Math 456: Introduction to Financial Mathematics

Homework Set 5

1. Suppose that S and R are modeled with a recombining binomial tree of $N \geq 1$ levels. Prove that

$$G(N-1,j) \;=\; F(N-1,j) \;=\; S(N-1,j)R(N-1,j)$$

for all states j.

Solution: From Eq.5.4 it follows that

$$F(N-1,j) = S(N-1,j)R(N-1,j).$$

It remains to compute $G(N-1,j)$ with the backward induction formula. Since $G(N,j) = S(N,j)$ for all j,

$$\begin{aligned} G(N-1,j) &= \pi G(N,j+1) + (1-\pi)G(N,j) \\ &= \pi S(N,j+1) + (1-\pi)S(N,j) \\ &= \frac{\pi S(N,j+1) + (1-\pi)S(N,j)}{R(N-1,j)} R(N-1,j) \\ &= S(N-1,j)R(N-1,j), \end{aligned}$$

where $\pi \stackrel{\text{def}}{=} p(N-1,j)$ is the risk neutral up probability at $n = N-1$ in state j. Conclude that

$$F(N-1,j) = S(N-1,j)R(N-1,j) = G(N-1,j).$$

Remark. Since $F(N,j) = G(N,j) = S(N,j)$ for all states j, both the ultimate ($n = N$) and the penultimate ($n = N-1$) no-arbitrage prices of Forwards and Futures must be equal in all states of a recombining binomial model. □

2. Suppose that S and R are modeled with a recombining binomial tree of $N = 2$ levels. Prove that if $(R(1,1) - R(1,0))(S(1,1) - S(1,0)) > 0$, then $G(0,0) > F(0,0)$.

Solution: Use the results from Exercise 1 above:

$$\begin{aligned} F(N,j) \;=\; G(N,j) &= S(N,j) \\ F(N-1,j) \;=\; G(N-1,j) &= S(N-1,j)R(N-1,j). \end{aligned}$$

Now fix $N = 2$ and suppose that

$$(R(1,1) - R(1,0))(S(1,1) - S(1,0)) > 0.$$

It may be assumed that $R(1,1) > R(1,0) > 0$ and $S(1,1) > S(1,0) > 0$, since these quantities must be positive, relabeling the up and down states if necessary. But then with the above equations,

$$G(1,1) = R(1,1)S(1,1) > R(1,0)S(1,0) = G(1,0)$$

Write $\pi \stackrel{\text{def}}{=} p(0,0)$, the risk neutral up probability, to compute

$$G(0,0) = \pi G(1,1) + (1-\pi)G(1,0),$$

using Eq.5.8. Likewise, compute the Forward price by backward induction for S and D with that π:

$$\begin{aligned}
F(0,0,2) &= S(0,0)/D(0,0) \\
&= \frac{[\pi S(1,1) + (1-\pi)S(1,0)]/R(0,0)}{[\pi D(1,1) + (1-\pi)D(1,0)]/R(0,0)} \\
&= \frac{\pi S(1,1) + (1-\pi)S(1,0)}{\pi D(1,1) + (1-\pi)D(1,0)} \\
&= \frac{\pi S(1,1) + (1-\pi)S(1,0)}{\pi/R(1,1) + (1-\pi)/R(1,0)} \\
&= \frac{\pi G(1,1)/R(1,1) + (1-\pi)G(1,0)/R(1,0)}{\pi/R(1,1) + (1-\pi)/R(1,0)} \\
&= \theta G(1,1) + (1-\theta)G(1,0),
\end{aligned}$$

where

$$\begin{aligned}
\theta &\stackrel{\text{def}}{=} [\pi/R(1,1)]/[\pi/R(1,1) + (1-\pi)/R(1,0)] \\
&= \pi R(1,0)/[\pi R(1,0) + (1-\pi)R(1,1)] \\
&< \pi,
\end{aligned}$$

since the denominator is between $R(1,0)$ and $R(1,1) > R(1,0)$. Thus also

$$1 - \theta > 1 - \pi.$$

Note that $\{\theta, 1-\theta\}$ is a probability function with a lesser weight $\theta < \pi$ for $G(1,1,2)$ and greater weight $1-\theta > 1-\pi$ for $G(1,0,2)$. But since $G(1,1,2) > G(1,0,2) > 0$, it follows that

$$\begin{aligned}
F(0,0,2) &= \theta G(1,1,2) + (1-\theta)G(1,0,2) \\
&< \pi G(1,1,2) + (1-\pi)G(1,0,2) \\
&= G(0,0,2),
\end{aligned}$$

so $G(0,0,2) > F(0,0,2)$, as claimed. □

3. Fix $N = 4$ in the function `FwdFut()` defined on p.135.

   (a) Find inputs S and R such that $F(0) > G(0)$, namely `F(1,1)>G(1,1)` in the output of `[F,G]=FwdFut(S,R,N)`.

