1 SA CCR

1.1 Capital requirements for counterparty credit risk

Counterparty credit risk is considered to be a part of credit risk by the regulator. Risk weighted assets have to be calculated and need to be backed by own capital. The three main inputs for calculating credit risk are the probability of default (PD) the loss given default (LGD) and the exposure at default (EAD). Assuming the default of a counterparty over the course of the next year, the EAD is the current estimation of money indebted by the counterparty to the bank at the time of default. Estimating EAD for traditional credit instruments s.a. loans, credit cards, mortgages or bonds is relatively simple. Such instruments do often times have deterministic payment schedules making it easy to predict the exposure in one years time. Credit lines or credit cards behave less deterministic but it is still simple to determine an upper bound to the future exposure by assuming that the entire credit line is exhausted. The counterparty credit risk incurred by derivatives has first been regarded in regulatory capital calculation in Basel II (**?**). Due to the stochastic nature of derivatives EAD calculation for counterparty credit risk has always been regulated separately ever since To calculate the counterparty credit risk associated with derivatives a different approach to calculating

1.2 Calculation of SA CCR

For analysis we create an SA CCR object that implements SA CCR as specified in **?**

When using *SA-CCR* the exposure at default (EAD) has to be calculated as:

$$EAD = \alpha * (RC + PFE)$$

$$\begin{aligned}\text{where} \qquad \alpha &= 1.4\\ RC &: \text{Replacement Cost}\\ PFE &: \text{Potential Future Exposure}\end{aligned}$$

```
[In]: SA_CCR.calculate_sa_ccr_ead(rc = 10, pfe = 20)
```

```
[Out]: 42.0
```

Relation of RC and PFE The purpose of the RC is to assess the imidiate loss suffered by the default of a counterparty. It is based on the current MtM of the derivative less the accessible collateral. If a bank has posted collateral to non-segregated accounts of a counterparty this collateral is also assumed to be lost in case of a default which increases the replacement cost.

The potential future exposure (PFE) on the other hand assesses how the RC might develop in the future. The future being defined as during the next year. If the RC today is 0 but is likely to be larger than 0 in the near future the estimated EAD should take this expected increase in RC into account.

See also Paragraph 130 and 131 of **?**

Paragraph 130 - case without margining:

> For unmargined transactions, the *RC* intends to caputre the loss that would occur if a counterparty were to default and were closed out of its transactions immediately. The *PFE* add-on represents a potential conversative increase in expousre over a one-year time horizon from the present date (i.e. the calculation date).

Paragraph 131 - case with margining:

> For margined trades, the *RC* intends to capture the loss that would occur if a counterparty were to default at the present or at a future time, assuming that the closeout and replacement of transactions occur instantaneously. However, there may be a period (the margin period of risk) between the last exchange of collateral before default and replacement of the trades in the market. The *PFE* add-on represents the potential change in value of the trades during this time period.

Definition of Potential Future Exposure (PFE)

$$PFE = \text{multiplier} * AddOn^{\text{aggregate}}$$

$$\text{where} \qquad AddOn^{\text{aggregate}} : \text{aggregate add-on component}$$
$$\text{multiplier} : f(V, C, AddOn^{\text{aggregate}})$$

AddOn is calculated differently for each asset *a* class. Since no netting is allowed between asset classes the aggregate is calculated as:

$$AddOn^{\text{aggregate}} = \sum_a AddOn^a$$

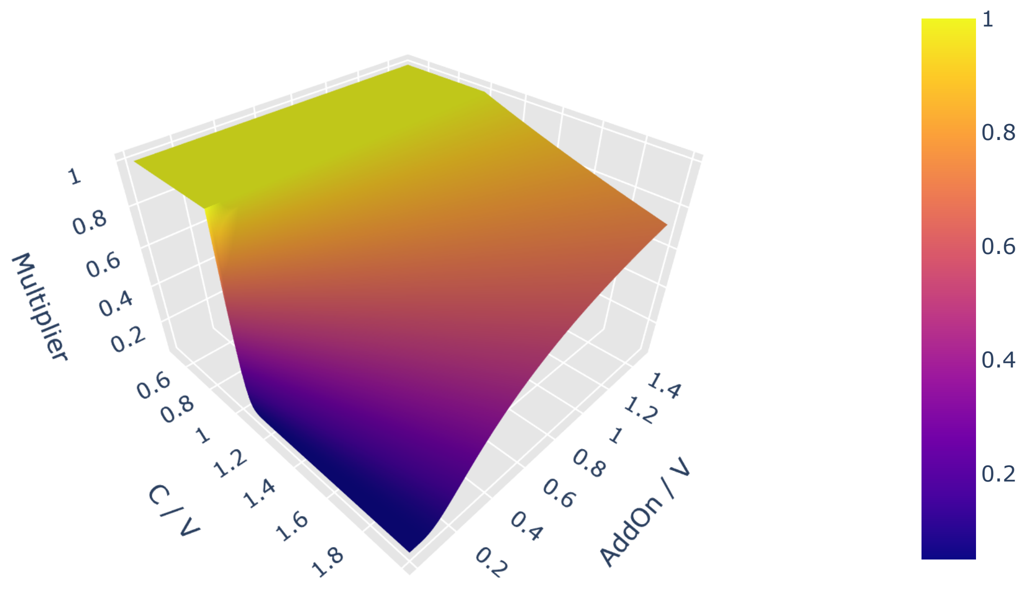
Collateralization is taken into account of the PFE calculation through the multiplier that uses the collateral held as an input. As overcollateralization e.g. through IM increases, the multiplier decreases. However, the multiplier is floored at 5%.

$$\text{multiplier} = \min\left\{1; Floor + (1 - Floor)\exp\left(\frac{V - C}{2(1 - Floor)\,AddOn^{\text{aggregate}}}\right)\right\}$$

$$\text{where} \qquad Floor = 5\%$$

It is important to note that the multiplier can only be below 1 if $C > V$, i.e. if the portfolio is overcollateralized. If the portfolio is overcollateralized, the *AddOn* comes into play. The idea behind the *AddOn* is related to the idea of value at risk. The higher the *AddOn* the faster the SA-CCR model expects the positions to lose in value. Therefore, the higher the value, the higher the multiplier.

In the example below the current NPV of the portfolio is 30 and the received collateral (*IM* + *VM*) is 37. The portfolio is overcollateralized as it should be when initial margin is used. On the other hand, as collateralization decreases e.g. V=C since only VM is exchanged or C=0 in the case of an uncollateralized portfolio the multiplier increases.



1.2.1 AddOn calculation

Most of the SA-CCR logic is hidden inside the AddOn calculation. At first it is important to define the following four data parameters:

M_i

> Maturity of the derivative contract. If the underlying of a derivative is another derivative - e.g. in the case of a swaption the maturity date of the underlying needs to be chosen.

S_i

> For interest rate and credit derivatives the start date of the time periodreferenced by an interst rate or credit contract. If the derivtives underlying is another interest rate or credit intsrument (eg swaption or bond option) S_i is the start date of the underlying instead.

E_i

> Defined as S_i but referencing the end date instead of the start date.

T_i

> For options across all asset classes this is the latest contractual exercise date.

Trade level adjusted notional

Each trade i has a trade level adjusted notional d_i^a assigned to it. This is calculated differently for the different asset classes.

Interest rate and credit derivatives

> The notional of the trade is usually a well defined value in domestic currency for interest rate and credit derivatives. It is multiplied by a supervisory duration factor. The basic idea is, that the value of the derivative can change more the longer the remaining

$$d_i = \text{Notional}_i * SD_i$$

$$\text{where} \qquad SD_i = \frac{\exp\left(-0.05 * S_i\right) - \exp\left(-0.05 * E_i\right)}{0.05}$$

FX derivatives

While the wording in the BCBS paper is a bit more specific we will just assume that every FX traded derivative has a USD leg and set the notional equal the to USD notional.

Equity and commodity derivatives

The notional is defined as the price of the underlying. Therefore, it fluctuates over time.

Notional of exotic derivatives

For more exotic derivatives which do have adjustable notionals, resetting notionals etc. detailed handling of the notional is defined in paragraph 158.

Within this thesis we investigate only equity and interest rate derivatives. For these we can make a few exemplary calculations of the trade level adjusted notional.

For equity trades determining the trade level adjusted notional is trivial as it always is the spot price of the underlying. As an example consider the two trades defined below:

[In]:
```
#When the strike is not set explicitly an at the money option is␣
 ↪created with K = S(t0)
eqOption1 = EquityOption(maturity = ql.Period(1, ql.Years),
                         tradeType= TradeType.CALL,
                         tradeDirection= TradeDirection.LONG,
                         underlying= Stock.ADS)

eqOption2 = EquityOption(maturity = ql.Period(1,   ql.Years),
                         tradeType= TradeType.PUT,
                         tradeDirection= TradeDirection.SHORT,
                         underlying= Stock.ADS,
                         strike = 60)
```

Let the spot price of Adidas stock be 42. Then, the adjusted notional of `eqOption1`, an at the money call on Adidas, is 42 and the adjusted notional of `eqOption2`, a short in the money put on Adidas, is also 42.

