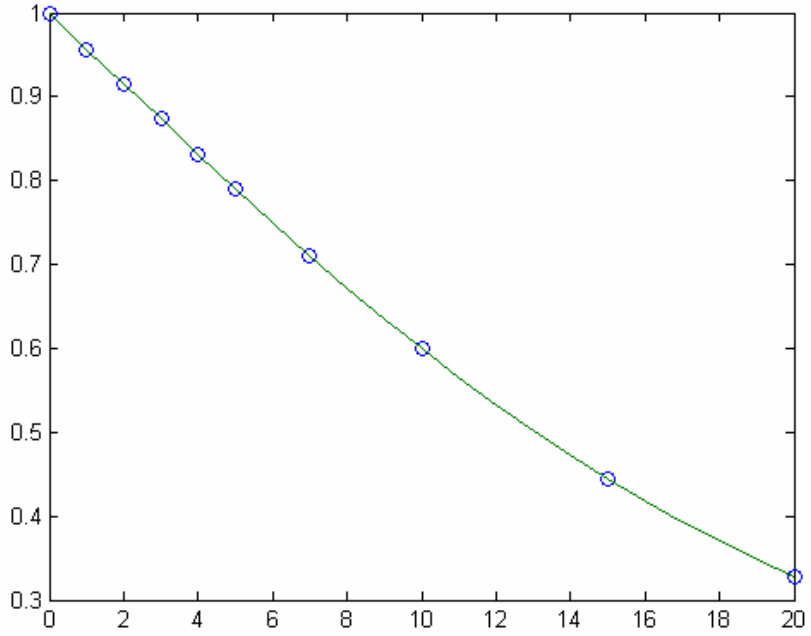
MATH 623 **Semester: F03**
HW#6 Solution

1a) Bond Pricing Formula (Zero-Coupon Bond)

$$P(0,T) = (1+y)^{-T}$$

And use linear interpolation to yield and calculate bond price for 0<T<20.

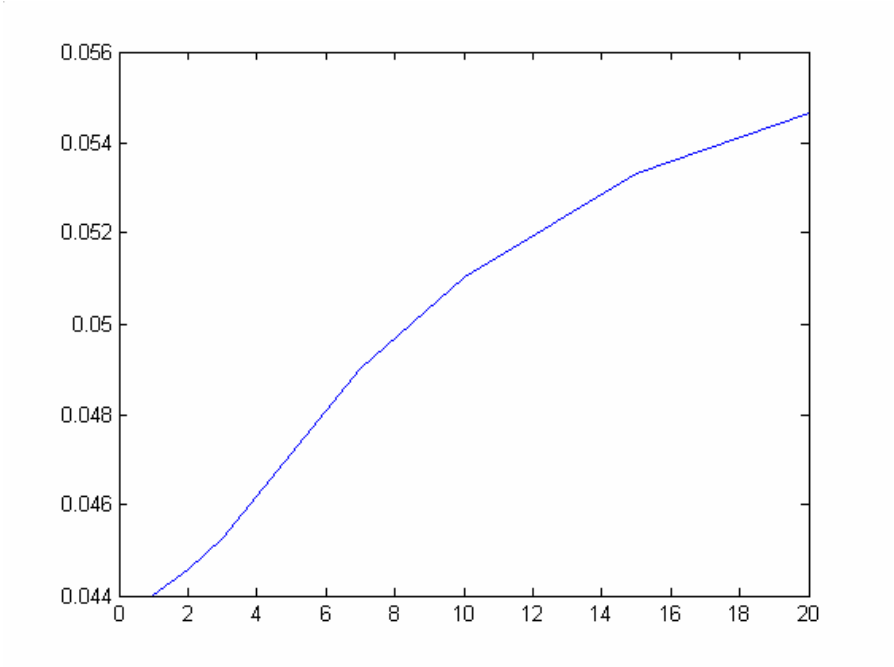


See code below.

2b) The Swap rate R(T) at time 0 with maturity T_n is given by

$$R(T) = \frac{P(0,T_0) - P(0,T_n)}{\sum_{i=1}^{n} (T_i - T_{i-1}) P(0,T_i)}$$

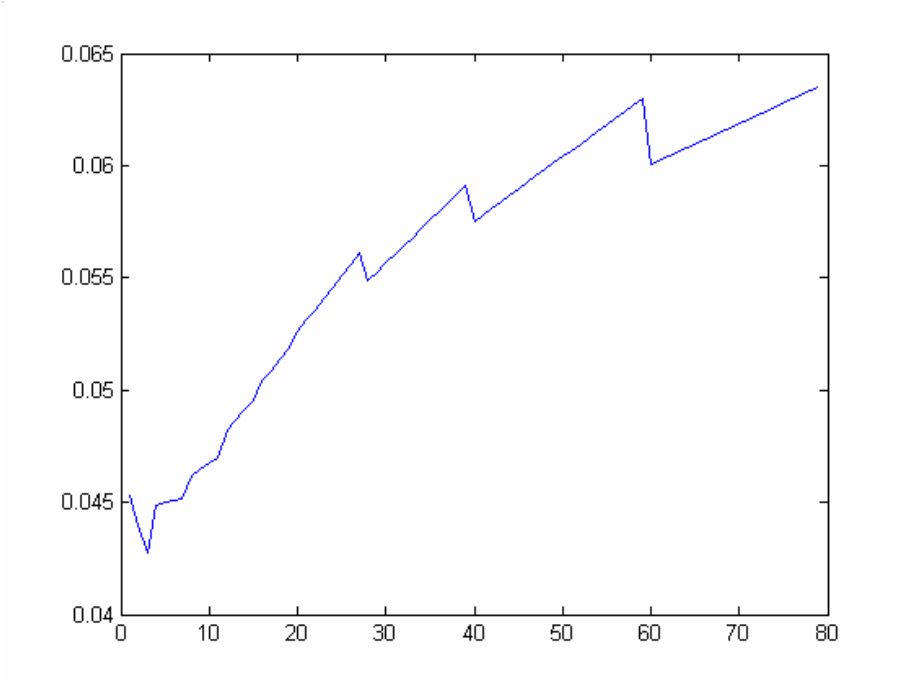
Maturity	R(t)
1	0.04403
2	0.04459
3	0.04529
4	0.04619
5	0.04713
7	0.04901
10	0.05103
15	0.05332
20	0.05463



See code below

3c) The LIBOR forward price $F(0,T_{i-1},T_i)$ is given by

$$F(0,T_{i-1},T_i) = \frac{\left(\frac{P(0,T_{i-1})}{P(0,T_i)} - 1\right)}{(T_i - T_{i-1})}$$



See code below

1d) Market Price of Cap is given by

$$Cap = K \sum_{i=1}^{n} P(0,T_i)(T_i - T_{i-1})\left(F_i N(d_1^i) - RN(d_2^i)\right)$$

