<u>Comparison of the Cox-Ross-Rubinstein Model (CRR) with the Black-Scholes Model for finding the value of a European Call Option on a stock that pays dividends</u>

By

Fredrik Jonsäll

Oliver Grace

Michael Aussieker

Abstract

The ordinary Black-Scholes Universe unfortunately doesn't handle absolute stock dividends in a fully consistent way. This can be observed e.g. in the case when there are two similar American Call Options, same underlying stock and same strike price, but one expiring just <u>before</u> a dividend and one expiring just <u>after</u> the dividend: Often, the theoretical price, yielded by a standard "Black-Scholes Universe model", is higher for the option expiring before the dividend, which of course is unreasonable when the options are American.

Introduction

This seminar explores the implications of single dividend payment and how that impacts the price of the European call. In general, a company has a schedule of dividend payments; however, we are only focusing on a single event. This is an important event to analyse, because the returns on a stock are impacted by the payment of dividends; specifically the price of the stock is lowered by the dividend amount on the ex-dividend date. More specifically, a stock with many dividends behaves at all times like a stock with no ex-dividend dates whose value is the spot price minus the present value of all future dividend payments (within the relevant period of time). We looked at a couple of different methods of adjusting the standard Black Scholes Model once relaxing the dividend assumption. Then we altered the inputs to try and find an appropriate solution to compare this to the standard Black Scholes value along with a Cox –Ross-Rubinstein dividend impacted binomial tree.

Solutions

<u>Solution 1 – Volatility Adjustment</u>

In this solution we made an adjustment of the volatility to account for the reduction in the real stock price so that we can implement the Black-Scholes model once a dividend payment has been declared. This approach results in increasing the overall volatility by a factor based on the present value of the dividend. Once the model was adjusted we fine-tuned the model to try to optimize the results compared to the Standard Black Scholes pricing. Overall, the model primarily depended on the time difference between the option's expiry date and the ex-dividend date. We also noticed larger differences when the option had greater value. This method seems feasible for a small dividend and when the option is at the money or in the money. This method also seems plausible for lower value stocks.

