FINM3405 Derivatives and risk management

Tutorial Sheet 8: Options - American and path dependent

Suggested solutions

September 7, 2024

In the following questions let $S = 50$, $K = 50$, $r = 5\%$, $T = \frac{1}{2}$, $\sigma = 25\%$ and the continuously compounded dividend yield $y = 0$ unless otherwise indicated.

American options

Question 1. Calculate by hand the prices of ATM American calls and puts using a 2 layer binomial tree. Then do the same for when $y = 7\%$. Also calculate the deltas. Again, you're welcome to use the CRR or JR schemes. By a 2 layer tree I mean $N = 2$ as per the lecture notes so there is 3 dates: $t_0 = 0$ (now), t_1 and $t_2 = T$ (expiry). Since $T = \frac{1}{2}$ (6 months) and there is two periods ($N = 2$), we have $\mathrm{d}t = \frac{1}{4}$ (3 months or a quarter). I like the JR scheme since it resembles geometric Brownian motion, which when $y = 0$ is

$$u = e^{(r-\frac{1}{2}\sigma^2)\mathrm{d}t+\sigma\sqrt{\mathrm{d}t}} = 1.138473 \qquad \text{and} \qquad d = u = e^{(r-\frac{1}{2}\sigma^2)\mathrm{d}t-\sigma\sqrt{\mathrm{d}t}} = 0.88664.$$

I then calculate the risk-neutral probability to be

$$q = \frac{e^{r\mathrm{d}t} - d}{u - d} = 0.50008.$$

The first layer of asset prices, at time $t_1 = \frac{1}{4}$, is

$$S_u = Su = 56.92363 \qquad \text{and} \qquad S_d = Sd = 44.3322.$$

The final asset prices are expiry are

$$S_{uu} = Suu = 64.80599, \quad S_{ud} = S_{du} = Sud = 50.471, \quad S_{dd} = Sdd = 39.30682.$$

Hence, the call option payoffs at expiry are easily seen to be

$$C_{uu} = 4.80599, \qquad C_{ud} = 0.471 \qquad \text{and} \qquad C_{dd} = 0.$$

We then calculate the t_1 call prices to be

$$C_u = e^{-r\mathrm{d}t}\big[qC_{uu} + (1-q)C_{ud}\big] = 7.54474,$$
$$C_d = e^{-r\mathrm{d}t}\big[qC_{ud} + (1-q)C_{dd}\big] = 0.23259.$$

To calculate American option prices, we compare these to the call's intrinsic values at these nodes. The asset prices are $S_u = 56.92363$, giving a call's intrinsic value of 6.92363 which is less than the call price, so we ignore it. The other asset price is $S_d = 44.3322$ giving an intrinsic value of 0, which we again ignore. So the call price is given by

$$C = e^{-r\mathrm{d}t}\big[qC_u + (1-q)C_d\big] = 3.84.$$

This compares with the Black-Scholes price of $C = 4.13$. Using the same process, I calculate the time t_1 put prices to be

$$P_u = e^{-r\mathrm{d}t}\big[qP_{uu} + (1-q)P_{ud}\big] = 0,$$
$$P_d = e^{-r\mathrm{d}t}\big[qP_{ud} + (1-q)P_{dd}\big] = 5.2793.$$

These compare to the intrinsic values of 0 at S_u and 5.66783 at S_d, which is greater than P_d so we use it instead to get

$$P = e^{-r\mathrm{d}t}\big[qP_u + (1-q)5.66783\big] = 2.7983.$$

As a check, my Python code gives the following asset and option price trees:

	0	1	2
0	50	44.3322	39.3068
1	nan	56.9236	50.471
2	nan	nan	64.806

	0	1	2
0	3.84095	0.23259	0
1	nan	7.54474	0.470954
2	nan	nan	14.806

	0	1	2
0	2.79826	5.66783	10.6932
1	nan	0	0
2	nan	nan	0

Recall that the trees are upside down here. The option deltas at time $t_0 = 0$ are then very easy to calculate as well:

$$\Delta_C = \frac{C_u - C_d}{S_u - S_d} = 0.5807 \qquad \text{and} \qquad \Delta_P = \frac{P_u - P_d}{S_u - S_d} = -0.4193.$$

