Name: Abderrazak DERDOURI
Object: PYTHON 300B: System Development with Python
Date: 2016/08/25

Part 1

We will consider the problem for the price of a digital call option $V(S;t)$. The resulting PDE at expiry T satisfied by V (S; t) is:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S}dS - rV = 0 \tag{1}$$

and the payoff of this option is:

$$\begin{aligned} V(S,T) &= H(S_T - E) \\ &= \begin{cases} 1\, S_T > E \\ 0\, S_T <= E \end{cases} \end{aligned}$$

where $H(.)$ is the heaviside function. S is the spot price of the underlying financial asset, t is the time, $E > 0$ is the strike price, T the expiry date, r the interest rate and σ is the volatility of S.
(1) has a closed form solution:

$$V(S,t) = e^{-r(T-t)}N(d_2)$$

where

$$d_2 = \frac{\log(\frac{S}{k}) + (r - \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{(T-t)}}$$

As conducted on the lecture "Further Numerical Methods pages 37-47", we will solve (1) by the Explicit Finite Difference method using a backward marching scheme, i.e. of the form:

$$V_n^m = F(V_{n-1}^m, V_n^m, V_{n+1}^m)$$

By expressing $S = n\delta S$ and $t = m\delta t$; we obtain a difference equation for the Black-Scholes equation.

$$V(S+\delta S, t+\delta t) = V(S,t) + \frac{\partial V}{\partial S}\delta S + \frac{\partial V}{\partial t}\delta t + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\delta S^2 + \frac{1}{2}\frac{\partial^2 V}{\partial t^2}\delta t^2 + \frac{\partial^2 V}{\partial S \partial t}\delta S \delta t + O(\delta S^3, \delta t^3)$$

Consider

$$V(S; t+\delta t) = V(S;t) + \frac{\partial V}{\partial t}\delta t + O(\delta S^2)$$

and rearranging gives a forward difference:

$$\frac{\partial V}{\partial t} = \frac{V(S;t+\delta t) - V(S;t)}{\delta t}$$

If we use a backward time difference:

$$\frac{\partial V}{\partial t} = \frac{V(S;t) - V(S;t-\delta t)}{\delta t}$$

which becomes, in finite difference form:

$$\frac{\partial V}{\partial t}(n\delta S, m\delta t) \sim \frac{V_n^m - V_n^{m-1}}{\delta t}$$

$$\frac{\partial V}{\partial S}(n\delta S, m\delta t) \sim \frac{V_{n+1}^m - V_{n-1}^m}{2\delta S}$$

$$\begin{aligned} V_n^{m-1} &= V_n^m + \delta t\left[\frac{1}{2}n^2\sigma^2(V_{n-1}^m - 2V_n^m + V_{n+1}^m)\right] \\ &+ \delta t\left[\frac{1}{2}rnV_{n+1}^m - V_{n-1}^m - rV_n^m\right] \\ &= \alpha_n V_{n-1}^m + \beta_n V_n^m + \gamma_n V_{n+1}^m \end{aligned}$$

where

$$\begin{aligned} \alpha_n &= \frac{1}{2}[n^2\sigma^2 - nr]\delta t \\ \beta_n &= [1 - (r + n^2\sigma^2]\delta t \\ \gamma_n &= \frac{1}{2}[n^2\sigma^2 + nr]\delta t \end{aligned}$$

Boundary conditions:
At S=0, we have: $V_0^{m-1} = [1 - r\delta t]V_0^m$.
When S becomes very large, we have: $V_N^{m-1} = [\alpha_N - \gamma_N]V_{N-1}^m + [\beta_N + 2\gamma_N]V_N^m$.
Python code: I coded the Explicit Finite Difference Method using a backward scheme for a binary call. The following table shows a binary call price using the backward scheme Explicit Finite Method. Stock price ranges from 60 to 80, expiration from 0.2 to 1. The number of asset step was fixed to 80. Columns "ErrWithBS" and "%ErrWithBS" show Error with the closed Black-Scholes formula.
When considering increase of the number of asset step, I observe that the error with the closed formula decrease especially for short expirations.