<u>Solution 2 – Weighted Volatility Adjustment</u>

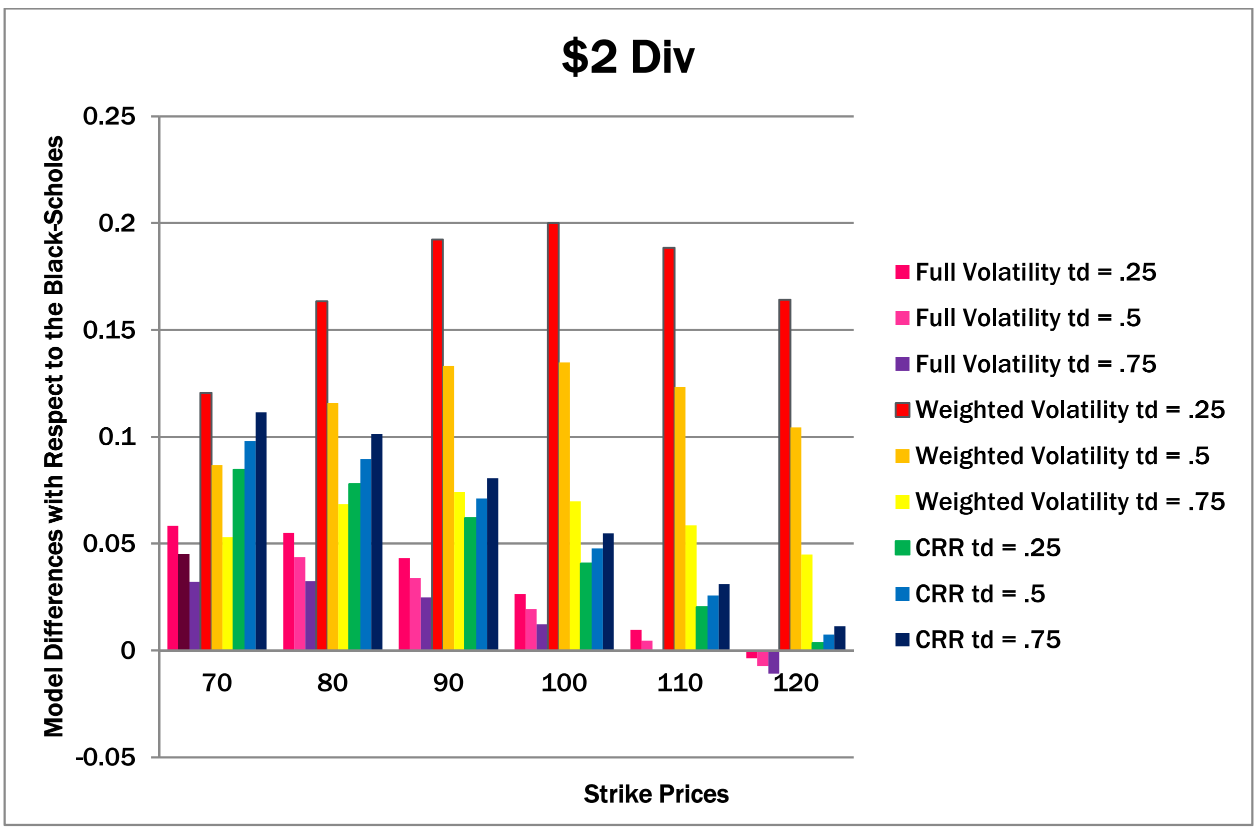
In this solution, the adjusted volatility had a multiplier like the Full Volatility however it also took into account when the payment was made during the life of the option. Once the adjusted volatility was entered into BSM model we tried to refine the results to seek a closer approximation. The interpretation of the multiplier is that the further the dividend payment date to expiry of the option, the less of an impact it has on the adjusted volatility, resulting in larger differences in the overall pricing. The implementation of this method results in large differences and does seem feasible.