   (b) Find S and R such that $F(0) < G(0)$, namely `F(1,1)<G(1,1)` in the output of `[F,G]=FwdFut(S,R,N)`.

   Note: To be valid, solutions S and R in parts (a) and (b) must be positive and must satisfy the no-arbitrage condition

$$S(n+1,j) < R(n,j)S(n,j) < S(n+1,j+1)$$

   for all $0 \le n < N$ and all $0 \le j \le n$.

   Solution: First note that the function `pu=RiskNeut(S,R,N)` will return risk neutral up probabilities satisfying

$$0 < \texttt{pu}(\texttt{n}+1,\texttt{j}+1) = \frac{R(n,j)S(n,j) - S(n+1,j)}{S(n+1,j+1) - S(n+1,j)} < 1,$$

   iff the no-arbitrage conditions hold for all $0 \le n < N$ and $0 \le j \le n$. Thus it may be used to check candidate solutions. Roughly speaking, the conditions will hold if $R(n,j) \approx 1$ and $S(n,j)$ has sufficient volatility so that the up and down factors are far from 1.

   Note also that solutions are not unique. For example, if S solves part (a) for some particular R, then so does cS for any $c > 0$ since both F and G are linear functions of S.

   (a) Adding a row to the example on p.135 gives

$$S = \begin{matrix} 100 & & & & \\ 85 & 115 & & & \\ 75 & 100 & 130 & & \\ 65 & 85 & 115 & 150 & \\ 55 & 80 & 100 & 130 & 170 \end{matrix} \qquad R = \begin{matrix} 1.025 & & & \\ 1.03 & 1.02 & & \\ 1.035 & 1.025 & 1.015 & \\ 1.04 & 1.03 & 1.02 & 1.01 \end{matrix}$$

   Enter this into Octave as follows:

```
Sa=[100,0,0,0,0; 85,115,0,0,0; 75,100,130,0,0;
     65,85,115,150,0; 55,80,100,130,170];
Ra=[1.025,0,0,0; 1.03,1.02,0,0; 1.035,1.025,1.015,0;
     1.04,1.03,1.02,1.01];
```

   Check the no-arbitrage conditions by running `RiskNeut(Sa,Ra,4)`, which returns valid probabilities:

```
   0.58333
   0.50200   0.57667
   0.63125   0.58333   0.48429
   0.50400   0.37750   0.57667   0.53750
```

Finally, compute $F(0) = \texttt{Fa}(1,1) = 109.92 > 109.52 = \texttt{Ga}(1,1) = G(0)$:

```
[Fa,Ga]=FwdFut(Sa,Ra,4); Fa(1,1),Ga(1,1)
```

(b) It is suggested by the previous exercises that if S, R solves part (a), then reversing the numbering of the R states will solve part (b). So, put

```
Sb=Sa; Rb=[1.025, 0,     0,     0;
           1.02,  1.03,  0,     0;
           1.015, 1.025, 1.035, 0;
           1.01,  1.02,  1.03,  1.04];
```

Confirm the no-arbitrage conditions with `RiskNeut(Sb,Rb,4)`, and then compute:

```
[Fb,Gb]=FwdFut(Sb,Rb,4); Fb(1,1),Gb(1,1)
```

which returns $F(0) = \texttt{Fb}(1,1) = 110.81 < 111.32 = \texttt{Gb}(1,1) = G(0)$. □

4. Suppose that a commodities exchange wishes to broker Futures contracts on an asset S, expiring in 0.5 years while riskless annual interest rates are expected to remain constant at 0.02%. Under consideration are margin requirements of 20, 30, 50, 80, and 150% of S_0 for each contract.

