MAT1856/APM466 Assignment 1

Chen Shen, Student #: 1009724924

February, 2024

Fundamental Questions - 25 points

1. Simple questions

   (a) Issuing bond is to borrow money for government spending; printing money also surges the whole money amount, inducing the inflation and decreasing its purchase power.

   (b) When investors think future economy is gonna be recession, they tend to buy long-term bonds, which surges the price of the LR bond, and makes the LR interest rate flatten.

   (c) QE is a monetary policy tool, used during Covid-19 by purchasing financial assets from the market to increase the money supply, and boost overall economic activity.

2. Selected bonds

Bond Code	Maturity date	Coupon	Issue Date
CAN 2.75 Aug 24	8/1/2024	2.750%	5/24/2022
CAN 1.25 Mar 25	3/1/2025	1.250%	10/11/2019
CAN 3.50 Aug 25	8/1/2025	3.500%	5/12/2023
CAN 3.00 Apr 26	4/1/2026	3.000%	1/20/2023
CAN 1.00 Sep 26	9/1/2026	1.000%	4/16/2021
CAN 1.25 Mar 27	3/1/2027	1.250%	10/15/2021
CAN 2.75 Sep 27	9/1/2027	2.750%	5/13/2022
CAN 3.50 Mar 28	3/1/2028	3.500%	10/21/2022
CAN 3.25 Sept 28	9/1/2028	3.250%	4/21/2023
CAN 4.00 Mar 29	3/1/2029	4.000%	10/13/2023

   Since we are selecting bonds to construct a yield curve, they should be consistent in some ways. I chose the bonds with similar coupon rates, recent issue dates. Besides, I potentially chose the maturity dates of bonds to be evenly gaped with 6 months. Hence, in the later analysis, I can assume there are exact 6 months in between any bonds, which is good for calculation and plotting.

3. Each eigenvalue represents the amount of variance explained by its corresponding eigenvector. A large eigenvalue means that the eigenvector accounts for a significant proportion of the variance in the data. The eigenvalues give us an understanding of the "spread" or variability of the stochastic processes along the directions defined by the eigenvectors.

   Eigenvectors are orthogonal directions in the feature space along which the variance is maximized. The first eigenvector points in the direction of greatest variance. The second eigenvector is orthogonal to the first and represents the second most significant mode of variation, so on and so forth.

Empirical Questions - 75 points

4.

(a) To calculate the ytm curve, I used the following formula:
Dirty price $= \sum_{t=1}^{n} C(1+\frac{r}{2})^{(-t)} + F(1+\frac{r}{2})^{(-n)}$ (as the coupon of bonds is given semi-yearly), where Dirty price = Accrued interest + Clean price,
n is the period count (every 6 months till maturity is a period).
The yield to maturity is determined for each bond at the closing price of each day, hence it is a given-bond-rate. I use linear interpolation, which is the easiest and straight forward.

(b) The algorithm for calculating a yield curve is as following:
B = the list of dirty prices of bonds sorted by their maturity dates from early to late
T = the list of months to maturity of bonds correspondingly
Y = a blank list to save spot rates

this is the case where we have our first spot rate at six months
Assume B[0] has maturity less than 6 months, $r(t) = \frac{(F+C)}{P} - 1$, save this to Y[0].
For i in range [1: len(B)]:
we deducted all the discounted values from the dirty price that we have spot rates
residual $= B[i] - \sum_{i=0}^{i-1} \frac{C}{(1+Y[i])^{i+1}}$
calculate the new spot rate from the residual since $residual = \frac{(C+F)}{(1+r(t))^{i+1}}$
$r(t) = [\frac{(C+F)}{residual}]^{\frac{1}{i+1}} - 1$, save this to Y[i].

