Aidan Ashrafi
Final Project: MATH 456

ANALYSIS

QUESTION 1

- Spot price at market close 4:00 pm EDT, on December 12th,2023, S0 = 352.61
- Dates chosen for near-future expiry, 12/15/23, 12/22/23
- Strike Prices chosen for near-the-money strike prices are 340, 342.5, 345, 347.5, 350, 352.5, 355, 357.5, 360

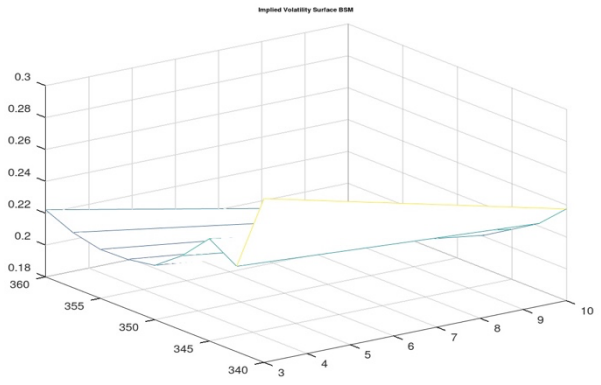
Rationale (Dates): The dividend date is December 28th, and the ex-dividend date is November 29th. Therefore, I have chosen dates such that no near-future-expiry option chain chosen has expected dividend payments. The reason for this is that because of these selected dates, the European Style Call Algorithm can be used for American Style Call without losing much accuracy. The principle of early exercise being optimal because of potential dividends goes away. Consequently, the value of American-style options converge to European-style options. So without expected dividends, it is clear that we are ok to make this approximation. Furthermore, the output below shows that using CRReur or CRRa produces the same implied volatility surface. Therefore, no dividend adjustment is necessary given the dates chosen.

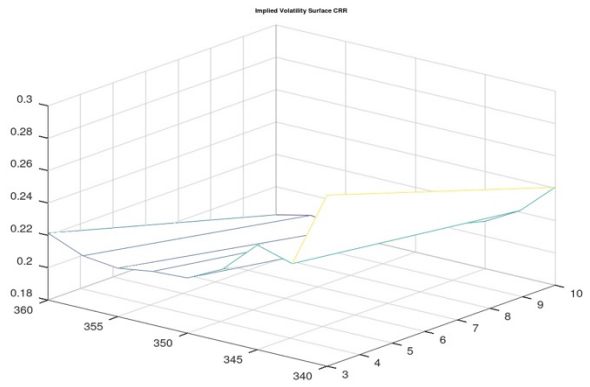
Rationale (Strike Prices): I chose these near-the-money strike prices because they have high trading volume, and are sensitive to changes in the underlying stock price. This means they hold a good representation of the underlying asset, and are responsive to short-term market changes. For my riskless rate, I averaged the 52-week rate over the past 5 days, as the options chosen expire within 52 weeks. Then I took a daily average and applied a scalar of 12 to encompass both rates with a closer range than 4 weeks.

(See Appendix #1 for Code)

Output:

```
VimpCRR2_equal_VimpCRR =

   1   1
   1   1
   1   1
   1   1
   1   1
   1   1
   1   1
   1   1
   1   1

VimpBS =

    0.2827    0.2375
    0.2336    0.2218
    0.2442    0.2113
    0.2268    0.2009
    0.2141    0.1947
    0.2111    0.1895
    0.2107    0.1866
    0.2146    0.1851
    0.2221    0.1846

VimpCRR =

    0.2857    0.2405
    0.2383    0.2210
    0.2447    0.2093
    0.2246    0.2030
    0.2141    0.1923
    0.2131    0.1915
    0.2099    0.1843
    0.2125    0.1874
    0.2214    0.1826

discrepancy =

   3.0057e-03   2.9608e-03
   4.7104e-03  -7.3273e-04
   4.7852e-04  -2.0262e-03
  -2.2281e-03   2.0935e-03
   5.2338e-05  -2.3701e-03
   1.9141e-03   1.9739e-03
  -8.0750e-04  -2.2804e-03
  -2.1234e-03   2.3477e-03
  -7.3273e-04  -1.9440e-03
```





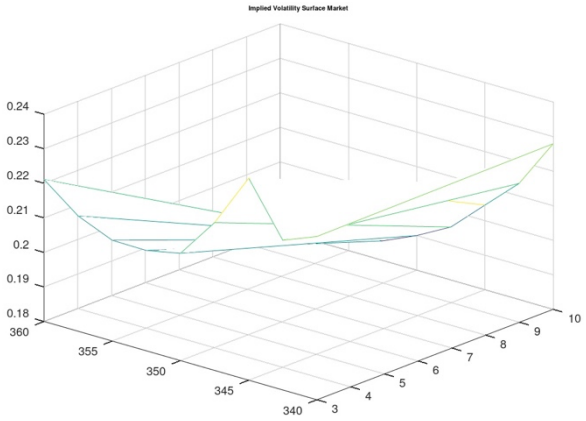
These outputs suggest, what we originally declared, which is that we can use CRReur as an approximation for CRRa in the specific context with no expected dividend payments in the options chain before the time to expiry. Furthermore, these outputs suggest the values for implied volatility from CRR and BSM are nearly identical. Therefore, as expected the two methods produced nearly identical values for implied volatility.

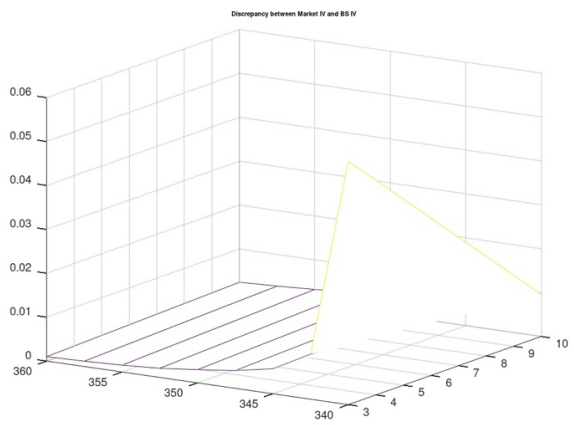
Now we will check to see how close our predicted values are to the market values for implied volatility.

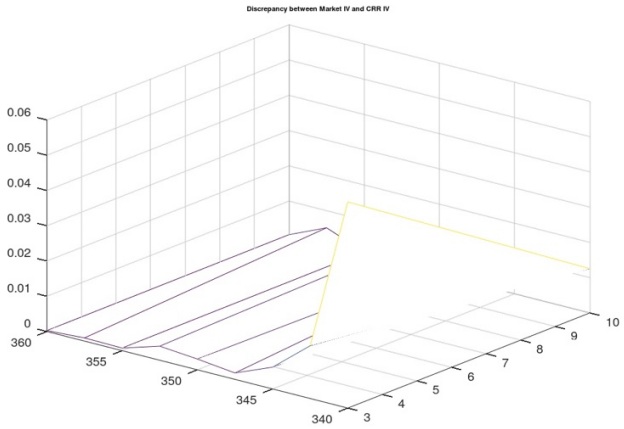
(See Appendix #2 for Code)

Output:

```
Market_Discrepancy =

   -5.5459e-02   -9.5891e-03
   -1.0254e-02   -8.1206e-03
   -5.6985e-03   -6.7855e-03
   -4.0423e-03   -5.6127e-03
   -3.0069e-03   -4.6696e-03
   -2.0811e-03   -3.8856e-03
   -1.5524e-03   -3.2771e-03
   -1.2203e-03   -2.7920e-03
   -9.9953e-04   -2.4235e-03
```







The output shows that the calculated implied volatility is very close to the market values, all deviations are in the range [0, 0.0569].
Thus, the model is decently sufficient at finding IV using various near-the-money strike prices K and near-future expiry dates T. However, the model has limitations, and there could be various reasons for the small discrepancy in the values.