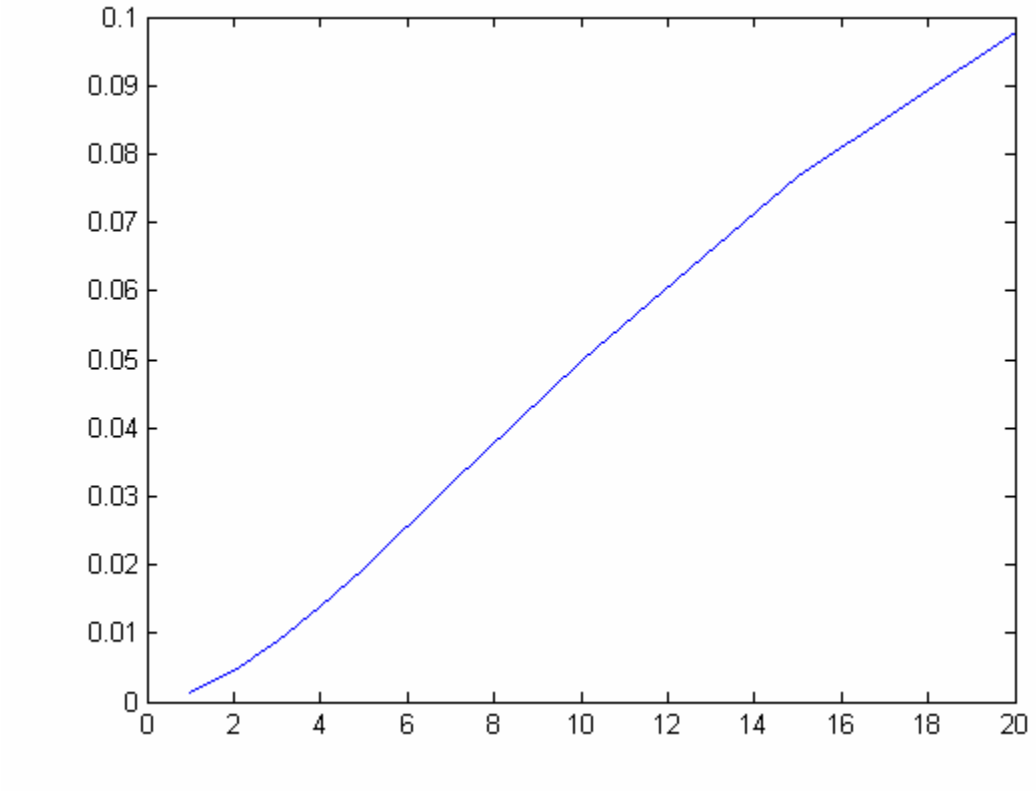
where $F_i = F(0,T_{i-1},T_i)$

$$d_1^i = \frac{1}{\sigma\sqrt{T_{i-1}}}\left(\log(\frac{F_i}{R}) + \frac{1}{2}\sigma^2 T_{i-1}\right)$$

$$d_2^i = d_1^i - \sigma\sqrt{T_{i-1}}$$

In our case, K=1

Maturity	Cap
1	0.00137
2	0.00460
3	0.00884
4	0.01383
5	0.01934
7	0.03162
10	0.04997
15	0.07690
20	0.09775



Code for 1

```
format long
Yield=[0.0480 0.0454 0.0456 0.0462 0.0471 0.0481 0.0502 0.0526 0.0556
0.0575];
Maturity=[0 1 2 3 4 5 7 10 15 20];
vol=[0.1520 0.1620 0.1640 0.1630 0.1605 0.1555 0.1475 0.1350 0.1260];
PP=(1+Yield).^(-Maturity);
dt=3/12;

%a
T=[0:20];
Y=interp1(Maturity',Yield',T','linear')';
P=(1+Y).^(-T);
fprintf(1,'Maturity= %6.0f -> P(0,T)= %8.5f \n',[T;P])
plot(Maturity,PP,'o',T,P)

%b
TT=[1:80];
YY=interp1(Maturity',Yield',TT'./4,'linear')';
PC=(1+YY).^(-TT.*dt);

for j=2:length(Maturity)
   sum=0;
   for i=2:Maturity(j)/dt
      sum=sum+dt*PC(i);
   end
   R(j-1)=((PC(1)-P(Maturity(j)+1))/sum);
end
fprintf(1,'Maturity= %6.0f -> R(T)= %8.5f \n',[Maturity(2:10);R])
%plot(Maturity(2:10),R)

%c
F=((PC(1:79)./PC(2:80))-1)/(dt);
fprintf(1,'i (month)= %6.0f -> F(0;T(i-1),T(i)) = %8.5f \n',[TT(1:79);F])
%plot(F)

%d
for j=2:length(Maturity)
   sumd=0;
   for i=2:Maturity(j)/dt
      d1=(log(F(i-1)/R(j-1))+0.5*(vol(j-1)^2)*(i-1)*dt)/(vol(j-1)*sqrt((i-1)*dt));
      d2=d1-vol(j-1)*sqrt((i-1)*dt);
      sumd=sumd+PC(i)*dt*(F(i-1)*normcdf(d1,0,1)-R(j-1)*normcdf(d2,0,1));
   end
   Cap(j-1)=sumd;
```

```
end
fprintf(1,'Maturity= %6.0f -> Cap price = %8.5f \n',[Maturity(2:10);Cap])
plot(Maturity(2:10),Cap)
```

2a) Black-Derman-Toy tree is a binomial tree, which interest rate at each node (m,j) is

$$r_j^m = r_0^m e^{2j\beta^m \Delta t}$$

At time m=0, calibrate zero-coupon price as follows.

$$P(0,T_0) = e^{-r_0^0 \Delta t} \Rightarrow r_0^0 = -\frac{1}{\Delta t}\log P(0,T_0)$$

and Arrow-Dereu price

$$Q_0^0 = 1$$

For m>0, calibrate zero-coupon price P^{m+1} and volatility σ^{m+1} to find r_0^m and β^m as follows.

As we know that $r_j^m = r_0^m e^{2j\beta^m \Delta t}$ for m>1

We can calculate Arrow-Debreu securities:

$$Q_0^m = \frac{1}{2} e^{-r_0^{m-1}\Delta t} Q_0^{m-1}$$

$$Q_j^m = \frac{1}{2} e^{-r_{j-1}^{m-1}\Delta t} Q_{j-1}^{m-1} + \frac{1}{2} e^{-r_j^{m-1}\Delta t} Q_j^{m-1}$$

$$Q_m^m = \frac{1}{2} e^{-r_{m-1}^{m-1}\Delta t} Q_{m-1}^{m-1}$$

And similarly we also calculate $Q_{j,0}^m$ and $Q_{j,1}^m$. Then calculate

$$P^{m+1} = \sum_{j=0}^{m} e^{-r_0^m e^{2j\beta^m \Delta t}\Delta t} Q_j^m$$

$$P_0^{m+1} = \sum_{j=0}^{m} e^{-r_0^m e^{2j\beta^m \Delta t}\Delta t} Q_{j,0}^m$$

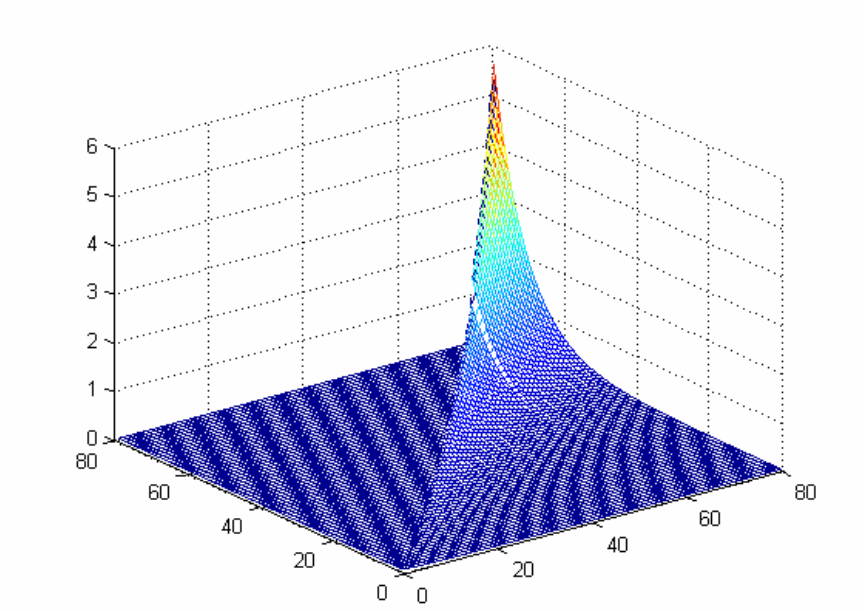
$$P_1^{m+1} = \sum_{j=0}^{m} e^{-r_0^m e^{2j\beta^m \Delta t}\Delta t} Q_{j,0}^m$$

