The Finite Difference Method

Author: Will Noone

Abstract

This paper describes the finite difference method and its context of valuing financial derivatives. Part I will outline the coverage of the derivation of the Black-Scholes PDE and the replacement of the partial derivatives with difference equations to approximate the derivatives. It will also discuss the three main approaches to this method: implicit, explicit, and the Crank-Nicholson. Part I will conclude with discussions on discretization error, stability, and convergence properties.

Part II will implement the explicit and implicit finite difference methods to value European Call and Put options that are out-of-the-money, at-the-money, or in-the-money. It will also demonstrate the convergence to Black-Scholes by changing the size of the grid based on American Call and Put options compared to European Call and Put options.

Part I
The Finite Difference Method

A. Derivation of the Black-Scholes PDE

The Black-Scholes partial differential equation doesn't encompass the real world growth rate, but rather the risk-free rate because of delta hedging. The equation below is based on geometric brownian motion:[1]

$$dS = \mu S dt + \sigma S dX$$

Ito's Lemma allows a relatively small change in a random variable which S can be similar to an incremental change in a function of the random variable, $V(S,t)$. Applying this to the previous equation for the GBM we derive:[2]

$$dV = \left(\mu S \frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + \frac{\partial V}{\partial t}\right) dt + \sigma S \frac{\partial V}{\partial S} dX$$

The equations above can replaced for construction of portfolio $\prod$, which is composed of a long option $V(S,t)$ and a short position in underlying asset ΔS, the following portfolio equation becomes:[3]

$$d\prod = dV - \Delta dS$$

$$d\prod = \sigma S \left(\frac{\partial V}{\partial S} - \Delta\right) dX + \left(\mu S \frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + \frac{\partial V}{\partial t} - \mu \Delta S\right) dt$$

To eliminate the *d*X term, the choice $\Delta = \frac{\partial V}{\partial S}$ converts to:[4]

$$d\prod = \left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}\right) dt$$

Based on the Black-Scholes next step of no-arbitrage principle, the value of the portfolio must be equal on average to the value the portfolio invested at the risk-free interest rate which is:[5]

$$r\prod dt = \left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}\right) dt$$

---

[1] Young, Stephen D. *Financial Engineering Notes*. FINA 6281. The George Washington University, p 162.
[2] Richardson, Mark. "Numerical Methods for Option Pricing." Mar 2009, p 2.
[3] Ibid p.3
[4] Ibid p.3
[5] Ibid p.3

Lastly, the Black-Scholes PDE is derived by combining the previous equations and canceling the term dt, becoming: [6]

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

B. Replacement of Partial Derivatives with Difference Equations to Approximate the Derivatives

A finite difference is a mathematical expression of the form:

$$f(x + b) - f(x + a).$$

If a finite difference is divided by $b - a$, one gets a difference quotient. The approximation of derivatives by finite differences plays a central role in finite difference methods for the numerical solution of differential equations, especially boundary value problems. [7] When using the finite difference methods with a single underlying asset one would use a specific pricing structure called a "grid". [8] There are three types of finite difference approximations:[9]

Forward Difference: $\frac{\partial f}{\partial t} \approx \frac{f_{i+1,j} - f_{i,j}}{\Delta t}$, $\frac{\partial f}{\partial S} \approx \frac{f_{i,j+1} - f_{i,j}}{\Delta S}$

Backward Difference: $\frac{\partial f}{\partial t} \approx \frac{f_{i,j} - f_{i-1,j}}{\Delta t}$, $\frac{\partial f}{\partial S} \approx \frac{f_{i,j} - f_{i,j-1}}{\Delta S}$

Central Difference: $\frac{\partial f}{\partial t} \approx \frac{f_{i+1,j} - f_{i-1,j}}{2\Delta t}$, $\frac{\partial f}{\partial S} \approx \frac{f_{i,j+1} - f_{i,j-1}}{2\Delta S}$

With the approximations above, it can be used for the various three finite difference methods: explicit (forward difference), implicit (backward difference), and Crank-Nicholson (central difference).[10]

[6] Ibid p.3
[7] http://en.wikipedia.org/wiki/Finite_difference
[8] Young, Stephen D. *Financial Engineering Notes*. FINA 6281. The George Washington University, p 166
[9] http://en.wikipedia.org/wiki/Finite_difference
[10] Young, Stephen D. *Financial Engineering Notes*. FINA 6281. The George Washington University, p 168

C. Explicit Method

The explicit method calculates the state of a system at a later time from the state of the system at the current time and is derived as follows:

$$\text{In } \frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf,$$

at point $(i\Delta t, j\Delta S)$, set back difference

$$\frac{\partial f}{\partial t} \approx \frac{f_{i,j} - f_{i-1,j}}{\Delta t},$$

central difference

$$\frac{\partial f}{\partial S} \approx \frac{f_{i,j+1} - f_{i,j-1}}{\Delta 2S} \quad \text{and} \quad \frac{\partial^2 f}{\partial S^2} \approx \frac{f_{i,j+1} - f_{i,j-1} - 2f_{i,j}}{\Delta S^2},$$

$r f = r f_{i,j}$ and $S = j\Delta S$.

