Homework 6 Advanced Derivatives

Albin Henriksson

October 2022

Exercise 1

In this exercise we will use the Andreasen-Huge algorithm to complete the implied volatility matrix. To start off, we calculate the call option prices at the data points where the IV exists. We then use these prices to calibrate a piece-wise constant volatility function. The piece-wise constant function is such that

$$\sum_{i=1}^{J}(C_{AH-model}(K_i, T_{j+1}) - C_{data}(K_i, T_j)^2$$

is minimized. Here, J is the number of data points at each maturity. Once the piece-wise constant local volatility function is determined, we use it to calculate the call option prices at each maturity and strike. Finally, we invert the BS call option formula at every point to obtain the implied volatility surface. In order to obtain the implied volatility at $T = 1$ and $T = 1.5$, we use the AH-algorithm again using the already calibrated local volatility to obtain call option prices. We then invert the BS call option formula as before to obtain the implied volatility.

Due to the discretization, the surfaces differ a bit based on what value of the stock price is used, however the qualitative behaviour of the surface remains the same. In the plots below, $S_0 = 1000$ is used. Figure 1 shows the implied volatility surface for the given strikes and maturities. Figure 2 shows the implied volatility interpolated at $T = 1$ and $T = 1.5$.

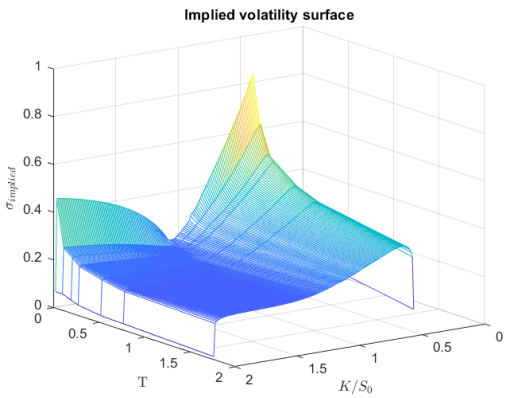

Figure 1

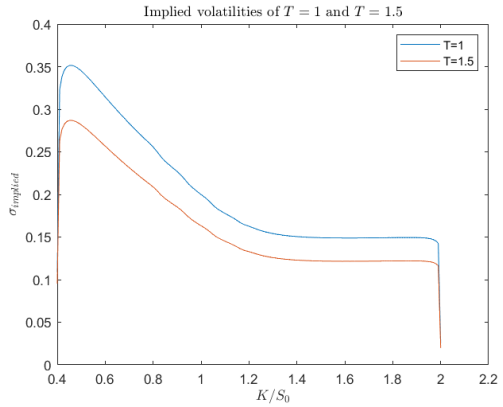

Figure 2

Code

```
clear all
close all

df=xlsread("SX5E_Impliedvols.xlsx");
strikes=df(2:end,1);
maturities=df(1,2:end);
data_vols=df(2:end,2:end);
S_0=1000;
strikes=S_0*strikes;

%% Create call option matrix where volatility data exists

data_prices=zeros(size(data_vols));
for j=1:length(maturities)
    for i=1:length(strikes)
        if data_vols(i,j)>0
            data_prices(i,j)=BSoptionprice(S_0,strikes(i,1),data_vols
                (i,j),0,maturities(1,j),0);
        end
    end
end

%% Create call option prices by iteration. Follows lecture 6 notes.

C_0=max(S_0-strikes,0);
C=horzcat(C_0,data_prices);
maturities=horzcat(0,maturities);
dt=diff(maturities);
dK=strikes(2)-strikes(1);

