Week8 Assignment

Objective

This assignment focuses on developing a bond pricing engine by working with interpolating yield curves, and discounting bond cash flows. You will gain insights into how yield curves influence bond prices and how bond-specific attributes like coupons and maturity are incorporated into pricing. The assignment consists of two main parts.

Part 1: Yield Curve Construction and Interpolation

A yield curve is a fundamental tool in fixed income analysis, representing the relationship between bond yields and their respective maturities. This assignment focuses on constructing a yield curve object from given market data and implementing different interpolation methods.

Implementation:

Task 1.1: Load the Data

The yield curve data is read from an external source, such as Bloomberg, Excel file, etc. In our assignment, the yield curve as of 1/3/2024 can be found in the Excel file S490.xlsx. The file contains maturities (tenors) and their corresponding interest rates. For example:

Tenor	Rate (%)
$1M$	5.341
$2M$	5.347
$3M$	5.327

In the code, the first step would be reading the data from Excel. Note that we have Tenor values in various formats (e.g., "3M", "2Y", "6M", etc.), and we need to convert them into a numerical representation in years. To do this, you will write a Python function convert_duration(tenor: str) -> float, to convert financial tenor values into years.

Specifically,

- Extract the numeric part from a given tenor string (e.g., "3" from "3M", "2" from "2Y").
- Identify the unit: "M" represents months, "Y" represents years, etc.

- Convert months into a fraction of years: Example: "6M" should be converted to 0.5Y (since 6 months = 0.5 years). "2Y" should remain 2.0 years.
- The function should accept a string (e.g., "3M", "1Y", "10Y") and return a floating number representing the duration in years. If the input format is invalid, the function should raise a ValueError.
- Use regular expressions to extract the numeric value and unit.

Task 1.2: Create YieldCurve object

You need to create a YieldCurve class that:

- Takes market data (tenors and rates) as input.
- Implements three interpolation methods:
    - Linear Interpolation (default method)
    - Cubic Spline
    - Nelson-Siegel Model
- Provides a function to retrieve interest rates for any given tenor.

Step 1.2.1: Initialize the Yield Curve Object

- Define an __init__ method that:
- Accepts a DataFrame containing tenor (years) and rate (yield in %).
- Sorts the data by tenor.
- Allows the user to specify the interpolation method (linear, cubic_spline, nelson_siegel).
- Initializes the interpolation function accordingly.

Step 1.2.2: Implement the Interpolation Methods

Note 1.2.2.1: Linear Interpolation:

Linear interpolation is a straightforward method to estimate values for missing maturities between known data points. If the yield curve provides rates at t1 and t2, the rate at an intermediate maturity t is calculated as:

$$\text{Rate}(t) = \text{Rate}(t_1) + (\text{Rate}(t_2) - \text{Rate}(t_1)) \times \frac{t - t_1}{t_2 - t_1}$$

This method assumes that rates change linearly between known points.

Note 1.2.2.2: Cubic Spline:

A cubic spline is a piecewise cubic polynomial used to interpolate a smooth curve between given data points. In yield curve modeling, cubic splines help estimate missing yield values while maintaining smooth transitions.

Given a set of $n+1$ data points:

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$$

we define a set of cubic polynomials $S_i(x)$, one for each interval $[x_i, x_{i+1}]$, of the form:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad x \in [x_i, x_{i+1}]$$

Each polynomial $S_i(x)$ should satisfy continuity conditions, ensuring smooth transitions between the segments.

Note 1.2.2.3: Nelson-Siegel Model

The Nelson-Siegel model is an advanced technique to fit a smooth yield curve by capturing its level, slope, and curvature. The formula is:

$$\text{Yield}(t) = \beta_0 + \beta_1 \frac{1 - e^{-t/\tau}}{t/\tau} + \beta_2 \left(\frac{1 - e^{-t/\tau}}{t/\tau} - e^{-t/\tau} \right)$$

- β_0: Represents the long-term level of the curve.
- β_1: Reflects the short-term slope.
- β_2: Captures the curvature of the yield curve.
- τ: Controls the exponential decay.

This method is highly flexible and provides a smooth approximation of the yield curve.

This method is highly flexible and provides a smooth approximation of the yield curve.

