Fixed Income Securities

Individual Project

Author:

Fernando Tiago (nº 20231535)

A) Build the complete yield curve using interpolation techniques.

I utilized cubic spline interpolation to construct a complete yield curve from the provided Bloomberg EUR yield curve data. Firstly, I performed cubic spline interpolation on the yield curve data:

```
#cubic spline the yield curve
yield_curve = CubicSpline(yield_curve_df.theta, yield_curve_df.rate, bc_type='not-a-knot')
```

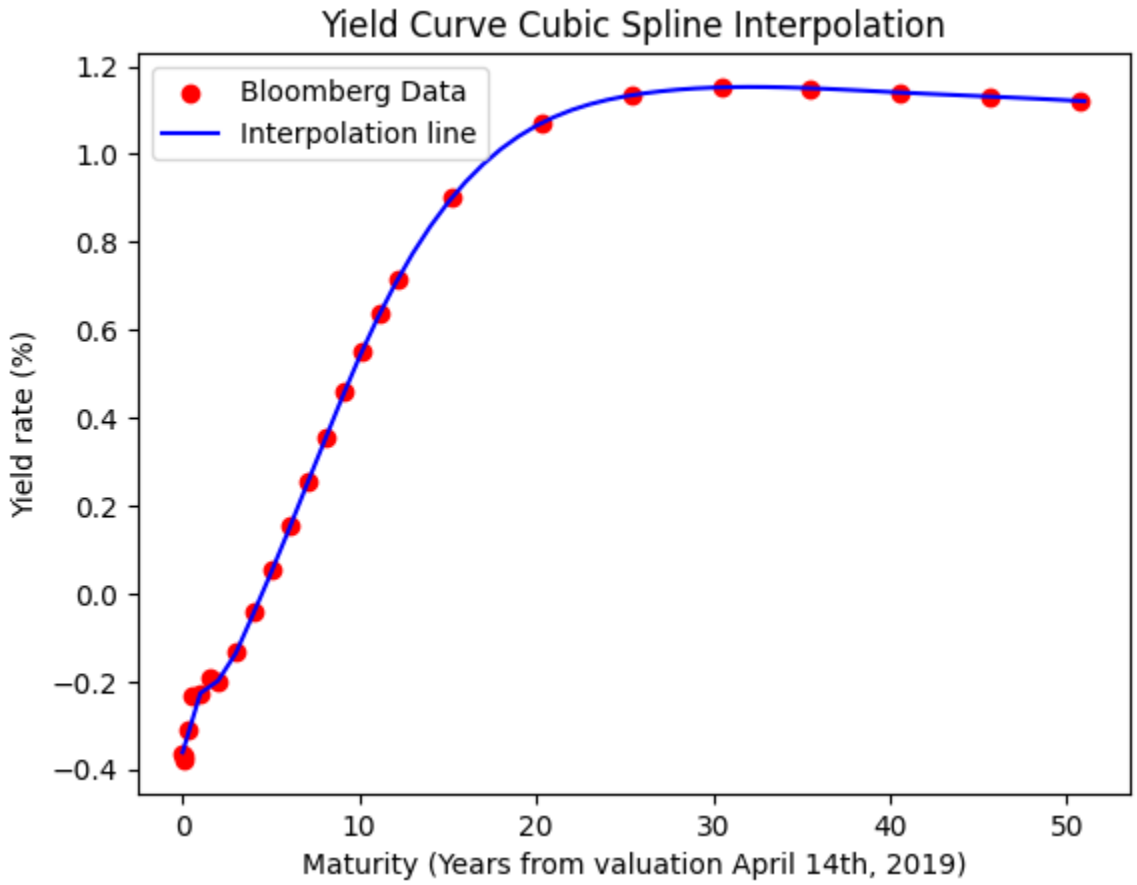
Subsequently, I generated a time axis spanning the range of maturities observed in the data:

```
time_axis = np.arange(0, max(yield_curve_df.theta)+1)
```

Following this, I interpolated the yield rates for each point on the time axis:

```
rate_axis = yield_curve(time_axis)
```

Finally, I plotted the Bloomberg yield curve data as scatter points and the interpolated yield curve as a continuous line:



B) Compute the accrued interest in the fixed and floating legs of the contract.

To compute the accrued interest in the fixed and floating legs of the contract, I utilized the provided code.
For the **fixed leg**, I calculated the accrued interest using the formula:

```
fixed_leg_ai = notional_value * fixed_leg_ai_u/fixed_leg_ai_w * fixed_rate
```

where *"fixed_leg_ai_u"* represents the number of accrued days, *"fixed_leg_ai_w"* represents the day count basis (360), and *"fixed_rate"* is the fixed leg rate.