- Using a midpoint average of bid and ask price, although reliable and useful for theoretical models, is not exact
- Riskless returns were estimated based on a daily average of the 52-week rate over the past 5 days, as the options chosen expire within 52 weeks with an applied scalar of 12 to encompass both rates. This is not exact
- Data was taken from 2 options chains (expiry containing no expected dividends) so predictive precision is limited

As a result, the small discrepancy between the IV from the market and BSM and CRR models is not unreasonable and fairly justified.

QUESTION 2:
Need to choose a time to expiry that includes at least 2 expected dividend payments. Based on the data above, we can safely say a time to expiry that fits this criterion is a time to expiry past July 1st, as on June 29th of last year the second set of dividends was paid out.
Accordingly, we will take a look at July 19th Options Chain Data from December 12th after market close.

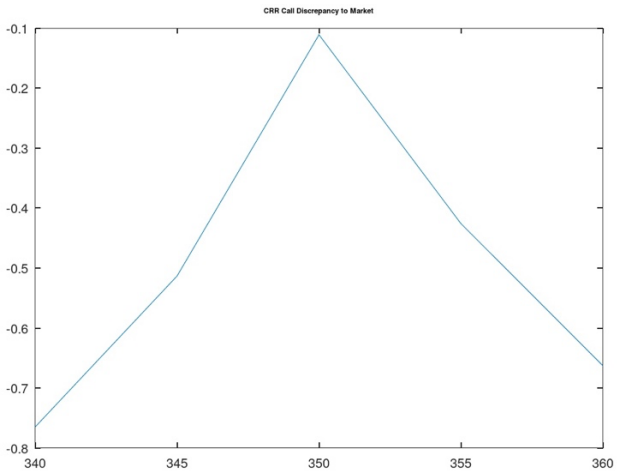
(See Appendix #3 for Code)
Further Rationale Necessary:

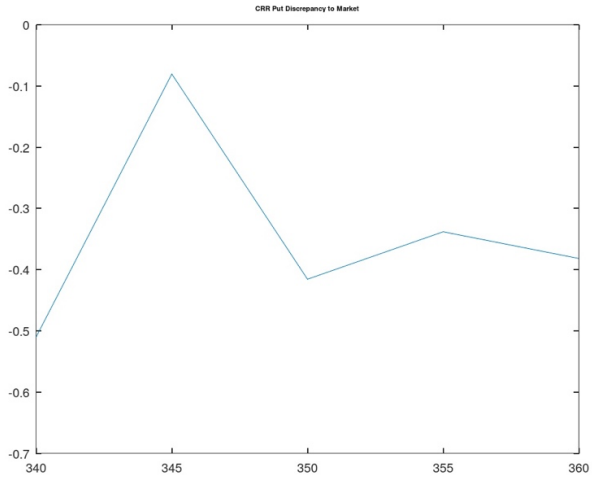
- We are going to interpolate between the 6-month and yearly rates to get a riskless rate for 220 days (r), our desired expiry. This is the number of days from December 12th, market closes at 4:00 pm EDT to July 19th, our expiry date.
- To get T, we divide the number of days to expiry by 365 to get the fraction of the year that the option falls under
- We need to ensure we take the present value of the historical dividends
- (1+r) is raised to the power of the percentage of the total duration til expiry has been forgone
- Div1 is paid out 109 days after Dec 12th market close, so 109/220
- Div2 is paid out 201 days after Dec 12th market close, so 201/220
- We must also make sure we allocate it correctly in the payment sequence
- We have N = 8, and we have T = 0.60207, With length of Di = N+2
- Div1 is paid out at 49% of time to expiry
- Div2 is padi out at 95% of time to expiry

Therefore, we can consider Div1 to be approximately at the 5th position (exact is between 4th and 5th position).
We can consider Div2 to be approximately at the 9th position (exact is between 9th and 10th position).
We are going to use the implied volatility from the December 12th market close for the July 19th options chain here.
Given further reasoning,
(See Appendix #4 for Code)

Output:

CRR_calls =	CRR_puts =	Market_call =	Market_put =	Call_diff =	Put_diff =
35.790	16.985	35.025	16.475	-0.7651	-0.509999
32.413	18.505	31.900	18.425	-0.5132	-0.080474
29.286	20.791	29.175	20.375	-0.1108	-0.415574
26.551	22.863	26.125	22.525	-0.4258	-0.338104
24.163	25.507	23.500	25.125	-0.6627	-0.381752





Analysis:
All deviations in Call & Put prices from the market, are in the range [0,0.7651]. This is a larger range the discrepancy we got for the first question.

Rationale:
Many factors are potentially causing this discrepancy:

- Because of the treatment of dividend payments, and the option chosen's time to expiry, T = 0.6027 (220/365), the best N = 8, but the timing of dividend payments is still approximated, not exact
- Therefore, the CRR model's predictive capabilities are limited and not as accurate, as with larger N, the model becomes more precise
- For my riskless rate, I interpolated between the 6-month T bill rate and 1 yr. T bill rate, for 220 or 7.23287 months, this is not exact, but it is a good proxy
- For my call prices, I used a midpoint of the bid and ask price, which is a good estimate, but may not always hold an accurate representation

Therefore, after considering all factors that influenced the model's predictive power, the deviations from the actual market prices are not unreasonable.

To answer the next part of Question 2, we will apply the assumption, observe the effects, and make sure that they align with expectations.

(See Appendix #5 for Code)

Output:

```
>> CRR_calls > CRR_calls_inc_second_div
ans =

  1
  1
  1
  1
  1
```

- This output shows that the CRR call option prices are less for the scenario with the second dividend being 20% higher than the first.
- Therefore, we can conclude our Call Price would decrease if the second dividend was 20% higher than the first.
- This is because an increase in dividends reduces the future value of the underlying stock, so in particular call option prices could decrease.
- This potential decrease is evident through the comparison of the call prices with different dividends

(See Appendix #6 for Code)

Output:

```
>> CRR_puts_inc_second_div > CRR_puts
ans =

  1
  1
  1
  1
  1
```

- This output shows that the CRR Put option prices are greater for the scenario with the second dividend being 20% higher than the first
- Therefore, we can conclude our Put Price would increase if the second dividend was 20% higher than the first
- This is because an increase in dividends reduces the future value of the underlying stock, so in particular put option prices could increase
- This potential increase is evident through the comparison of the put prices with different dividends

Therefore, the call prices would decrease and the put prices would increase if the second dividend was 20% higher than the first. Furthermore, these results agree with expectations, which is a good sign for the model's functionality.

QUESTION 3:

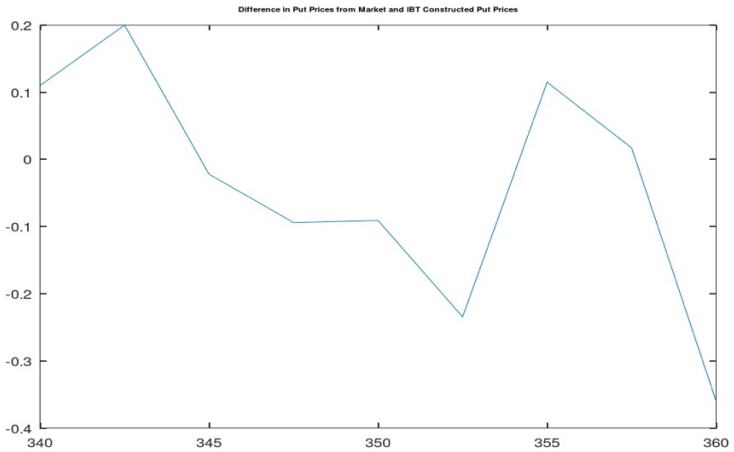
For this question, need to choose a time to expiry that does not include expected dividends. It is after the ex-dividend date, before the dividend date, so after November 29th, and before December 28th. The date chosen is December 15th.

