ORE User Guide

Quaternion Risk Management

7 October 2016

Document History

Date	Author	Comment
7 October 2016	Quaternion Risk Management	initial release

Contents

1 Introduction

The *Open Source Risk Project* [1] aims at providing a transparent platform for pricing and risk analysis that serves as

- a benchmarking, validation, training, and teaching reference,
- an extensible foundation for tailored risk solutions.

Its main software project is *Open Source Risk Engine* (ORE), an application that provides

- a Monte Carlo simulation framework for contemporary risk analytics and value adjustments
- simple interfaces for trade data, market data and system configuration
- simple launchers and result visualisation in Jupyter, Excel, LibreOffice
- unit tests and various examples.

ORE is open source software, provided under the Modified BSD License. It is based on QuantLib, the open source library for quantitative finance [2].

Audience

The project aims at reaching quantitative risk management practitioners (be it in financial institutions, audit firms, consulting companies or regulatory bodies) who are looking for accessible software solutions, and quant developers in charge of the implementation of pricing and risk methods similar to those in ORE. Moreover, the project aims at reaching academics and students who would like to teach or learn quantitative risk management using a freely available, contemporary risk application.

Contributions

Quaternion Risk Management [3] is committed to sponsoring the Open Source Risk project through ongoing project administration, through providing this initial release and a series of subsequent releases in order to achieve a wide analytics, product and risk factor class coverage. The community is invited to contribute to ORE, for example through feedback, discussions and suggested enhancement in the forum on the ORE site [1], as well as contributions of ORE enhancements in the form of source code. See the FAQ section on the ORE site [1] on how to get involved.

Scope and Roadmap

ORE currently provides portfolio pricing, cash flow generation, and a range of contemporary derivative portfolio analytics. The latter are based on a Monte Carlo simulation framework which yields the evolution of various **credit exposure** and **market risk measures**:

- EE aka EPE (Expected Exposure or Expected Positive Exposure)
- ENE (Expected Negative Exposure, i.e. the counterparty's perspective)

- 'Basel' exposure measures relevant for regulatory capital charges under internal model methods
- PFE (Potential Future Exposure at some user defined quantile)
- Value at Risk and Expected Shortfall

and **derivative value adjustments**

- CVA (Credit Value Adjustment)
- DVA (Debit Value Adjustment)
- FVA (Funding Value Adjustment)
- COLVA (Collateral Value Adjustment)
- MVA (Margin Value Adjustment)

for portfolios with netting, variation and initial margin agreements. Subsequent ORE releases will also compute **regulatory capital charges** for counterparty credit risk under the new standardised approach (SA-CCR), and the Monte Carlo based market risk measures will be complemented by parametric methods, e.g. for benchmarking various initial margin calculation models applied in cleared and non-cleared derivatives business.

The first release of ORE in October 2016 covers the simulation of interest rate and FX risk factors and portfolios of Interest Rate Swaps, Caps/Floors, Swaptions, FX Forwards, Cross Currency Swaps and FX Options. Subsequent releases from Q1 2017 onwards will extend the derivative product and the risk factor range to Inflation, Credit, Equity and Commodity. With the introduction of credit risk factors, the scope will also be extended to cover cash products (loans and bonds) and related portfolio analytics.

The simulation models applied in ORE's risk factor evolution implement the models discussed in detail in *Modern Derivatives Pricing and Credit Exposure Analysis* [17]: The IR/FX risk factor evolution is based on a cross currency model consisting of an arbitrage free combination of Linear Gauss Markov models for all interest rates and lognormal processes for the FX rates. The model components are calibrated to cross currency discounting and forward curves, Swaptions and FX Options.

Further Resources

- Open Source Risk Project site: `http://www.opensourcerisk.org`
- Frequently Asked Questions: `http://www.opensourcerisk.org/faqs`
- Forum: `http://www.opensourcerisk.org/forum`
- Source code and releases: `https://github.com/opensourcerisk/engine`
- Follow ORE on Twitter `@OpenSourceRisk` for updates on releases and events

An ORE Book will follow in 2017 that will elaborate on the engine's design and make the connection between methodology and implementation. It will be announced in due course on the channels mentioned above.

Organisation of this document

This document focuses on instructions how to use ORE to cover basic workflows from individual deal analysis to portfolio processing. After an overview over the core ORE data flow in section 2 and installation instructions in section 3 we start in section 4 with a series of examples that illustrate how to launch ORE using its command line application, and we discuss typical results and reports. We then illustrate in section 5 interactive analysis of resulting 'NPV cube' data. The final sections of this text document ORE parametrisation and the structure of trade and market data input.

2 ORE Data Flow

The core processing steps followed in ORE to produce risk analytics results are sketched in Figure 1. All ORE calculations and output are generated in three fundamental process steps as indicated in the three boxes in the upper part of the figure. In each of these steps appropriate data (described below) is loaded and results are generated, either in form of a human readable report, or in an intermediate step as pure data files (e.g. NPV data, exposure data).

