APM466: Mathematical Theory of Finance

Assignment 1: Yield Curves

Tianyu Ren

1002379188

University of Toronto
Feb 2019

- Fundamental Questions

1. Why shouldn't you use bonds with an initial term (maturity - issue date) of 30 years for calculating points on the yield curve less than 5 years out? Should the on-the-run Canadian 5 year bond be used to calculate the 5 year point of the yield curve instead of a 10 year bond even if a 10 year bond has a maturity closer to exactly 5 years away? *(Please provide at least two reasons for each.*)

   (Hint: Coupons, and financial flows.)

   Yield of bonds with an initial term of 30 years represents a long-term trend. Thus, if I use it to calculate points on the yield curve less than 5 years out, it can not picture the this short-term trend explicitly. Additionally, yield is directly related to coupons. Those coupons are set 30 years ago, which may not be able to reflect the trend or change in recent 5 years. Liquidity might be another reason since the short-term bond has a higher liquidity than the long-term one.

   Also, on-the-run Canadian5 year bond should be used to calculate the 5 year point of the yield curve because if I use a 10 year bond with a maturity closer to exactly 5 years away, I may not be able to calculate all the rates that cover the entire 5 years (Since the mature time is less than 5 yeas). Besides, rates depends on financial flows. Hence, on-the-run Canadian 5 year bond could have a better representation than a 10 year bond.

2. In the second part of this assignment, you are to derive the spot curve up to 5 years out via bootstrapping for each day in your data-set. Please list a subset of bonds for which you will feed into a bootstrapping algorithm and explain your choice for each bond chosen. You may use a day count which rounds days to the nearest month (I.e., Jan 10th becomes Jan 1, Mar 20 becomes Apr 1, and the length of time between Mar 20 and Jan 10 would be 3 months).

   (Note: To easily refer to a bond, please use the following convention: CAN 2.5 Jun 24 refers to the Canadian Government bond with a maturity in June 24 and a coupon o 2.5).

   CAN 1.75 Mar 01 19; CAN 1.75 Sep 01 19; CAN 1.5 Mar 01 20; CAN 0.75 Sep 01 20; CAN 0.75 Mar 01 21; CAN 0.75 Sep 01 21; CAN 0.5 Mar 01 22; CAN 1 Sep 01 22; CAN 1.75 Mar 01 23; CAN 2 Sep 01 23

   In order to get a better approximation of the curve, I choose 10 bonds where the maturity days are in March and September. This is because can use them to interpolate rates in June and December so that the time lag is approximately to half a year given the date today is Jan 7th. In other words, when we use bootstrapping algorithm, we can easily deal with the time variable since they are consistent with each other. For other bonds, for example, February and August, some data are missing and it's more complicated to bootstrap or interpolate.

3. The final part of this assignment asks you to treat the 1,2,3,4,5 year points of the yield curve as time series, and compute the eigenvalues and eigenvectors of the associated 5x5 covariance matrix of log-returns.

   (a) Explain what the distribution of eigenvalues tells you about the characteristics of the time series associated with the 1,2,3,4,5 year points on the yield curve.

The eigenvectors can then be interpreted as independent fundamental yield curve shifts. The distribution of eigenvalues tells me the relative size, which can be used to identify the fundamental yield curve shifts. This is related to PCA, and rates move together along the curve.

