Computational finance assignment 1

Kees Til (10385827)
Maarten Muller (10430148)

February 2017

1 Part 1

1.1 Question 1

The analytical value of the price f of a call option is given by the Black-Scholes Merton formulas:

$$f = S_0 N(d_1) - K e^{rT} N(d_2), \tag{1}$$

$$d_1 = \frac{log(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}, \tag{2}$$

$$d_2 = d_1 - \sigma\sqrt{T}. \tag{3}$$

In the case of $K = 99$, $S_0 = 100$, $\sigma = 0.2$, $r = 0.06$, and $T = 1$, we get an analytical solution of $f = 11.5443$.

We have written a Matlab code, which can be seen in the appendix, to simulate the bionimal tree. We first calculated all the stock prices, $S_{i,j} = S_0 u^j d^{i-j}$. With $u = e^{\sigma\sqrt{\delta t}}$ and $d = e^{-\sigma\sqrt{\delta t}}$. The value of the call option at the expire date can now be computed and is $MAX(0, S_{N,j} - K)$. Using the back-ward induction scheme of equation 4, the option price can be computed.

$$f_{i,j} = e^{-r\delta t}(p f_{i+1,j+1} + (1-p) f_{i+1,j}), p = \frac{e^{r\delta t} - d}{u - d} \tag{4}$$

With the same values as for the analytical value we now get $f = 11.5464$. The difference with the analytical value is in the order of 10^{-3}.

With a different volatality the option price will be different. In the figures below we can see the call option price for different volatalities, both the analytical solution and the solutions according to our model. Here we see, with the exception of low volatility a linear connection. When the volatility rises or decreases, the price of an option will rise and decrease respectively.

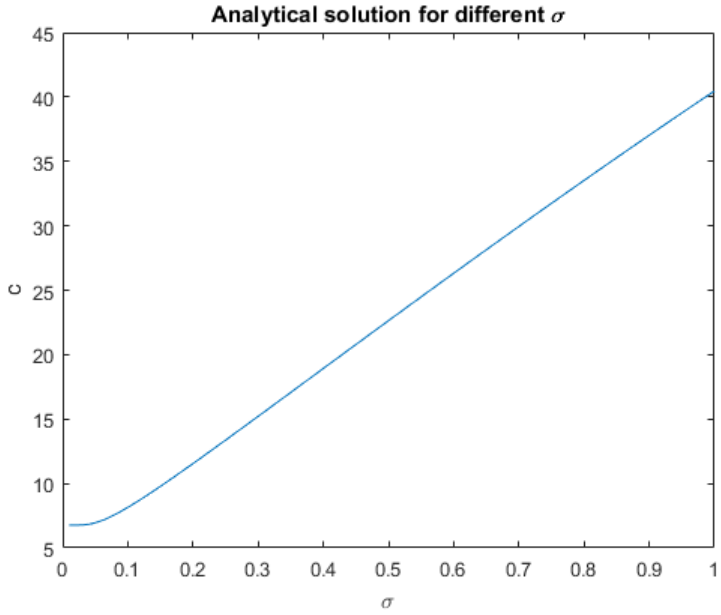

Figure 1:

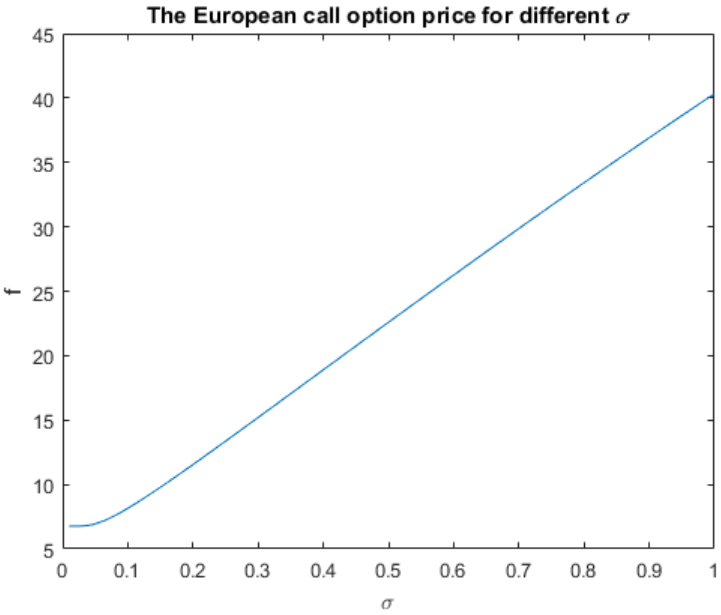

Figure 2:

1.2 Question 2

The computed call option price depends on the number of steps of the binomial tree. To study the convergence of the method we have computed the call option price for different number of steps. This can be seen in the figure 3. With a low number of steps the call option price differs a bit, but we can see that if the steps increase, the call option price converges to the right value.