For interest rate derivatives such as interest rate swaps or swaptions on the other hand, the notional is adjusted by the supervisory duration factor. As the supervisory duration depends on S and E it is important to understand how these are determined for the different interest rate derivatives.

Trade Type	S	E
Interest Rate Swap	Current date	Maturity date
Forward starting IRS	Start date of the underlying swap	Maturity date of the underlying swap
Swaption	Start date of the underlying swap	Maturity data of the underlying swap

Supervisory delta adjustments

For linear derivatives δ is 1 for long derivatives and -1 for short derivatives.

For options δ is defined as under Black-Scholes:

$$\delta_{\text{long Call}} = +\Phi\left(\frac{\ln\left(P_i/K_i\right) + 0.5 * \sigma_i^2 * T_i}{\sigma_i * \sqrt{T_i}}\right)$$

where Φ : standard normal cdf

σ_i : supervisory volatility as defined in Table 2 in paragraph 183

This delta is multiplied by -1 in case of a long Put option or a short Call option. This formula is used for both, equity options and swaptions.

No detail is given at this point on the delta calculation of CDO tranches as these are not in the scope of this thesis.

In the case of an european equity option the parametrization is quite straight forward.

σ_i: 1.2 is the supervisory volatility for a single stock option

K_i: The strike of the option

P_i: The spot price of the underlying stock

T_i: The maturity of the option

A swaption on the other hand is parametrized as follows for calculation of its supervisory delta:

σ_i: 0.5 is the supervisory volatility for any interst rate option.

K_i: The strike of the option is the fixed rate of the underlying swap

P_i: Is the current par rate of the underlying (forward starting) swap

T_i: The maturity of the option. Please note the difference to E_i used for calculation of the adjusted notional, which is the maturity of the underlying swap.

SA-CCR uses the same Black-Scholes based formula for Swaps as it uses for Equities. It differentiates options in two dimensions. Whether they are *bought* or *sold* and whether they are *Call* or *Put* options (Compare paragraph 159).

SA-CCR defines an option as a call option if it rises in value as the underlying rises in value. A fixed payer swap rises in value as the underlying interest rate rises in value. Therefore, an option to buy a fixed payer swap at a predetermined strike also rises in value as the underlying interest rate rises in value. Therefore, a swaption on a payer swap is considered a *Call* under SA-CCR, while a swaption on a receiver swap is considered a *Put*.

For the at the money option `eqOption1` that was set up above we yield a supervisory delta adjustment of 0.7257.

For an examplary short european swaption that has a par swap as underlying (i.e. the NPV of the swap is 0) that is set up as follows:

[In]:
```
swap = IRS(notional=100,
           timeToSwapStart=ql.Period(1, ql.Years),
           timeToSwapEnd=ql.Period(3, ql.Years),
           swapDirection=SwapDirection.PAYER,
           index=InterestRateIndex.EURIBOR6M
          )


swaption = Swaption(underlyingSwap=swap,
                    optionMaturity=ql.Period(1, ql.Years),
                    tradeDirection=TradeDirection.SHORT)


SA_CCR.calculate_sa_ccr_delta(swaption)
```