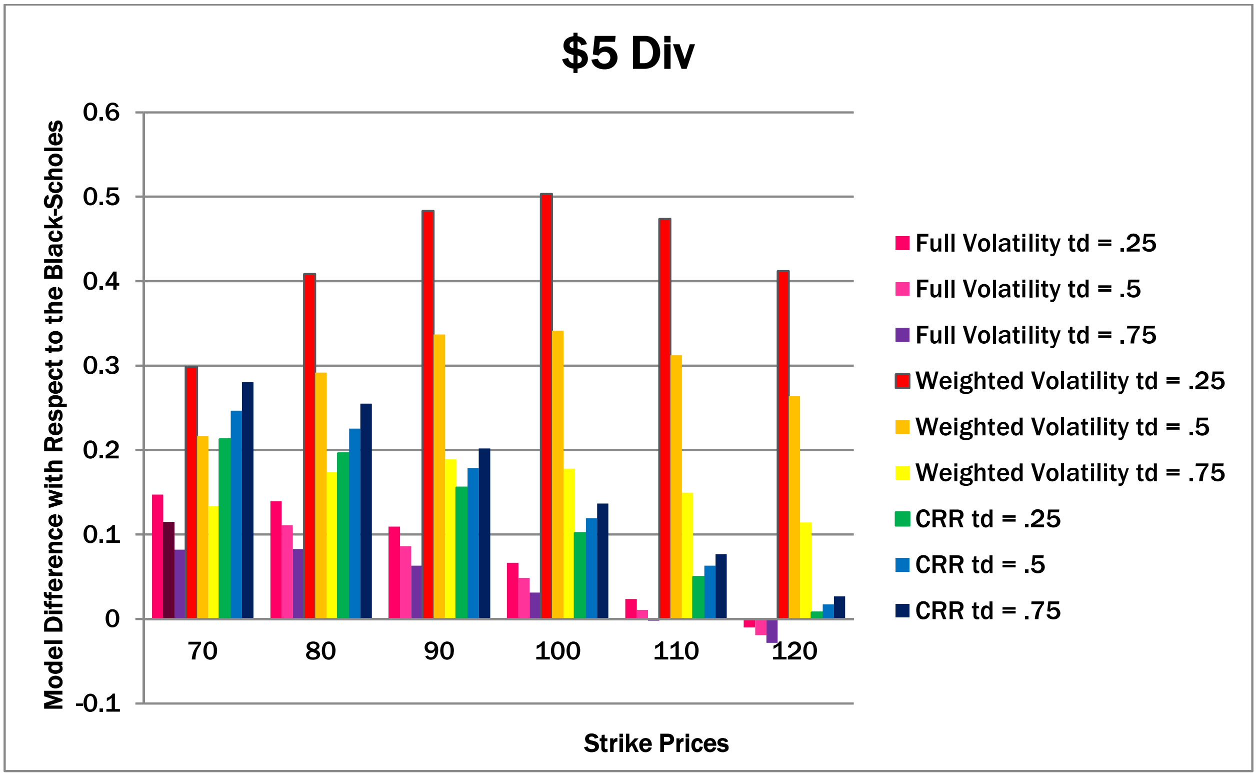
The Dividend adjusted CRR model for European call was calculated with constant volatility, where the dividend amount is known in advance, with n=500 and with a single dividend payment at first, second, and, third quarter respectively. We used a method in which the prior result from the payment date does not recombine with after. We used this as a standard discounted base case for comparison with the standard CRR and were investigating whether there would be convergence to the BSM. However we could not find convergence to standard BSM with the dividend adjusted CRR even after 10,000 steps but there was improvement in the data.