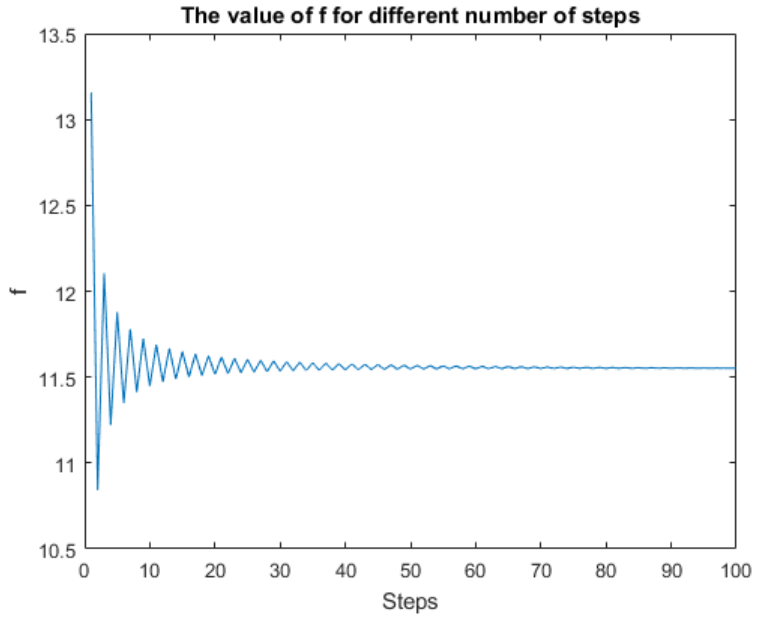

Figure 3:

In figure 4 we can see the computing time of the call option price for different number of steps. The time clearly increases exponentially by the number of steps. From figure 3 we know that after 100 steps we have a good value for the call option price and we can see that the computing time is quite low then.

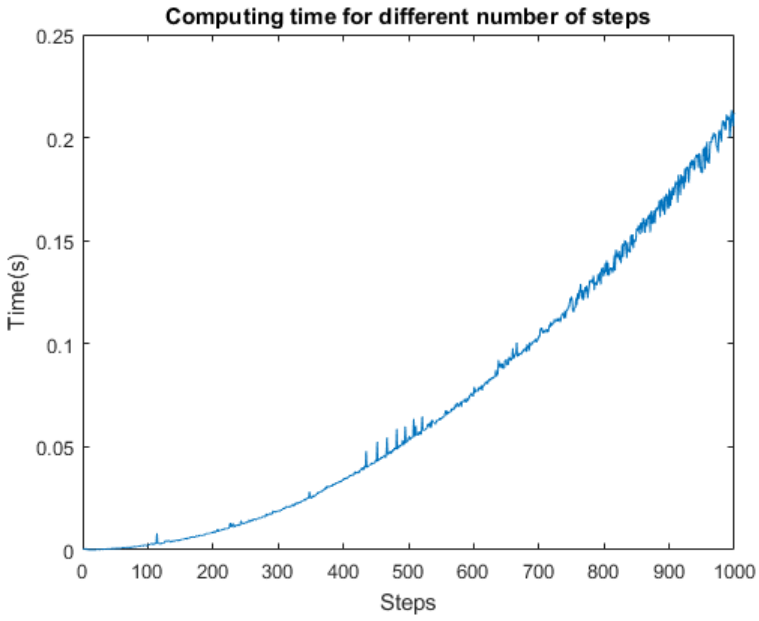

Figure 4:

1.3 Question 3

The put option price, P, in a binomial tree at the expire date is given by $MAX(0, K - S_{N,j})$. Again we can use equation 4 to compute the put option price. The computed value we got was P = 4.7811.

To check if this is right, we can compare this with the put-call parity relationship. This is given by

$$C - P = S - Ke^{-rT} \tag{5}$$

Filling in all the values gives us: 6.7654 = 6.7653. Because this difference is really small, we state that our computed call and put option prices satisfy the put-call parity relationship.

