Computational finance, take-home exam 1: practical part

Karl Albin Henriksson

October 2022

Exercise 4

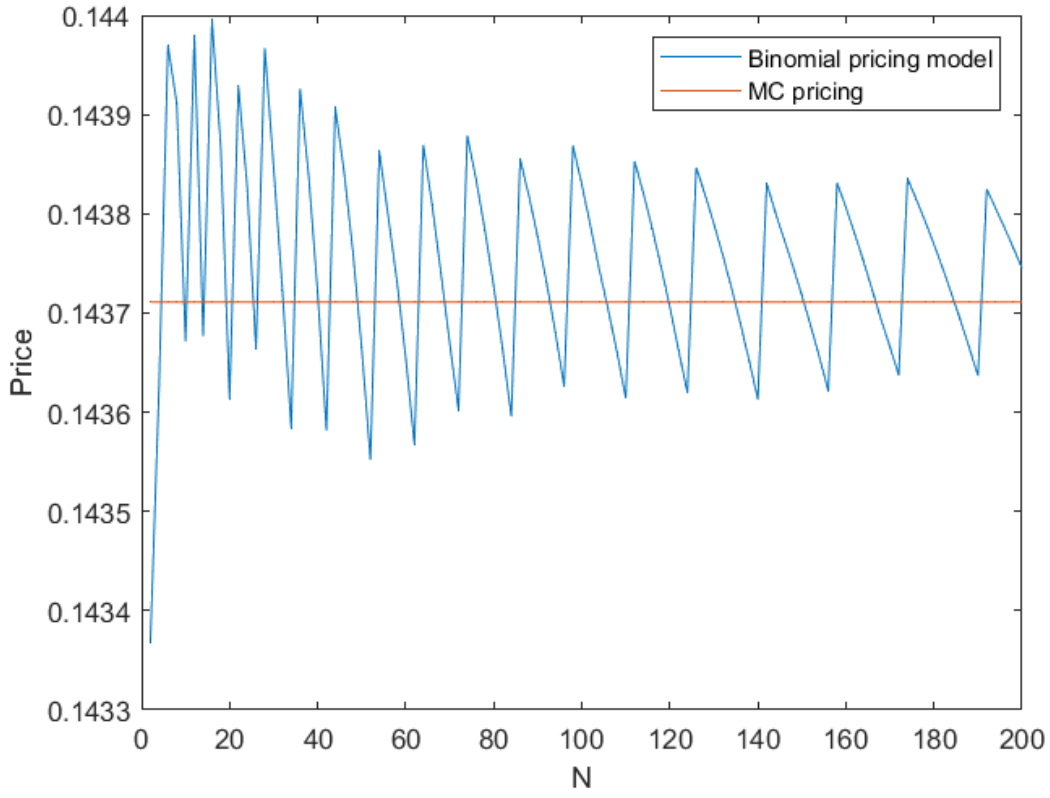

Figure 1

BinomialpriceBarrierUODM

```
1 function option_price_bin=BinomialpriceBarrierUODM(r,d,u,N,T,s,K,b)
2 r=r*T/N;
3 q_u=(1+r-d)/(u-d);
```

```
4 q_d=1-q_u;
5 prices=zeros(N+1,1);
6
7
8 for i = 0:N
9     S_T = s*u^i*d^(N-i);
10    if S_T < b
11        prices(i+1) = max(S_T-K,0);
12    else
13        prices(i+1) = 0;
14    end
15 end
16
17 for k = (N-1):-1:N/2
18    price_k = zeros(k+1,1);
19    for i = 1:k+1
20        price_k(i) =(q_u*prices(i+1) + q_d*prices(i))/(1+r);
21    end
22    prices = price_k;
23 end
24 prices;
25 for i = 0:N/2
26    price_T_2 =d^(N-i)*u^i*s;
27
28    if price_T_2 > b
29        prices(i+1) = 0;
30    end
31 end
32
33 for k = (N/2-1):-1:0
34    price_k = zeros(k+1,1);
35    for i = 1:k+1
36        price_k(i) = (q_u*prices(i+1) + q_d*prices(i))/(1+r);
37    end
38    prices = price_k;
39 end
40 option_price_bin=prices;
41 end
```