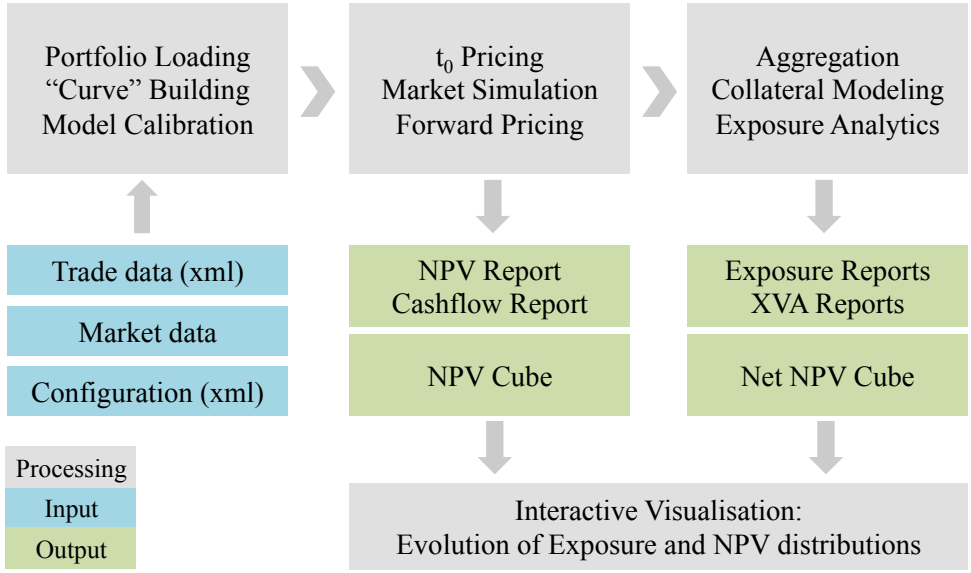

Figure 1: Sketch of the ORE process, inputs and outputs.

The overall ORE process needs to be parametrised using a set of configuration XML files which is the subject of section 6. The portfolio is provided in XML format which is explained in detail in sections 7 and 8. Note that ORE comes with 'Schema' files for all supported products so that any portfolio xml file can be validated before running through ORE. Market data is provided in a simple three-column text file with unique human-readable labelling of market data points, as explained in section 9.
The first processing step (upper left box) then comprises

- loading the portfolio to be analysed,
- building any yield curves or other 'term structures' needed for pricing,
- calibration of pricing and simulation models.

The second processing step (upper middle box) is then

- portfolio valuation, cash flow generation,
- going forward - conventional risk analysis such as sensitivity analysis and stress testing, standard-rule capital calculations such as SA-CCR, etc,
- and in particular, more time-consuming, the market simulation and portfolio valuation through time under Monte Carlo scenarios.

This process step produces several reports (NPV, cashflows etc) and in particular an **NPV cube**, i.e. NPVs per trade, scenario and future evaluation date. The cube is written to a file in both condensed binary and human-readable text format.
The third processing step (upper right box) performs more 'sophisticated' risk analysis by post-processing the NPV cube data:

- aggregating over trades per netting set,
- applying collateral rules to compute simulated variation margin as well as simulated (dynamic) initial margin posting,
- computing various XVAs including CVA, DVA, FVA, MVA for all netting sets, with and without taking collateral (variation and initial margin) into account, on demand with allocation to the trade level.

The output of this process step are XVA reports and the 'net' NPV cube, i.e. after aggregation, netting and collateral.
The example section 4 demonstrates for representative product types how the described processing steps can be combined in a simple batch process which produces the mentioned reports, output files and exposure evolution graphs in one 'go'.
Moreover, both NPV cubes can be further analysed interactively using a visualisation tool introduced in section 5.1. And finally, sections 5.2 and 5.3 demonstrate how ORE processes can be launched in spreadsheets and key results presented automatically within the same sheet.

3 Getting and Building ORE

You can get ORE in two ways, either by downloading a release bundle as described in section 3.1 or by checking out the source code from the github repository as described in section 3.2.

3.1 ORE Releases

ORE releases are regularly provided in the form of source code archives, Windows executables `ore.exe`, example cases and documentation. Release archives will be provided at `https://github.com/opensourcerisk/engine/releases`.

The release consists of a single archive in zip format

- `ORE-1.8.zip`

When unpacked, it creates a directory `ORE-1.8` with the following files respectively subdirectories

1. App/
2. Docs/
3. Examples/
4. OREAnalytics/
5. OREData/
6. QuantExt/
7. ThirdPartyLibs/
8. userguide.pdf
9. xsd

The first three items and userguide.pdf are sufficient to run the compiled ORE application on the list of examples described in the user guide (this works on Windows only). The Windows executables are located in App/bin/Win32/Release/ respectively App/bin/x64/Release/. To continue with the compiled executables:

- Ensure that the scripting language Python is installed on your computer, see also section 3.3 below;
- Move on to the examples in section 4.

To build ORE from sources:

- Set up Boost as described in section 3.2.2, unless already installed
- Set up QuantLib 1.8 [2, 4] from its github or sourceforge download page, unless already installed; QuantLib needs to be located in this project directory ORE-1.8. Alternatively, you can create a symbolic link named QuantLib here that points to the actual QuantLib directory
- Build QuantExt, OREData, OREAnalytics, App (in this order) as described in section 3.2.3
- Note that ThirdPartyLibs does not need to be built, it contains RapdidXml, header only code for reading and writing XML files
- Move on to section 3.3 and the examples in section 4.