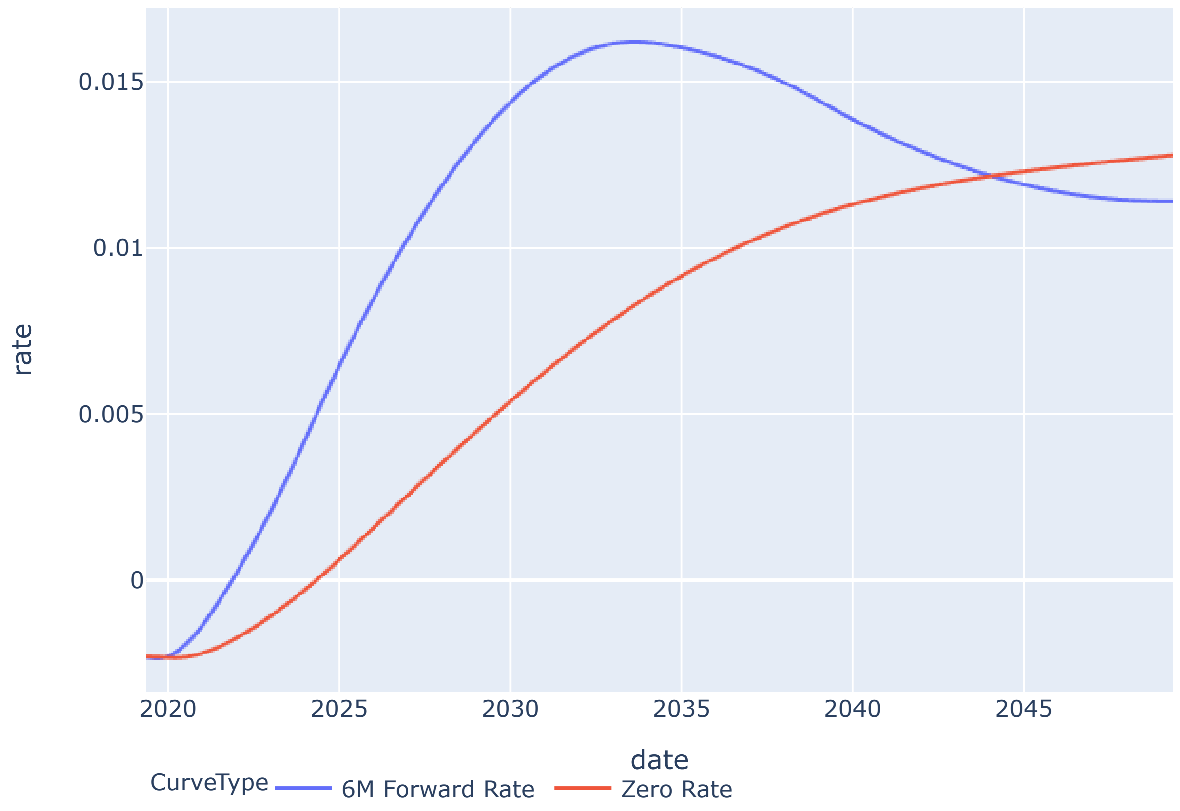
[Out]:
```
-0.5987063256829928
```

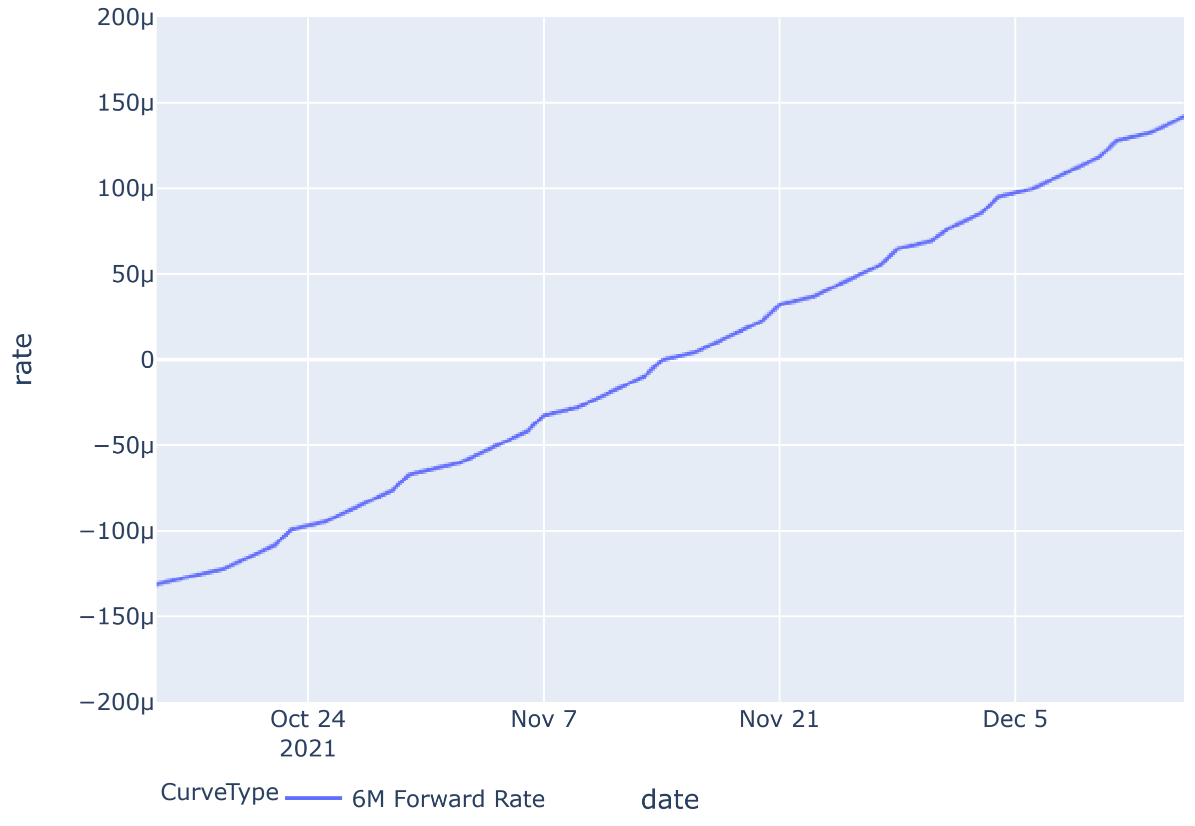
we yield a regulatory delta of -0.5987.