1.4 Question 4

The analytical value of the hedge parameter Δ is given by $N(d_1)$. In this case $\Delta = 0.6737$. For other values of σ, Δ can be seen in figure 5

In our model we compute Δ by using: $\Delta = \frac{f_{1,1} - f_{1,2}}{S_{1,1} - S_{1,2}}$. This gives us $\Delta = 0.6726$. This is a relative small difference from the analytical value. In figure 6 we see the computed Δ for different values of σ.

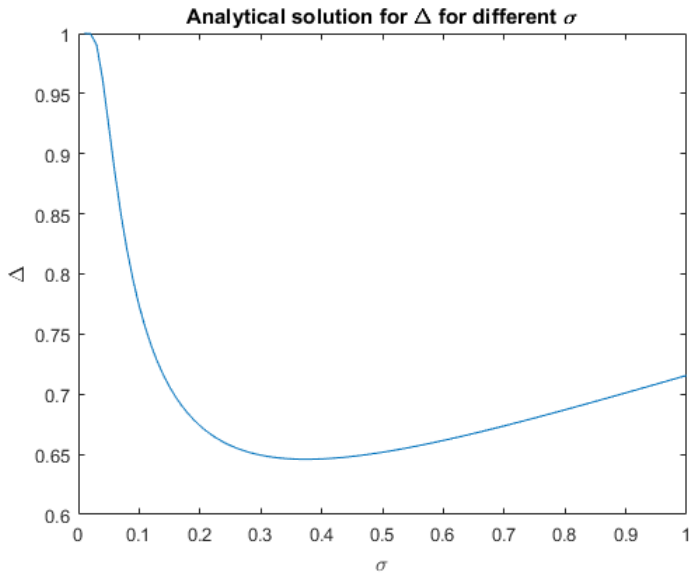

Figure 5:

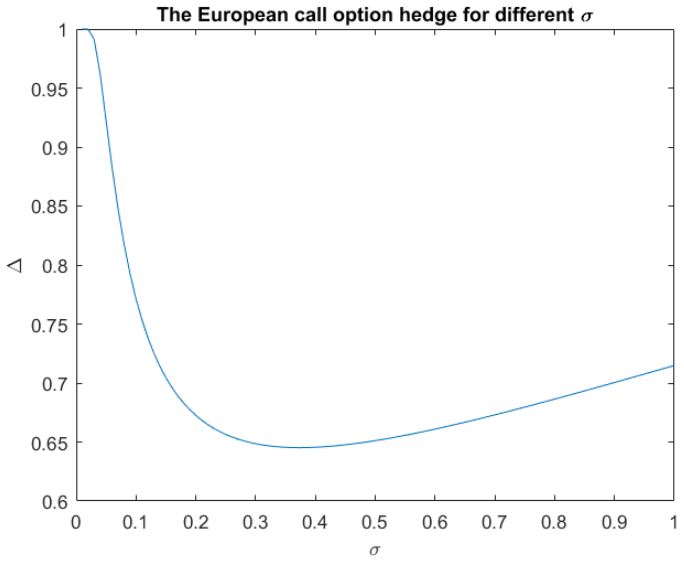

Figure 6:

1.5 Question 5

For Amercian options, one needs to handle early exercise opportunities. At their expire date, the value of the options are same as for European options.

Now instead of using equation 4, we have to use $MAX((S_{i,j} - K)^+, f_{i,j})$ or $MAX((K - S_{i,j})^+, f_{i,j})$ for respectively a call or a put option. For the corresponding parameters, we get $C = 13.4090$ and $P = 15.7465$. The different values of C and P for different values of σ can be seen in figure 7 and 8.

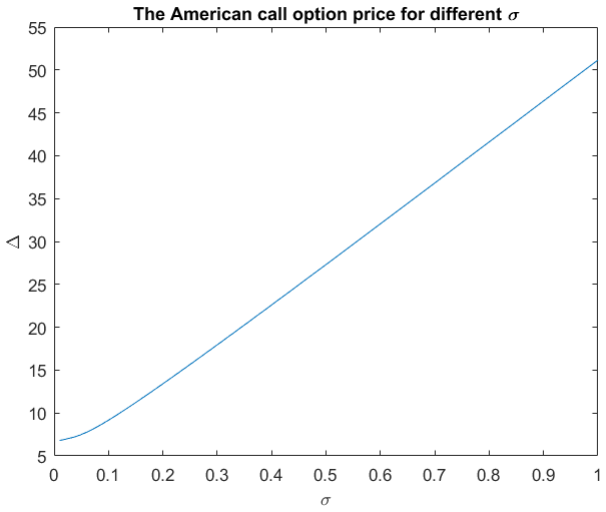

Figure 7:

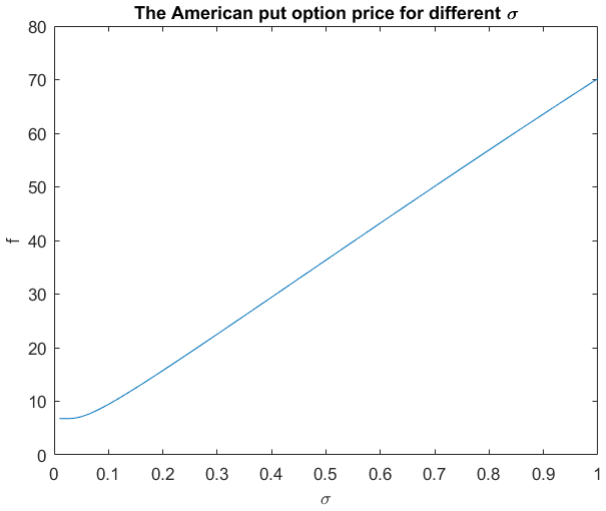

Figure 8:

2 Part 2

2.1 Hedging simulation

We wrote a code in Matlab that performed a hedging simulation. We considered a short position in a European call option on a non-dividend-playing stock with a maturity of one year. We used equation 6 to simulate the dynamics of the stock S. The option price is calculated with the Black-Scholes model in equation 1 and $\Delta = N(d_1)$.

$$dS = rSdt + \sigma SdZ \tag{6}$$

We calculate Δ at the beginning. This is the number of stocks you have to buy. At the next timestep, you have to adjust your hedge. Δ will be different and this difference have to be bought or sold. Until at the expire date you have the whole stock or nothing, depending on whether your client is going to buy the stock or not. Because of this you get a debt with an interest rate of r. However, you also have made money by selling the call option. This also has an interest rate of r. At the expire date you also get K euro if you sell the stock. If you hedge correctly, there should not be any profit or loss.

For this hedging simulation we used the following parameters: $S_0 = 100$, $K = 99$, $\sigma = 0.2$, $r = 0.06$. We varied the frequency of the hedge adjustment from daily to weekly. We did 1000 simulations for both and the results can be seen in a histogram in figure 9 and figure 10.

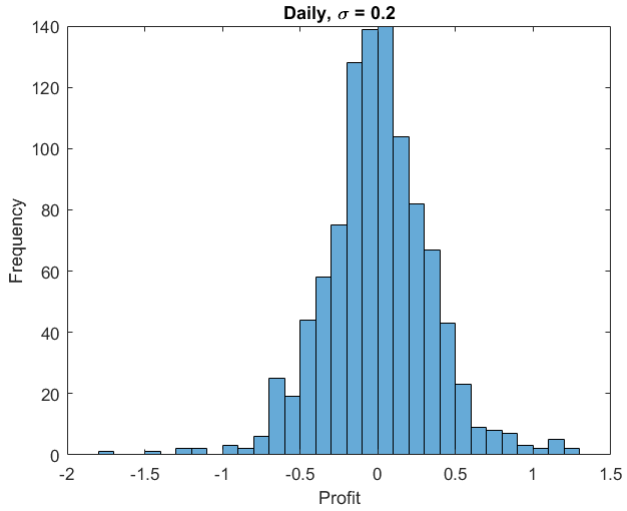

Figure 9:

The simulation of figure 9 with a daily hedge adjustment has a mean of 0.0004 and a standard deviation of 0.3433. The simulation of figure 10 with

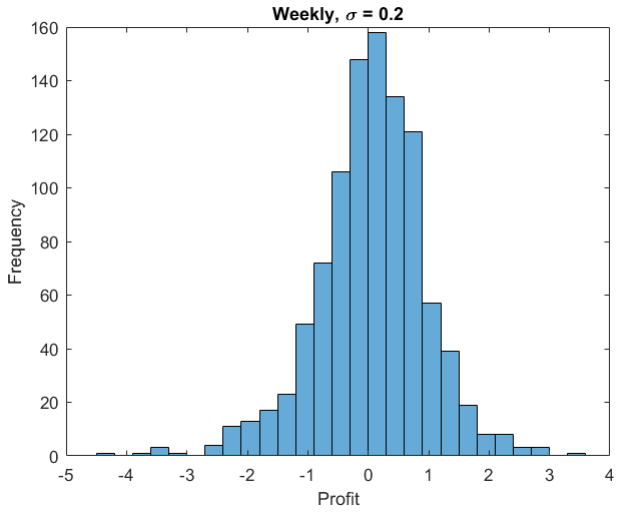

Figure 10:

a weekly hedge adjustment has a mean of 0.0563 and a standard deviation of 0.9075. Both simulation have a mean close to zero like you would expect. However, the simulation of figure 10 has a much larger standard deviation. So from this we can conclude that the more hedge adjustments you do, the better i is