```

```
30 for j=1:length(maturities)-1
31     nonzero_indices=find(data_vols(:,j));
32     positive_data_vols=data_vols(:,j)>0;
33     fun=@(local_vol) sum(positive_data_vols.*(Andreasen_Huge(
           local_vol,nonzero_indices,C(:,j),strikes,dt(j),dK)-C(:,j+1))
           .^2);
34     lower_bound = zeros(length(nonzero_indices),1);
35     upper_bound = S_0*ones(length(nonzero_indices),1);
36     local_vols_start_guess = 0.3*S_0*ones(length(nonzero_indices),1);
37     local_vols{j}= fmincon(fun,local_vols_start_guess,[],[],[],[],
           lower_bound,upper_bound);
38     C(:,j+1) = Andreasen_Huge(local_vols{j},nonzero_indices,C(:,j),
           strikes,dt(j),dK);
39 end
40
41
42 %% Finally, inverting the BS call option formula to obtain implied
       volatilities.
43
44 impvols=zeros(length(strikes),length(maturities)-1);
45
46 for j=1:length(maturities)-1
47     for i=1:length(strikes)
48         fun=@(x) (BSoptionprice(S_0,strikes(i),x,0,maturities(j+1),0)
               -C(i,j+1))^2;
49         if j>1
50             impvol_start_guess=impvols(i,j-1);
51         else
52             impvol_start_guess=0.7;          % Lower start guesses
                   don't find the implied volatility for the first
                   maturity.
53         end
54         impvols(i,j)=fminsearch(fun,impvol_start_guess);
55     end
56 end
57
58
59 %% Constructing implied volatility at T=1 and T=1.5 based on
       calibration at maturities where data exist
60
61 C_1=zeros(length(maturities),1);
62 C_15=zeros(length(maturities),1);
63 dt1=1-.7720;

       % .7720 is the last maturity in the data before 1 and 1.5.
64 dt15=1.5-.7720;
65 C_1=Andreasen_Huge(local_vols{6},nonzero_indices,C(:,7),strikes,dt1,
       dK);          % We use the same parameters for local volatility as
       calibrated previously
```

```
C_15=Andreasen_Huge(local_vols{6},nonzero_indices,C(:,7),strikes,dt15
    ,dK);
impvols1=zeros(length(strikes),1);
impvols15=zeros(length(strikes),1);

for i=1:length(strikes)
    fun1=@(x) (BSoptionprice(S_0,strikes(i),x,0,1,0)-C(i,7))^2;
    fun15=@(x) (BSoptionprice(S_0,strikes(i),x,0,1.5,0)-C(i,7))^2;
    impvol_start_guess=impvols(i,6);
    impvols1(i)=fminsearch(fun1,impvol_start_guess);
    impvols15(i)=fminsearch(fun15,impvol_start_guess);
end

%% Plotting

figure
mesh(strikes/S_0,maturities(2:end)',impvols')
title('Implied volatility surface')
zlabel('$\sigma_{implied}$',Interpreter='latex')
xlabel("$K/S_0$",Interpreter="latex")
ylabel('T',Interpreter='latex')

figure
plot(strikes/S_0,impvols1)
hold on
plot(strikes/S_0,impvols15)
title('Implied volatilities of $T=1$ and $T=1.5$',Interpreter='latex'
    )
legend('T=1','T=1.5')
ylabel('$\sigma_{implied}$',Interpreter='latex')
xlabel("$K/S_0$",Interpreter="latex")


%% Functions, Andreasen-Huge algorithm and BS call option pricing
    formula below.

function call_price=Andreasen_Huge(local_vol,nonzero_indices,C,K,dt,
    dK)

local_vols=zeros(length(K));
midpoints = int16((nonzero_indices(1:end-1,1) + nonzero_indices(2:end
    ,1)) / 2);
kk = 1;

for i = 1:length(midpoints)
    while kk < midpoints(i)
        local_vols(kk) = local_vol(i);
        kk = kk+1;
    end
```

```
110
111 end
112 local_vols(kk:end) = local_vol(end);
113
114 z=zeros(length(K)-2,1);
115 A=zeros(length(K));
116 A(1,1)=1;
117 A(end,end)=1;
118
119 for i=2:length(K)-1
120     z=0.5*dt/dK^2*local_vols(i)^2;
121     A(i,i-1)=-z;
122     A(i,i)=1+2*z;
123     A(i,i+1)=-z;
124 end
125 call_price=A\C;
126 end
127
128
129 function price=BSoptionprice(S,K,sigma,r,T,q)
130 d_1=(log(S/K)+(r-q+0.5*sigma^2)*T)/(sqrt(T)*sigma);
131 d_2=d_1-sqrt(T)*sigma;
132 price=exp(-q*T)*S*normcdf(d_1)-K*exp(-r*T)*normcdf(d_2);
133 end
```