However, the approach of SA-CCR to use the Black-Scholes for interest rate derivatives does have a significant flaw.

Obviously, $\ln(P_i/K_i)$ is not defined, if P_i/K_i is negative. This has, however been very commonplace in recent years, especially for Euro swaptions. One can always set the fixed rate of the underlying swap and therefore K_i as a negative value. But based on the market data of the 10th of May 2019 it is even possible to yield negative par rates for the underlying swap.

As an example we are plotting the zero and the 6M forward curve for the EURIBOR 6M index below. When zooming further in, we can see that that the 6M forward curve crosses 0 on the 15th of November 2021.





To highlight the relationship between the 6M forward curve and the par rate P_i of forward starting swap we can set up the following two swaps depicted below. Since the $EndDate - StartDate$ for these two swaps is six months their par rate or fair price can directly be read from the 6M Forward curve.

[In]:
```
print('StartDate: ' + str(today+ql.Period(30,ql.Months)))
swap1 = IRS(notional=100,
            timeToSwapStart=ql.Period(30,ql.Months),
            timeToSwapEnd=ql.Period(36,ql.Months),
            swapDirection=SwapDirection.PAYER,
            index=InterestRateIndex.EURIBOR6M,
            fixed_rate=0.01
           )
print('ParRate: %.6f%%' % (swap1.get_par_rate()*100))
```

```
StartDate: November 10th, 2021
ParRate: -0.002346%
```

[In]:
```
print('StartDate: ' + str(today+ql.Period(31,ql.Months)))
swap2 = IRS(notional=100,
            timeToSwapStart=ql.Period(31,ql.Months),
            timeToSwapEnd=ql.Period(37, ql.Months),
            swapDirection=SwapDirection.PAYER,
            index=InterestRateIndex.EURIBOR6M,
            fixed_rate=0.01
            )
print('ParRate: %.6f%%' % (swap2.get_par_rate()*100))
```

```
StartDate: December 10th, 2021
ParRate: 0.011976%
```

[In]:
```
swap3 = IRS(notional=100,
            timeToSwapStart=ql.Period(30, ql.Months),
            timeToSwapEnd=ql.Period(36, ql.Months),
            swapDirection=SwapDirection.PAYER,
            index=InterestRateIndex.EURIBOR6M,
            fixed_rate=-0.01)
```

[In]:
```
#calculatin SA_CCR delta for swaptions that convert into these swaps

swaption1 = Swaption(underlyingSwap=swap1,
                     optionMaturity=ql.Period(30, ql.Months),
                     tradeDirection=TradeDirection.LONG)

SA_CCR.calculate_sa_ccr_delta(swaption1)
```

[Out]:
```
0.004138768807253323
```

[In]:
```
swaption2 = Swaption(underlyingSwap=swap2,
                     optionMaturity=ql.Period(31, ql.Months),
                     tradeDirection=TradeDirection.LONG)

SA_CCR.calculate_sa_ccr_delta(swaption2)
```

[Out]:
```
0.005109036476851662
```

The following is a very deep in the money swaption with negative strike and price. The delta should be close to 1. It is however close to zero without offsetting.

```
[In]: swaption3 = Swaption(underlyingSwap=swap3,
                          optionMaturity=ql.Period(30, ql.Months),
                          tradeDirection=TradeDirection.LONG)


      SA_CCR.calculate_sa_ccr_delta(swaption3)
```

```
[Out]: 0.9996934007461944
```

```
[In]: print(swaption1.get_delta())
      print(swaption2.get_delta())
```

```
4.875082678775719
5.248490507272288
```

Risk Horizon For unmargined transaction the margining factor is

$$MF_i^{\text{unmargined}} = \sqrt{\frac{\min\left(M_i; 1 \text{ year}\right)}{1 \text{ year}}}$$

This factor can be used to scale down a risk weight calibrated for a 1 year horizon to a shorter period.

