FRE 6811 Financial Software Lab: Java

Dr. Sateesh Mane

Friday October 16, 2020, 11:59 pm

Final

- Please upload your solution in a Java file with the following naming format.

  `StudentId_first_last_FRE6811_Fall2020_Final.java`

- You are permitted multiple uploads: *only your final submission will be graded.*

Main function

- Create a file `FinalExam.java` in the "`JavaLab`" package.
- Write a class "`FinalExam`" as follows.

```
package JavaLab;

import java.util.Scanner;

public class FinalExam
{
    public static void main(String[] args) {
        // see below
    }

    // see below
    public static void Q1(Scanner sc) ...
}
```

- Write a `main` function as follows.

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);    // instantiate scanner object
    Q1(sc);                                 // pass scanner object to functions
    System.out.println("done");
    sc.close();                             // close scanner object
}
```

- Write functions as follows for the questions in this assignment.

```
public static void Q1(Scanner sc)
```

- Instantiate a `Scanner` object at the start of `main` and pass it as an input to the functions.
- This is to give you practice to write functions and pass inputs.
- All the functions have return type `void`.
- At the end of `main`, close the scanner object and exit.

Submission of file

- When you submit your work, **copy your code in `FinalExam.java` into a blank file named `StudentId_first_last_FRE6811_Fall2020_Final.java`.**
- *DO NOT attempt to compile and run this file.*
- Just upload the file "`StudentId_first_last_FRE6811_Fall2020_Final.java`" as your submission and I will deal with it on my side.

1 YieldCurve class

- We shall write a `YieldCurve` class.
- We treat only a semiannual frequency.
- Declare a `YieldCurve` class with the following data and constructor and methods.

```
final class YieldCurve
{
    private ArrayList<Double> par_yields = new ... // empty ArrayList
    private ArrayList<Double> par_tenors = new ... // empty ArrayList
    private ArrayList<Double> discount_factors = new ... // empty ArrayList
    private ArrayList<Double> spot_rates = new ... // empty ArrayList
    private ArrayList<Double> spot_tenors = new ... // empty ArrayList

    public YieldCurve(double y[]) {
        for (int i = 0; i < y.length; i++) {
            par_yields.add(y[i]);
            par_tenors.add(0.5*(i+1));
        }
        bootstrap();
    }

    // public accessor methods

    public double lin_interp(double t) { ... }  // linear interpolation
    public double cfr_interp(double t) { ... }  // CFR interpolation

    private void bootstrap() {
      // implement the bootstrap
    }

}
```

2 `bootstrap()`

- This is a private method, called in the constructor.
- **The par yields are input in percent by the calling application.**
- Implement the bootstrap algorithm.
  1. The formulas for the first few discount factors are as follows ("y_{dec}" denotes a decimal yield).

$$d_{0.5} = \frac{1}{1+\frac{1}{2}y_{\text{dec},0.5}} \tag{2.1}$$

$$d_{1.0} = \frac{1-\frac{1}{2}y_{\text{dec},1.0}\,d_{0.5}}{1+\frac{1}{2}y_{\text{dec},1.0}} \tag{2.2}$$

$$d_{1.5} = \frac{1-\frac{1}{2}y_{\text{dec},1.5}\,(d_{0.5}+d_{1.0})}{1+\frac{1}{2}y_{\text{dec},1.5}} \tag{2.3}$$

$$\vdots$$

  2. **It is your responsibility to extend the above formulas to the general case.**
  3. For each successful calculation of a tenor in the spot curve, add to the relevant ArrayLists for the spot tenor, discount factor and spot rate.
  4. **The spot rate must be in percent.**
  5. If the calculation of a tenor in the spot curve fails (because we obtain a "discount factor value ≤ 0") ***then break out of the loop and exit the function.***
  6. Hence the spot curve may be shorter than the par yield curve.
- ***It is because the bootstrap may fail, that is why we must use ArrayLists, because we do not know exactly how long the spot curve will be.***

3 Accessors/getters

- Write a set of public accessor methods.
- These are obvious.

```
public int lenParCurve() { return par_tenors.size(); }
public int lenSpotCurve() { return spot_tenors.size(); }
```

- Get an element in the relevant `ArrayList` (return 0 if the index i is out of bounds).

```
public double DiscountFactor(int i);
public double SpotRate(int i);           // return value as percent
public double SpotTenor(int i);
public double ParYield(int i);           // return value as percent
public double ParTenor(int i);
```

4 Interpolation methods

- **Implement methods for linear interpolation and CFR interpolation.**

```
public double lin_interp(double t)
public double cfr_interp(double t)
```

- If $t \le$ (first tenor in the spot curve), return the first spot rate in the spot curve.
- If $t \ge$ (last tenor in the spot curve), return the last spot rate in the spot curve.
- Else implement the formulas from the lecture pdf.

$$\lambda = \frac{t - t_i}{t_{i+1} - t_i} \tag{4.1}$$

$$r_{\text{lin}} = (1-\lambda) r_i + \lambda\, r_{i+1} \tag{4.2}$$

$$r_{\text{cfr}} = \frac{(1-\lambda)\, r_i t_i + \lambda\, r_{i+1} t_{i+1}}{t} \tag{4.3}$$

5 Q1

- Write a function `Q1(Scanner sc)`.
- Prompt the user to an integer value $n > 0$ for the number of par yields.
- Allocate an array y of length n to hold the par yields.
- Prompt the user to enter values for the par yields (in percent) for tenors of 0.5, 1.0, etc. years and populate the array y.
- Instantiate a `YieldCurve` object, using the array of par yields.
- **Print the resulting spot curve.**
  1. Print the (tenor, spot rate, discount factor) in a loop.

     ```
     for (int i = 0; i < (length of spot curve); i++) {
         // print tenor, spot rate, discount factor
     }
     ```

  2. **Format the output so that the columns are neatly aligned.**
  3. The spot rate should be printed as a percentage to 2 decimal places.
  4. The discount factor should be printed as a decimal to 6 decimal places.
- **Prompt the user to input an interpolation time.**
  1. Prompt a few times, at least three times.
  2. Each time, print the value of the interpolated spot rate using linear and CFR interpolation.
  3. **The interpolated spot rates should be printed as percents to 2 decimal places.**
- **Your functions must work properly even if bad inputs are entered (i.e. your program must not crash or throw an exception).**