   (a) Compute the probability of a margin call, which will occur if and only if the margin balance falls below 0, for both Long and Short Futures contracts, with these five margin requirements. Use $N = 6$ time steps and volatilities $\sigma \in \{0.10, 0.15, 0.20, 0.25, 0.30, 0.40, 0.50, 0.70\}$. Tabulate the results and compare Long and Short margin requirements.

   (b) Profile the computation with $N = 6$ and again with $N = 13$ to compare the run times. Include in the count all times above 1% of the total. Compare the run times to test the $O(2^N)$ order of complexity.

   Solution: (a) The following Octave commands use the programs in the textbook to fill tables `TabL` and `TabS` with the Long and Short margin call probabilities, respectively:

```
T=0.5; r=0.02; N=6;  S0=100; % S0>0 is arbitrary
as=[0.20,0.30,0.50,0.80,1.50];  % alphas
vs=[0.10,0.15,0.20,0.25,0.30,0.40,0.50,0.70]; % volatilities
TabL=zeros(length(vs),length(as)); % output table - Longs
TabS=zeros(length(vs),length(as)); % output table - Shorts
for i=1:length(vs)
  v=vs(i);
  for j=1:length(as)
    a=as(j);
    [ML,MS,Pr] = CRRmargin(T,S0,a*S0,a*S0,r,v,N);
    TabL(i,j) = PrNeg(ML, Pr, 1, N);
    TabS(i,j) = PrNeg(MS, Pr, 1, N);
  end
end
```

The results, tabulated by volatility σ and margin deposit multiple, are in the table below:

σ	α (%)	TabL			α (%)		TabS		
(%)	20	30	50	80	20	30	50	80	150
10	0	0	0	0	0	0	0	0	0
15	0.029	0	0	0	0.034	0	0	0	0
20	0.125	0.016	0	0	0.125	0.031	0	0	0
25	0.227	0.033	0	0	0.210	0.119	0.014	0	0
30	0.235	0.121	0	0	0.203	0.113	0.028	0	0
40	0.493	0.182	0.020	0	0.414	0.191	0.105	0.012	0
50	0.508	0.261	0.042	0	0.399	0.399	0.181	0.023	0
70	0.538	0.538	0.177	0	0.622	0.371	0.163	0.163	0.02

Long Futures (`TabL`) and Short Futures (`TabS`) margin call probabilities ($N = 6$).

The probability of a margin call mostly increases with volatility and decreases with increasing margin deposit. This experiment suggests that, to avoid margin calls at the same volatiliy, the Short margin account needs a larger initial deposit than the Long.

(b) To profile the run times, the Octave commands in part (a) may be enclosed as follows:

```
profile off; profile clear; profile on
T=0.5; r=0.02; N=13;  S0=100; % S0>0 is arbitrary
% (insert code from above)
profile off; profshow  % N=13 results
profile clear; profile on
T=0.5; r=0.02; N=6;  S0=100; % S0>0 is arbitrary
% (insert code from above)
profile off; profshow  % N=6 results
```

The results will be machine dependent. One example gave $5.4s$ for $N = 6$ and $590s$ for $N = 13$. The ratio is approximately 110, close to the expected ratio $2^{13}/2^6 = 2^7 = 128$ for $O(2^N)$ complexity. □

5. Suppose that S is a dividend paying stock with a declared dividend amount D at ex-dividend date t^e. Let $S(t^e-)$ denote the closing price of S on the day before the ex-dividend date, and $S(t^e+)$ denote the opening price of S on the ex-dividend date. Show that

$$S(t^e+) = S(t^e-) - D$$

by constructing an arbitrage otherwise.

Solution: If $S(t^e+) > S(t^e-) - D$, then

- buy S at time t^e-: $-S(t^e-)$
- collect D at time t^p: $+D$
- sell S at t^e+: $+S(t^e+)$

Net: $S(t^e+) + D - S(t^e-) > 0$.

Else $S(t^e+) < S(t^e-) - D$, so

- sell S short at time t^e-: $+S(t^e-)$
- pay out D at time t^p: $-D$
- buy S to cover at t^e+: $-S(t^e+)$

Net: $S(t^e-) - D - S(t^e+) > 0$.