When incorporating a continuously compounded dividend yield of $y = 7\%$, the only changes we make are to the:

1. JR parameterisation scheme, in which case we get

$$u = e^{(r-q-\frac{1}{2}\sigma^2)\mathrm{d}t + \sigma\sqrt{\mathrm{d}t}} = 1.118723 \quad \text{and} \quad d = u = e^{(r-q-\frac{1}{2}\sigma^2)\mathrm{d}t - \sigma\sqrt{\mathrm{d}t}} = 0.871262.$$

2. Risk-neutral probability, which is

$$q = \frac{e^{(r-q)\mathrm{d}t} - d}{u - d} = 0.50008.$$

We then proceed in the same way as before. I get the following trees:

	0	1	2
0	50	43.5631	37.9549
1	nan	55.9361	48.735
2	nan	nan	62.577

	0	1	2
0	3.06762	0	0
1	nan	6.2114	0
2	nan	nan	12.577

	0	1	2
0	3.55285	6.57151	12.0451
1	nan	0.624528	1.26497
2	nan	nan	0

The American call price is $C = 3.06762$ and the put price is $P = 3.55285$. The deltas are calculated from this and I get $\Delta_C = 0.502$ and $\Delta_P = -0.481$.

Question 2. Modify your Excel spreadsheet for using a 7 layer binomial model to price European options to calculate American option prices and deltas, including for the case of $y = 7\%$. See the provided spreadsheet.

Question 3. Let the USD:EUR exchange rate be 0.9. Use your 7 layer binomial models to calculate the prices of 6 month ATM USD:EUR FX American and European options. Let $\sigma = 15\%$, Euribor be 3.38%, and Term SOFR be 4.62%. Recall that for FX options, we just view the foreign interest rate as the "dividend yield". We also have to be careful about our notation in terms of the quoting convention. We want to price an option on the foreign currency when it is viewed as the "underlying asset". Here, the spot price is $S_{\text{f:d}}$ which is a f:d quote and gives the price of 1 unit of the foreign currency in terms of the domestic currency. The strike price is the same quoting convention and written as $K_{\text{f:d}}$. The resulting option prices are then in the domestic currency. So, in our question we have $S_{\text{f:d}} = K_{\text{f:d}} = 0.9$, in which case we are viewing the USD as the foreign currency and the resulting option premiums are in EUR. We also have $r_{\text{d}} = 3.38\%$ and $r_{\text{f}} = 4.62\%$. We then just need to plug these numbers correctly into our code or spreadsheets. See the provided spreadsheets. For the European FX call I get $C^{\text{Eu}} = 0.0357$ and the American call is $C^{\text{Am}} = 0.03642$. I calculate the European and American put prices to be the same of $P = 0.041214$. For the Black-Scholes European FX option prices I get $C = 0.034644$ and $P = 0.04011$.

Path dependent options

Question 4. Modify your excel code to price an ATM chooser option via a 6 layer binomial tree, so each date or layer of the tree coincides with the end of a month, with choice date in 3 months. Also do the same for the case of $y = 7\%$. See spreadsheet provided. Below is the call, put and chooser option trees I get for the case of $y = 0.07$:

Call price tree

						25.144
					20.011	
				15.257		15.045
			11.010		10.629	
		7.569		6.855		6.302
	5.001		4.191		3.138	
3.200		2.474		1.563		0.000
	1.425		0.778		0.000	
		0.387		0.000		0.000
			0.000		0.000	
				0.000		0.000
					0.000	
						0.000

Put price tree

						0.000
					0.000	
				0.000		0.000
			0.156		0.000	
		0.791		0.314		0.000
	1.996		1.433		0.630	
3.685		3.217		2.565		1.265
	5.405		5.027		4.521	
		7.639		7.532		7.815
			10.315		10.606	
				13.184		13.485
					15.873	
						18.393

