The University of Reading

ISMA Centre - Numerical Methods 1 Project Report

Implied Binomial Trees

Gerhard Rebel

19 April 2004

1 Project Requirements

Chapter 7, "How Precise are Price Distributions Predicted by Implied Binomial Trees", by Hardle and Zheng in the Springer Verlag Applied Quantitative Finance book (ISBN: 3540434607) describes the construction of Derman and Kani (DK) and Barle and Cakici (BC) implied binomial trees (IBT) and analyses the precision of the predicted implied price distributions.

See files : Rebel_NM1P_Applied_Quantitative_Finance_Book.pdf

Rebel_NM1P_Volatility_Smile_and_its_Implied_Tree_Derman_1994.pdf

Rebel_NM1P_Growing_a_Smiling_Tree_Barle_1995.pdf

In the IBT option price modelling approach one first specifies a smile volatility surface, in terms of strike and time to maturity, which has been determined from observed option prices in the market. The binomial trees then need to produce option prices, which when inserted back in to the standard Black and Scholes option pricing equation, give implied volatilities which match those specified at the outset (i.e. by the smile volatility surface).

The objective of this project was to reproduce the results in the Hardle and Zheng paper in Microsoft Excel VBA and to compare the calculated results with those available in the paper and also by simulation using XploRe Quantlets. It is possible to run the XploRe statistical software using a remote Java client platform available through :

http://www.xplore-stat.de/java/java.html

See file : Rebel_NM1P_XploRe_Handbook_2004-03.pdf

It was agreed at the start of the project that the implementation should stop at the end of Section 7.2.2 and Figure 7.5 of Applied Quantitative Finance Chapter 7. Note that the relative efficiencies of the computer algorithms developed have not been considered in the project.

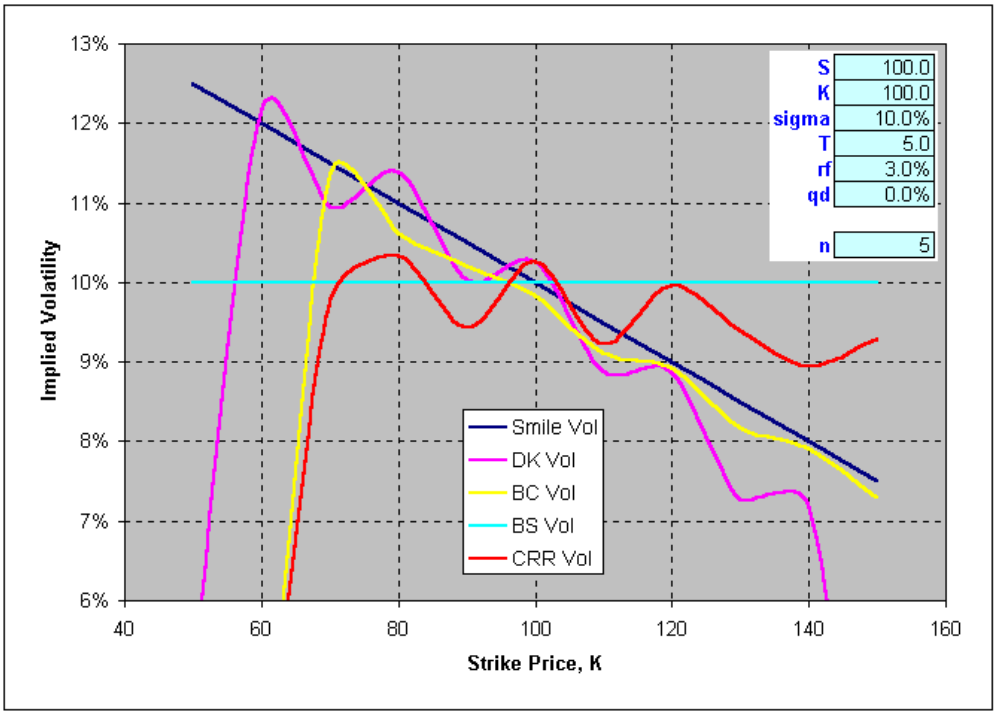
2 Main Calculation Worksheet - Call_Option_Implied_Vol

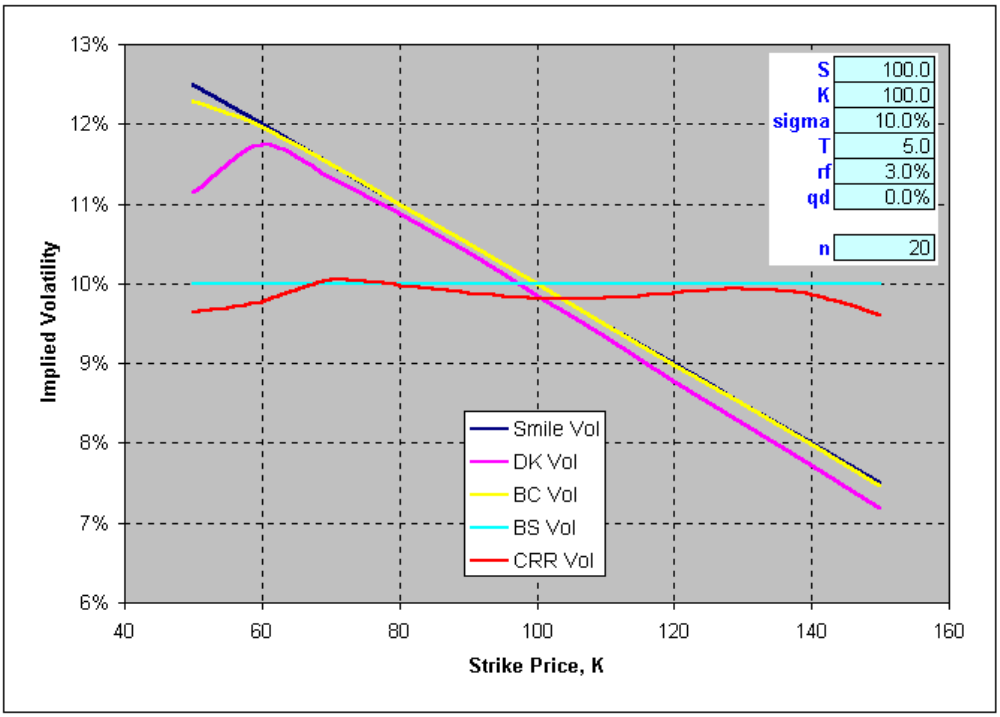
The purpose of the calculations in the sheet Call_Option_Implied_Vol is to show how the DK and BC implied binomial trees are capable of producing European call option prices that are consistent with implied volatility smiles observed in the market (i.e. as defined by an implied volatility surface). Using the functions described in Section 3 of this report, the sheet calculates the following results for given option and tree parameters (S, K, sigma, T, rf, qd, n):

- Black and Scholes (BS) call option price and its implied volatility (as a check on the implied volatility function).
- CRR (Cox, Ross and Rubinstein) binomial tree call option price and its implied volatility.
- DK implied binomial tree call option price and its implied volatility.
- BC implied binomial tree call option price and its implied volatility.
- Smile volatility as defined by the implied volatility surface.

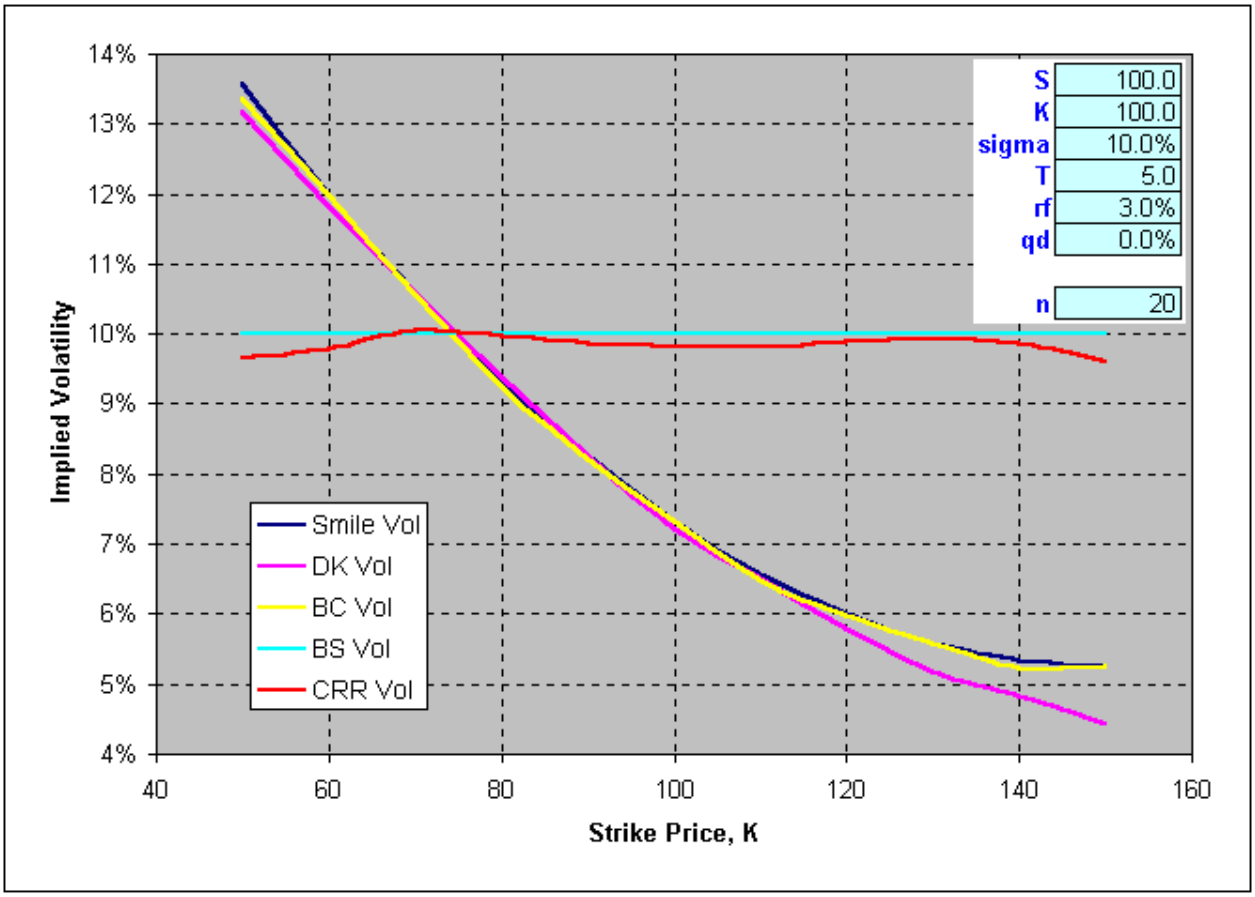
In theory, the CRR implied volatility should equal the constant volatility, sigma, specified in the construction of the CRR binomial tree. As the number of steps in the tree, n, becomes large, this was found to be the case. The DK and BC trees attempt to reproduce the implied volatility surface defined at the outset. In theory, the implied volatilities calculated from DK and BC tree option prices should match the smile / skew volatility. As number of steps in the trees, n, become large, the accuracies of the implied binomial trees improve significantly. In general the BC method is more precise than the DK method and is considerably faster since intermediate call and put option prices are calculated using the closed form BS approach rather than a CRR binomial tree.

An Excel data table is used to produce a set of results for strike prices between 50 and 150 and these results are presented on the same chart. The two figures below show the data table calculation results for 5-year, 5-step and 20-step DK, BC and CRR trees.





The previous two figures were for linear variations in the inputted smile (skew) volatility with the option strike price. The figure below indicates the results of 20-step, 5 year calculations with a smile which has a quadratic dependence on the option strike price. In all these examples, the BC method is more accurate at reproducing the specified smile volatility than the DK method.



3 Comparison of Project and Applied Quantitative Finance / XploRe Results

The results of the calculation procedures developed in this project have been validated against results in the Applied Quantitative Finance book Chapter 7 and through application of the online XploRe Quantlet Java Client software.

The 5-step, 5-year calculations of the sheet Sample_IBT_Output for stock prices, transition probabilities and Arrow-Debreu state prices at each node generally compare accurately with the XploRe results. The XploRe data is included on separate pages with the Sample_IBT_Output sheet output in Section 6 of this document. Slightly larger variations in data are evident for the DK results than for the BC values. This is most likely due to the CRR binomial tree in this project returning different intermediate option prices compared to the CRR tree used by XploRe (see XploRe function IBTcrr). It could be that they have used a different compounding convention or that their CRR parameters are formulated differently. For the same reason, the DK results do not compare exactly with those presented by Derman and Kani in their 1994 Goldman Sachs paper, page 11, Figure 6.

The sheet DK_BC_Trees compares the Implied_Bin_Option_Price(…) function stock price, transition probability and Arrow-Debreu state price output with the data presented in the Applied Quantitative Finance book (also included in the sheet). The results compare accurately.

Terminal stock price probability densities for 20-step, 5-year, DK and BC trees are compared with the equivalent XploRe calculations on the XploRe_Results sheet. As with the 5-step, 5-year calculations of sheet Sample_IBT_Output, here the BC values are more accurate than the DK values, particularly for lower terminal stock prices. This could be attributed to different CRR binomial tree option prices and also slightly different implementation of the compensation against arbitrage corrections that are made to ensure that all node transition probabilities remain within the range 0 to 1.

The sheet DK_BC_Terminal_Prob_Densities shows the DK and BC probability densities which are the results in Figures 7.2 (DK) and 7.4 (BC) of the Applied Quantitative Finance book.

Sheet AQF_IBTlocsigma compares the results of local implied volatility produced by the Implied_Bin_Option_Price(…) function with results obtained with the XploRe function IBTlocsigma. There is an almost perfect correlation between the two sets of results. Related to this, the Implied_Local_Volatility sheet shows calculations and charts which represent the data in Figures 7.3 (DK) and 7.5 (BC) of the Applied Quantitative Finance book for local implied volatilities in 20-step, 5-year trees.

4 Description of Functions Developed

The following functions are contained in the Main_IBT VBA module of the file:

Rebel_NM1P_Implied_Binomial_Trees.xls

Function Implied_Bin_Option_Price(S, K, T, rf, qd, n, DK_or_CB, Call_or_Put, Output_Format)

Implied binomial tree European option pricing implementation function using continuous compounding for the Derman and Kani (DK) and Barle and Cakici (BC) methods. For examples of typical function output, see sheet Sample_IBT_Output.

The sheet DK_BC_Trees compares this function's stock price, transition probability and Arrow-Debreu state price output with the data presented in the Applied Quantitative Finance book (also included in the sheet). Terminal stock price probability densities for 20-step, 5-year, DK and BC trees, as determined in this project, are compared with the results from equivalent XploRe calculations on the XploRe_Results sheet. The sheet DK_BC_Terminal_Prob_Densities shows the DK and BC probability densities which are the results in Figures 7.2 (DK) and 7.4 (BC) of the Applied Quantitative Finance book. Sheet

AQF_IBTlocsigma compares the results of local implied volatility produced by this function with results obtained with the XploRe function IBTlocsigma. Related to this, the Implied_Local_Volatility sheet shows calculations and figures which represent the data in Figures 7.3 (DK) and 7.5 (BC) of the Applied Quantitative Finance book for local implied volatilities in 20-step, 5-year trees. Sheet BC_100_Step_Example contains results of the estimated terminal probability density of a 100-step, 5-year, BC tree compared with the lognormal probability density. The difference between the two continuous densities is also indicated (i.e. as per Derman and Kani, 1994).

Inputs:

S = stock price
K = option strike price
T = time to expiry (years)
rf = annual risk-free interest rate in decimal form (e.g. 0.1 = 10%)
qd = annual continuous dividend rate in decimal form (e.g. 0.05 = 5%)
n = number of steps in the tree

DK_or_CB = 0 is for DK
DK_or_CB = 1 is for BC

Call_or_Put = 0 is for a call option
Call_or_Put = 1 is for a put option

Note that this function also considers the specified smile volatility, as defined in the module function Smile_Vol(S, K, T), when determining the structure of the DK and BC trees.

Outputs:

Call or put option price (Output_Format = 1)

> **Format:** scalar value, same units as S and K

Stock price at each node (Output_Format = 2)

> **Format:** Matrix with n + 1 rows and n + 1 columns.
>
> S_node(0, 0) = S
>
> Stock prices on the same level (row) are up movements.
> The columns represent successive time steps to maturity.
>
> Note that j are the discrete time steps from 0 to T (columns) and i are the vertical nodes (rows) at each time step e.g. S_node(i, j).
> S_node(0, 0) is the root of the tree at time 0.
> S_node(0, n) is the highest stock price at expiry time T from a series of up movements only.
> S_node(n, n) is the lowest stock price at expiry time T from a series of down movements only.

Transition probability at each node (Output_Format = 3)

> **Format:** Matrix with n rows and n columns.
>
> Compensation against arbitrage and enforcement of
> 0 < P_node_trans < 1 has been implemented. A warning message
> is given when P_node_trans is not within these specified limits.
> Transition probabilities are the probabilities of an up movement in the stock price.

Arrow-Debreu state price at each node (Output_Format = 4)

Format: Matrix with n + 1 rows and n + 1 columns.

Lamda_node(0, 0) = 1

This is the price of an option that pays 1 unit only at the node where it is calculated (i, j) and zero elsewhere.

Probability of reaching each node (Output_Format = 5)

Format: Matrix with n + 1 rows and n + 1 columns.

P_node(0, 0) = 1

Terminal stock prices, cumulative probabilities and terminal node probabilities (Output_Format = 6)

This output format is required as a range input into the function Est_Continuous_Prob(…) which is described below. Given a set of discrete terminal node probabilities and the associated cumulative probability distribution, Est_Continuous_Prob(…) produces an estimate of the continuous terminal probability density.

Format: Matrix with n + 1 rows and 3 columns. The first column contains the terminal stock prices, the second column the (discrete) cumulative probabilities and the third column the corresponding terminal node probabilities (discrete). Note that the row order is reversed when compared with the other related output matrices. This is a requirement for input into the function Est_Continuous_Prob(…).

Forward price at each node (Output_Format = 7)

Format: Matrix with n + 1 rows and n + 1 columns.

The first column and last row are not used.

Implied local volatility at each node for the time to expiry = dt = T/n (Output_Format = 8)

This data contains some of the more detailed values computed under Output_Format 9.

Format: Matrix with n rows and n columns.

Stock prices, times to expiry and implied local volatilities (Output_Format = 9)

Calculates the local volatilities implied by the implied binomial tree as per :
http://www.xplore-stat.de/help/IBTlocsigma.html and the Applied Quantitative Finance PDF Book, Equation (7.17). This implementation assumes that the temporary node probability P_node_temp(i, j) = 1 where (i, j) corresponds to the node (and stock price) for which the implied volatility is being calculated, i.e. S_node(i, j). This output option is only available if n <= 20 otherwise the size of the matrix becomes too large (note the number of rows below).

Format: Matrix with n * (n + 1) * (n + 2) / 6 rows and 3 columns. Column 1 is the stock prices at all the nodes in the tree up to the last time step before expiry. Column 2 is the corresponding time to expiry in years and column 3 the implied local volatilities in decimal form percentages.

Function Est_Continuous_Prob(XY_range, n, Output_Format)

Approximates the continuous probability density using a cubic spline fit to the discrete cumulative probability distribution determined from the terminal node probabilities of a DK, BC or CRR binomial tree. Where the number of rows in the XY_range becomes large, the accuracy of the estimated probability density increases. This has been demonstrated on the sheet CRR_Prob_Density which shows how the estimated probability density determined with this function converges on the lognormal density for a 500-step CRR binomial tree. The estimated probability density is the first derivative of the cumulative probability distribution with respect to the stock price.

The function creates an array of data points for the cubic spline evaluation such that the first point equals the minimum terminal stock price and the last point equals the maximum terminal stock price. There are n+1 points in total. The stock prices need to be ascending. Note that the variation between minimum and maximum stock prices is quadratic. This is based on observed variations in terminal stock prices for the IBT and CRR trees when viewed with uniform spacing on the X axis. With a linear variation in the cubic spline evaluation points it was not possible to accurately describe the probability density data in the range of meaningful stock prices. See the sheet CRR_Prob_Density for an example [i.e. CRR_Bin_Option_Price(100, 100, 5, 0.1, 0.03, 0, 500, 0, 5)]. There the terminal stock prices range from 0.67 to 14841 and the area of interest is only between 50 and 230.

Inputs:

XY_range = range of XY values with the terminal node stock prices in the first column and the corresponding cumulative probabilities (discrete) in the second column. There should be at least two rows but in practice sensible results can only be expected if more than for example 20 rows are used. Note that the required input for this range is produced by the first two columns of the output option 6 of the function Implied_Bin_Option_Price(…) and the first two columns of the output option 5 of the function CRR_Bin_Option_Price(…).

n = number of steps in the quadratic variation between the minimum terminal stock price and the maximum terminal stock price in the first column of XY_range. Note that this n is not related to the n in the functions Implied_Bin_Option_Price(…) and CRR_Bin_Option_Price(…) which generate the inputs for XY_range. It is quite acceptable to have 20 steps in the inputted XY_range data but say n = 40 steps for the cubic spline evaluation. The sheet DK_BC_Terminal_Prob_Densities illustrates this clearly.

Outputs:

Output_Format = 0 has the stock price as column 1, the fitted cumulative distribution as column 2 and the continuous probability density as column 3. Note that the stock prices do not match the terminal node stock prices of the input data XY_range, only the first and last values are the same.

Format: Matrix with n + 1 rows and 3 columns.

Output_Format = 1 has the stock price as column 1 and the continuous probability density as column 2. This format is needed for the function SplineIntegrate(...) in module Cubic_Spline. The integral from -ve infinity to +ve infinity of the probability density should equal 1. This is a useful cross check on the estimated probability density. The sheet DK_BC_Terminal_Prob_Densities shows the results of such integration for DK, BC, CRR and lognormal probability densities. In all cases the integrals equal 1.000.

Format: Matrix with n + 1 rows and 2 columns.

Function CBS(S, K, T, sigma, rf, qd)

Black and Scholes closed form call option pricing implementation with continuous dividend rate qd.

Inputs:

S = stock price
K = option strike price
T = time to expiry (years)
rf = annual risk-free interest rate in decimal form (e.g. 0.1 = 10%)
qd = annual continuous dividend rate in decimal form (e.g. 0.05 = 5%)

Output:

Format: scalar value, Black and Scholes call option price, same units as S and K.

Function PBS(S, K, T, sigma, rf, qd)

Black and Scholes closed form put option pricing implementation with continuous dividend rate qd.

Inputs:

S = stock price
K = option strike price
T = time to expiry (years)
rf = annual risk-free interest rate in decimal form (e.g. 0.1 = 10%)
qd = annual continuous dividend rate in decimal form (e.g. 0.05 = 5%)

Output:

Format: scalar value, Black and Scholes put option price, same units as S and K.

Function Imp_Vol(S, K, T, rf, qd, Option_Price, Tol, Call_or_Put)

Function to calculate the Black and Scholes volatility implied by a given call or put option price using a simple bisection method. Starting values of High_sigma = 2 (i.e. 200%) Low_sigma = 0 are assumed internally in the function. Several examples of the application of this function are given on the sheet Call_Option_Implied_Vol.

Inputs:

S = stock price
K = option strike price
T = time to expiry (years)
rf = annual risk-free interest rate in decimal form (e.g. 0.1 = 10%)
qd = annual continuous dividend rate in decimal form (e.g. 0.05 = 5%)
Option_Price = observed put or call option price
Tol = tolerance between the High_sigma and the Low_sigma to be reached before the iteration stops. A typical value for the tolerance is 0.00001.