In either case there is profit with no risk. Here it is assumed that any borrowing is cost-free since t^e+, t^e-, and t^p are nearly equal. Note that the borrower of shares sold short is responsible for paying any dividend owed to the lender. □

6. Prove Eq.6.9 from Eq.6.8.

   Solution: Substitute $n \leftarrow n+1$ into Eq.6.8 to get

$$I(n+1) = \sum_{k=1}^{\infty} \frac{d_{n+1+k}}{R_{n+1} \cdots R_{n+1+k-1}} = \sum_{k=2}^{\infty} \frac{d_{n+k}}{R_{n+1} \cdots R_{n+k-1}},$$

   where the sums are actually finite because $d_{N+2} = d_{N+3} = \cdots = 0$. It follows that

$$\frac{I(n+1)}{R_n} = \sum_{k=2}^{\infty} \frac{d_{n+k}}{R_n R_{n+1} \cdots R_{n+k-1}} = I(n) - \frac{d_{n+1}}{R_n}.$$

   Adding d_{n+1}/R_n to both sides gives Eq.6.9. □

7. Use the data in Table 6.1 (p.146) and Table 6.2 (p.154) to compute the present value sequence of BAC dividends from December 2014 through March 2021, namely time indices 0–25. Disregard any dividends outside of that time period.

   (a) Find the values quarterly.

   (b) Find the values monthly.

   (c) Plot the two dividend present value sequences, and the dividends themselves, on the same graph.

   Solution: Reuse the Octave commands that produced `BACdivQ`, `ri`, and so on.

   (a) First, create the array `BACdivQ(1:26)`, the sequence of 26 quarterly dividends from Table 6.1, using the code on p.157.

   Next, create the array `ri(1:25)`, the annualized riskless rates starting in December 2014 and sampled quarterly thereafter, using the code on p.158. These are exponentiated to give quarterly returns.

   Finally, apply the recursion formula from Lemma 6.4 to find the quarterly present value sequence `IpvQ`:

```
% Part (a):
Na=24;  IpvQ=zeros(1,Na+1); % allocate the output array
RQ=exp(ri/4);  % annualized rates ==> quarterly returns
% Backward recursion for IpvQ
IpvQ(Na+1)=BACdivQ(Na+2)/RQ(Na+1); % ignore later dividends
for n = (Na-1):(-1):0        % textbook times {Na-1,...,1,0}
  IpvQ(n+1)=(IpvQ(n+2)+BACdivQ(n+2))/RQ(n+1); % from Lemma
end
```

Remark. These computations are all done in the function `CRRmD()` which may be reused as follows:

```
T=6; Na=24; S0=17.90; sigma=0.1779;
[S,Sx,IpvQ]=CRRmD(T,S0,BACdivQ,ri,sigma,Na);
```

The values S_0 and σ are not used when computing `IpvQ`. Other nominal values such as $S_0 = \sigma = 1$ will give the same result.

(b) Create the array `BACdivM(1:26)`, the sequence of 72 monthly dividends, using the code on p.158. Create the array `r3i`, the annualized riskless rates replicated monthly, using the code on p.159. These are exponentiated to give monthly returns. Then apply the recursion formula from Lemma 6.4 to find `IpvM`:

```
% Part (b):
Nb=72;  IpvM=zeros(1,Nb+1); % allocate the output array
RM=exp(r3i/12);  % annualized rates ==> monthly returns
% Backward recursion for IpvM
IpvM(Nb+1) = BACdivM(Nb+2)/RM(Nb+1); % ignore later dividends
for n = (Nb-1):(-1):0 % textbook times {Nb-1,...,1,0}
  IpvM(n+1) = (IpvM(n+2)+BACdivM(n+2))/RM(n+1);
end
```

`CRRmD()` may be used instead, as follows, with the nominal values $S_0 = K = \sigma = 1$:

```
T=6; Nb=72; S0=1; sigma=1;
[S,Sx,IpvM]=CRRmD(T,S0,BACdivM,r3i,sigma,Nb);
```

(c) Use the plotting function as in the Octave code below.

```
% Part (c):
tM=0:72; tQ=0:3:72; tD=0:3:75; % time indices
plot(tM,IpvM,"b--", tQ,IpvQ,"r-", tD,BACdivQ,"k*");
xlabel("Months from start of 2015");
ylabel("Dollars"); legend("IpvM","IpvQ","BACdivQ");
title("BAC dividends as cash flow present values");
```