And $\sigma^{m+1} = \frac{1}{2\sqrt{\Delta t}} \log \frac{\log P_1^{m+1}}{\log P_0^{m+1}}$

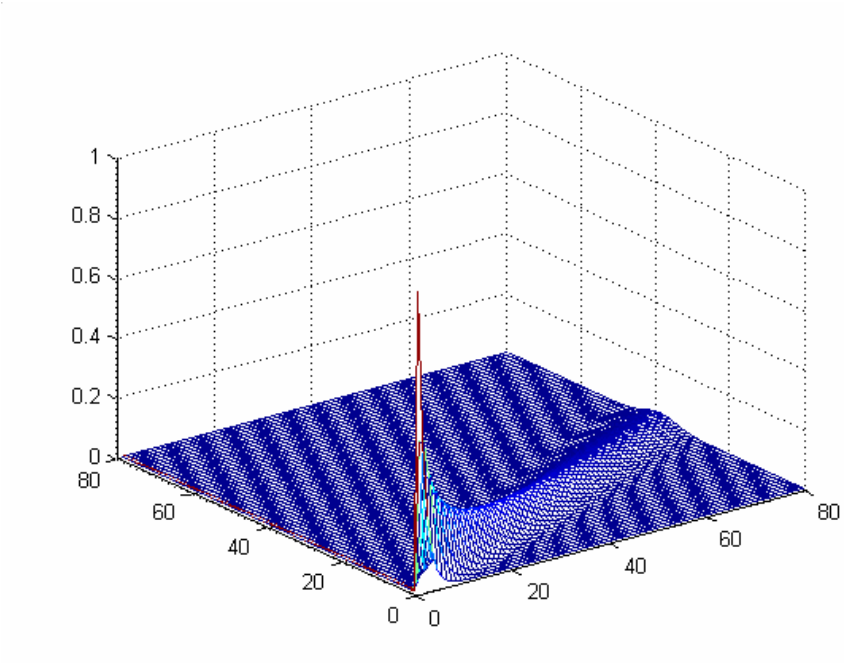
Then we can compute, r_0^m and β^m by fsolve function and at each node we can compute r_j^m by equation $r_j^m = r_0^m e^{2j\beta^m \Delta t}$.

See code below.

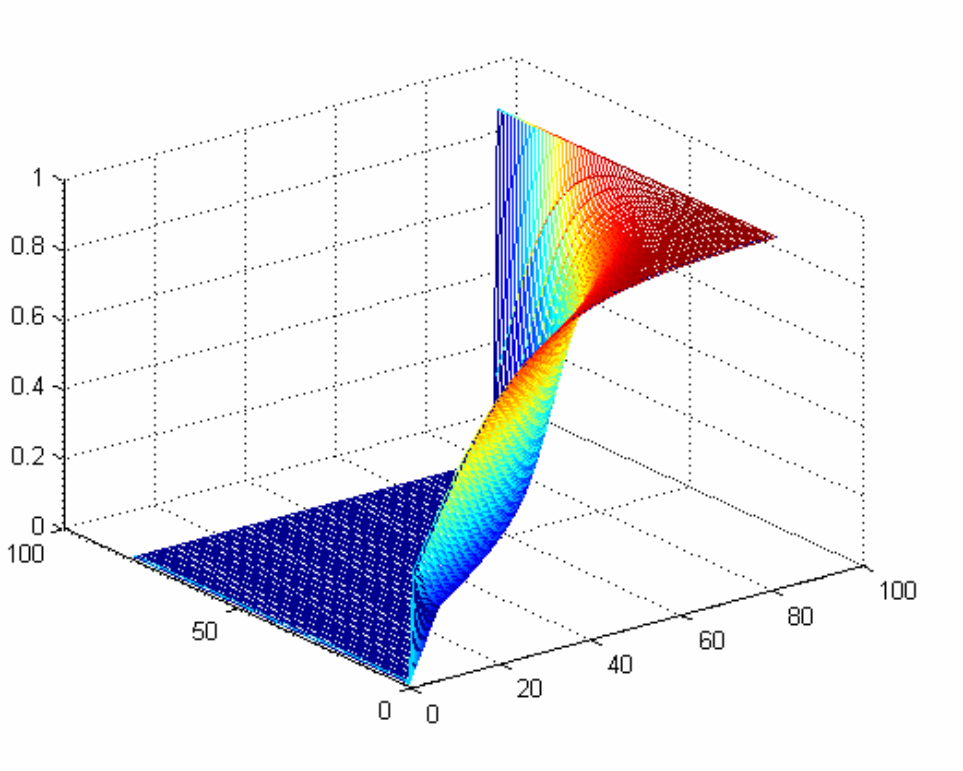
2b) See code below



2c) See code below



2d) See Code below



2e) See Code below

European bond put option price = 75741.2539

2f) See Code below

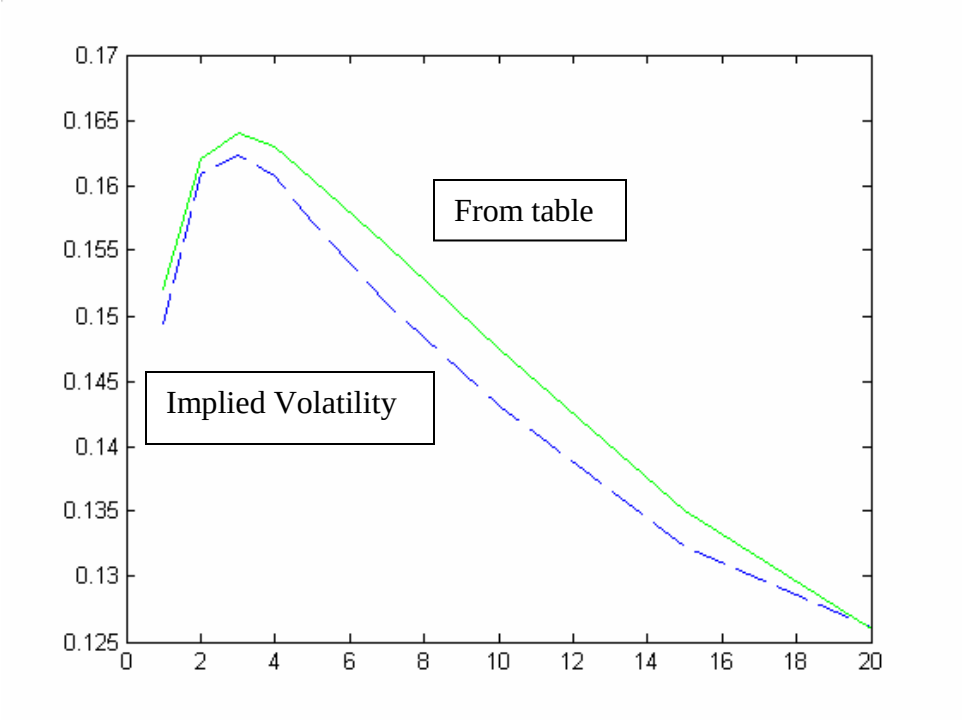
Bermudan put option price = 156515.8068

2g) See Code below

Maturity	Cap
1	0.00135
2	0.00457
3	0.00875
4	0.01365
5	0.01896
7	0.03080
10	0.04868
15	0.07557
20	0.09780

2h)

Maturity	Cap	Implied Volatility
1	0.00135	0.14939
2	0.00457	0.16084
3	0.00875	0.16227
4	0.01365	0.16076
5	0.01896	0.15715
7	0.03080	0.15099
10	0.04868	0.14311
15	0.07557	0.13226
20	0.09780	0.12607