Notes: (1) N is the notional, C is the coupon, F is the face value. (2) Due to the seek of consistence, I used discrete methods in all of these 3 sub-questions. (3) The diagram of this sub-question is similar to the ytm curve, I have fully checked my code and concluded as follows: For all of 10 days data, only the first spot rate (6 month rate) is approximately equal to the first ytm rate, and gaps are less than 10^{-15}. All the subsequent spot rates (from the 2nd to the last), have small gaps of 0.01% compared with the corresponding ytm rates. I guess this was because those bonds are so similar in coupon rates and their prices, and the last payment (face value and coupon) accounts for the majority of the spot rate. For detailed difference data, please see the table at the end of section 2.2 in Colab notebook.

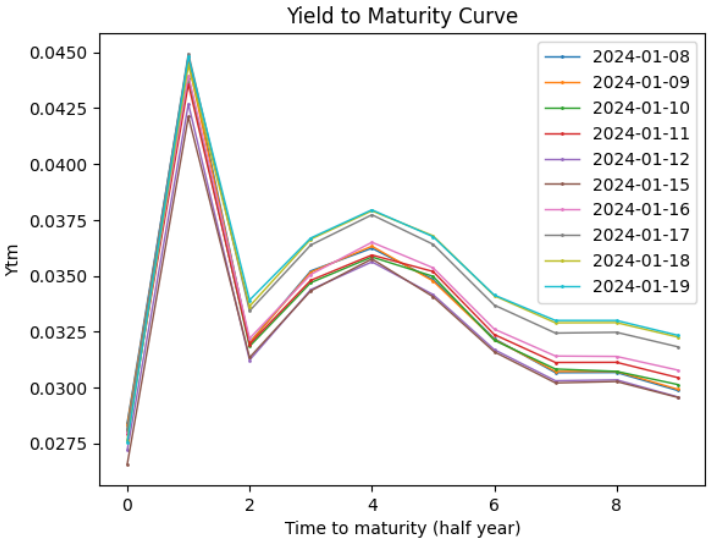
(c) Forward rates calculation is relatively independent from bond to bond (not that dependent like a sequence in calculating the spot curve using bootstrapping), so I would like to explain this using the following sentences. First, since we only cares about forward rates between 1 year to 2, 3, 4, 5 years. We only need 5 bonds out of 10 at each day (specifically, the bonds that will be matured in 1, 2, 3, 4, 5 years from this year). Second, for each forward rate from t to t+n:
$F_{t,t+n} = (\frac{(1+S_{t+n})^{2(t+n)}}{(1+S_t)^{2t}})^{\frac{1}{2n}} - 1$

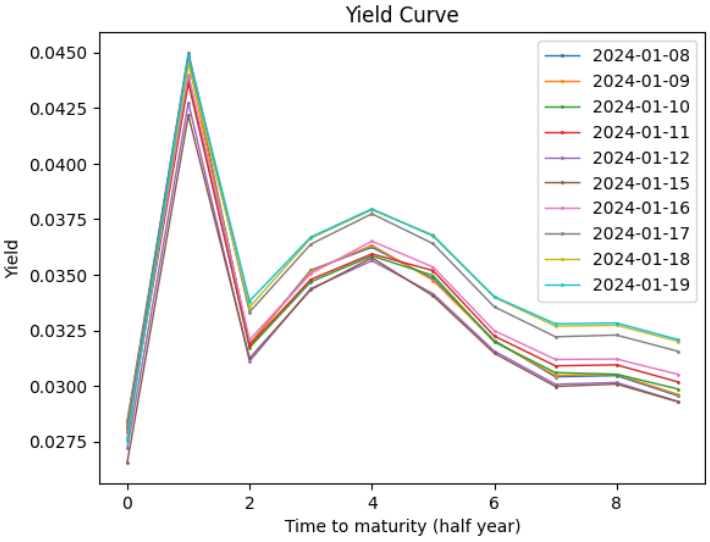
Where S_{t+n} is the spot rate at $t+n$, n is period (6 months), semi-annual compounding. Note: since I used discrete compounding in interest in (b), I adopt the discrete method here as well.

