MA 374 – Financial Engineering II

Lab 1

Report

Name: Vibhanshu
Roll No.:120123049

Write a program, using the binomial pricing algorithm, to determine the price of an European call and an European put option (in the binomial model framework) with the following data :
$S(0) = 9$; $K = 10$; $T = 3$; $r = 0.06$; $\sigma = 0.3$.
Take u = e
σ√Δt+(r− 12σ2)Δt
and d = e−σ√Δt+(r− 12σ2)Δt, where $\Delta t = T/M$,
with M being the number of subintervals in the time interval [0, T]. Use the continuous compounding convention in your calculations (i.e., both in p˜ and in the pricing formula).

1. Run your program for M = 1, 5, 10, 20, 50, 100, 200, 400 to get the initial option prices and tabulate them.

2. How do the values of options at time t = 0 compare for various values of M? Compute and plot graphs (of the initial option prices) varying M in steps of 1 and in steps of 5. What do you observe about the convergence of option prices?

3. Tabulate the values of the options at t = 0, 0.30, 0.75, 1.50, 2.70 for the case M = 20.

Note that your program should check for the no-arbitrage condition of the model before proceeding to compute the prices.

Question 1:

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cmath>
using namespace std;

float S0=9;
float K=10;
float T=3;
float r=0.06;
float sigma=0.3;
int M;

float max(float a,float b)
{
	if( a>b ) return a;
	else return b;
}

float u(float delta_t)
{
	return exp(sigma*sqrt(delta_t)+(r-0.5*sigma*sigma)*delta_t);
}


float d(float delta_t)
{
	return exp(-sigma*sqrt(delta_t)+(r-0.5*sigma*sigma)*delta_t);
}

float discount_rate(float t)
{
	return exp(r*t);
}

float p(float delta_t)
{
	return ( exp(r*delta_t) - d(delta_t) ) / ( u(delta_t) -d(delta_t) ) ;
}

float q(float delta_t)
{
	return ( u(delta_t) - exp(r*delta_t) ) / ( u(delta_t) - d(delta_t) );
```

```
}

// This is for the call option(recursive)
float get_C(float t,float S,float delta_t)    // returns the price of the option at time t if
the price os stock is S )
{
	if(t>=T)
	{
		if(S>=K) return (S-K);
		else return 0;
	}
	else
	{
		float t1=get_C(t+delta_t , S*u(delta_t) , delta_t);
		float t2=get_C(t+delta_t , S*d(delta_t) , delta_t);
		return  ( p(delta_t)*t1 +q(delta_t)*t2)/discount_rate(delta_t) ;
	}
}

// This is for the put option(recursive)
float get_P(float t,float S,float delta_t)    // returns the price of the option at time t if
the price os stock is S )
{
	if(t>=T)
	{
		if(S<=K) return (K-S);
		else return 0;
	}
	else
	{
		float t1=get_P(t+delta_t , S*u(delta_t) , delta_t);
		float t2=get_P(t+delta_t , S*d(delta_t) , delta_t);
		return  ( p(delta_t)*t1 +q(delta_t)*t2)/discount_rate(delta_t) ;
	}
}

// This is the fast version of the algorithm CALL
float get_C_fast(float t,float S,float delta_t)
{
	float U=u(delta_t);
	float D=d(delta_t);
	float P=p(delta_t);
	float Q=p(delta_t);
	float R=discount_rate(delta_t);
	float* SS=new float[M+2];
```

```
	float* value=new float[M+2];
	int i,j,k;
	for(i=0;i<=M;i++)
		SS[i]=S0*(pow(U,(float)(M-i)))*(pow(D,(float)i));
	for(i=0;i<=M;i++)
		value[i]=max(SS[i]-K,0);
	for(j=M; j>=0; j--)
	{
		for(k=0; k<j; k++)
			value[k]= (P*value[k+1]+Q*value[k])/R;
		if( abs(t-j*delta_t)<0.1 )
		{
			for(k=0;k<j;k++ )
				cout<<value[k]<<"\t";
			cout<<endl;
		}
	}
	return value[0];
}