Code for 2

```
global P vol Q Qu Qd r dt m j R y Cap Maturity
T=20;
dt=0.25;
M=T/dt;
t=[dt:dt:T];
Yield=[0.0480 0.0454 0.0456 0.0462 0.0471 0.0481 0.0502 0.0526 0.0556
0.0575];
Maturity=[0 1 2 3 4 5 7 10 15 20];
v=[0.1520 0.1620 0.1640 0.1630 0.1605 0.1555 0.1475 0.1350 0.1260];
y=interp1(Maturity,Yield,t,'linear');
P=(1+y).^(-t);
vol=interp1(Maturity(2:10),v,t,'linear','extrap');

mytol=1e-12;

%a
b=zeros(1,M);
r=zeros(M);
Q=zeros(M);
Qu=zeros(M);
Qd=zeros(M);
Q(1,1)=1;
r(1,1)=-log(P(1))/dt;

for m=1:M-1
  Q1=Q(1:m,m).*exp(-r(1:m,m)*dt);
  Q(1:m+1,m+1)=0.5*([Q1;0]+[0;Q1]);
  if m==1
    Qu(2,2)=1;
    Qd(1,2)=1;
  else
    Q1=Qu(1:m,m).*exp(-r(1:m,m)*dt);
    Qu(1:m+1,m+1)=0.5*([Q1;0]+[0;Q1]);
    Q1=Qd(1:m,m).*exp(-r(1:m,m)*dt);
    Qd(1:m+1,m+1)=0.5*([Q1;0]+[0;Q1]);
  end
  r0=r(1,m); b0=0.1;
  x0=[r0 b0];
  x0=fsolve(@bdtfun1,x0,optimset('Display','off','Tolfun',mytol,'TolX',mytol));
  r0=x0(1);b0=x0(2);
  rvec=r0*exp(2*b0*[0:m]*sqrt(dt));
  r(1:m+1,m+1)=rvec';
  b(m+1)=b0;
  yu(m+1)=-log(exp(-rvec*dt)*Qu(1:m+1,m+1))/(m*dt);
  yd(m+1)=-log(exp(-rvec*dt)*Qd(1:m+1,m+1))/(m*dt);
```

```
end
%b
mesh(r)

%c
mesh(Q)

%d

B=[zeros(M+1,M) ones(M+1,1)];
for m=M:-1:1
  B(1:m,m)=0.5*exp(-r(1:m,m)*dt).*(B(1:m,m+1)+B(2:m+1,m+1));
end
mesh(B)

% e

T0=5;
M0=T0/dt;
K=5e5;
Pe=zeros(M0+1,M0+1);
Pe(1:M0+1,M0+1)=max(0,K-B(1:M0+1,M0+1)*1e6);
for m=M0:-1:1
  Pe(1:m,m)=0.5*exp(-r(1:m,m)*dt).*(Pe(1:m,m+1)+Pe(2:m+1,m+1));
end
fprintf('European bond put option price = %6.4f\n',Pe(1,1))

% f
Pa=zeros(M0+1,M0+1);
Pa(1:M0+1,M0+1)=max(0,K-B(1:M0+1,M0+1)*1e6);
for m=M0:-1:1
  Pa(1:m,m)=0.5*exp(-r(1:m,m)*dt).*(Pa(1:m,m+1)+Pa(2:m+1,m+1));
  if m*dt==1 | m*dt==2 | m*dt==3 | m*dt==4 | m*dt==5
    Pa(1:m,m)=max(K-B(1:m,m)*1e6,Pa(1:m,m));
  end
end
fprintf('Bermudan put option price = %6.4f\n',Pa(1,1))

%g
R =[0.04402801188195   0.04459243938536   0.04528562229214
0.04618713855026   0.04713314114882   0.04901437220027...
   0.05103204571436   0.05332131588320   0.05463364052803]; %----------
SWAP price from Q1
K1=1;
```

```
for i=2:10
   MM=Maturity(i)/dt;
   Temp=zeros(MM,MM);
   Temp(1:MM,MM)=max(K1*dt*(r(1:MM,MM)-R(i-1)).*exp(-
r(1:MM,MM)*dt),0);
   for m=MM-1:-1:1
      if m~=1
         Temp(1:m,m)=max(K1*dt*(r(1:m,m)-R(i-1)).*exp(-
r(1:m,m)*dt),0)+0.5*exp(-r(1:m,m)*dt).*(Temp(1:m,m+1)+Temp(2:m+1,m+1));
      else
         Temp(1:m,m)=0.5*exp(-
r(1:m,m)*dt).*(Temp(1:m,m+1)+Temp(2:m+1,m+1));
      end
   end
   Cap(i-1)=Temp(1,1);
end
plot(Maturity(2:10),Cap)

%h
for j=2:length(Maturity)
   x0=[0.15];
   x0=fsolve(@impvol,x0,optimset('Display','off','Tolfun',mytol,'TolX',mytol));
   sig(j-1)=x0;
end
plot(Maturity(2:10),sig,'--')
hold
plot(Maturity(2:10),v,'g')
```

Function bdtfun1 using in 2a

```
function f=bdtfun1(x0)
global P vol Q Qu Qd dt m;

r0=x0(1);b0=x0(2);
f=ones(2,1);
rvec=r0*exp(2*b0*[0:m]*sqrt(dt));
pu0=exp(-rvec*dt)*Qu(1:m+1,m+1);
pd0=exp(-rvec*dt)*Qd(1:m+1,m+1);
f(1)=P(m+1)-exp(-rvec*dt)*Q(1:m+1,m+1);
f(2)=vol(m+1)-log(log(pu0)/log(pd0))/(2*sqrt(dt));
```

Function impvol using in 2h

```
function f=impvol(x0)
global P vol Q Qu Qd r dt m j R y Cap Maturity
F=((P(1:79)./P(2:80))-1)/dt;
sig=x0;
```

```
sumd=0;
   for i=2:Maturity(j)/dt
      d1=(log(F(i-1)/R(j-1))+0.5*(sig^2)*(i-1)*dt)/(sig*sqrt((i-1)*dt));
      d2=d1-sig*sqrt((i-1)*dt);
      sumd=sumd+P(i)*dt*(F(i-1)*normcdf(d1,0,1)-R(j-1)*normcdf(d2,0,1));
   end
f=Cap(j-1)-sumd;
```

3a) The Black-Karasinski interest rate tree is trinomial as Hull White one. So we pick parameters as in Hull White model, we use forward induction to get tree interest rate tree as follow.

The interest rate at node (m,j) is $e^{j\Delta t+\alpha^m}$

For m=0, set Q_0^0 =1

And compute $\alpha^0 = \log\left(-\frac{1}{\Delta t}\log P^1\right)$

For m>1

Compute Arrow Debreu securities at time $m\Delta t$ as follows

$$Q_j^m = \sum_{k=-\min\{m-1,J\}}^{\min\{m-1,J\}} Q_k^{m-1} q_{k,j} e^{-e^{(j\Delta t+\alpha^{m-1})\Delta t}}$$

where $q_{k,j}$ is the probability of going from node (m,k) to (m+1,j).