The following is an Explicit scheme after rewriting the equation:

$$f_{i-1,j} = a_j f_{i,j-1} + b_j f_{i,j} + c_j f_{i,j+1}$$

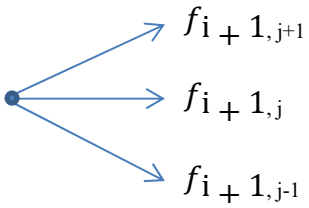
Where:

$$a_j = \frac{1}{2}\Delta t(\sigma^2 j^2 - rj)$$

$$b = 1 - \Delta t(\sigma^2 j^2 + r)$$

$$c_j = \frac{1}{2}\Delta t(\sigma^2 j^2 + rj)$$

The Explicit method can be depicted by the trinomial tree below which mentioned earlier is a forward difference.[11]



[11] Ibid p.168

D. Implicit Method

Implicit methods find a solution by solving an equation involving both the current state of the system and the later one. This method needs a system of linear equations to be solved at each time step, which is derived as follows:[12]

$$\text{In } \frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf,$$

we set forward difference

$$\frac{\partial f}{\partial t} \approx \frac{f_{i+1,j} - f_{i,j}}{\Delta t},$$

central difference

$$\frac{\partial f}{\partial S} \approx \frac{f_{i,j+1} - f_{i,j-1}}{\Delta 2S} \quad \text{and} \quad \frac{\partial^2 f}{\partial S^2} \approx \frac{f_{i,j+1} + f_{i,j-1} - 2f_{i,j}}{\Delta S^2}, \; rf = rf_{i,j}.$$

The following is an Implicit scheme after rewriting the equation:

$$a_j f_{i,j-1} + b_j f_{i,j} + c_j f_{i,j+1} = f_{i+1,j}$$

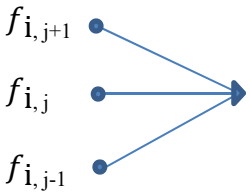
Where:

$$a_j = \frac{1}{2}\Delta t(-\sigma^2 j^2 + rj)$$

$$b = 1 + \Delta t(\sigma^2 j^2 + r)$$

$$c_j = -\frac{1}{2}\Delta t(\sigma^2 j^2 + rj)$$

The Implicit method, the three previous nodes provide information to calculate the next option value, shown with the diagram below.[13]



[12] Richardson, Mark. "Numerical Methods for Option Pricing." Mar 2009, p 5

[13] Young, Stephen D. *Financial Engineering Notes*. FINA 6281. The George Washington University, p 168

E. Crank-Nicholson Method

The Crank-Nicolson method combines the stability of an implicit method with an improved accuracy that is second-order in both space and time. Simply by the average of the explicit and implicit schemes, we can gather the following: 14

$$g_{i,j} = a_j f_{i-1,j-1} + b_j f_{i-1,j} + c_j f_{i-1,j+1} - f_{i-1,j}$$

The Crank-Nicholson is superior method considering it applies both aspects of the explicit and implicit methods.

F. Discretization Error

Discretization error is the error inherent in discretization. It results from the fact that a function of a continuous variable is represented in the computer by a finite number of evaluations, for example, on a lattice. "Discretization error can usually be reduced by using a more finely spaced lattice, with an increased computational cost. Whenever continuous data is discretized, there is always some amount of discretization error". 15 The effectiveness of the finite difference scheme is determined by a mixture of discretization error, stability, and convergence.16

G. Stability

Stability refers to whether or not the method is sensitive to small distresses in specific inputs.17 Stability issues are related to a numerical algorithm which can produce poor solutions to well-conditioned problems.

H. Convergence Properties

Convergence is a goal to come to the correct solution based on approximations as soon as possible.18 Given an approximate solution x^k, usually one or a combination of the four common tests can be used to check convergence:

Absolute difference:

$$\left|\left|x^{(k)} - x^{(k-1)}\right|\right| < \varepsilon_1$$

Relative difference:

$$\frac{\left|\left|x^{(k)} - x^{(k-1)}\right|\right|}{\left|\left|x^{(k-1)}\right|\right|} < \varepsilon_2$$

---

[14] Bergara, Aitor. *Finite-difference Numerical Methods of Partial Differential Equations in Finance with Matlab, p.26*
[15] http://en.wikipedia.org/wiki/Discretization_error
[16] Young, Stephen D. *Financial Engineering Notes*. FINA 6281. The George Washington University, p 168
[17] Ibid p. 169.
[18] http://www.math.yorku.ca/~hmzhu/Math-6911/lectures/Lecture6/6_BlkSch_FDM_Amer.pdf.