Call_or_Put = 0 is for a call option
Call_or_Put = 1 is for a put option

Output:

Format: scalar value, the implied Black and Scholes volatility in decimal form percentage.

Function CRR_Bin_Option_Price(S, K, T, sigma, rf, qd, n, Call_or_Put, Output_Format)

CRR binomial tree call and put option pricing implementation function with continuous dividend rate qd. Note that the function uses continuous compounding. For the node transition probability to lie between 0 and 1 : sigma /(rf - qd) > $dt^{0.5}$ If sigma / (rf - qd)) <= $dt^{0.5}$ then a CRR parameter error message is given. For examples of typical function output, see the sheet Sample_IBT_Output.

Inputs:

S = stock price
K = option strike price
T = time to expiry (years)
sigma = constant volatility in decimal form (e.g. 0.2 = 20%)
rf = annual risk-free interest rate in decimal form (e.g. 0.1 = 10%)
qd = annual continuous dividend rate in decimal form (e.g. 0.05 = 5%)
n = number of steps in the tree

Call_or_Put = 0 is for a call option
Call_or_Put = 1 is for a put option

Outputs:

Call or put option price (Output_Format = 1)

Format: scalar value, same units as S and K

Stock price at each node (Output_Format = 2)

Format: Matrix with n + 1 rows and n + 1 columns.

S_node(0, 0) = S

Stock prices on the same level (row) are up movements.
The columns represent successive time steps to maturity.

Note that j are the discrete time steps from 0 to T (columns) and i are the vertical nodes (rows) at each time step e.g. S_node(i, j).
S_node(0, 0) is the root of the tree at time 0.
S_node(0, n) is the highest stock price at expiry time T from a series of up movements only.
S_node(n, n) is the lowest stock price at expiry time T from a series of down movements only.

Probability at each node (Output_Format = 3)

Format: Matrix with n + 1 rows and n + 1 columns.

P_node(0, 0) = 1

Arrow-Debreu state price at each node (Output_Format = 4)

Format: Matrix with n + 1 rows and n + 1 columns.

Lamda_node(0, 0) = 1

This is the price of an option that pays 1 unit only at the node where it is calculated (i, j) and zero elsewhere.

Terminal stock prices, cumulative probabilities and terminal node probabilities (Output_Format = 5)

This output format is required as a range input into the function Est_Continuous_Prob(…) which is described above. Given a set of discrete terminal node probabilities and the associated cumulative probability distribution, Est_Continuous_Prob(…) produces an estimate of the continuous terminal probability density.

Format: Matrix with n + 1 rows and 3 columns. The first column contains the terminal stock prices, the second column the (discrete) cumulative probabilities and the third column the corresponding terminal node probabilities (discrete). Note that the row order is reversed when compared with the other related output matrices. This is a requirement for the function Est_Continuous_Prob(…).

Function Smile_Vol(S, K, T)

Function to return the volatility according to the equation which defines the implied volatility surface in terms of the strike, K, and the time to maturity, T, as determined from market option price data. The implied volatility surface is a required input for the DK and BC binomial option pricing trees. An example of the application of this function is given on the sheet Call_Option_Implied_Vol.

To change the market implied volatility surface, the equation in the module function needs to be altered manually. Below are four examples of equations that have been used in this investigation. Note that none of these are functions of time:

Smile_Vol = 0.15 - 0.0005 * K, i.e. assuming S = 100

Smile_Vol = 0.1 + (S - K) / S / 10 * 0.5, i.e. as per XploRe Handbook, page 58 lBTdk

Smile_Vol = 1 / 120000 * K ^ 2 - 1 / 400 * K + 0.24, i.e. a quadratic variation with strike price

Smile_Vol = 0.1, i.e. no smile or skew, just constant volatility

Inputs:

S = stock price
K = option strike price
T = time to expiry (years)

Output:

Format: scalar value, volatility in decimal form percentage.

Function Log_Normal_Density(S0, ST, T, sigma, rf, qd)

Function to return the lognormal probability density with constant volatility as per the Applied Quantitative Finance PDF Book, Equation (7.1).

Inputs:

S0 = stock price at time 0
ST = stock price at time T
T = time to expiry (years)
sigma = constant volatility in decimal form (e.g. 0.2 = 20%)
rf = annual risk-free interest rate in decimal form (e.g. 0.1 = 10%)
qd = annual continuous dividend rate in decimal form (e.g. 0.05 = 5%)

Output:

Format: scalar value, continuous lognormal probability density.

---

5 VBA Program Module Listing - Main_IBT

The following pages contain the Main_IBT VBA module with the functions described in the previous section. Note that the file Rebel_NM1P_Implied_Binomial_Trees.xls contains a second module which allows for fitting of cubic splines to given XY range data and the differentiation and integration of the resulting cubic functions. The functions in the Cubic_Spline module were taken directly from the Google Groups 14 November 2000 microsoft.public.excel.programming posting by David Braden. These cubic spline functions were not modified in this project and were used as they were given in the Google listing.

```
' Rebel_NM1P_Implied_Binomial_Trees.xls - Module: Main_IBT - GERHARD REBEL

Option Explicit

' ------------------------------------------------------------------
' Implied Binomial Tree European option pricing implementation
' DK_or_CB = 0 is for DK
' DK_or_CB = 1 is for BC
' Call_or_Put = 0 is Call Option
' Call_or_Put = 1 is Put Option
' See as references: XploRe function XFGIBTcbc, XploRe function XFGIBTcdk and
' CASS Business School, London, March 2004 online Course Notes and XLS examples on DK and BC Implied Binomial Trees by Mike Staunton
' Note that j are the discrete time steps from 0 to T (columns) and i are the vertical nodes (rows) at each time step e.g. S_node(i, j).
' See sheet Sample_IBT_Output for meaning of the i and j array indices.
' S_node(0, 0) is the root of the tree at time 0.
' S_node(0, n) is the highest stock price at expiry time T from a series of up movements only.
' S_node(n, n) is the lowest stock price at expiry time T from a series of down movements only.
' ------------------------------------------------------------------

Function Implied_Bin_Option_Price(S, K, T, rf, qd, n, DK_or_CB, Call_or_Put, Output_Format)

    ' ------------------------------------------------------------------
    ' Define the variables
    ' ------------------------------------------------------------------

    Dim dt, time_to_exp, infl_no_qd, infl_with_qd, T_node, K_temp, S_temp

    Dim Option_Price, Output_Option_Price, rho_sum, S_numer, S_denom

    Dim i As Integer, i_start As Integer, i_below As Integer, i_above As Integer, j As Integer, i_sum As Integer

    Dim Output_array, Cum_sum

    Dim row_count As Integer

    Dim v As Integer, w As Integer

    Dim mean_log_price_sum

    Dim log_variance_sum

    Dim C_sum, P_sum

    ' ------------------------------------------------------------------
    ' Create required matrices / arrays
    ' ------------------------------------------------------------------

    ReDim Forward_price(0 To n, 0 To n)  ' Forward prices at each node

    ReDim Local_impl_vol_dt(0 To n - 1, 0 To n - 1)  ' note the reduced size of this array

    Dim max_impl_vol_rows As Integer

    ' We need to limit the size of the Local_impl_vol matrix, note that max_impl_vol_rows becomes very large

    If n <= 20 Then
```

```
    max_impl_vol_rows = n * (n + 1) * (n + 2) / 6 - 1 ' i.e. from loop For j = 0 To n - 1, For i = 0 To j, For w = j + 1 To n

    ReDim Local_impl_vol(0 To max_impl_vol_rows, 0 To 2)

    ReDim P_node_temp(0 To n, 0 To n)  ' Temp probabilities of reaching each node, used in the calculation of Local_impl_vol(...)

End If

ReDim Lamda_node(0 To n, 0 To n)  ' Arrow-Debreu state prices

ReDim P_node(0 To n, 0 To n)  ' Probabilities of reaching each node in the tree

ReDim Output_array(0 To n, 0 To n)  ' temp array for formatting output data

ReDim P_node_trans(0 To n - 1, 0 To n - 1)  ' Transition probabilities at each node, note the reduced size of this array

ReDim S_node(0 To n, 0 To n)  ' Stock prices at each node

' --------------------------------------------------------------------
' Tree parameters (note the continuous compounding)
' --------------------------------------------------------------------

dt = T / n

infl_no_qd = Exp(rf * dt)

infl_with_qd = Exp((rf - qd) * dt)

' --------------------------------------------------------------------
' Set the starting values at the base of the tree
' --------------------------------------------------------------------

S_node(0, 0) = S

Lamda_node(0, 0) = 1

P_node(0, 0) = 1

T_node = 0

' --------------------------------------------------------------------
' Calculate DK and BC tree node stock prices and other values
' --------------------------------------------------------------------

For j = 1 To n  ' the time steps

    T_node = T_node + dt  ' time at each node

    ' ----------------------------------------------------------------
    ' Calculate the forward prices at the vertical nodes
    ' ----------------------------------------------------------------

    For i = 0 To j - 1  ' vertical nodes

        Forward_price(i, j) = S_node(i, j - 1) * infl_with_qd
```

```
Next i

' ---------------------------------------------------------------------
' Determine the vertical nodes for calculating the central, above and below tree data
' ---------------------------------------------------------------------

i_start = Int((j + 0.0001) / 2)  ' central nodes

i_below = Int((j + 1.0001) / 2 + 1)  ' nodes below the centre (decreasing stock prices)

If i_below > j Then i_below = j

i_above = i_start - 1  ' nodes above the centre (increasing stock prices)

If i_above < 0 Then i_above = 0

' ---------------------------------------------------------------------
' Calculate the tree parameters for the nodes at the centre of the tree
' ---------------------------------------------------------------------

i = i_start

If j Mod 2 = 0 Then  ' i.e. j is even so we have only one central node

    If DK_or_CB = 0 Then  ' DK

        S_node(i, j) = S

    Else  ' BC

        S_node(i, j) = S * (infl_with_qd ^ j)

    End If

    ' ---------------------------------------------------------------------
    ' Compensation against arbitrage and ensure 0 < P_node_trans < 1
    ' Applied Quantitative Finance PDF Book, Equation (7.15)
    ' ---------------------------------------------------------------------

    If i > 0 And i < j Then

        If S_node(i, j) <= Forward_price(i, j) Or S_node(i, j) >= Forward_price(i - 1, j) Then

            S_node(i, j) = (Forward_price(i, j) + Forward_price(i - 1, j)) / 2  ' i.e. the average of the two

        End If

    End If

Else  ' j is odd and there are two central nodes

    If DK_or_CB = 0 Then  ' DK

        K_temp = S_node(i, j - 1)
```

```
            S_temp = S
            Option_Price = CRR_Bin_Option_Price(S, K_temp, T_node, Smile_Vol(S, K_temp, T_node), rf, qd, j, 0, 1)
        Else  ' BC
            K_temp = Forward_price(i, j)
            S_temp = K_temp
            Option_Price = CBS(S, K_temp, T_node, Smile_Vol(S, K_temp, T_node), rf, qd)
        End If
        rho_sum = 0
        If j > 2 Then
            For i_sum = 0 To i - 1  ' Applied Quantitative Finance PDF Book, Equation (7.12 and 7.19)
                rho_sum = rho_sum + Lamda_node(i_sum, j - 1) * (Forward_price(i_sum, j) - K_temp)
            Next i_sum
        End If
        S_numer = S_temp * (infl_no_qd * Option_Price + Lamda_node(i, j - 1) * K_temp - rho_sum)
        S_denom = Lamda_node(i, j - 1) * Forward_price(i, j) - infl_no_qd * Option_Price + rho_sum
        S_node(i, j) = S_numer / S_denom  ' Applied Quantitative Finance PDF Book, Equation (7.11 and 7.18)
        S_node(i + 1, j) = S_temp * S_temp / S_node(i, j)
        ' ---------------------------------------------------------------------
        ' Compensation against arbitrage and ensure 0 < P_node_trans < 1
        ' Applied Quantitative Finance PDF Book, Equation (7.15)
        ' ---------------------------------------------------------------------
        If i > 0 And i < j Then
            If S_node(i, j) <= Forward_price(i, j) Or S_node(i, j) >= Forward_price(i - 1, j) Then
                S_node(i, j) = (Forward_price(i, j) + Forward_price(i - 1, j)) / 2
            End If
            If S_node(i + 1, j) <= Forward_price(i + 1, j) Or S_node(i + 1, j) >= Forward_price(i, j) Then
                S_node(i + 1, j) = (Forward_price(i + 1, j) + Forward_price(i, j)) / 2
            End If
        End If
    End If
```

```
' ------------------------------------------------------------------
' Calculate the tree parameters for the nodes above the centre (increasing stock prices)
' ------------------------------------------------------------------

For i = i_above To 0 Step -1  ' vertical nodes

    If DK_or_CB = 0 Then  ' DK

        K_temp = S_node(i, j - 1)

        Option_Price = CRR_Bin_Option_Price(S, K_temp, T_node, Smile_Vol(S, K_temp, T_node), rf, qd, j, 0, 1)

    Else  ' BC

        K_temp = Forward_price(i, j)

        Option_Price = CBS(S, K_temp, T_node, Smile_Vol(S, K_temp, T_node), rf, qd)

    End If

    S_temp = S_node(i + 1, j)

    rho_sum = 0

    For i_sum = 0 To i - 1  ' Applied Quantitative Finance PDF Book, Equation (7.12 and 7.19)

        rho_sum = rho_sum + Lamda_node(i_sum, j - 1) * (Forward_price(i_sum, j) - K_temp)

    Next i_sum

    S_numer = S_temp * (infl_no_qd * Option_Price - rho_sum) - Lamda_node(i, j - 1) * K_temp * (Forward_price(i, j) - S_temp)

    S_denom = infl_no_qd * Option_Price - rho_sum - Lamda_node(i, j - 1) * (Forward_price(i, j) - S_temp)

    S_node(i, j) = S_numer / S_denom  ' Applied Quantitative Finance PDF Book, Equation (7.13 and 7.20)

    ' ------------------------------------------------------------------
    ' Compensation against arbitrage and ensure 0 < P_node_trans < 1
    ' Applied Quantitative Finance PDF Book, Equation (7.15)
    ' ------------------------------------------------------------------

    If j > 1 And i = 0 Then

       If S_node(i, j) <= Forward_price(i, j) Then

            S_node(i, j) = S_node(i + 1, j) * Forward_price(i, j) / Forward_price(i + 1, j)

        End If

    End If

    If i > 0 And i < j Then

       If S_node(i, j) <= Forward_price(i, j) Or S_node(i, j) >= Forward_price(i - 1, j) Then
```

```
            S_node(i, j) = (Forward_price(i, j) + Forward_price(i - 1, j)) / 2

        End If

    End If

Next i

' ---------------------------------------------------------------------
' Calculate the tree parameters for the nodes below the centre (decreasing stock prices)
' ---------------------------------------------------------------------

For i = i_below To j  ' vertical nodes

    If DK_or_CB = 0 Then  ' DK

        K_temp = S_node(i - 1, j - 1)

        Option_Price = CRR_Bin_Option_Price(S, K_temp, T_node, Smile_Vol(S, K_temp, T_node), rf, qd, j, 1, 1)  ' Note Put Option

    Else  ' BC

        K_temp = Forward_price(i - 1, j)

        Option_Price = PBS(S, K_temp, T_node, Smile_Vol(S, K_temp, T_node), rf, qd)  ' Note Put Option

    End If

    S_temp = S_node(i - 1, j)

    rho_sum = 0

    For i_sum = i To j - 1  ' Applied Quantitative Finance PDF Book, Equation (7.12 and 7.19)

        rho_sum = rho_sum + Lamda_node(i_sum, j - 1) * (K_temp - Forward_price(i_sum, j))

    Next i_sum

    S_numer = S_temp * (infl_no_qd * Option_Price - rho_sum) + Lamda_node(i - 1, j - 1) * K_temp * (Forward_price(i - 1, j) - S_temp)

    S_denom = infl_no_qd * Option_Price - rho_sum + Lamda_node(i - 1, j - 1) * (Forward_price(i - 1, j) - S_temp)

    S_node(i, j) = S_numer / S_denom  ' Applied Quantitative Finance PDF Book, Equation (7.14 and 7.21)

    ' ---------------------------------------------------------------------
    ' Compensation against arbitrage and ensure 0 < P_node_trans < 1
    ' Applied Quantitative Finance PDF Book, Equation (7.15)
    ' ---------------------------------------------------------------------

    If i > 0 And i < j Then

       If S_node(i, j) <= Forward_price(i, j) Or S_node(i, j) >= Forward_price(i - 1, j) Then

            S_node(i, j) = (Forward_price(i, j) + Forward_price(i - 1, j)) / 2

        End If
```

```
    End If


    If i > 1 And i = j Then

       If S_node(i, j) >= Forward_price(i - 1, j) Then

            S_node(i, j) = S_node(i - 1, j) * Forward_price(i - 1, j) / Forward_price(i - 2, j)

        End If

    End If

Next i

' ---------------------------------------------------------------------
' Calculate the transition probabilities
' Applied Quantitative Finance PDF Book, Equation (7.5)
' Calculate the implied local volatilities the time to expiry = the time step, dt = T/n
' ---------------------------------------------------------------------

For i = 0 To j - 1  ' vertical nodes

    P_node_trans(i, j - 1) = (Forward_price(i, j) - S_node(i + 1, j)) / (S_node(i, j) - S_node(i + 1, j))

    If P_node_trans(i, j - 1) <= 0 Or P_node_trans(i, j - 1) >= 1 Then

        MsgBox "Caution: P_node_trans(" & i & ", " & j - 1 & ")= " & P_node_trans(i, j - 1)

    End If

    ' This is for the time to expiry = the time step dt, see page 152 Applied Quantitative Finance PDF Book, just below equation (7.16):

    Local_impl_vol_dt(i, j - 1) = (P_node_trans(i, j - 1) * (1 - P_node_trans(i, j - 1))) ^ 0.5 * Log(S_node(i, j) / S_node(i + 1, j))

Next i

' ---------------------------------------------------------------------
' Calculate the Arrow-Debreu state prices Applied Quantitative Finance PDF Book, Equations (7.6)
' ---------------------------------------------------------------------

Lamda_node(j, j) = Lamda_node(j - 1, j - 1) * (1 - P_node_trans(j - 1, j - 1)) / infl_no_qd

For i = j - 1 To 1 Step -1  ' vertical nodes

    Lamda_node(i, j) = (Lamda_node(i, j - 1) * _
                        P_node_trans(i, j - 1) + Lamda_node(i - 1, j - 1) * (1 - P_node_trans(i - 1, j - 1))) / infl_no_qd

Next i

Lamda_node(0, j) = Lamda_node(0, j - 1) * P_node_trans(0, j - 1) / infl_no_qd

' ---------------------------------------------------------------------
' Calculate the probabilities of reaching each node in the tree
' ---------------------------------------------------------------------
```

```
    For i = 0 To j  ' vertical nodes

       P_node(i, j) = Lamda_node(i, j) * infl_no_qd ^ j   ' from the definition of Arrow-Debreu state price

    Next i

Next j

' ---------------------------------------------------------------------
' Calculate the local volatilities implied by the implied binomial tree
' See : http://www.xplore-stat.de/help/IBTlocsigma.html for comparison
' See also page 152 Applied Quantitative Finance PDF Book, Equation (7.17)
' This implementation assumes that the temp node probability P_node_temp(i, j) = 1
' where (i,j) corresponds to the node (i.e. stock price) for which the implied
' volatility is being calculated i.e. S_node(i, j)
' ---------------------------------------------------------------------

If n <= 20 Then  ' otherwise the size of the Local_impl_vol array becomes too large, see max_impl_vol_rows

    row_count = 0

    For j = 0 To n - 1  ' time steps

        For i = 0 To j  ' vertical nodes

            time_to_exp = dt  ' used later in the output array

            ReDim P_node_temp(0 To n, 0 To n) ' this is required to clear the contents of P_node_temp, do not move it to top of function

            P_node_temp(i, j) = 1  ' we assume that the probability of being at S_node(i, j) is 1

            For w = j + 1 To n  ' the possible time steps (or points) at which local volatilities can be calculated

                ' Calculate the temporary node probabilities

                P_node_temp(i, w) = P_node_temp(i, w - 1) * P_node_trans(i, w - 1)

                For v = i + 1 To w - 1

                    P_node_temp(v, w) = P_node_temp(v, w - 1) * P_node_trans(v, w - 1) + _
                                        P_node_temp(v - 1, w - 1) * (1 - P_node_trans(v - 1, w - 1))

                Next v

                P_node_temp(w, w) = P_node_temp(w - 1, w - 1) * (1 - P_node_trans(w - 1, w - 1))

                ' Calculate the expected value of the log stock price, Applied Quantitative Finance PDF Book, Equation (7.17)

                mean_log_price_sum = 0

                For v = i To w

                    mean_log_price_sum = mean_log_price_sum + P_node_temp(v, w) * Log(S_node(v, w))

                Next v
```

```
                ' Calculate the log variance, Applied Quantitative Finance PDF Book, Equation (7.17)
                log_variance_sum = 0
                For v = i To w
                    log_variance_sum = log_variance_sum + P_node_temp(v, w) * (Log(S_node(v, w)) - mean_log_price_sum) ^ 2
                Next v
                ' Write the calculated values into the output array
                Local_impl_vol(row_count, 0) = S_node(i, j) ' stock price
                Local_impl_vol(row_count, 1) = time_to_exp ' time to expiry for calculation of the implied local volatility
                Local_impl_vol(row_count, 2) = log_variance_sum ^ 0.5 ' implied local volatility from variance
                row_count = row_count + 1
                time_to_exp = time_to_exp + dt
            Next w
        Next i
    Next j
End If  '  n <= 20

' ---------------------------------------------------------------------
' Calculate the price of the call using the terminal AD state
' prices and the terminal call payoffs
' Applied Quantitative Finance PDF Book, Equations (7.9) and (7.10)
' ---------------------------------------------------------------------

C_sum = 0

P_sum = 0

For i = 0 To n  ' terminal vertical nodes

    C_sum = C_sum + Lamda_node(i, n) * Application.Max(S_node(i, n) - K, 0)

    P_sum = P_sum + Lamda_node(i, n) * Application.Max(K - S_node(i, n), 0)

Next i

If Call_or_Put = 0 Then  ' Call option

    Output_Option_Price = C_sum

Else  ' Put option

    Output_Option_Price = P_sum
```

```
End If

' ---------------------------------------------------------------------
' Control the format of the output from this function
' ---------------------------------------------------------------------

If Output_Format = 1 Then

    Implied_Bin_Option_Price = Output_Option_Price

ElseIf Output_Format = 2 Then

    Implied_Bin_Option_Price = S_node  ' Stock prices at each node

ElseIf Output_Format = 3 Then

    Implied_Bin_Option_Price = P_node_trans  ' Transition probabilities at each node

ElseIf Output_Format = 4 Then

    Implied_Bin_Option_Price = Lamda_node  ' AD state prices at each node

ElseIf Output_Format = 5 Then

    Implied_Bin_Option_Price = P_node  ' Probabilities of reaching each node

ElseIf Output_Format = 6 Then  ' Terminal stock prices, cumulative probabilities and terminal node probabilities

    Cum_sum = 1

    ReDim Output_array(0 To n, 0 To 2)

    For i = 0 To n  ' terminal vertical nodes (reverse order required for Function Est_Continuous_Prob(...))

        Output_array(n - i, 0) = S_node(i, n)

        Output_array(n - i, 1) = Cum_sum

        Output_array(n - i, 2) = P_node(i, n)  ' Note that these are discrete probabilities

        Cum_sum = Cum_sum - P_node(i, n)

    Next i

    Implied_Bin_Option_Price = Output_array

ElseIf Output_Format = 7 Then

    Implied_Bin_Option_Price = Forward_price  ' Forward prices at each node

ElseIf Output_Format = 8 Then

    Implied_Bin_Option_Price = Local_impl_vol_dt  ' Implied local volatilities at each node for the time to expiry = dt

ElseIf Output_Format = 9 Then  ' Stock prices, times to expiry and implied local volatilities
```

```
        If n <= 20 Then

            Implied_Bin_Option_Price = Local_impl_vol

        Else

            Implied_Bin_Option_Price = "Error"  ' this output option does not exist for n > 20

        End If

    ElseIf Output_Format = 10 Then  ' Stock prices, implied local volatilities for time step dt just before terminal values,
                                    ' annualised local volatilities (output option not in use, for debug / testing only)

        ReDim Output_array(0 To n - 1, 0 To 2)

        For i = 0 To n - 1  ' vertical nodes, just before terminal column

            Output_array(i, 0) = S_node(i, n - 1)

            Output_array(i, 1) = Local_impl_vol_dt(i, n - 1)

            Output_array(i, 2) = Local_impl_vol_dt(i, n - 1) * (1 / dt) ^ 0.5

        Next i

        Implied_Bin_Option_Price = Output_array

    Else

        Implied_Bin_Option_Price = Output_Option_Price  ' default output

    End If

End Function  ' Implied_Bin_Option_Price(...)

' ---------------------------------------------------------------------
' Using a cubic spline fit to the discrete cumulative distribution
' approximate the continuous probability density
' Output_Format = 0 is Stock price, Cumulative Distribution, Prob Density
' Output_Format = 1 is Stock price, Prob Density - needed for
' the function SplineIntegrate(...). The integral from - to + infinity
' of the probability density should = 1, this is a useful cross check.
' ---------------------------------------------------------------------

Function Est_Continuous_Prob(XY_range As Range, n, Output_Format)

    Dim i As Integer

    Dim xy As Variant

    Dim Eval_array_T As Variant

    Dim Spline_Data_array As Variant

    xy = XY_range
```

```
ReDim Eval_array(1 To n + 1)

' ---------------------------------------------------------------------
' Create an array of data points for the cubic spline evaluation such that the
' first point = min terminal S_node and the last point = max terminal S_node
' and there are n+1 points in total, the stock prices need to be ascending.
' Note that the variation between min and max for Eval_array is quadratic,
' this is based on observed variations in terminal stock prices for the
' IBT trees when viewed with uniform spacing on the x axis
' ---------------------------------------------------------------------

Eval_array(1) = xy(1, 1)  ' lowest terminal stock price from the tree (must be exact)

Eval_array(n + 1) = xy(UBound(xy, 1), 1) ' highest terminal stock price from the tree (must be exact)

For i = 2 To n  ' quadratic variation between min and max terminal stock price

    Eval_array(i) = Eval_array(1) + (Eval_array(n + 1) - Eval_array(1)) / (n ^ 2) * (i - 1) ^ 2

Next i

Eval_array_T = Application.Transpose(Eval_array)  ' i.e. make Eval_array a vector, required for SplineData

Spline_Data_array = SplineData(XY_range, Eval_array_T, True)  ' see function in separate module Cubic_Spline

If Output_Format = 0 Then

    ReDim Output_array(1 To n + 1, 1 To 3)

ElseIf Output_Format = 1 Then

    ReDim Output_array(1 To n + 1, 1 To 2)

End If

' ---------------------------------------------------------------------
' Control the format of the output from this function
' ---------------------------------------------------------------------

For i = 1 To n + 1  ' from the lowest terminal stock price to the highest

    Output_array(i, 1) = Eval_array(i)  ' common for both output formats, terminal stock price

    If Output_Format = 0 Then  ' populate all three columns, i.e. Stock price, Cumulative Distribution, Prob Density

        Output_array(i, 2) = Spline_Data_array(i, 0)  ' fitted cumulative distribution

        If Spline_Data_array(i, 1) > 0 Then

            Output_array(i, 3) = Spline_Data_array(i, 1)  ' estimated probability density, 1st derivative of cumulative distribution

        Else

            If i = 1 Then
```

```
                    Output_array(i, 3) = 0  ' estimated probability density should not be less than 0
                Else
                    Output_array(i, 3) = Output_array(i - 1, 3)  ' estimated probability density should not be less than 0
                End If
            End If
        ElseIf Output_Format = 1 Then  ' only give Stock price and Prob Density
            If Spline_Data_array(i, 1) > 0 Then
                Output_array(i, 2) = Spline_Data_array(i, 1)  ' estimated probability density, 1st derivative of cumulative distribution
            Else
                If i = 1 Then
                    Output_array(i, 2) = 0  ' estimated probability density should not be less than 0
                Else
                    Output_array(i, 2) = Output_array(i - 1, 2)  ' estimated probability density should not be less than 0
                End If
            End If
        End If
    Next i
    Est_Continuous_Prob = Output_array
End Function  ' Est_Continuous_Prob(...)

' ------------------------------------------------------------------
' Black and Scholes Call option pricing implementation with continuous
' dividend rate qd (closed form)
' ------------------------------------------------------------------

Function CBS(S, K, T, sigma, rf, qd)
    Dim d1, d2
    Dim Nd1, Nd2
    d1 = (Log(S / K) + (rf - qd + 0.5 * sigma ^ 2) * T) / (sigma * T ^ 0.5)
    d2 = d1 - (sigma * T ^ 0.5)
    Nd1 = Application.NormSDist(d1)
    Nd2 = Application.NormSDist(d2)
```

```
    CBS = S * Exp(-qd * T) * Nd1 - K * Exp(-rf * T) * Nd2

End Function  ' CBS(...)

' ------------------------------------------------------------------
' Black and Scholes Put option pricing implementation with continuous
' dividend rate qd (closed form)
' ------------------------------------------------------------------

Function PBS(S, K, T, sigma, rf, qd)

    Dim d1, d2

    Dim Nd1, Nd2

    d1 = (Log(S / K) + (rf - qd + 0.5 * sigma ^ 2) * T) / (sigma * T ^ 0.5)

    d2 = d1 - (sigma * T ^ 0.5)

    Nd1 = Application.NormSDist(-d1)  ' note -ve d1

    Nd2 = Application.NormSDist(-d2)  ' note -ve d2

    PBS = K * Exp(-rf * T) * Nd2 - S * Exp(-qd * T) * Nd1

End Function  ' PBS(...)

' ------------------------------------------------------------------
' Function to calculate the Black and Scholes volatility implied
' by a given Call or Put Option_Price
' Ref: XLS examples from McCombs School of Business,
' The University of Texas at Austin.
' Call_or_Put = 0 is Call Option
' Call_or_Put = 1 is Put Option
' ------------------------------------------------------------------

Function Imp_Vol(S, K, T, rf, qd, Option_Price, Tol, Call_or_Put)

    Dim High_sigma, Low_sigma

    High_sigma = 2 ' i.e. 200%

    Low_sigma = 0

    If Call_or_Put = 0 Then  ' Call Option

        Do While (High_sigma - Low_sigma) > Tol

            If CBS(S, K, T, (High_sigma + Low_sigma) / 2, rf, qd) > Option_Price Then

                High_sigma = (High_sigma + Low_sigma) / 2

            Else

                Low_sigma = (High_sigma + Low_sigma) / 2
```

```
            End If

        Loop

    Else  ' Put Option

        Do While (High_sigma - Low_sigma) > Tol

            If PBS(S, K, T, (High_sigma + Low_sigma) / 2, rf, qd) > Option_Price Then

                High_sigma = (High_sigma + Low_sigma) / 2

            Else

                Low_sigma = (High_sigma + Low_sigma) / 2

            End If

        Loop

    End If

    Imp_Vol = (High_sigma + Low_sigma) / 2

End Function  ' Imp_Vol(...)

' ------------------------------------------------------------------
' CRR Binomial Call and Put option pricing implementation with
' continuous dividend rate qd
' Call_or_Put = 0 is Call Option
' Call_or_Put = 1 is Put Option
' Note that j are the discrete time steps from 0 to T (columns) and i are the vertical nodes (rows) at each time step e.g. S_node(i, j).
' See sheet Sample_IBT_Output for meaning of the i and j array indices.
' S_node(0, 0) is the root of the tree at time 0.
' S_node(0, n) is the highest stock price at expiry time T from a series of up movements only.
' S_node(n, n) is the lowest stock price at expiry time T from a series of down movements only.
' ------------------------------------------------------------------

Function CRR_Bin_Option_Price(S, K, T, sigma, rf, qd, n, Call_or_Put, Output_Format)

    ' ------------------------------------------------------------------
    ' Define the variables
    ' ------------------------------------------------------------------

    Dim dt              ' time step

    Dim i As Integer    ' counter / index for binomial tree

    Dim j As Integer    ' counter / index for binomial tree

    Dim u, d            ' factors by which stock prices move up and down

    Dim q               ' risk neutral probability

    Dim disc            ' discount factor
```

```
Dim Output_Option_Price  ' call or put tree price

Dim Cum_sum         ' cumulative distribution sum

' ---------------------------------------------------------------------
' Create required matrices / arrays
' ---------------------------------------------------------------------

ReDim S_node(0 To n, 0 To n)     ' array of stock prices

ReDim P_node(0 To n, 0 To n)     ' array of node probabilities

ReDim Lamda_node(0 To n, 0 To n) ' array of node state prices

ReDim call_val(0 To n)           ' array of terminal call values

ReDim Output_array(0 To n, 0 To n)   ' temp array for formatting output data

' ---------------------------------------------------------------------
' CRR Parameters (note continuous compounding)
' ---------------------------------------------------------------------

dt = T / n

u = Exp(sigma * dt ^ 0.5)  ' up movement

d = 1 / u  ' down movement

q = (Exp((rf - qd) * dt) - d) / (u - d) ' transition probability at every node

disc = Exp(-rf * dt)  ' discount factor


' Note that for the probability to lie between 0 and 1 : sigma /(rf - qd) > dt ^ 0.5

If (sigma / (rf - qd)) <= (dt ^ 0.5) Then

    MsgBox ("CRR Parameter Error :  sigma /(rf - qd) <= dt ^ 0.5")

End If

' ---------------------------------------------------------------------
' Calculate node stock prices, probabilities, AD state prices and terminal call values
' Note that the index values (i, j) were chosen such that the
' matrices display in a sensible manner in MS XLS
' Same (i,j) convention as Function Implied_Bin_Option_Price(...)
' ---------------------------------------------------------------------

Dim C_sum, P_sum

C_sum = 0

For j = 0 To n  ' time steps to maturity

    For i = 0 To j  ' vertical nodes
```

```
        S_node(i, j) = S * (d ^ i) * u ^ (j - i)  ' node stock prices
        P_node(i, j) = Application.Combin(j, i) * ((1 - q) ^ i) * q ^ (j - i)  ' node probabilities
        Lamda_node(i, j) = P_node(i, j) * disc ^ j  ' node AD prices or state prices
        If j = n Then  ' determine the terminal call payoffs
            call_val(i) = Application.Max(S_node(i, j) - K, 0)  ' terminal call payoffs
            C_sum = C_sum + call_val(i) * Lamda_node(i, j)  ' sum of product of terminal payoffs and terminal state prices
        End If
    Next i
Next j
If Call_or_Put = 0 Then  ' Call Option
    Output_Option_Price = C_sum
Else  ' Put Option
    Output_Option_Price = C_sum + K * Exp(-rf * T) - S * Exp(-qd * T)  ' from the put call parity relationship
End If
' ---------------------------------------------------------------------
' Control the format of the output from this function
' ---------------------------------------------------------------------
If Output_Format = 1 Then
    CRR_Bin_Option_Price = Output_Option_Price  ' the call or put price
ElseIf Output_Format = 2 Then
    CRR_Bin_Option_Price = S_node  ' Stock prices at each node
ElseIf Output_Format = 3 Then
    CRR_Bin_Option_Price = P_node  ' Probabilities at each node
ElseIf Output_Format = 4 Then
    CRR_Bin_Option_Price = Lamda_node  ' AD state prices at each node
ElseIf Output_Format = 5 Then  ' Terminal stock prices and terminal probabilities for plotting distributions
    Cum_sum = 1
    ReDim Output_array(0 To n, 0 To 2)
    For i = 0 To n  ' terminal vertical nodes (reverse order required for Function Est_Continuous_Prob(...))
```

```
            Output_array(n - i, 0) = S_node(i, n)

            Output_array(n - i, 1) = Cum_sum

            Output_array(n - i, 2) = P_node(i, n)  ' Note that these are discrete probabilities

            Cum_sum = Cum_sum - P_node(i, n)

        Next i

        CRR_Bin_Option_Price = Output_array

    Else

        CRR_Bin_Option_Price = Output_Option_Price  ' default output

    End If


End Function  ' CRR_Bin_Option_Price(...)

' --------------------------------------------------------------------
' Function to return the volatility according to the equation
' which defines the implied volatility surface in terms of the strike, K,
' and the time to maturity, T, as determined from market price data.
' --------------------------------------------------------------------

Function Smile_Vol(S, K, T)

    ' This function can be changed to also include effects from T

    ' Smile_Vol = 0.15 - 0.0005 * K  ' i.e. assuming S = 100

    Smile_Vol = 0.1 + (S - K) / S / 10 * 0.5  ' as per XploRe Handbook, page 58 IBTdk

    ' Smile_Vol = 1 / 120000 * K ^ 2 - 1 / 400 * K + 0.24

    ' Smile_Vol = 1 / 60000 * K ^ 2 - 5 / 1200 * K + 0.31  ' this causes problems with the 100 step BC example

    ' Smile_Vol = 0.1  ' i.e. no smile or skew, just constant volatility

End Function  ' Smile_Vol(...)


' --------------------------------------------------------------------
' Function to return the log normal density with constant volatility
' Applied Quantitative Finance PDF Book, Equation (7.1)
' --------------------------------------------------------------------

Function Log_Normal_Density(S0, ST, T, sigma, rf, qd)

    Log_Normal_Density = 1 / (ST * (2 * Application.Pi() * sigma ^ 2 * T) ^ 0.5) * _
                         Exp((Log(ST / S0) - ((rf - qd) - 0.5 * sigma ^ 2) * T) ^ 2 / (-2 * T * sigma ^ 2))

End Function  ' Log_Normal_Density(...)
```