Open Docs/html/index.html to see the API documentation for QuantExt, OREData and OREAnalytics, generated by doxygen.

3.2 Building ORE

ORE's source code is hosted on github.com at https://github.com/opensourcerisk/engine using git, a free and open source distributed version control system.

3.2.1 Git

To access the current code base on GitHub, one needs to get `git` installed first.

1. Install and setup Git on your machine following instructions at [5]
2. Fetch ORE from github by running the following:

   ```
   % git clone https://github.com/opensourcerisk/engine.git ore
   ```

   This will create a folder 'ore' in your current directory that contains the codebase.
3. Initially, the QuantLib subdirectory under `ore` is empty as it is a submodule pointing to the official QuantLib repository. To pull down locally, use the following commands:

   ```
   % cd ore
   % git submodule init
   % git submodule update
   ```

3.2.2 Boost

QuantLib and ORE depend on the boost C++ libraries. Hence these need to be installed before building QuantLib and ORE. With Unix (Linux, OS X), we recommend boost version 1_55 or higher, with Windows we recommend boost version 1_57 or higher. Older versions may work on some platforms and system configurations, but were not tested.

Windows

1. Download the pre-compiled binaries for MSVC-14 (MSVC2015) from [6]
   - 32-bit: [6]\VERSION\boost_VERSION-msvc-14.0-32.exe\download
   - 64-bit: [6]\VERSION\boost_VERSION-msvc-14.0-64.exe\download
2. Start the installation file and choose an installation folder. Take a note of that folder as it will be needed later on.
3. Finish the installation by clicking Next a couple of times.

Alternatively, compile all Boost libraries directly from the source code:

1. Open a Visual Studio Tools Command Prompt
   - 32-bit: VS2015/VS2013 x86 Native Tools Command Prompt
   - 64-bit: VS2015/VS2013 x86 x64 Cross Tools Command Prompt
2. Navigate to the boost root directory
3. Run bootstrap.bat
4. Build the libraries from the source code
   - 32-bit:

     ```
     .\b2 --stagedir=.\lib\Win32\lib --build-type=complete toolset=msvc-14.0 \
     --address-model=32 --with-test --with-system --with-filesystem \
     --with-serialization --with-regex --with-date_time stage
     ```

- 64-bit:

```
.\b2 --stagedir=.\lib\x64\lib --build-type=complete toolset=msvc-14.0 \
--address-model=64 --with-test --with-system --with-filesystem \
--with-serialization --with-regex --with-date_time stage
```

Unix

1. Download Boost from [7] and build following the instructions on the site
2. Define the environment variable BOOST that points to the boost directory (so includes should be in BOOST and libs should be in BOOST/stage/lib)

3.2.3 ORE Libraries and Application

Windows

1. Download and install Visual Studio Community Edition. During the installation, make sure you install the Visual C++ support under the Programming Languages features (disabled by default).
2. To configure the boost paths in Visual Studio open any of the Visual Studio solution files in item 3 below and select View → Other Windows → Property Manager. It does not matter which solution you open, if it is for example the QuantExt solution you should see two Projects 'QuantExt' and 'quantexttest-suite' in the property manager. Expand any of them (e.g. QuantExt) and then one of the Win32 or x64 configurations. The settings will be specific for the Win32 or x64 configuration but otherwise it does not matter which of the projects or configurations you expand, they all contain the same configuration file. You should now see 'Microsoft.Cpp.Win32.user' respectively 'Microsoft.Cpp.x64.user' depending on whether you chose a Win32 or a x64 configuration. Click on this file to open the property pages. Select VC++ Directories and then add your boost directory to the 'Include Directories' entry. Likewise add your boost library directory to the 'Library Directories' entry. If for example your boost installation is in `C:\boost_1_57_0` and the libraries reside in the `stage\lib` subfolder, add `C:\boost_1_57_0` to the 'Include Directories' entry and `C:\boost_1_57_0\stage\lib` to the 'Library Directories' entry. Press OK. If you want to configure the boost paths for Win32 resp. x64 as well, repeat the previous step for 'Microsoft.Cpp.Win32.user' respectively 'Microsoft.Cpp.x64.user'. To complete the configuration just close the property manager window.
3. Open each of the sub-projects and compile them in the following order: QuantLib, QuantExt, ORData, OREAnalytics and App. For each project, do the following:
   - Switch to the correct platform (i.e. Win32 or x64) from the Configuration Manager. The selection should match the pre-compiled version of Boost. Trying to compile using a mixed configuration (e.g. Boost 64-bit and 32-bit QuantLib) will fail.
   - Compile the project: Build → Build Solution
   - Once the compilation is complete, run the test suite.

Unix

1. Build QuantLib as usual.

   ```
   % cd QuantLib
   % ./autogen.sh
   % ./configure --with-boost-include=$BOOST --with-boost-lib=$BOOST/stage/lib
   % make -j4
   ```

2. Build QuantExt

   ```
   % cd QuantExt
   % ./autogen.sh
   % ./configure
   % make -j4
   ```

   This will build both the QuantExt library and test suite.

