

Often we have several stochastic processes in a market model. Let us consider a model with stochastic interest rate:

$$dS(t) = S(t)(r(t)\,dt + 0.5dB_1(t)),$$
$$dr(t) = (0.05 - r(t))\,dt + 0.02\,dB_2(t),$$

where B_1 and B_2 are independent Brownian motions. Use Euler-Maruyama method for defining a function that for given n and m outputs n generated values of $S(0.5)$ in the case $S(0) = 100$, $r(0) = 0.04$.

Solution

```
In [24]: def Ex3(m,n):
             #parameters fixed in the problem
             T=0.5
             S0=100
             r0=0.04
             delta_t=T/m
             S=np.zeros(shape=(m+1,n))
             r=np.zeros(shape=(m+1,n))
             S[0,:]=S0
             r[0,:]=r0
             for i in range(1,m+1):
                 dB1=np.sqrt(delta_t)*np.random.randn(n)
                 dB2=np.sqrt(delta_t)*np.random.randn(n) #each time we use randn, new (indepen
                 S[i,:]=S[i-1,:]*(1+r[i-1,:]*delta_t+0.5*dB1)
                 r[i,:]=(0.05-r[i-1,:])*delta_t+0.02*dB2+r[i-1,:]
             #need to return only values for stock prices at T=0.5
             return S[m,]
         #try if the function works (each time we use the generator, we get different numbers)
         print(Ex3(10,5))
         print(Ex3(10,5))
         #the average of a large number of generated values is an approximation
         #of the expected value of S(T) for reasonably large values of m
         ST_values=np.mean(Ex3(100,100000))
         print("average of 100000 values for m=100:",ST_values)
```

```
[  91.28955645   89.699855     91.10688581  105.73033266  125.26055018]
[ 134.87974335   98.14246629  127.20016126  115.88339273  144.073344  ]
average of 100000 values for m=100: 101.989826068
```