6 Excel Worksheets Output

The attached pages contain the output of the nine Excel worksheets contained in the file:

Rebel_NM1P_Implied_Binomial_Trees.xls

1 Call_Option_Implied_Vol

2 Sample_IBT_Output

3 DK_BC_Trees

4 XploRe_Results

5 DK_BC_Terminal_Prob_Densities

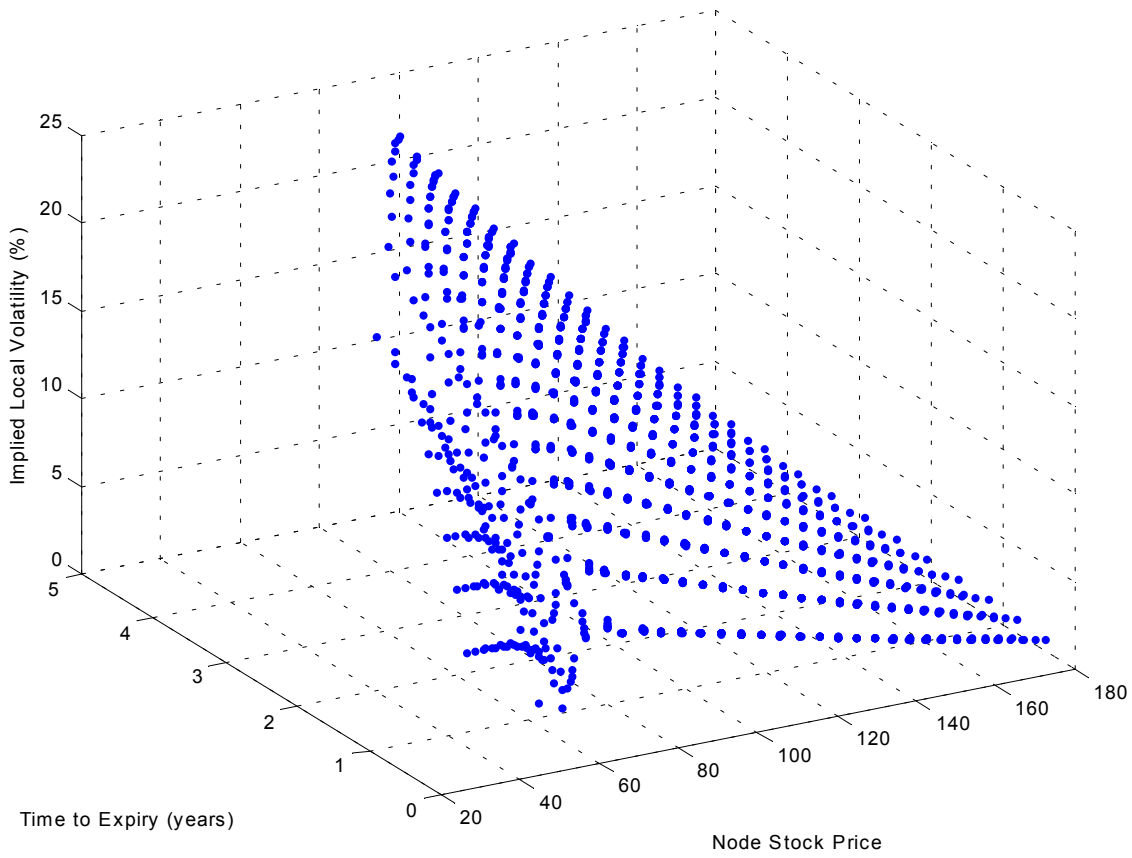
6 AQF_IBTlocsigma

7 Implied_Local_Volatility

8 CRR_Prob_Density

9 BC_100_Step_Example

The first two pages relate to the data in the sheet Implied_Local_Volatility. They simply show the data in a three-dimensional format which is not readily achieved in Excel.



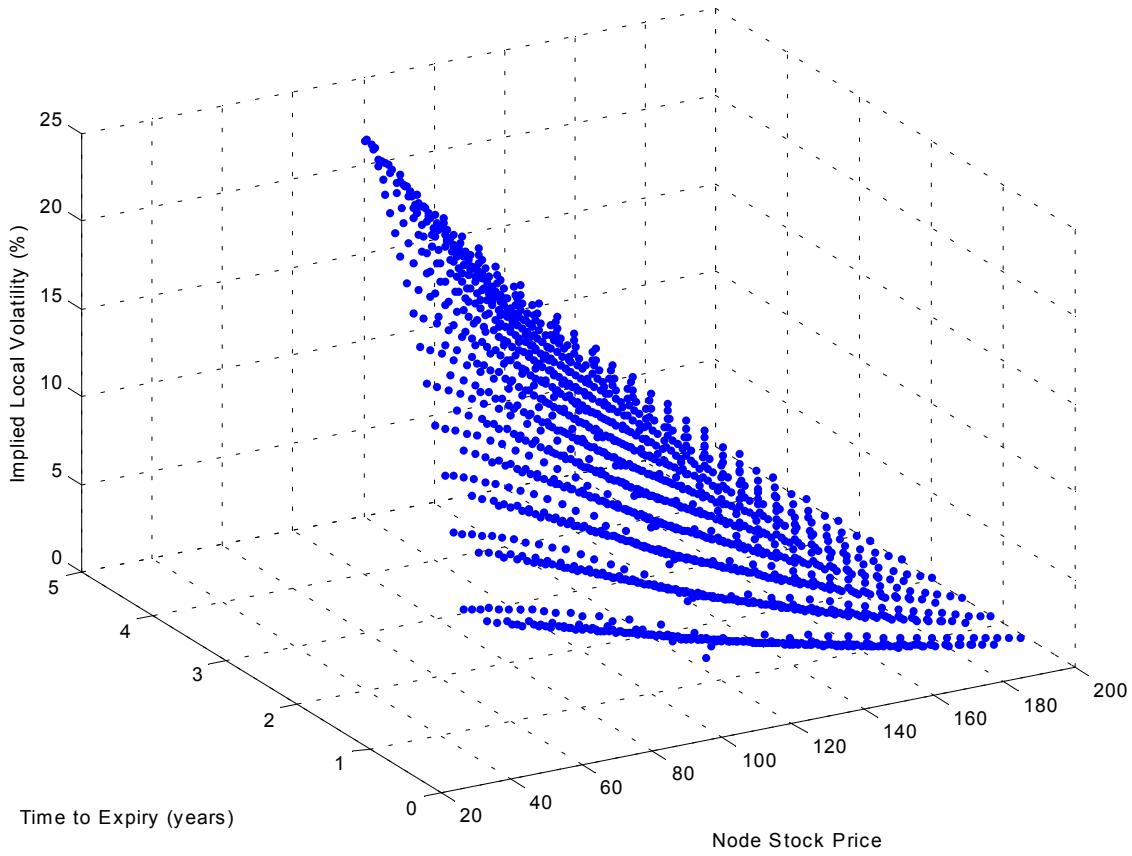
Implied_Bin_Option_Price(S, K, T, rf, qd, n, DK_or_CB, Call_or_Put, Output_Format)

Calculated local implied volatility data corresponding to the Applied Quantitative Finance Figure 7.3 DK, see also Excel sheet Implied_Local_Volatility for exact numerical results.

Output from : Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 20, 0, 0, 9) (1540 rows x 3 columns)

Smile_Vol = 0.1 + (S - K) / S / 10 * 0.5 as per XploRe Handbook, page 58 IBTdk.

Sheet: Implied_Local_Volatility



Implied_Bin_Option_Price(S, K, T, rf, qd, n, DK_or_CB, Call_or_Put, Output_Format)

Calculated local implied volatility data corresponding to the Applied Quantitative Finance Figure 7.5 BC, see also Excel sheet Implied_Local_Volatility for exact numerical results.

Output from : Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 20, 1, 0, 9) (1540 rows x 3 columns)

Smile_Vol = 0.1 + (S - K) / S / 10 * 0.5 as per XploRe Handbook, page 58 IBTdk.