The results may be seen in Figure 1. Note that the quarterly present value sequence lies strictly above the monthly, since it includes the March 2021 dividend that the monthly sequence does not. □

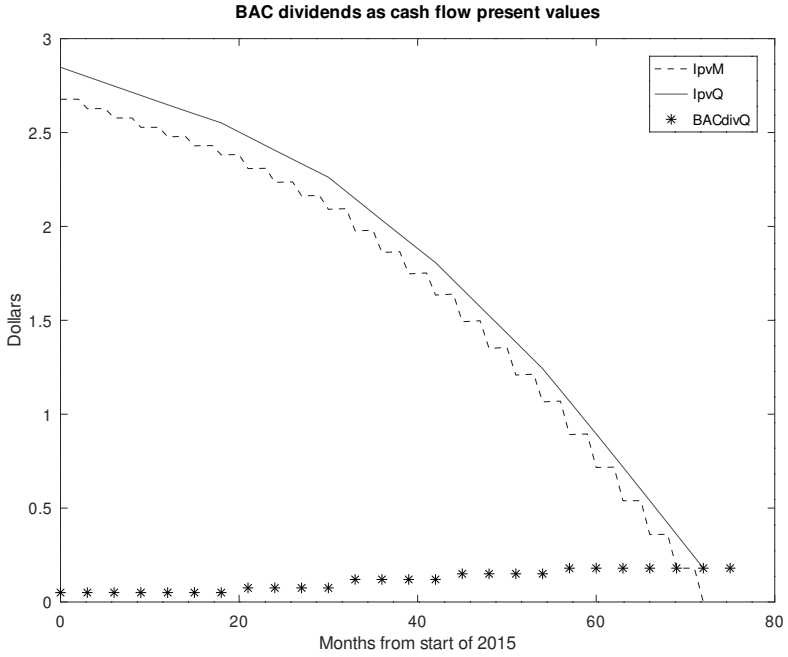


Figure 1: Graph from Exercise 7.

8. For the experiments below, use `CRRDaeC()` with the parameters $T = 1$, $S_0 = 100$, $K = 101$, $r = 0.02$, $v = 0.20$, and $N = 12$.

   (a) Find a nonzero dividend sequence for which early Call exercise is sometimes optimal.

   (b) Find another nonzero dividend sequence for which early Call exercise is never optimal.

   Solution: (a) Sufficiently large dividends will make early exercise optimal. For the given parameters $r = 0.02$, $T = 1$, $K = 101$, and $N = 12$ of this experiment, quarterly \$1 dividends are sufficient:

```
T=1; S0=100; K=101; r=0.02; v=0.20, N=12;
Diq1=[0 1 0 0 1 0 0 1 0 0 1 0 0 1]; % Quarterly 1.00
[Ca,Ce,EE]=CRRDaeC(T,S0,K,Diq1,r,v,N);
EE     % shows where early exercise is optimal
Ca>Ce  % shows how early exercise adds value
```

   From `EE` it may be seen that early exercise is optimal at states $(n, j) \in \{(6, 5), (6, 6), (9, 7), (9, 8), (9, 9)\}$ before expiry.

   (b) By the Remark on p.151 concerning Lemma 6.2, very small positive dividends will not make early exercise optimal for American-style Calls.

The threshold is roughly

$$d < (R-1)K = (\exp(rT/N) - 1)K \approx 0.17,$$

So even with quarterly dividends of 0.16, early exercise will never be optimal in this 12-step model. Check this with the previously entered parameters:

```
Dtiny = (exp(r*T/N)-1)*K   % about 0.16847
[Ca,Ce,EE]=CRRDaeC(T,S0,K,Dtiny*Diq1,r,v,N); % Qrtrly Dtiny
EE     % shows that early exercise is never optimal
Ca>Ce  % shows how early exercise adds no value
```

Of course, these solutions are not unique. □

9. Implement CRR pricing for American and European Put options on an asset with dividends, using decomposition into risky ex-dividend and riskless cash flow portions. (Hint: Make a few changes to `CRRDaeC()`.)

   For the experiments below, set $T = 1$, $S_0 = 100$, $K = 101$, $r = 0.02$, $v = 0.20$, and $N = 12$.