// This is the fast version of the algorithm PUT
float get_P_fast(float t,float S,float delta_t)
{
	float U=u(delta_t);
	float D=d(delta_t);
	float P=p(delta_t);
	float Q=p(delta_t);
	float R=discount_rate(delta_t);
	float* SS=new float[M+2];
	float* value=new float[M+2];
	int i,j,k;
	for(i=0;i<=M;i++)
		SS[i]=S0*(pow(U,(float)(M-i)))*(pow(D,(float)i));
	for(i=0;i<=M;i++)
		value[i]=max(K-SS[i],0);
	for(j=M; j>=0; j--)
	{
		for(k=0; k<j; k++)
			value[k]= (P*value[k+1]+Q*value[k])/R;
		if( abs(t-j*delta_t)<0.01 )
		{
			for(k=0;k<j;k++)
				cout<<value[k]<<"\t";
			cout<<endl;
		}
```

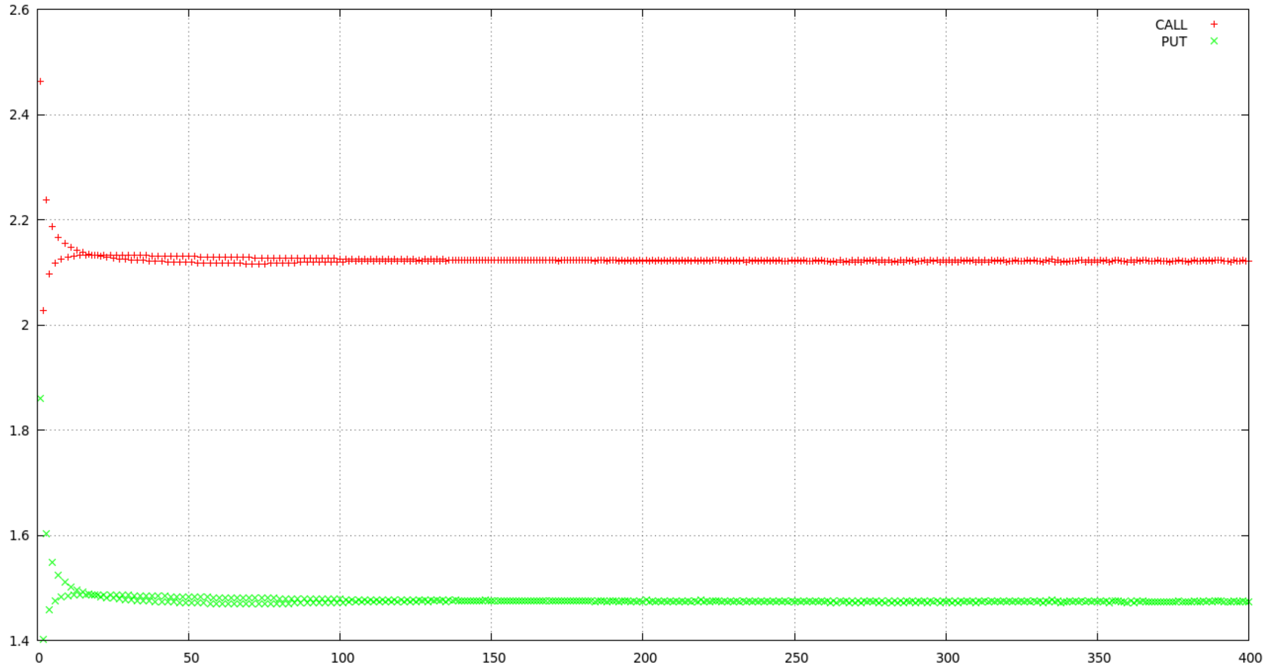
```
	}
	return value[0];
}