(Following as in Hull White: Pu, Pm, Pd)

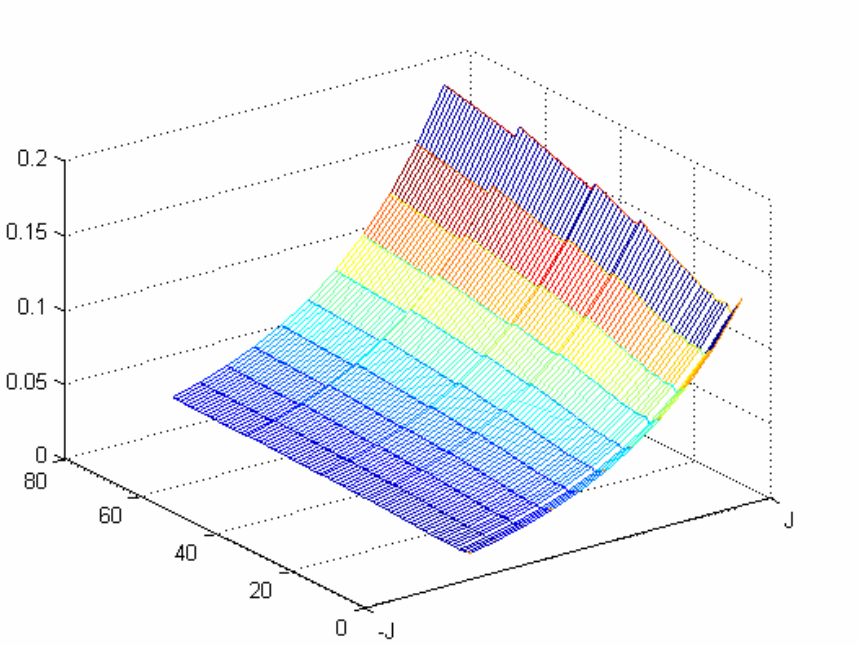
Then calibrate to the bond of maturity $(m+1)\Delta t$:

$$P^{m+1} = \sum_{j=-\min\{m,J\}}^{\min\{m,J\}} Q_j^m e^{-e^{(j\Delta t+\alpha^m)\Delta t}}$$

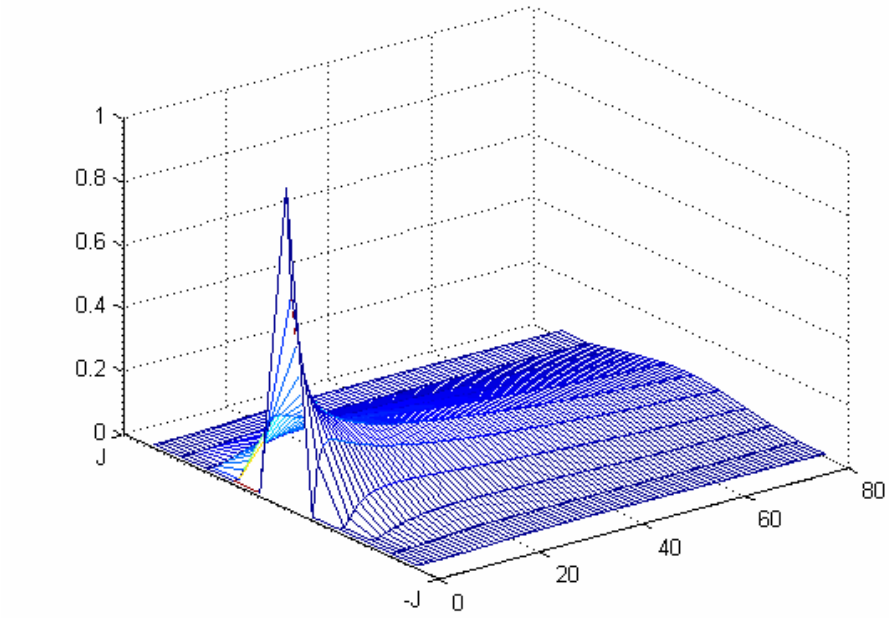
and we can solve α^m numerically by fsolve function.

See Code below

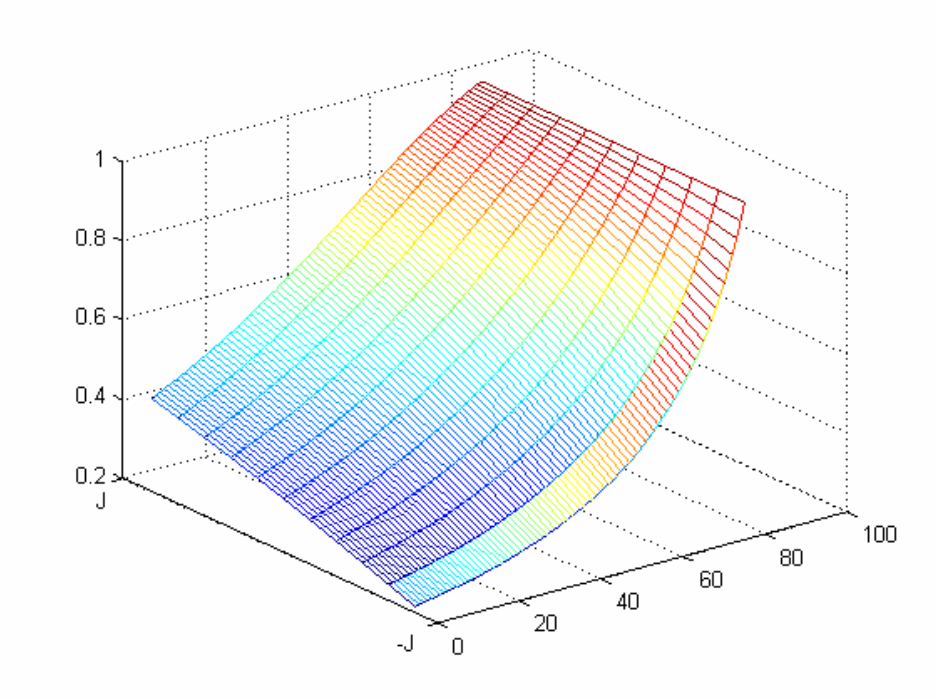
3b) See Code below



3c) See Code below



3d)



3e)

European put option price = 69205.922

3f)

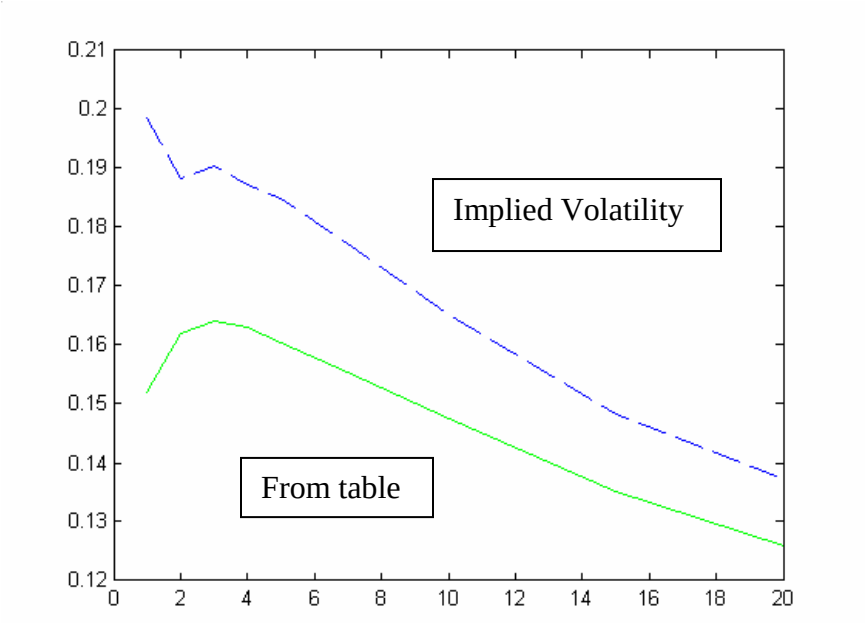
Bermudan put option price = 151450.131

3g)

Maturity	Cap
1	0.001766
2	0.005330
3	0.010225
4	0.015795
5	0.022105
7	0.035611
10	0.055155
15	0.083366
20	0.105154

3h)

Maturity	Cap	Implied Volatility
1	0.001766	0.198619
2	0.005330	0.188160
3	0.010225	0.190320
4	0.015795	0.187129
5	0.022105	0.184941
7	0.035611	0.177357
10	0.055155	0.165012
15	0.083366	0.148293
20	0.105154	0.137224