(See Appendix #7 for Code)

Output:

```
S =

   352.5678         0         0         0         0         0         0
   348.2701   356.8491         0         0         0         0         0
   346.2262   352.4041   359.0739         0         0         0         0
   344.7437   350.7383   354.1168   360.7276         0         0         0
   343.6800   349.4855   352.3778   355.5337   361.9201         0         0
   342.9469   348.4867   351.0015   353.8873   356.5973   362.8017         0
   342.5000   347.5000   350.0000   352.5000   355.0000   357.5000   363.4451


pu =

   0.5039         0         0         0         0         0         0
   0.3349    0.6703         0         0         0         0         0
   0.2515    0.5006    0.7538         0         0         0         0
   0.1875    0.4419    0.5591    0.8174         0         0         0
   0.1368    0.4072    0.4857    0.6170    0.8621         0         0
   0.0943    0.4048    0.4107    0.5651    0.6492    0.8962         0
        0         0         0         0         0         0         0
```

```
PutPrices_IBT_1 =

       0       0  0.3920  0.7840  1.4210  2.4744  3.9150  5.7328  7.8299

Market_PutPrices =

   0.1100   0.2000   0.3700   0.6900   1.3300   2.2400   4.0300   5.7500   7.4700

Market_Put_diff =

   0.110000   0.200000  -0.021987  -0.093974  -0.090953  -0.234418   0.115030   0.017191  -0.359940
```



As we can see, all the deviations are in the range [0, 0.36]
The reasons for these deviations include:

- Estimated a risk-free rate using an average of the 4-week T-bill rates, still accounting for the number of days till expiry using rho. This is an estimate
- Used the midpoint between bid and ask prices for call options from the market
- Used a specific weight function, w=@(x),x: general linear weight function
- The relatively small number of options chain data points (restraints because of no dividend payments in the timeframe)
- Used last price for market value for comparison

Based on these reasons, the estimates are within a reasonable range.
For the last part of Question 3, we must compare the IBT put prices with the put prices given from the call-put parity.
(See Appendix # 8 for Code)
Output:

```
PutPrices_CallPutParity =

   0.1677   0.1416   0.5155   0.8645   1.4634   2.4873   3.9062   5.7101   7.7990

Put_differential =

   1.6770e-01   1.4162e-01   1.2355e-01   8.0482e-02   4.2420e-02   1.2872e-02  -8.7634e-03  -2.2686e-02  -3.0899e-02
```

As we can see the max difference between the call-put parity put price and the calculated put price from the binomial tree using w= @(x)x, is 0.1677. Therefore, all deviations in the put prices are in the range [0,0.1677].

The reasons for these deviations include:

- Estimated a risk-free rate using an average of the 4-week T-bill rates, still accounting for the number of days till expiry using rho. This is an estimate
- Used the midpoint between bid and ask prices for call option prices from the market
- Used a specific weight function, w=@(x),x: general linear weight function
- The relatively small number of options chain data points (restraints because of no dividend payments in the timeframe)

This deviation is therefore reasonable, and the estimates are therefore reasonable as well.

Appendix on Next Page

APPENDIX (Entire Project in Octave):
https://drive.google.com/file/d/1i23yF9La3V8UmAJoho8IXUdzJfLtgWv-/view?usp=share_link

Appendix #1:

```
S0 = 352.61;
r = (0.0482 + 0.0480 + 0.0487 + 0.0488 + 0.0488)/5;
r = 0.048500*12/365;
% We will use these functions to find the implied volatility
function [a,b]=bisection(f,y,a,b,tol)
  do
    m=(a+b)/2;        % midpoint
    if(f(m)<y) a=m;   % then use [m,b]
    else b=m;         % else use [a,m]
    end
  until(abs(b-a)<tol) % close enough!
  return
end

function [pu,up,R] = CRRparams(T,r,v,N)

  dt = T/N;       % one time step of N in [0,T]
  up = exp(v*sqrt(dt)); % up factor, will be >1
  down = 1/up;        % down factor, will be <1
  R = exp(r*dt);  % riskless return over dt
  pu = (R-down)/(up-down); % risk-neutral up prob.
  return; % Parameters are all computed
end

function [C, P] = CRReur(T, S0, K, r, v, N)
  [pu,up,R] = CRRparams(T,r,v,N); % Use CRR values
  C=zeros(N+1,N+1); P=zeros(N+1,N+1); % Initial C,P
  for j=0:N  % Set terminal values at time step N
   SNj=S0*up^(2*j-N);        % Expiry price S(N,j)
   C(N+1,j+1)=max(SNj-K, 0); % Plus part of (SNj-K)
   P(N+1,j+1)=max(K-SNj, 0); % Plus part of (K-SNj)
  end
  for n=N-1:-1:0   %  Price earlier grid values
   for j=0:n % ...with the backward pricing formula
    C(n+1,j+1)=(pu*C(n+2,j+2)+(1-pu)*C(n+2,j+1))/R;
    P(n+1,j+1)=(pu*P(n+2,j+2)+(1-pu)*P(n+2,j+1))/R;
   end
  end
  return; % Prices in C and P are now fully defined.
end

function [C, P] = CRRa(T, S0, K, r, v, N)
% Octave/MATLAB function to price American Call
% and Put options using the Cox-Ross-Rubinstein
% (CRR) binomial pricing model.
```

```
% INPUTS:                          (Example)
%   T =  expiration time             (1 year)
%   S0 = spot stock price              (90)
%   K =  strike price                  (95)
%   r =  risk-free yield              (0.02)
%   v =  volatility; must be >0       (0.15)
%   N =  height of the binomial tree    (10)
% OUTPUT:
%   C =  price of the Call option at all (n,j).
%   P  =  price of the Put option at all (n,j).
% EXAMPLE:
%   [C,P] = CRRa(1, 90, 95, 0.02, 0.15, 10);
%   C(1,1),P(1,1)  % to get just C(0) and P(0)
%
 [pu,up,R] = CRRparams(T,r,v,N); % Use CRR values
 C=zeros(N+1,N+1); P=zeros(N+1,N+1); % Initial C,P
 for j = 0:N  % Set terminal values at time step N
   V = S0*up^(2*j-N)-K;   % S(N,j) - K
   C(N+1,j+1)=max(V,0); P(N+1,j+1) = max(-V,0);
 end
 for n = (N-1):(-1):0 % Price earlier grid values
   for j = 0:n  % states j={0,1,...,n} at time n
     % Binomial pricing model value
     bC=(pu*C(n+2,j+2) + (1-pu)*C(n+2,j+1))/R;
     bP=(pu*P(n+2,j+2) + (1-pu)*P(n+2,j+1))/R;
     % Exercise value: X for Call, -X for Put
     X = S0*up^(2*j-n)-K;  % S(n,j) - K
     % Price at this node is the larger:
     C(n+1,j+1)=max(bC,X); P(n+1,j+1)=max(bP,-X);
   end
 end
 return; % Prices are in matrices C and P.
end

function [C0,P0] = BS(T,S0,K,r,v)
 normcdf = @(x) 0.5*(1.0+erf(x/sqrt(2.0)));
 d1 = (log(S0/K)+T*(r+v^2/2)) / (v*sqrt(T));
 d2 = (log(S0/K)+T*(r-v^2/2)) / (v*sqrt(T));
 % Alternatively, d2=d1-v*sqrt(T);
 C0 = S0*normcdf(d1)-K*exp(-r*T)*normcdf(d2);
 P0 = K*exp(-r*T)*normcdf(-d2)-S0*normcdf(-d1);
 return
end

Ks = 340:2.5:360; Ts = [3,10];
C=[(12.45 + 13.40)/2,  (13.75 + 14.10)/2;...
 (9.80   + 11.00)/2, (11.50       + 11.85)/2;...
(8.10     + 8.45)/2, (9.45         + 9.75)/2;...
(5.95     + 6.30)/2,  (7.50        + 7.80)/2;...
```

```
(4.10   + 4.35)/2, (5.85        + 6.05)/2;...
(2.70   + 2.80)/2, (4.40        + 4.55)/2;...
(1.64   + 1.70)/2, (3.20        + 3.35)/2;...
(0.93   + 1.02)/2, (2.27        + 2.39)/2;...
(0.53   + 0.60)/2, (1.57        + 1.66)/2];
VimpBS=zeros(length(Ks),length(Ts)); % BS implied volatilities
    VimpCRR=zeros(length(Ks),length(Ts)); % CRR American implied volatility
     VimpCRR2=zeros(length(Ks),length(Ts)); % CRR European implied volatility
    minv=0.01; maxv=0.99; tol=0.00001; % bisection parameters
    for col=1:length(Ts)
      T=Ts(col)/365; % time to expiry in years
      for row=1:length(Ks)
        f=@(v) BS(T,S0,Ks(row),r,v); % Black-Scholes Call
        VimpBS(row,col)=bisection(f,C(row,col),minv,maxv,tol);
        g=@(v) CRRa(T,S0,Ks(row),r,v,20)(1,1); % CRR American Style Call
        VimpCRR(row,col)=bisection(g,C(row,col),minv,maxv,tol);
        g=@(v) CRReur(T,S0,Ks(row),r,v,20)(1,1); % CRR European Style Call
        VimpCRR2(row,col)=bisection(g,C(row,col),minv,maxv,tol);
      end
    end

mesh(Ts,Ks,VimpBS); % note the transposed order (T,K,V(K,T))
title("Implied Volatility Surface BSM");

mesh(Ts,Ks,VimpCRR); % note the transposed order (T,K,V(K,T))
title("Implied Volatility Surface CRR");
VimpCRR2_equal_VimpCRR = VimpCRR2 == VimpCRR % (==1) IF TRUE
VimpBS
VimpCRR
discrepancy= VimpCRR – VimpBS;
discrepancy
```

