Case 1: Binomial Trees and Geometric Brownian Motion

Aytek Mutlu - 2648232

17 September 2019

1 Introduction

In this case project, variety of option prices are calculated with the help of binomial trees and the results are compared with Black-Scholes calculations. For this purpose, S&P-500 Index is selected as the underlying. All implementations are done in Matlab.

2 Part A

S&P-500 Index historical data is obtained from Yahoo Finance with the following details and summary statistics (Table 1):

Start Date	01/01/1991
End Date	01/09/2019
Number of Observations	345
Number of Returns Calculated	344
Last Price	2978.43
Mean Monthly Return	0.63%
Standard Deviation of Monthly Return	4.11%
Skewness of Monthly Return	-0.8220
Kurtosis of Monthly Return	4.8698
Max Monthly Return	10.58%
Min Monthly Return	-18.56%

Table 1: Main Statistical Features of Input Data

Figure 1 illustrates a graph of the monthly returns:

3 Part B

As each of the option price calculations in this project are based on binomial trees, a Matlab function that prepares a binomial tree, its interest rates for each

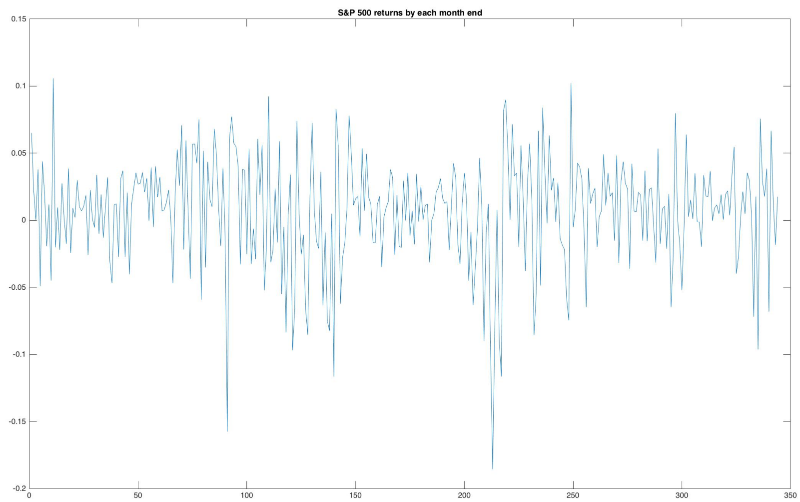

Figure 1: Monthly Returns of S&P-500 Index

step and its up and down probabilities is coded. The inputs needed for this function and therefore preparation of a binomial tree is as follows in Table 2:

Price	Price of the underlying at valuation date
Volatility	Standard deviation of the underlying (function expects the input to be monthly standard deviation and then it re-scales it to the necessary time-scale within the function)
Number of Steps	Number of steps in binomial tree
Interest Rate	Annualized continuously compounded interest rate
Option Maturity	Option maturity denoted in years

Table 2: Inputs needed for preparation of Binomial Tree

Function code can be seen in Annex: Binomial Tree. For a binomial tree request with 3 steps and initial price of 2978.40 (price of S&P-500 as of 31/08/2019), binomial tree illustrated in Figure 2 is obtained.

4 Part C

Tree function mentioned in Part B and illustrated in Annex: Binomial Tree also calculates the risk-neutral upward and downward probabilities on its 15th to 16th lines. Here are the results (Table 3):

BinTree

4x4 double

	1	2	3	4
1	2978.4	3103.3	3233.4	3369.0
2	0	2858.6	2978.4	3103.3
3	0	0	2743.5	2858.6
4	0	0	0	2633.1

Figure 2: Binomial Tree with 3 steps

Upward Probability	49.99%
Downward Probability	50.01%

Table 3: Upward and downward probabilities of a binomial tree for S&P-500 Index ATM Option with 3 months maturity with 3 steps

5 Part D

In order to calculate the call price with binomial model, a call function (Annex: European Call) is coded. The inputs of the function are simply the binomial tree to be used, upward and downward probabilities and the corresponding interest rate. For comparison, well-known Black-Scholes models is built as a function (Annex: European Call with Black-Scholes).