3. Run the test suite

   ```
   % ./test/quantext-test-suite
   ```

4. Build OREData, OREAnalytics and their test suites.

   Follow the same steps as for QuantExt. To run the unit test suites, do

   ```
   % ./test/ored-test-suite
   ```

   and

   ```
   % ./test/orea-test-suite
   ```

   in the respective library directories.

5. Build App/ore

   ```
   % cd App
   % ./autogen.sh
   % ./configure
   % make -j4
   ```

   Note: On Linux systems, the 'locale' settings can negatively affect the ORE process and output. To avoid this, we recommend setting the environment variable `LC_NUMERIC` to `C`, e.g. in a bash shell, do

   ```
   % export LC_NUMERIC=C
   ```

   before running ORE or any of the examples below. This will suppress thousand separators in numbers when converted to strings.

6. Run Examples (see section 4)

   ```
   % cd Examples/Example_1
   % python run.py
   ```

3.3 Python and Jupyter

Python (version 3.5 or higher) is required to run the examples in section 4 and plot exposure evolutions. Moreover, we use Jupyter [8] in section 5 to visualise simulation results. Both are part of the 'Anaconda Open Data Science Analytics Platform' [9].

Anaconda installation instructions for Windows, OS X and Linux are available on the Anaconda site, with graphical installers for Windows[1], Linux and OS X.
With Linux and OS X, the following environment variable settings are required

- set `LANG` and `LC_ALL` to `en_US.UTF-8` or `en_GB.UTF-8`
- set `LC_NUMERIC` to `C`.

The former is required for both running the Python scripts in the examples section, as well as successful installation of the following packages.
The full functionality of the Jupyter notebook introduced in section 5.1 requires furthermore installing

- jupyter_dashboards: `https://github.com/jupyter-incubator/dashboards`
- ipywidgets: `https://github.com/ipython/ipywidgets`
- pythreejs: `https://github.com/jovyan/pythreejs`
- bqplot: `https://github.com/bloomberg/bqplot`

With Python and Anaconda already installed, this can be done by running these commands

- `conda install -c conda-forge ipywidgets`
- `pip install jupyter_dashboards`
- `jupyter dashboards quick-setup --sys-prefix`
- `conda install -c conda-forge bqplot`
- `conda install -c conda-forge pythreejs`

Note that the bqplot installation requires the environment settings mentioned above.

4 Examples

The examples shown in table 1 are intended to help with getting started with ORE, and to serve as plausibility checks for the simulation results generated with ORE.
All example results can be produced with the Python scripts `run.py` in the ORE release's `Examples/Example_#` folders which work on both Windows and Unix platforms. In a nutshell, all scripts call ORE's command line application with a single input XML file

```
ore[.exe] ore.xml
```

They produce a number of standard reports and exposure graphs in PDF format. The structure of the input file and of the portfolio, market and other configuration files referred to therein will be explained in section 6.

ORE is driven by a number of input files, listed in table 2 and explained in detail in sections 6 to 10. In all examples, these input files are either located in the example's sub directory `Examples/Example_#/Input` or the main input directory `Examples/Input` if

Example	Description
1	Vanilla at-the-money Swap with flat yield curve
2	Vanilla Swap with normal yield curve
3	European Swaption
4	Bermudan Swaption
5	Callable Swap
6	Cap/Floor
7	FX Forward European FX Option
8	Cross Currency Swap without notional reset
9	Cross Currency Swap with notional reset
10	Three-Swap portfolio with netting and collateral XVAs - CVA, DVA, FVA, MVA, COLVA Exposure and XVA Allocation to trade level
11	Basel exposure measures - EE, EPE, EEPE
12	Long term simulation with horizon shift
13	Dynamic Initial Margin and MVA

Table 1: ORE examples.

File Name	Description
`ore.xml`	Master input file, selection of further inputs below and selection of analytics
`portfolio.xml`	Trade data
`netting.xml`	Collateral (CSA) data
`simulation.xml`	Configuration of simulation model and market
`market.txt`	Market data snapshot
`fixings.txt`	Index fixing history
`curveconfig.xml`	Curve and term structure composition from individual market instruments
`conventions.xml`	Market conventions for all market data points
`todaysmarket.xml`	Configuration of the market composition, relevant for the pricing of the given portfolio as of today (yield curves, FX rates, volatility surfaces etc)
`pricingengines.xml`	Configuration of pricing methods by product

Table 2: ORE input files

used across several examples. The particular selection of input files is determined by the 'master' input file `ore.xml`.
The typical list of output files and reports is shown in table 3. The names of output files can be configured through the master input file `ore.xml`. Whether these reports are generated also depends on the setting in `ore.xml`. For the examples, all output will be written to the directory `Examples/Example_#/Output`.