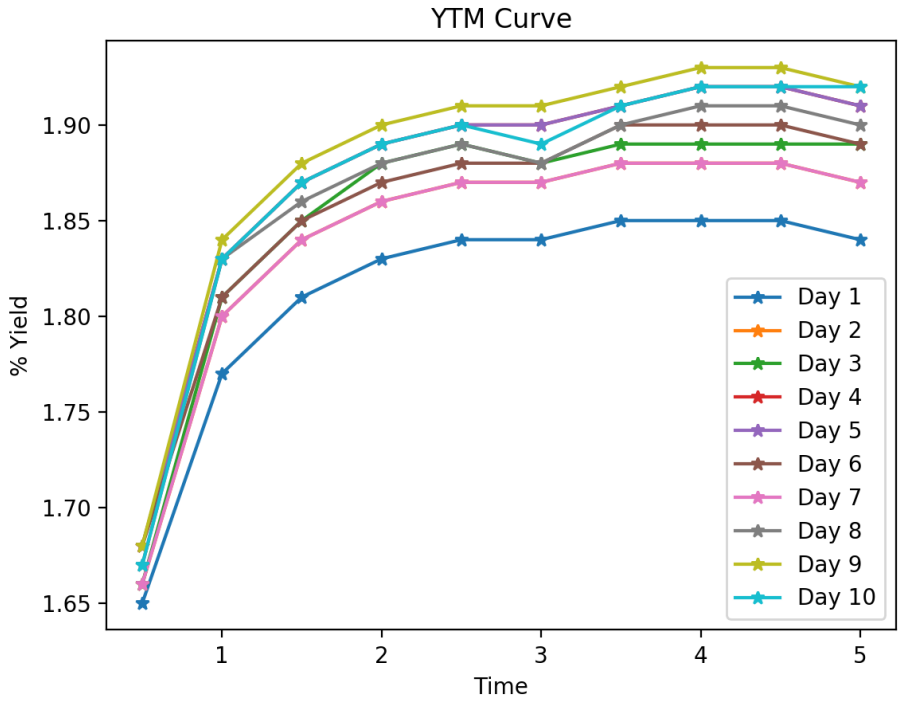
(b) Give two examples of financial classes for which you would expect to have the co- variance matrices of the log-returns of assets within said classes produce completely different distributions of eigenvalues, and explain why this would be so. Examples of financial classes could be US Stocks, Canadian Government Bonds, Euro Hedge Funds, FX Swaps, etc.

For example, US Stocks and Canadian Government Bonds. As we know matrix of covariances among various assets' returns is used to determine the relative amounts of different assets that investors should choose to hold in a context of diversification. According to this, stocks and bonds are not strongly correlated which means they have different co-variance matrices. Thus, I would expect them to have completely different distributions of eigenvalue.

4. Describe the motivation and mathematics behind a mathematical model other than bootstrapping that at least two central banks (state which banks) use to derive their long-term spot curves. Please also state two advantages and one disadvantage of the described model. (Complete answers only - no half marks.)

- Empirical Questions

5. (a) Provide a well-labeled plot with a yield curve (ytm curve) corresponding to each day of data superimposed on-top of each other. You may use any interpolation technique you deem appropriate provided you include a reasonable explanation for the technique used.



For this YTM plot, I use the $Yield$ from the dataset directly. No interpolation used here.

(b) Write a pseudo-code (a simple explanation of an algorithm) for how you would derive the spot curve with terms ranging from 1-5 years from your chosen bonds in part 2. (Please also recall the day convention simplifications provided in part 2 as well.) Then a well-labeled plot with a spot curve corresponding to each day of data superimposed on-top of each other.

Since I use Python for this question, I will attach my code directly with some explanations. I use the bootstrapping formula:

$$r_i = -\log\left(\frac{p_{dirty} - \sum_{k=1}^{i-1} C_i/2 \cdot e^{-r_k \cdot t_k}}{100 + c/2}\right) / \frac{n}{365}$$