Here are the calculation results (Table 4):

Option	European Call
Strike	3000
Maturity	3 months
Number of Steps	3
Calculated Price	**84.6272**
Price with Black-Scholes	**77.8747**

Table 4: Calculation Details of a European Call Option on S&P-500 Index

6 Part E

Call options with 3000 strike is re-priced with increasing number of steps in order to observe its convergence. For this purpose, binomial trees are re-generated for each steps and option calculations are repeated (Annex: Main Code line 45-59). The results are illustrated in numbers in Table 5 and plotted against Black-Scholes call price in Figure 3.

Option	European Call
Strike	3000
Maturity	3 months
Calculated Price with 3 steps	84.6272
Calculated Price with 4 steps	76.4043
Calculated Price with 5 steps	81.7267
Calculated Price with 6 steps	77.3692
Calculated Price with 7 steps	80.4895
Calculated Price with 8 steps	77.7887
Calculated Price with 9 steps	79.8054
Calculated Price with 10 steps	78.0046
Calculated Price with 25 steps	78.2851
Calculated Price with 50 steps	77.7203
Calculated Price with 75 steps	78.0846
Calculated Price with 100 steps	77.9983
Calculated Price with 150 steps	77.9399
Calculated Price with 250 steps	77.8972

Table 5: Change in price of European Call Option on S&P-500 Index with increasing number of steps in binomial tree

Table 5 and Figure 3 clearly shows that the increasing number of steps in binomial tree provides a convergence towards Black-Scholes option price.

7 Part F

Similar to call price calculation, a function is coded for put prices (Annex: European Put, Annex: European Put with Black-Scholes). Results are as follows in Table 6:

Option	European Put
Strike	3000
Maturity	3 months
Number of Steps	3
Calculated Price	**98.6785**
Price with Black-Scholes	**91.9260**

Table 6: Calculation Details of a European Put Option on S&P-500 Index

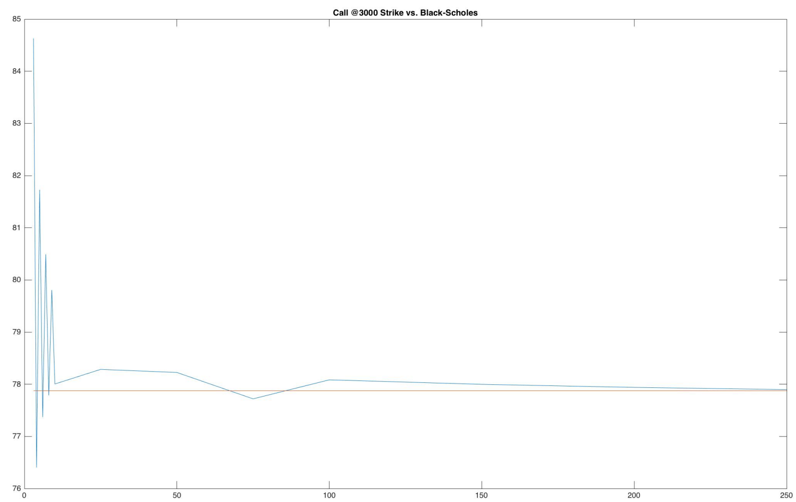

Figure 3: European Call Prices with strike 3000 with increasing number of steps vs. Black-Scholes Call Price

A similar convergence analysis (Part E) is also performed to European put prices and the following plot is obtained (Figure 4):

Figure 4 clearly shows that the increasing number of steps in binomial tree provides a convergence towards Black-Scholes option price for put prices as well.

