Math 456: Introduction to Financial Mathematics

Homework Set 4

1. Implement a compound Put option, the option to purchase a European-style Put option, with expiry T and strike price K, for price L at time T_1 satisfying $0 < T_1 < T$. Use your code to price such an option with parameters $(T, T_1, S_0, K, L, r, v, N) = (1, 0.4, 97, 99, 1.50, 0.04, 0.23, 20)$. (Hint: modify `CRRcc.m`.)

   Solution: The Octave function `CRRcp()` differs from `CRRcc()` at line 25 and in a few updated comments.

```
1  function [W,P] = CRRcp(T,T1,S0,K,L,r,v,N)
2  %  Octave/MATLAB function to price a compound Put
3  %  in the Cox-Ross-Rubinstein (CRR) binomial model.
4  % INPUTS:                                        (Example)
5  %   T  = expiration time in years               (1)
6  %   T1 = choice time; must have 0<T1<T           (0.5)
7  %   S0 = spot stock price                       (90)
8  %   K =  stock strike price at expiry T         (95)
9  %   L =  option strike price at time T1         (4.50)
10 %   r =  risk-free yield                        (0.02)
11 %   v =  volatility; must be >0                 (0.15)
12 %   N =  height of the binomial tree            (20)
13 % OUTPUT:
14 %   W =  price of the compound Put option
15 %   P =  price of the vanilla European Put option
16 % EXAMPLE:
17 %  [W,P] = CRRcp(1,0.5,90,95,4.50,0.02,0.15,20);
18 %  W(1,1),P(1,1)  % to get just W(0) and P(0)
19 %
20    [pu,up,R]=CRRparams(T,r,v,N); % Use CRR values
21    [C,P]=CRReur(T,S0,K,r,v,N); % vanilla options at T
22    M=round(T1*N/T); % number of time steps to time T1
23    W=zeros(M+1,M+1);    % smaller output matrix
24    for j=0:M  % Set terminal values at expiry T1
25      W(M+1,j+1) = max(P(M+1,j+1)-L,0); % plus part
26    end % ... for a Call option on a vanilla Put
27    for n=M-1:-1:0    % Backward induction
28      for j=0:n  % Binomial pricing model value
29       W(n+1,j+1)=(pu*W(n+2,j+2)+(1-pu)*W(n+2,j+1))/R;
30      end
31    end
```

```
32    return  % Prices in matrices W and P are defined.
33  end
```

Compute the compound Put premium with the requested parameters as follows:

```
[W,P] = CRRcp(1,0.4,97,99,1.50,0.04,0.23,20);
W(1,1),P(1,1)  % 6.5215,  7.9112
```

Conclude that the premium is about \$6.52. □

2. Let $\mathcal{M}(n)$ be the set of path numbers in a recombining binomial tree to depth n, as defined in Eq.4.6. Prove that

$$\mathcal{M}(n+1) = [2\mathcal{M}(n)] \cup [2\mathcal{M}(n)+1]\,,$$

where $aX+b \stackrel{\text{def}}{=} \{ax+b : x \in X\}$ for sets X of numbers.

Solution: Apply the definitions:

$$\begin{aligned} \mathcal{M}(n) &= \{2^n, 2^n+1, \ldots, 2^n+(2^n-1)\}, \\ 2\mathcal{M}(n) &= \{2^{n+1}, 2^{n+1}+2, \ldots, 2^{n+1}+(2^{n+1}-2)\}, \\ 2\mathcal{M}(n)+1 &= \{2^{n+1}+1, 2^{n+1}+3, \ldots, 2^{n+1}+(2^{n+1}-1)\}, \end{aligned}$$

so the union of the last two sets is evidently

$$\mathcal{M}(n+1) = \{2^{n+1}, 2^{n+1}+1, \ldots, 2^{n+1}+(2^{n+1}-1)\},$$

as claimed. □

3. Implement floating strike option pricing in the CRR model using geometric means instead of arithmetic means, as in `CRRgro` versus `CRRaro`. Compare the results on the inputs `(T,S0,r,v,N)=(1,97,0.04,0.25,20)`.

