Parallelizing Yield to Maturity Calculations for a Portfolio of Bonds

Anjali Smith (as6467)

November 25, 2022

Overview

Similar to Simon Marlow's parallel implementation of a sudoku solver that solves 49,000 puzzles, my project will parallelize yield to maturity (YTM) calculations for a portfolio of bonds with semi-annual coupon payments. I plan to incrementally speed up my program by first running the calculation for each bond in parallel, and then introducing parallelism within the YTM calculation. In order to calculate the YTM, I plan to use the Newton-Raphson function in the Numeric.RootFinding module.

Background

For context, bonds are fixed income securities that an investor can purchase from either the government, a corporation, a municipality, or an agency. When an investor purchases a bond, they are essentially lending an amount of money that they know will be paid back to them on the bond's maturity date. Bonds with semi-annual coupon payments, which are the type of bonds my program will be working with, pay the investor a fixed amount of money every six months at a fixed interest rate. Investors can choose to keep their bond until it matures or sell it to another investor. The yield to maturity (YTM) of a bond is the rate of return that the investor earns if they keep their bond until it matures. To calculate this value, we assume that "all coupon payments are reinvested at the same rate as the bond's current yield" and we "take into account the bond's current market price, par value, coupon interest rate, and term to maturity" (Investopedia). Yield to maturity calculations are useful to investors because it allows them to decide whether or not a bond is worth purchasing.

Calculating Yield to Maturity:

The following equation is the bond's price in terms of its yield to maturity:

$$Bond\ Price\ =\ \frac{coupon}{(1+YTM)^1} + \frac{coupon}{(1+YTM)^2} + \frac{coupon}{(1+YTM)^3} + \ldots + \frac{coupon}{(1+YTM)^n}$$

where n equals the number of compounding periods.

In order to solve for the bond's yield to maturity, we rearrange the terms and substitute YTM with the variable x to give us a function $f(x)$:

$$f(x)\ =\ \frac{coupon}{(1+x)^1} + \frac{coupon}{(1+x)^2} + \frac{coupon}{(1+x)^3} + \ldots + \frac{coupon}{(1+x)^n}\ -\ bond\ price\ =\ 0$$

In order to solve for x, we can use the Newton-Raphson method. The Newton-Raphson method is an algorithm used to solve for the roots of a function. In other words, for a given differentiable function f, we want to find x such that:

$$f(x)\ =\ 0$$

Given the information that the root of the function, x, is close to the value x_0, we can use the Newton-Raphson method to find a better estimate of the root with the formula:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

This process of finding a better estimation is repeated until the best estimate is found, which is decided by an error tolerance value at the beginning of the execution of the algorithm.

In Haskell, the Numeric.RootFinding module contains a Newton Raphson function which takes 3 parameters, the first being a data type called NewtonParam which contains a maximum number of iterations for the algorithm and an error tolerance for the root approximation. The second argument to the function is a triple containing a lower bound, an initial guess, and an upper bound for the approximations. The third argument to

the function is a user-supplied function that takes a bond's yield and returns a tuple with its corresponding price and the first derivative of the price.

Objectives

I plan to incrementally speed up the YTM calculations for a portfolio of bonds by first running all calculations in parallel and then by introducing parallelism in the YTM calculation. In order to run the Newton Raphson algorithm, I need to calculate the function f and its derivative, as shown in the example in the above section of my proposal. In order to improve performance, I plan to parallelize the calculation of f.

References

https://www.forbes.com/advisor/investing/what-is-a-bond/

https://www.investopedia.com/terms/y/yieldtomaturity.asp

https://hackage.haskell.org/package/math-functions-0.3.4.2/docs/Numeric-RootFinding.html

https://medium.datadriveninvestor.com/calculate-bond-yields-using-newtons-method-fe0ecf88c293

https://brilliant.org/wiki/newton-raphson-method/#:~:text=The%20Newton%2DRaphson%20method%20(also,straight%20line%20tangent%20to%20it.