Appendix #2:

```
VimpMarket = [0.22728, 0.22796;...
0.22334,0.21363 ;...
0.23849, 0.20452;...
0.2228, 0.19533;...
0.21108, 0.19003;...
0.20906, 0.18564;...
0.20917, 0.18334;...
0.21339, 0.18227;...
0.22111, 0.18213];
Market_Discrepancy = VimpMarket - VimpBS;
Market_Discrepancy
Max(abs(Market_Discrepancy))
% Graph of Market IV
mesh(Ts,Ks, VimpMarket)
title("Implied Volatility Surface Market")
% Graph of Discrepancy IV with BS
mesh(Ts,Ks,abs(Market_Discrepancy))
```

```
title("Discrepancy between Market IV and BS IV")
% Graph of Discrepancy IV with CRR
mesh(Ts,Ks,abs(VimpMarket-VimpCRR))
title("Discrepancy between Market IV and CRR IV")
```

Appendix #3:

```
% We will use these functions to price Calls and Puts for an American-style option with
expected dividend payments
function [pu,up,R] = CRRparams(T,r,v,N)

  dt = T/N;       % one time step of N in [0,T]
  up = exp(v*sqrt(dt)); % up factor, will be >1
  down = 1/up;        % down factor, will be <1
  R = exp(r*dt);  % riskless return over dt
  pu = (R-down)/(up-down); % risk-neutral up prob.
  return; % Parameters are all computed
end

function [S, Sx, Ipv] = CRRD(T, S0, GSdivQ, r, v, N)
% Octave/MATLAB function to model an asset price
% as a risky ex-dividend price Sx plus a riskless
% dividend cash flow present value Ipv, using the
% Cox-Ross-Rubinstein (CRR) binomial pricing model.
% INPUTS:                          (Example)
%   T  =  expiration time              (1 year)
%   S0 =  spot stock price               (100)
%   Di =  2:N+2 dividends in a vector (see below)
%   r  =  constant risk-free annual yield  (0.02)
%   v  =  volatility; must be >0          (0.15)
%   N  =  height of the binomial tree       (12)
% OUTPUTS:
%   S =  cum-dividend binomial tree at all (n,j).
%   Sx =  ex-dividend binomial tree at all (n,j).
%   Ipv =  dividend present value at all n.
% EXAMPLE:
%   Di=[0 1 0 0 1 0 0 2 0 0 2 0 0 2]; % Di(k+1)=d_k
%   [S,Sx,Ipv]=CRRD(1,100,Di, 0.02, 0.15, 12);
%
  [pu,up,R] = CRRparams(T,r,v,N); % Use CRR values
  Sx=zeros(N+1,N+1); S=zeros(N+1,N+1);
  Ipv=zeros(1,N+1); % ...allocate output matrices
  Ipv(N+1)=GSdivQ(N+2)/R; % ignore Di(n+1)=d_n if n>N+1
  for n=(N-1):(-1):0  % textbook time indices
    Ipv(n+1)=(Ipv(n+2)+GSdivQ(n+2))/R; % Di(n)=d_{n+1}
  end  % ...backward recursion for Ipv.
  Sx(1,1)=S0-Ipv(1); S(1,1)=S0; % initial values
  for n=1:N     % textbook time indices
    for j=0:n   % states j={0,1,...,n} at time n
      Sx(n+1,j+1)=Sx(1,1)*up^(2*j-n);
      S(n+1,j+1)=Sx(n+1,j+1)+Ipv(n+1);
```

```
   end
  end  % ...forward recursion for Sx and S.
  return; % Prices are in S, Sx, and Ipv.
end

function [Ca,Ce,EE] = CRRDaeC(T,S0,K,GSdivQ, r,v,N)
% Octave/MATLAB function to price American and
% European Call options on an asset decomposed
% into S=Sx+Ipv by CRRD(), using the CRR model.
% INPUTS:                          (Example)
%   T =  expiration time             (1 year)
%   S0 = stock price                   (100)
%   K =  strike price                  (101)
%   Di = dividend sequence 2:N+2    (see below)
%   r =  annualized risk-free yield    (0.02)
%   v =  volatility; must be >0        (0.15)
%   N =  height of the binomial tree     (12)
% OUTPUT:
%   Ca =  price of American Call at all (n,j).
%   Ce =  price of European Call 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+1)=D_k
%   [Ca,Ce,EE]=CRRDaeC(1,100,101,Di,0.02,0.15,12);
%
  [pu,up,R]=CRRparams(T,r,v,N); % Use CRR values
  [S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N); % decompose
  Ca=zeros(N+1,N+1); Ce=zeros(N+1,N+1); % output
  EE=zeros(N,N); % early exercise T/F
  for j = 0:N  % to set terminal values at (N,j)
    xC=S(N+1,j+1)-K; % Call payoff at expiry...
    Ca(N+1,j+1)=Ce(N+1,j+1)=max(xC,0); % plus part
  end
  for n = (N-1):(-1):0 % textbook time indices
    for j = 0:n % states j={0,1,...,n} at time n
      % Backward pricing for A and E:
      bCe=(pu*Ce(n+2,j+2)+(1-pu)*Ce(n+2,j+1))/R;
      bCa=(pu*Ca(n+2,j+2)+(1-pu)*Ca(n+2,j+1))/R;
      xC=S(n+1,j+1)-K; % Call exercise value
      % Set prices at node (n,j):
      Ce(n+1,j+1)=bCe; % always binomial price
      Ca(n+1,j+1)=max(bCa,xC); % highest price
      % Is early exercise optimal?
      EE(n+1,j+1)=(xC>bCa); % Yes, if xC>bCa
    end
  end % ...backward induction pricing:
  return; % Ca,Ce, and EE are defined.
end
```

```
function [Pa,Pe,EE] = CRRDaeP(T,S0,K,GSdivQ,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
 [S,Sx,Dpv]=CRRD(T,S0,GSdivQ,r,v,N); % decompose
 Pa=zeros(N+1,N+1); Pe=zeros(N+1,N+1); % outputs
 EE=zeros(N,N); % early exercise T/F
 for j = 0:N % to set terminal values at (N,j)
   xP=K-S(N+1,j+1); % Put payoff at expiry
   Pa(N+1,j+1) = Pe(N+1,j+1) = max(xP,0);
 end
 % Use backward induction pricing:
 for n = (N-1):(-1):0 % times n={N-1,...,1,0}
   for j = 0:n % states j={0,1,...,n} at time n
     % Backward pricing for A and E:
     bPe=(pu*Pe(n+2,j+2)+(1-pu)*Pe(n+2,j+1))/R;
     bPa=(pu*Pa(n+2,j+2)+(1-pu)*Pa(n+2,j+1))/R;
     xP=K-S(n+1,j+1);  % Put exercise value
     % Set prices at node (n,j):
     Pe(n+1,j+1)=bPe; % always binomial price
     Pa(n+1,j+1)=max(bPa,xP); % highest price
     % Is early exercise optimal?
     EE(n+1,j+1)=(xP>bPa); % Yes, if xP>bPa
   end
 end
 return; % Pa, Pe, and EE are defined.
End
% r4 = (5.16 + 5.14 + 5.17 + 5.17 + 5.18)/5; 26 wk rate, 6 mnth rate averaged
% r5= (4.82 + 4.80 + 4.87 + 4.88 + 4.88)/5; 52 wk rate, yearly rate averaged
%r4= 0.051640
%r5 = 0.048500
```