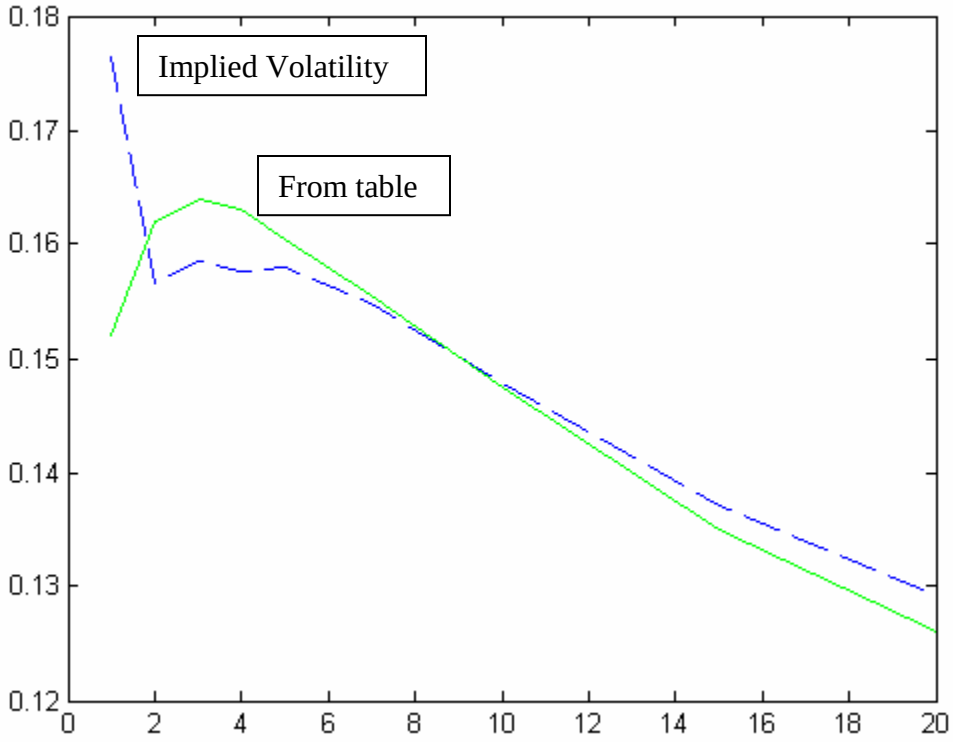
3i) See Code below

Using fminsearch function in Matlab, we could find the optimal a and sigma as follow.

a=0.09379
sigma=0.20247

With these parameters, we obtain minimal sum square = 0.000699049
And redo e) and f) with these parameters, we get result as follow.

European put option price = 70569.905
Bermudan put option price = 151407.220



Code for 3a-3h

```
clear all;
global P Q j dr dt m Cap Maturity R

dt=0.25;
a=0.15;
sigma=0.25;
T=20;
M=T/dt;

J=ceil(0.184/(a*dt));
dr=sigma*sqrt(3*dt);
j=(J:-1:-J)';

alpha=zeros(1,M);
r=zeros(2*J+1,M);
rstar=zeros(2*J+1,M);
Q=zeros(2*J+1,M);
pu=zeros(2*J+1,1);
pm=zeros(2*J+1,1);
pd=zeros(2*J+1,1);
rstar=repmat(j*dr,1,M);
```

```
%|j|<J
pd=1/6+1/2*(a*j*dt).^2-1/2*a*j*dt;
pm=2/3-(a*j*dt).^2;
pu=1/6+1/2*(a*j*dt).^2+1/2*a*j*dt;

%j=J
pd(1)=7/6+1/2*((a*J*dt)^2-3*a*J*dt);
pm(1)=-1/3-(a*J*dt)^2+2*a*J*dt;
pu(1)=1/6+1/2*((a*J*dt)^2-a*J*dt);

%j=-J
pd(2*J+1)=1/6+1/2*((a*J*dt)^2-a*J*dt);
pm(2*J+1)=-1/3-(a*J*dt)^2+2*a*J*dt;
pu(2*J+1)=7/6+1/2*((a*J*dt)^2-3*a*J*dt);

% Calibrate

t=(dt:dt:T);
Yield=[0.0480 0.0454 0.0456 0.0462 0.0471 0.0481 0.0502 0.0526 0.0556
0.0575];
vol=[0.1520 0.1620 0.1640 0.1630 0.1605 0.1555 0.1475 0.1350 0.1260];
Maturity=[0 1 2 3 4 5 7 10 15 20];
y=interp1(Maturity,Yield,t,'linear');
P=(1+y).^(-t);

alpha(1)=log(-log(P(1))/dt);
r(:,1)=exp(rstar(:,1)+alpha(1));
Q(:,1)=(j==0);
mytol=1e-12;

for m=2:M

  Q(1,m)=Q(1,m-1)*pd(1)*exp(-r(1,m-1)*dt)+...
         Q(2,m-1)*pd(2)*exp(-r(2,m-1)*dt);

  Q(2,m)=Q(1,m-1)*pm(1)*exp(-r(1,m-1)*dt)+...
         Q(2,m-1)*pm(2)*exp(-r(2,m-1)*dt)+...
         Q(3,m-1)*pd(3)*exp(-r(3,m-1)*dt);

  Q(3,m)=Q(1,m-1)*pu(1)*exp(-r(1,m-1)*dt)+...
         Q(2,m-1)*pu(2)*exp(-r(2,m-1)*dt)+...
         Q(3,m-1)*pm(3)*exp(-r(3,m-1)*dt)+...
         Q(4,m-1)*pd(4)*exp(-r(4,m-1)*dt);
```

```
  Q(4:2*J-2,m)=...
     Q(3:2*J-3,m-1).*pu(3:2*J-3).*exp(-r(3:2*J-3,m-1)*dt)+...
     Q(4:2*J-2,m-1).*pm(4:2*J-2).*exp(-r(4:2*J-2,m-1)*dt)+...
     Q(5:2*J-1,m-1).*pd(5:2*J-1).*exp(-r(5:2*J-1,m-1)*dt);

  Q(2*J-1,m)=...
     Q(2*J+1,m-1)*pd(2*J+1)*exp(-r(2*J+1)*dt)+...
     Q(2*J,m-1)*pd(2*J)*exp(-r(2*J,m-1)*dt)+...
     Q(2*J-1,m-1)*pm(2*J-1)*exp(-r(2*J-1,m-1)*dt)+...
     Q(2*J-2,m-1)*pu(2*J-2)*exp(-r(2*J-2,m-1)*dt);

  Q(2*J,m)=...
     Q(2*J+1,m-1)*pm(2*J+1)*exp(-r(2*J+1,m-1)*dt)+...
     Q(2*J,m-1)*pm(2*J)*exp(-r(2*J,m-1)*dt)+...
     Q(2*J-1,m-1)*pu(2*J-1)*exp(-r(2*J-1,m-1)*dt);

  Q(2*J+1,m)=...
     Q(2*J+1,m-1)*pu(2*J+1)*exp(-r(2*J+1,m-1)*dt)+...
     Q(2*J,m-1)*pu(2*J)*exp(-r(2*J,m-1)*dt);

  x0=alpha(m-1);
  x0=fsolve(@alphafun,x0,optimset('Display','off','Tolfun',mytol,'TolX',mytol));
  alpha(m)=x0;
  r(:,m)=exp(rstar(:,m)+alpha(m));
end

%b
mesh(r)

%c
mesh(Q)

% d
B=zeros(2*J+1,M+1);

B(:,M+1)=1;

for m=M:-1:1

  B(1,m)=exp(-r(1,m)*dt)*(pd(1)*B(1,m+1)+...
                          pm(1)*B(2,m+1)+...
                          pu(1)*B(3,m+1));

  B(2*J+1,m)=exp(-r(2*J+1,m)*dt)*(pu(2*J+1)*B(2*J+1,m+1)+...
                                  pm(2*J+1)*B(2*J,m+1)+...
                                  pd(2*J+1)*B(2*J-1,m+1));
```

```
 B(2:2*J,m)=exp(-r(2:2*J,m)*dt).*(pd(2:2*J).*B(1:2*J-1,m+1)+...
                                  pm(2:2*J).*B(2:2*J,m+1)+...
                                  pu(2:2*J).*B(3:2*J+1,m+1));

end
mesh(B)

% e

M1=5/dt;

V=zeros(2*J+1,M1+1);

EV=1e6*(0.5-B(:,1:M1+1));

V(:,M1+1)=max(EV(:,M1+1),0);

for m=M1:-1:1

  V(1,m)=exp(-r(1,m)*dt)*(pd(1)*V(1,m+1)+...
                          pm(1)*V(2,m+1)+...
                               pu(1)*V(3,m+1));

  V(2*J+1,m)=exp(-r(2*J+1,m)*dt)*(pu(2*J+1)*V(2*J+1,m+1)+...
                                  pm(2*J+1)*V(2*J,m+1)+...
                                  pd(2*J+1)*V(2*J-1,m+1));

  V(2:2*J,m)=exp(-r(2:2*J,m)*dt).*(pd(2:2*J).*V(1:2*J-1,m+1)+...
                                   pm(2:2*J).*V(2:2*J,m+1)+...
                                   pu(2:2*J).*V(3:2*J+1,m+1));

end
fprintf('European put option price = %6.3f\n',V(J+1,1))

% f

VV=zeros(2*J+1,M1+1);

EVV=1e6*(0.5-B(:,1:M1+1));

VV(:,M1+1)=max(EVV(:,M1+1),0);
```

```
for m=M1:-1:1

  VV(1,m)=exp(-r(1,m)*dt)*(pd(1)*VV(1,m+1)+...
                           pm(1)*VV(2,m+1)+...
                                pu(1)*VV(3,m+1));

  VV(2*J+1,m)=exp(-r(2*J+1,m)*dt)*(pu(2*J+1)*VV(2*J+1,m+1)+...
                                   pm(2*J+1)*VV(2*J,m+1)+...
                                   pd(2*J+1)*VV(2*J-1,m+1));

  VV(2:2*J,m)=exp(-r(2:2*J,m)*dt).*(pd(2:2*J).*VV(1:2*J-1,m+1)+...
                                    pm(2:2*J).*VV(2:2*J,m+1)+...
                                    pu(2:2*J).*VV(3:2*J+1,m+1));

  if (m==2/dt+1) | (m==1/dt+1) | (m==3/dt+1) | (m==4/dt+1) | (m==5/dt+1)

    VV(:,m)=max(VV(:,m),EVV(:,m));

  end

end
fprintf('Bermudan put option price = %6.3f\n',VV(J+1,1))

%g
R =[0.04402801188195   0.04459243938536   0.04528562229214…
          0.04618713855026       0.04713314114882   0.04901437220027...
          0.05103204571436   0.05332131588320   0.05463364052803]; %----------
-------SWAP price from Q1

for j=2:length(Maturity)
   MM=Maturity(j)/dt;
   Temp=zeros(2*J+1,MM);


for m=MM-1:-1:1

  Temp(1,m)=dt*max(r(1,m)-R(j-1),0).*exp(-r(1,m)*dt) +...
            exp(-r(1,m)*dt)*(pd(1)*Temp(1,m+1)+...
                             pm(1)*Temp(2,m+1)+...
                                  pu(1)*Temp(3,m+1));

  Temp(2*J+1,m)=dt*max(r(2*J+1,m)-R(j-1),0).*exp(-r(2*J+1,m)*dt) +...
                exp(-r(2*J+1,m)*dt)*(pu(2*J+1)*Temp(2*J+1,m+1)+...
                                     pm(2*J+1)*Temp(2*J,m+1)+...
                                     pd(2*J+1)*Temp(2*J-1,m+1));
```

```
  Temp(2:2*J,m)=dt*max(r(2:2*J,m)-R(j-1),0).*exp(-r(2:2*J,m)*dt) +...
                exp(-r(2:2*J,m)*dt).*(pd(2:2*J).*Temp(1:2*J-1,m+1)+...
                                      pm(2:2*J).*Temp(2:2*J,m+1)+...
                                      pu(2:2*J).*Temp(3:2*J+1,m+1));
end
Cap(j-1)=Temp(J+1,1);
end
plot(Maturity(2:10),Cap)

%h
for j=2:length(Maturity)
   x0=[0.15];
   x0=fsolve(@impvol,x0,optimset('Display','off','Tolfun',mytol,'TolX',mytol));
   sig(j-1)=x0;
end
plot(Maturity(2:10),sig,'--')
hold
plot(Maturity(2:10),vol,'g')
```