Based on identical number of steps and same binomial trees, one should expect put-call parity to hold.

$$C + PV(x) = P + S \tag{1}$$

where C is the call price, $PV(x)$ is the present value of the strike discounted with risk-free rate, P is the put price and S is the current price of the underlying. Call price calculated in Part D and put price calculated in Part F is used as C and P and the calculation is made in Appendix: Main Code in lines 70-71. The result shows that the LHS and RHS of the equality are indeed identical (with some negligible rounding errors). Therefore, put-call parity holds (Table 7)

Call Option Value	84.6272
Present Value of Strike	2992.50
SUM	**3077.1**
Put Option Value	98.6785
Stock Price	2978.4
SUM	**3077.1**

Table 7: Verification of Put-Call Parity

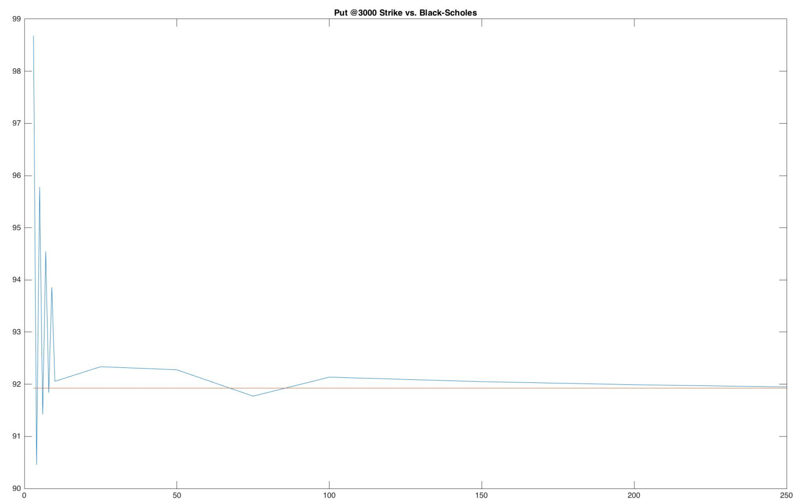

Figure 4: European Put Prices with strike 3000 with increasing number of steps vs. Black-Scholes Put Price

8 Part G

Call and put option prices are calculated with the help of same functions for different strikes. Summary of the results is as follows (Table 8):

Strike	**Call Price with Binomial Tree**	**Call Price with Black-Scholes**	**Put Price with Binomial Tree**	**Put Price with Black-Scholes**
2500	484.6955	485.0880	0.0000	0.3925
2600	384.9462	386.8867	0.0000	1.9405
2700	293.5388	292.3443	8.3420	7.1475
2800	206.2644	205.9107	20.8170	20.4633
2900	134.4894	133.0626	48.7913	47.3645
3000	84.6272	77.8747	98.6785	91.9260
3100	34.7650	40.9170	148.5657	154.7177
3200	21.0630	19.2158	234.6131	232.7659
3300	8.6006	8.0598	321.9000	321.3593
3400	0.0000	3.0251	413.0488	416.0740
3500	0.0000	1.0199	512.7982	513.8181

Table 8: Change in Call and Put Prices with changing strikes

Both Table 8 and Figure 5 indicates expected results such that binomial tree pricing behaves almost same as Black-Scholes pricing.

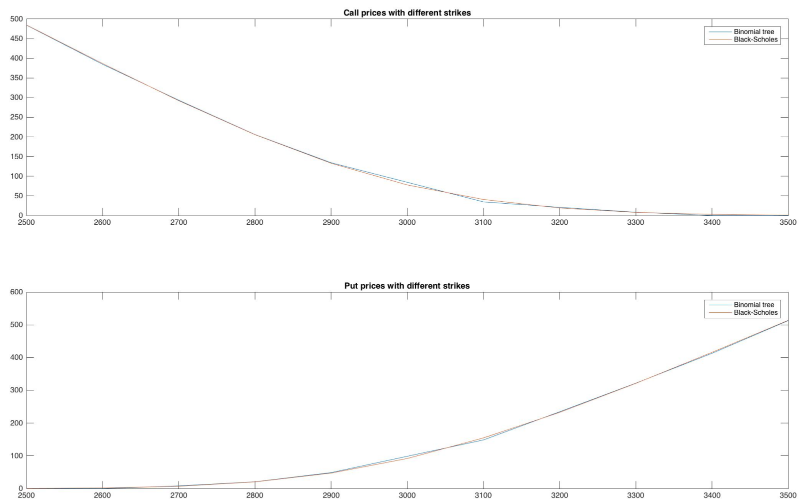

Figure 5: European Call and Put prices with different strikes compared with Black-Scholes prices