Absolute residual:

$$\left|\left|r(x^{(k)})\right|\right| < \varepsilon_3$$

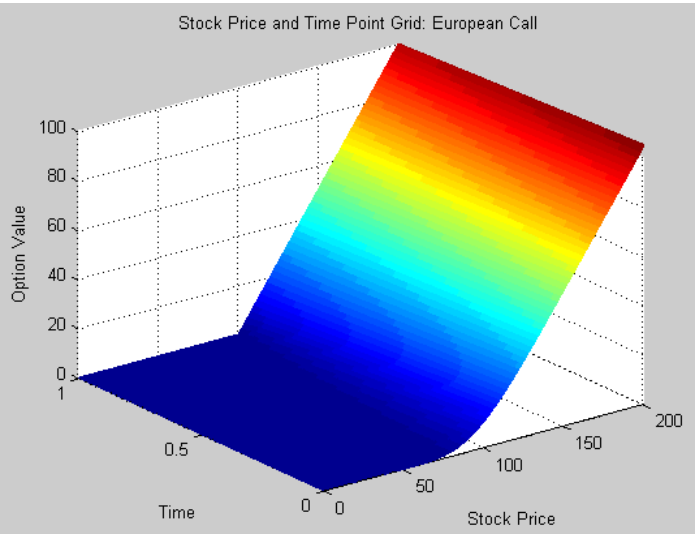
Relative residual:

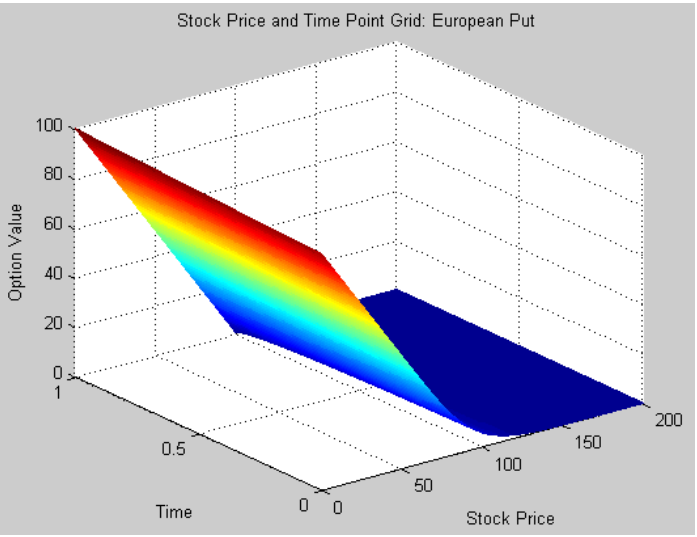
$$\frac{\left|\left|r(x^{(k)})\right|\right|}{||b||} < \varepsilon_4$$

Part II
Implementing Finite Difference Methods

A. Finite Difference Method Visualization

The finite difference method can be visualized using MATLAB for European Calls and European Puts across a range of underlying stock prices and time increments. With respect to the underlying grid, Stock Price Points (M) has been set at 500 and Time Points (N) has been set at 100. We expect that as the grid in these given dimensions is discretized further, the curves will approach those of the Black Scholes PDE. Said another way, the function approaches the smooth curve of the continuous Black Scholes PDE function and Stock Price and Time increments approach lim → 0.





B. European Option Valuation: Explicit vs Finite Difference Method

To present a numerical example, we next utilize both the explicit and implicit finite difference methods to value In-the-Money, At-the-Money, and Out-the-Money European options. The following inputs have been selected:

Inputs	Range		
Stock Price	80	100	120
Strike Price	100		
Risk Free rate	0.05		
Annual Volatility	0.20		
Time (yrs)	1		

To implement both finite difference methods, we first select 3 scenarios representing increasingly "small" discretization of the Stock Price and Time Points: Base, Adjustment 1, and Adjustment 2. Note that for this example we assume a Stock Price range of $0 to $150 and a Time to Maturity of 1 year. Using MATLAB, we first create vectors of Stock Prices and Time increments to facilitate a changing grid size among scenarios. Next, we can estimate the coefficients of the Explicit and Implicit schemes across the vectors. Finally, we are able to set boundary conditions for European Calls and Puts and find the average discounted option value via the system of nodes.

European Options		
Grid Iteration	Stock Price Points	Time Points
Base	0:5:150	0:0.001:1
Adjustment 1	0:2.5:150	0:0.0001:1
Adjustment 2	0:0.5:150	0:0.00001:1

As expected, as the discretization of both Stock Price and Time approaches lim → 0, both the Explicit and Implicit Difference Method value approximations for European options converge upon the estimates yielded by the Black Scholes model. Note that the Implicit Finite Difference Method, does not closely approximate the Black Scholes estimates for European Calls as the option increases in "money-ness".