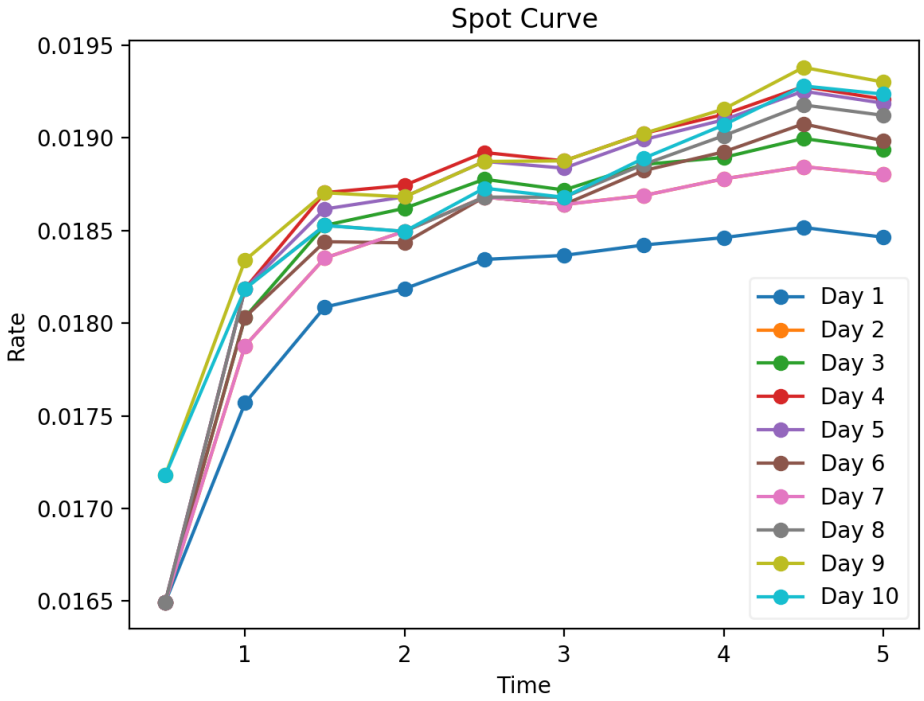
where

$$P_{dirty} = P_{clean} + accured \quad interst$$

```
# Using Bootstrap to calculate spot rate.
pre_sum = 0      # summation of all previous rates in bootstrapping formula
pre_t = 0        # previous time used in bootstrapping formula
r1 = 0
spot_rates = []
zcb_price = []
for index, row in selected_bonds.iterrows():
        if index % 10 == 0:      # check if this is the r_1
                pre_sum = 0
                pre_t = (row["Maturity"] - TODAY).days / 365
                # if index != 0:
                #         TODAY += pd.DateOffset(1)

        t = (row["Maturity"] - TODAY).days / 365
        r = -math.log((row["Dirty price"] - row["Coupon"] *
                pre_sum / 2)/(100 + row["Coupon"] / 2)) / t
        spot_rates.append(r)
        zcb_price.append(math.exp(-r * t))
        pre_sum += math.exp(-r * pre_t)
        pre_t = t

selected_bonds["Spot rate"] = spot_rates
selected_bonds["ZCB price"] = zcb_price
```



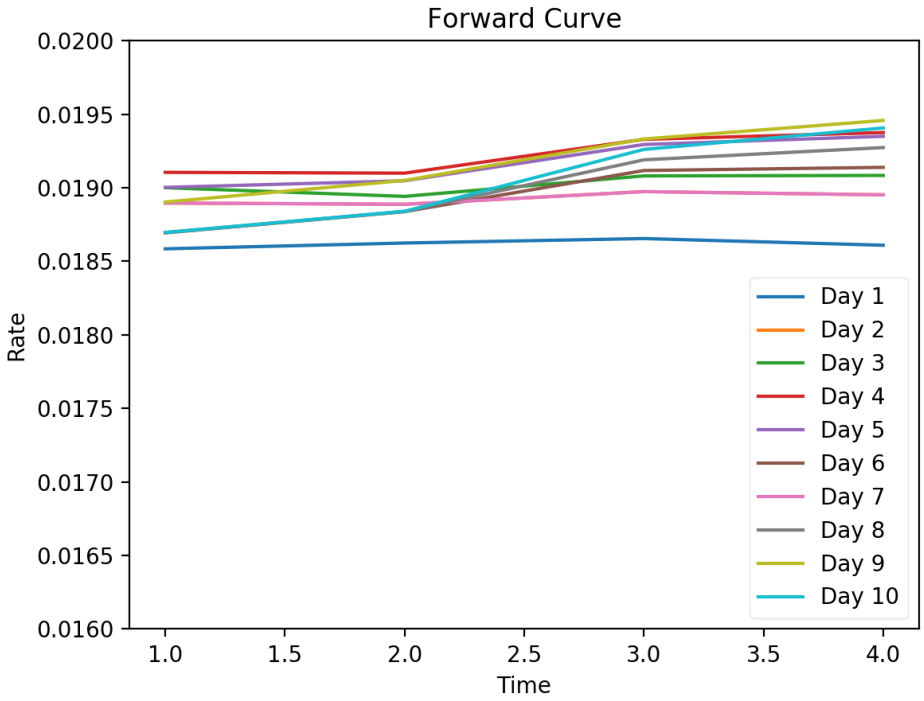
(c) Write a pseudo-code for how you would derive the 1-year forward curve with terms ranging from 2-5 years from your chosen bonds in part 2 (I.e., a curve with the first point being the 1yr-1yr forward rate and the last point being the 1yr-4yr rate). Then provide a well-labeled plot with a forward curve corresponding to each day of data superimposed on-top of each other.