Appendix #4:

```
% Call Price At K = 340
T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6));% Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = 2.75/(1+r)^0.95;
K = 340; N = 8; v= 0.2413;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
present_value = Ipv(1)  % present value of dividend cash flow
ex_div_price = Sx(1,1) % ex-dividend spot price.
[Ca1,Ce1,EE]=CRRDaeC(T,S0,K,GSdivQ,r,v,N);
Ca1(1,1),Ce1(1,1) % 0.29852, 0.28688 ==> early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_340 = Ca1>Ce1 %  TRUE (=1) where American-style > European-style

% Call Price At K = 345
T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); % Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = 2.75/(1+r)^0.95;
K = 345; N = 8; v= 0.2398;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Ca2,Ce2,EE]=CRRDaeC(T,S0,K,GSdivQ,r,v,N);
Ca2(1,1),Ce2(1,1)  %early exercise premium
EE   %  TRUE (=1) shows where early exercise is optimal
k_345 = Ca2>Ce2 %  TRUE (=1) where American-style > European-style

% Call Price At K = 350
T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); % Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = 2.75/(1+r)^0.95;
K = 350; N = 8; v= 0.23746;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Ca3,Ce3,EE]=CRRDaeC(T,S0,K,GSdivQ,r,v,N);
Ca3(1,1),Ce3(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_350 = Ca3>Ce3 %  TRUE (=1) where American-style > European-style

% Call Price At K = 355
T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); % Interpolation
```

```
Div1 = 2.75/(1+r)^0.49545455;
Div2 = 2.75/(1+r)^0.95;
K = 355; N = 8; v= 0.22954 ;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Ca4,Ce4,EE]=CRRDaeC(T,S0,K,GSdivQ,r,v,N);
Ca4(1,1),Ce4(1,1)  %early exercise premium
EE   %  TRUE (=1) shows where early exercise is optimal
k_355 = Ca4>Ce4 %  TRUE (=1) where American-style > European-style

% Call Price At K = 360
T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); %
Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = 2.75/(1+r)^0.95;
K = 360; N = 8; v= 0.22517;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Ca5,Ce5,EE]=CRRDaeC(T,S0,K,GSdivQ,r,v,N);
Ca5(1,1),Ce5(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_360 = Ca5>Ce5 %  TRUE (=1) where American-style > European-style

% Put Price At K = 340
T=0.6027; S0=352.61; r = 0.0541 + (((0.0514 - 0.0541)/(12-6)) * (7.23287-6)); % Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = 2.75/(1+r)^0.95;
K = 340; N = 8; v= 0.2374;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1)
Sx(1,1)
[Pa1,Pe1,EE]=CRRDaeP(T,S0,K,GSdivQ,r,v,N);
Pa1(1,1),Pe1(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_340 = Pa1>Pe1 %  TRUE (=1) where American-style > European-style

% Put Price At K = 345
T=0.6027; S0=352.61; r = 0.0541 + (((0.0514 - 0.0541)/(12-6)) * (7.23287-6)); % Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = 2.75/(1+r)^0.95;
K = 345; N = 8; v= 0.2355;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
```

```
Sx(1,1);
[Pa2,Pe2,EE]=CRRDaeP(T,S0,K,GSdivQ,r,v,N);
Pa2(1,1),Pe2(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_345 = Pa2>Pe2 %  TRUE (=1) where American-style > European-style

% Put Price At K = 350
T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); %
Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = 2.75/(1+r)^0.95;
K = 350; N = 8; v= 0.2327;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Pa3,Pe3,EE]=CRRDaeP(T,S0,K,GSdivQ,r,v,N);
Pa3(1,1),Pe3(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_350 = Pa3>Pe3 %  TRUE (=1) where American-style > European-style

% Put Price At K = 355
T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); %
Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = 2.75/(1+r)^0.95;
K = 355; N = 8; v= 0.2243;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Pa4,Pe4,EE]=CRRDaeP(T,S0,K,GSdivQ,r,v,N);
Pa4(1,1),Pe4(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_355 = Pa4>Pe4 %  TRUE (=1) where American-style > European-style

% Put Price At K = 360
T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); %
Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = 2.75/(1+r)^0.95;
K = 360; N = 8; v= 0.2216;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Pa5,Pe5,EE]=CRRDaeP(T,S0,K,GSdivQ,r,v,N);
Pa5(1,1),Pe5(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
```

```
k_360 = Pa5>Pe5 %  TRUE (=1) where American-style > European-style

% Store Call & Put prices for comparison with the Market
Ca1 = Ca1(1,1);
Ca2 = Ca2(1,1);
Ca3 = Ca3(1,1);
Ca4 = Ca4(1,1);
Ca5 = Ca5(1,1);
Pa1 = Pa1(1,1);
Pa2 = Pa2(1,1);
Pa3 = Pa3(1,1);
Pa4 = Pa4(1,1);
Pa5 = Pa5(1,1);
% Calls
CRR_calls = [Ca1;...
 Ca2;...
 Ca3;...
 Ca4;...
 Ca5];  % American Style
% Puts
CRR_puts = [Pa1;...
 Pa2;...
 Pa3;...
 Pa4;...
 Pa5];  % American Style

% Store Market Call & Put Prices for comparison with CRR
% Calls
Market_call= [ (34.40 +          35.65)/2;...
(31.60  + 32.20)/2;...
(28.65  + 29.70)/2;...
(25.90  + 26.35)/2;...
(23.30  + 23.70)/2];
% Puts
Market_put = [(16.20  + 16.75)/2;...
(18.10  + 18.75)/2;...
(20.05  + 20.70)/2;...
(22.30  + 22.75)/2;...
(24.60  + 25.65)/2];

Call_diff = Market_call - CRR_calls
Put_diff = Market_put - CRR_puts
Kw = 340:5.0:360;
plot(Kw,Call_diff)
title("CRR Call Discrepancy to Market");


plot(Kw,Put_diff)
title("CRR Put Discrepancy to Market");
```

```
max(abs(Call_diff))
max(abs(Put_diff))
```

