Draft Notebook for Key Rate Duration calculation in FinancePy

by github.com/sagayev

In [ ]:
```python
from financepy.utils.date import Date
from financepy.utils.day_count import DayCountTypes
from financepy.utils.frequency import FrequencyTypes
from financepy.products.bonds import *
from financepy.market.curves import *
```

In [ ]:
```python
accrual_type, frequencyType, settlementDays, exDiv, calendar = get_bond_market_conventions(
    BondMarkets.UNITED_STATES)

# interest accrues on this date. Issue date is 01/08/2022
issue_date = Date(31, 7, 2022)
maturity_date = Date(31, 7, 2027)
cpn = 2.75/100
bond = Bond(issue_date, maturity_date, cpn, frequencyType, accrual_type)
settlement_date = Date(20, 3, 2023)  # next settle date for this bond
bond
```

Out[ ]:
```
OBJECT TYPE: Bond
ISSUE DATE: 31-JUL-2022
MATURITY DATE: 31-JUL-2027
COUPON (%): 2.75
FREQUENCY: FrequencyTypes.SEMI_ANNUAL
ACCRUAL TYPE: DayCountTypes.ACT_ACT_ICMA
FACE AMOUNT: 100.0
```

			Mnemonic	Description	Ovrd	Value
1)	☑	DU020	KEY_RATE_DUR_3MO	Key Rate Duration 3mo		.003
2)	☑	DU025	KEY_RATE_DUR_6YR	Key Rate Duration 6yr		.000
3)	☑	DU026	KEY_RATE_DUR_6MO	Key Rate Duration 6mo		-.004
4)	☑	DU027	KEY_RATE_DUR_1YR	Key Rate Duration 1yr		-.008
5)	☑	DU028	KEY_RATE_DUR_2YR	Key Rate Duration 2yr		-.018
6)	☑	DU029	KEY_RATE_DUR_3YR	Key Rate Duration 3yr		-.028
7)	☑	DU030	KEY_RATE_DUR_4YR	Key Rate Duration 4yr		2.585
8)	☑	DU031	KEY_RATE_DUR_5YR	Key Rate Duration 5yr		1.497
9)	☑	DU032	KEY_RATE_DUR_7YR	Key Rate Duration 7yr		.000
10)	☑	DU033	KEY_RATE_DUR_10YR	Key Rate Duration 10yr		.000
11)	☑	DU084	KEY_RATE_DUR_8YR	Key Rate Duration 8yr		.000
12)	☑	DU085	KEY_RATE_DUR_9YR	Key Rate Duration 9yr		.000

Data taken on 17 March 2023

In [ ]:
```python
# assign a value to yield to maturity
# US Government Equivalent yield on Bloomberg as of 17 March 2023
ytm = 3.803140/100

# initialize key_rates to None
key_rates = None

# check if key_rates is None
if not key_rates:
    # if it is None, create an array of key rates ranging from 0.08 to 10
    key_rates = np.array([0.25, 0.5,  1,  2,  3, 4,  5, 6, 7, 8, 9, 10])

# calculate the dates for each key rate using the settlement_date
#  variable and the key_rates array
dates = settlement_date.add_years(key_rates)

# set the shift to a small value
shift = 1/10000

# initialize an empty list for the key rate durations
key_rate_durations = []

# iterate over each key rate (tenor) and calculate the key rate duration
for ind, key_rate in enumerate(key_rates):
    # create an array of rates where each rate is equal to the ytm value
```

```
    rates = np.ones(len(key_rates)) * ytm

    # calculate the discount factors using rates (tenors) and key_rates
    # adjust for the bond frequency
    dfs = 1.0 / np.power(1.0 + rates / bond._frequency,
                         bond._frequency * key_rates)

    # create a discount curve using the calculated dfs
    curve = DiscountCurve(settlement_date, dates, dfs,
                          InterpTypes.LINEAR_ZERO_RATES)

    # calculate the full price of the bond using the discount curve
    p_zero = bond.full_price_from_discount_curve(settlement_date, curve)

    # create a discount curve with the key rate
    # shifted up by the shift value
    rates[ind] += shift
    dfs_up = 1.0 / np.power(1.0 + rates / bond._frequency,
                            bond._frequency * key_rates)
    curve_up = DiscountCurve(settlement_date, dates,
                             dfs_up, InterpTypes.LINEAR_ZERO_RATES)

    # calculate the full price of the bond
    # using the discount curve with the key rate shifted up
    p_up = bond.full_price_from_discount_curve(settlement_date, curve_up)

    # create a discount curve with the key rate shifted down
    # by twice the shift value.
    rates[ind] -= shift * 2
    dfs_down = 1.0 / np.power(1.0 + rates / bond._frequency,
                              bond._frequency * key_rates)
    curve_down = DiscountCurve(
        settlement_date, dates, dfs_down, InterpTypes.LINEAR_ZERO_RATES)

    # calculate the full price of the bond using
    # the discount curve with the key rate shifted down
    p_down = bond.full_price_from_discount_curve(settlement_date, curve_down)

    # calculate the key rate duration
    # using the formula (P_down - P_up) / (2 * shift * P_zero)
    key_rate_duration = (p_down - p_up) / (2 * shift * p_zero)

    # append the key rate duration to the key_rate_durations list
    key_rate_durations.append(key_rate_duration)


print("P_zero from curve ", p_zero)
print("Full price from YTM", bond.full_price_from_ytm(settlement_date, ytm))
print("Accrued Interest ", bond.calc_accrued_interest(settlement_date))
print("Macaulay duration", bond.macauley_duration(
    settlement_date, ytm, YTMCalcType.US_STREET))
print("Modified duration ", bond.modified_duration(
    settlement_date, ytm, YTMCalcType.US_STREET))
print("Sum of key rates ", sum(key_rate_durations))
print("Key rates ")
for key_rates, key_rate_durations in zip(key_rates, key_rate_durations):
    print(key_rates, "  ", round(key_rate_durations, 3))


print()
print("Bloomberg Modified duration ", 4.047)
print("Bloomberg sum of key rate durations ", 4.025)
```

```
P_zero from curve  96.1839512734032
Full price from YTM 96.16092289405012
Accrued Interest  0.36464088397790057
Macaulay duration 4.1243096153518115
Modified duration  4.047346488726142
Sum of key rates  4.041129255598644
Key rates
0.25    0.003
0.5    0.005
1.0    0.023
2.0    0.052
3.0    0.075
4.0    2.503
5.0    1.38
6.0    0.0
7.0    0.0
8.0    0.0
9.0    0.0
10.0    0.0

Bloomberg Modified duration  4.047
Bloomberg sum of key rate durations  4.025
```