	Stock Price		
Base	80	100	120
Explicit Finite			
European Call	1.8343	10.2811	25.1444
European Put	16.9605	5.5119	1.2623
Implicit Finite			
European Call	1.6863	7.5071	10.3841
European Put	16.9617	5.5099	1.2616
Black Scholes			
European Call	1.8594	10.4506	26.1690
European Put	16.9824	5.5735	1.2920

	Stock Price		
Adjustment 1	80	100	120
Explicit Finite			
European Call	1.8510	10.3307	25.1688
European Put	16.9769	5.5580	1.2820
Implicit Finite			
European Call	1.7194	7.5942	10.3928
European Put	16.9770	5.5578	1.2819
Black Scholes			
European Call	1.8594	10.4506	26.1690
European Put	16.9824	5.5735	1.2920

	Stock Price		
Adjustment 2	80	100	120
Explicit Finite			
European Call	1.8565	10.3467	25.1770
European Put	16.9821	5.5728	1.2884
Implicit Finite			
European Call	1.7301	7.6212	10.3965
European Put	16.9821	5.5728	1.2884
Black Scholes			
European Call	1.8594	10.4506	26.1690
European Put	16.9824	5.5735	1.2920

C. American Option Valuation: Finite Difference Methods

Finally, we consider Finite Difference Method valuation of American options. We again used MATLAB to implement three scenarios of an increasingly discretized grid: Base, Adjustment 1, and Adjustment 2. Given the exercise rights at any given time increment, we match Stock Price and Time Points in increasing and equal iterations to facilitate this modification.

American Options		
Grid Iteration	Stock Price Points	Time Points
Base	50	50
Adjustment 1	100	100
Adjustment 2	150	150

	Stock Price		
Base	80	100	120
Finite Difference			
American Call	2.9051	8.9925	23.6542
American Put	16.0942	7.0858	1.7049

	Stock Price		
Adjustment 1	80	100	120
Finite Difference			
American Call	2.3211	10.2841	25.8919
American Put	18.1043	6.1814	1.4007

	Stock Price		
Adjustment 2	80	100	120
Finite Difference			
American Call	2.1423	10.7292	26.6448
American Put	18.8051	5.8976	1.3073

The American option has the flexibility to exercise prior to maturity. Because, for example, an American put holder may exercise a deep In-the-Money put and reinvest the earnings prior to the option maturity, a premium is incorporated into the price over the equivalent European counterpart. For this reason, there is a premium embedded within the American option price when compared to a European option.

References

http://www.math.yorku.ca/~hmzhu/Math-6911/lectures/Lecture6/6_BlkSch_FDM_Amer.pdf

Bergara, Aitor. *Finite-difference Numerical Methods of Partial Differential Equations in Finance with Matlab*

Richardson, Mark. "Numerical Methods for Option Pricing." Mar 2009

Wikipedia: http://en.wikipedia.org/wiki/Discretization_error

Wikipedia: http://en.wikipedia.org/wiki/Finite_difference

Young, D. Stephen, *Financial Engineering Notes*

APPENDIX
MATLAB SYNTAX

FUNCTIONS

Note: SYNTAX for this section was created with reference to the following:

Goddard, Phil. "Explicit Finite Difference Method - A MATLAB Implementation." Explicit Finite Difference Method - A MATLAB Implementation. 2007.

Goddard, Phil. "Implicit Finite Difference Method - A MATLAB Implementation." Implicit Finite Difference Method - A MATLAB Implementation. 2007.

Hoyle, Mark. "Pricing American Options" File Exchange. 20 Sept. 2007.

1.a FUNCTION FOR EXPLICIT FINITE DIFFERENCE METHOD

```
function oPrice = finDiffExplicit(X,S0,r,sig,Svec,tvec,oType)

% FUNCTION TO CALCULATE PRICE OF A EUROPEAN OPTION
% USING EXPLICIT FINITE DIFERENCE METHOD

% INPUTS
% X=strike price
% S0=stock price
% r=risk free rate
% sig=annual volatility
% Svec=Vector of stock prices (i.e. grid points)
% tvec=Vector of times (i.e. grid points)
% oType='PUT' or 'CALL'.

% Output: oPrice-option price

%GET # OF GRID POINTS
M = length(Svec)-1;
N = length(tvec)-1;

%GET GRID SIZES(assuming equi-spaced points)
dt = tvec(2)-tvec(1);

%CALCULATE THE COEFFICIENTS
j = 1:M-1;
sig2 = sig*sig;
j2 = j.*j;
aj = 0.5*dt*(sig2*j2-r*j);
bj = 1-dt*(sig2*j2+r);
cj = 0.5*dt*(sig2*j2+r*j);

% PRE-ALLOCATE THE OUTPUT
price(1:M+1,1:N+1) = nan;

% SPECIFY BOUNDARY CONDITIONS
switch oType
    case 'CALL'
        price(:,end) = max(Svec-X,0);
```

```
        price(1,:) = 0;
        price(end,:) = (Svec(end)-X)*exp(-r*tvec(end:-1:1));
    case 'PUT'
        price(:,end) = max(X-Svec,0);
        price(1,:) = (X-Svec(end))*exp(-r*tvec(end:-1:1));
        price(end,:) = 0;
end

% FORM THE TRIDIAGONAL MATRIX
A = diag(bj);  % DIAGONAL TERMS
A(2:M:end) = aj(2:end); % BELOW DIAGONAL
A(M:M:end) = cj(1:end-1); % ABOVE DIAGONAL

% CALCULATE PRICE AT ALL INTERIOR NODES
offsetConstants = [aj(1); cj(end)];
for i = N:-1:1
    price(2:end-1,i) = A*price(2:end-1,i+1);
    price([2 end-1],i) = price([2 end-1],i) + ...
        offsetConstants.*price([1 end],i+1);
end

% CALCULATE OPTION PRICE
oPrice = interp1(Svec,price(:,1),S0);
```