Similarly, for the **floating leg**, I calculated the accrued interest using the formula:

```
floating_leg_ai = notional_value * floating_leg_ai_u / floating_leg_ai_w * floating_rate / 2
```

where "*floating_leg_ai_u*" represents the number of accrued days, "*floating_leg_ai_w"* represents the day count basis (180, as I used Actual/360), "*floating_rate*" is the floating rate, and I divided by 2 as the floating leg typically pays semiannually.
These computations yielded the accrued interest in the fixed and floating legs of the contract.

Obtaining the following values:

	Fixed Leg	Floating Leg
Value	1379.18	5441.11
Percentage	0.013918	-0.544

C) Calculate the clean (principal) and dirty market value of the swap contract.

Fixed Leg:

To compute the clean market value of the fixed leg, I first calculated the present value (PV) of the fixed leg payments. This involved discounting each fixed leg cash flow to its present value using the corresponding discount factors obtained from the yield curve. Then, I summed up these present values to obtain the fixed leg PV.

```python
def get_fixed_leg_payments(rate):
    fixed_leg_cashflows = []
    coupon_dates = all_coupon_dates[1::2]
    discount_factors = get_discount_factors(yield_curve)[1::2]
    cashflows = discount_factors * rate * notional_value
    #specific cases
    cashflows[0] *= short_discount
    cashflows[-1] += (notional_value * discount_factors[-1])
    return cashflows

fixed_leg_cashflows = get_fixed_leg_payments(fixed_rate)
print(f"Fixed leg payments: {fixed_leg_cashflows}")
fixed_leg_cashflows = np.array(fixed_leg_cashflows)
fixed_leg_pv = fixed_leg_cashflows.sum()
print(f"Fixed leg PV: €{fixed_leg_pv}")
```

The clean market value of the fixed leg is then determined by dividing the fixed leg PV by the notional value of the swap contract and multiplying by 100 to express it as a percentage of the notional value.

```python
print(f"Clean price: {round(fixed_leg_pv/notional_value * 100, 4)}")
print(f"Dirty price: {round((fixed_leg_pv+fixed_leg_ai)/notional_value * 100, 4)}")
```

Obtaining a clean price of 91.2112 and a dirty price of 91.225

Floating Leg:

Similarly, I computed the PV of the floating leg payments by discounting each cash flow using the corresponding discount factors obtained from the yield curve. The sum of these present values yields the floating leg PV.

```
float_leg_cashflows = []
for i, coupon_date in enumerate(reversed(floating_leg_coupon_dates)):
    spot_t = count_years(coupon_date, maturity_date, floating_leg_basis)
    spot_rate = yield_curve(spot_t)
    df = 1 / ((1 + spot_rate / 2) ** (i + 1))

    coupon = euribor_forward_rates[i] / 2 * notional_value
    if i == 0:
        coupon *= short_discount
    elif i == len(fixed_leg_coupon_dates) - 1:
        coupon += notional_value

    float_leg_cashflows.append(coupon * df)

print(f"Float leg payments: {float_leg_cashflows}")
floating_leg_pv = sum(float_leg_cashflows)
print(f"Float leg PV: €{floating_leg_pv}")

✓ 0.0s
```

The clean market value of the floating leg is calculated in the same manner as the fixed leg, by dividing the floating leg PV by the notional value of the swap contract and expressing it as a percentage.

```
print(f"Clean price: {round(floating_leg_pv/notional_value * 100, 4)}")
print(f"Dirty price: {round((floating_leg_pv+floating_leg_ai)/notional_value * 100, 4)}")
✓ 0.0s
```

Obtaining a clean price of 99.8791 and a dirty price of 99.9335.

Swap Contract Value:

The value of the swap contract is determined as the difference between the fixed leg PV and the floating leg PV. This represents the net market value of the swap contract, considering the present values of both legs, this leads to a swap contract value of -866788.90€.

```
swap_value = fixed_leg_pv-floating_leg_pv
print(f"Swap contract: €({swap_value})")
✓ 0.0s
```

D) Estimate the net present value of the contract.

Fixed Leg NPV:

The NPV of the fixed leg is calculated by subtracting the accrued interest from the present value (PV) of the fixed leg payments. This represents the present value of future cash flows for the fixed leg.

```
fixed_leg_npv = fixed_leg_pv - fixed_leg_ai
```

Floating Leg NPV:

Similarly, the NPV of the floating leg is computed by subtracting the accrued interest from the present value (PV) of the floating leg payments. This represents the present value of future cash flows for the floating leg.

```
floating_leg_npv = floating_leg_pv - floating_leg_ai
```

Net Present Value (NPV) of the Contract:

The net present value (NPV) of the contract is determined as the sum of the swap value, the floating leg accrued interest and the fixed leg accrued interest This represents the overall present value of the contract, considering both legs.

```
net_present_value = swap_value + floating_leg_ai + fixed_leg_ai
```

After all this we obtained the vales of 9122496.09€ for the fixed leg net present value, 9993346.92€ for the floating leg net present value, and -859968.60 €for the net present value of the contract.