Solution: The Octave function `CRRflg()` differs from `CRRflt()` at lines 21 and 24, which are directly copied from lines 19 and 22 of `CRRgro()` where `lU` is computed and used. It also differs in the use of `Geom` instead of `Avg` and in a few updated comments.

```
1  function [C, P] = CRRflg(T, S0, r, v, N)
2  % Octave/MATLAB function to price floating strike
3  % Call and Put options with geometric means and the
4  % Cox–Ross–Rubinstein (CRR) binomial pricing model.
5  % INPUTS:                                  (Example)
6  %   T  =  expiration time in years          (1)
7  %   S0 =  spot stock price                  (90)
8  %   r  =  riskless yield per year           (0.02)
9  %   v  =  volatility; must be >0            (0.20)
10 %   N  =  height of the tree                (4)
11 % OUTPUTS:
```

```
%     C0 = Call option premium at t=0.
%     P0 = Put option premium at t=0.
% EXAMPLE:
%     [C, P] = CRRflg(1, 90, 0.02, 0.20, 4);
% To get just the premiums at t=0, use
%   C(1), P(1)
%
   [pu,up,R] = CRRparams(T,r,v,N);
   Sbar=NRTCRR(S0,up,1/up,N); % expanded S tree
   lU=NRTpsums(log(Sbar),N); % partial sums of logs
   C=zeros(size(Sbar));   P=zeros(size(Sbar));
   for m=2^N:(2*2^N -1) % m indices at expiry
     Geom = exp(lU(m)/(N+1));  % geometric mean
     C(m) = max(0,Sbar(m)-Geom); % Call payoff
     P(m) = max(0,Geom-Sbar(m)); % Put payoff
   end  % ... prices set at expiry.
   for m=(2^N-1):-1:1 % recursive previous indices
     C(m) = (pu*C(2*m+1) + (1-pu)*C(2*m))/R;
     P(m) = (pu*P(2*m+1) + (1-pu)*P(2*m))/R;
   end % ... all prices set by backward induction.
   return
end
```

Compare (and test for bugs) by running the geometric mean and arithmetic mean functions on the same given inputs:

```
[Cg,Pg]=CRRflg(1,97,0.04,0.25,20);Cg(1),Pg(1) % 6.7866, 4.3492
[Ca,Pa]=CRRflt(1,97,0.04,0.25,20);Ca(1),Pa(1) % 6.4869, 4.5732
```

The respective premiums are similar, though not identical. Note how long this calculation takes. □

4. Use `CRRaro` to compute the average-rate Call and Put premiums in the CRR model for $S_0 = K = 420$, $T = 2$, $r = 0.035$, $v = 0.33$, and different values of N such as $N = 5, 10, 15, 20$. Compare $\overline{C}(0) - \overline{P}(0)$ with the limit value in Eq.4.20.

   Solution: The Octave commands below perform the computations with $N = 5, 10, 15, 20$:

```
r=0.035; T=2; K=420; S0=420; v=0.33;
[C,P]=CRRaro(T,S0,K,r,v,5); N5=C(1)-P(1)    %  N5  =  14.065
[C,P]=CRRaro(T,S0,K,r,v,10); N10=C(1)-P(1)  %  N10 =  14.048
[C,P]=CRRaro(T,S0,K,r,v,15); N15=C(1)-P(1)  %  N15 =  14.043
[C,P]=CRRaro(T,S0,K,r,v,20); N20=C(1)-P(1)  %  N20 =  14.040
limit=S0*(1-exp(-r*T))/(r*T)-K/exp(r*T)     % limit =  14.032
```

The outputs suggest that every $N \geq 15$ achieves an approximation within one cent. □

5. Implement floating strike option pricing in the CRR model using path-dependent Arrow-Debreu securities. Check that the results agree with `CRRflt` on that function's suggested example inputs.

   Solution: The Octave function `CRRfltAD()` differs from `CRRflt()` in three ways, just as `CRRaroAD()` differs from `CRRaro()`:

```
function [C0, P0] = CRRfltAD(T, S0, r, v, N)
% Octave/MATLAB function to price floating strike
% Call and Put options using path-dependent
% Arrow-Debreu expansions with the Cox-Ross-
% Rubinstein (CRR) binomial pricing model.
% INPUTS:                                  (Example)
%   T  =  expiration time in years          (1)
%   S0 =  spot stock price                  (90)
%   r  =  riskless yield per year           (0.02)
%   v  =  volatility; must be >0            (0.20)
%   N  =  height of the tree                (4)
% OUTPUTS:
%    C0 = Call option premium at t=0.
%    P0 = Put option premium at t=0.
% EXAMPLE:
%    [C0,P0]=CRRfltAD(1,90,0.02,0.20,4);
%
  [pu,up,R]=CRRparams(T,r,v,N);
  Sbar=NRTCRR(S0,up,1/up,N); % expanded S tree
  Ubar=NRTpsums(Sbar,N); % all partial sums
  Lbar=PathAD(pu*ones(N,N),R*ones(N,N),N);
  mN=2^N:(2*2^N-1); % all m-indices at expiry
  AvgN=Ubar(mN)/(N+1); % S averages at expiry
  C0=max(0,Sbar(mN)-AvgN)*Lbar(mN)'; % Call
  P0=max(0,AvgN-Sbar(mN))*Lbar(mN)'; % Put
  return % ..inner producs give A-D expansions
end
```