Function impvol using in 3h

```
function f=impvol(x0)
global P vol Q Qu Qd r dt m j R y Cap Maturity
F=((P(1:79)./P(2:80))-1)/dt;
sig=x0;
sumd=0;
   for i=2:Maturity(j)/dt
      d1=(log(F(i-1)/R(j-1))+0.5*(sig^2)*(i-1)*dt)/(sig*sqrt((i-1)*dt));
      d2=d1-sig*sqrt((i-1)*dt);
      sumd=sumd+P(i)*dt*(F(i-1)*normcdf(d1,0,1)-R(j-1)*normcdf(d2,0,1));
   end
f=Cap(j-1)-sumd;
```

Code for 3i Extra Credit

```
clear
format long

global dr dt M T Maturity sig vol P Q Qu Qd r m j R y Cap J pu pm pd B

dt=0.25;
T=20;
M=T/dt;

y0=[0.15 0.25];
[y0,fval]=fminsearch(@parameter,y0,optimset('Display','iter','Tolfun',1e-
8,'TolX',1e-8));
```

```
a=y0(1);
sigma=y0(2);

fprintf('a and sigma that minimize RSS is \n a=%6.5f \n sigma=%6.5f\n',a,sigma)
plot(Maturity(2:10),sig,'--');
hold
plot(Maturity(2:10),vol,'g');

% d
B=zeros(2*J+1,M+1);

B(:,M+1)=1;

for m=M:-1:1

  B(1,m)=exp(-r(1,m)*dt)*(pd(1)*B(1,m+1)+...
                          pm(1)*B(2,m+1)+...
                          pu(1)*B(3,m+1));


B(2*J+1,m)=exp(-r(2*J+1,m)*dt)*(pu(2*J+1)*B(2*J+1,m+1)+...
                                pm(2*J+1)*B(2*J,m+1)+...
                                pd(2*J+1)*B(2*J-1,m+1));

  B(2:2*J,m)=exp(-r(2:2*J,m)*dt).*(pd(2:2*J).*B(1:2*J-1,m+1)+...
                                pm(2:2*J).*B(2:2*J,m+1)+...
                                pu(2:2*J).*B(3:2*J+1,m+1));

end

% e
M1=5/dt;

V=zeros(2*J+1,M1+1);

EV=1e6*(0.5-B(:,1:M1+1));

V(:,M1+1)=max(EV(:,M1+1),0);

for m=M1:-1:1

  V(1,m)=exp(-r(1,m)*dt)*(pd(1)*V(1,m+1)+...
                          pm(1)*V(2,m+1)+...
                          pu(1)*V(3,m+1));
```

```
  V(2*J+1,m)=exp(-r(2*J+1,m)*dt)*(pu(2*J+1)*V(2*J+1,m+1)+...
                                  pm(2*J+1)*V(2*J,m+1)+...
                                  pd(2*J+1)*V(2*J-1,m+1));

  V(2:2*J,m)=exp(-r(2:2*J,m)*dt).*(pd(2:2*J).*V(1:2*J-1,m+1)+...
                                   pm(2:2*J).*V(2:2*J,m+1)+...
                                   pu(2:2*J).*V(3:2*J+1,m+1));

end
fprintf('European put option price = %6.3f\n',V(J+1,1))

% f
VV=zeros(2*J+1,M1+1);

EVV=1e6*(0.5-B(:,1:M1+1));

VV(:,M1+1)=max(EVV(:,M1+1),0);

for m=M1:-1:1

  VV(1,m)=exp(-r(1,m)*dt)*(pd(1)*VV(1,m+1)+...
                           pm(1)*VV(2,m+1)+...
                                pu(1)*VV(3,m+1));

  VV(2*J+1,m)=exp(-r(2*J+1,m)*dt)*(pu(2*J+1)*VV(2*J+1,m+1)+...
                                   pm(2*J+1)*VV(2*J,m+1)+...
                                   pd(2*J+1)*VV(2*J-1,m+1));

  VV(2:2*J,m)=exp(-r(2:2*J,m)*dt).*(pd(2:2*J).*VV(1:2*J-1,m+1)+...
                                    pm(2:2*J).*VV(2:2*J,m+1)+...
                                    pu(2:2*J).*VV(3:2*J+1,m+1));

  if (m==2/dt+1) | (m==1/dt+1) | (m==3/dt+1) | (m==4/dt+1) | (m==5/dt+1)

    VV(:,m)=max(VV(:,m),EVV(:,m));

  end

end
fprintf('Bermudan put option price = %6.3f\n',VV(J+1,1))
```