File Name	Description
`npv.csv`	NPV report
`flows.csv`	Cashflow report
`curves.csv`	Generated yield (discount) curves report
`xva.csv`	XVA report, value adjustments at netting set and trade level
`exposure_trade_*.csv`	Trade exposure evolution reports
`exposure_nettingset_*.csv`	Netting set exposure evolution reports
`rawcube.csv`	NPV cube in readable text format
`netcube.csv`	NPV cube after netting and colateral, in readable text format
`*.dat`	Intermediate storage of NPV cube and scenario data in binary format
`*.pdf`	Exposure graphics produced by the python script `run.py` after ORE completed

Table 3: ORE output files

Note: When building ORE from sources on Windows platforms, make sure that you copy

[1]With Windows, after a fresh installation of Python the user may have to run the `python` command once in a command shell so that the Python executable will be found subsequently when running the example scripts in section 4.

your `ore.exe` to the binary directory `bin/win32/` respectively `bin/x64/`. Otherwise the examples may be run using the pre-compiled executables which come with the ORE release.

4.1 Interest Rate Swap Exposure

We start with a vanilla single currency Swap (currency EUR, maturity 20y, notional 10m, receive fixed 2% annual, pay 6M-Euribor flat). The market yield curves (for both discounting and forward projection) are set to be flat at 2% for all maturities, i.e. the Swap is at the money initially and remains at the money on average throughout its life. Running ORE in directory `Examples/Example_1` with

```
python run.py
```

yields the exposure evolution in

```
Examples/Example_1/Output/*.pdf
```

and shown in figure 2. Both Swap simulation and Swaption pricing are run with calls

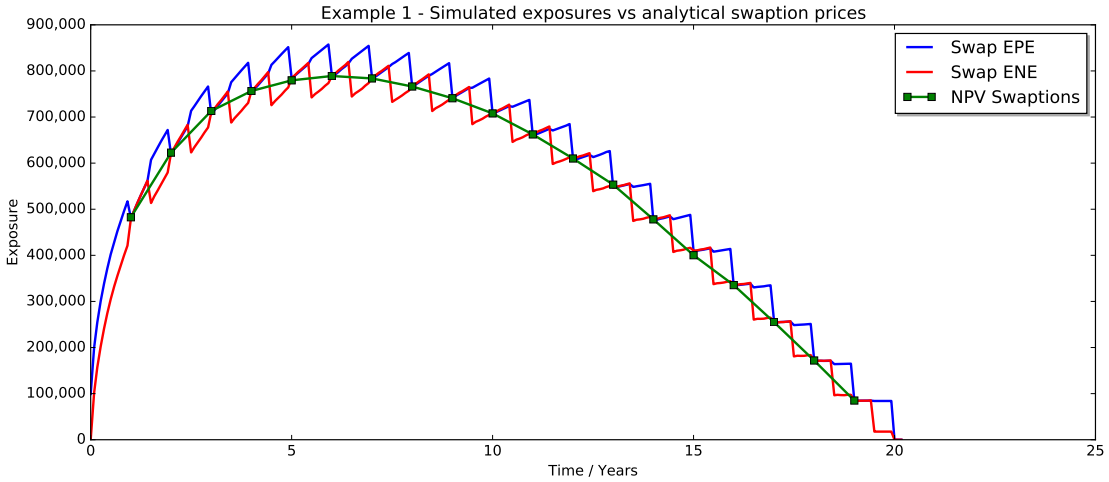

Figure 2: Vanilla ATM Swap expected exposure in a flat market environment from both parties' perspectives. The symbols are European Swaption prices. The simulation was run with monthly time steps. To demonstrate the convergence of EPE and ENE profiles we have used an exceptionally large number of samples (100,000) to produce this graph. A similar outcome can be obtained more quickly with 5,000 samples on a quarterly time grid which is the default setting of Example_1.

to the ORE executable, essentially

```
ore[.exe] ore.xml
ore[.exe] ore_swaption.xml
```

which are wrapped into the script `Examples/Example_1/run.py` provided with the ORE release. It is instructive to look into the input folder in Examples/Example_1, the content of the main input file `ore.xml`, together with the explanations in section 6.
This simple example is an important test case which is also run similarly in one of the unit test suites of ORE. The expected exposure can be seen as a European option on the underlying netting set, see also appendix A.2. In this example, the expected exposure at

some future point in time, say 10 years, is equal to the European Swaption price for an option with expiry in 10 years, underlying Swap start in 10 years and underlying Swap maturity in 20 years. We can easily compute such standard European Swaption prices for all future points in time where both Swap legs reset, i.e. annually in this case[2]. And if the simulation model has been calibrated to the points on the Swaption surface which are used for European Swaption pricing, then we can expect to see that the simulated exposure matches Swaption prices at these annual points, as in figure 2. In Example_1 we used co-terminal ATM Swaptions for both model calibration and Swaption pricing. Moreover, as the the yield curve is flat in this example, the exposures from both parties' perspectives (EPE and ENE) match not only at the annual resets, but also for the period between annual reset of both legs to the point in time when the floating leg resets. Thereafter, between floating leg (only) reset and next joint fixed/floating leg reset, we see and expect a deviation of the two exposure profiles.