E) Estimate the swap par rate

The *"find_swap_par_rate"* function is defined to find the swap par rate by calculating the difference between the present value (PV) of the fixed leg payments (obtained using the *"get_fixed_leg_payments"* function) and the PV of the floating leg payments.

```
def find_swap_par_rate(fixed_rate):
    return sum(get_fixed_leg_payments(fixed_rate)) - floating_leg_pv
```

The *"fsolve"* function from the scipy.optimize module is then used to find the root of the equation *"find_swap_par_rate"* using an initial guess value of 6%.

```
swap_par_rate = fsolve(find_swap_par_rate, 6/100)[0]
```

This calculation yields the swap par rate, which is the fixed rate at which the present value of the fixed leg payments equals the present value of the floating leg payments. The result is rounded and expressed as a percentage.

The swap par rate obtained is equal to 0.7597%.

F) Estimate the following IRS Greeks: present value of a one basis point shift (PV01), DV01, Gamma and discuss the interest rate risk of the contract.

Function to Calculate PV01:

The "*calculate_pv01*" function computes the change in present value resulting from a one basis point shift in interest rates. It first calculates the present value *("pv_original")* using the original fixed rate. Then, it calculates the present value ("*pv_shifted*") with a shifted fixed rate, where the rate is increased by one basis point *("basis_point").* Finally, it returns the difference between the shifted and original present values, which represents the PV01.

```
def calculate_pv01(fixed_rate, basis_point=0.0001):
    pv_original = calculate_swap_pv(fixed_rate)

    pv_shifted = calculate_swap_pv(fixed_rate + basis_point)

    return pv_shifted - pv_original
```

Function to Calculate DV01:

The *"calculate_dv01"* function computes the change in the dollar value of the swap contract resulting from a one basis point shift in interest rates. Similar to "*calculate_pv01"*, it calculates the present values with both the original fixed rate and a shifted fixed rate. However, it calculates the DV01 by finding the difference between the original and shifted present values, indicating the absolute change in dollar value.

```
def calculate_dv01(fixed_rate, basis_point=0.0001):
    pv_original = calculate_swap_pv(fixed_rate)

    pv_shifted = calculate_swap_pv(fixed_rate, basis_point)

    return pv_original - pv_shifted
```

Function to Calculate Gamma:

The *"calculate_gamma"* function calculates the second derivative of the PV01 function with respect to the interest rate, providing insights into how the PV01 changes as interest rates vary. It first computes the PV01 ("*pv01_original*") for the original fixed rate. Then, it calculates PV01 for two slightly shifted rates: one higher and one lower. Finally, it applies the second derivative approximation formula to compute Gamma, which quantifies the curvature of the swap's value curve with respect to interest rates.

```
def calculate_gamma(fixed_rate, basis_point=0.0001):
    pv01_original = calculate_pv01(fixed_rate, basis_point)

    pv01_shifted_higher = calculate_pv01(fixed_rate + basis_point, basis_point)

    pv01_shifted_lower = calculate_pv01(fixed_rate - basis_point, basis_point)

    return (pv01_shifted_higher - 2 * pv01_original + pv01_shifted_lower) / (basis_point ** 2)
```

By analyzing these metrics, stakeholders can assess the level of interest rate risk associated with the swap contract. Higher PV01 and DV01, along with positive gamma, suggest elevated interest rate risk, necessitating appropriate risk management strategies. Conversely, lower PV01 and DV01, coupled with negative gamma, indicate lower interest rate risk, offering a more stable outlook for the contract in varying interest rate environments.

Based on the calculations, the interest rate risk of the swap contract can be summarized as follows:

PV01: 12383.937914442271

DV01: 12317.49760125205

Gamma: 0.0

These values indicate a significant sensitivity of the swap contract's present value and dollar value to changes in interest rates, as evidenced by the relatively high PV01 and DV01. However, the gamma value of 0.0 suggests that the curvature of the swap's value curve with respect to interest rates is minimal, indicating a linear relationship between PV01 and interest rate changes. Overall, while the contract exhibits notable interest rate risk in terms of present and dollar value sensitivity, the absence of gamma implies that this risk is primarily linear rather than convex or concave.