1.b FUNCTION FOR IMPLICIT FINITE DIFFERENCE METHOD

```
function oPrice = finDiffImplicit(X,S0,r,sig,Svec,tvec,oType)

% FUNCTION TO CALCULATE PRICE OF A EUROPEAN OPTION
% USING IMPLICIT FINITE DIFERENCE METHOD

% INPUTS
% X=strike
% S0=stock price
% r=risk free rate
% sig=annual volatility
% Svec=Vector of stock prices (i.e. grid points)
% tvec=Vector of times (i.e. grid points)
% oType='PUT' or 'CALL'.

% Output: oPrice-option price

% GET # GRID POINTS
M = length(Svec)-1;
N = length(tvec)-1;
% Get the grid sizes (assuming equi-spaced points)
dt = tvec(2)-tvec(1);

% CALCULATE COEFFICIENTS
j = 0:M;
sig2 = sig*sig;
aj = (dt*j/2).*(r - sig2*j);
bj = 1 + dt*(sig2*(j.^2) + r);
cj = -(dt*j/2).*(r + sig2*j);
```

```
% PRE-ALLOCATE THE OUTPUT
price(1:M+1,1:N+1) = nan;

% SPECIFY BOUNDARY CONDITIONS
switch oType
    case 'CALL'
        price(:,end) = max(Svec-X,0);
        price(1,:) = 0;
        price(end,:) = (Svec(end)-X)*exp(-r*tvec(end:-1:1));
    case 'PUT'
        price(:,end) = max(X-Svec,0);
        price(1,:) = (X-Svec(end))*exp(-r*tvec(end:-1:1));
        price(end,:) = 0;
end

% FORM TRIDIAGONAL MATRIX
B = diag(aj(3:M),-1) + diag(bj(2:M)) + diag(cj(2:M-1),1);
[L,U] = lu(B);

% SOLVE AT EACH NODE
offset = zeros(size(B,2),1);
for idx = N:-1:1
    offset(1) = aj(2)*price(1,idx);
    % NOTE offset(end) = c(end)*price(end,idx);
    % This will always be zero
    price(2:M,idx) = U\(L\(price(2:M,idx+1) - offset));
end

% CALCULATE OPTION PRICE
oPrice = interp1(Svec,price(:,1),S0);
```

1.c FUNCTION FOR FINITE DIFFERENCE METHOD-AMERICAN OPTION VALUATION

```
function [P_FD,P,s,t] = AmericanOptFD(S0,X,r,T,sigma,N,M,type)

% FUNCTION TO CALCULATE PRICE OF AN AMERICAN OPTION
% USING FINITE DIFERENCE METHOD

% INPUTS
% S0=stock price
% X=strike price
% r=risk free rate
% t=Time to maturity of option
% sigma=annual volatility
% N=Number of pts in time grid (min=3,default=50)
% M=Number of points in asset price grid to use (minimum is 3, default is 50)
% type= TRUE(default) for a put, FALSE for a call

if nargin < 6 || isempty(N), N = 50;
elseif N < 3, error('N has to be at least 3');
end
if nargin < 7 || isempty(M), M = 50;
```

```
elseif M < 3, error('M has to be at least 3');
end
if nargin < 8, type = true;
end

% CREATE TIME GRID
t = linspace(0,T,N+1);
dt = T/N; % Time step

% CREATE STOCK PRICE GRID
Smax = 2*max(S0,X)*exp(r*T); % Maximum price considered
dS = Smax/(M);
s = 0:dS:Smax;

% INTERPOLATE OPTION PRICE AROUND INITIAL STOCK PRICE
idx = find(s < S0); idx = idx(end); a = S0-s(idx); b = s(idx+1)-S0;
Z = 1/(a+b)*[a b]; % Interpolation vector

% CREATE OPTION PRICING MATRIX-Pricing Matrix (t,S)
P = NaN*ones(N+1,M+1);

% CREATE BOUNDARY CONDITIONS
if type
    P(end,:) = max(X-(0:M)*dS,0); %PUT VALUE AT MATURITY
else
    P(end,:) = max((0:M)*dS-X,0); %CALL VALUE AT MATURITY
end
P(:,1) = X; %VALUE OF OPTION WHEN STOCK IS AT MIN PRICE
P(:,end) = 0; %VALUE OF OPTION WHEN STOCK IS AT MAX PRICE

% CREATE MATRIX FOR FINITE DIFFERENCE CALCULATIONS
J = (1:M-1)';
a = r/2*J*dt-1/2*sigma^2*J.^2*dt;
b = 1+sigma^2*J.^2*dt+r*dt;
c = -r/2*J*dt-1/2*sigma^2*J.^2*dt;
D = spdiags([[a(2:end);0] b [0;c(1:end-1)]],[-1 0 1],M-1,M-1);

% FINITE DIFFERENCE CALCULATION
for ii = N:-1:1
    y = P(ii+1,2:end-1)'+[-a(1)*X; zeros(M-3,1); -c(end)*0];
    x = D\y;
    if type
        P(ii,2:end-1) = max(x,X-s(2:end-1)'); % Put
    else
        P(ii,2:end-1) = max(x,s(2:end-1)'-X); % Call
    end
end

% CALCULATION OPTION PRICE
P_FD = Z*P(1,idx:idx+1)';

end
```

