FRE6831
COMPUTATIONAL FINANCE LABORATORY (PYTHON)

Edward D. Weinberger, Ph.D., F.R.M
Adjunct Professor
Dept. of Finance and Risk Engineering
edw2026@nyu.edu
Office Hours by appointment

PROJECT: IMPLEMENTING A LIBOR YIELD CURVE OBJECT

The class project is to write a Python program that infers the short end of the USD LIBOR yield curve from market observations, via the methodology in use prior to 2008 (More modern methodologies are too complex for an introductory course.). It is to support two calculations:

1. The "discount factor," or present value of a US dollar for a given future date, up to 3 years after a given "spot" date (See below for a definition of the spot date.).
2. The "forward rate," which is the simple interest rate, expressed as a percentage, over a given period between two dates, computed from the above discount factor curve as explained below. Because this calculation uses the above discount factor calculation, both of the input dates must also be less than approximately 3 years after the spot date.

The yield curve is to be implemented as a Python class that I will import into my environment and run via the commands

```
import USDYieldCurve
usdCurve = USDYieldCurve("depoRates.txt", "futuresPrices.txt",
                         "holidayCalendar.txt")
print usdCurve.getDfToDate(date1) #date1 in standard Python format
print usdCurve.getDfToDate(date2) #date2 in standard Python format
print usdCurve.getDfToDate(date3) #date3 in standard Python format
#etc.
print usdCurve.getFwdRate(date4, date5)
print usdCurve.getFwdRate(date6, date7)
print usdCurve.getFwdRate(date8, date9)
#etc.
```

Inputs

Typical inputs to such a calculation – and the inputs to assume here – are to be read from the three text files named above.

1. `depoRates.txt` contains a series of annualized cash deposit rates. These cash deposits are effectively jumbo versions of the certificates of deposit that are on offer at any bank. These deposits should be read from a text file, each line of which has the following format:

   <5 character instrument code> white space <rate as percentage, rounded to nearest basis point>

   The instrument code will always have the prefix "USD" and the suffix "D", "W", or "M", which will indicate whether the deposit is for a period of days, weeks or months. In between this prefix and suffix will be a number, indicating the number of days, weeks, or months to maturity. For

example, “USD1W” would be the code for the cash deposit maturing in one week; “USD2M” would be the cash deposit maturing in 2 months, etc. The start date of these cash deposits is always assumed to be the “spot date” (More about the “spot date” later.).

The first few lines of such a file might read

USD1D	2.73
USD1W	2.74
USD1M	2.75
USD3M	2.80
USD6M	2.85

You can assume that the number in the above codes is only a single digit (Although 12 month LIBOR deposits do exist, they are seldom used to construct yield curves.). Some of the above tenors may be omitted; in fact, the only one that is really necessary is USD6M (Why?).

2. `futuresPrices.txt` contains a series of prices of Eurodollar futures, which are futures on 3 month cash LIBOR deposits. You can assume that the futures on successive lines of the file mature on successively later dates. Each line of the file has the following format:

   <4 character instrument code> <price, given as a four digit integer>

   The instrument code will always have the prefix “ED”, followed by one of the letters “H”, “M”, “U”, or “Z”, followed by a single numeric character. The letters “H”, “M”, “U”, and “Z” correspond to the maturity date of the future: the respective 3rd Wednesdays of March, June, September, and December of the year indicated by the numeric character. Thus, “EDZ2” would be the code for the future expiring in December, 2022. Necessarily, these maturity dates are dates in the future, so there is no ambiguity, given that all futures must mature prior to ten years from the current date.

   Thus, the first few lines of such a file might read

EDH9	97.25
EDM9	97.20
EDU9	97.19

3. `holidayCalendar.txt` contains the dates of Federal Reserve holidays, given as character strings in the form YYYY-MM-DD, for the next five years (available from, among other places, the Federal Reserve website).

As we will see in more detail in the calculation below, it may be impossible to compute the yield curve if the input files are incomplete. In this case, an error condition should be raised and the error message “Cannot build curve from given inputs” should be generated.

Calculation

“Modified Following” date adjustment

USD payments can only be made on days in which the US Federal Reserve (the “Fed”) is open. If the date of a scheduled payment falls on a weekend or a Federal Reserve holiday, it must be adjusted according to the so-called *modified following* business convention. Under this convention, if a payment

would otherwise be scheduled on a non-business day, the payment date "rolls forward" to the next business day unless the next business day falls into the subsequent month. If so, the payment date "rolls backward" to the next previous month. Thus, a payment that would be scheduled for December 30, 2018 (a Sunday) would be rolled forward to December 31, 2018 (a Monday and a business day), but a payment scheduled for January 1, 2019 (New Year's Day and a bank holiday) would be rolled backward to December 31.

The spot date

The beginning of the period over which USD interest is computed is usually the so-called "spot date," which is always two business days after the trade date (The intervening time period is required for clearing and settlement.). Given this interest computation convention, discount factors are also only defined from the spot date. The spot date will be denoted as *s* below.

Date arithmetic

Here and subsequently, if dates are to appear in a formula at all, one date will be subtracted from a later date. Use the Python built-in library `datetime` to do the date arithmetic. If d_1 and d_2 are two such dates, $d_1 - d_2$ is interpreted as the number of calendar days between them (More precisely, this difference calculation results in a `timedelta` object, which can then be converted to a Python `int`.). As we will see, this is just what is needed for the calculations below.