Now we have also simulated two cases where the volatility in the stock price process(σ_1) is not matching the volatility used in the option valuation(σ_2). We used a weekly hedge adjustment here. The first case we have $\sigma_1 = 0.01$ and $\sigma_2 = 0.2$ and for the second case we have $\sigma_1 = 0.5$ and $\sigma_2 = 0.2$. These are somewhat extreme cases but we clearly can see the consequences in the histograms of figure 11 and figure 12.

In figure 11 we can see the case that you overestimate the volatility. Here the mean is 5.0753 with a standard deviation of 0.2776. So you would make a profit. However, because you overestimated the volatility, it is unlikely you would sell the call option because it is too expensive.

In figure 12 we can see the case that you underestimate the volatility. Here the mean is -11.4971 with a standard deviation of 6.7260. We can see that if you underestimate the volatility, you would make a loss. This is because you sell your call option for not enough money.

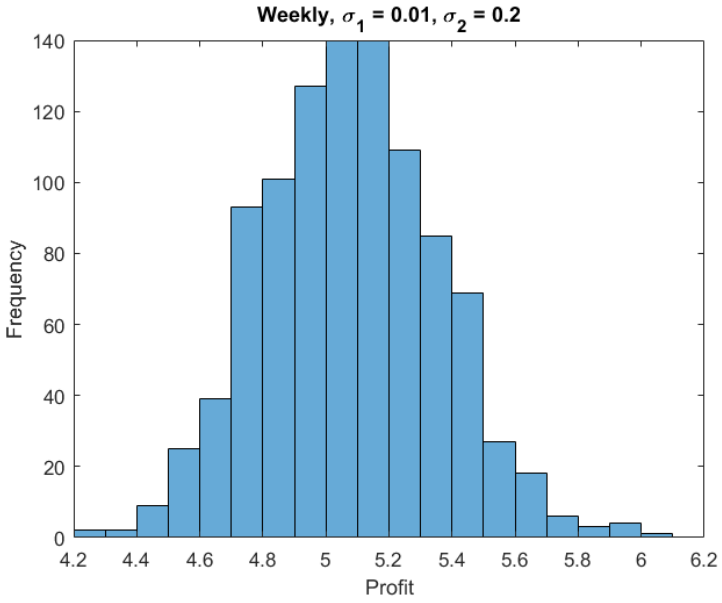

Figure 11:

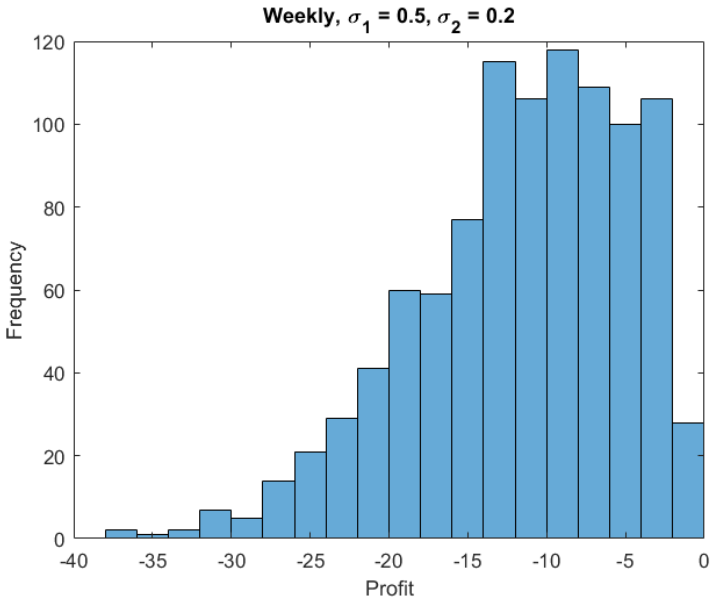

Figure 12:

3 Appendix

3.1 Code Part 1

3.1.1 European

```
function tree = bintree(S_0,K,sigma,r,steps)
% Here we define variables which can be found in the slide
dt = 1/steps;
u = exp(sigma*sqrt(dt));
d = exp(-sigma*sqrt(dt));
a = exp(r*dt);
p = (a-d)/(u-d);

% We make the tree and put the results in the matrix
Matrix = zeros(steps+1);
for i = 1:steps+1;
    for j = 1:i;
        Matrix(i,j) = S_0*u^(j-1)*d^(i-j);
    end
end
Matrix;

% Take the last row
last = Matrix(steps+1,:);

% Here under we use the binomial method algorithm as in Seydals book. Uncomment
% where one has a put or a call option
%Call option
%payoff = max(Matrix(steps+1,:) - K,0);
%Put option
%payoff = max(K-Matrix(steps+1,:),0);
Matrix2 = zeros(steps+1);
Matrix2(steps+1,:) = payoff;

%back-ward induction scheme
for i = steps+1:-1:2
    for j = i:-1:2
        Matrix2(i-1,j-1) = exp(-r*dt)*(p*Matrix2(i,j)+(1-p)*Matrix2(i,j-1));
    end
end
Matrix2(1,1)
```

3.1.2 American

```
function Atree = Abintree(S_0,K,sigma,r,steps)
% Here we define variables which can be found in the slide
dt = 1/steps;
u = exp(sigma*sqrt(dt));
d = exp(-sigma*sqrt(dt));
a = exp(r*dt);
p = (a-d)/(u-d);

% We make the tree and put the results in the matrix
Matrix = zeros(steps+1);
for i = 1:steps+1;
    for j = 1:i;
        Matrix(i,j) = S_0*u^(j-1)*d^(i-j);
    end
end
Matrix;

%Call option
%payoff = max(Matrix(steps+1,:) - K,0);
%Put option
%payoff = max(K-Matrix(steps+1,:),0);
Matrix2 = zeros(steps+1);
Matrix2(steps+1,:) = payoff;

%back-ward induction scheme
for i = steps+1:-1:2
    for j = i:-1:2
        %Here is the big difference with European where we change the payoff in
        %Call
        %payoff = max(Matrix(i,j) - K,0);
        %Put
        %payoff = max(K-Matrix(i,j),0);
        Matrix2(i-1,j-1) = max(payoff,exp(-r*dt)*(p*Matrix2(i,j)+(1-p)*Matrix2(i
    end
end
Matrix2
hedge = (Matrix2(2,1)-Matrix2(2,2))/(Matrix(2,1)-Matrix(2,2));
```

3.2 Code Part 2

```
%This function returns delta_i
function  Delta = BS(S,K,sigma,r,i,steps)
t = i/steps;
```

```
d1 = (log(S/K)+(r + (sigma)^2/2)*(1-t))/((sigma)*sqrt(1-t));
d2 = d1 - (sigma)*sqrt(1-t);
normcdf(d1);
analytic = S*normcdf(d1)- K*exp(-r*(1-t))*normcdf(d2);
Delta = normcdf(d1);
```

```
%This function simply returns the analytic value which is delta_0
function  Delta2 = BS2(S,K,sigma,r)
t = 0;
d1 = (log(S/K)+(r + (sigma)^2/2))/(sigma);
d2 = d1 - (sigma);
normcdf(d1);
analytic = S*normcdf(d1)- K*exp(-r)*normcdf(d2);
Delta2 = analytic
```

```
%This function simulates the hedging process
function sim = H_sim(S,r,sigma1,sigma2,steps,K)
analytic_val = BS2(S,K,sigma2,r);
dt = 1/steps;
hedge_v = zeros(1,steps);
debt = BS(S,K,sigma2,r,0,steps)*S;
a = BS(S,K,sigma2,r,0,steps);
for i = 1:steps;
    dz = normrnd(0,1)*sqrt(dt);
    dS = r*S*dt + sigma1*S*dz;
    S = S+dS;
    hedge_v(i) = BS(S,K,sigma2,r,i,steps);
    debt = (debt*exp(r*dt) + (hedge_v(i)-a)*S);
    a = hedge_v(i);
end
sim = S;
hedge_v;
debt;
profit = analytic_val*exp(r) - debt - max(S-K,0) + a*S;
sim = profit
```

```
%Just a simple histogram of the H_sim
function Hist = no_risk(S,r,sigma1,sigma2,steps,K)
data = zeros(1,steps);
for i = 1:steps
    data(i) = H_sim(S,r,sigma1,sigma2,steps,K);
end
histogram(data);
```