SCRIPTS

2.a EXPLICIT FINITE DIFFERENCE METHOD AND BSM VALUATION FOR EUROPEAN OPTIONS

```
%INPUTS
X=100;
r=0.05;
sigma=0.20;
t=1;
spot1=80;
spot2=100;
spot3=120;

%GRID PARAMETERS
svec1=0:5:150;
svec2=0:2.5:150;
svec3=0:0.5:150;
tvec1=0:0.001:1;
tvec2=0:0.0001:1;
tvec3=0:0.00001:1;

%BLACK SCHOLES VALUATION
[BSC1,BSP1]=blsprice(spot1,X,r,t,sigma) %S0=80
[BSC2,BSP2]=blsprice(spot2,X,r,t,sigma) %S0=100
[BSC3,BSP3]=blsprice(spot3,X,r,t,sigma) %S0=120

%EXPLICIT FINITE DIFFERENCE METHOD

%BASE CASE GRID-EXPLICIT METHOD FOR EUROPEAN OPTIONS
%S0=80
EC1spot1=finDiffExplicit(X,spot1,r,sigma,svec1,tvec1,'CALL')
EP1spot1=finDiffExplicit(X,spot1,r,sigma,svec1,tvec1,'PUT')

%S0=100
EC1spot2=finDiffExplicit(X,spot2,r,sigma,svec1,tvec1,'CALL')
EP1spot2=finDiffExplicit(X,spot2,r,sigma,svec1,tvec1,'PUT')

%S0=120
EC1spot3=finDiffExplicit(X,spot3,r,sigma,svec1,tvec1,'CALL')
EP1spot3=finDiffExplicit(X,spot3,r,sigma,svec1,tvec1,'PUT')


%ADJUSTMENT 1 GRID-EXPLICIT METHOD FOR EUROPEAN OPTIONS
%S0=80
EC2spot1=finDiffExplicit(X,spot1,r,sigma,svec2,tvec2,'CALL')
EP2spot1=finDiffExplicit(X,spot1,r,sigma,svec2,tvec2,'PUT')

%S0=100
EC2spot2=finDiffExplicit(X,spot2,r,sigma,svec2,tvec2,'CALL')
EP2spot2=finDiffExplicit(X,spot2,r,sigma,svec2,tvec2,'PUT')

%S0=120
EC2spot3=finDiffExplicit(X,spot3,r,sigma,svec2,tvec2,'CALL')
EP2spot3=finDiffExplicit(X,spot3,r,sigma,svec2,tvec2,'PUT')
```

```
%ADJUSTMENT 2 GRID-EXPLICIT METHOD FOR EUROPEAN OPTIONS
%S0=80
EC3spot1=finDiffExplicit(X,spot1,r,sigma,svec3,tvec3,'CALL')
EP3spot1=finDiffExplicit(X,spot1,r,sigma,svec3,tvec3,'PUT')

%S0=100
EC3spot2=finDiffExplicit(X,spot2,r,sigma,svec3,tvec3,'CALL')
EP3spot2=finDiffExplicit(X,spot2,r,sigma,svec3,tvec3,'PUT')

%S0=120
EC3spot3=finDiffExplicit(X,spot3,r,sigma,svec3,tvec3,'CALL')
EP3spot3=finDiffExplicit(X,spot3,r,sigma,svec3,tvec3,'PUT')
```