Moving to `Examples/Example_2`, we see what changes when using a realistic (non-flat) market environment. Running the example with

```
python run.py
```

yields the exposure evolution in

```
Examples/Example_2/Output/*.pdf
```

shown in figure 3. In this case, where the curves (discount and forward) are upward

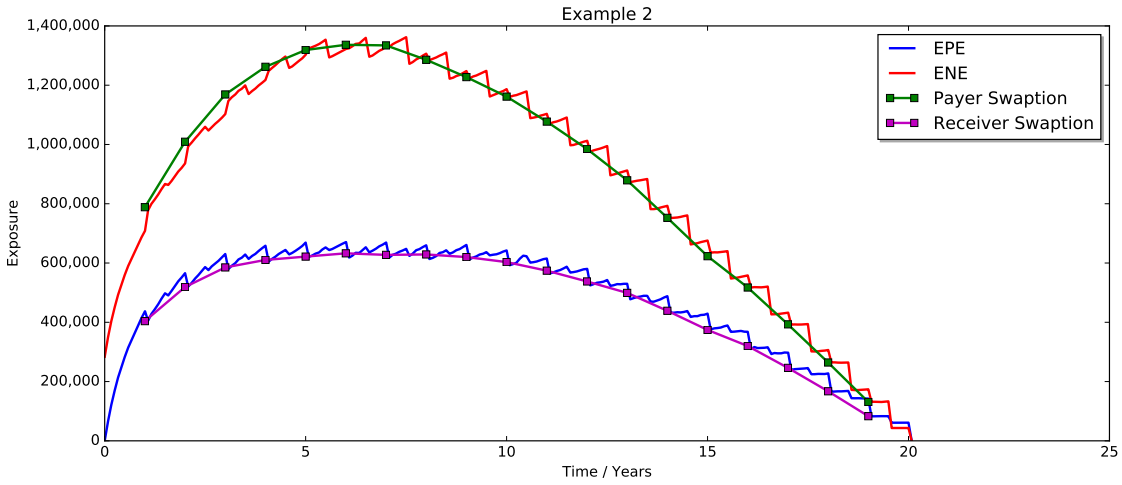

Figure 3: Vanilla ATM Swap expected exposure in a realistic market environment as of 05/02/2016 from both parties' perspectives. The Swap is the same as in figure 2 but receiving fixed 1%, roughly at the money. The symbols are the prices of European payer and receiver Swaptions. Simulation with 5000 paths and monthly time steps.

sloping, the receiver Swap is at the money at inception only and moves (on average) out of the money during its life. Similarly, the Swap moves into the money from the counterparty's perspective. Hence the expected exposure evolutions from our perspective (EPE) and the counterparty's perspective (ENE) 'detach' here, while both can still be be reconciled with payer or respectively receiver Swaption prices.

[2]Using closed form expressions for standard European Swaption prices.

4.2 European Swaption Exposure

This demo case in folder `Examples/Example_3` shows the exposure evolution of European Swaptions with cash and physical delivery, respectively, see figure 4. The delivery type

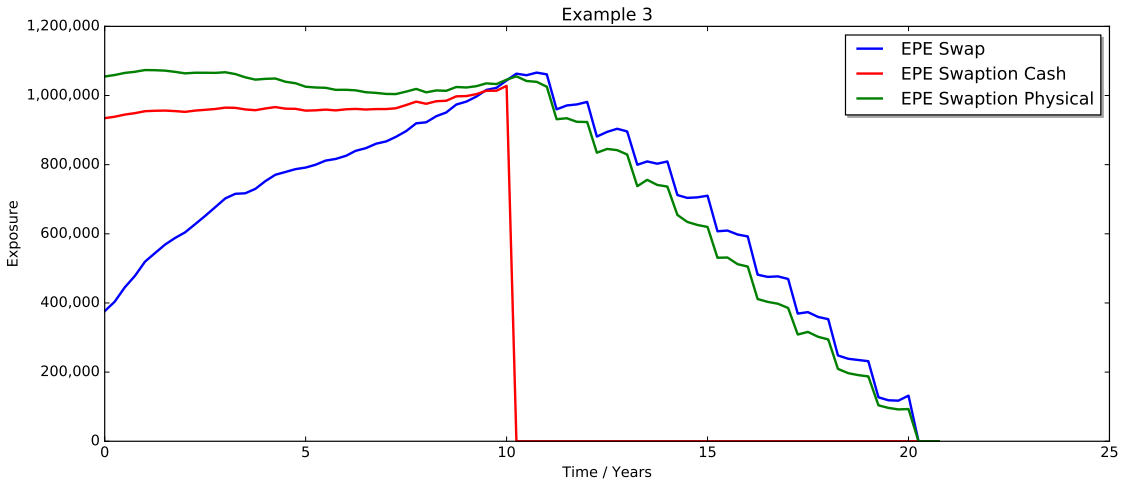

Figure 4: European Swaption exposure evolution, expiry in 10 years, final maturity in 20 years, for cash and physical delivery. Simulation with 1000 paths and quarterly time steps.

(cash vs physical) yields significantly different valuations as of today due to the steepness of the relevant yield curves (EUR). The cash settled Swaption's exposure graph is truncated at the exercise date, whereas the physically settled Swaption exposure turns into a Swap-like exposure after expiry. For comparison, the example also provides the exposure evolution of the underlying forward starting Swap which yields a somewhat higher exposure after the forward start date than the physically settled Swaption. This is due to scenarios with negative Swap NPV at expiry (hence not exercised) and positive NPVs thereafter.