   (a) Find a nonzero dividend sequence for which early Put exercise is sometimes optimal.

   (b) Find a nonzero dividend sequence for which early Put exercise is never optimal.

   Solution: Following the hint, change the payoffs from $[S-K]^+$ to $[K-S]^+$ in `CRRDaeC()` to get the CRR pricing function for Puts:

```
function [Pa,Pe,EE] = CRRDaeP(T,S0,K,Di,r,v,N)
% Octave/MATLAB function to price American and
% European Put options on an asset decomposed
% into S=Sx+Dpv by CRRD(), using the CRR model.
% INPUTS:                                  (Example)
%   T =  expiration time                    (1 year)
%   S0 = stock price                          ($100)
%   K =  strike price                         ($101)
%   Di = dividend sequence                (see below)
%   r =  risk-free yield                      (0.02)
%   v =  volatility; must be >0               (0.15)
%   N =  height of the binomial tree            (12)
% OUTPUT:
%   Pa =  price of American Put at all (n,j).
%   Pe =  price of European Put at all (n,j).
%   EE =  Is early exercise optimal at (n,j)?
% EXAMPLE:
%   Di = [0 1 0 0 1 0 0 2 0 0 2 0 0 2]; % Di(k)=D_k
%   [Pa,Pe,EE]=CRRDaeP(1,100,101,Di,0.02,0.15,12);
%
  [pu,up,R]=CRRparams(T,r,v,N); % Use CRR values
```

```
22    [S,Sx,Dpv]=CRRD(T,S0,Di,r,v,N); % decompose
23    Pa=zeros(N+1,N+1); Pe=zeros(N+1,N+1); % outputs
24    EE=zeros(N,N); % early exercise T/F
25    for j = 0:N % to set terminal values at (N,j)
26      xP=K-S(N+1,j+1); % Put payoff at expiry
27      Pa(N+1,j+1) = Pe(N+1,j+1) = max(xP,0);
28    end
29    % Use backward induction pricing:
30    for n = (N-1):(-1):0 % times n={N-1,...,1,0}
31      for j = 0:n % states j={0,1,...,n} at time n
32        % Backward pricing for A and E:
33        bPe=(pu*Pe(n+2,j+2)+(1-pu)*Pe(n+2,j+1))/R;
34        bPa=(pu*Pa(n+2,j+2)+(1-pu)*Pa(n+2,j+1))/R;
35        xP=K-S(n+1,j+1);  % Put exercise value
36        % Set prices at node (n,j):
37        Pe(n+1,j+1)=bPe; % always binomial price
38        Pa(n+1,j+1)=max(bPa,xP); % highest price
39        % Is early exercise optimal?
40        EE(n+1,j+1)=(xP>bPa); % Yes, if xP>bPa
41      end
42    end
43    return; % Pa, Pe, and EE are defined.
44  end
```

(It is a good idea to update the comments, as is done here.)

(a) Quarterly \$1 dividends provide an example where early exercise is sometimes optimal:

```
T=1; S0=100; K=101; r=0.02; v=0.20, N=12;
Diq1 = [0 1 0 0 1 0 0 1 0 0 1 0 0 1]; % Quarterly 1.00
[Pa,Pe,EE]=CRRDaeP(T,S0,K,Diq1,r,v,N);
EE     % shows where early exercise is optimal
Pa>Pe  % shows how early exercise adds value
```

From `EE` it may be seen that early exercise is optimal at states $(n, j) \in \{(10,0),\ldots,(10,4);(11,0),\ldots,(11,5)\}$ before expiry. This is also clear from the premiums: `Pa(1,1)=9.55` versus `Pe(1,1)=9.41`, so the early exercise option is worth a little more.