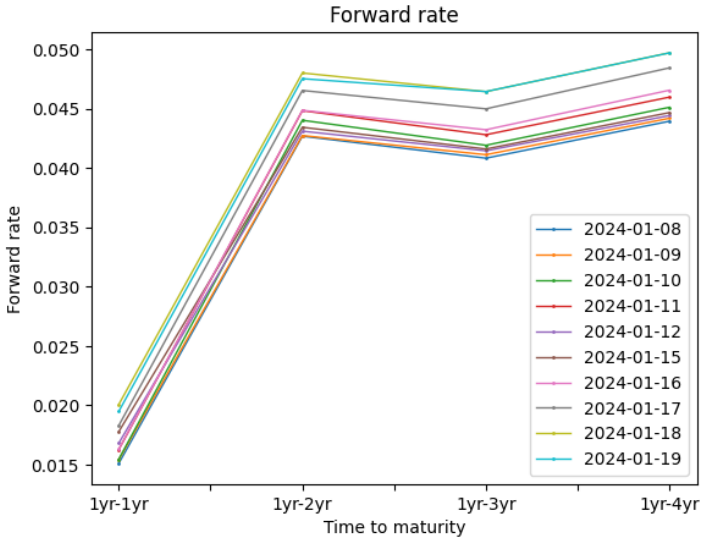
5. Two covariance matrices have been put together with plotting at next page.

6. The eigenvalues and eigenvectors have been put together with plotting at next page. The asset portfolio coming up with this eigenvector can represent this largest eigenvalue% of market direction.