Sheet: Implied_Local_Volatility

THE OUTPUT OF 9 EXCEL WORKSHEETS FOLLOW BELOW

Rebel_NM1P_Implied_Binomial_Trees.xls Gerhard Rebel

Implied Binomial Tree (IBT) Programme - according to the Derman and Kani (DK) and Barle and Cakici (BC) methods

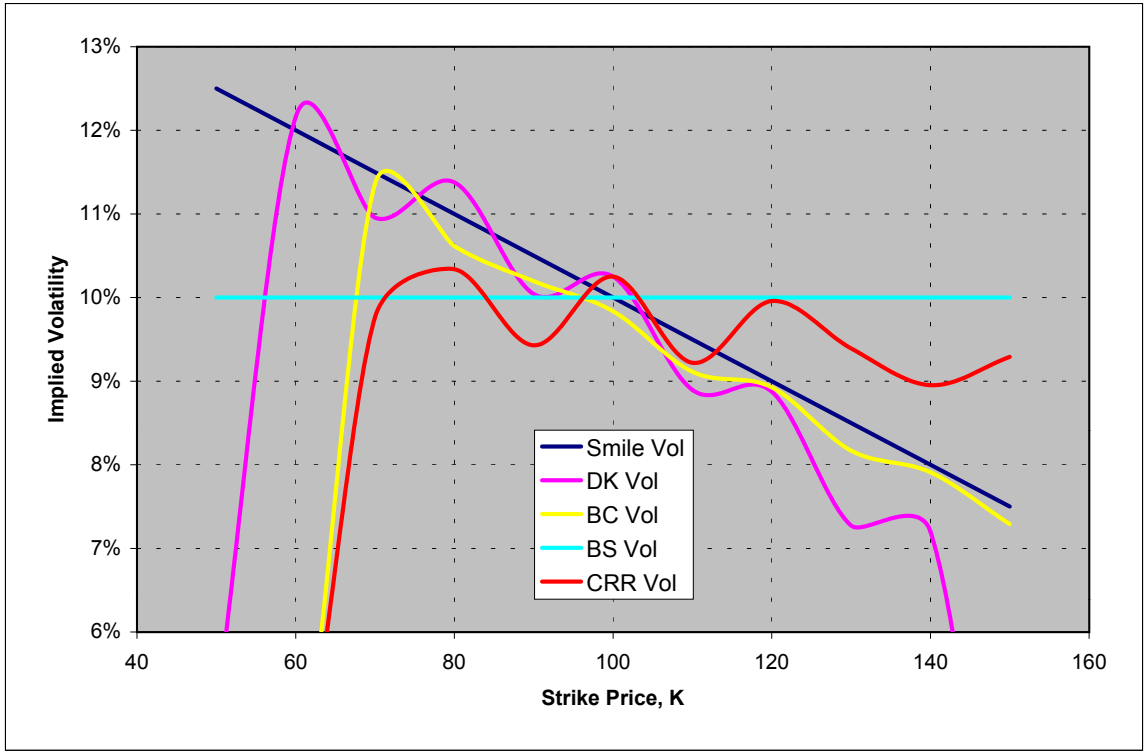
The purpose of these calculations is to show how the DK and BC implied binomial trees are capable of producing European call option prices that are consistent with implied volatility smiles observed in the market

European Call Pricing by BS Closed Form, CRR Binomial Tree, DK IBT and BC IBT

S	100.0	stock price
K	100.0	option strike price
sigma	10.0%	constant for CRR and BS
T	5.0	years
rf	3.0%	risk free per year
qd	0.0%	dividend per year
n	5	number of steps in binomial tree
Calculated BS call price	17.0317	closed form solution
Calculated implied volatility BS	10.00%	should = sigma
CRR binomial tree call price	17.1975	
Calculated implied volatility CRR	10.25%	
DK Implied binomial tree call price	17.1975	
DK Calculated implied volatility IBT	10.25%	
BC Implied binomial tree call price	16.9246	
BC Calculated implied volatility IBT	9.84%	
Inputted market smile volatility	10.00%	

Defined manually in the function : Smile_Vol(S,K,T)

K	Smile Vol	DK Vol	BC Vol	BS Vol	CRR Vol
	10.00%	**10.25%**	**9.84%**	**10.00%**	**10.25%**
50	12.50%	5.06%	4.99%	10.00%	5.00%
60	12.00%	12.16%	3.94%	10.00%	3.97%
70	11.50%	10.96%	11.36%	10.00%	9.76%
80	11.00%	11.38%	10.61%	10.00%	10.34%
90	10.50%	10.04%	10.20%	10.00%	9.43%
100	10.00%	10.25%	9.84%	10.00%	10.25%
110	9.50%	8.90%	9.11%	10.00%	9.22%
120	9.00%	8.87%	8.93%	10.00%	9.96%
130	8.50%	7.27%	8.17%	10.00%	9.39%
140	8.00%	7.20%	7.91%	10.00%	8.95%
150	7.50%	1.47%	7.29%	10.00%	9.29%



CTRL-ALT-F9 = recalculate whole workbook

There are 9 worksheets in this workbook

Rebel_NM1P_Implied_Binomial_Trees.xls Gerhard Rebel

Implied Binomial Tree (IBT) Programme - according to the Derman and Kani (DK) and Barle and Cakici (BC) methods

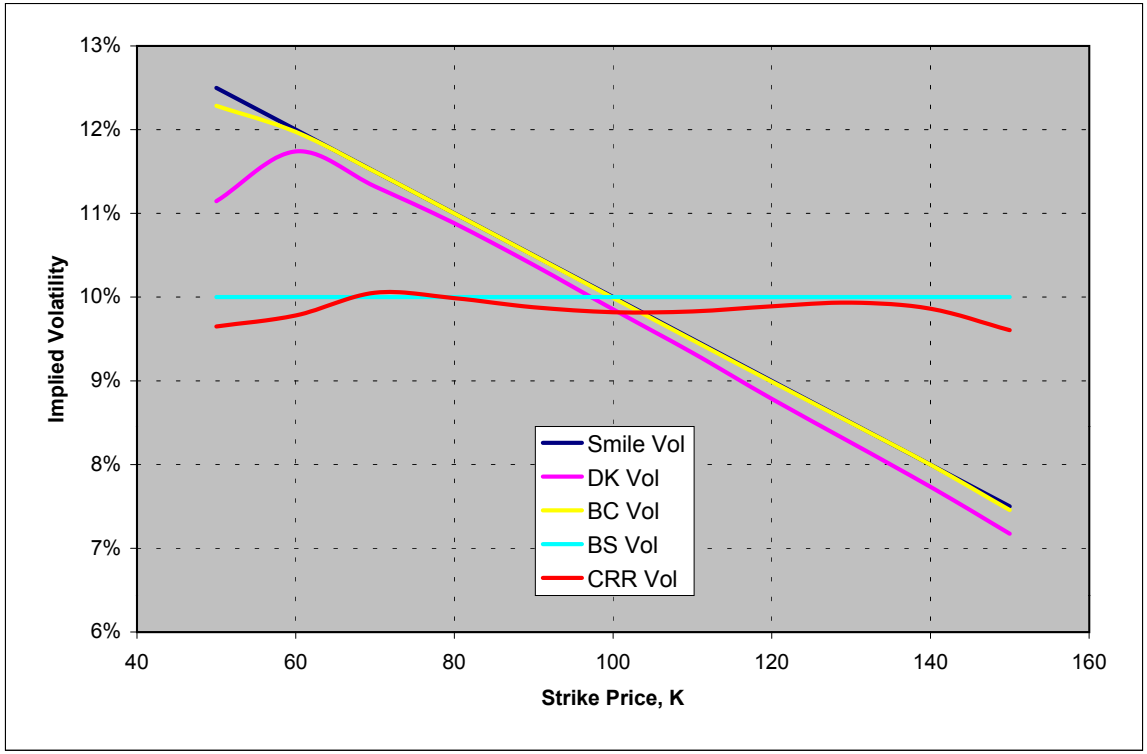
The purpose of these calculations is to show how the DK and BC implied binomial trees are capable of producing European call option prices that are consistent with implied volatility smiles observed in the market

European Call Pricing by BS Closed Form, CRR Binomial Tree, DK IBT and BC IBT

S	100.0	stock price
K	100.0	option strike price
sigma	10.0%	constant for CRR and BS
T	5.0	years
rf	3.0%	risk free per year
qd	0.0%	dividend per year
n	20	number of steps in binomial tree
Calculated BS call price	17.0317	closed form solution
Calculated implied volatility BS	10.00%	should = sigma
CRR binomial tree call price	16.9131	
Calculated implied volatility CRR	9.82%	
DK Implied binomial tree call price	16.9361	
DK Calculated implied volatility IBT	9.85%	
BC Implied binomial tree call price	17.0263	
BC Calculated implied volatility IBT	9.99%	
Inputted market smile volatility	10.00%	

Defined manually in the function : Smile_Vol(S,K,T)

K	Smile Vol	DK Vol	BC Vol	BS Vol	CRR Vol
	10.00%	**9.85%**	**9.99%**	**10.00%**	**9.82%**
50	12.50%	11.15%	12.28%	10.00%	9.65%
60	12.00%	11.74%	11.97%	10.00%	9.78%
70	11.50%	11.32%	11.50%	10.00%	10.05%
80	11.00%	10.88%	11.00%	10.00%	9.99%
90	10.50%	10.38%	10.50%	10.00%	9.88%
100	10.00%	9.85%	9.99%	10.00%	9.82%
110	9.50%	9.34%	9.49%	10.00%	9.83%
120	9.00%	8.79%	8.99%	10.00%	9.89%
130	8.50%	8.26%	8.50%	10.00%	9.93%
140	8.00%	7.73%	8.00%	10.00%	9.86%
150	7.50%	7.17%	7.46%	10.00%	9.61%



CTRL-ALT-F9 = recalculate whole workbook

There are 9 worksheets in this workbook

Rebel_NM1P_Implied_Binomial_Trees.xls Gerhard Rebel

Implied Binomial Tree (IBT) Programme - according to the Derman and Kani (DK) and Barle and Cakici (BC) methods

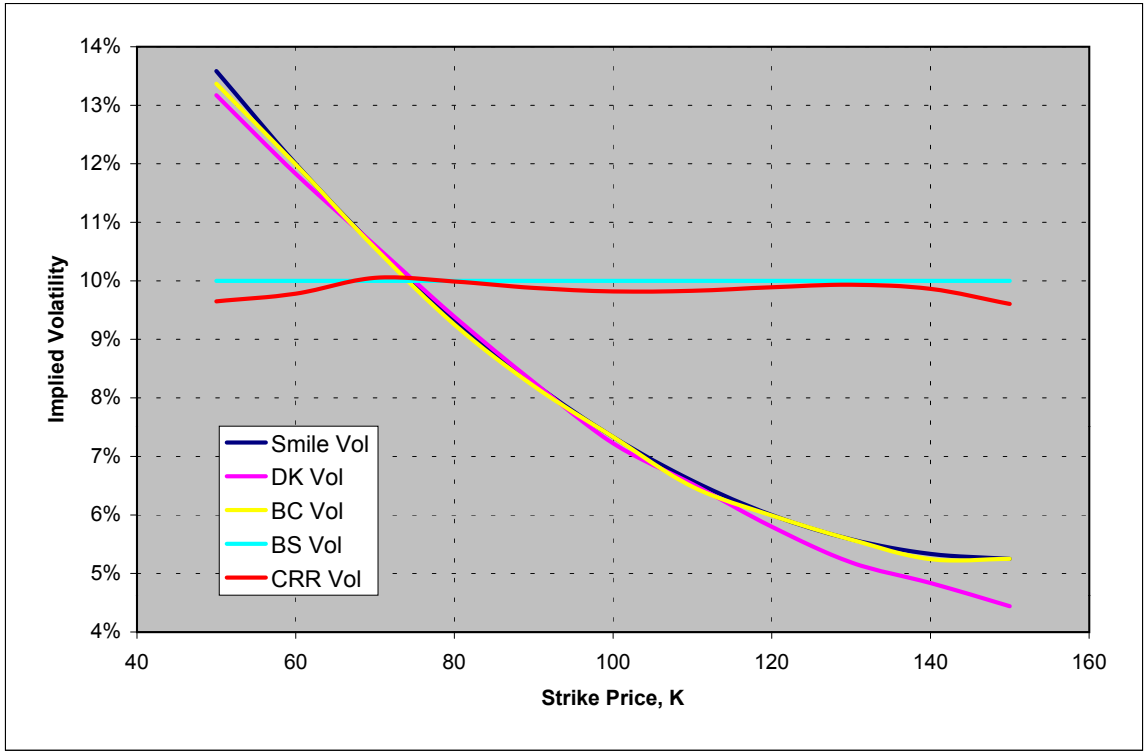
The purpose of these calculations is to show how the DK and BC implied binomial trees are capable of producing European call option prices that are consistent with implied volatility smiles observed in the market

European Call Pricing by BS Closed Form, CRR Binomial Tree, DK IBT and BC IBT

S	100.0	stock price
K	100.0	option strike price
sigma	10.0%	constant for CRR and BS
T	5.0	years
rf	3.0%	risk free per year
qd	0.0%	dividend per year
n	20	number of steps in binomial tree
Calculated BS call price	17.0317	closed form solution
Calculated implied volatility BS	10.00%	should = sigma
CRR binomial tree call price	16.9131	
Calculated implied volatility CRR	9.82%	
DK Implied binomial tree call price	15.3543	
DK Calculated implied volatility IBT	7.22%	
BC Implied binomial tree call price	15.4132	
BC Calculated implied volatility IBT	7.33%	
Inputted market smile volatility	7.33%	

Defined manually in the function : Smile_Vol(S,K,T)

K	Smile Vol	DK Vol	BC Vol	BS Vol	CRR Vol
	7.33%	**7.22%**	**7.33%**	**10.00%**	**9.82%**
50	13.58%	13.17%	13.37%	10.00%	9.65%
60	12.00%	11.83%	11.99%	10.00%	9.78%
70	10.58%	10.61%	10.56%	10.00%	10.05%
80	9.33%	9.39%	9.25%	10.00%	9.99%
90	8.25%	8.27%	8.20%	10.00%	9.88%
100	7.33%	7.22%	7.33%	10.00%	9.82%
110	6.58%	6.53%	6.48%	10.00%	9.83%
120	6.00%	5.80%	5.99%	10.00%	9.89%
130	5.58%	5.19%	5.58%	10.00%	9.93%
140	5.33%	4.84%	5.24%	10.00%	9.86%
150	5.25%	4.44%	5.25%	10.00%	9.61%



CTRL-ALT-F9 = recalculate whole workbook

There are 9 worksheets in this workbook

Sheet: Call_Option_Implied_Vol

Rebel_NM1P_Implied_Binomial_Trees.xls Gerhard Rebel

These calculations demonstrate the possible output formats from the main project function Implied_Bin_Option_Price(…) using 5 steps on a call option (i.e. n = 5):

Function used : Implied_Bin_Option_Price(S, K, T, rf, qd, n, DK_or_CB, Call_or_Put, Output_Format)

Function used : CRR_Bin_Option_Price(S, K, T, sigma, rf, qd, n, Call_or_Put, Output_Format)

S	100	stock price
K	100	option strike price
T	5	years
sigma	10.0%	constant for CRR
rf	3.0%	risk free per year
qd	0.0%	dividend per year

DK

Call option price (Output_Format = 1)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 0, 0, 1)

17.1975

Stock prices at each node (Output_Format = 2)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 0, 0, 2)

	j=0	j=1	j=2	j=3	j=4	j=5
i=0	100.00	110.52	120.30	130.08	139.17	147.74
i=1	0.00	90.48	100.00	110.57	120.37	130.10
i=2	0.00	0.00	79.29	90.44	100.00	110.63
i=3	0.00	0.00	0.00	71.33	79.28	90.39
i=4	0.00	0.00	0.00	0.00	58.75	71.20
i=5	0.00	0.00	0.00	0.00	0.00	54.35

Transition probabilities at each node (Output_Format = 3)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 0, 0, 3)

i=0	0.627	0.684	0.687	0.727	0.754	
i=1	0.000	0.674	0.626	0.684	0.688	
i=2	0.000	0.000	0.543	0.671	0.625	
i=3	0.000	0.000	0.000	0.719	0.547	
i=4	0.000	0.000	0.000	0.000	0.367	
i=5						

Arrow-Debreu state prices at each node (Output_Format = 4)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 0, 0, 4)

i=0	1.000	0.609	0.404	0.269	0.190	0.139
i=1	0.000	0.362	0.423	0.380	0.324	0.262
i=2	0.000	0.000	0.115	0.214	0.256	0.253
i=3	0.000	0.000	0.000	0.051	0.104	0.148
i=4	0.000	0.000	0.000	0.000	0.014	0.051
i=5	0.000	0.000	0.000	0.000	0.000	0.009

Probabilities of reaching each node (Output_Format = 5)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 0, 0, 5)

i=0	1.000	0.627	0.429	0.294	0.214	0.161
i=1	0.000	0.373	0.449	0.416	0.365	0.304
i=2	0.000	0.000	0.122	0.234	0.288	0.294
i=3	0.000	0.000	0.000	0.056	0.117	0.172
i=4	0.000	0.000	0.000	0.000	0.016	0.059
i=5	0.000	0.000	0.000	0.000	0.000	0.010
Sum	1.000	1.000	1.000	1.000	1.000	1.000

Terminal stock prices, cumulative probabilities and terminal node probabilities (Output_Format = 6)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 0, 0, 6)

	j=5		
i=5	54.35	0.0099	0.0099
i=4	71.20	0.0687	0.0588
i=3	90.39	0.2407	0.1720
i=2	110.63	0.5347	0.2940
i=1	130.10	0.8385	0.3038
i=0	147.74	1.0000	0.1615

Note reverse order

Forward prices at each node (Output_Format = 7)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 0, 0, 7)

i=0	0.00	103.05	113.88	123.96	134.04	143.41
i=1	0.00	0.00	93.24	103.05	113.94	124.03
i=2	0.00	0.00	0.00	81.70	93.19	103.05
i=3	0.00	0.00	0.00	0.00	73.51	81.69
i=4	0.00	0.00	0.00	0.00	0.00	60.54
i=5	0.00	0.00	0.00	0.00	0.00	0.00

Implied local volatilities at each node for the time to expiry = dt (Output_Format = 8)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 0, 0, 8)

i=0	9.67%	8.59%	7.53%	6.47%	5.47%	
i=1	0.00%	10.88%	9.73%	8.62%	7.51%	
i=2	0.00%	0.00%	11.82%	10.91%	9.78%	
i=3	0.00%	0.00%	0.00%	13.47%	11.88%	
i=4	0.00%	0.00%	0.00%	0.00%	13.02%	
i=5						

Stock prices, times to expiry and implied local volatilities (Output_Format = 9)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 0, 0, 9)

100.00	1.00	9.67%
100.00	2.00	13.64%
100.00	3.00	16.54%
100.00	4.00	18.99%
100.00	5.00	21.01%
110.52	1.00	8.59%
110.52	2.00	12.00%
110.52	3.00	14.54%
110.52	4.00	16.56%
90.48	1.00	10.88%
90.48	2.00	15.17%
90.48	3.00	18.57%
90.48	4.00	21.21%
120.30	1.00	7.53%
120.30	2.00	10.48%
120.30	3.00	12.61%
100.00	1.00	9.73%
100.00	2.00	13.70%
100.00	3.00	16.61%
79.29	1.00	11.82%
79.29	2.00	17.05%
79.29	3.00	20.58%
130.08	1.00	6.47%
130.08	2.00	8.93%
110.57	1.00	8.62%
110.57	2.00	12.02%
90.44	1.00	10.91%
90.44	2.00	15.23%
71.33	1.00	13.47%
71.33	2.00	18.24%
139.17	1.00	5.47%
120.37	1.00	7.51%
100.00	1.00	9.78%
79.28	1.00	11.88%
58.75	1.00	13.02%

BC

Call option price (Output_Format = 1)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 1, 0, 1)

16.9246

Stock prices at each node (Output_Format = 2)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 1, 0, 2)

	j=0	j=1	j=2	j=3	j=4	j=5
i=0	100.00	111.47	127.04	138.72	151.41	162.78
i=1	0.00	95.26	106.18	119.33	132.06	144.62
i=2	0.00	0.00	84.06	100.33	112.75	126.41
i=3	0.00	0.00	0.00	79.53	92.56	106.78
i=4	0.00	0.00	0.00	0.00	71.15	86.22
i=5	0.00	0.00	0.00	0.00	0.00	64.69

Transition probabilities at each node (Output_Format = 3)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 1, 0, 3)

i=0	0.480	0.416	0.597	0.563	0.628	
i=1	0.000	0.637	0.478	0.529	0.531	
i=2	0.000	0.000	0.341	0.536	0.479	
i=3	0.000	0.000	0.000	0.505	0.445	
i=4	0.000	0.000	0.000	0.000	0.400	
i=5						

Arrow-Debreu state prices at each node (Output_Format = 4)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 1, 0, 4)

i=0	1.000	0.466	0.188	0.109	0.060	0.036
i=1	0.000	0.504	0.576	0.341	0.221	0.136
i=2	0.000	0.000	0.177	0.350	0.338	0.258
i=3	0.000	0.000	0.000	0.114	0.213	0.263
i=4	0.000	0.000	0.000	0.000	0.055	0.136
i=5	0.000	0.000	0.000	0.000	0.000	0.032

Probabilities of reaching each node (Output_Format = 5)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 1, 0, 5)

i=0	1.000	0.480	0.200	0.119	0.067	0.042
i=1	0.000	0.520	0.612	0.373	0.250	0.157
i=2	0.000	0.000	0.188	0.383	0.381	0.300
i=3	0.000	0.000	0.000	0.124	0.240	0.306
i=4	0.000	0.000	0.000	0.000	0.062	0.158
i=5	0.000	0.000	0.000	0.000	0.000	0.037
Sum	1.000	1.000	1.000	1.000	1.000	1.000

Terminal stock prices, cumulative probabilities and terminal node probabilities (Output_Format = 6)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 1, 0, 6)

	j=5		
i=5	64.69	0.0369	0.0369
i=4	86.22	0.1949	0.1580
i=3	106.78	0.5007	0.3057
i=2	126.41	0.8003	0.2996
i=1	144.62	0.9578	0.1575
i=0	162.78	1.0000	0.0422

Note reverse order

Forward prices at each node (Output_Format = 7)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 1, 0, 7)

i=0	0.00	103.05	114.86	130.91	142.95	156.02
i=1	0.00	0.00	98.16	109.42	122.96	136.08
i=2	0.00	0.00	0.00	86.62	103.39	116.18
i=3	0.00	0.00	0.00	0.00	81.96	95.38
i=4	0.00	0.00	0.00	0.00	0.00	73.31
i=5	0.00	0.00	0.00	0.00	0.00	0.00

Implied local volatilities at each node for the time to expiry = dt (Output_Format = 8)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 1, 0, 8)

i=0	7.85%	8.84%	7.39%	6.78%	5.72%	
i=1	0.00%	11.23%	8.66%	7.89%	6.72%	
i=2	0.00%	0.00%	11.01%	9.84%	8.43%	
i=3	0.00%	0.00%	0.00%	13.16%	10.63%	
i=4	0.00%	0.00%	0.00%	0.00%	14.08%	
i=5						

Stock prices, times to expiry and implied local volatilities (Output_Format = 9)

= Implied_Bin_Option_Price(100, 100, 5, 0.03, 0, 5, 1, 0, 9)

100.00	1.00	7.85%
100.00	2.00	12.91%
100.00	3.00	15.78%
100.00	4.00	18.50%
100.00	5.00	20.72%
111.47	1.00	8.84%
111.47	2.00	12.07%
111.47	3.00	14.72%
111.47	4.00	16.82%
95.26	1.00	11.23%
95.26	2.00	14.85%
95.26	3.00	18.20%
95.26	4.00	20.86%
127.04	1.00	7.39%
127.04	2.00	10.38%
127.04	3.00	12.45%
106.18	1.00	8.66%
106.18	2.00	12.52%
106.18	3.00	15.31%
84.06	1.00	11.01%
84.06	2.00	16.50%
84.06	3.00	20.34%
138.72	1.00	6.78%
138.72	2.00	9.20%
119.33	1.00	7.89%
119.33	2.00	10.98%
100.33	1.00	9.84%
100.33	2.00	13.76%
79.53	1.00	13.16%
79.53	2.00	18.28%
151.41	1.00	5.72%
132.06	1.00	6.72%
112.75	1.00	8.43%
92.56	1.00	10.63%
71.15	1.00	14.08%

CRR

Call option price (Output_Format = 1)

= CRR_Bin_Option_Price(100, 100, 5, 0.1, 0.03, 0, 5, 0, 1)

17.1975

Stock prices at each node (Output_Format = 2)

= CRR_Bin_Option_Price(100, 100, 5, 0.1, 0.03, 0, 5, 0, 2)

	j=0	j=1	j=2	j=3	j=4	j=5
i=0	100.00	110.52	122.14	134.99	149.18	164.87
i=1	0.00	90.48	100.00	110.52	122.14	134.99
i=2	0.00	0.00	81.87	90.48	100.00	110.52
i=3	0.00	0.00	0.00	74.08	81.87	90.48
i=4	0.00	0.00	0.00	0.00	67.03	74.08
i=5	0.00	0.00	0.00	0.00	0.00	60.65

Probabilities at each node (Output_Format = 3)