Computing discount factors from cash deposit rates

LIBOR interest calculations for periods less than a year always assume simple interest applied to the relevant fraction of a year, computed as the actual number of days that a loan is outstanding, including weekends and holidays, divided by 360 (Remember, all of this was standardized before there were computers to facilitate the calculations!). Thus, if the given rate is R and the maturity date of the deposit is d and the spot date is s, the amount to be repaid for every dollar lent is $1 + R(d - s)/3600$. It follows that the discount factor, or present value of a dollar n days from the spot date is $(1 + Rn/3600)^{-1}$. If a discount factors df_1 and df_2 have been computed for dates d_1 and d_2 after the spot date, the discount factor, df, for a date d between d_1 and d_2 is given by logarithmic interpolation

$$\ln(df) = \ln(df_1) + [(d - d_1)/(d_2 - d_1)]\,[\ln(df_2) - \ln(df_1)].$$

Computing discount factors from futures prices

Interest rate futures are futures on forward rates, *i.e.* rates over some pair of fixed future dates (In actuality, 3 month LIBOR really means 3 calendar months, but assume, for simplicity, that it is the period between successive futures maturity dates.). Thus, these futures are the market's best guess as to what these rates will actually be when the beginning of the forward period coincides with the spot date, and 3 month "cash" LIBOR becomes known. In the meantime, the rate implied by a futures price P is computed as a percentage according to the formula

$$R = 100.0 - P/100.0$$

As before, simple interest is assumed, with the number of days in the three calendar months starting with the futures expiration date.

However, the discount factor thus computed is a *forward* discount factor; that is, it is the discount factor *from the expiration of the future* to the expiration of the underlying 3 month LIBOR deposit. To get the

discount factor *from spot* to the expiration of the LIBOR deposit, this forward discount factor must be multiplied by the discount factor from spot to the start date of the LIBOR deposit, which is also the maturity date of the future. Thus, given the discount factor, df_s, from spot to the earliest futures expiration date and the forward discount factor, df_1, computed as above, the discount factor to the expiration date of the first future is $df_s \times df_1$. In general, the maturity dates of the any of the futures will not coincide with any of the expiration dates of the cash deposits, so the interpolation formula given above will be required. This computation cannot be done if the rates for cash LIBOR deposits are unavailable for the required maturities. If this is the case, an error condition should be raised, and the message "Insufficient LIBOR cash rate data" should be printed to STDOUT.

Again, we simplify things by ignoring the awkward fact that the futures maturity dates are almost never exactly three months apart; instead, we assume that the maturity date of the 3 month LIBOR deposit underlying one futures contract is also the expiration date of the next futures contract. Thus, the discount factor to the maturity date of the underlying of the second future is $df_s \times df_1 \times df_2$, where df_2 is the forward discount factor computed from the price of second future. This process can be continued to calculate the discount factor from spot to the n^{th} futures underlying expiry date as $df_s \times df_1 \times df_2 \times \ldots \times df_n$.

The `df` *method*

This method should return the value of the *df* curve for the date, *d*, that has been passed as an argument. In general, *d* will not coincide with any of the dates for which the above calculation will have produced discount factors. In that case, determine the required *df* curve value by finding dates d_1 and d_2 from the spot date, the discount factor, *df*, for a number, *d*, of days between d_1 and d_2 is given by logarithmic interpolation

$$\ln(df) = \ln(df_1) + [(d - d_1)/(d_2 - d_1)]\,[\ln(df_2) - \ln(df_1)]$$

If *d* is before the spot date or *d* is after the last date for which the discount factor curve is defined, an error condition should be raised, and an appropriate error message should be printed to STDOUT.

The `fwdRate` *method*

Given two dates, $d_1 < d_2$, this method should return the forward rate, r_f, between these two dates, inferred from the discount curve computed above, according to the formula

$$r_f = \frac{360}{d_2 - d_1}\left[\frac{df(d_1)}{df(d_2)} - 1\right]$$

Use Python's built-in `datetime` library to do the date arithmetic. This formula will compute the rate as a decimal fraction, rather than a percentage (*e.g.* something like 0.03, rather than 3%.).

If d_1 is before the spot date, $d_1 \geq d_2$, or d_2 is after the last date for which the discount factor curve is defined, an error condition should be raised, and an appropriate error message should be printed to STDOUT.

Testing

In the "Project" folder on the Course website, I have posted the spreadsheet "`YCtestcase.xlsx`", which implements the calculations for the spot date April 24, 2015.

Notes:

1. More modern methodologies take into consideration the changes in the LIBOR markets that arose from the debt crisis of 2008, which involves the use of a truly risk free curve for discounting (As the financial community learned to its chagrin, interbank lending, the "I" in LIBOR, is *not* risk free!). Thus, one curve is needed for discounting and another is needed for predicting forward rates.
2. There is considerable disagreement about exactly which mix of the available cash deposits, futures, and swaps – or even whether instruments besides these – should be used in the above calculation. What is generally agreed upon is that the most liquid instruments should be used, as these are the best indicators of the market.
3. Although you are to use the interpolation methods given above, they are often replaced in practice by more sophisticated methods, such as splines.
4. Because owners of futures contracts are required to post margin daily as their futures prices vary, the rates inferred from futures are actually biased estimates. In practice, a so-called "convexity adjustment" is often applied to these rates. For simplicity, we will ignore this adjustment.
5. The above conventions and instrument details are only applicable to USD. The construction of the LIBOR discount curve for other currencies differs in detail, but not in general, from this outline. For example, GBP has a spot date one day after the trade date, and 365 replaces 360 in the above calculations.