S	E	r	Sigma	T	NAS	BS Price	FDM Price	ErrWithBS	ErrWithBSPer
60	100	0.05	0.2	0.2	80	8.24E-09	9.46E-09	-1.22E-09	14.77%
60	100	0.05	0.2	0.4	80	3.94E-05	3.85E-05	8.30E-07	2.11%
60	100	0.05	0.2	0.6	80	0.000711781	0.000694424	1.74E-05	2.44%
60	100	0.05	0.2	0.8	80	0.00312255	0.00305759	6.50E-05	2.08%
60	100	0.05	0.2	1	80	0.00771023	0.00757512	0.000135112	1.75%
70	100	0.05	0.2	0.2	80	4.37E-05	4.13E-05	2.43E-06	5.56%
70	100	0.05	0.2	0.4	80	0.00315247	0.00300891	0.000143558	4.55%
70	100	0.05	0.2	0.6	80	0.0139771	0.0135233	0.00045381	3.25%
70	100	0.05	0.2	0.8	80	0.0302296	0.0294736	0.000755977	2.50%
70	100	0.05	0.2	1	80	0.0486983	0.0477074	0.000990902	2.03%
80	100	0.05	0.2	0.2	80	0.00752105	0.00638082	0.00114024	15.16%
80	100	0.05	0.2	0.4	80	0.046594	0.0427122	0.00388183	8.33%
80	100	0.05	0.2	0.6	80	0.0899788	0.084715	0.00526376	5.85%
80	100	0.05	0.2	0.8	80	0.127594	0.121766	0.00582815	4.57%
80	100	0.05	0.2	1	80	0.158944	0.152936	0.00600745	3.78%
90	100	0.05	0.2	0.2	80	0.131983	0.141905	-0.00992259	7.52%
90	100	0.05	0.2	0.4	80	0.225674	0.235718	-0.0100437	4.45%
90	100	0.05	0.2	0.6	80	0.277943	0.287115	-0.00917192	3.30%
90	100	0.05	0.2	0.8	80	0.311891	0.320242	-0.00835062	2.68%
90	100	0.05	0.2	1	80	0.335936	0.343596	-0.00765992	2.28%
100	100	0.05	0.2	0.2	80	0.521501	0.493877	0.0276234	5.30%
100	100	0.05	0.2	0.4	80	0.527141	0.507885	0.0192566	3.65%
100	100	0.05	0.2	0.6	80	0.530105	0.514587	0.0155179	2.93%
100	100	0.05	0.2	0.8	80	0.531666	0.518398	0.0132676	2.50%
100	100	0.05	0.2	1	80	0.532325	0.520608	0.0117171	2.20%
110	100	0.05	0.2	0.2	80	0.862656	0.870727	-0.00807055	0.94%
110	100	0.05	0.2	0.4	80	0.786003	0.793145	-0.00714212	0.91%
110	100	0.05	0.2	0.6	80	0.745047	0.751273	-0.00622623	0.84%
110	100	0.05	0.2	0.8	80	0.718289	0.723826	-0.00553756	0.77%
110	100	0.05	0.2	1	80	0.6987	0.703709	-0.00500886	0.72%

Part 2

In this part we will consider the expected value of the discounted payoff under the risk-neutral density Q:

$$V(S,t) = E^Q\left[e^{-\int_t^T r_\tau d\tau}\textbf{Payoff}(S_T)\right] \quad (2)$$

We will consider the interest rate as a constant. So (3) becomes:

$$V(S,t) = e^{-r(T-t)}E^Q\left[\textbf{Payoff}(S_T)\right] \quad (3)$$

to obtain an approximation to the Black-Scholes price given by (2) where the underlying should be simulated using the Milstein scheme.
The Milstein scheme applied to the GBM: $dS_t = rS_t dt + \sigma S_t dW_t$ give us :

$$S_{t+\delta t} = S_t\left[1 + r\delta t + \sigma\phi\sqrt{\delta t} + \frac{1}{2}\sigma^2(\phi^2 - 1)dt\right]$$

where $\phi \sim N(0,1)$.
The Python code for the Monte Carlo method gives us the following results compared to the closed Black and Scholes formula.
We will examine the impact of varying the time step and the number of simulations on the price.