Function parameter using in 3i Extra Credit

```
function f=parameter(y0)
%need function impvol
global dr dt M T Maturity sig vol P Q Qu Qd r m j R y Cap J pu pm pd B

a=y0(1);
sigma=y0(2);

J=ceil(0.184/(a*dt));
dr=sigma*sqrt(3*dt);
j=(J:-1:-J)';

alpha=zeros(1,M);
r=zeros(2*J+1,M);
rstar=zeros(2*J+1,M);
Q=zeros(2*J+1,M);
pu=zeros(2*J+1,1);
pm=zeros(2*J+1,1);
pd=zeros(2*J+1,1);

rstar=repmat(j*dr,1,M);

%|j| < J
pd=1/6+1/2*(a*j*dt).^2-1/2*a*j*dt;
pm=2/3-(a*j*dt).^2;
pu=1/6+1/2*(a*j*dt).^2+1/2*a*j*dt;

%if j=J
pd(1)=7/6+1/2*((a*J*dt)^2-3*a*J*dt);
pm(1)=-1/3-(a*J*dt)^2+2*a*J*dt;
pu(1)=1/6+1/2*((a*J*dt)^2-a*J*dt);

%j=J
pd(2*J+1)=1/6+1/2*((a*J*dt)^2-a*J*dt);
pm(2*J+1)=-1/3-(a*J*dt)^2+2*a*J*dt;
pu(2*J+1)=7/6+1/2*((a*J*dt)^2-3*a*J*dt);

% Calibrate

t=(dt:dt:T);
Yield=[0.0480 0.0454 0.0456 0.0462 0.0471 0.0481 0.0502 0.0526 0.0556
0.0575];
vol=[0.1520 0.1620 0.1640 0.1630 0.1605 0.1555 0.1475 0.1350 0.1260];
Maturity=[0 1 2 3 4 5 7 10 15 20];
y=interp1(Maturity,Yield,t,'linear');
P=(1+y).^(-t);
```

```
alpha(1)=log(-log(P(1))/dt);
r(:,1)=exp(rstar(:,1)+alpha(1));
Q(:,1)=(j==0);
mytol=1e-12;

for m=2:M

  Q(1,m)=Q(1,m-1)*pd(1)*exp(-r(1,m-1)*dt)+...
         Q(2,m-1)*pd(2)*exp(-r(2,m-1)*dt);

  Q(2,m)=Q(1,m-1)*pm(1)*exp(-r(1,m-1)*dt)+...
         Q(2,m-1)*pm(2)*exp(-r(2,m-1)*dt)+...
         Q(3,m-1)*pd(3)*exp(-r(3,m-1)*dt);

  Q(3,m)=Q(1,m-1)*pu(1)*exp(-r(1,m-1)*dt)+...
         Q(2,m-1)*pu(2)*exp(-r(2,m-1)*dt)+...
         Q(3,m-1)*pm(3)*exp(-r(3,m-1)*dt)+...
         Q(4,m-1)*pd(4)*exp(-r(4,m-1)*dt);

  Q(4:2*J-2,m)=...
     Q(3:2*J-3,m-1).*pu(3:2*J-3).*exp(-r(3:2*J-3,m-1)*dt)+...
     Q(4:2*J-2,m-1).*pm(4:2*J-2).*exp(-r(4:2*J-2,m-1)*dt)+...
     Q(5:2*J-1,m-1).*pd(5:2*J-1).*exp(-r(5:2*J-1,m-1)*dt);

  Q(2*J-1,m)=...
     Q(2*J+1,m-1)*pd(2*J+1)*exp(-r(2*J+1)*dt)+...
     Q(2*J,m-1)*pd(2*J)*exp(-r(2*J,m-1)*dt)+...
     Q(2*J-1,m-1)*pm(2*J-1)*exp(-r(2*J-1,m-1)*dt)+...
     Q(2*J-2,m-1)*pu(2*J-2)*exp(-r(2*J-2,m-1)*dt);

  Q(2*J,m)=...
     Q(2*J+1,m-1)*pm(2*J+1)*exp(-r(2*J+1,m-1)*dt)+...
     Q(2*J,m-1)*pm(2*J)*exp(-r(2*J,m-1)*dt)+...
     Q(2*J-1,m-1)*pu(2*J-1)*exp(-r(2*J-1,m-1)*dt);

  Q(2*J+1,m)=...
     Q(2*J+1,m-1)*pu(2*J+1)*exp(-r(2*J+1,m-1)*dt)+...
     Q(2*J,m-1)*pu(2*J)*exp(-r(2*J,m-1)*dt);

  x0=alpha(m-1);
  x0=fsolve(@alphafun,x0,optimset('Display','off','Tolfun',mytol,'TolX',mytol));
  alpha(m)=x0;
  r(:,m)=exp(rstar(:,m)+alpha(m));

end
```

```
%Calculate Cap price from the Tree
R =[0.04402801188195   0.04459243938536   0.04528562229214
0.04618713855026... 0.04713314114882   0.04901437220027...
   0.05103204571436   0.05332131588320   0.05463364052803]; %----------------
-SWAP price from Q1

for j=2:length(Maturity)
   MM=Maturity(j)/dt;
   Temp=zeros(2*J+1,MM);

for m=MM-1:-1:1

  Temp(1,m)=dt*max(r(1,m)-R(j-1),0).*exp(-r(1,m)*dt) +...
          exp(-r(1,m)*dt)*(pd(1)*Temp(1,m+1)+...
                      pm(1)*Temp(2,m+1)+...
                            pu(1)*Temp(3,m+1));

  Temp(2*J+1,m)=dt*max(r(2*J+1,m)-R(j-1),0).*exp(-r(2*J+1,m)*dt) +...
              exp(-r(2*J+1,m)*dt)*(pu(2*J+1)*Temp(2*J+1,m+1)+...
                                   pm(2*J+1)*Temp(2*J,m+1)+...
                                   pd(2*J+1)*Temp(2*J-1,m+1));

  Temp(2:2*J,m)=dt*max(r(2:2*J,m)-R(j-1),0).*exp(-r(2:2*J,m)*dt) +...
               exp(-r(2:2*J,m)*dt).*(pd(2:2*J).*Temp(1:2*J-1,m+1)+...
                                     pm(2:2*J).*Temp(2:2*J,m+1)+...
                                     pu(2:2*J).*Temp(3:2*J+1,m+1));
end
Cap(j-1)=Temp(J+1,1);
end


%Calculate implied volatility from Black's formula
for j=2:length(Maturity)
   x0=[0.15];  %--------------------initial guess
   x0=fsolve(@impvol,x0,optimset('Display','off','Tolfun',mytol,'TolX',mytol));
   sig(j-1)=x0;
end
f=sum((sig-vol).^2);
```