= CRR_Bin_Option_Price(100, 100, 5, 0.1, 0.03, 0, 5, 0, 3)

i=0	1.000	0.627	0.393	0.247	0.155	0.097
i=1	0.000	0.373	0.468	0.440	0.368	0.288
i=2	0.000	0.000	0.139	0.262	0.328	0.343
i=3	0.000	0.000	0.000	0.052	0.130	0.204
i=4	0.000	0.000	0.000	0.000	0.019	0.061
i=5	0.000	0.000	0.000	0.000	0.000	0.007
Sum	1.000	1.000	1.000	1.000	1.000	1.000

Terminal stock prices, cumulative probabilities and terminal node probabilities (Output_Format = 5)

= CRR_Bin_Option_Price(100, 100, 5, 0.1, 0.03, 0, 5, 0, 5)

	j=5		
i=5	60.65	0.0072	0.0072
i=4	74.08	0.0679	0.0607
i=3	90.48	0.2719	0.2040
i=2	110.52	0.6148	0.3429
i=1	134.99	0.9031	0.2883
i=0	164.87	1.0000	0.0969

Note reverse order

Sheet: Sample_IBT_Output

Gerhard Rebel

XploRe Quantlet Java Client Version 1.4

See pages 58 and 59 of XploRe Handbook for Derman and Kani's method implementation:

```
library("finance")
proc(sigma)=volafunc(K, S, time)
sigma=0.1+(S-K)/S/10*0.5
endp
r=0.03
S=100
lev=5
expiration=5
ibtree=IBTdk(S, r, lev, expiration, "volafunc")
ptree=IBTresort(ibtree.Tree)
prob=IBTresort(ibtree.prob)
lb=IBTresort(ibtree.lb)
IBTnicemat("stock price", ptree, 2)
IBTnicemat("transition probability", prob, 3)
IBTnicemat("Arrow-Debreu price", lb, 3)
```

DK stock price

```
                                        147.83
                                139.22
                        130.12          130.29
                120.31          120.43
          110.52          110.65          110.91
  100.00          100.00          100.00
           90.48           90.37           90.16
                    79.29           79.45
                            71.27           70.89
                                    59.48
                                            54.25
```

DK transition probability

```
                            0.751
                    0.727
                0.684           0.680
          0.684           0.686
  0.627           0.625           0.621
          0.673           0.666
                  0.546           0.569
                          0.699
                                  0.423
```

DK Arrow-Debreu price

```
                                        0.138
                                0.189
                        0.268           0.260
                0.404           0.324
          0.608           0.380           0.254
  1.000           0.423           0.255
          0.362           0.215           0.151
                  0.115           0.104
                          0.051           0.050
                                  0.015
                                          0.008
```

Refer to Sheet: Sample_IBT_Output

Gerhard Rebel

XploRe Quantlet Java Client Version 1.4

See pages 52 and 53 of XploRe Handbook for Barle and Cakici's method implementation:

```
library("finance")
proc(sigma)=volafunc(K, S, time)
sigma=0.1+(S-K)/S/10*0.5
endp
r=0.03
S=100
lev=5
expiration=5
ibtree=IBTbc(S, r, lev, expiration, "volafunc")
ptree=IBTresort(ibtree.Tree)
prob=IBTresort(ibtree.prob)
lb=IBTresort(ibtree.lb)
IBTnicemat("stock price", ptree, 2)
IBTnicemat("transition probability", prob, 3)
IBTnicemat("Arrow-Debreu price", lb, 3)
```

BC stock price

				162.77
			151.42	
		138.72		144.62
	127.05		132.06	
111.47		119.33		126.41
100.00		106.18		112.75
95.26		100.33		106.78
	84.06		92.56	
		79.53		86.22
			71.14	
				64.74

BC transition probability

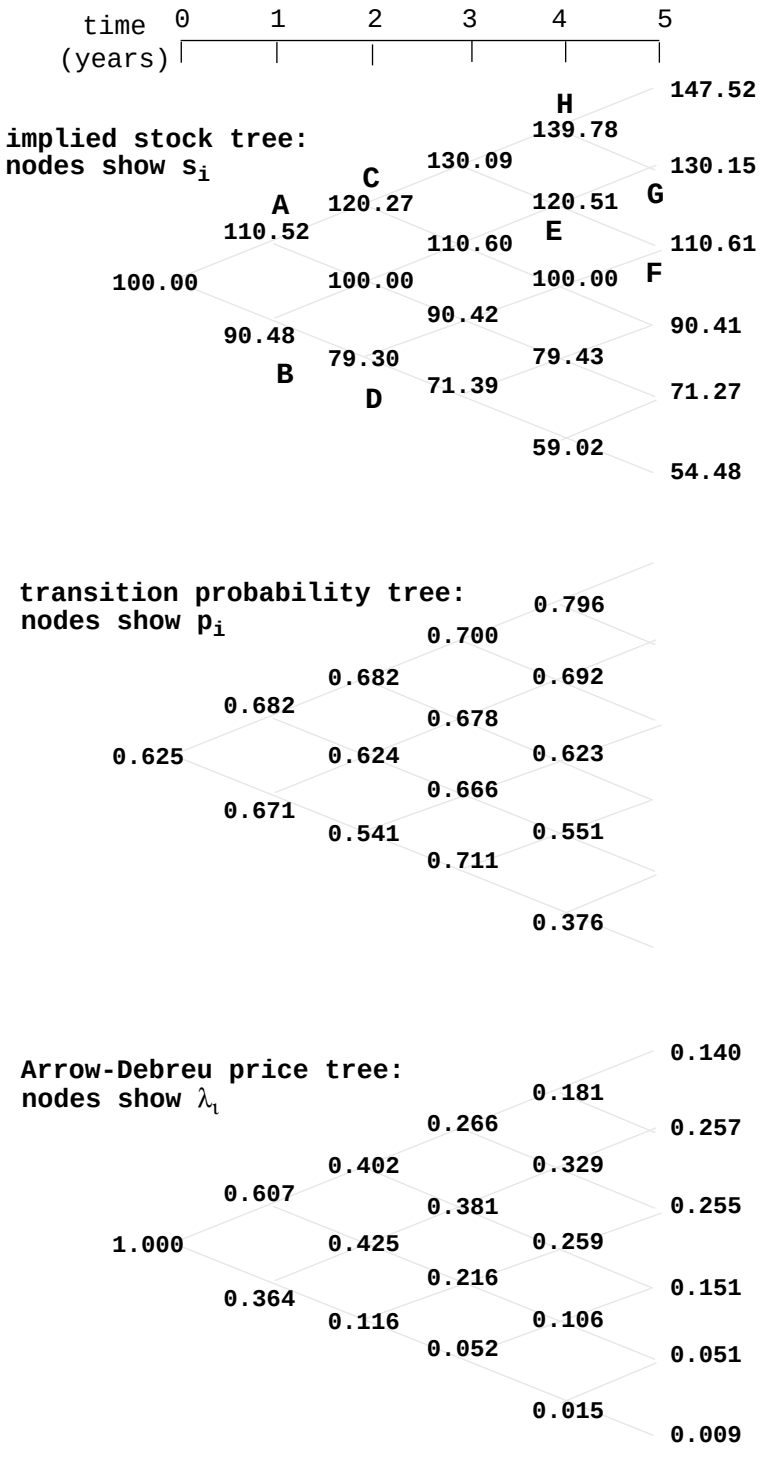
			0.629
		0.563	
	0.597		0.531
0.416		0.529	
0.480		0.478	0.479
0.637		0.536	
	0.341		0.445
		0.505	
			0.399

BC Arrow-Debreu price

				0.036
			0.060	
		0.109		0.136
	0.188		0.221	
0.466		0.341		0.258
1.000		0.576	0.338	
0.504		0.350		0.263
	0.177		0.213	
		0.114		0.136
			0.055	
				0.032

Refer to Sheet: Sample_IBT_Output

FIGURE 6. The Implied Tree, Probability Tree and Arrow-Debreu Tree



These calculations show the DK and BC one year four step tree parameters compared to those listed in the Applied Quantitative Finance PDF Book:

Function used : Implied_Bin_Option_Price(S, K, T, rf, qd, n, DK_or_CB, Call_or_Put, Output_Format) Output_Format = 2, 3, 4

DK

S Node

100.00	105.13	110.04	115.06	119.91
0.00	95.12	100.00	105.13	110.06
0.00	0.00	89.93	95.12	100.00
0.00	0.00	0.00	85.22	89.92
0.00	0.00	0.00	0.00	80.01

P Trans

0.56	0.59	0.58	0.60	
0.00	0.59	0.56	0.59	
0.00	0.00	0.54	0.59	
0.00	0.00	0.00	0.59	

Lamda

1.000	0.559	0.327	0.187	0.111
0.000	0.434	0.480	0.405	0.312
0.000	0.000	0.178	0.305	0.343
0.000	0.000	0.000	0.080	0.172
0.000	0.000	0.000	0.000	0.033

Derman and Kani one year (four step) implied binomial tree

stock price

```
                                    119.91
                           115.06
                  110.04                    110.06
         105.13             105.13
100.00            100.00                    100.00
         95.12              95.12
                   89.93                     89.92
                           85.22
                                     80.01
```

transition probability

```
                    0.60
               0.58
          0.59          0.59
0.56           0.56
          0.59          0.59
               0.54
                    0.59
```

Arrow-Debreu price

```
                                    0.111
                           0.187
                  0.327                     0.312
         0.559              0.405
1.000             0.480                     0.343
         0.434              0.305
                   0.178                     0.172
                           0.080
                                     0.033
```

BC

S Node

100.00	104.84	112.23	117.02	123.85
0.00	96.83	101.51	107.03	112.93
0.00	0.00	90.53	97.73	103.05
0.00	0.00	0.00	87.60	93.08
0.00	0.00	0.00	0.00	82.01

P Trans

0.49	0.38	0.61	0.46	
0.00	0.64	0.49	0.48	
0.00	0.00	0.36	0.54	
0.00	0.00	0.00	0.57	

Lamda

1.000	0.486	0.185	0.111	0.050
0.000	0.506	0.619	0.373	0.240
0.000	0.000	0.181	0.378	0.394
0.000	0.000	0.000	0.116	0.237
0.000	0.000	0.000	0.000	0.050


```
                                    123.85
                           117.02
                  112.23                    112.93
         104.84             107.03
100.00            101.51                    103.05
         96.83              97.73
                   90.53                     93.08
                           87.60
                                     82.00
```

transition probability

```
                    0.46
               0.61
          0.38          0.48
0.49           0.49
          0.64          0.54
               0.36
                    0.57
```

Arrow-Debreu price

```
                                    0.050
                           0.111
                  0.185                     0.240
         0.486              0.373
1.000             0.619                     0.394
         0.506              0.378
                   0.181                     0.237
                           0.116
                                     0.050
```

These calculations and figures compare the terminal stock price probability densities for DK and BC
20 step, 5 year trees determined in this project with the results from equivalent XploRe calculations

Function used : Implied_Bin_Option_Price(S, K, T, rf, qd, n, DK_or_CB, Call_or_Put, Output_Format) Output_Format = 6

Function used : Est_Continuous_Prob(XY_range, n, Output_Format) Output_Format = 0

Project DK

Terminal Node Stock Price	Discrete Cum. Distrib.	Discrete Prob.
29.23	0.0000	0.0000
33.65	0.0000	0.0000
38.73	0.0001	0.0000
44.43	0.0002	0.0002
50.34	0.0009	0.0006
55.05	0.0064	0.0055
59.55	0.0148	0.0084
69.89	0.0427	0.0280
79.74	0.0971	0.0544
89.80	0.1871	0.0900
100.00	0.3136	0.1265
110.12	0.4657	0.1521
119.97	0.6229	0.1572
129.40	0.7625	0.1396
138.29	0.8689	0.1064
146.59	0.9379	0.0690
154.29	0.9756	0.0377
161.38	0.9924	0.0168
167.87	0.9983	0.0059
173.80	0.9998	0.0015
179.26	1.0000	0.0002

XploRe DK

Terminal Node Stock Price	Discrete Cum. Distrib.	Discrete Prob.
24.71	0.0000	0.0000
27.48	0.0000	0.0000
30.56	0.0000	0.0000
37.91	0.0001	0.0001
45.40	0.0010	0.0009
52.39	0.0042	0.0032
60.78	0.0151	0.0109
70.35	0.0469	0.0318
80.52	0.1021	0.0552
90.38	0.1932	0.0911
100.00	0.3129	0.1197
110.23	0.4668	0.1539
120.13	0.6250	0.1582
129.58	0.7649	0.1399
138.49	0.8709	0.1060
146.81	0.9392	0.0683
154.51	0.9762	0.0370
161.59	0.9927	0.0164
168.08	0.9984	0.0057
174.01	0.9998	0.0014
179.46	1.0000	0.0002

XploRe Quantlet Java Client Version 1.4

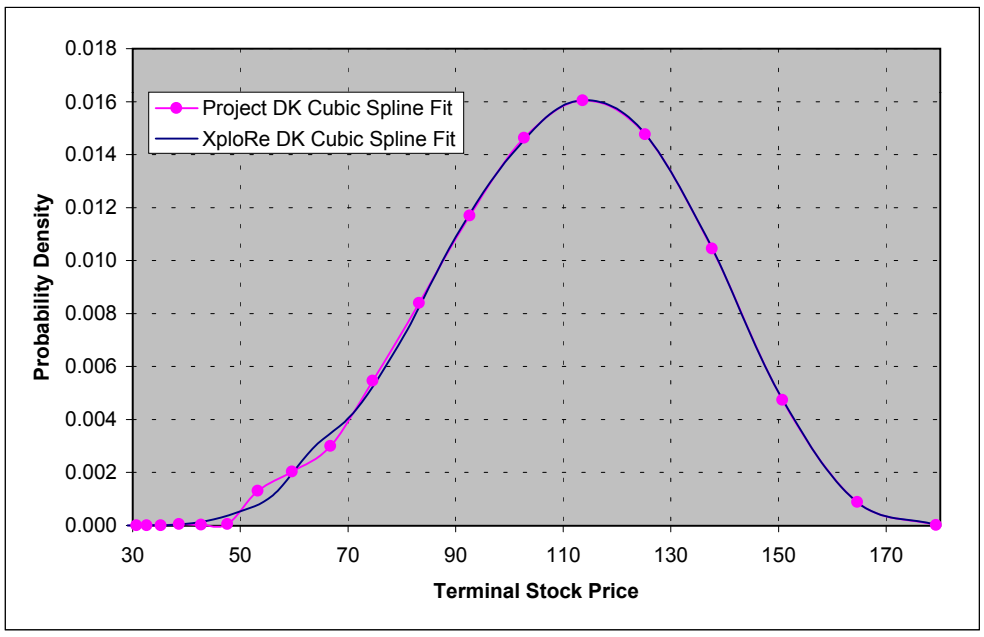
```
library("finance")
proc(sigma)=volafunc(K, S, time)
    sigma=0.1+(S-K)/S/10*0.5
endp
r=0.03    ; riskless annual interest rate
S=100    ; the underlying asset price
lev=20    ; the number of time steps
expiration=5    ; time to expiration
ibtree=IBTdk(S, r, lev, expiration, "volafunc")
dat=ibtree.Tree[,lev+1]~ibtree.lb[,lev+1]*exp(r*expiration)
dat
```

Project DK Cubic Spline Fit

Terminal Stock Price	Cont. Cum. Distrib.	Cont. Prob.
29.23	0.0000	0.0000
29.61	0.0000	0.0000
30.73	0.0000	0.0000
32.61	0.0000	0.0000
35.23	0.0000	0.0000
38.61	0.0000	0.0000
42.74	0.0003	0.0000
47.61	0.0001	0.0001
53.24	0.0037	0.0013
59.61	0.0149	0.0020
66.74	0.0319	0.0030
74.62	0.0648	0.0055
83.24	0.1245	0.0084
92.62	0.2187	0.0117
102.75	0.3528	0.0146
113.62	0.5216	0.0160
125.25	0.7036	0.0148
137.63	0.8620	0.0104
150.75	0.9612	0.0047
164.63	0.9962	0.0009
179.26	1.0000	0.0000

XploRe DK Cubic Spline Fit

Terminal Stock Price	Cont. Cum. Distrib.	Cont. Prob.
24.71	0.0000	0.0000
25.10	0.0000	0.0000
26.26	0.0000	0.0000
28.19	0.0000	0.0000
30.90	0.0000	0.0000
34.38	0.0000	0.0000
38.64	0.0001	0.0000
43.67	0.0006	0.0002
49.47	0.0025	0.0005
56.05	0.0074	0.0011
63.40	0.0218	0.0029
71.52	0.0520	0.0044
80.42	0.1014	0.0072
90.09	0.1900	0.0109
100.54	0.3204	0.0140
111.76	0.4911	0.0160
123.75	0.6810	0.0151
136.52	0.8502	0.0109
150.06	0.9577	0.0050
164.37	0.9959	0.0009
179.46	1.0000	0.0000



Project BC

Terminal Node Stock Price	Discrete Cum. Distrib.	Discrete Prob.
29.37	0.0000	0.0000
35.62	0.0001	0.0001
42.21	0.0006	0.0005
49.58	0.0027	0.0021
57.73	0.0098	0.0071
66.61	0.0289	0.0191
76.10	0.0710	0.0421
86.01	0.1473	0.0763
96.13	0.2628	0.1155
106.26	0.4100	0.1472
116.18	0.5692	0.1592
125.76	0.7163	0.1471
134.87	0.8330	0.1167
143.46	0.9127	0.0797
151.50	0.9598	0.0471
159.02	0.9838	0.0240
166.06	0.9944	0.0106
172.69	0.9984	0.0040
178.99	0.9996	0.0012
184.92	0.9999	0.0003
192.11	1.0000	0.0001

XploRe BC

Terminal Node Stock Price	Discrete Cum. Distrib.	Discrete Prob.
29.37	0.0000	0.0000
35.62	0.0001	0.0001
42.18	0.0006	0.0005
49.51	0.0026	0.0021
57.63	0.0096	0.0070
66.49	0.0285	0.0189
76.05	0.0709	0.0424
86.01	0.1473	0.0763
96.13	0.2628	0.1155
106.26	0.4100	0.1472
116.18	0.5692	0.1592
125.76	0.7163	0.1471
134.87	0.8330	0.1167
143.46	0.9127	0.0797
151.55	0.9602	0.0475
159.07	0.9838	0.0236
166.05	0.9944	0.0106
172.68	0.9984	0.0040
179.00	0.9996	0.0012
184.93	0.9999	0.0003
192.12	1.0000	0.0001

XploRe Quantlet Java Client Version 1.4

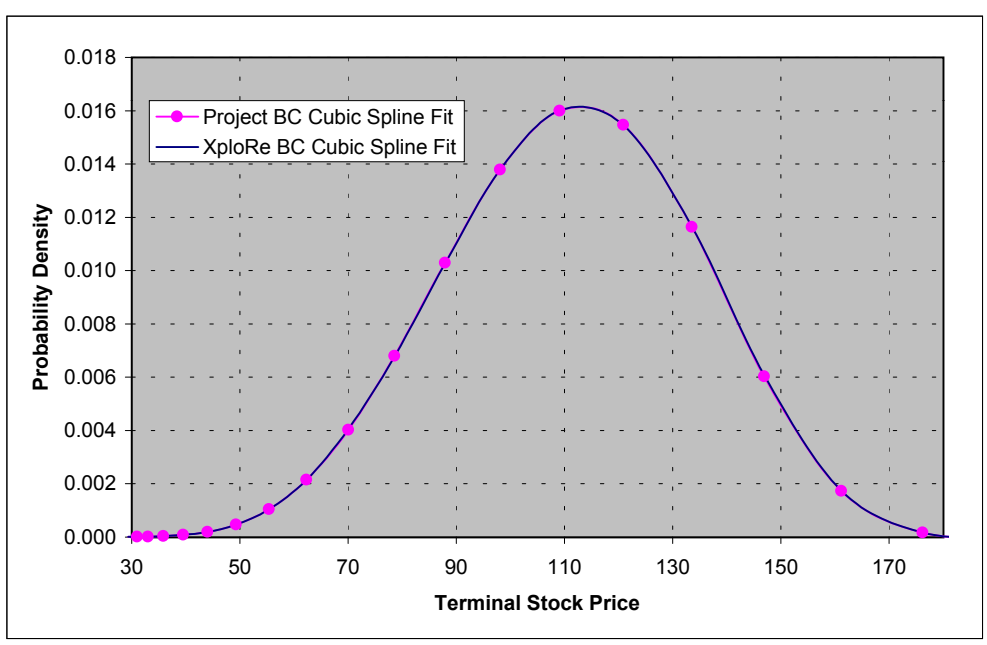
```
library("finance")
proc(sigma)=volafunc(K, S, time)
    sigma=0.1+(S-K)/S/10*0.5
endp
r=0.03    ; riskless annual interest rate
S=100    ; the underlying asset price
lev=20    ; the number of time steps
expiration=5    ; time to expiration
ibtree=IBTbc(S, r, lev, expiration, "volafunc")
dat=ibtree.Tree[,lev+1]~ibtree.lb[,lev+1]*exp(r*expiration)
dat
```

Project BC Cubic Spline Fit

Terminal Stock Price	Cont. Cum. Distrib.	Cont. Prob.
29.37	0.0000	0.0000
29.78	0.0000	0.0000
31.00	0.0000	0.0000
33.04	0.0000	0.0000
35.88	0.0001	0.0000
39.54	0.0003	0.0001
44.02	0.0009	0.0002
49.31	0.0026	0.0005
55.41	0.0070	0.0010
62.33	0.0177	0.0021
70.06	0.0411	0.0040
78.60	0.0869	0.0068
87.96	0.1666	0.0103
98.13	0.2897	0.0138
109.11	0.4552	0.0160
120.91	0.6439	0.0155
133.52	0.8177	0.0116
146.95	0.9362	0.0060
161.19	0.9880	0.0017
176.24	0.9993	0.0002
192.11	1.0000	0.0000

XploRe BC Cubic Spline Fit

Terminal Stock Price	Cont. Cum. Distrib.	Cont. Prob.
29.37	0.0000	0.0000
29.78	0.0000	0.0000
31.00	0.0000	0.0000
33.03	0.0000	0.0000
35.88	0.0001	0.0000
39.54	0.0003	0.0001
44.02	0.0009	0.0002
49.31	0.0025	0.0005
55.41	0.0070	0.0010
62.33	0.0177	0.0021
70.06	0.0412	0.0041
78.60	0.0871	0.0068
87.96	0.1666	0.0103
98.13	0.2897	0.0138
109.12	0.4553	0.0160
120.92	0.6440	0.0154
133.53	0.8178	0.0116
146.96	0.9364	0.0061
161.20	0.9879	0.0017
176.25	0.9993	0.0002
192.12	1.0000	0.0000



These calculations and figures compare the terminal stock price probability densities for DK, BC and CRR 20 step trees with the lognormal probability density (compare DK and BC probability densities with results in Figures 7.2 and 7.4 of the Applied Quantitative Finance PDF Book):

Function used : Implied_Bin_Option_Price(S, K, T, rf, qd, n, DK_or_CB, Call_or_Put, Output_Format) Output_Format = 6

Function used : Est_Continuous_Prob(XY_range, n, Output_Format) Output_Format = 0 and 1

Function used : Log_Normal_Density(S0, ST, T, sigma, rf, qd)

DK:

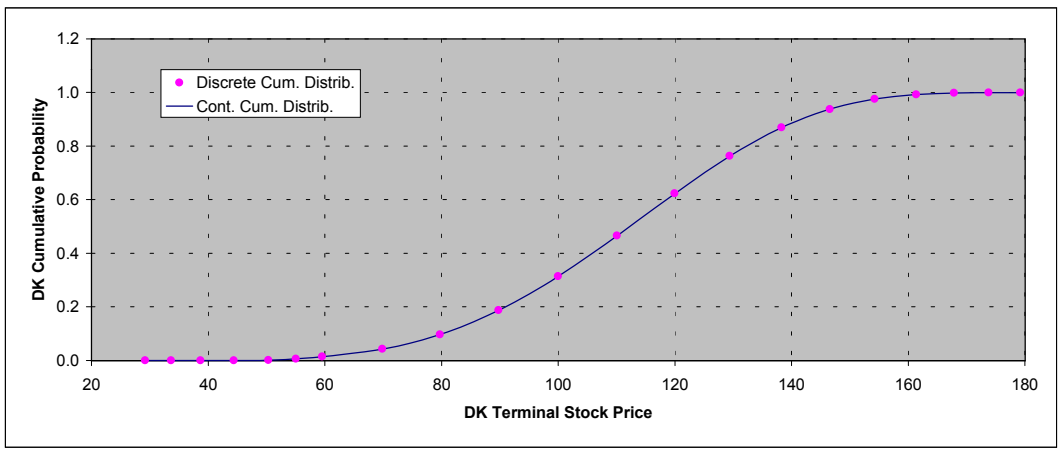
Terminal Node Stock Price	Discrete Cum. Distrib.	Discrete Prob.
29.23	0.0000	0.0000
33.65	0.0000	0.0000
38.73	0.0001	0.0000
44.43	0.0002	0.0002
50.34	0.0009	0.0006
55.05	0.0064	0.0055
59.55	0.0148	0.0084
69.89	0.0427	0.0280
79.74	0.0971	0.0544
89.80	0.1871	0.0900
100.00	0.3136	0.1265
110.12	0.4657	0.1521
119.97	0.6229	0.1572
129.40	0.7625	0.1396
138.29	0.8689	0.1064
146.59	0.9379	0.0690
154.29	0.9756	0.0377
161.38	0.9924	0.0168
167.87	0.9983	0.0059
173.80	0.9998	0.0015
179.26	1.0000	0.0002

BC:

Terminal Node Stock Price	Discrete Cum. Distrib.	Discrete Prob.
29.37	0.0000	0.0000
35.62	0.0001	0.0001
42.21	0.0006	0.0005
49.58	0.0027	0.0021
57.73	0.0098	0.0071
66.61	0.0289	0.0191
76.10	0.0710	0.0421
86.01	0.1473	0.0763
96.13	0.2628	0.1155
106.26	0.4100	0.1472
116.18	0.5692	0.1592
125.76	0.7163	0.1471
134.87	0.8330	0.1167
143.46	0.9127	0.0797
151.50	0.9598	0.0471
159.02	0.9838	0.0240
166.06	0.9944	0.0106
172.69	0.9984	0.0040
178.99	0.9996	0.0012
184.92	0.9999	0.0003
192.11	1.0000	0.0001

CRR:

Terminal Node Stock Price	Discrete Cum. Distrib.	Discrete Prob.
36.79	0.0000	0.0000
40.66	0.0000	0.0000
44.93	0.0000	0.0000
49.66	0.0002	0.0002
54.88	0.0010	0.0009
60.65	0.0046	0.0036
67.03	0.0161	0.0115
74.08	0.0457	0.0296
81.87	0.1076	0.0619
90.48	0.2138	0.1062
100.00	0.3641	0.1503
110.52	0.5400	0.1759
122.14	0.7098	0.1698
134.99	0.8442	0.1345
149.18	0.9308	0.0865
164.87	0.9753	0.0445
182.21	0.9932	0.0179
201.38	0.9987	0.0054
222.55	0.9998	0.0012
245.96	1.0000	0.0002
271.83	1.0000	0.0000



DK

Terminal Stock Price	Cont. Cum. Distrib.	Cont. Prob.
29.23	0.0000	0.0000
29.33	0.0000	0.0000
29.61	0.0000	0.0000
30.08	0.0000	0.0000
30.73	0.0000	0.0000
31.58	0.0000	0.0000
32.61	0.0000	0.0000
33.83	0.0000	0.0000
35.23	0.0000	0.0000
36.83	0.0000	0.0000
38.61	0.0000	0.0000
40.58	0.0002	0.0001
42.74	0.0003	0.0000
45.08	0.0002	0.0000
47.61	0.0001	0.0001
50.33	0.0009	0.0006
53.24	0.0037	0.0013
56.33	0.0086	0.0018
59.61	0.0149	0.0020
63.08	0.0223	0.0023
66.74	0.0319	0.0030
70.58	0.0455	0.0041
74.62	0.0648	0.0055
78.84	0.0908	0.0069
83.24	0.1245	0.0084
87.84	0.1667	0.0100
92.62	0.2187	0.0117
97.59	0.2808	0.0133
102.75	0.3528	0.0146
108.09	0.4338	0.0156
113.62	0.5216	0.0160
119.34	0.6130	0.0158
125.25	0.7036	0.0148
131.34	0.7883	0.0129
137.63	0.8620	0.0104
144.10	0.9204	0.0076
150.75	0.9612	0.0047
157.60	0.9852	0.0024
164.63	0.9962	0.0009
171.85	0.9995	0.0002
179.26	1.0000	0.0000

BC

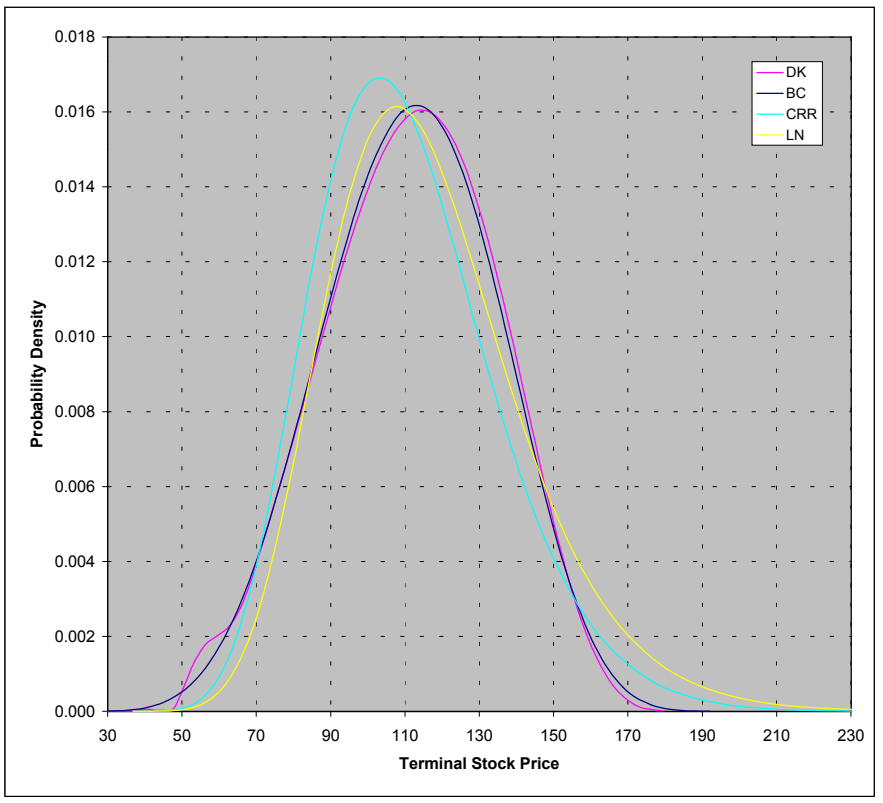
Terminal Stock Price	Cont. Cum. Distrib.	Cont. Prob.
29.37	0.0000	0.0000
29.48	0.0000	0.0000
29.78	0.0000	0.0000
30.29	0.0000	0.0000
31.00	0.0000	0.0000
31.92	0.0000	0.0000
33.04	0.0000	0.0000
34.36	0.0001	0.0000
35.88	0.0001	0.0000
37.61	0.0002	0.0001
39.54	0.0003	0.0001
41.68	0.0005	0.0001
44.02	0.0009	0.0002
46.56	0.0015	0.0003
49.31	0.0026	0.0005
52.26	0.0043	0.0007
55.41	0.0070	0.0010
58.77	0.0112	0.0015
62.33	0.0177	0.0021
66.09	0.0273	0.0030
70.06	0.0411	0.0040
74.23	0.0605	0.0053
78.60	0.0869	0.0068
83.18	0.1218	0.0085
87.96	0.1666	0.0103
92.94	0.2225	0.0121
98.13	0.2897	0.0138
103.52	0.3679	0.0151
109.11	0.4552	0.0160
114.91	0.5487	0.0161
120.91	0.6439	0.0155
127.12	0.7355	0.0139
133.52	0.8177	0.0116
140.13	0.8856	0.0089
146.95	0.9362	0.0060
153.97	0.9695	0.0035
161.19	0.9880	0.0017
168.61	0.9964	0.0007
176.24	0.9993	0.0002
184.07	0.9999	0.0000
192.11	1.0000	0.0000

CRR

Terminal Stock Price	Cont. Cum. Distrib.	Cont. Prob.
36.79	0.0000	0.0000
36.93	0.0000	0.0000
37.38	0.0000	0.0000
38.11	0.0000	0.0000
39.14	0.0000	0.0000
40.46	0.0000	0.0000
42.08	0.0000	0.0000
43.99	0.0000	0.0000
46.19	0.0000	0.0000
48.69	0.0001	0.0001
51.48	0.0003	0.0001
54.56	0.0010	0.0003
57.94	0.0024	0.0006
61.61	0.0057	0.0012
65.58	0.0125	0.0023
69.84	0.0253	0.0038
74.39	0.0476	0.0060
79.24	0.0830	0.0086
84.38	0.1347	0.0115
89.82	0.2043	0.0141
95.55	0.2909	0.0160
101.57	0.3905	0.0169
107.89	0.4969	0.0166
114.50	0.6024	0.0152
121.40	0.7002	0.0130
128.60	0.7847	0.0104
136.09	0.8531	0.0079
143.88	0.9049	0.0055
151.96	0.9418	0.0037
160.33	0.9663	0.0023
169.00	0.9816	0.0013
177.96	0.9906	0.0007
187.21	0.9955	0.0004
196.76	0.9980	0.0002
206.60	0.9992	0.0001
216.74	0.9997	0.0000
227.17	0.9999	0.0000
237.89	1.0000	0.0000
248.91	1.0000	0.0000
260.22	1.0000	0.0000
271.83	1.0000	0.0000

LN

Cont. Prob.
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0001
0.0002
0.0003
0.0007
0.0014
0.0025
0.0041
0.0063
0.0089
0.0116
0.0140
0.0156
0.0161
0.0156
0.0140
0.0118
0.0094
0.0070
0.0050
0.0033
0.0021
0.0013
0.0008
0.0004
0.0002
0.0001
0.0001
0.0000
0.0000
0.0000
0.0000



DK:

Terminal Stock Price	Cont. Prob.
29.23	0.0000
29.33	0.0000
29.61	0.0000
30.08	0.0000
30.73	0.0000
31.58	0.0000
32.61	0.0000
33.83	0.0000
35.23	0.0000
36.83	0.0000
38.61	0.0000
40.58	0.0001
42.74	0.0000
45.08	0.0000
47.61	0.0001
50.33	0.0006
53.24	0.0013
56.33	0.0018
59.61	0.0020
63.08	0.0023
66.74	0.0030
70.58	0.0041
74.62	0.0055
78.84	0.0069
83.24	0.0084
87.84	0.0100
92.62	0.0117
97.59	0.0133
102.75	0.0146
108.09	0.0156
113.62	0.0160
119.34	0.0158
125.25	0.0148
131.34	0.0129
137.63	0.0104
144.10	0.0076
150.75	0.0047
157.60	0.0024
164.63	0.0009
171.85	0.0002
179.26	0.0000
Area	**1.000**

BC:

Terminal Stock Price	Cont. Prob.
29.37	0.0000
29.48	0.0000
29.78	0.0000
30.29	0.0000
31.00	0.0000
31.92	0.0000
33.04	0.0000
34.36	0.0000
35.88	0.0000
37.61	0.0001
39.54	0.0001
41.68	0.0001
44.02	0.0002
46.56	0.0003
49.31	0.0005
52.26	0.0007
55.41	0.0010
58.77	0.0015
62.33	0.0021
66.09	0.0030
70.06	0.0040
74.23	0.0053
78.60	0.0068
83.18	0.0085
87.96	0.0103
92.94	0.0121
98.13	0.0138
103.52	0.0151
109.11	0.0160
114.91	0.0161
120.91	0.0155
127.12	0.0139
133.52	0.0116
140.13	0.0089
146.95	0.0060
153.97	0.0035
161.19	0.0017
168.61	0.0007
176.24	0.0002
184.07	0.0000
192.11	0.0000
Area	**1.000**

CRR:

Terminal Stock Price	Cont. Prob.
36.79	0.0000
36.93	0.0000
37.38	0.0000
38.11	0.0000
39.14	0.0000
40.46	0.0000
42.08	0.0000
43.99	0.0000
46.19	0.0000
48.69	0.0001
51.48	0.0001
54.56	0.0003
57.94	0.0006
61.61	0.0012
65.58	0.0023
69.84	0.0038
74.39	0.0060
79.24	0.0086
84.38	0.0115
89.82	0.0141
95.55	0.0160
101.57	0.0169
107.89	0.0166
114.50	0.0152
121.40	0.0130
128.60	0.0104
136.09	0.0079
143.88	0.0055
151.96	0.0037
160.33	0.0023
169.00	0.0013
177.96	0.0007
187.21	0.0004
196.76	0.0002
206.60	0.0001
216.74	0.0000
227.17	0.0000
237.89	0.0000
248.91	0.0000
260.22	0.0000
271.83	0.0000
Area	**1.000**

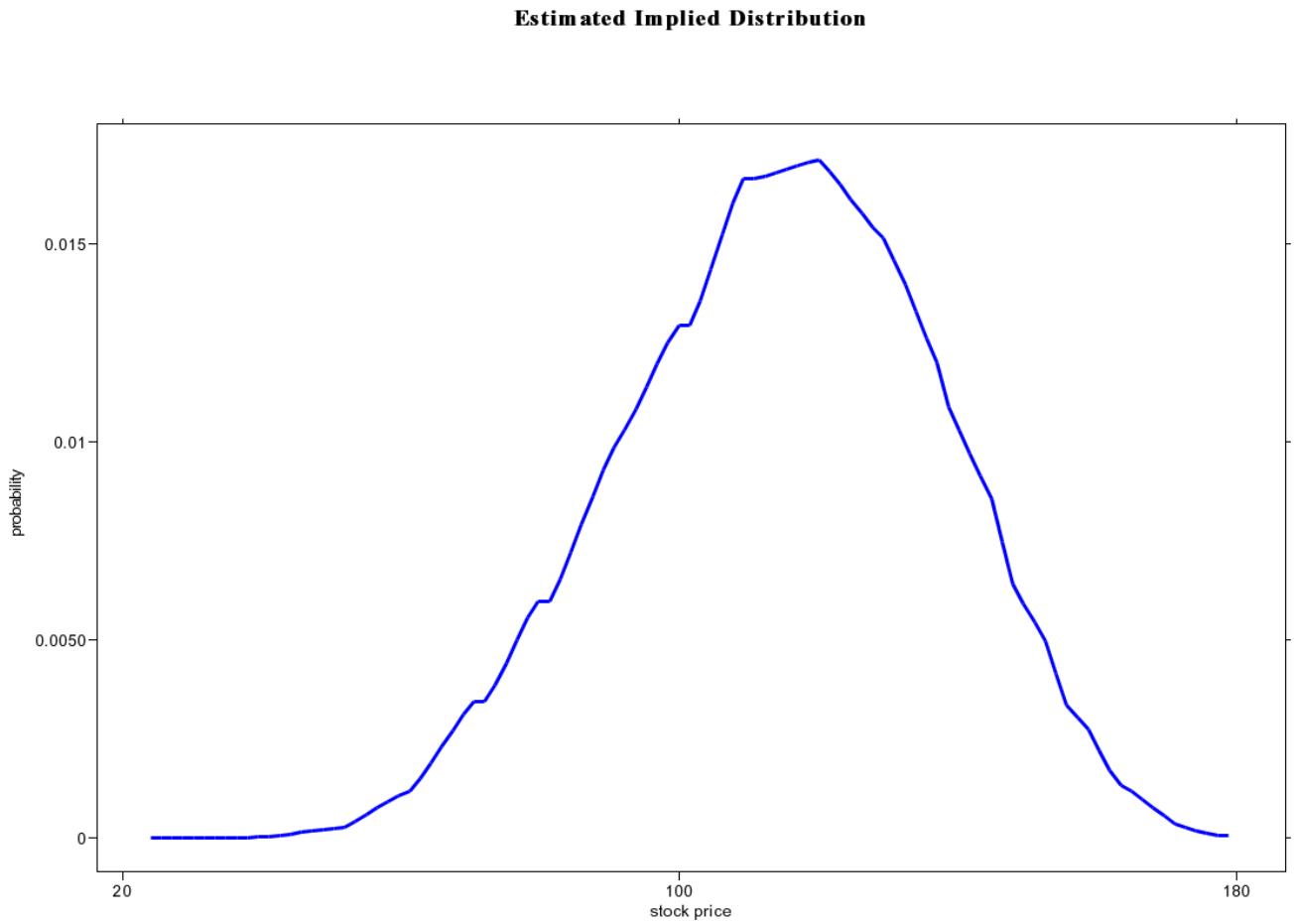
LN:

Terminal Stock Price	Cont. Prob.
36.79	0.0000
36.93	0.0000
37.38	0.0000
38.11	0.0000
39.14	0.0000
40.46	0.0000
42.08	0.0000
43.99	0.0000
46.19	0.0000
48.69	0.0000
51.48	0.0001
54.56	0.0002
57.94	0.0003
61.61	0.0007
65.58	0.0014
69.84	0.0025
74.39	0.0041
79.24	0.0063
84.38	0.0089
89.82	0.0116
95.55	0.0140
101.57	0.0156
107.89	0.0161
114.50	0.0156
121.40	0.0140
128.60	0.0118
136.09	0.0094
143.88	0.0070
151.96	0.0050
160.33	0.0033
169.00	0.0021
177.96	0.0013
187.21	0.0008
196.76	0.0004
206.60	0.0002
216.74	0.0001
227.17	0.0001
237.89	0.0000
248.91	0.0000
260.22	0.0000
271.83	0.0000
Area	**1.000**

Note that as a check on the estimated probability density, as determined from the first derivative of the cumulative probability distribution, we should find that the area under the probability density curve = 1.000 for the interval -ve infinity to +ve infinity.

XploRe Quantlet Java Client Version 1.4

```
library("finance")
proc(sigma)=volafunc(K, S, time)
    sigma=0.1+(S-K)/S/10*0.5
endp
r=0.03    ; riskless annual interest rate
S=100    ; the underlying asset price
lev=20    ; the number of time steps
expiration=5    ; time to expiration
ibtree=IBTdk(S, r, lev, expiration, "volafunc")
dat=ibtree.Tree[,lev+1]~ibtree.lb[,lev+1]*exp(r*expiration)
dat
bandwidth=10
mh=IBTsdisplot(dat, bandwidth)
```

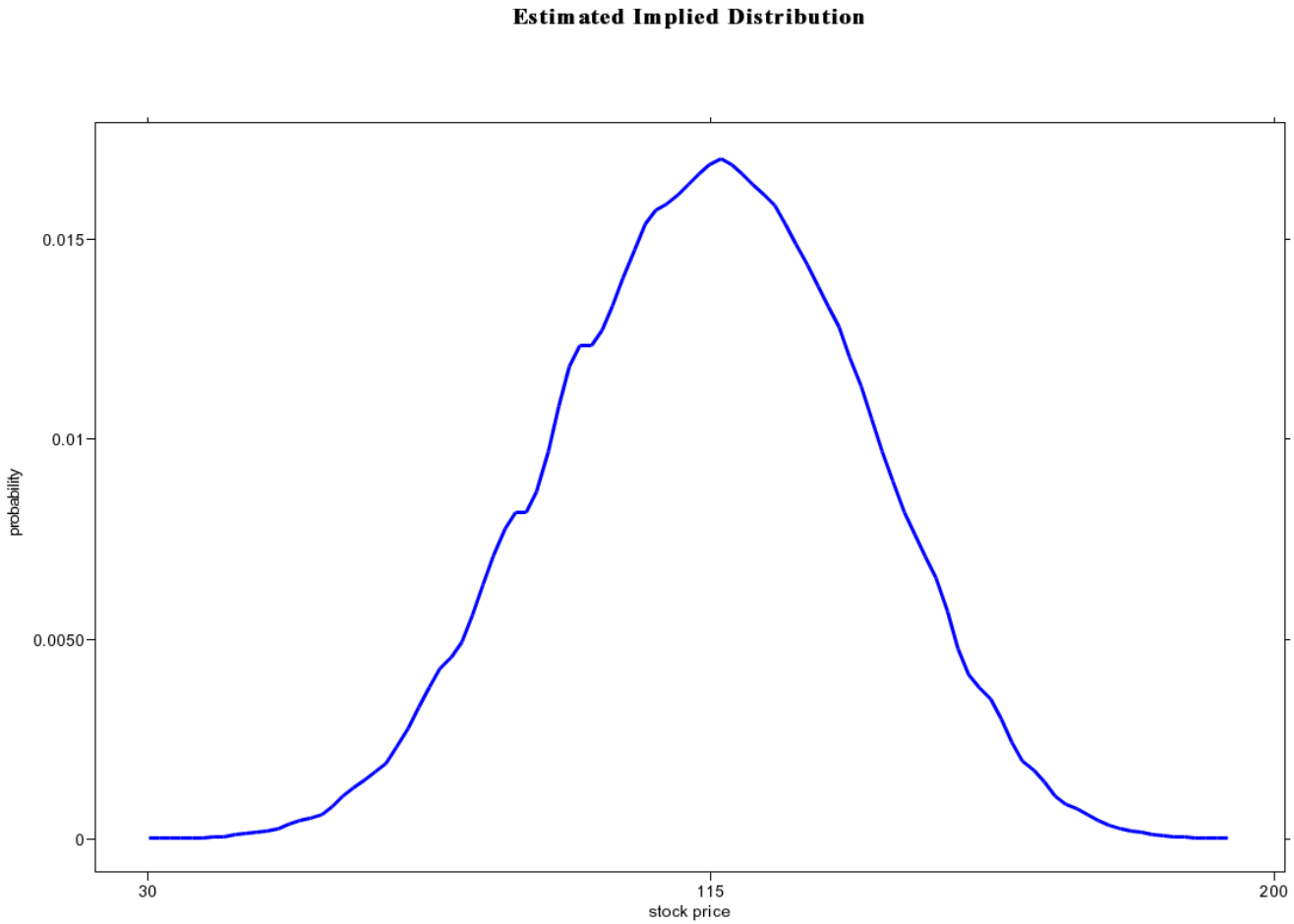


Applied Quantitative Finance book, Figure 7.2 - DK 5-year, 20-step terminal stock price probability density.