9 Part H

For American call and put pricing, new functions are coded (Annex: American Call, Annex: American Put). An early-exercise flag is included as output in order to detect an early exercise. Here are the results (Table 9 and Table 10):

Option	American Call
Strike	3000
Maturity	3 months
Number of Steps	3
Calculated Price	**84.6272**
Early-Exercise Flag	**False**
European Call Price	**84.6272**

Table 9: American Call Option on S&P-500 Index Pricing Details

Observation of European option prices also verifies and re-confirms that American call option has no early exercise opportunity and therefore resulted in exact same price with European call option of same strike. However, American Put option has an early-exercise opportunity and therefore it is more expensive than European Put option of same strike. The present value of the early exercise opportunity for put option is basically around 0.63 $.

Option	American Put
Strike	3000
Maturity	3 months
Number of Steps	3
Calculated Price	**99.3048**
Early-Exercise Flag	**True**
European Put Price	**98.6785**

Table 10: American Put Option on S&P-500 Index Pricing Details

10 Part I

In this section, an exotic digital European option is defined. This specific option is priced with a new function (Annex: European Exotic Option) for different strikes and the results are shown in Table 11 and Figure 6.

Strike	Option Price
2500	280,557.90
2750	100,553.49
3000	45,235.79
3250	114,604.82
3500	308,660.57

Table 11: European Exotic Option Price with changing strikes

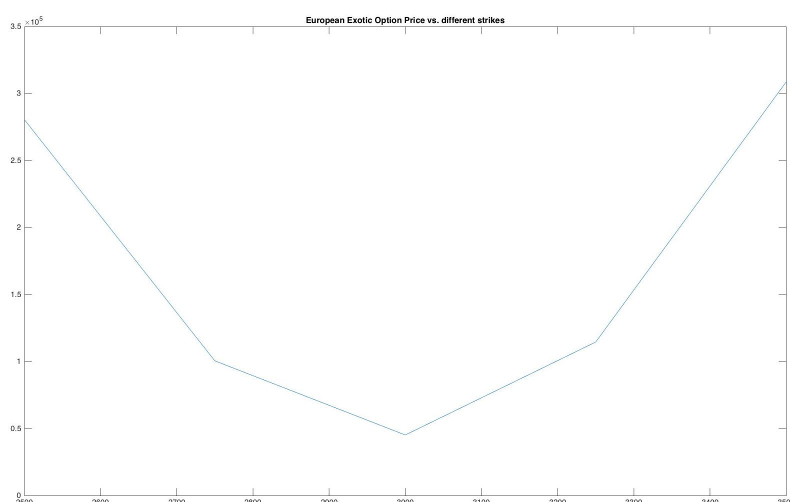

Figure 6: European Exotic option prices with different strikes

It is natural for this option to be more expensive as the strike moves away from the current price of the underlying. Probability to end up with a terminal value around the strike reduces as the strike moves away from the current price and therefore the pay-off of the option increases. This is why, the option is more expensive at edges and cheaper at the middle of the strike range.

11 Part J

In this section, two different approaches are assumed to be implied. It is decided to implement both assumptions.

11.1 Part J - Brownian Motion Assumption

Under this assumption, S&P-500 Index is believed to follow a Geometric Brownian Motion with the mean of risk-free rate and its own historical standard deviation. Then, 1000 paths are generated for S&P-500, its returns are calculated and the potential payoff of the options are discounted to present. Finally, average of the option prices are compared with Black-Scholes calculation and Binomial Tree calculation. This operations are done in Main Code in line 123-135.

Here are the results in Table 12 and Table 13:

Option	European Call with Random Independent Paths
Strike	3000
Maturity	3 months
Number of Steps	250
Calculated Price	**77.9735**
European Call Price with Single Binomial Tree with 250 steps	**77.8970**
European Call Price with Black-Scholes	**77.8747**

Table 12: European Call Option price calculated with 1000 random independent paths with Geometric Brownian Motion

11.2 Part J - Binomial Assumption

In this assumption, it is believed that the question asks for generation of 1000 independent binomial trees where interest rate of the underlying is not constant but follows a normal distribution with mean of the return of the underlying and the standard deviation of the return on the underlying. Therefore, interest rate cannot be assumed constant at each step but needs to be independent an random.

