Goddard Consulting

Modeling. Simulation. Data Analysis. Visualization.

Home

Financial Engineering

    Option Pricing

    Econometrics

    Portfolio Optimization

    Technical Analysis

Control System Design

    PID Controllers

    Kalman Filters

Software Tutorials

    MATLAB

    Simulink

    VBA

    C++

Services

    Financial Modeling

    Control System Design

    Generic Modeling

    Auditing Code

    Software Training

About Us

Contact Us

Cox Ross Rubinstein in MATLAB

This tutorial presents MATLAB code that implements the Cox Ross Rubinstein (CRR) version of the binomial model as discussed in the Cox Ross Rubinstein section of the Binomial model tutorial.

The code may be used to price vanilla European or American, Put or Call, options. Given appropriate input parameters a full lattice of prices for the underlying asset is calculated, and backwards induction is used to calculate an option price at each node in the lattice. Creating a full lattice is wasteful (of memory and computation time) when only the option price is required. However the code could easily be modified to show how the price evolves over time in which case the full lattices would be required.

Note that the primary purpose of the code is to show how to implement the Cox Ross Rubinstein binomial model. The code contains no error checking and is not optimized for speed or memory use. As such it is not suitable for inclusion into a larger application without modifications.

A Pricing Example

Consider pricing a European Call option with the following parameters, $X = \$60$, $S_0 = \$50$, $r = 5\%$, $\sigma = 0.2$, $\Delta t = 0.01$, $N = 100$.

The Black-Scholes price for this option is *$1.624*.

A MATLAB function called **binPriceCRR** is given below. The following shows an example of executing **binPriceCRR** (and pricing the above option) in MATLAB,

```
>> oPrice = binPriceCRR(60,50,0.05,0.2,0.01,100,'CALL',false)
oPrice =
      1.628
```

If the number of time steps is doubled then

```
>> oPrice = binPriceCRR(60,50,0.05,0.2,0.005,200,'CALL',false)
oPrice =
      1.626
```

MATLAB Function: binPriceCRR

```
function oPrice = binPriceCRR(X,S0,r,sig,dt,steps,oType,earlyExercise)
```

```matlab
% Function to calculate the price of a vanilla European or American
% Put or Call option using a Cox Ross Rubinstein binomial tree.
%
% Inputs: X - strike
%       : S0 - stock price
%       : r - risk free interest rate
%       : sig - volatility
%       : dt - size of time steps
%       : steps - number of time steps to calculate
%       : oType - must be 'PUT' or 'CALL'.
%       : earlyExercise - true for American, false for European.
%
% Output: oPrice - the option price
%
% Notes: This code focuses on details of the implementation of the
%                              Cox Ross Rubinstein (CRR)
%        algorithm.
%        It does not contain any programatic essentials such as error
%        checking.
%        It does not allow for optional/default input arguments.
%        It is not optimized for memory efficiency or speed.

% Author: Phil Goddard (phil@goddardconsulting.ca)
% Date   : Q4, 2007

% Calculate the Cox Ross Rubinstein model parameters
a = exp(r*dt);
u = exp(sig*sqrt(dt));
d = 1/u;
p = (a-d)/(u-d);

% Loop over each node and calculate the Cox Ross Rubinstein underlying price tree
priceTree = nan(steps+1,steps+1);
priceTree(1,1) = S0;
for idx = 2:steps+1
    priceTree(1:idx-1,idx) = priceTree(1:idx-1,idx-1)*u;
    priceTree(idx,idx) = priceTree(idx-1,idx-1)*d;
end

% Calculate the value at expiry
valueTree = nan(size(priceTree));
switch oType
    case 'PUT'
        valueTree(:,end) = max(X-priceTree(:,end),0);
    case 'CALL'
        valueTree(:,end) = max(priceTree(:,end)-X,0);
end
```

```matlab
% Loop backwards to get values at the earlier times
steps = size(priceTree,2)-1;
for idx = steps:-1:1
    valueTree(1:idx,idx) = ...
        exp(-r*dt)*(p*valueTree(1:idx,idx+1) ...
        + (1-p)*valueTree(2:idx+1,idx+1));
    if earlyExercise
        switch oType
            case 'PUT'
                valueTree(1:idx,idx) = ...
                    max(X-priceTree(1:idx,idx),valueTree(1:idx,idx));
            case 'CALL'
                valueTree(1:idx,idx) = ...
                    max(priceTree(1:idx,idx)-X,valueTree(1:idx,idx));
        end
    end
end

% Output the option price
oPrice = valueTree(1);
```

Home | Financal Engineering | Control System Design | Software Tutorials | Services | About Us | Contact Us | Notices