   Compare (and test for bugs) by running both functions on the same suggested example inputs:

```
[C0,P0]=CRRfltAD(1,90,0.02,0.20,4)
[C,P]=CRRflt(1,90,0.02,0.20,4); C(1),P(1)
```

   The outputs agree: `C0=4.4853=C(1)` and `P0=3.5920=P(1)`. □

6. Write an Octave program to compute the maximums along all paths in a non-recombining binary tree of depth N. (Hint: modify `NRTmin()`.) Run both programs on the inputs `N=3` and

```
Sbar=[2 1 3 4 1 0 4 5 3 1 2 4 6 1 2]
```

   Compare the results to confirm that path maxima are at least as large as path minima.

Solution: Let $\bar{S}_{\max}(m)$ be the maximum value of $S(n,j)$ along the path indexed by m. It may be computed from the non-recombining tree $\bar{S}$ by the following recursion:

$$\begin{aligned} \bar{S}_{\max}(1) &\stackrel{\text{def}}{=} \bar{S}(1), \\ \bar{S}_{\max}(2m) &= \max\{\bar{S}_{\max}(m), \bar{S}(2m)\}, \\ \bar{S}_{\max}(2m{+}1) &= \max\{\bar{S}_{\max}(m), \bar{S}(2m{+}1)\}. \end{aligned} \tag{1}$$

Eq.1 is easily implemented in Octave. Following the hint, start with `NRTmin()` and modify lines 15 and 16, replacing `min` with `max`.

```
1  function Maxb = NRTmax(Sbar, N)
2  % Octave/MATLAB function to compute maximums
3  % along paths in a non-recombining tree (NRT).
4  % INPUTS:                                  (Example)
5  %   Sbar = NRT array of length 2*2^N-1      (1:15)
6  %   N = tree depth, must be >=0               (3)
7  % OUTPUT:
8  %   Maxb = NRT of maximums, same size as Sbar
9  % EXAMPLE:
10 %  Maxb = NRTmax(1:15,3)
11 %
12   Maxb=zeros(size(Sbar)); % allocate the output
13   Maxb(1)=Sbar(1); % max along the trivial path
14   for m=1:2^N-1   % all future times up to N-1
15     Maxb(2*m)=max(Maxb(m),Sbar(2*m));    % down
16     Maxb(2*m+1)=max(Maxb(m),Sbar(2*m+1)); % up
17   end
18   return
19 end
```

Run the two programs as follows:

```
Sbar=[2 1 3 4 1 0 4 5 3 1 2 4 6 1 2];  N=3;
Maxb=NRTmax(Sbar,N); Maxb # 2 2 3 4 2 3 4 5 4 2 2 4 6 4 4
Minb=NRTmin(Sbar,N); Minb # 2 1 2 1 1 0 2 1 1 1 1 0 0 1 2
Maxb-Minb    # nonegative:  0 1 1 3 1 3 2 4 3 1 1 4 6 3 2
```

□

7. Implement lookback option pricing in the CRR model using path-dependent Arrow-Debreu securities. Check that the results agree with `CRRlb` on the inputs `(T,S0,r,v,N)=(1.5,213,0.04,0.21,10)`.

   Solution: The Octave function `CRRlbAD()` differs from `CRRlb()` in three ways, just as `CRRaroAD()` differs from `CRRaro()`:

```
1  function [C0,P0] = CRRlbAD(T,S0,r,v,N)
2  % Octave/MATLAB function to price Lookback Call
3  % and Put options using path-dependent Arrow-Debreu
```

```
% expansions with the Cox-Ross-Rubinstein (CRR)
% binomial pricing model.
% INPUTS:                                    (Example)
%   T  =  expiration time in years            (1)
%   S0 =  spot stock price                    (100)
%   r  =  riskless yield per year             (0.05)
%   v  =  volatility; must be >0              (0.20)
%   N  =  height of the tree                  (4)
% OUTPUTS:
%    C0 = Call option premium at t=0
%    P0 = Put option premium at t=0
% EXAMPLE:
%    [C0,P0]=CRRlb(1,100,0.05,0.20,4)
%
   [pu,up,R]=CRRparams(T,r,v,N);
   Sbar=NRTCRR(S0,up,1/up,N); % expanded S tree
   MinS=NRTmin(Sbar,N); MaxS=NRTmax(Sbar,N);
   Lbar=PathAD(pu*ones(N,N),R*ones(N,N),N);
   mN=2^N:(2*2^N-1); % all m-indices at expiry
   C0=(Sbar(mN)-MinS(mN))*Lbar(mN)'; % Call
   P0=(MaxS(mN)-Sbar(mN))*Lbar(mN)'; % Put
   return % ..inner producs give A-D expansions
end
```