a) Fixing the number of simulations to 10000 and the number of time step to 1000 gives us a price closed to the price obtained by the Black and Scholes formula.

b) Fixing the step size to 0.01 and varying the number of simulation form 1000 to 10000 we can observe that the Monte Carlo price converges to the Black-Scholes price starting 3000 simulations. The increase of the number of simulations increases the Monte Carlo accuracy as showed on the following table.

S	**E**	**r**	**sigma**	**T**	**tStep**	**NbSimu**	**BS Price**	**MC Price**	**ErrWithBS**	**%ErrWithBS**
100	100	0.05	0.2	1	0.01	1000	0.532325	0.552664	-0.0203395	3.82%
100	100	0.05	0.2	1	0.01	2000	0.532325	0.522701	0.00962427	1.81%
100	100	0.05	0.2	1	0.01	3000	0.532325	0.53903	-0.00670517	1.26%
100	100	0.05	0.2	1	0.01	4000	0.532325	0.530073	0.00225224	0.42%
100	100	0.05	0.2	1	0.01	5000	0.532325	0.539728	-0.00740274	1.39%
100	100	0.05	0.2	1	0.01	6000	0.532325	0.531262	0.0010632	0.20%
100	100	0.05	0.2	1	0.01	7000	0.532325	0.537988	-0.00566335	1.06%
100	100	0.05	0.2	1	0.01	8000	0.532325	0.535304	-0.00297952	0.56%
100	100	0.05	0.2	1	0.01	9000	0.532325	0.532583	-0.000257951	0.05%
100	100	0.05	0.2	1	0.01	10000	0.532325	0.53773	-0.00540516	1.02%

On the other hand, fixing the number of simulation to 10000 and varying the time step size from 0.01 to 0.001, we can observe that the Monte Carlo price is sensitive to the time step size as we can see on the following table.

S	**E**	**r**	**sigma**	**T**	**tStep**	**NbSimu**	**BS Price**	**MC Price**	**ErrWithBS**	**%ErrWithBS**
100	100	0.05	0.2	1	0.01	10000	0.532325	0.531167	0.00115832	0.22%
100	100	0.05	0.2	1	0.005	10000	0.532325	0.519942	0.0123828	2.33%
100	100	0.05	0.2	1	0.0033	10000	0.532325	0.528503	0.00382177	0.72%
100	100	0.05	0.2	1	0.0025	10000	0.532325	0.542391	-0.0100662	1.89%
100	100	0.05	0.2	1	0.002	10000	0.532325	0.532308	1.68E-05	0.00%
100	100	0.05	0.2	1	0.0017	10000	0.532325	0.541915	-0.00959057	1.80%
100	100	0.05	0.2	1	0.0014	10000	0.532325	0.530501	0.00182418	0.34%
100	100	0.05	0.2	1	0.0012	10000	0.532325	0.533545	-0.00121975	0.23%
100	100	0.05	0.2	1	0.0011	10000	0.532325	0.541725	-0.00940032	1.77%
100	100	0.05	0.2	1	0.001	10000	0.532325	0.534496	-0.00217098	0.41%

Python Code

The submitted solution consists of:

- BlackScholes.py : Code for a binary call option using Black-Scholes formula.
- unittest_BlackScholes.py: Black-Scholes formula tests.
- ExplicitFiniteDifference.py : Code for a binary call option using FDM forward scheme.
- unittest_ExplicitFiniteDifference.py: Tests of the FDM method.
- MonteCarlo.py: Monte Carlo code for a binary call.
- unittest_MonteCarlo.py: Monte Carlo test examples.

Output: generated in the same directory of the provided solution.

- binaryCallFDMPrice.csv
- binaryCallMCPriceTest1.csv
- binaryCallMCPriceTest2.csv