2.b IMPLICIT FINITE DIFFERENCE METHOD AND BSM VALUATION FOR EUROPEAN OPTIONS

```
%INPUTS
X=100;
r=0.05;
sigma=0.20;
t=1;
spot1=80;
spot2=100;
spot3=120;

%GRID PARAMETERS
svec1=0:5:150;
svec2=0:2.5:150;
svec3=0:0.5:150;
tvec1=0:0.001:1;
tvec2=0:0.0001:1;
tvec3=0:0.00001:1;

%BLACK SCHOLES VALUATION
[BSC1,BSP1]=blsprice(spot1,X,r,t,sigma) %S0=80
[BSC2,BSP2]=blsprice(spot2,X,r,t,sigma) %S0=100
[BSC3,BSP3]=blsprice(spot3,X,r,t,sigma) %S0=120

%IMPLICIT FINITE DIFFERENCE METHOD

%BASE CASE GRID-IMPLICIT METHOD FOR EUROPEAN OPTIONS
%S0=80
IC1spot1=finDiffImplicit(X,spot1,r,sigma,svec1,tvec1,'CALL')
IP1spot1=finDiffImplicit(X,spot1,r,sigma,svec1,tvec1,'PUT')

%S0=100
IC1spot2=finDiffImplicit(X,spot2,r,sigma,svec1,tvec1,'CALL')
IP1spot2=finDiffImplicit(X,spot2,r,sigma,svec1,tvec1,'PUT')

%S0=120
IC1spot3=finDiffImplicit(X,spot3,r,sigma,svec1,tvec1,'CALL')
IP1spot3=finDiffImplicit(X,spot3,r,sigma,svec1,tvec1,'PUT')


%ADJUSTMENT 1 GRID-IMPLICIT METHOD FOR EUROPEAN OPTIONS
```

```
%S0=80
IC2spot1=finDiffImplicit(X,spot1,r,sigma,svec2,tvec2,'CALL')
IP2spot1=finDiffImplicit(X,spot1,r,sigma,svec2,tvec2,'PUT')

%S0=100
IC2spot2=finDiffImplicit(X,spot2,r,sigma,svec2,tvec2,'CALL')
IP2spot2=finDiffImplicit(X,spot2,r,sigma,svec2,tvec2,'PUT')

%S0=120
IC2spot3=finDiffImplicit(X,spot3,r,sigma,svec2,tvec2,'CALL')
IP2spot3=finDiffImplicit(X,spot3,r,sigma,svec2,tvec2,'PUT')


%ADJUSTMENT 2 GRID-IMPLICIT METHOD FOR EUROPEAN OPTIONS
%S0=80
IC3spot1=finDiffImplicit(X,spot1,r,sigma,svec3,tvec3,'CALL')
IP3spot1=finDiffImplicit(X,spot1,r,sigma,svec3,tvec3,'PUT')

%S0=100
IC3spot2=finDiffImplicit(X,spot2,r,sigma,svec3,tvec3,'CALL')
IP3spot2=finDiffImplicit(X,spot2,r,sigma,svec3,tvec3,'PUT')

%S0=120
IC3spot3=finDiffImplicit(X,spot3,r,sigma,svec3,tvec3,'CALL')
IP3spot3=finDiffImplicit(X,spot3,r,sigma,svec3,tvec3,'PUT')
```

2.c FINITE DIFFERENCE METHOD-AMERICAN OPTION VALUATION

```
%INPUTS
X=100;
r=0.05;
sigma=0.20;
t=1;
spot1=80;
spot2=100;
spot3=120;

%FINITE DIFFERENCE METHOD-AMERICAN OPTIONS

%BASE CASE GRID
%S0=80
AC1spot1=AmericanOptFD(spot1,X,r,t,sigma,50,50,false)
AP1spot1=AmericanOptFD(spot1,X,r,t,sigma,50,50,true)

%S0=100
AC1spot2=AmericanOptFD(spot2,X,r,t,sigma,50,50,false)
AP1spot2=AmericanOptFD(spot2,X,r,t,sigma,50,50,true)

%S0=120
AC1spot3=AmericanOptFD(spot3,X,r,t,sigma,50,50,false)
AP1spot3=AmericanOptFD(spot3,X,r,t,sigma,50,50,true)

%ADJUSTMENT 1 GRID
%S0=80
```

```
AC2spot1=AmericanOptFD(spot1,X,r,t,sigma,100,100,false)
AP2spot1=AmericanOptFD(spot1,X,r,t,sigma,100,100,true)

%S0=100
AC2spot2=AmericanOptFD(spot2,X,r,t,sigma,100,100,false)
AP2spot2=AmericanOptFD(spot2,X,r,t,sigma,100,100,true)

%S0=120
AC2spot3=AmericanOptFD(spot3,X,r,t,sigma,100,100,false)
AP2spot3=AmericanOptFD(spot3,X,r,t,sigma,100,100,true)

%ADJUSTMENT 2 GRID
%S0=80
AC3spot1=AmericanOptFD(spot1,X,r,t,sigma,150,150,false)
AP3spot1=AmericanOptFD(spot1,X,r,t,sigma,150,150,true)

%S0=100
AC3spot2=AmericanOptFD(spot2,X,r,t,sigma,150,150,false)
AP3spot2=AmericanOptFD(spot2,X,r,t,sigma,150,150,true)

%S0=120
AC3spot3=AmericanOptFD(spot3,X,r,t,sigma,150,150,false)
AP3spot3=AmericanOptFD(spot3,X,r,t,sigma,150,150,true)
```

VISUALIZATIONS

Note: SYNTAX for this section was created with reference to the following:

Richardson, Mark. "Numerical Methods for Option Pricing.": Mar 2007.