To calculate the forward rate, I use the formula:

$$r(t, T_i, T_{i+1}) = -\frac{\log P(t, T_{i+1}) - \log P(t, T_i)}{T_{i+1} - T_i} \quad , where \quad i = 1, 2, 3, 4$$

```
# Calculate forward rate
TODAY = pd.to_datetime("2019-Jan-07")
forward_df = selected_bonds[selected_bonds.index.values % 2 == 1]
price1 = 0
time1 = 0
forward_rates = []
for index, row in forward_df.iterrows():
        if index % 10 == 1:      # here to find the first forward rate
                price1 = row["ZCB price"]
                time1 = (row["Maturity"] - TODAY).days / 365
                # if index != 1:
```

```
            #         TODAY += pd.DateOffset(1)
        else:                       # then use forward rate formula
            forward_rate = -math.log(row["ZCB price"]/price1) /
                ((row["Maturity"] - TODAY).days / 365 - time1)
            forward_rates.append(forward_rate)
```



6. Calculate two covariance matrices for the time series of daily log-returns of yield, and forward rates (no spot rates). In other words, first calculate the covariance matrix of the random variables X_i , for $i = 1, ..., 5$, where each random variable Xi has a time series $X_{i,j}$ given by:

$$X_{i,j} = log(r_{i,j+1}/r_{i,j}), j = 1, ..., 9$$

then do the same for the following forward rates - the 1yr-1yr, 1yr-2yr, 1yr-3yr, 1yr-4yr.

For the time series of daily log-returns of yield:

$$COV(X_{ytm}) = \begin{bmatrix} 0.01680712 & 0.01626052 & 0.01617286 & 0.01608614 & 0.01617286 \\ 0.00554018 & 0.01069529 & 0.00533335 & 0.00530505 & 0.0106384 \\ 0.01098912 & 0.00530505 & 0.01058211 & 0.01574836 & 0.01052641 \\ 0. & 0. & 0. & 0. & 0. \\ -0.01098912 & -0.0106384 & -0.01058211 & -0.0104713 & -0.01052641 \\ -0.00554018 & -0.00536194 & -0.00533335 & -0.01058211 & -0.0106384 \\ 0.0165293 & 0.01069529 & 0.00533335 & 0.01583147 & 0.01591546 \\ 0.0054496 & 0.01058211 & 0.01583147 & 0.01041676 & 0.0104713 \\ -0.0054496 & -0.00527706 & -0.01052641 & -0.00519482 & 0. \end{bmatrix}$$