Option	European Put with Random Independent Paths
Strike	3000
Maturity	3 months
Number of Steps	250
Calculated Price	**91.4582**
European Put Price with Single Binomial Tree with 250 steps	**91.9480**
European Put Price with Black-Scholes	**91.9260**

Table 13: European Put Option price calculated with 1000 random independent paths with Geometric Brownian Motion

Abovementioned random annualized interest rates are generated for each step of each independent path in Appendix: Main Code in line 139-147.

Then, new functions for Binomial tree generation (Appendix: Binomial Tree with Independent Random Paths), call pricing (Appendix: European Call with Independent Random Paths) and put pricing (Appendix: European Put with Independent Random Paths) is coded. The only change in those functions from its originals are the ability to use matrices with length of number of steps for the interest rates, probability of ups and probability of downs instead of constants. This way, functions utilize independently generated random interest rates at each step.

Finally, call prices and put prices are individually averaged on 1000 independent paths. Here are the results (Table 14 and Table 15):

Option	European Call with Random Independent Paths
Strike	3000
Maturity	3 months
Number of Steps	250
Calculated Price	**77.9305**
European Call Price with Single Binomial Tree with 250 steps	**77.8970**
European Call Price with Black-Scholes	**77.8747**

Table 14: European Call Option price calculated with 1000 random independent paths with binomial trees

To conclude, simulation with random independent interest rates resulted in very similar option prices on both call option and put option using both GBM assumption and Binomial Tree assumption.

Option	European Put with Random Independent Paths
Strike	3000
Maturity	3 months
Number of Steps	250
Calculated Price	**91.9050**
European Put Price with Single Binomial Tree with 250 steps	**91.9480**
European Put Price with Black-Scholes	**91.9260**

Table 15: European Put Option price calculated with 1000 random independent paths with binomial trees

12 Conclusion

This case project was based on option pricing with Binomial Trees and it has investigated the accuracy of it based on Black-Scholes as benchmark. Moreover, it is convergence to Black-Scholes pricing with increasing steps is proven. On top of that, it is proven that early exercising an American call option with underlying of a non-dividend paying stock whereas it may be an opportunity to early-exercise an American put option.

Finally, a custom-defined exotic digital option is priced and last but not the least, a simulation is run for 1000 independent paths in order to observe that the average option price converges to Black-Scholes price.