Compare (and test for bugs) by running both functions on the same suggested example inputs:

```
[C0,P0]=CRRlbAD(1.5,213,0.04,0.21,10); C0, P0
[C,P]=CRRlb(1.5,213,0.04,0.21,10); C(1),P(1)
```

The outputs agree: `C0=C(1)=39.632` and `P0=P(1)=31.180`. □

8. Implement ladder Put option pricing in the CRR model by modifying `CRRladC` appropriately. Check that the premium is at least as great as that for the vanilla European-style Put with the same parameters, using the example inputs suggested within `CRRladC`.

   Solution: For the ladder Put option, there is a specified strike price K and a decreasing ladder $\{L_i\}$ of in-the-money prices below K:

$$S(0) \geq \min\{S(0), K\} > L_1 > \cdots > L_k.$$

Let $M(T) = \min\{S(t) : 0 \leq t \leq T\}$ be the minimum price of S up to expiry. Then the payoff is

$$P(T) \stackrel{\text{def}}{=} \max\{[K - S(T)]^+,\ K - L_i\}, \tag{2}$$

where L_i is the lowest rung reached by $S(t)$ as recorded by $M(T)$:

$$L_{i+1} < M(T) \leq L_i.$$

In the discrete CRR implementation with N time steps to expiry, both S and M are stored as linear arrays with m indexing. The paths of length N are indexed by $m \in \mathcal{M}(N) = \{2^N, \ldots, 2^N + (2^N - 1)\}$, so those are the indices in the non-recombining tree P where terminal values are set using Eq.2. Then the option premium is found by backward induction as usual:

$$P(m) = \frac{pP(2m+1) + (1-p)P(2m)}{R}, \quad m = 2^N - 1, 2^N - 2, \ldots, 2, 1,$$

where p is the risk neutral up probability from the CRR model, and R is the riskless return per time step.

```
function LadP = CRRladP(T,S0,K,L,r,v,N)
% Octave/MATLAB function to price a ladder Put
% option using a Cox-Ross-Rubinstein (CRR) model.
% INPUTS:                                (Example)
%   T  = expiration time in years        (1)
%   S0 = spot price                      (50)
%   K  = strike price                    (55)
%   L  = decreasing prices < S0,K        ([45,40,35])
%   r  = riskless yield per year         (0.05)
%   v  = volatility; must be >0          (0.20)
%   N  = height of the tree              (4)
% OUTPUTS:
%   LadP = Ladder Put option price array.
% EXAMPLE:
%   LadP=CRRladP(1,50,55,[45,40,35],0.05,0.20,4)
%
  [pu,up,R] = CRRparams(T,r,v,N);
  Sbar=NRTCRR(S0,up,1/up,N); % S as an NRT
  MinS=NRTmin(Sbar,N); % non-recombining min tree
  LadP=zeros(size(Sbar)); % Put prices NRT
  % Initialize with the payoffs at expiry:
  k=length(L);          % number of ladder levels, >1
  for m=2^N:2*2^N-1  % state indexes at expiry
    if(MinS(m)>L(1)) % above level L(1) is special
      LadP(m)=max(K-Sbar(m),0);
    else % MinS =< L(1)
      if(MinS(m)>L(k)) % ...use levels L(2)>...>L(k)
        for l=2:k       % loop to find the level
          if(MinS(m)>L(l))
            LadP(m)=max(max(K-Sbar(m),K-L(l-1)),0);
            break; % found l: L(l)<MinS=<L(l-1),
          end
        end         % ... so exit the "l" loop
      else % MinS=<L(k), another special case
        LadP(m)=max(max(K-Sbar(m),K-L(k)),0);
      end
    end
  end
  for m=(2^N-1):-1:1 % backward recursion
```

```
40        LadP(m)=(pu*LadP(2*m+1)+(1-pu)*LadP(2*m))/R;
41      end
42      return % LadP(1) is the option premium
43  end
```

Note that, as with the Call,

$$P(T) \geq [K - S(T)]^+ \geq 0,$$

so this exotic option cannot cost less than a vanilla European Put with the same parameters. A few experiments will show how big the differences can be, for example,

```
LadP=CRRladP(1,50,55,[45,40,35],0.05,0.20,4);LadP(1) % 5.8784
[eC,eP]=CRReur(1,50,55,0.05,0.20,4);eP(1,1)          % 5.5372
```

The returned values round to \$5.88 for the ladder Put versus \$5.54 for the vanilla European Put, ordered as expected. □