For the forward rates:

$$COV(X_{forward}) = \begin{bmatrix} 1.0167458 & 1.01414425 & 1.01711158 & 1.0184122 \\ 1.00552247 & 1.00283907 & 1.00562822 & 1.00696049 \\ 1.00549382 & 1.00834552 & 1.01302514 & 1.01527393 \\ 0.99464762 & 0.99728277 & 0.99819556 & 0.99867919 \\ 0.98372649 & 0.98898029 & 0.99082827 & 0.98909383 \\ 1.01084646 & 1.00265939 & 0.99249493 & 0.99024086 \\ 0.98938091 & 0.99744796 & 1.01138716 & 1.01695873 \\ 1.0110875 & 1.01116266 & 1.00738538 & 1.00956011 \\ 0.98905997 & 0.98896533 & 0.99633512 & 0.9973826 \end{bmatrix}$$

7. Calculate the eigenvalues and eigenvectors of both covariance matrices.

   For the X_{ytm}, eigenvalues are:

   $$4.87914126e^{-04} \quad 2.13502219e^{-05} \quad 9.20266275e^{-08} \quad 5.41939248e^{-06} \quad 9.60192059e^{-06}$$

   with corresonding eigenvectors:

   $$\begin{bmatrix} -0.43787983 & -0.33348087 & 0.31858176 & 0.74780215 & 0.1906551 \end{bmatrix}^T$$
   $$\begin{bmatrix} -0.40596312 & 0.1837304 & -0.52794914 & 0.24276494 & -0.68100806 \end{bmatrix}^T$$
   $$\begin{bmatrix} -0.4339807 & 0.81538314 & 0.31101527 & -0.07660943 & 0.21026551 \end{bmatrix}^T$$
   $$\begin{bmatrix} -0.49409821 & -0.22343434 & -0.55395895 - 0.29533739 & 0.55843474 \end{bmatrix}^T$$
   $$\begin{bmatrix} -0.45932876 & -0.37451446 & 0.46494554 & -0.53736761 & -0.37923306 \end{bmatrix}^T$$

   For the $X_{forward}$, eigenvalues are:

   $$3.32622733e^{-04} \quad 1.01177525e^{-04} \quad 2.55063341e^{-06} \quad 4.04210512e^{-07}$$

with corresonding eigenvectors:

$$\begin{bmatrix}-0.5191171 & -0.68564354 & -0.50903131 & 0.03602093\end{bmatrix}^T$$

$$\begin{bmatrix}-0.46617698 & -0.26830295 & 0.84096999 & 0.0588389\end{bmatrix}^T$$

$$\begin{bmatrix}-0.47439652 & 0.38162668 & -0.08604378 & -0.78860984\end{bmatrix}^T$$

$$\begin{bmatrix}-0.53679083 & 0.55880904 & -0.16202797 & 0.61101145\end{bmatrix}^T$$