13 Annex

13.1 Main Code

```

%%%Case 1

%%%read data
filename = '^GSPC.csv';
sp_500 = readtable(filename);

%%calculate returns
[returns,intervals] = price2ret(sp_500.('AdjClose'));
sp_500_returns = returns;

%%%statistical properties
mean_sp_500_returns = mean(sp_500_returns);
std_sp_500_returns = std(sp_500_returns);
skewness_sp_500_returns = skewness(sp_500_returns);
kurtosis_sp_500_returns = kurtosis(sp_500_returns);

max_sp_500_returns = max(sp_500_returns);
min_sp_500_returns = min(sp_500_returns);

%%plot
plot((1:length(sp_500_returns)),sp_500_returns)
title('S&P 500 returns by each month end')

%%lastprice and dates
last_price = sp_500.('AdjClose')(end);
NumPeriods = 3;
int_rate = 0.01;
compound_freq = 0.25;
option_maturity = 0.25;
annual_simple_int_rate = power((1+int_rate*compound_freq)
    ,1/compound_freq)-1;

[BinTree,rate,p_up,p_down] = tree(last_price,
    std_sp_500_returns,NumPeriods,annual_simple_int_rate,
    option_maturity);

%%3000 strike european call
europ_call_3000 = call(BinTree,3000,rate,p_up,p_down);
europ_call_3000_bs  = bs_call(last_price,3000,
    annual_simple_int_rate,option_maturity,(
    std_sp_500_returns*sqrt(12)));
```

```
%%european put strike 3000
europ_put_3000 = put(BinTree,3000,rate,p_up,p_down);
europ_put_3000_bs  = bs_put(last_price,3000,annual_simple_int_rate,option_maturity,std_sp_500_returns*sqrt(12));



%%build tree with dif. number of steps
steps = [3,4,5,6,7,8,9,10,25, 50,75, 100, 150, 200, 250];
europ_call_3000_prices = zeros(1,length(steps));
europ_put_3000_prices = zeros(1,length(steps));

step_count = 1;

for NumPeriods = steps
    [BinTree,rate,p_up,p_down] = tree(last_price,std_sp_500_returns,NumPeriods,annual_simple_int_rate,option_maturity);

    europ_call_3000_prices(1,step_count) = call(BinTree,3000,rate,p_up,p_down);
    europ_put_3000_prices(1,step_count) = put(BinTree,3000,rate,p_up,p_down);

    step_count=step_count+1;
end

subplot(2,1,1);
plot(steps,europ_call_3000_prices,steps,europ_call_3000_bs*ones(1,length(steps)));
title('Call @3000 Strike vs. Black-Scholes')
subplot(2,1,2);
plot(steps,europ_put_3000_prices,steps,europ_put_3000_bs*ones(1,length(steps)));
title('Put @3000 Strike vs. Black-Scholes')


%%put-call parity
strike_pv = 3000/(1+annual_simple_int_rate*option_maturity);
put_call_parity_check  = ((europ_call_3000 + strike_pv) - (europ_put_3000 + last_price))
```

```
%%option prices for all strikes
NumPeriods = 3;
[BinTree,rate,p_up,p_down] = tree(last_price,std_sp_500_returns,NumPeriods,annual_simple_int_rate,option_maturity);


strikes = 2500:100:3500;
call_prices = zeros(1,length(strikes));
put_prices = zeros(1,length(strikes));
call_prices_bs = zeros(1,length(strikes));
put_prices_bs = zeros(1,length(strikes));
strike_count=1;

for strike=strikes
    call_prices(1,strike_count) = call(BinTree,strike,rate,p_up,p_down);
    put_prices(1,strike_count) = put(BinTree,strike,rate,p_up,p_down);
    call_prices_bs(1,strike_count) = bs_call(last_price,strike,annual_simple_int_rate,option_maturity,std_sp_500_returns*sqrt(12));
    put_prices_bs(1,strike_count) = bs_put(last_price,strike,annual_simple_int_rate,option_maturity,std_sp_500_returns*sqrt(12));
    strike_count = strike_count+1;
end

subplot(2,1,1);
plot(strikes,call_prices,strikes,call_prices_bs);
title('Call prices with different strikes')
legend('Binomial tree','Black-Scholes')

subplot(2,1,2);
plot(strikes,put_prices,strikes,put_prices_bs);
title('Put prices with different strikes')
legend('Binomial tree','Black-Scholes')

%%%%american call and american put
strike = 3000;
[american_call_3000,american_call_3000_early_exercise] = call_american(BinTree,strike,rate,p_up,p_down);
[american_put_3000,american_put_3000_early_exercise] = put_american(BinTree,strike,rate,p_up,p_down);
```

```
%%%european exotic
strikes = 2500:250:3500;
european_exotic_prices = zeros(1,length(strikes));
strike_count=1;
for strike=strikes
    european_exotic_prices(1,strike_count) = european_exotic(BinTree,strike,rate,p_up,p_down);
    strike_count = strike_count+1;
end

plot(strikes,european_exotic_prices);
title('European Exotic Option Price vs. different strikes')

%%%%1000 independent scenarios - Brownian Motion assumption
NumPeriods = 250;
time_interval = 1/(option_maturity/NumPeriods);
numScenario =2000;
strike = 3000;

random_scenario = log(1+annual_simple_int_rate * (1/time_interval)*ones(numScenario,NumPeriods) + std_sp_500_returns * sqrt(12/time_interval) * randn(numScenario,NumPeriods));
random_returns = exp(sum(random_scenario,2));

random_call_prices = max(last_price * random_returns - strike,0)*exp(-annual_simple_int_rate*option_maturity);
random_put_prices = max(strike - last_price * random_returns,0)*exp(-annual_simple_int_rate*option_maturity);

europ_call_3000_random = sum(random_call_prices)/numScenario;
europ_put_3000_random = sum(random_put_prices)/numScenario;


%%%%1000 independent scenarios - binomial tree assumption
numScenario = 1000;
strike  = 3000;
NumPeriods = 250;
mean = annual_simple_int_rate;
std = std_sp_500_returns * sqrt(12);
```

```
pd   = makedist('normal','mu',mean,'sigma',std);
int_rates_random = random(pd,numScenario,NumPeriods);
europ_call_3000_prices_random = zeros(1,length(
    numScenario));
europ_put_3000_prices_random = zeros(1,length(numScenario
    ));

path_count = 1;
for i=1:length(int_rates_random)
    rate_random = int_rates_random(i,:);
    [BinTree,rate_matrix,p_up_matrix,p_down_matrix] =
        tree_random(last_price,std_sp_500_returns,
        NumPeriods,rate_random,option_maturity);

    europ_call_3000_prices_random(1,path_count) =
        call_random(BinTree,strike,rate_matrix,p_up_matrix
        ,p_down_matrix);
    europ_put_3000_prices_random(1,path_count) =
        put_random(BinTree,strike,rate_matrix,p_up_matrix,
        p_down_matrix);

    path_count = path_count + 1;
end
europ_call_3000_random_binomial = sum(
    europ_call_3000_prices_random)/numScenario;
europ_put_3000_random_binomial = sum(
    europ_put_3000_prices_random)/numScenario;
```