Appendix #5:

```
% Second Dividend Payment = Div1*1.2 : Assumption for third part of Question 2
T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); %
Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = Div1*1.2;
K = 340; N = 8; v= 0.2413;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Ca,Ce,EE]=CRRDaeC(T,S0,K,GSdivQ,r,v,N);
Ca(1,1),Ce(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_340 = Ca>Ce %  TRUE (=1) where American-style > European-style

T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); %
Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = Div1*1.2;
K = 345; N = 8; v= 0.2398;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Ca1,Ce1,EE]=CRRDaeC(T,S0,K,GSdivQ,r,v,N);
Ca1(1,1),Ce1(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_345 = Ca1>Ce1 %  TRUE (=1) where American-style > European-style

T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); %
Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = Div1 * 1.2;
K = 350; N = 8; v= 0.23746;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Ca2,Ce2,EE]=CRRDaeC(T,S0,K,GSdivQ,r,v,N);
Ca2(1,1),Ce2(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_350 = Ca2>Ce2 %  TRUE (=1) where American-style > European-style

T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); %
Interpolation
```

```
Div1 = 2.75/(1+r)^0.49545455;
Div2 = Div1 * 1.2;
K = 355; N = 8; v= 0.22954;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Ca3,Ce3,EE]=CRRDaeC(T,S0,K,GSdivQ,r,v,N);
Ca3(1,1),Ce3(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_355 = Ca3>Ce3 %  TRUE (=1) where American-style > European-style

T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); %
Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = Div1 * 1.2;
K = 360; N = 8; v= 0.22517;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Ca4,Ce4,EE]=CRRDaeC(T,S0,K,GSdivQ,r,v,N);
Ca4(1,1),Ce4(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_360 = Ca4>Ce4 %  TRUE (=1) where American-style > European-style

Ca = Ca(1,1);
Ca1 = Ca1(1,1);
Ca2 = Ca2(1,1);
Ca3 = Ca3(1,1);
Ca4 = Ca4(1,1);
CRR_calls_inc_second_div = [Ca;...
 Ca1;...
 Ca2;...
 Ca3;...
 Ca4];  % American Style
CRR_calls > CRR_calls_inc_second_div
% Will Print 1 if the Call Prices with the second dividend being 20% higher than the first is less
than the Call Prices with PV dividends
```

Appendix #6:

```
% Put Price At K = 340
T=0.6027; S0=352.61; r = 0.0541 + (((0.0514 - 0.0541)/(12-6)) * (7.23287-6)); % Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = Div1*1.2;
K = 340; N = 8; v= 0.2374 ;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
```

```
[Pa1,Pe1,EE]=CRRDaeP(T,S0,K,GSdivQ,r,v,N);
Pa1(1,1),Pe1(1,1)  %early exercise premium
EE  %  TRUE (=1) shows where early exercise is optimal
k_340 = Pa1>Pe1 %  TRUE (=1) where American-style > European-style

% Put Price At K = 345
T=0.6027; S0=352.61; r = 0.0541 + (((0.0514 - 0.0541)/(12-6)) * (7.23287-6)); % Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = Div1*1.2;
K = 345; N = 8; v= 0.2355 ;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Pa2,Pe2,EE]=CRRDaeP(T,S0,K,GSdivQ,r,v,N);
Pa2(1,1),Pe2(1,1)  %early exercise premium
EE   %  TRUE (=1) shows where early exercise is optimal
k_345 = Pa2>Pe2 %  TRUE (=1) where American-style > European-style

% Put Price At K = 350
T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); %
Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = Div1*1.2;
K = 350; N = 8; v= 0.2327;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Pa3,Pe3,EE]=CRRDaeP(T,S0,K,GSdivQ,r,v,N);
Pa3(1,1),Pe3(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_350 = Pa3>Pe3 %  TRUE (=1) where American-style > European-style

% Put Price At K = 355
T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); %
Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = Div1*1.2;
K = 355; N = 8; v=0.2243 ;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Pa4,Pe4,EE]=CRRDaeP(T,S0,K,GSdivQ,r,v,N);
Pa4(1,1),Pe4(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_355 = Pa4>Pe4 %  TRUE (=1) where American-style > European-style
```

```
% Put Price At K = 360
T=0.6027; S0=352.61; r = 0.051640 + (((0.04850 - 0.051640)/(12-6)) * (7.23287-6)); %
Interpolation
Div1 = 2.75/(1+r)^0.49545455;
Div2 = Div1*1.2;
K = 360; N = 8; v= 0.2216;
GSdivQ= [0, 0, 0, 0, Div1, 0, 0, 0, Div2, 0];
[S,Sx,Ipv]=CRRD(T,S0,GSdivQ,r,v,N);
Ipv(1);
Sx(1,1);
[Pa5,Pe5,EE]=CRRDaeP(T,S0,K,GSdivQ,r,v,N);
Pa5(1,1),Pe5(1,1)  %early exercise premium
EE    %  TRUE (=1) shows where early exercise is optimal
k_360 = Pa5>Pe5 %  TRUE (=1) where American-style > European-style

Pa1 = Pa1(1,1);
Pa2 = Pa2(1,1);
Pa3 = Pa3(1,1);
Pa4 = Pa4(1,1);
Pa5 = Pa5(1,1);
CRR_puts_inc_second_div = [Pa1;...
 Pa2;...
 Pa3;...
 Pa4;...
 Pa5];  % American Style
CRR_puts_inc_second_div > CRR_puts
% Will Print 1 if the Put Prices with the second dividend being 20% higher than the first is
greater than original Put Prices
```