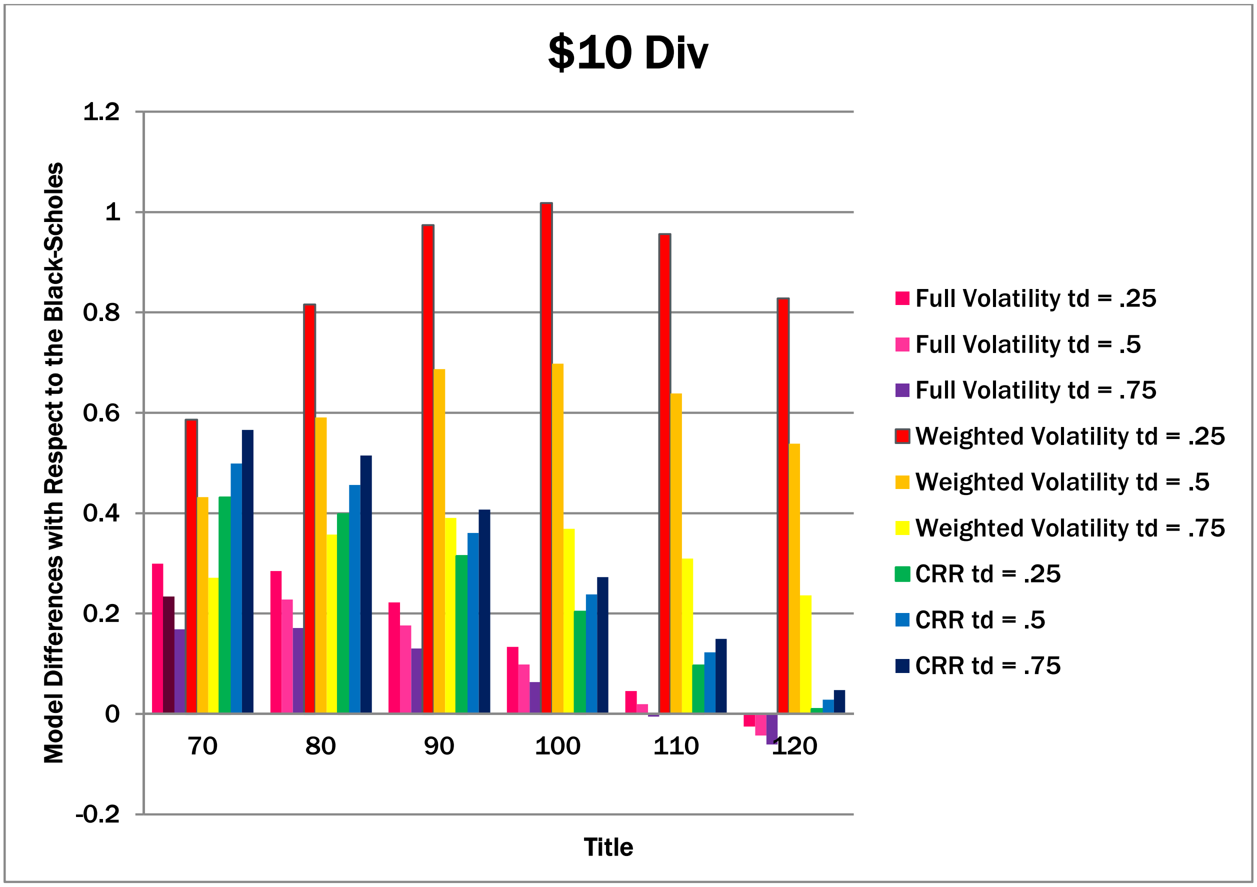
Methodology

The Full Volatility Adjustment, Weighted Volatility Adjustment and Standard Black-Scholes Model were implemented in the R language while the CRR adjusted for dividend payments was implemented in the python language. For all models, we simulated 6 strike prices. Each simulation only allowed for one divided payment per year and the models were simulated with 3 payment dates (1/4, 1/2/ 3/4 of a year). The differences with their respected counterparts were then graphed.

Results




$5 Div
Model Difference with Respect to the Black-Scholes
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
70
80
90
100
110
120
Strike Prices
Full Volatility td = .25
Full Volatility td = .5
Full Volatility td = .75
Weighted Volatility td = .25
Weighted Volatility td = .5
Weighted Volatility td = .75
CRR td = .25
CRR td = .5
CRR td = .75



Conclusions

- A Full volatility adjustment better approximates the BS than the Weighted volatility adjustment or CRR.
- For the Black Scholes adjusted models the difference is a (negative slope) linear function of dividend time. This dependence is greater for the Weighted model, than the Full volatility implementation.
- The larger the dividend, the greater the impact on pricing.
- Out of money options, the dividend has greater impact on pricing differences.
- In the money options, results in less impact for the Full volatility adjusted model.
- Weighted model is not a viable option when accounting for dividends.
- The Full volatility model is feasible when the underlying pays small dividends and the option is at or in the money.
- Pricing improves for all models as the value of the underlying falls.

