Jacobs University November 11, 2019
Fall 2019

Stochastic Methods + Lab

Assignment Sheet 9

Due on November 18, 2019

Problem 1 [**7 points**]
Let us consider the exponential Ornstein-Uhlenbeck stochastic process

$$dX_t = \mu(1 - c \ln X_t) X_t dt + \sigma X_t dW_t, \tag{1}$$

with $\mu, \sigma > 0$ and $0 \leq c \leq 1$.

(a) Generate an ensemble of paths of the process (1) on the interval $[0, 1]$, and plot the empirical mean and standard deviation together with 10 sample paths. Produce at least two plots with reasonable parameters such that the influence of the parameter c as compared to geometric Brownian motion becomes visible. Briefly describe what happens when c is near zero or c is near 1.

(b) Now assume the model (1) for the stock price (instead of geometric Brownian motion as we did in class). Use Ito's lemma and "Merton's trick" to derive a deterministic equation for the option price (similar to the Black-Scholes equation we derived in class). The result might seem surprising; briefly explain why it isn't.

Problem 2 [**6 points**]
Modify your binomial tree algorithm from one of the previous assignment sheets to price also American options (i.e., the holder may exercise the option at any time before expiration). Write a summarizing program with two functions

`binomial_american(payoff,n,r,sigma,S,K,T)`

and

`binomial_european(payoff,n,r,sigma,S,K,T)`

and plot the option price for different strike prices for American and European calls and puts (i.e., four graphs in the same coordinate system) with some reasonable parameters. Is the price of an American put higher or lower than that of a European put with otherwise identical parameters?

Problem 3 [**7 points**]
Rewrite your binomial tree function `binomial_tree(payoff,n,r,sigma,S,K,T)` from one of the previous assignment sheets so that it stores the option value at each node of the tree. Then visualize the tree using `imshow` for some reasonable parameters. Here, you have to think about an appropriate color map, how to mask the missing values (hint: use Numpy's masked arrays), and how to best map the computed values to pixel coordinates.