MCpriceBarrierUODM

```
1 function option_price_BS = MCpriceBarrierUOD(r,sigma,N_time,N_sim,T,s
      ,K,b)
2 dt = T/N_time;
3
4 S = s*ones(N_sim,1);
5
6 for i = 1:N_time/2
7     Z = randn(N_sim,1);
8     S = S.*(1 + r*dt + Z*sigma*sqrt(dt));
```

```
end

T_2 = S < b;

for i = (N_time/2+1):N_time
    Z = randn(N_sim,1);
    S = S.*(1 + r*dt + Z*sigma*sqrt(dt));
end


T_1 = S < b;
payoff=0;
for i=1:N_sim

    if T_2(i) && T_1(i)
    payoff =payoff+ max(S(i)-K,0);
    end
end

option_price_BS = exp(-r*T)*payoff/N_sim;
end
```

Plotting

```
close all
r=.1;
S=1;
T=.5;
K=.9;
sigma=.1;
b=1.3;
N_time=100;
N_sim=10^6;
MC_price=MCpriceBarrierUOD(r,sigma,N_time,N_sim,T,S,K,b);
Bin_price=zeros(100,1);

for i=1:100
    u=1+r*T/(i*2)+sigma*sqrt(T/(i*2));
    d=u-2*sigma*sqrt(T/(i*2));
    Bin_price(i)=BinomialpriceBarrierUOD(r,d,u,i*2,T,S,K,b);
end
x=2*(1:100);

plot(x,[Bin_price,MC_price*ones(100,1)])
ylabel('Price')
xlabel('N')
legend('Binomial pricing model','MC pricing')

function option_price_bin=BinomialpriceBarrierUOD(r,d,u,N,T,s,K,b)
r=r*T/N;
```

```
27 q_u=(1+r-d)/(u-d);
28 q_d=1-q_u;
29 prices=zeros(N+1,1);
30
31
32 for i = 0:N
33     S_T = s*u^i*d^(N-i);
34     if S_T < b
35         prices(i+1) = max(S_T-K,0);
36     else
37         prices(i+1) = 0;
38     end
39 end
40
41 for k = (N-1):-1:N/2
42     price_k = zeros(k+1,1);
43     for i = 1:k+1
44         price_k(i) =(q_u*prices(i+1) + q_d*prices(i))/(1+r);
45     end
46     prices = price_k;
47 end
48 prices;
49 for i = 0:N/2
50     price_T_2 =d^(N-i)*u^i*s;
51
52     if price_T_2 > b
53         prices(i+1) = 0;
54     end
55 end
56
57 for k = (N/2-1):-1:0
58     price_k = zeros(k+1,1);
59     for i = 1:k+1
60         price_k(i) = (q_u*prices(i+1) + q_d*prices(i))/(1+r);
61     end
62     prices = price_k;
63 end
64 option_price_bin=prices;
65 end
66
67
68 function option_price_BS = MCpriceBarrierUOD(r,sigma,N_time,N_sim,T,s
       ,K,b)
69 dt = T/N_time;
70
71 S = s*ones(N_sim,1);
72
73 for i = 1:N_time/2
74     Z = randn(N_sim,1);
```

```
75      S = S.*(1 + r*dt + Z*sigma*sqrt(dt));
76  end
77
78  T_2 = S < b;
79
80  for i = (N_time/2+1):N_time
81      Z = randn(N_sim,1);
82      S = S.*(1 + r*dt + Z*sigma*sqrt(dt));
83  end
84
85
86  T_1 = S < b;
87  payoff=0;
88  for i=1:N_sim
89
90      if T_2(i) && T_1(i)
91      payoff =payoff+ max(S(i)-K,0);
92      end
93  end
94
95  option_price_BS = exp(-r*T)*payoff/N_sim;
96  end
```