int main()
{

	cout<<"Insert M :";
	cin>>M;
	float delta_t=T/M;
//	cout<<"ANS CALL :"<<get_C(0,S0,delta_t)<<endl;
//	cout<<"AND PUT  :"<<get_P(0,S0,delta_t)<<endl;
	cout<<"ANS CALL  :" << get_C_fast(0,S0,delta_t) <<endl;
	cout<<"ANS PUT  :" << get_P_fast(0,S0,delta_t) <<endl;
	return 0;
}
```

OBSERVATIONS:

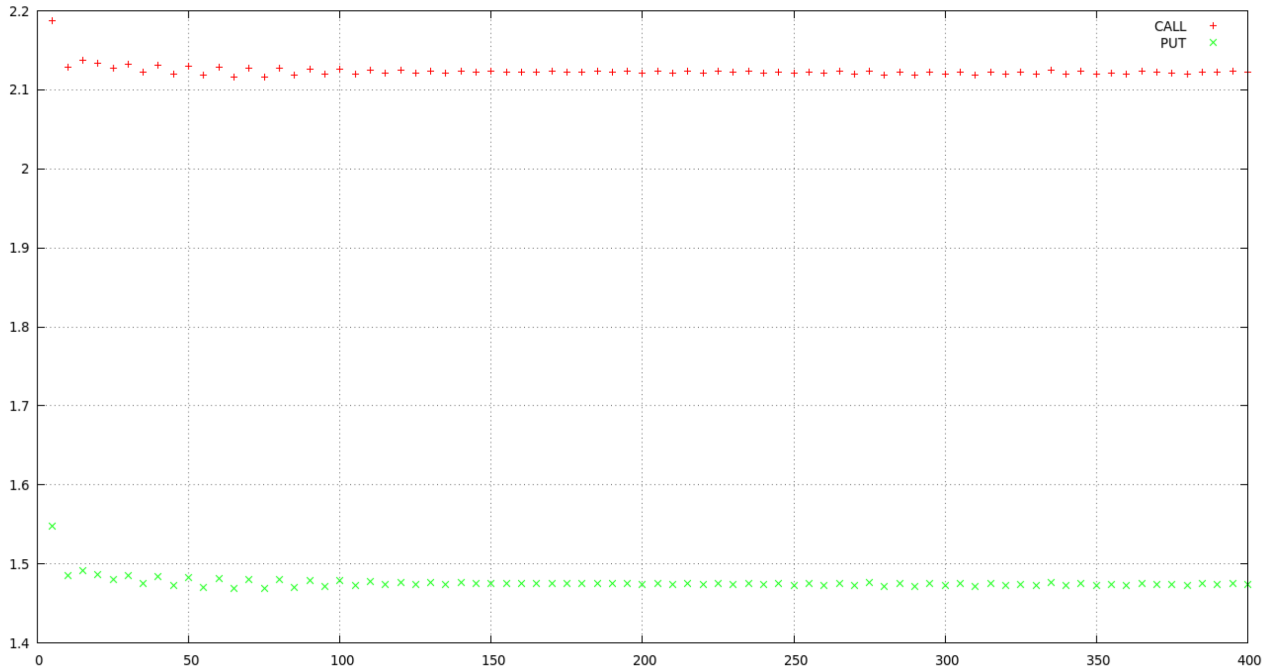
Value of M	Price of European call option	Price of European put option
1	2.46337	1.85986
5	2.18776	1.54789
10	2.12916	1.48491
20	2.13337	1.48711
50	2.1296	1.4827
100	2.12585	1.47834
200	2.12142	1.47395
400	2.1219	1.47411

Question 2:

<u>Values of options at time t=0 for various M in step size 1</u>



<u>Values of options at time t=0 for various M in step size 5</u>



Observations :

a) As value of M increases , the call option price converges to a value of 2.1219 (at M=400)

b) As value of M increases , the put option price converges to a value of 1.47411 (at M=400)

c) It can be inferred from the graphs that the convergence becomes more accurate as the size of the increment increases from 1 to 5.

Question 3:

Output for M=20
The price of the European Call Option at various times are:

At time t=2.7
66.0524 50.3196 37.8491 27.9644 20.1293 13.9188 8.99614 5.0942 2.14521 0.460757 0 0 0 0 0 0 0 0 0

At time t=1.5
20.2346 14.1876 9.38457 5.79673 3.06588 1.38392 0.492667 0.125645 0.0234565 0.00365 0

At time t=0.75
7.73367 4.67013 2.54098 1.20912 0.486801 0.158234

At time t=0.3
3.71979 1.99601 0.947459

At time t=0
2.1294

The price of the European Put Option at various times are:

At time t=2.7
0 0 0 0 0 0 0 0 0 0.143863 0.919845 0.923867 2.3867 3.92534 5.15456 6.12229 6.88936 7.49738 7.97932 8.36133 8.66412

At time t=1.5
0.000523475 0.00750139 0.048979 0.20606 0.599267 1.32254 2.33508 3.48861 4.57792 5.50945 6.26007

At time t=0.75
0.206602 0.515525 1.05845 1.84568 2.80386 3.80769

At time t=0.3
0.824824 1.44768 2.28865

At time t=0
1.48345