See Sheet: DK_BC_Terminal_Prob_Densities

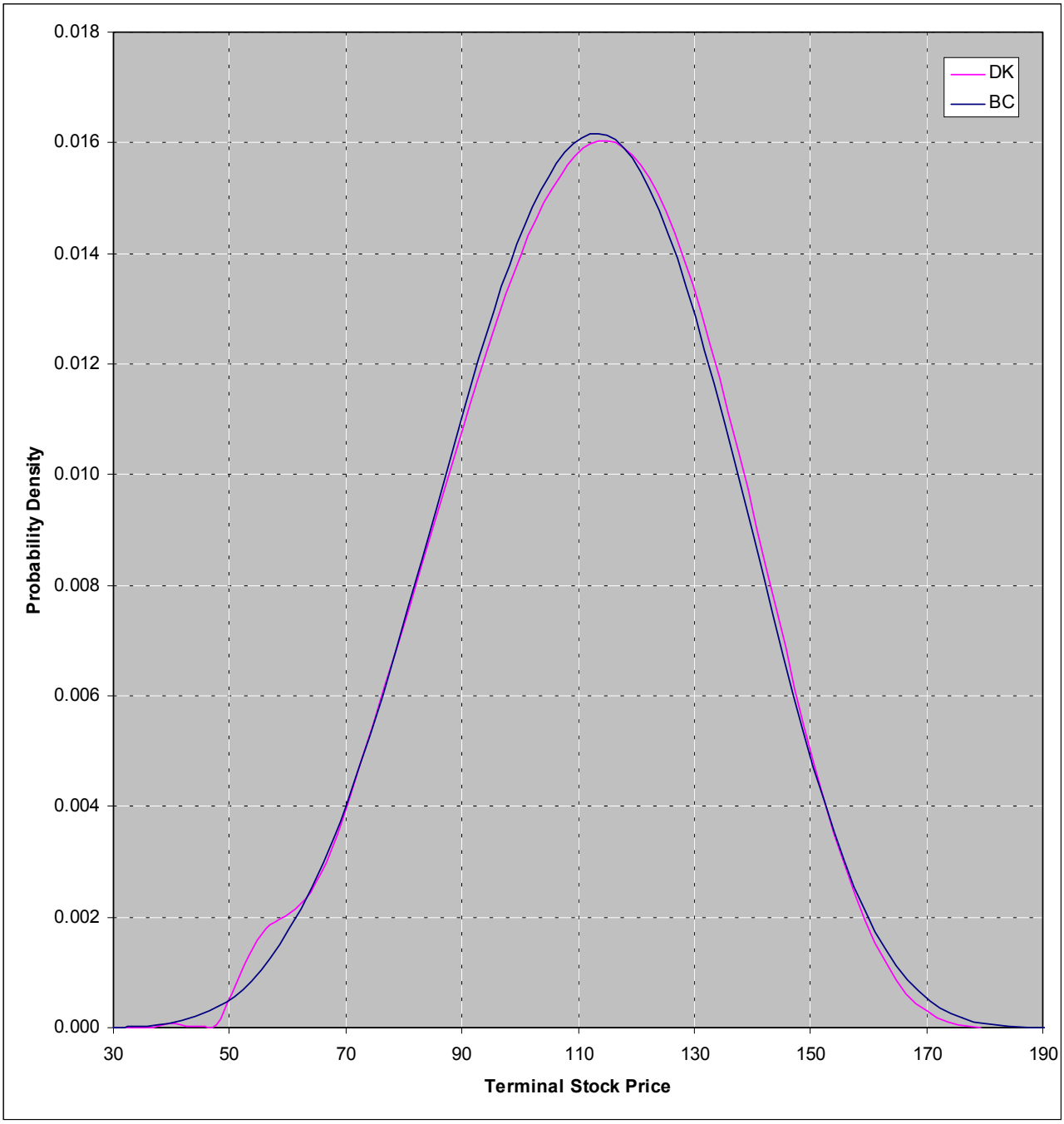
XploRe Quantlet Java Client Version 1.4

```
library("finance")
proc(sigma)=volafunc(K, S, time)
    sigma=0.1+(S-K)/S/10*0.5
endp
r=0.03    ; riskless annual interest rate
S=100    ; the underlying asset price
lev=20    ; the number of time steps
expiration=5    ; time to expiration
ibtree=IBTbc(S, r, lev, expiration, "volafunc")
dat=ibtree.Tree[,lev+1]~ibtree.lb[,lev+1]*exp(r*expiration)
dat
bandwidth=10
mh=IBTsdisplot(dat, bandwidth)
```



Applied Quantitative Finance book, Figure 7.4 - BC 5-year, 20-step terminal stock price probability density.

See Sheet: DK_BC_Terminal_Prob_Densities



Project Results - DK and BC 5-year, 20-step terminal stock price probability density.

See Sheet: DK_BC_Terminal_Prob_Densities

The data on this sheet compares estimated implied local volatility of each node in an implied binomial tree from the XploRe function IBTlocsigma and the equivalent implementation developed in this project:

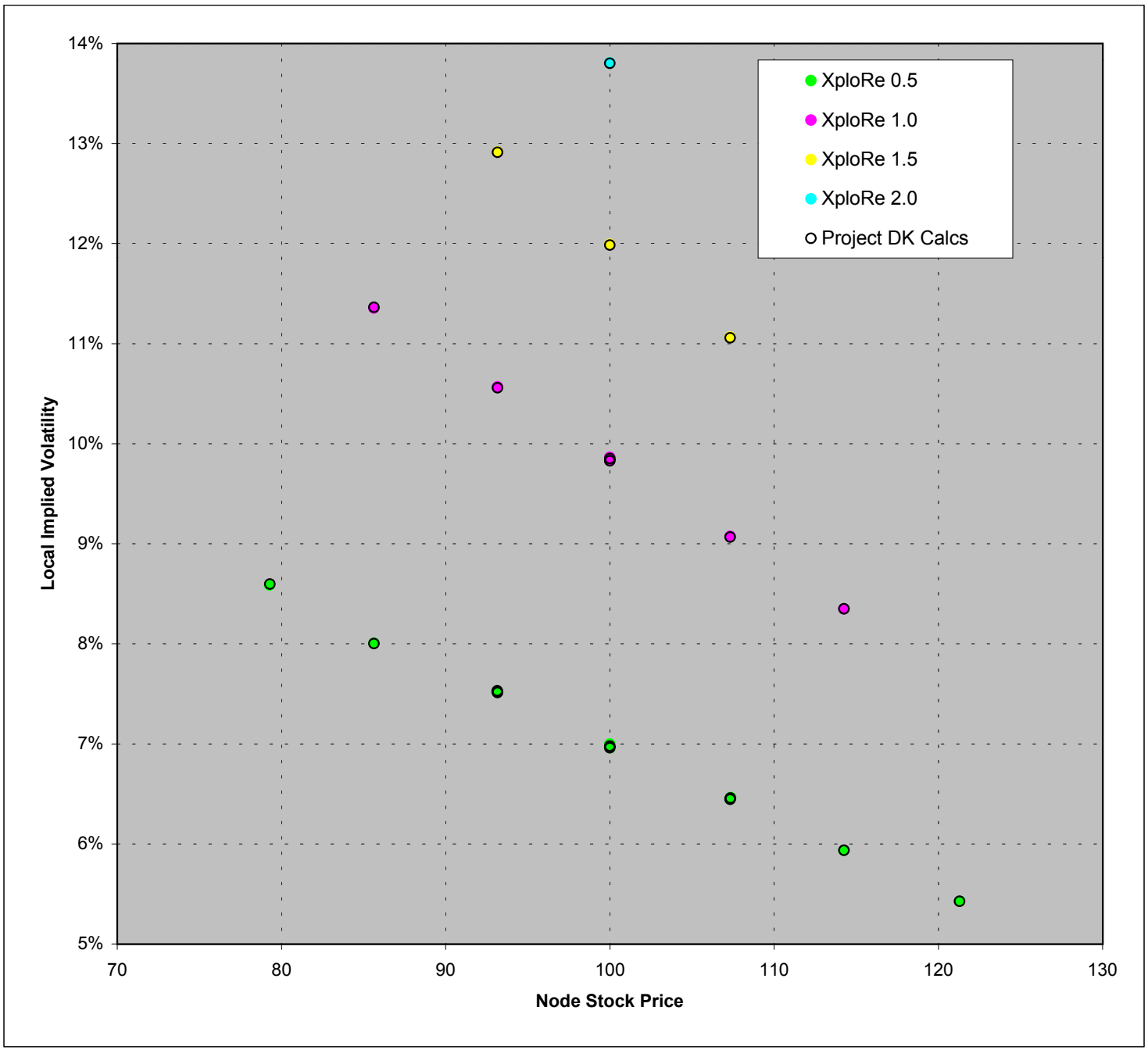
See : http://www.xplore-stat.de/help/IBTlocsigma.html but with m = 4 for XploRe Calcs

Estimates the implied local volatility of each node in an implied binomial tree

Function used : Implied_Bin_Option_Price(S, K, T, rf, qd, n, DK_or_CB, Call_or_Put, Output_Format) Output_Format = 9

XploRe IBTlocsigma DK Calcs

Node Stock Price	Time to Expiry (years)	Implied Local Volatility (%)
79.29	XploRe 0.5	8.59%
85.65	0.5	8.00%
93.14	0.5	7.52%
93.17	0.5	7.51%
100.00	0.5	6.96%
100.00	0.5	7.00%
107.33	0.5	6.45%
107.37	0.5	6.46%
114.26	0.5	5.94%
121.32	0.5	5.43%
85.65	XploRe 1.0	11.36%
93.17	1.0	10.57%
100.00	1.0	9.83%
100.00	1.0	9.86%
107.33	1.0	9.07%
114.26	1.0	8.35%
93.17	XploRe 1.5	12.92%
100.00	1.5	11.99%
107.33	1.5	11.06%
100.00	XploRe 2.0	13.81%



Project DK Calcs

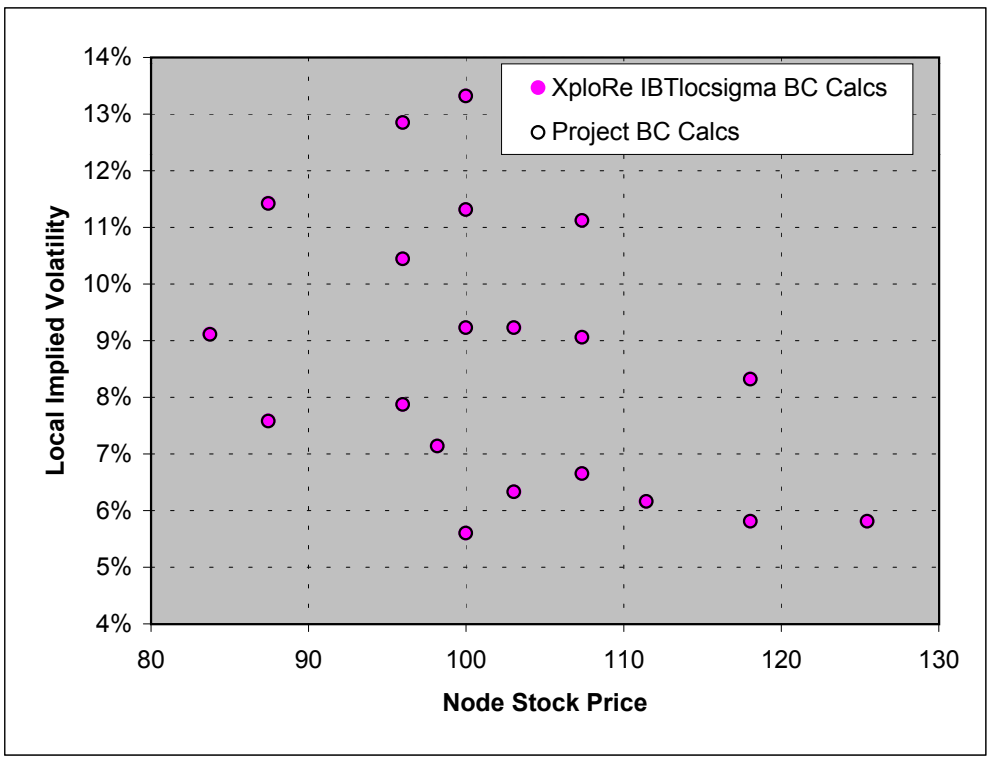
Node Stock Price	Time to Expiry (years)	Implied Local Volatility (%)
100.00	0.5	6.96%
100.00	1.0	9.83%
100.00	1.5	11.98%
100.00	2.0	13.80%
107.33	0.5	6.44%
107.33	1.0	9.07%
107.33	1.5	11.06%
93.17	0.5	7.51%
93.17	1.0	10.56%
93.17	1.5	12.91%
114.26	0.5	5.93%
114.26	1.0	8.35%
100.00	0.5	6.98%
100.00	1.0	9.85%
85.65	0.5	8.00%
85.65	1.0	11.36%
121.30	0.5	5.43%
107.35	0.5	6.46%
93.16	0.5	7.53%
79.31	0.5	8.60%

Project BC Calcs

Node Stock Price	Time to Expiry (years)	Implied Local Volatility (%)
100.00	0.50	5.60%
100.00	1.00	9.23%
100.00	1.50	11.31%
100.00	2.00	13.32%
107.36	0.50	6.65%
107.36	1.00	9.06%
107.36	1.50	11.12%
95.98	0.50	7.87%
95.98	1.00	10.44%
95.98	1.50	12.85%
118.06	0.50	5.81%
118.06	1.00	8.32%
103.05	0.50	6.33%
103.05	1.00	9.22%
87.46	0.50	7.58%
87.46	1.00	11.42%
125.46	0.50	5.81%
111.44	0.50	6.16%
98.18	0.50	7.14%
83.75	0.50	9.11%

XploRe IBTlocsigma BC Calcs

Node Stock Price	Time to Expiry (years)	Implied Local Volatility (%)
100	0.5	5.60%
100	1.0	9.23%
100	1.5	11.31%
100	2.0	13.32%
95.983	0.5	7.87%
95.983	1.0	10.44%
95.983	1.5	12.85%
107.36	0.5	6.65%
107.36	1.0	9.06%
107.36	1.5	11.12%
87.463	0.5	7.58%
87.463	1.0	11.42%
103.05	0.5	6.33%
103.05	1.0	9.22%
118.06	0.5	5.81%
118.06	1.0	8.32%
83.747	0.5	9.11%
98.183	0.5	7.14%
111.44	0.5	6.16%
125.46	0.5	5.81%

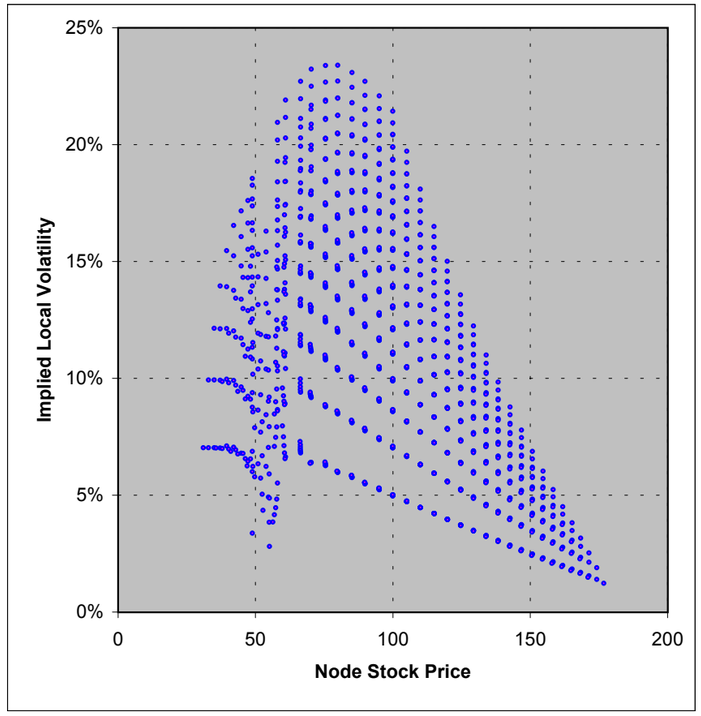


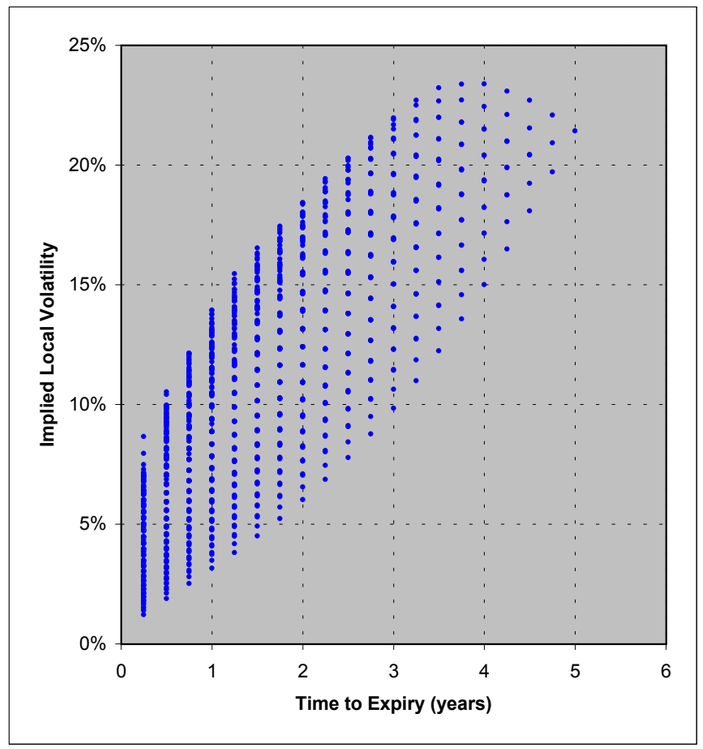
These calculations and figures represent the data in Figures 7.3 (DK) and 7.5 (BC) of the Applied Quantitative Finance PDF Book:

Function used : Implied_Bin_Option_Price(S, K, T, rf, qd, n, DK_or_CB, Call_or_Put, Output_Format) Output_Format = 9

DK:

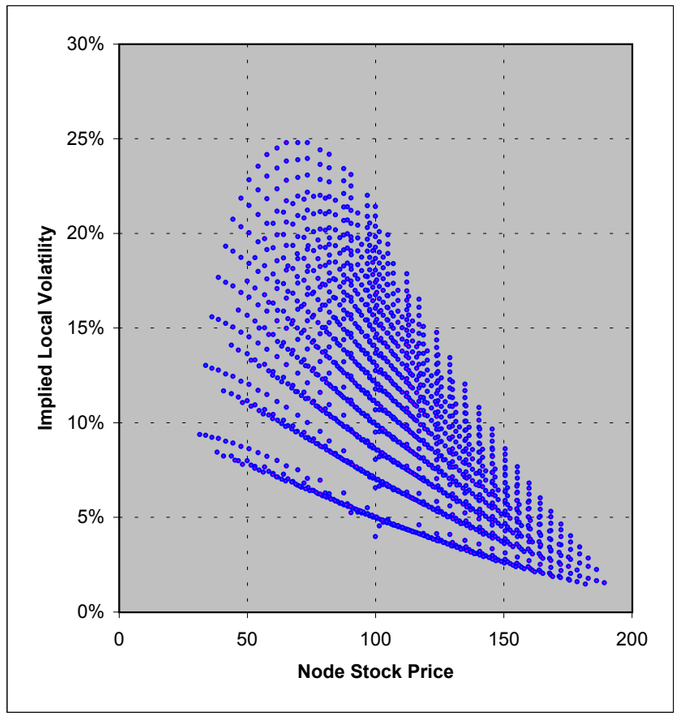
Node Stock Price	Time to Expiry (years)	Implied Local Volatility (%)
100.00	0.25	4.96%
100.00	0.50	7.01%
100.00	0.75	8.57%
100.00	1.00	9.88%
100.00	1.25	11.03%
100.00	1.50	12.06%
100.00	1.75	13.00%
100.00	2.00	13.88%
100.00	2.25	14.69%
100.00	2.50	15.46%
100.00	2.75	16.18%
100.00	3.00	16.87%
100.00	3.25	17.53%
100.00	3.50	18.16%
100.00	3.75	18.75%
100.00	4.00	19.33%
100.00	4.25	19.88%
100.00	4.50	20.41%
100.00	4.75	20.92%
100.00	5.00	21.42%
105.13	0.25	4.71%
105.13	0.50	6.64%
105.13	0.75	8.12%
105.13	1.00	9.36%
105.13	1.25	10.44%
105.13	1.50	11.41%
105.13	1.75	12.30%
105.13	2.00	13.12%
105.13	2.25	13.89%
105.13	2.50	14.61%
105.13	2.75	15.29%
105.13	3.00	15.93%
105.13	3.25	16.54%
105.13	3.50	17.13%
105.13	3.75	17.69%
105.13	4.00	18.22%
105.13	4.25	18.74%
105.13	4.50	19.23%
105.13	4.75	19.71%
95.12	0.25	5.23%
95.12	0.50	7.37%
95.12	0.75	9.02%
95.12	1.00	10.39%
95.12	1.25	11.61%
95.12	1.50	12.69%
95.12	1.75	13.69%
95.12	2.00	14.61%
95.12	2.25	15.48%
95.12	2.50	16.29%
95.12	2.75	17.06%
95.12	3.00	17.78%
95.12	3.25	18.48%
95.12	3.50	19.14%
95.12	3.75	19.78%
95.12	4.00	20.39%
95.12	4.25	20.97%
95.12	4.50	21.54%
95.12	4.75	22.08%
110.04	0.25	4.46%

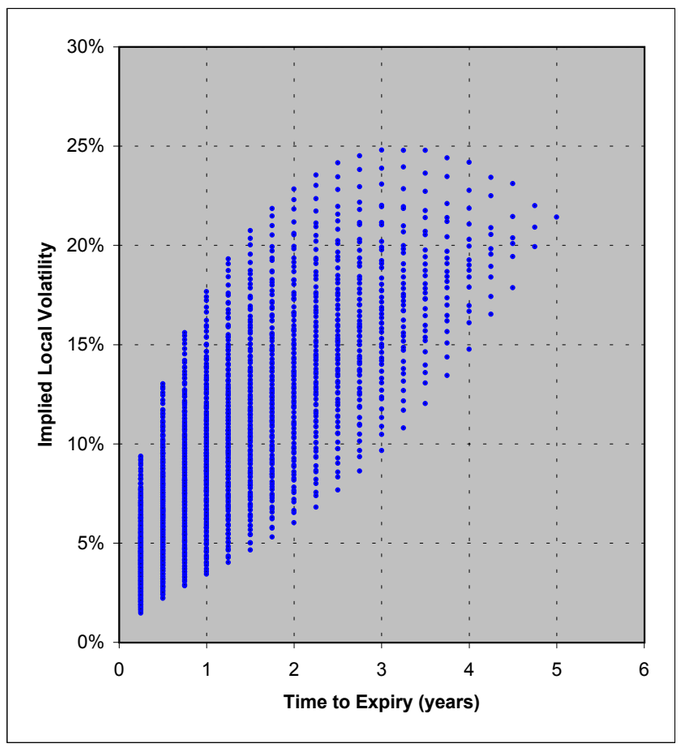




BC:

Node Stock Price	Time to Expiry (years)	Implied Local Volatility (%)
100.00	0.25	3.97%
100.00	0.50	6.56%
100.00	0.75	8.05%
100.00	1.00	9.50%
100.00	1.25	10.68%
100.00	1.50	11.77%
100.00	1.75	12.75%
100.00	2.00	13.66%
100.00	2.25	14.50%
100.00	2.50	15.29%
100.00	2.75	16.03%
100.00	3.00	16.74%
100.00	3.25	17.41%
100.00	3.50	18.06%
100.00	3.75	18.67%
100.00	4.00	19.26%
100.00	4.25	19.83%
100.00	4.50	20.38%
100.00	4.75	20.90%
100.00	5.00	21.42%
104.84	0.25	4.88%
104.84	0.50	6.63%
104.84	0.75	8.16%
104.84	1.00	9.39%
104.84	1.25	10.49%
104.84	1.50	11.48%
104.84	1.75	12.38%
104.84	2.00	13.21%
104.84	2.25	13.99%
104.84	2.50	14.72%
104.84	2.75	15.41%
104.84	3.00	16.06%
104.84	3.25	16.69%
104.84	3.50	17.28%
104.84	3.75	17.85%
104.84	4.00	18.40%
104.84	4.25	18.93%
104.84	4.50	19.43%
104.84	4.75	19.92%
96.83	0.25	5.50%
96.83	0.50	7.31%
96.83	0.75	9.02%
96.83	1.00	10.35%
96.83	1.25	11.58%
96.83	1.50	12.65%
96.83	1.75	13.65%
96.83	2.00	14.56%
96.83	2.25	15.42%
96.83	2.50	16.22%
96.83	2.75	16.99%
96.83	3.00	17.71%
96.83	3.25	18.40%
96.83	3.50	19.06%
96.83	3.75	19.69%
96.83	4.00	20.30%
96.83	4.25	20.89%
96.83	4.50	21.45%
96.83	4.75	22.00%
112.23	0.25	4.36%