Chooser option price tree

			11.010
		7.569	
	6.054		4.191
6.047		4.590	
	6.089		5.027
		7.639	
			10.315

Here's the Python code that includes a dividend yield with the JR scheme:

```python
S = 50; K = 50; sigma = 0.25; r = 0.05; T = 1/2; y = 0.07
N = 6; dt = T/N
dates = np.linspace(0,T,N+1) # asset price tree dates
cdate = 1/4 # the choice date
cidx = math.ceil(N*cdate/T) # the index of the choice date
u = np.exp((r - y - 0.5*sigma**2)*dt + sigma*np.sqrt(dt)) # JR
d = np.exp((r - y - 0.5*sigma**2)*dt - sigma*np.sqrt(dt)) # JR
q = (np.exp((r-y)*dt)-d)/(u-d)
# expiry payoffs
Ct = np.zeros([N+1 ,N+1])*np.nan
Pt = np.zeros([N+1, N+1])*np.nan
for i in range(N+1):
    ST = S*(u**i)*d**(N-i)
    Ct[i,N]=max(0, ST-K)
    Pt[i,N]=max(0, K-ST)
# chooser option value
Vt = np.zeros([N+1, N+1])*np.nan
for j in reversed(range(N)):
    if j>cidx: # after the choice date
        for i in range(j+1):
            Ct[i,j] = np.exp(-r*dt)*(q*Ct[i+1, j+1]+(1-q)*Ct[i, j+1])
            Pt[i,j] = np.exp(-r*dt)*(q*Pt[i+1, j+1]+(1-q)*Pt[i, j+1])
    if j==cidx: # at the choice date
        for i in range(j+1):
            Ct[i,j] = np.exp(-r*dt)*(q*Ct[i+1, j+1]+(1-q)*Ct[i, j+1])
            Pt[i,j] = np.exp(-r*dt)*(q*Pt[i+1, j+1]+(1-q)*Pt[i, j+1])
            Vt[i,j] = max(Ct[i,j], Pt[i,j]) # take the max value on choice date
    if j<cidx: # before the choice date
        for i in range(j+1):
            Vt[i,j] = np.exp(-r*dt)*(q*Vt[i+1, j+1]+(1-q)*Vt[i, j+1])
V = Vt[0,0]
```

Question 5. Use Excel to create 100 asset price paths over 10 time steps to calculate the prices of the lookback and Asian path dependent options. This is not hard to do, albeit a bit laborious, is Excel. See spreadsheet provided.

Question 6. How might you incorporate a continuous dividend yield of $y = 7\%$ into Monte Carlo option pricing? Do this first for European calls and puts, and compare the prices to the Black-Scholes European prices to check that you've got things right. Then do it for the European FX and lookback, barrier and Asian path dependent options. *Hint*: It's much simpler than one may think. Hopefully your guess would be to simply simulate $i = 1, \ldots, N$ asset price paths of geometric Brownian motion over M time steps, with each path starting at S and path i being given by

$$S_{ij} = S_{i,j-1}e^{(r-y-\frac{1}{2}\sigma^2)\mathrm{d}t+\sigma\sqrt{\mathrm{d}t}Z_{ij}} \qquad \text{for } j = 1, \ldots, M,$$

where each Z_{ij} are independent, identically distributed standard normal random variables. Then everything is the same as before. Some Python code for plain vanilla European calls and puts (but not the simplified method since we also want to price path dependent options) is:

```
S = 50; K = 50; sigma = 0.25; r = 0.05; T = 1/2; y = 0.07
N = 15000; M = 2000; dt = T/M
St = np.zeros([N, M+1])
St[:,0] = S
CT = np.zeros(N)
PT = np.zeros(N)
for i in range(N):
    for j in range(1, M+1):
        St[i,j] = St[i,j-1]*np.exp((r-y-0.5*sigma**2)*dt + sigma*np.sqrt(dt)*norm.rvs()) # -y
    CT[i] = max(0, St[i,M]-K)
    PT[i] = max(0, K-St[i,M])
C = np.exp(-r*T)*np.mean(CT)
P = np.exp(-r*T)*np.mean(PT)
```