13.2 Binomial Tree

```
function [BinTree,rate,p_up,p_down] = tree(last_price,
    std_sp_500_returns,NumPeriods,annual_simple_int_rate,
    option_maturity)
    u = exp(std_sp_500_returns*sqrt(3/NumPeriods));
    d = 1/u;

    BinTree = zeros(NumPeriods+1);

    %%build tree by hand
    for i = 1:NumPeriods+1
        for j=1:i
            BinTree(j,i) = last_price * power(u,i-j) *
                power(d,j-1);
        end
    end

```

```
    rate = exp(annual_simple_int_rate*option_maturity/NumPeriods)-1;
    p_up = (1+rate-d)/(u-d);
    p_down = 1-p_up;
end
```

13.3 European Call

```
function f = call(BinTree,Strike,rate,p_up,p_down)

    treeLength = length(BinTree);
    OptPrice(:,treeLength) = max(0,BinTree(:,treeLength) - Strike);
    for i = treeLength-1:-1:1
        for j=1:i
            OptPrice(j,i) = (OptPrice(j,i+1)*p_up + OptPrice(j+1,i+1)*p_down)/(1+rate);
        end
    end
    f = OptPrice(1,1);
end
```

13.4 European Put

```
function f = put(BinTree,Strike,rate,p_up,p_down)

    treeLength = length(BinTree);
    OptPrice(:,treeLength) = max(0,Strike - BinTree(:,treeLength));
    for i = treeLength-1:-1:1
        for j=1:i
            OptPrice(j,i) = (OptPrice(j,i+1)*p_up + OptPrice(j+1,i+1)*p_down)/(1+rate);
        end
    end
    f = OptPrice(1,1);
end
```

13.5 European Call with Black-Scholes

```
function f = bs_call(price,strike,int_rate,expiry,vol)
    lso = (log(price/strike)+(int_rate+(vol.*vol)/2)*expiry);
    d1 = lso/(vol*sqrt(expiry));
    d2 = d1 - vol*sqrt(expiry);
    f = price*normcdf(d1)-strike*exp(-int_rate*expiry)*normcdf(d2);
```

```
end
```

13.6 European Put with Black-Scholes

```
function f = bs_put(price,strike,int_rate,expiry,vol)
    lso = (log(price/strike)+(int_rate+(vol.*vol)/2)*
        expiry);
    d1 = lso/(vol*sqrt(expiry));
    d2 = d1 - vol*sqrt(expiry);
    f = price*normcdf(d1)-strike*exp(-int_rate*expiry)*
        normcdf(d2)-price+strike*exp(-int_rate*expiry);
end
```