Step 1.2.3: Implement the get_rate(tenor) Function

This function should:

- Take a tenor (time to maturity in years), and an interpolation method (linear, cubic_spline, or nelson_siegel) as input
- Return the corresponding interpolated rate using the specified interpolation method

---

Part 2: Bond Pricing with Discounted Cash Flow

Discounted Cash Flow (DCF) is a fundamental method used to price bonds by calculating the present value of all future cash flows, including periodic coupon payments and the face value at maturity. The idea is that a bond's value today should be the sum of its future cash flows, discounted back to the present using an appropriate discount rate.

Implementation:

You will create a Bond class that represents a bond and computes its price using discounted cash flow. The bond will interact with the YieldCurve object (from Part 1) to determine appropriate discount rates.

Task 2.1: Load the data

The attached Excel spreadsheet (46625HJM3.xlsx) contains the projected cash flows for the corporate bond JPM 5.625 08/16/43 as of January 3, 2024. Additionally, a screenshot from the YAS page on Bloomberg is provided for this bond. You will need to choose the appropriate spread data from the screenshot.

Task 2.2: Create Bond object

You will need to create a Bond object containing cash flow and spread data. Then, you will implement a function to calculate the accrued interest. Lastly, you will compute the bond price using the discounted cash flow method with a given yield curve.

Step 2.2.1: Initialize the Bond Object

Define the __init__ method that:

- Accepts a list of cash flows in the format [(time, amount), ...], where:
  - time represents the future payment time (in years)
  - amount represents the corresponding cash flow (coupon + principal)
- Stores the appropriate spread

Note 2.2.1.1: Date count convention

In bond pricing and fixed-income calculations, a day-count convention defines how interest accrues between two dates. It determines how days are counted within a coupon period and how interest payments are calculated. Different conventions are used across

markets and bond types to standardize calculations. We can use the 30/360 day-count convention in this assignment, here's what it means:

- Each month has 30 days, regardless of the actual number of days.
- A full year is 360 days (12 months × 30 days).

$$Days = 360 \times (Year_2 - Year_1) + 30 \times (Month_2 - Month_1) + (Day_2 - Day_1)$$

This simplifies interest calculations by ensuring equal-length periods across different coupon payments.

Step 2.2.2: Calculate the accrued interest

Accrued interest represents the interest earned but has not yet paid since the last coupon payment. It is computed using the formula:

$$Accrued\ Interest = \frac{Coupon\ Payment \times Days\ Since\ Last\ Coupon}{Days\ in\ Coupon\ Period}$$

Step 2.2.3: Calculate the bond price by implementing calculate_price(yield_curve)

This method will:

1. Take a YieldCurve object as input.
2. For each future cash flow:
    - Retrieve the appropriate discount rate from the yield curve.
    - Adjust the discount rate using Spread.
    - Compute the discounted present value of each cash flow.
3. Sum up the discounted values to obtain the dirty price.
4. Compute the clean price by deducting accrued interest.

$$P = \sum_{t=1}^{N} \frac{C_t}{(1 + r_t + spread)^t}$$

$$Clean\ Price = Dirty\ Price - Accrued\ Interest$$

Where:

- P = Present value of the bond (dirty price).
- C_t = Cash flow at time t.
- r_t = Discount rate from the yield curve at time t.
- $spread$ = Spread added to the discount rate.

Note 2.2.3.1: Clean price vs. dirty price

- **Dirty Price (Full Price):** The dirty price of a bond is the total price a buyer **actually pays** when purchasing the bond. It includes both: The present value of all future cash flows, and the accrued interest since the last coupon payment.
- **Clean Price:** The clean price is the **quoted price** of the bond excluding accrued interest. Since bond prices fluctuate due to market movements but interest accrues daily, financial markets quote the clean price separately to standardize comparisons.
- **Why Do We Need the Clean Price:**
  - Standardized Quotation: Bonds accrue interest daily, so if we only used dirty price, two traders looking at the same bond on different days would see different prices even if market rates didn't change.
  - Comparison Across Bonds: Clean price allows investors to compare bonds without the distortion of accrued interest.

Part 3: Bond Pricing

After completing Part 1 and Part 2, price the bond JPM 5.625 08/16/43 as of January 3, 2024, and compare your calculated price with the price shown in the screenshot.