Bonds' Yield To Maturity with Python

Bond's yield is an important in investment decision-making.

Below I provide 3 methods of computing the YTM with Python.

Let's start with a simple example:

What amount should an investor be willing to pay for a USD1,000, 5-year bond which pays USD50 interest semiannually and is sold to yield 8 %?

The bond value is calculated using the Bullet Bond Class b1 defined in my notebook @ https://github.com/kyramichel/Math_Finance/blob/master/Coupon%20Bonds1.pdf (https://github.com/kyramichel/Math_Finance/blob/master/Coupon%20Bonds1.pdf)

In [1]:
```
import b1
price = b1.BulletBond(10,1000,0.05).Price(0.04)
price
```

Out[1]:
```
1081.1089577935504
```

NOTE:

- There are 10 (=5x2) periods
- The fixed coupon rate is 5% (=50/1000)
- The semiannual interest is 4% (=8/2)

The bond's yield is 8%. The yield to maturity (YTM) reflects the *annual* return that an investor will earn on a bond, if the investor purchases the bond today and holds it until maturity.

Expected Rate of Return on a Bond: Yield to Maturity

The expected rate of return on a bond, known as the bond's yield to maturity (YTM) or the yield to redemption is the internal rate of return (IRR) on a bond's expected cash flows.

If the bond's price is known, then YTM is the discount rate that equates the present value (PV) of the bond's expected cash flows until maturity with the bond's price. That is, the yield-to-maturity is the solution for the rate, r, in the bond valuation model:

Price (bond) = PV (expected cash flow)

or, equivalent

NPV = 0

Below is the formula for calculating bond's price using the time-value-of-money calculation to obtain the present value (PV) for a market discount rate:

Calculating Yield to Maturity:

There are several ways to find YTM.

Method 1: Trial & error:

First, one must recall the relationship between: Price - Yield to Maturity

As shown in https://github.com/kyramichel/Math_Finance/blob/master/Coupon%20Bonds1.pdf (https://github.com/kyramichel/Math_Finance/blob/master/Coupon%20Bonds1.pdf) the price of a fixed-rate security depends on the relationship between its yield to maturity (r) and the interest rate (c). That is:

- if the yield to maturity is greater than the interest rate: the price will be less than par value
- if the the yield to maturity is equal to the interest rate: the price will be equal to par;
- if the yield to maturity is less than the interest rate, the price will be greater than par.

Example 2:

Suppose an investor is offered a 10-year, 8% coupon, USD 1,000 par value bond at a market price of USD877.07. What rate of return could the investor earn if buys the bond & holds it to maturity? In other words, what is the investor's expected rate of return or the YTM?

To find YTM, notice that since the bond is selling at a discount (bond's market price is less than the par value: 877.07 < 1000), the bond's yield is above the coupon rate of 8%.

By trying & error, we can try substitution a rate of 9 percent in the bond valuation model:

In [2]:
```
price = b1.BulletBond(10,1000, 0.08).Price(0.09)
price
```

Out[2]:
```
935.8234229884099
```

The calculated bond value is above the actual market price (935.82 > 877.07), it means that the yield is not 9 percent. To lower the calculated bond value, the rate must be raised.

Trying 10 percent it gives:

In [3]:
```
price = b1.BulletBond(10,1000, 0.08).Price(0.10)
price
```

Out[3]:
```
877.1086578859063
```

Since this calculated value is exactly the market price of the bond:

YTM = 10%

Method 2:

The following formula can be used to approximate yield to maturity on a bond:

$$\text{YTM} = \frac{C+(M-Price)/n}{(M+V)/2}$$

In [4]:
```
%%writefile y1.py

import math

class BulletBond:
    def __init__(self, n, M, c):
        self.n = n
        self.M = M
        self.c = c

    def Price(self, r):
        return self.c * self.M *(1 - math.pow(1+r,-self.n))/r + self.M * math

    def YTM(self, n, price):
        return (2* self.c * self.M + 2*(self.M - price)/self.n)/(self.M + pri
```

```
Overwriting y1.py
```

In [5]:
```
import y1
```

Based on the same data from Example 2 and using the formula for approximating the rate of return on the bond:

In [6]:
```
ytm = y1.BulletBond(10,1000, 0.08).YTM(10, 877.60)
ytm
```

Out[6]:
```
0.09825308904985088
```

The approximated value 9.8% is indeed very close to the exact rate of 10%.

Method 3

Using numeric approximation.

Based on the same data from Example 2 and using scipy optimize.newton:

In [10]:
```
import math
YTM = lambda r: 80 *(1 - math.pow(1+r,-10))/r + 1000 * math.pow(1+r,-10) - 87
```

In [11]:
```
import scipy
from scipy import optimize
optimize.newton(YTM, 0.05)
```

Out[11]:
```
0.10000011713160618
```

The calculated value is exactly YTM = 10%