13.7 American Call

```
function [f,t] = call_american(BinTree,Strike,rate,p_up,
    p_down)
    early_exercise = false;
    treeLength = length(BinTree);
    OptPrice(:,treeLength) = max(0,BinTree(:,treeLength)
        - Strike);
    for i = treeLength-1:-1:1
        for j=1:i
            if (BinTree(j,i) - Strike) >( OptPrice(j,i+1)
                *p_up + OptPrice(j+1,i+1)*p_down)/(1+rate)
                early_exercise = true;
            end
            OptPrice(j,i) = max((BinTree(j,i) - Strike),(
                OptPrice(j,i+1)*p_up + OptPrice(j+1,i+1)*
                p_down)/(1+rate));
        end
    end
    f = OptPrice(1,1);
    t = early_exercise;
end
```

13.8 American Put

```
function [f,t] = put_american(BinTree,Strike,rate,p_up,
    p_down)
    early_exercise = false;
    treeLength = length(BinTree);
    OptPrice(:,treeLength) = max(0,Strike - BinTree(:,
        treeLength));
    for i = treeLength-1:-1:1
        for j=1:i
```

```
            if (Strike - BinTree(j,i)) >( OptPrice(j,i+1)*p_up + OptPrice(j+1,i+1)*p_down)/(1+rate)
                early_exercise = true;
            end
            OptPrice(j,i) = max((Strike - BinTree(j,i)),(OptPrice(j,i+1)*p_up + OptPrice(j+1,i+1)*p_down)/(1+rate));
        end
    end
    f = OptPrice(1,1);
    t = early_exercise;
end
```

13.9 European Exotic Option

```
function f = european_exotic(BinTree,Strike,rate,p_up,p_down)

    treeLength = length(BinTree);
    OptPrice(:,treeLength) = power((BinTree(:,treeLength) - Strike),2);
    for i = treeLength-1:-1:1
        for j=1:i
            OptPrice(j,i) = (OptPrice(j,i+1)*p_up + OptPrice(j+1,i+1)*p_down)/(1+rate);
        end
    end
    f = OptPrice(1,1);
end
```

13.10 Binomial Tree with Independent Random Paths

```
function [BinTree,rate_matrix,p_up_matrix,p_down_matrix] = tree_random(last_price,std_sp_500_returns,NumPeriods,rate_random_matrix,option_maturity)
    u = exp(std_sp_500_returns*sqrt(3/NumPeriods));
    d = 1/u;

    BinTree = zeros(NumPeriods+1);

    %%build tree by hand
    for i = 1:NumPeriods+1
        for j=1:i
            BinTree(j,i) = last_price * power(u,i-j) * power(d,j-1);
```

```
        end
    end

    rate_matrix = exp(rate_random_matrix*option_maturity/NumPeriods)-1;

    p_up_matrix = (1+rate_matrix-d)/(u-d);
    p_down_matrix = 1-p_up_matrix;
end
```

13.11 European Call with Independent Random Paths

```
function f = call_random(BinTree,Strike,rate_matrix,p_up_matrix,p_down_matrix)

    treeLength = length(BinTree);
    OptPrice(:,treeLength) = max(0,BinTree(:,treeLength) - Strike);
    for i = treeLength-1:-1:1
        for j=1:i
            OptPrice(j,i) = (OptPrice(j,i+1)*p_up_matrix(i) + OptPrice(j+1,i+1)*p_down_matrix(i))/(1+rate_matrix(i));
        end
    end
    f = OptPrice(1,1);
end
```

13.12 European Put with Independent Random Paths

```
function f = put_random(BinTree,Strike,rate_matrix,p_up_matrix,p_down_matrix)

    treeLength = length(BinTree);
    OptPrice(:,treeLength) = max(0,Strike - BinTree(:,treeLength));
    for i = treeLength-1:-1:1
        for j=1:i
            OptPrice(j,i) = (OptPrice(j,i+1)*p_up_matrix(i) + OptPrice(j+1,i+1)*p_down_matrix(i))/(1+rate_matrix(i));
        end
    end
    f = OptPrice(1,1);
end
```