It gives $C = 3.19696$ and $P = 3.666273$. I calculate the Black-Scholes prices to be 3.1804 and $P = 3.66563$. For the European FX options from above, in this Python code we set $S = K = 0.9$, $\sigma = 0.15$, $r = 0.0388$ (Euribor is the domestic interest rate) and $y = 0.0462$ (Term SOFR is the foreign interest rate) to get $C = 0.034352$ and $P = 0.04054$, and we recall that the Black-Scholes prices were $C = 0.034644$ and $P = 0.04011$.

See the modified Excel spreadsheet provided incorporating dividends into the lookback and Asian path dependent options from the other question. Below is the Python code incorporating dividends for the barrier options not included in the spreadsheet and other question. It gives:

```
In [117]: Cui
Out[117]: 2.4019143966023897

In [118]: Pui
Out[118]: 0.057130200263640595

In [119]: Cdi
Out[119]: 0.016672365836403802

In [120]: Pdi
Out[120]: 2.495376140650439

In [121]: Cuo
Out[121]: 0.6830039531210736

In [122]: Puo
Out[122]: 3.65317232197383

In [123]: Cdo
Out[123]: 3.06824598388706

In [124]: Pdo
Out[124]: 1.2149263815870315
```

```python
S = 50
K = 50
Bu = 60 # set up barrier above S
Bd = 40 # set down barrier below S
sigma = 0.25
r = 0.05
T = 1/2
y = 0.07
N = 10000 # number of paths
M = 2000 # number of time steps or dates
dt = T/M
CTui = np.zeros(N)
PTui = np.zeros(N)
CTdi = np.zeros(N)
PTdi = np.zeros(N)
CTuo = np.zeros(N)
PTuo = np.zeros(N)
CTdo = np.zeros(N)
PTdo = np.zeros(N)
St = np.zeros([N, M+1])
St[:,0] = S
for i in range(N):
    for j in range(1, M+1):
        St[i,j] = St[i,j-1]*np.exp((r-y-0.5*sigma**2)*dt + sigma*np.sqrt(dt)*norm.rvs())
    ST = St[i,M] # final asset prices at expiry
    Smax = np.max(St[i, 0:M]) # maximum asset price of path i
    Smin = np.min(St[i, 0:M]) # minimum asset price of path i
    CTui[i] = max(0,ST-K) if Smax>=Bu else 0 # up-and-in call
    PTui[i] = max(0,K-ST) if Smax>=Bu else 0 # up-and-in put
    CTdi[i] = max(0,ST-K) if Smin<=Bd else 0 # down-and-in call
    PTdi[i] = max(0,K-ST) if Smin<=Bd else 0 # down-and-in put
    CTuo[i] = 0 if Smax>=Bu else max(0,ST-K) # up-and-out call
    PTuo[i] = 0 if Smax>=Bu else max(0,K-ST) # up-and-out put
    CTdo[i] = 0 if Smin<=Bd else max(0,ST-K) # down-and-out call
    PTdo[i] = 0 if Smin<=Bd else max(0,K-ST) # down-and-out put
Cui = np.exp(-r*T)*np.mean(CTui)
Pui = np.exp(-r*T)*np.mean(PTui)
Cdi = np.exp(-r*T)*np.mean(CTdi)
Pdi = np.exp(-r*T)*np.mean(PTdi)
Cuo = np.exp(-r*T)*np.mean(CTuo)
Puo = np.exp(-r*T)*np.mean(PTuo)
Cdo = np.exp(-r*T)*np.mean(CTdo)
Pdo = np.exp(-r*T)*np.mean(PTdo)
```