Source Codes

Python

CRR.py

```
'''
Created on October 11, 2013
Written in Python and makes use of the NumPy v1.7.1 package
It is an implementation of the Cox-Ross-Rubinstein numerical method

NumPy is the fundamental package for scientific computing with Python. It contains
among other things:

    a powerful N-dimensional array object
    sophisticated (broadcasting) functions
    tools for integrating C/C++ and Fortran code
    useful linear algebra, Fourier transform, and random number capabilities

See www.numpy.org

The code is inspired from:
Binomial Tree for America and European options by Mehdi Bounouar

@author: Oliver Grace
@group team members: Michael Aussieker, Fredrik Jonsall
'''


import numpy as np


def BinomialTreeCRR(n, Spot, k, r, v, T, D, td):
    """
    n: steps
    Spot: Spot price
    K: Strike price
    r: Risk free rate
    v: volatility
    T: Maturity
    td: Dividend payment date
    """
    D0 = D*np.exp(r*td)
    true_sigma = v*(Spot/(Spot - D0))

    dt = T/n
    u = np.exp(true_sigma*np.sqrt(dt))
    d = 1./u
    p = (np.exp(r*dt)-d)/(u-d)

    #Binomial price tree
    stkval = np.zeros((n+1,n+1))
    stkval[0,0] = Spot - D0
```

```
    for i in range(1,n+1):
        stkval[i,0] = stkval[i-1,0]*u
        for j in range(1,i+1):
            stkval[i,j] = stkval[i-1,j-1]*d

    #option value at each final node
    optval = np.zeros((n+1,n+1))
    for j in xrange(n+1):
        optval[n,j] = max(0, stkval[n,j]-k+D)

    #backward recursion for option price
    for i in xrange(n-1,-1,-1):
        for j in xrange(i+1):
            optval[i,j] = max(stkval[i,j]-k+D, np.exp(-r*dt)*(p*optval[i+1,j]+(1-
p)*optval[i+1,j+1]))
    return optval[0,0]


if __name__ == "__main__":
    Spot = 100.            # Spot Price
    k = 100.               # Strike Price
    r = .03                # Annual Risk-free rate
    v = .3                 # Annual Volatility
    T = 1.0                # Time in year (days/365)
    n = 500                # Number of steps
    D = 10.0               # Dividend per share
    td = .5                # Time to Dividend Payment

    print "European Call on Dividend Paying Stock: %s"%(BinomialTreeCRR(n, Spot, k,
r, v, T, D, td))
```

R

```
# Define a function BSWeightedAdj and then run with the inputs.

# Author: Michael Aussieker

# Group Members: Oliver Grace, Fredrik Jonsall

BSWeightedAdj = function(S, X, rf, T,td, sigma,D0) {

    values <- matrix(,3,2)

    Dd<-(D0*exp(-rf*td))    #discounted dividend from Ex date assume Ex and pay date
is same#

    Sd<-(S-Dd)      #Stock adjusted for present value of dividends#

    Xd1=(X-D0)      #Strike adjusted after dividend paid#

    K=td/T      #dividend payment to terminal date impact#

    SigA=(sigma*S)/(Sd)     #adjusted volatility full#

    WSig=(sigma*S)/(S-(Dd*K)) #volatility adjustment impact of payment date#
```

```
    d1 <- (log(Sd/Xd1)+(rf*T))/(SigA*sqrt(T)) +((SigA*sqrt(T))/2)

    d2 <- d1 - (SigA * sqrt(T))

    d3 <- (log(Sd/Xd1)+(rf*T))/(WSig*sqrt(T)) +((WSig*sqrt(T))/2)

    d4 <- d3 - (WSig * sqrt(T))

    d5 <- (log(S/X)+(rf*T))/(sigma*sqrt(T)) +((sigma*sqrt(T))/2)

    d6 <- d5 - (sigma * sqrt(T))


    values[1] <- (Sd)*pnorm(d1) - (Xd1)*exp(-rf*T)*pnorm(d2)

    values[4] <- (Xd1)*exp(-rf*T)*pnorm(-d2)-(Sd)*pnorm(-d1)

    values[2] <- (Sd)*pnorm(d3) - (Xd1)*exp(-rf*T)*pnorm(d4)

    values[5] <- (Xd1)*exp(-rf*T)*pnorm(-d4)-(Sd)*pnorm(-d3)

    values[3] <- (S)*pnorm(d5) - (X)*exp(-rf*T)*pnorm(d6)

    values[6] <- (X)*exp(-rf*T)*pnorm(-d6)-(S)*pnorm(-d5)


    print("Calls COL=1, Puts COL=2, Full Vol ADJ ROW=1, Weighted Vol ROW=2, Std BSM
ROW=3")

    values }
```

References

http://www.volopta.com/ComputerCode/Python_Other/BinomialTree_CRR_py.txt

Options, Futures and Other Derivatives, Eight Edition 2012 by John C. Hull

Lecture notes in Analytical Finance I, August 29, 2013, by Jan R. M. Roman

Black-Scholes and Beyond: Option Pricing Models, September 1, 1996 by Neil A. Chriss