Appendix

Python Code

```
import pandas as pd
import matplotlib.pyplot as plt
import math
import numpy as np


TIME = [i / 2 for i in range(1, 11)]
TODAY = pd.to_datetime("2019-Jan-07")
df = pd.read_csv("a1_data.csv")
selected_bonds = df[(df["Maturity"] == "2019-Mar-01") |
                                        (df["Maturity"] == "2019-Sep-01") |
                                        (df["Maturity"] == "2020-Mar-01") |
                                        (df["Maturity"] == "2020-Sep-01") |
                                        (df["Maturity"] == "2021-Mar-01") |
                                        (df["Maturity"] == "2021-Sep-01") |
                                        (df["Maturity"] == "2022-Mar-01") |
                                        (df["Maturity"] == "2022-Sep-01") |
                                        (df["Maturity"] == "2023-Mar-01") |
                                        (df["Maturity"] == "2023-Sep-01")].
                                    drop_duplicates(subset=["Maturity", "Day"]).
                                    reset_index(drop=True)


# Calculate Accured interest
selected_bonds["Maturity"] = pd.to_datetime(selected_bonds["Maturity"])
selected_bonds['Accured_interest'] = 0

#if Maturity is Mar or Sep, n = 128
selected_bonds['Accured_interest'] = 128 / 365 * selected_bonds['Coupon']

# Calculate Dirty price
selected_bonds['Dirty_price'] = selected_bonds['Accured_interest']
        + selected_bonds['Price']

# Using Bootstrap to calculate spot rate.
pre_sum = 0
pre_t = 0
r1 = 0
spot_rates = []
zcb_price = []
```

```
for index, row in selected_bonds.iterrows():
        if index % 10 == 0:
                pre_sum = 0
                pre_t = (row["Maturity"] - TODAY).days / 365
                # if index != 0:
                #         TODAY += pd.DateOffset(1)

        t = (row["Maturity"] - TODAY).days / 365
        r = -math.log((row["Dirty price"] - row["Coupon"] * pre_sum / 2)/
                (100 + row["Coupon"] / 2)) / t
        spot_rates.append(r)
        zcb_price.append(math.exp(-r * t))
        pre_sum += math.exp(-r * pre_t)
        pre_t = t

selected_bonds["Spot rate"] = spot_rates
selected_bonds["ZCB price"] = zcb_price


# Calculate forward rate
TODAY = pd.to_datetime("2019-Jan-07")
forward_df = selected_bonds[selected_bonds.index.values % 2 == 1]
price1 = 0
time1 = 0
forward_rates = []
for index, row in forward_df.iterrows():
        if index % 10 == 1:
                price1 = row["ZCB price"]
                time1 = (row["Maturity"] - TODAY).days / 365
                # if index != 1:
                #         TODAY += pd.DateOffset(1)
        else:
                forward_rate = -math.log(row["ZCB price"]/price1) /
                        ((row["Maturity"] - TODAY).days / 365 - time1)
                forward_rates.append(forward_rate)

# Calculate Cov(Xi) where Xi,j = log(ri,j+1/ri,j), j = 1,...,9
cov_mat1 = np.empty([9, 5])
cov_mat2 = np.empty([1, 36])
for i in range(1, 10):
        all_ratio = np.log(selected_bonds.loc[selected_bonds['Day'] ==
            i + 1, ['Yield']].values.reshape(1, -1)[0] /
            selected_bonds.loc[selected_bonds['Day'] == i,
            ['Yield']].values.reshape(1, -1)[0])
        cov_mat1[i - 1] = all_ratio[1::2]
```

```
ytm_cov = np.cov(cov_mat1.T)
eig_val1, eig_vec1 = np.linalg.eig(ytm_cov)
for i in range(36):
        cov_mat2[0][i] = forward_rates[i + 4] / forward_rates[i]

forward_cov = np.cov(cov_mat2.reshape(9, 4).T)
eig_val2, eig_vec2 = np.linalg.eig(forward_cov)

# Plot YTM
plt.figure()
for i in range(1, 11):
        day = selected_bonds[selected_bonds["Day"] == i]
        plt.plot(TIME, day["Yield"], marker='*')

plt.legend(["Day " + str(i) for i in range(1, 11)])
plt.xlabel("Time")
plt.ylabel("% Yield")
plt.title('YTM Curve')

# Plot spot rate
plt.figure()
for i in range(1, 11):
        day = selected_bonds[selected_bonds["Day"] == i]
        plt.plot(TIME, day["Spot rate"], marker='o')

plt.legend(["Day " + str(i) for i in range(1, 11)], loc="lower right",
                                                    framealpha=0.3)
plt.xlabel("Time")
plt.ylabel("% Yield")
plt.title('Spot Curve')

# Plot forward rate
plt.figure()
TIME1 = [i for i in range(1, 5)]
for i in range(0, 40, 4):
        plt.plot(TIME1, forward_rates[i:i+4])

plt.legend(["Day " + str(i) for i in range(1, 11)], loc="lower right",
                                                    framealpha=0.3)
plt.xlabel("Time")
plt.ylabel("% Yield")
plt.title('Forward Curve')
plt.show()

# selected_bonds.to_csv("Output.csv", encoding='utf-8', index=False)
```