Appendix #7:

```
% We will use this function to compute an IBT
  function [S,Q,up,down,pu,N1]=IBT123J(S0,Ks,Cs,rho,w)
% Octave/MATLAB function to compute an implied binomial
% tree from spot prices and market European-style Call
% option ask prices, by Rubinstein's "1,2,3" method
% with Jackwerth's generalization.
% INPUTS:                                  (Example)
%   S0= spot stock price     (BAC 2021/12/11:    44.52)
%   Ks= equispaced, increasing strike prices    (40:46)
%   Cs= Call asks   ([5.6,4.8,4.1,3.45,2.87,2.29,1.81])
%   rho= riskless return (to 2022/3/20:(exp(r*109/365))
%   w= weight function                        (w=@(x)x)
% OUTPUT:
%   S = binomial tree of stock prices at all (n,j)
%   Q = tree of probabilities of all states (n,j)
%   up = upfactors at (n,j)
%   down = downfactors at (n,j)
%   pu = risk neutral up probabilities at (n,j)
%   N1 = depth of the pruned tree
% NOTE: since Octave array indices start at 1, output
```

```
%   array value for (n,j) is stored at index (n+1,j+1).
% EXAMPLE:
%   S0=44.22; Ks=40:46; % BAC spot price and strikes
%   Cs=[5.6,4.8,4.1,3.45,2.87,2.29,1.81]; % Call asks
%   r=0.06/100; rho=exp(r*109/365); % 13-wk T-bill APR
%   [S,Q,up,down,pu,N1] = IBT123J(S0,Ks,Cs,rho,w)
%
m = length(Ks);        % expect valid Ks(1),...,Ks(m)
% Check that Ks and Cs have the same length
if(length(Cs) != m ) % expect valid Cs(1),...,Cs(m)
  error ("length(Cs)!=length(Ks)");
end
N=m+1;           % initial tree depth, before pruning
Del=Ks(2)-Ks(1); % expect Del=Ks(j)-Ks(j-1), all j
if(!(Del>0))     % expect Ks(1)<Ks(2)<...
  error("must have Ks(1)<Ks(2)")
end
for j=3:m          % expect equispaced Ks
 if( Ks(j)!= Ks(j-1)+Del )
   error("must have equispaced Ks")
 end
end
%% Generate time-T risk neutral probabilities Q(N,j):
QN=ones(1,N+1); % so QN(j+1)=Q(N,j) for j=0,...,N=m+1
for j=2:m-1    % normalized butterfly spread premiums:
  QN(j+1)=rho*(Cs(j-1)-2*Cs(j)+Cs(j+1))/Del;
end
QN(2)=0; QN(m+1)=0;   % Q(N,1) and Q(N,m) are special
QN(N+1)=rho*(Cs(m-1)-Cs(m))/Del; % Q(N,N) is special
QN(1)=1-sum(QN(2:(N+1)));  % so sum(Q(N,j))=1, as prob.
%% Generate the initial bottom row of stock prices
SN=zeros(1,N+1);  % so SN(j+1)=S(N,j) for the initial tree
b=(Cs(2)-Cs(1))/Del;   % temporary "slope" variable
SN(1)=rho*(S0-Cs(1)+Ks(1)*b)/(1+rho*b); % S(N,0) is special
SN(2:m+1)=Ks(1:m);  % S(N,j)=Ks(j) for j=1:m
SN(N+1)=Ks(m)+Del*Cs(m)/(Cs(m-1)-Cs(m)); % S(N,N) special
%% Prune the bottom row of S to use only positive Qs:
Q1=QN(QN>0);      %  subvector of the positive Q(N,j)
S1=SN(QN>0);      % ...and the corresponding S(N,j)
N1=length(Q1)-1;  % depth of the pruned tree
R=rho^(1/N1);     % new one-step riskless return
%% Initialize and assign the output binomial trees
S = zeros(N1+1,N1+1);    % array of stock prices.
Q = zeros(N1+1,N1+1);    % array of path probabilities.
pu = zeros(N1+1,N1+1);   % array of one-step up probs.
up = zeros(N1+1,N1+1);   % up factors
down = zeros(N1+1,N1+1); % down factors
S(N1+1,:)=S1; % Last row of pruned stock price tree
Q(N1+1,:)=Q1; % Last row of state probabilities tree
```

```
% Backward induction to fill the trees
for n=N1-1:-1:0
  for j=0:n
    Qup=w((j+1)/(n+1))*Q(n+2,j+2);
    Qdown=(1-w(j/(n+1)))*Q(n+2,j+1);
    Q(n+1,j+1)=Qup+Qdown;
    pu(n+1,j+1) = Qup/Q(n+1,j+1); % Eq.7.11
    S(n+1,j+1) = (pu(n+1,j+1)*S(n+2,j+2)
                + (1-pu(n+1,j+1))*S(n+2,j+1))/R;
    up(n+1,j+1) = S(n+2,j+2)/S(n+1,j+1);   % upfactor
    down(n+1,j+1) = S(n+2,j+1)/S(n+1,j+1); % downfactor
  end
end
return;
end
Ks=340:2.5:360; Cs=[(12.45  + 13.40)/2 (9.80 + 11.00)/2 (8.10        + 8.45)/2 (5.95          +
6.30)/2 (4.10   + 4.35)/2 (2.70+ 2.80)/2 (1.64+ 1.70)/2 (0.93+ 1.02)/2 (0.53+ 0.60)/2];
S0=352.61; r=0.052720; rho=exp(r*3/365); w=@(x)x;w1 = @(x)sqrt(x); w2= @(x)x.^2; w3=@(x)(1-
cos(pi*x))/2;
[S,Q,up,down,pu,N1]=IBT123J(S0,Ks,Cs,rho,w); S, pu
[S1,Q1,up,down,pu1,N1]=IBT123J(S0,Ks,Cs,rho,w1); S1, pu1
[S2,Q2,up,down,pu2,N1]=IBT123J(S0,Ks,Cs,rho,w2); S2, pu2
[S3,Q3,up,down,pu3,N1]=IBT123J(S0,Ks,Cs,rho,w3); S3, pu3

Ti= 3/365;
N1=6;
S_avg = (S + S1 + S2 + S3 )/4;
pu_avg = (pu + pu1 + pu2 + pu3)/4;
pu1 = pu1;
pu2 = pu2;
pu3 = pu3;

R = exp(r * Ti/N1); % Define the one-step riskless rate
PutPrices_IBT = zeros(1, length(Ks)); % To store Put prices
for idx = 1:length(Ks)
   K = Ks(idx); % Select a strike price
   PutPayoff = max(K - S_avg(end, :), 0); % Payoff at expiry
   % Backward induction through the tree
   for n = N1-1:-1:0
      for j = 0:n
         ExpectedPayoff = (pu_avg(n+1,j+1) * PutPayoff(j+2) + ...
                  (1 - pu_avg(n+1,j+1)) * PutPayoff(j+1)) / R;
         PutPayoff(j+1) = ExpectedPayoff;
      end
   end

   PutPrices_IBT(idx) = PutPayoff(1); % Store the calculated price
end
% Put Prices using average of 4 different weight functions
```

```
PutPrices_IBT

R = exp(r * Ti/N1); % Define the one-step riskless rate
PutPrices_IBT_1 = zeros(1, length(Ks)); % To store Put prices
for idx = 1:length(Ks)
    K = Ks(idx); % Select a strike price
    PutPayoff = max(K - S(end, :), 0); % Payoff at expiry
    % Backward induction through the tree
    for n = N1-1:-1:0
        for j = 0:n
            ExpectedPayoff = (pu(n+1,j+1) * PutPayoff(j+2) + ...
                              (1 - pu(n+1,j+1)) * PutPayoff(j+1)) / R;
            PutPayoff(j+1) = ExpectedPayoff;
        end
    end

    PutPrices_IBT_1(idx) = PutPayoff(1); % Store the calculated price
end
%  Put Prices using linear weight function, w = @(x)x
PutPrices_IBT_1
 % Market Put Prices
Market_PutPrices = [0.11, 0.20,       0.37,    0.69,    1.33,  2.24,     4.03,    5.75, 7.47]; % Last
price as of December 12th, 2023 market close for prices in Ks
% Market Put Price Differential with general linear weight, w=@(x)x;
Market_Put_diff = Market_PutPrices - PutPrices_IBT_1
% Market Put Price Differential with average of 4 different weight functions, w=@(x)x;w1 =
@(x)sqrt(x); w2= @(x)x.^2; w3=@(x)(1-cos(pi*x))/2;
Market_Put_diff_w_avg = Market_PutPrices - PutPrices_IBT
% As we can see, using an average of the 4 weights, will not always bring us closer to the
market put price
abs(Market_Put_diff_w_avg) >= abs(Market_Put_diff)
% In fact, this average is only a better predictor of market put price for 1 near-the-money strike
price
% Therefore, although good for comparison, and understanding, the put prices using an
average of the weights, will not be preferred to the general linear weight function, w = @(x)x

% Largest difference between predicted and market
max_deviation = max(abs(Market_Put_diff))
% Graphical Representation of Difference in Put Prices from Market and IBT Put Prices with
respect to various strike prices
plot(Ks, Market_Put_diff)
title("Difference in Put Prices from Market and IBT Constructed Put Prices");
```

Appendix #8:

```
% Initialize array for call-put parity put prices
PutPrices_CallPutParity = zeros(1, length(Ks));
r = 0.052720;
t=3/365;
Ks;
```

```
Cs;
% Calculate and compare
for idx = 1:length(Ks)
    K = Ks(idx);
    C = Cs(idx);  % Call price for the respective strike price
    P_CallPutParity = C - (S0 - K * exp(-r * t));
    PutPrices_CallPutParity(idx) = P_CallPutParity;
end


% Now we will show the results for comparison
PutPrices_CallPutParity  % Put prices from call-put parity
PutPrices_IBT_1          % Put prices from binomial tree using general linear weight, w=@(x)x;
Put_differential = PutPrices_CallPutParity - PutPrices_IBT_1
max(Put_differential)
```

BASIC PROJECT INFORMATION
Publicly Traded Stock: Goldman Sachs
Stock Symbol: GS
Exchange: NYSE

SOURCES FOR DATA COLLECTION
https://finance.yahoo.com/quote/GS/key-statistics?p=GS
https://finance.yahoo.com/quote/GS/options?p=GS
https://www.nasdaq.com/market-activity/stocks/gs/dividend-history
https://home.treasury.gov/resource-center/data-chart-center/interest-rates/TextView?type=daily_treasury_bill_rates&field_tdr_date_value=2023
https://www.nasdaq.com/market-activity/stocks/gs/option-chain-greeks
https://www.nasdaq.com/market-activity/stocks/gs/option-chain

RISKLESS RATES DATA
US Treasury Bill Rates for the previous 5 days will be used to determine the riskless rate.