[diagrams footnote]: The diagrams of 4(1), 4(2), 4(3) are listed in sequence at the left hand side of the next page. The diagrams of question 5, and 6 are listed at right: 5(1) gives log returns covariance matrix, 5(2) gives forward rate covariance matrix; 6(1) gives eigenvalues and vectors of log returns covariance matrix, 6(2) gives eigenvalues and vectors of forward rate covariance matrix.








```
# question 5(1): log-returns of yield cov matrix
calculate_covariance(ytm_table_by_day)
```

```
array([[0.00054357, 0.00044925, 0.00027645, 0.0002227 , 0.00019615,
        0.00036156, 0.00030236, 0.00036346, 0.0003456 , 0.00037368],
       [0.00044925, 0.00043397, 0.0003078 , 0.00026726, 0.00023715,
        0.00033152, 0.00029986, 0.0003406 , 0.0003327 , 0.0003453 ],
       [0.00027645, 0.0003078 , 0.00030356, 0.00026187, 0.00022854,
        0.00030816, 0.0002837 , 0.00031254, 0.00031178, 0.00032226],
       [0.0002227 , 0.00026726, 0.00026187, 0.00025215, 0.00022095,
        0.00025104, 0.00024886, 0.00026153, 0.00026702, 0.00026383],
       [0.00019615, 0.00023715, 0.00022854, 0.00022095, 0.00019883,
        0.00021155, 0.0002111 , 0.00022052, 0.00022632, 0.0002226 ],
       [0.00036156, 0.00033152, 0.00030816, 0.00025104, 0.00021155,
        0.00037711, 0.00032417, 0.00037343, 0.0003624 , 0.00039013],
       [0.00030236, 0.00029986, 0.0002837 , 0.00024886, 0.0002111 ,
        0.00032417, 0.00029318, 0.00032753, 0.0003237 , 0.00033738],
       [0.00036346, 0.0003406 , 0.00031254, 0.00026153, 0.00022052,
        0.00037343, 0.00032753, 0.00037465, 0.00036607, 0.00038811],
       [0.0003456 , 0.0003327 , 0.00031178, 0.00026702, 0.00022632,
        0.0003624 , 0.0003237 , 0.00036607, 0.00036216, 0.00037807],
       [0.00037368, 0.0003453 , 0.00032226, 0.00026383, 0.0002226 ,
        0.00039013, 0.00033738, 0.00038811, 0.00037807, 0.0004046 ]])
```

```
# question 5(2): forward rates cov matrix
calculate_covariance(forward_table)
```

```
array([[3.38640973e-03, 2.11912417e-04, 1.22430869e-04, 3.08282694e-05],
       [2.11912417e-04, 6.27443068e-04, 5.44478070e-04, 5.55862611e-04],
       [1.22430869e-04, 5.44478070e-04, 5.02881841e-04, 5.10500935e-04],
       [3.08282694e-05, 5.55862611e-04, 5.10500935e-04, 5.25320661e-04]])
```

```
# question 6(1): log-returns of yield eigenvalues
EValue1
```

```
array([ 3.12239229e-03,  2.56046791e-04,  1.39533031e-04,  1.62424353e-05,
        5.56673432e-06,  3.02912356e-06,  5.12776080e-07,  4.51062051e-07,
        3.27635420e-20, -1.91787297e-20])
```

```
# question 6(1): log-returns of yield eigenvectors
EVector1.round(5)
```

```
array([[-3.5633e-01, -7.5355e-01,  1.7870e-02, -2.5686e-01,  2.9545e-01,
        -1.5380e-01,  2.9557e-01, -1.6046e-01, -2.5940e-02,  1.2019e-01],
       [-3.4278e-01, -3.5783e-01,  4.7173e-01,  3.5495e-01, -4.2230e-01,
         2.1177e-01, -3.2507e-01,  2.3242e-01,  5.2730e-02, -1.5184e-01],
       [-2.9579e-01,  2.6313e-01,  1.8380e-01,  6.7641e-01,  1.7634e-01,
        -2.8939e-01,  3.4082e-01, -1.9205e-01, -3.5120e-02,  2.8032e-01],
       [-2.5347e-01,  3.1869e-01,  4.0665e-01, -3.5097e-01, -1.6188e-01,
         2.4374e-01,  5.1400e-01, -1.7515e-01, -1.3268e-01, -3.0749e-01],
       [-2.1851e-01,  2.6016e-01,  4.5697e-01, -2.8159e-01,  5.7288e-01,
        -1.2315e-01, -4.8811e-01,  8.6650e-02,  4.3430e-02,  5.2460e-02],
       [-3.3753e-01,  6.6140e-02, -3.6595e-01,  1.3595e-01,  3.0600e-01,
         4.7246e-01,  2.4780e-02,  3.9304e-01, -5.0612e-01,  1.5580e-01],
       [-3.0098e-01,  1.7001e-01, -9.1110e-02, -2.6164e-01, -2.5893e-01,
         3.0374e-01,  6.6320e-02,  9.3070e-02,  3.7333e-01,  5.1286e-01],
       [-3.4087e-01,  7.7540e-02, -2.6379e-01, -4.6340e-02, -2.1265e-01,
         4.3000e-04, -4.2808e-01, -7.0608e-01, -2.8091e-01,  9.3430e-02],
       [-3.3470e-01,  1.3971e-01, -1.8064e-01, -2.1373e-01, -3.3416e-01,
        -6.7613e-01,  5.8500e-03,  4.2823e-01, -1.8076e-01, -1.6000e-04],
       [-3.5117e-01,  8.2660e-02, -3.5320e-01,  1.1520e-01,  1.7698e-01,
         2.3370e-02, -7.6500e-03, -1.0110e-02,  6.8462e-01, -7.0042e-01]])
```

```
# question 6(2): forward rates eigenvalues
EValue2
```

```
array([3.41657562e-03, 1.60334194e-03, 2.00976795e-05, 2.04006552e-06])
```

```
# question 6(2): forward rates eigenvectors
EVector2
```

```
array([[ 0.99222343,  0.11911471,  0.02845867,  0.02223654],
       [ 0.09661742, -0.60302644, -0.78859575, -0.07170026],
       [ 0.06692571, -0.54820284,  0.48880316, -0.67532663],
       [ 0.04097189, -0.56713706,  0.37199236,  0.73368832]])
```