4.3 Bermudan Swaption Exposure

This demo case in folder `Examples/Example_4` shows the exposure evolution of Bermudan rather than European Swaptions with cash and physical delivery, respectively, see figure 5. The underlying Swap is the same as in the European Swaption example in section 4.2. Note in particular the difference between the Bermudan and European Swaption exposures with cash settlement: The Bermudan shows the typical step-wise decrease due to the series of exercise dates. Also note that we are using the same Bermudan option pricing engines for both settlement types, in contrast to the European case, so that the Bermudan option cash and physical exposures are identical up to the first exercise date. When running this example, you will notice the significant difference in computation time compared to the European case (ballpark 30 minutes here for 2 Swaptions, 1000 samples, 90 time steps). The Bermudan example takes significantly more computation time because we use an LGM grid engine for pricing under scenarios in this case. In a realistic context one would more likely resort to American Monte Carlo simulation, feasible in ORE, but not provided in the first release. However, this implementation can be used to benchmark any faster / more sophisticated approach to Bermudan Swaption exposure simulation.

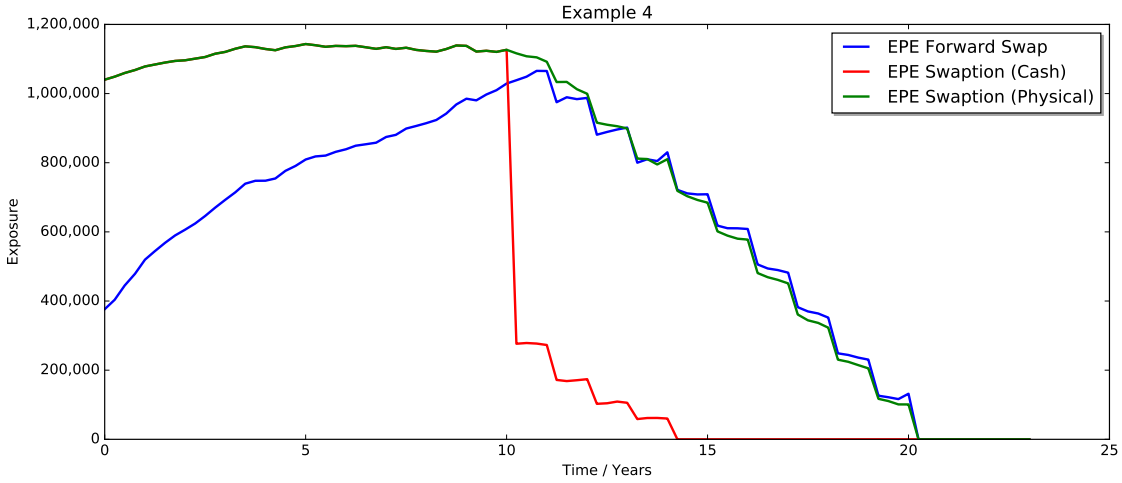

Figure 5: Bermudan Swaption exposure evolution, 5 annual exercise dates starting in 10 years, final maturity in 20 years, for cash and physical delivery. Simulation with 1000 paths and quarterly time steps.

4.4 Callable Swap Exposure

This demo case in folder `Examples/Example_5` shows the exposure evolution of a European callable Swap, represented as two trades - the non-callable Swap and a Swaption with physical delivery. We have sold the call option, i.e. the Swaption is a right for the counterparty to enter into an offsetting Swap which economically terminates all future flows if exercised. The resulting exposure evolutions for the individual components (Swap, Swaption), as well as the callable Swap are shown in figure 6. The example is an extreme case where the underlying Swap is deeply in the money (receiving fixed 5%), and hence the call exercise probability is close to one. Modify the Swap and Swaption fixed rates closer to the money ($\approx 1\%$) to see the deviation between net exposure of the callable Swap and the exposure of a 'short' Swap with maturity on exercise.

4.5 Cap/Floor Exposure

The example in folder `Examples/Example_6` generates exposure evolutions of several Swaps, caps and floors. The example shown in figure 7 ('portfolio 1') consists of a 20y Swap receiving 1% fixed and a 20y Collar with both cap and floor at 2.5% so that the net exposure corresponds to a Swap receiving 0.5% fixed.
The second example in this folder shown in figure 8 ('portfolio 2') consists of a short Cap, long Floor and a Collar that corresponds to the netted Cap and Floor.
Further three test portfolios are provided as part of this example. Run the example and inspect the respective output directories `Examples/Example_7/Output/portfolio_#`. Note that these directories have to be present/created before running the batch with `python run.py`.