	4 wk	8 wk	13 wk	17 wk	26 wk	52 wk
12/06/2023	5.27	5.25	5.25	5.23	5.16	4.82
12/07/2023	5.26	5.27	5.24	5.23	5.14	4.80
12/08/2023	5.28	5.27	5.24	5.25	5.17	4.87
12/11/2023	5.28	5.28	5.26	5.25	5.17	4.88
12/12/2023	5.27	5.28	5.26	5.25	5.18	4.88

DIVIDEND INFORMATION
Dividend Yield = 0.0313
Annual Dividend = 11.00
Ex-Dividend Date = 11/29/2023
Dividend Date = 12/28/2023
Dividends paid out every 3 months, last 4 = 11/29/2023, 8/30/2023, 5/31/2023, 3/01/2023

Ex/Ef Date	Type	Amnt	Declare D	Record D	Payment D
11/29/2023	Cash	$2.75	10/12/2023	11/30/2023	12/28/2023
08/30/2023	Cash	$2.75	07/19/2023	08/31/2023	09/28/2023
05/31/2023	Cash	$2.50	04/18/2023	06/01/2023	06/29/2023
03/01/2023	Cash	$2.50	01/13/2023	03/02/2023	03/30/2023
11/30/2022	Cash	$2.50	10/17/2022	12/01/2022	12/29/2022
08/31/2022	Cash	$2.50	07/14/2022	09/01/2022	09/29/2022
05/31/2022	Cash	$2.00	04/13/2022	06/01/2022	06/29/2022

OPTIONS CHAINS

December 15th, Data									
	Call: Bid	Ask	Vol	OpnInt	K	Put: Bid	Ask	Vol	OpnInt
Dec 15	12.45	13.40	58	1247	340.00	0.10	0.14	404	809
Dec 15	9.80	11.00	4	344	342.50	0.17	0.22	89	537
Dec 15	8.10	8.45	13	710	345.00	0.34	0.40	678	595
Dec 15	5.95	6.30	17	305	347.50	0.67	0.76	453	156
Dec 15	4.10	4.35	137	1960	350.00	1.29	1.41	202	562
Dec 15	2.70	2.80	258	435	352.50	2.29	2.40	82	96
Dec 15	1.64	1.70	697	1972	355.00	3.70	3.85	30	159
Dec 15	0.93	1.02	328	985	357.50	3.90	5.75	--	1
Dec 15	0.53	0.60	1272	3027	360.00	7.50	7.90	1	368

December 15th, Data		
	K	Last Price: Put
Dec 15	340.00	0.11
Dec 15	342.50	0.20
Dec 15	345.00	0.37
Dec 15	347.50	0.69
Dec 15	350.00	1.33
Dec 15	352.50	2.24
Dec 15	355.00	4.03
Dec 15	357.50	5.75
Dec 15	360.00	7.47

December 22nd, Data									
	Call: Bid	Ask	Vol	OpnInt	K	Put: Bid	Ask	Vol	OpnInt
Dec 22	13.75	14.10	15	158	340.00	0.54	0.62	70	188
Dec 22	11.50	11.85	3	26	342.50	0.80	0.90	23	45
Dec 22	9.45	9.75	9	159	345.00	1.16	1.33	39	73
Dec 22	7.50	7.80	1	33	347.50	1.75	1.91	15	20
Dec 22	5.85	6.05	35	347	350.00	2.55	2.70	50	142
Dec 22	4.40	4.55	15	81	352.50	3.60	3.75	62	26
Dec 22	3.20	3.35	84	246	355.00	4.85	5.10	9	5
Dec 22	2.27	2.39	34	50	357.50	6.45	6.65	30	--
Dec 22	1.57	1.66	283	191	360.00	8.25	8.60	3	5

December 29th, Data									
	Call: Bid	Ask	Vol	OpnInt	K	Put: Bid	Ask	Vol	OpnInt
Dec 29	14.65	15.00	11	148	340.00	0.98	1.13	30	198
Dec 29	8.10	12.90	--	--	342.50	1.30	1.48	10	5
Dec 29	10.25	10.85	22	156	345.00	1.85	2.00	191	45
Dec 29	8.65	9.00	2	--	347.50	2.47	2.63	8	--
Dec 29	7.05	7.25	9	197	350.00	3.25	3.50	7	39
Dec 29	5.50	5.75	27	8	352.50	4.30	4.50	23	1
Dec 29	4.25	4.50	7	194	355.00	5.55	5.80	16	3
Dec 29	3.20	3.45	3	4	357.50	7.05	7.35	4	--
Dec 29	2.39	2.58	116	193	360.00	8.65	9.10	--	--

January 5th, Data									
	Call: Bid	Ask	Vol	OpnInt	K	Put: Bid	Ask	Vol	OpnInt
Jan 5	15.35	16.05	3	106	340.00	1.58	1.75	11	30
Jan 5	11.40	12.00	6	88	345.00	2.57	2.77	2	35
Jan 5	8.15	8.50	2	76	350.00	4.15	4.35	2	20
Jan 5	5.45	5.70	62	170	355.00	6.35	6.65	4	26
Jan 5	3.35	3.60	19	69	360.00	9.40	10.15	--	--

January 12th, Data									
	Call: Bid	Ask	Vol	OpnInt	K	Put: Bid	Ask	Vol	OpnInt
Jan 12	16.60	17.10	14	13	340.00	2.27	2.48	1	15
Jan 12	12.90	13.25	5	18	345.00	3.35	3.60	2	26
Jan 12	9.50	9.85	12	26	350.00	5.05	5.35	1	5
Jan 12	6.70	7.00	44	262	355.00	7.25	7.55	--	--
Jan 12	4.50	4.80	73	118	360.00	10.10	10.60	--	--

July 19th, Data									
Call: Bid		Ask	Vol	OpnInt	K	Put: Bid	Ask	Vol	OpnInt
Jul 19	34.40	35.65	--	5	340.00	16.20	16.75	1	3
Jul 19	31.60	32.20	--	45	345.00	18.10	18.75	--	--
Jul 19	28.65	29.70	--	12	350.00	20.05	20.70	--	352
Jul 19	25.90	26.35	--	51	355.00	22.30	22.75	--	102
Jul 19	23.30	23.70	--	18	360.00	24.60	25.65	--	2

Implied Volatility for Option Chain, December 15th as of market close December 12th	
IV	K
0.22728	340
0.22334	342.5
0.23849	345
0.2228	347.5
0.21108	350
0.20906	352.5
0.20917	355
0.21339	357.5
0.22111	360

Implied Volatility for Option Chain, December 22nd as of market close December 12th	
IV	K
0.22796	340
0.21363	342.5
0.20452	345
0.19533	347.5
0.19003	350
0.18564	352.5
0.18334	355
0.18227	357.5
0.18213	360

Implied Volatility for Option Chain, July 19th, 2024 as of market close December 12th		
IV Call	K	IV Put
0.2413	340	0.2374
0.2398	345	0.2355
0.23746	350	0.2327
0.22954	355	0.2243
0.22517	360	0.2216