(b) Since there is no similar result as Lemma 6.2, it is not clear what makes early exercise nonoptimal for Puts. However, with a little experimentation one might discover that a \$1 monthly dividend suffices:

```
Dim1=[0 1 1 1 1 1 1 1 1 1 1 1 1]; % Monthly 1.00
[Pa,Pe,EE]=CRRDaeP(T,S0,K,Dim1,r,v,N);
EE     % all zeros, so early exercise is never optimal
Pa>Pe  % shows how early exercise adds no value
```

Remark. Increasing the monthly dividend, say to \$2, preserves the never-optimal property:

```
[Pa,Pe,EE]=CRRDaeP(T,S0,K,2*Dim1,r,v,N); EE % still all zeros
```

Reducing the monthly dividend to the Call threshold `Dtiny`, on the other hand, has an effect:

```
Dtiny = (exp(r*T/N)-1)*K   % about 0.16847
[Pa,Pe,EE]=CRRDaeP(T,S0,K,Dtiny*Dim1,r,v,N); EE % some ones!
```

The presence of nonzeros in `EE` shows that early exercise is optimal at states $(n, j) \in \{(10, 2), (10, 3), (10, 4), (11, 2), (11, 5)\}$ before expiry. Reducing the monthly dividend still further, say to `Dtiny/2`, adds even more early exercise states. □

10. Suppose that government bonds are available with maturities of 1, 2, 3, and 4 years, with annual coupons for 0.5, 0.6, 0.8, and 1.1% of face value, respectively. Their spot prices are respectively 0.9994, 0.9992, 0.9989, and 0.9985 times face value.

    (a) Compute the zero-coupon bond discounts $Z(0, 1)$, $Z(0, 2)$, $Z(0, 3)$, and $Z(0, 4)$ implied by these inputs.

    (b) Plot the yield curve implied by $\{Z(0, T) : T = 1, 2, 3, 4\}$.

    Solution: For this case, the linear system relating coupon bond discounts to zero coupon bond discounts is nonsingular.

    (a) Write an Octave/MATLAB program to perform the calculation using forward substitution as in Eq.6.17:

```
c=[0.5,0.6,0.8,1.1]/100;  % coupon rates
B=[0.9994,0.9992,0.9989,0.9985]; % spot prices
Z=B; Z(1)=Z(1)/(1+c(1)); % allocate Z, set Z(1)
for i=2:4
 Z(i) = (Z(i)-c(i)*sum(Z(1:i-1)))/(1+c(i));
end
```

    This returns `Z(i)` $= Z(0, i)$ values corresponding to

T	1	2	3	4
$Z(0, T)$	0.99443	0.98731	0.97524	0.95546

    (b) Plot the yield curve with Octave/MATLAB after converting discounts to yields:

```
rtT=@(Z,t,T)log(Z)./(t-T); % convert discounts to yields
plot(1:4,100*rtT(Z,0,1:4)); % use percentages
title("Yield Curve"); xlabel("T (years)"); ylabel("APR (%)");
```

    The results may be seen in Figure 2. □

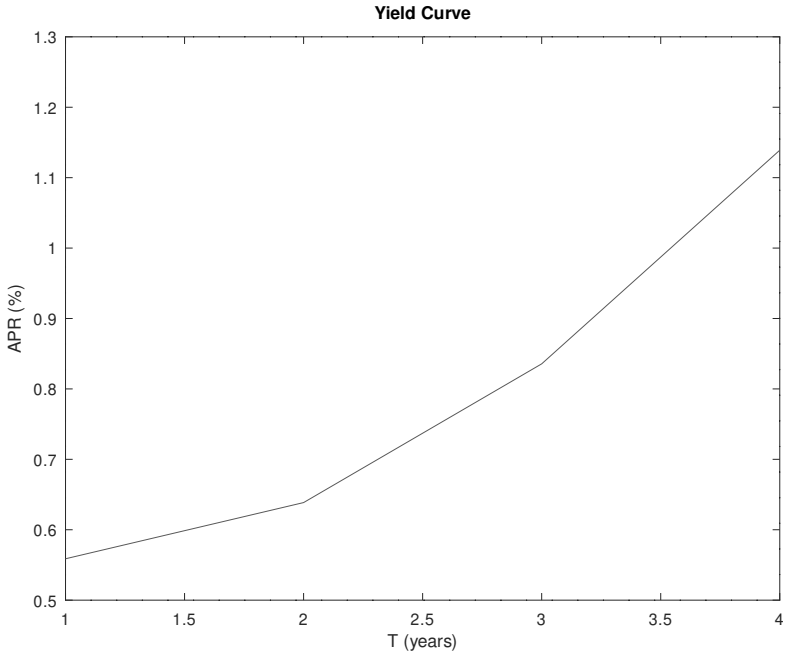


Figure 2: Graph from Exercise 10.