With margining the margin period of risk (MPOR) is:

- 10 business days for small, uncleared OTC portfolios
- 5 business days for cleared derivatives
- 20 business days for netting sets with more than 5000 transactions that are not with a central counterparty
- and doubling this period for portfolios with outstanding disputes

The margining factor is then

$$MF_i^{\text{margined}} = \frac{3}{2}\sqrt{\frac{MPOR_i}{1 \text{ year}}}$$

At this point we need to introduce a collateral agreement object. For simplicities sake we will not differentiate between collateral and netting sets in this thesis. All trades that are covered by the same collateral agreement are also admissible for netting with each other. (Also refer to the introduction of close out netting above). To take into account the different parameters determining the risk horizon a couple of parameters are required to create a collateral agreement. As an example, below we are setting up a collateral agreement for uncleared derivatives without exchange of variation margin or initial margin.

[In]:
```
ca = CollateralAgreement(
        margining=Margining.UNMARGINED,
        clearing=Clearing.UNCLEARED,
        initialMargining=InitialMargining.NO_IM,
        tradecount=Tradecount.UNDER_FIVE_THOUSAND,
        dispute=Dispute.NO_OUTSTANDING_DISPUTES,
        threshold=0.0,      #Threshold to trigger a margin call
        mta=0.0,            #received initial margin
        )
```

With this collateral set object we can define a function for calculation the margining factor:

For trades of differing maturity let's compare the margining factor for the three most common scenarios:

1. No margining
2. Bilateral margining
3. Centrally cleared

[Out]:

	Three days	Two weeks	Six months	One year	Ten years
No margining	0.2000	0.2000	0.7071	1.0000	1.0000
Bilateral margining	0.3000	0.3000	0.3000	0.3000	0.3000
Centrally cleared	0.2121	0.2121	0.2121	0.2121	0.2121

AddOn for interest rate derivatives

Step 1 - calculation of effective notional D_{jk}^{IR}

$$D_{jk}^{IR} = \sum_{i \in \{Ccy_j, MB_k\}} \delta_i * d_i^{IR} * MF_i$$

Here, the notation $i \in \{Ccy_j, MB_k\}$ refers to trades whose underlying is the interest rate of a common currency j and which mature in a common maturity bucket k

We can test our implementation of the effective notional against a small exemplary portfolio in Annex 4a Example 1 of the SA_CCR paper. It consists of the following trades:

Trade #	Nature	Residual Maturity	Base Ccy	Notional (tsd)	Pay Leg	Receive Leg	Market value (tsd)
1	Interest rate swap	10 years	USD	10000	Fixed	Floating	30
2	Interest rate swap	4 years	USD	10000	Floating	Fixed	-20
3	European Swaption	1 into 10 years	EUR	5000	Floating	Fixed	50

To set up this exemplary portfolio we need to find fixed rates for the swaps and underlying swaps to match the desired market values.

Through optimization and using the market data of the 10th of May 2019 the fixed rates to match the market values in Example 1 were identified.

For trade 1 the matching fixed rate is 2.3754%, for trade 1 it is 2.2108% and for the underlying swap of trade 3 it is 0.1610%

[In]:
```
ca=CollateralAgreement()

print(SA_CCR.calculate_sa_ccr_delta(trade_1))
print(SA_CCR.trade_level_adjusted_notional(trade_1))
print(SA_CCR.calculate_sa_ccr_delta(trade_2))
print(SA_CCR.trade_level_adjusted_notional(trade_2))
print(SA_CCR.calculate_sa_ccr_delta(trade_3))
print(SA_CCR.trade_level_adjusted_notional(trade_3))

SA_CCR.interest_rate_addOn(test_portfolio,ca)
```

```
1
78638320.21725275
-1
36198301.54418307
-0.00343382383244269
31712286.360503413
```

```
[Out]: 89019.81894092409
```

```
[In]: export("SA_CCR_ird_addon.ipynb")
```