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

CRR With Drift in MATLAB

This tutorial presents MATLAB code that implements the Cox-Ross-Rubinstein With Drift version of the binomial model as discussed in the Binomial model tutorial. A similar tutorial discussing the implementation of the original Cox-Ross-Rubinstein model in MATLAB is also available.

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 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*.

To use the CRR With Drift model an arbitrary drift term must be specified. Here a drift of $\eta = (ln(X)-ln(S_0))/T$ is chosen. This force the lattice to drift so that at expiry it is symmetric about the strike.

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

```
>> oPrice = binPriceCRRDrift(60,50,0.05,0.2,0.01,100,'CALL',false,log(60)-log(50))
oPrice =
      1.612
```

If the number of time steps is doubled then

```
>> oPrice = binPriceCRRDrift(60,50,0.05,0.2,0.005,200,'CALL',false,log(60)-log(50)
oPrice =
      1.618
```

For this particular option the CRR With Drift Approach does not perform any better than the other binomial models.

MATLAB Function: binPriceCRRDrift

```
function oPrice = ...
    binPriceCRRDrift(X,S0,r,sig,dt,steps,oType,earlyExercise,drift)
% Function to calculate the price of a vanilla European or American
% Put or Call option using a Cox-Ross-Ruinstein 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 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 CRR With Drift model parameters
a = exp(r*dt);
u = exp(drift*dt + sig*sqrt(dt));
d = exp(drift*dt - sig*sqrt(dt));
p = (a-d)/(u-d);

% Loop over each node and calculate the 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

% 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);
```

Back To Top | Option Pricing

© 2003-2019 Goddard Consulting.