Columbia University
MATH GR5260 Spring 2023
Programming for Quant and Computational Finance
Ka Yi Ng
Homework Assignment 1
Release date: Jan 20, 2023 (Fri)
Due date: Feb 9, 2023 (Thu) 11:00pm

HOMEWORK GUIDELINE

Submit your solution file(s) onto CourseWorks by the specified due date and time.

A. Theory parts
   Your solution to theory questions can be (i) hand-written and scanned as a pdf file or (ii) prepared using Word and be converted into a pdf file or (iii) be included in the solution for practice parts.
B. Practice parts
   Your Python source code and outputs shall be prepared and submitted in a format (eg. pdf, html, .ipynb, .py, etc.) that the graders can understand. Before submitting your solution, make sure that you have run your program successfully to generate all required outputs. Points may be deducted if some outputs are missing.
   You are free to use standard python packages distributed by Anaconda.
   Do not submit additional packages that can be downloaded from the web. Just provide instructions in the solution file.
C. Other files
   Include files that may be requested by the homework assignment. Make proper references in the solution file.

Note: If you suspect there are typos in this homework, or some questions are wrong, please first discuss with your TAs.

QUESTION 1

In this question, we will use swap rate quotes to construct a discount factor curve using log-linear interpolation.

Suppose that a discount factor curve is represented by the discount factors for a set of dates D_i's where D_0 is today and D_i $(0 < i \leq n)$ is the maturity date of the ith swap. Discount factor for any other date is interpolated from the discount factor curve.

Let t_i be the time in years from today to the date D_i and let DF_i be its associated discount factor for $0 \leq i \leq n$. Assume that $t_0 = 0$ and $DF_0 = 1.0$.

For a given time t where $t_i \leq t < t_{i+1}$, the log-linearly interpolated discount factor for time t is:

$$DF(t) = \exp\{\ln(DF_i) + (t - t_i)(\ln(DF_{i+1}) - \ln(DF_i))/(t_{i+1} - t_i)\}$$

a) Define a function to return the log-linearly interpolated discount factor for a given time t. For $t > t_n$, this function shall return the discount factor computed from the zero rate of the last discount factor DF_n in the curve. That is: $DF(t) = \exp(-zt)$ and $z = -\ln(DF_n)/t_n$

The function signature shall look something like this:

```
def get_df(t, df_times, df_values)
```

b) Define a function to return the simple forward rate of a period. The forward rate output is implied from a given discount factor curve as follows.

For a given forward rate period from t_S to t_E, the simple forward rate is computed as:

$$F(t_S, t_E) = (\frac{DF(t_S)}{DF(t_E)} - 1)/(t_E - t_S)$$

The function signature shall look something like this:

```
def get_forward_rate(t1, t2, df_times, df_values)
```

Whenever you have defined some functions, it's always a good practice to test your functions for some boundary cases.

c) Create a discount factor with only two points. $t_0 = 0$ and $DF_0 = 1.0$ and $t_1 = 1.0$ and $DF_1 = 1.0/(1 + 0.05)$. Then call your get_df function as follows for sanity checks.

```
print('DF(0.0):', get_df(0.0, df_times, df_values))
print('DF(0.5):', get_df(0.5, df_times, df_values))
print('DF(1.0):', get_df(1.0, df_times, df_values))
print('DF(1.5):', get_df(1.5, df_times, df_values))
```

Now we define a function for pricing swaps.

For simplicity, we only consider swaps receiving fixed rate annually and paying floating rate quarterly. We also assume that the swaps start from today (ie. zero spot days) and the swap periods are evenly distributed. For example, for a 1-year swap with annual fixed periods and quarterly floating periods, the fixed period starts at time 0.0 and ends at time 1.0. The floating periods start at time 0.0, 0.25, 0.5,0.75 and end at time 0.25, 0.5,0.75,1.0.

The present value of the swap is:

$$PV = \sum_{i=1}^{k-1} R(\mathcal{T}_i - \mathcal{T}_{i-1})DF(\mathcal{T}_i) - \sum_{j=1}^{m-1} F(T_{j-1}, T_j)(T_j - T_{j-1})DF(T_j)$$

Where:

R is the swap fixed rate

$\mathcal{T}_1, \dots, \mathcal{T}_{k-1}$ are the time (in years) to the fixed period end dates.

$T_1, \dots, T_{m-1}$ are the time (in years) to the floating rate period end dates.

$\mathcal{T}_0 = T_0 = 0.0$

d) Define a function to return the present value of a swap with given fixed rate and maturity. The function signature shall look something like this:

```
def swap_price(rate, maturity, df_times, df_values)
```

e) Now test your function. Create a discount factor with three points.

$t_0 = 0$ and $DF_0 = 1.0$, $t_1 = 0.5$ and $DF_1 = 1.0/(1 + 0.04 * 0.5)$, $t_2 = 1.0$ and $DF_2 = 1.0/(1 + 0.05)$.

Then call your swap_price function as follows for sanity checks. Both results should be zero (up to a machine error).

```
print('Swap PV:', swap_price(0.05, 1.0, df_times, df_values))
print('Swap PV:', swap_price(0.04, 0.5, df_times, df_values))
```

We will use optimize.newton method in scipy to solve for the discount factor for the swap maturity t_i given that the discount factors $DF_1, \dots, DF_{i-1}$ are solved.

f) The method optimize.newton requires an input for the function to solve. Define such function to return the present value of a swap if the DF_i in the discount factor curve is replaced with the input value df_input. The function signature shall look something like this:

```
def function_solve(df_input, i, rate, maturity, df_times, df_values)
```

g) Define a function to generate a discount factor curve from a list of swap rate quotes. The function signature shall look something like this:

```
def generate_discount_curve(maturs, swap_rates)
```

h) Use the following list of swap rate quotes to generate a discount factor curve.

Maturity	1M	3M	6M	1Y	2Y	3Y	5Y
Swap rate %	4.4	4.6	4.9	4.5	3.8	3.2	2.8

i) Use the curve generated above to compute the daily forward rates for the next 2000 calendar days. Plot the computed daily forward rates. Comment on the shape of the curve and explain why this is the case.

--end of Assignment—