4 Calculate Convertible Bond Price by Binomial Tree Method

declaration*

04/28/2021

Contents

1 Introduction

Convertible bond is a kind of company issued bond. This kind of bond's holder has the right to convert the bond to some amount of the company's stocks before expiration date, the amount of stocks can be converted by one unit of bond is conversion ratio. At the meantime, usually this kind of bond can be called back by some price by the company before expiration date, the call back price for one unit of the bond is callback price. If the company will call back the bond, the bond's holder can choose to convert the bond into stocks, i.e. the conversion's priority is higher than being called back.

When calculating convertible bond's price, we can not neglect the possibility that the company may default. A relatively simple way of calculating convertible bond's price, is assuming the stock price changing obeys geometrical Brownian motion, can be described by a tree, and at each Δt the company has probability $1 - e^{-\lambda \Delta t}$ to default. Where λ is the risk-free default probability density. If the company has defaulted, then at this moment the convertible bond's price is the bond's par value times recycle ratio.

2 Procedures of Calculating Convertible Bond Price

1. From given parameters build a binomial tree for company's stock price changing. Where the branching factors are (because of the possibility to default, they are different from ordinary binomial tree):

$$u = e^{\sqrt{(\sigma^2 - \lambda)\Delta t}}, \quad d = \frac{1}{u}, \quad p_u = \frac{e^{r\Delta t} - de^{-\lambda \Delta t}}{u - d}, \quad p_d = \frac{ue^{-\lambda \Delta t} - e^{r\Delta t}}{u - d} \,. \tag{1}$$

2. Calculate leaf level every node's stock price corresponded convertible bond's price $B_{i,j}$, which is $B_{i,j} = \max(S_{i,j} \times \text{conversion ratio, par value})$. Where i is level number, j is the index of position at current level (from low stock price to high stock price).

3. Calculate previous level's every node's stock price corresponded convertible bond's price. First calculate the expected price of the bond, ignoring the bond can be converted or called back,

$$B''_{i,j} = e^{-r\Delta t} \left[p_u B_{i+1,j+1} + p_d B_{i+1,j} + p_{default} \times \text{par value} \times \text{recycle ratio} \right], \quad p_{default} = 1 - p_u - p_d \,. \tag{2}$$

If the price is higher than the callback price, then the company will call back the bond, so

$$B'_{i,j} = \min\left(B''_{i,j}, \text{ callback price}\right). \tag{3}$$

The bond holder can choose to convert the bond to stocks before the bond being called back, so at this node the final price of the bond is:

$$B_{i,j} = \max\left(B'_{i,j}, \text{ conversion ratio} \times S_{i,j}\right) = \max\left[\min\left(B''_{i,j}, \text{ callback price}\right), \text{ conversion ratio} \times S_{i,j}\right]. \tag{4}$$

*The author is not responsible for the correctness of the contents. If you would like to use part of the contents in your presentation, article, etc., please add the website "Quantitative Finance Numerics" as a reference.

4. Repeat procedure 3, until get the convertible bond's price at the root node.

3 Procedures Implementation with Python

```
import numpy as np
E = np.e
class Tree_convertible_bond:
    def __init__(self, r, sigma, S_0, T, lbd, conversion_ratio, callback_price, par_value,
                 recycle_ratio, steps):
        self.r = r
        self.sigma = sigma
        self.S_0 = S_0
        self.T = T
        self.lbd = lbd

        self.conversion_ratio = conversion_ratio
        self.callback_price = callback_price
        self.par_value = par_value
        self.recycle_ratio = recycle_ratio
        self.steps = steps

        self.dt = self.T/self.steps
        self.u = E**(((self.sigma*self.sigma-self.lbd)*self.dt)**0.5)
        self.d = 1/self.u

        self.p_u = (E**(self.r*self.dt)-self.d*E**(-self.lbd*self.dt))/(self.u-self.d)
        self.p_d = (self.u*E**(-self.lbd*self.dt)-E**(self.r*self.dt))/(self.u-self.d)
        self.p_default = 1-self.p_u-self.p_d

        self.bond_price = None
        self.tree = None
        self.build_tree()

    def build_tree(self):
        self.tree = list()
        for lvl in range(self.steps+1):
            row = list()
            for j in range(lvl+1):
                node = dict()
                node["S"] = self.S_0*(self.u**j)*(self.d**(lvl-j))
                node["B"] = None
                row.append(node)
            self.tree.append(row)
        return

    def calculate_bond_price(self):
        # Shorten names.
        tree = self.tree
        r, steps =self.r, self.steps
        conversion_ratio, callback_price = self.conversion_ratio, self.callback_price
        recycle_ratio, par_value = self.recycle_ratio, self.par_value
        dt, u, d = self.dt, self.u, self.d
        p_u, p_d, p_default = self.p_u, self.p_d, self.p_default

        # Discount factor.
        a = E**(-r*dt)

        # Boundary condition.
        for node in tree[-1]:
            node["B"] = max(par_value, node["S"]*conversion_ratio)
        for lvl in range(steps-1, -1, -1):
```

```
        for j in range(lvl+1):
            # Backwards induction from next level.
            tree[lvl][j]["B"] = a*p_u*tree[lvl+1][j+1]["B"]+a*p_d*tree[lvl+1][j]["B"]
            tree[lvl][j]["B"] += a*p_default*par_value*recycle_ratio
            # Determine whether the bond will be called back.
            tree[lvl][j]["B"] = min(tree[lvl][j]["B"], callback_price)
            # Determine whether the bond should be converted.
            tree[lvl][j]["B"] = max(tree[lvl][j]["B"], conversion_ratio*tree[lvl][j]["S"])

    self.bond_price = tree[0][0]["B"]
    return
```

4 Calculation Example

Suppose risk-free rate is 0.05, the volatility of company stock price is 0.3, initial stock price is 50. Consider a convertible bond which expires 9 months later, conversion ratio is 2 (1 unite bond can be converted to 2 stocks by the bond holder), the company can call back the bond with price 113, the bond's par value is 100. And suppose the company has probability 0.01 to default every year, the recycle ratio is 0.4, i.e. price after default is 40.

For this case, we use a 10-step binomial tree to calculate the convertible bond's price, detailed result is the following:

```
tree_obj = Tree_convertible_bond(0.05, 0.3, 50, 0.75, 0.01, 2, 113, 100, 0.4, 10)
tree_obj.calculate_bond_price()
bond_price = tree_obj.bond_price

print("Convertible bond price: {0:.5f}".format(bond_price))

Output:
    Convertible bond price: 106.61156 .
```

5 Reference

References

[1] John C. Hull. chapter 27 of *Options, Futures and Other Derivatives*, 9th Edition, Pearson Education.