3.a VISUALIZATION GRID FOR EUROPEAN CALL

```
%CREATES GRID FOR EUROPEAN CALL-IMPLICIT METHOD

%INPUTS
S = 100; % stock price
K = 100; % Strike price
r = 0.05; % Risk free rate
sigma = 0.20; % annual volatility
T = 1; % Time to maturity

%METHOD PARAMETERS
M = 100; % Number of asset mesh points
N = 500; % Number of time mesh points
Szero = 0; % Specify extremes
Smax= 200;

% SETUP MESH & BCs
solution_mesh=zeros(N+1,M+1); % Create Mesh
Smesh=0:(Smax/M):Smax; % Mesh of equally spaced S values
Tmesh=T:-(T/N):0; % Mesh of equally spaced T values
dt=T/N; % Specify timestep, dt

% Option values for terminal payoffs, lowermost nodes, uppermost nodes
% Need to change these for calls versus puts
solution_mesh(1,:)= max(Smesh-K,0);
```

```
solution_mesh(:,1)= 0;
solution_mesh(:,M+1)= Smax - K*exp(-r*(T-Tmesh));

% Values for A, B, C which are terms derived from discretized PDE
A = @(i) 0.5*dt*(r*i-sigma^2*i^2);
B = @(i) 1 + (sigma^2*i^2 + r)*dt; % Define the functions A, B & C
C = @(i) -0.5*dt*(sigma^2*i^2+r*i);

% Construct Tridiagonal Matrix, Tri
Acoeffs = zeros(M+1,1);Bcoeffs = zeros(M+1,1);Ccoeffs = zeros(M+1,1);
for i=1:M+1
Acoeffs(i) = A(i-1); Bcoeffs(i) = B(i-1); Ccoeffs(i) = C(i-1);
end
Tri=diag(Acoeffs(2:end),-1)+diag(Bcoeffs)+diag(Ccoeffs(1:end-1),+1);
Tri_Inv=inv(Tri); % Compute inverse

% Implicit Euler Iteration
for j=1:N
temp=zeros(M+1,1);
temp(1)=A(0)*solution_mesh(j+1,1);
temp(end)=C(M)*solution_mesh(j+1,M+1); % Boundary terms
RHS=solution_mesh(j,:)'-temp;
temp=Tri_Inv*RHS;
solution_mesh(j+1,(2:end-1))=temp(2:end-1);
end
mesh(Smesh,Tmesh,solution_mesh)
xlabel('Stock Price');ylabel('Time');zlabel('Option Value')
title('Stock Price and Time Point Grid: European Call')
axis([0 200 0 1 0 100]);

% Extract Desired Values Using Interpolation
interp1(Smesh,solution_mesh(N+1,:),S
```

3.b VISUALIZATION GRID FOR EUROPEAN PUT

```
%CREATES GRID FOR EUROPEAN PUT-IMPLICIT METHOD

%INPUTS
S = 100; % stock price
K = 100; % Strike price
r = 0.05; % Risk free rate
sigma = 0.20; % annual volatility
T = 1; % Time to maturity

%METHOD PARAMETERS
M = 100; % Number of asset mesh points
N = 500; % Number of time mesh points
Szero = 0; % Specify extremes
Smax= 200;

% SETUP MESH BCs
solution_mesh=zeros(N+1,M+1); % Create Mesh
Smesh=0:(Smax/M):Smax; % Mesh of equally spaced S values
Tmesh=T:-(T/N):0; % Mesh of equally spaced T values
dt=T/N; % Specify timestep, dt
```

```
solution_mesh(1,:)= max(K-Smesh,0); % Payoff BC (Put)
solution_mesh(:,1)= K*exp(-r*(T-Tmesh)); % BC at S=0
solution_mesh(:,M+1)= 0; % BC at S=M
A = @(i) 0.5*dt*(r*i-sigma^2*i^2);
B = @(i) 1 + (sigma^2*i^2 + r)*dt; % Define the functions A, B & C
C = @(i) -0.5*dt*(sigma^2*i^2+r*i);

% Construct Tridiagonal Matrix, Tri
Acoeffs = zeros(M+1,1);Bcoeffs = zeros(M+1,1);Ccoeffs = zeros(M+1,1);
for i=1:M+1
Acoeffs(i) = A(i-1); Bcoeffs(i) = B(i-1); Ccoeffs(i) = C(i-1);
end
Tri=diag(Acoeffs(2:end),-1)+diag(Bcoeffs)+diag(Ccoeffs(1:end-1),+1);
Tri_Inv=inv(Tri); % Compute inverse

% Implicit Euler Iteration
for j=1:N
temp=zeros(M+1,1);
temp(1)=A(0)*solution_mesh(j+1,1);
temp(end)=C(M)*solution_mesh(j+1,M+1); % Boundary terms
RHS=solution_mesh(j,:)'-temp;
temp=Tri_Inv*RHS;
solution_mesh(j+1,(2:end-1))=temp(2:end-1);
end
mesh(Smesh,Tmesh,solution_mesh)
xlabel('Stock Price');ylabel('Time');zlabel('Option Value')
title('Stock Price and Time Point Grid: European Put')

% Extract Desired Values Using Interpolation
interp1(Smesh,solution_mesh(N+1,:),S)
```