These results show how the terminal stock price probability density of the CRR binomial tree converges to the continuous lognormal density:

Function used : CRR_Bin_Option_Price(S, K, T, sigma, rf, qd, n, Call_or_Put, Output_Format) Output_Format = 5

Function used : Est_Continuous_Prob(XY_range, n, Output_Format) Output_Format = 0

Function used : Log_Normal_Density(S0, ST, T, sigma, rf, qd)

CRR Binomial Tree:

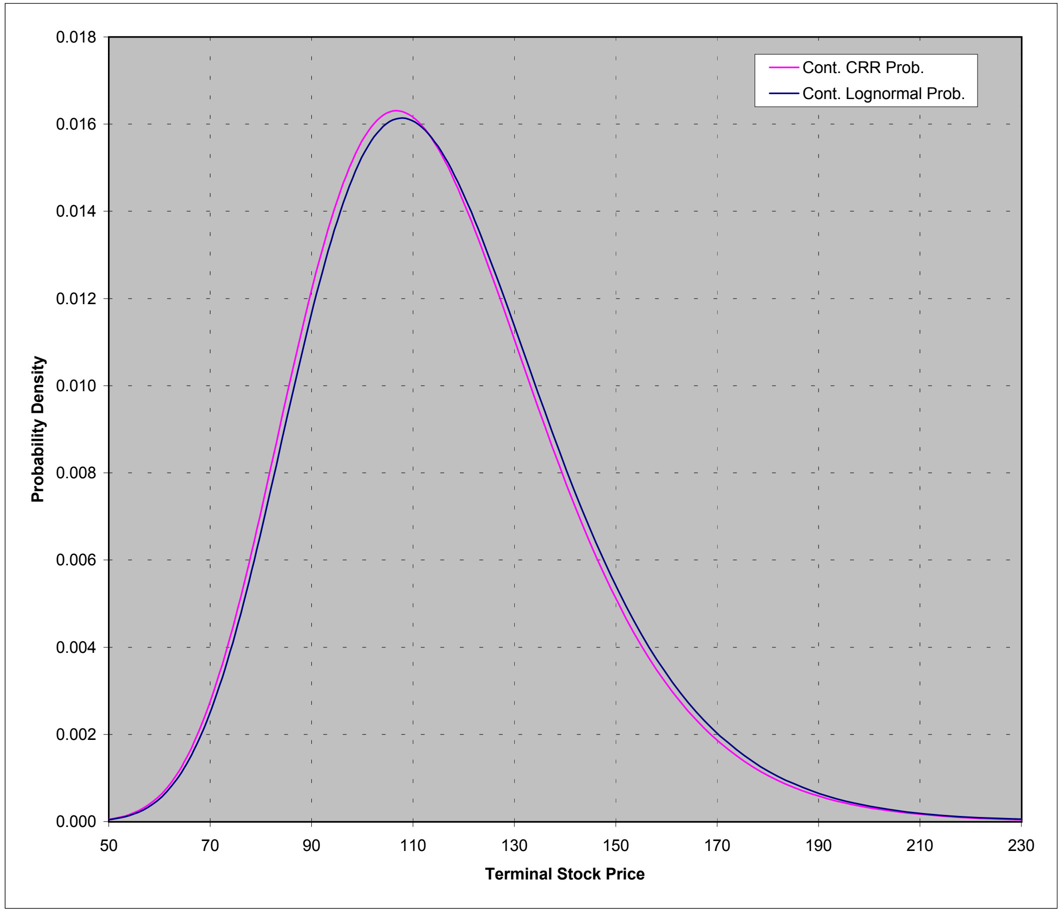
Terminal Node Stock Price	Discrete Cum. Distrib.	Discrete CRR Prob.
0.6738	0.0000	0.0000
0.6874	0.0000	0.0000
0.7013	0.0000	0.0000
0.7155	0.0000	0.0000
0.7299	0.0000	0.0000
0.7447	0.0000	0.0000
0.7597	0.0000	0.0000
0.7750	0.0000	0.0000
0.7907	0.0000	0.0000
0.8067	0.0000	0.0000
0.8230	0.0000	0.0000
0.8396	0.0000	0.0000
0.8566	0.0000	0.0000
0.8739	0.0000	0.0000
0.8915	0.0000	0.0000
0.9095	0.0000	0.0000
0.9279	0.0000	0.0000
0.9466	0.0000	0.0000
0.9658	0.0000	0.0000
0.9853	0.0000	0.0000
1.0052	0.0000	0.0000
1.0255	0.0000	0.0000
1.0462	0.0000	0.0000
1.0673	0.0000	0.0000
1.0889	0.0000	0.0000
1.1109	0.0000	0.0000
1.1333	0.0000	0.0000
1.1562	0.0000	0.0000
1.1796	0.0000	0.0000
1.2034	0.0000	0.0000
1.2277	0.0000	0.0000
1.2525	0.0000	0.0000
1.2778	0.0000	0.0000
1.3037	0.0000	0.0000
1.3300	0.0000	0.0000
1.3569	0.0000	0.0000
1.3843	0.0000	0.0000
1.4122	0.0000	0.0000
1.4408	0.0000	0.0000
1.4699	0.0000	0.0000
1.4996	0.0000	0.0000
1.5299	0.0000	0.0000
1.5608	0.0000	0.0000
1.5923	0.0000	0.0000

Cubic Spline Estimation:

Terminal Stock Price	Cum. Distrib.	Cont. CRR Prob.
0.6738	0.0000	0.0000
0.7332	0.0000	0.0000
0.9112	0.0000	0.0000
1.2081	0.0000	0.0000
1.6236	0.0000	0.0000
2.1579	0.0000	0.0000
2.8108	0.0000	0.0000
3.5826	0.0000	0.0000
4.4730	0.0000	0.0000
5.4822	0.0000	0.0000
6.6101	0.0000	0.0000
7.8567	0.0000	0.0000
9.2220	0.0000	0.0000
10.7061	0.0000	0.0000
12.3089	0.0000	0.0000
14.0304	0.0000	0.0000
15.8706	0.0000	0.0000
17.8296	0.0000	0.0000
19.9073	0.0000	0.0000
22.1037	0.0000	0.0000
24.4188	0.0000	0.0000
26.8527	0.0000	0.0000
29.4053	0.0000	0.0000
32.0766	0.0000	0.0000
34.8666	0.0000	0.0000
37.7754	0.0000	0.0000
40.8029	0.0000	0.0000
43.9491	0.0000	0.0000
47.2140	0.0001	0.0000
50.5977	0.0002	0.0001
54.1001	0.0005	0.0002
57.7212	0.0015	0.0004
61.4611	0.0035	0.0008
65.3196	0.0078	0.0015
69.2969	0.0156	0.0025
73.3929	0.0288	0.0040
77.6077	0.0496	0.0059
81.9412	0.0800	0.0081
86.3933	0.1212	0.0104
90.9643	0.1741	0.0126
95.6539	0.2378	0.0145
100.4623	0.3106	0.0157
105.3894	0.3897	0.0163
110.4352	0.4718	0.0161

Lognormal:

Cont. Lognormal Prob.
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0001
0.0001
0.0003
0.0007
0.0013
0.0023
0.0037
0.0055
0.0076
0.0099
0.0121
0.0140
0.0154
0.0161
0.0160



Notice that in the 20 step example on sheet DK_BC_Terminal_Prob_Densities, the difference between the CRR and the lognormal probabilities is greater as would be expected.

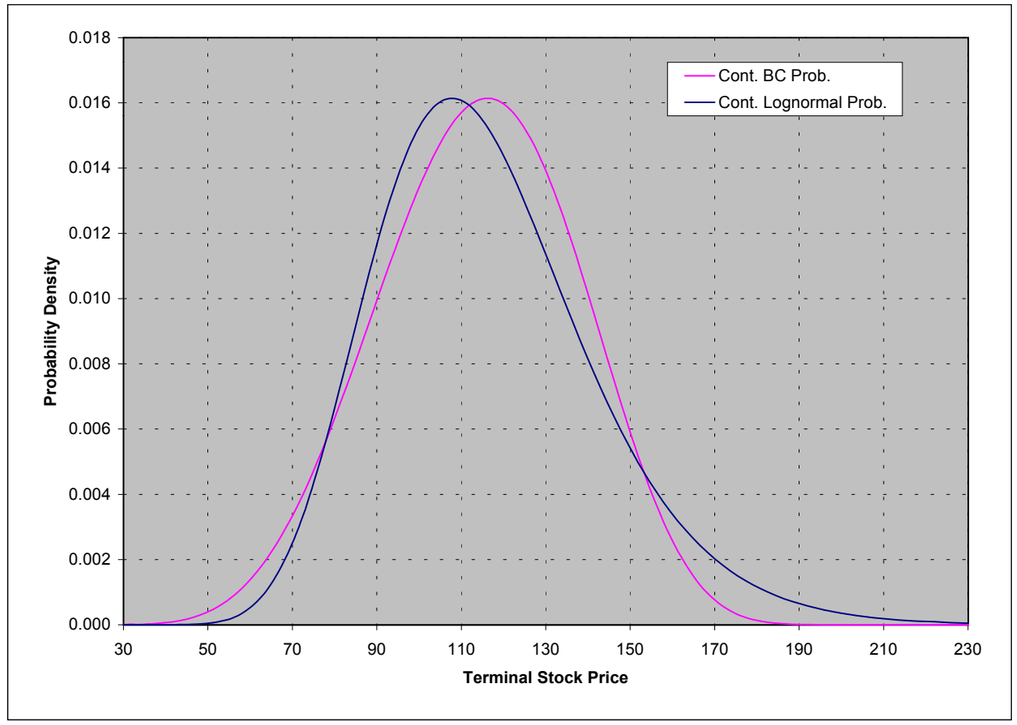
These results show an example of the estimated terminal probability density of a 100 step BC tree compared with the lognormal probability density:

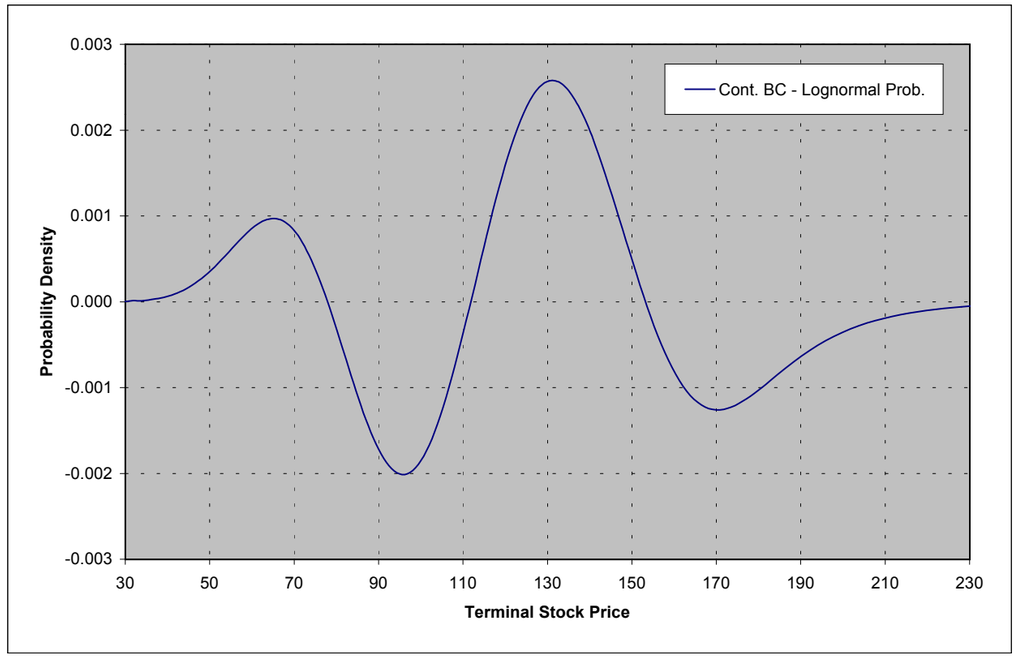
Function used : Implied_Bin_Option_Price(S, K, T, rf, qd, n, DK_or_CB, Call_or_Put, Output_Format) Output_Format = 6

Function used : Est_Continuous_Prob(XY_range, n, Output_Format) Output_Format = 0

Function used : Log_Normal_Density(S0, ST, T, sigma, rf, qd)

BC Implied Binomial Tree:			Cubic Spline Estimation:			Lognormal:	Difference:
Terminal Node Stock Price	Discrete Cum. Distrib.	Discrete BC Prob.	Terminal Stock Price	Cum. Distrib.	Cont. BC Prob.	Cont. Lognormal Prob.	Cont. BC - Lognormal Prob.
20.6340	0.0000	0.0000	20.6340	0.0000	0.0000	0.0000	0.0000
20.9109	0.0000	0.0000	20.6551	0.0000	0.0000	0.0000	0.0000
21.1916	0.0000	0.0000	20.7186	0.0000	0.0000	0.0000	0.0000
21.4761	0.0000	0.0000	20.8245	0.0000	0.0000	0.0000	0.0000
21.7644	0.0000	0.0000	20.9727	0.0000	0.0000	0.0000	0.0000
22.0566	0.0000	0.0000	21.1632	0.0000	0.0000	0.0000	0.0000
22.3526	0.0000	0.0000	21.3961	0.0000	0.0000	0.0000	0.0000
22.6527	0.0000	0.0000	21.6713	0.0000	0.0000	0.0000	0.0000
22.9568	0.0000	0.0000	21.9888	0.0000	0.0000	0.0000	0.0000
23.2649	0.0000	0.0000	22.3487	0.0000	0.0000	0.0000	0.0000
23.5772	0.0000	0.0000	22.7509	0.0000	0.0000	0.0000	0.0000
23.8937	0.0000	0.0000	23.1955	0.0000	0.0000	0.0000	0.0000
24.2145	0.0000	0.0000	23.6824	0.0000	0.0000	0.0000	0.0000
24.5395	0.0000	0.0000	24.2117	0.0000	0.0000	0.0000	0.0000
24.8689	0.0000	0.0000	24.7833	0.0000	0.0000	0.0000	0.0000
25.2028	0.0000	0.0000	25.3972	0.0000	0.0000	0.0000	0.0000
25.5414	0.0000	0.0000	26.0534	0.0000	0.0000	0.0000	0.0000
25.8849	0.0000	0.0000	26.7520	0.0000	0.0000	0.0000	0.0000
26.2344	0.0000	0.0000	27.4930	0.0000	0.0000	0.0000	0.0000
26.5915	0.0000	0.0000	28.2763	0.0000	0.0000	0.0000	0.0000
26.9602	0.0000	0.0000	29.1019	0.0000	0.0000	0.0000	0.0000
27.3478	0.0000	0.0000	29.9699	0.0000	0.0000	0.0000	0.0000
27.7668	0.0000	0.0000	30.8802	0.0000	0.0000	0.0000	0.0000
28.2370	0.0000	0.0000	31.8328	0.0000	0.0000	0.0000	0.0000
28.7858	0.0000	0.0000	32.8278	0.0000	0.0000	0.0000	0.0000
29.4471	0.0000	0.0000	33.8651	0.0000	0.0000	0.0000	0.0000
30.2580	0.0000	0.0000	34.9448	0.0001	0.0000	0.0000	0.0000
31.2547	0.0000	0.0000	36.0668	0.0001	0.0000	0.0000	0.0000
32.4709	0.0000	0.0000	37.2311	0.0001	0.0000	0.0000	0.0000
34.8444	0.0001	0.0000	38.4378	0.0002	0.0000	0.0000	0.0000
37.3917	0.0001	0.0001	39.6868	0.0002	0.0001	0.0000	0.0001
40.0987	0.0003	0.0001	40.9782	0.0003	0.0001	0.0000	0.0001
42.9697	0.0005	0.0003	42.3119	0.0004	0.0001	0.0000	0.0001
46.0075	0.0010	0.0005	43.6879	0.0006	0.0001	0.0000	0.0001
49.2134	0.0019	0.0009	45.1063	0.0008	0.0002	0.0000	0.0002
52.5871	0.0034	0.0015	46.5670	0.0011	0.0002	0.0000	0.0002
56.1259	0.0060	0.0026	48.0701	0.0015	0.0003	0.0000	0.0003
59.8249	0.0101	0.0041	49.6155	0.0020	0.0004	0.0000	0.0003
63.6768	0.0164	0.0064	51.2032	0.0027	0.0005	0.0001	0.0004
67.6715	0.0259	0.0094	52.8333	0.0036	0.0006	0.0001	0.0005
71.7964	0.0394	0.0135	54.5057	0.0047	0.0007	0.0002	0.0006
76.0365	0.0581	0.0187	56.2205	0.0061	0.0009	0.0002	0.0007
80.3746	0.0829	0.0249	57.9776	0.0078	0.0011	0.0003	0.0008
84.7915	0.1149	0.0319	59.7770	0.0100	0.0013	0.0005	0.0008
89.2669	0.1544	0.0396	61.6188	0.0128	0.0016	0.0007	0.0009
93.7796	0.2018	0.0474	63.5029	0.0161	0.0019	0.0010	0.0010
98.3084	0.2567	0.0549	65.4294	0.0202	0.0023	0.0013	0.0010
102.8326	0.3181	0.0614	67.3982	0.0251	0.0027	0.0018	0.0009
107.3320	0.3845	0.0664	69.4093	0.0311	0.0032	0.0023	0.0009
111.7878	0.4541	0.0696	71.4628	0.0382	0.0037	0.0030	0.0007
116.1834	0.5247	0.0706	73.5586	0.0465	0.0043	0.0037	0.0005
120.5039	0.5942	0.0694	75.6967	0.0564	0.0049	0.0046	0.0003
124.7362	0.6604	0.0662	77.8772	0.0679	0.0056	0.0056	0.0000
128.8702	0.7216	0.0612	80.1001	0.0812	0.0064	0.0067	-0.0003
132.8970	0.7765	0.0549	82.3652	0.0965	0.0071	0.0078	-0.0007
136.8099	0.8244	0.0479	84.6727	0.1139	0.0080	0.0090	-0.0010
140.6042	0.8649	0.0406	87.0226	0.1337	0.0088	0.0102	-0.0014
144.2772	0.8983	0.0334	89.4148	0.1559	0.0097	0.0114	-0.0017
147.8273	0.9251	0.0268	91.8493	0.1807	0.0106	0.0125	-0.0019
151.2543	0.9460	0.0209	94.3262	0.2081	0.0115	0.0135	-0.0020
154.5593	0.9619	0.0159	96.8454	0.2382	0.0124	0.0144	-0.0020
157.7442	0.9736	0.0118	99.4070	0.2710	0.0132	0.0151	-0.0019
160.8116	0.9822	0.0085	102.0108	0.3065	0.0140	0.0157	-0.0017
163.7647	0.9882	0.0060	104.6571	0.3444	0.0147	0.0160	-0.0013





The chart above is similar in form to that shown by Derman and Kani (1994), Page 14, Figure 7c.

Sheet: BC_100_Step_Example

Numerical Methods I
Project submission

Project description
There should be, on a separate sheet, a compact description of what the programme implements, together with a list of inputs to be provided by the user, and outputs that can be produced; give a literature reference to what has been implemented; mention against which data it was tested. Keep it short and crisp, presentation style, but do keep in mind that the summary has to be comprehensible not only to the first marker but also to others who may be less familiar with the topic.

File naming
The first part of the file names should be your **last name**. Following this, other file identification can be added, separated by underscore. There is absolutely no need for first names as in this FEQA group all last names are different. The reason for all this is that the work will be transcribed to a read-only CD on which files are arranged by name. The above specification keeps everything of the same author together, and in a convenient order.

Materials to be submitted
- ◊ each VBA module should have as top line the name of the workbook (as this is not printed automatically), and as second line the last name of the author; one of these is for use in marking; the other copy is for the exam archive; this is a security measure in case anything goes wrong with the electronic material
- ◊ submit all work on disk or on read-only CD
- ◊ in addition submit two printed copies of the VBA programme and of the relevant spreadsheets; to prevent line overflow when printing, VBA programmes are often best printed `Landscape` (while in VBA see `File Print Setup`); similarly arrange the spreadsheet page layout as `Portrait` or `Landscape` and arrange page breaks suitably; use Excel page `Custom Footer` to mark your name; put the workbook name as page `Header` (see menus `View | Page Break Preview` and `File | Page Setup`)
- ◊ all material must be labelled on the outside with your name or exam number
- ◊ do not submit any other paperwork than the copies mentioned above
- ◊ keep a copy of everything yourself

13
Implied Binomial Trees

as described in Härdle / Kleinow / Stahl *Applied Quantitative Finance* Chapter 7
Springer e-book; see www.xplore-stat.de

validate against results from programs on website.

Further reference:
James Option Theory Ch 8
Clewlow / Strikland *Implementing Derivatives Models* Ch 2.11

assigned to student: G. Rebel

THE ISMA CENTRE **THE UNIVERSITY OF READING**

SUBMISSION OF COURSEWORK AND RESEARCH PROJECTS

A fully completed and signed copy of this declaration must accompany all pieces of coursework and research projects submitted for assessment.

Essays must be handed in at the ISMA Centre Front Desk and a signed receipt will be given. You are advised to retain this receipt for future reference.

Students are reminded of the University's penalty for late submission of work:
a 10 out of 100 mark penalty deduction for work submitted between one and seven calendar days late, after which zero will be awarded. Extensions must be applied for on the relevant Extensions form and remissions must be applied for on the Extenuating Circumstances form. Both forms are available from the Department and also the Faculty website:

www.rdg.ac.uk/fess/students/undergraduate_and_postgraduate.htm

Student's name	Gerhard Rebel
Degree programme	ISMA MSc ISIB
Year	Masters
Type of work	Numerical Methods 1 Project
Title of module	Numerical Methods 1
Name of lecturer/tutor	Ubbo Wiersema
Submission deadline	19 April 2004
This piece of work was undertaken	Independently

Plagiarism: Plagiarism is the misrepresentation of the work of others as one's own (including ideas, arguments, words, diagrams, images or data). It includes the explicit claim that another's work is one's own and, no less seriously, the failure to acknowledge adequately the sources used. This applies whatever the source of the material (for example, a published source, the World Wide Web, a verbal communication, or the work of another student). Plagiarism is a form of academic misconduct and will be penalised accordingly.

Declaration: "I certify that this is my own work and use of material from other sources has been properly and fully acknowledged in the text. I have read the definition of plagiarism given above and the Department's advice on good academic practice contained in the Programme Handbook. I understand that the consequences of committing plagiarism, if proven and in the absence of mitigating circumstances, may include failure in the Year or Part of my programme or removal from membership of the University. I also certify that neither this piece of work, nor any part of it, has been submitted in connection with another assessment."

Student's signature: 	Date submitted **19 April 2004**

Please note that you should retain all your coursework as it may be required for inspection by the External Examiner at the time of the Finals.