4.6 FX Forward Exposure

The example in folder `Examples/Example_7` generates the exposure evolution for a EUR / USD FX Forward transaction with value date in 10Y. This is a particularly simple

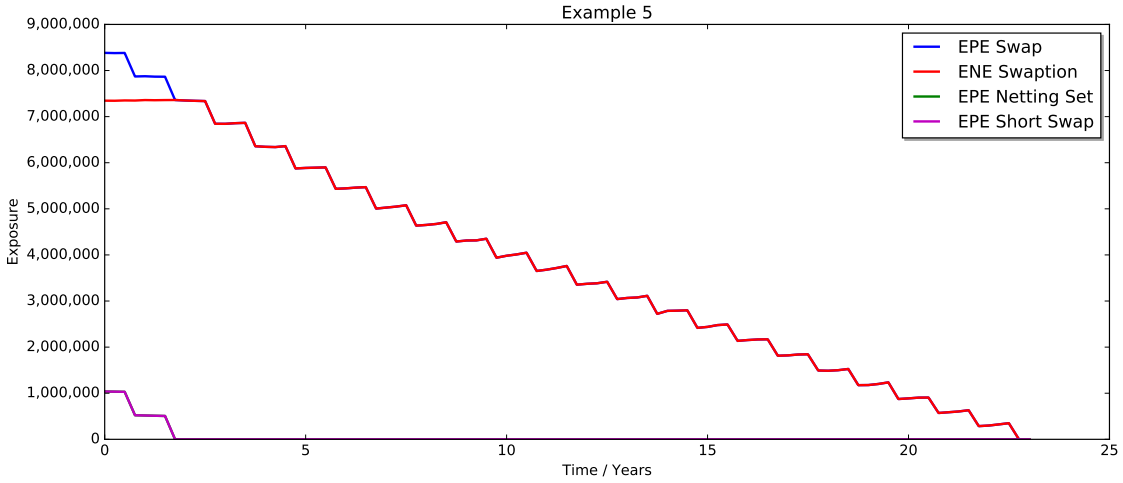

Figure 6: European callable Swap represented as a package consisiting of non-callable Swap and Swaption. The Swaption has physical delivery and offsets all future Swap cash flows if exercised. The exposure evolution of the package is shown here as 'EPE NettingSet' (green line). This is covered by the pink line, the exposure evolution of the same Swap but with maturity on the exercise date. The graphs match perfectly here, because the example Swap is deep in the money and exercise probability is close to one. Simulation with 5000 paths and quarterly time steps.

Figure 7: Swap+Collar, portfolio 1. The Collar has identical cap and floor rates at 2.5% so that it corresponds to a fixed leg which reduces the exposure of the Swap, which receives 3% fixed. Simulation with 1000 paths and quarterly time steps.

show case because of the single cash flow in 10Y. On the other hand it checks the cross currency model implementation by means of comparison to analytic limits - EPE and ENE at the trade's value date must match corresponding Vanilla FX Option prices, as shown in figure 9.

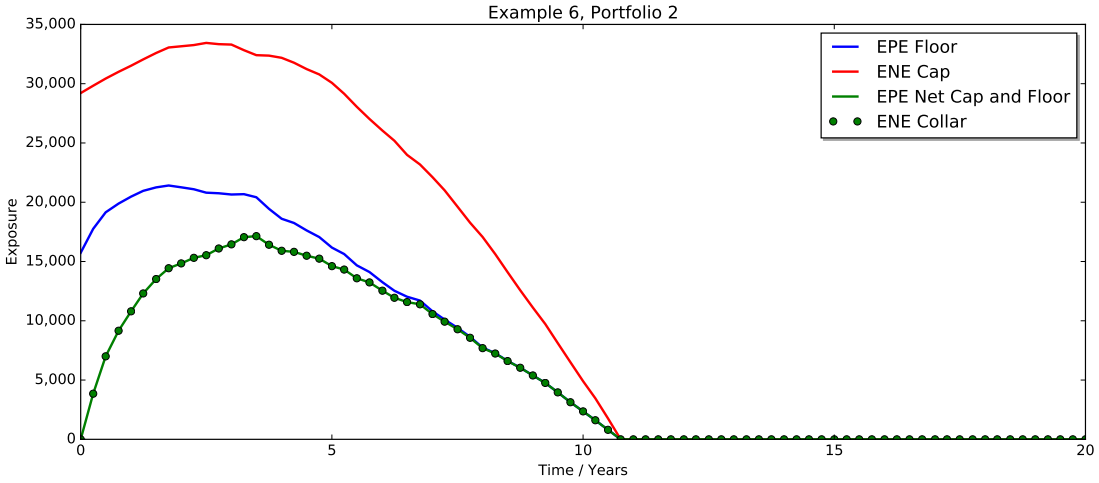

Figure 8: Short Cap and long Floor vs Collar, portfolio 2. Simulation with 1000 paths and quarterly time steps.

4.7 FX Option Exposure

This example (in folder `Examples/Example_7`, as the FX Forward example) illustrates the exposure evolution for an FX Option, see figure 10. Recall that the FX Option value $NPV(t)$ as of time $0 \leq t \leq T$ satisfies

$$\frac{NPV(t)}{N(t)} = \text{Nominal} \times \mathbb{E}_t \left[\frac{(X(T) - K)^+}{N(T)} \right]$$
$$NPV(0) = \mathbb{E} \left[\frac{NPV(t)}{N(t)} \right] = \mathbb{E} \left[\frac{NPV^+(t)}{N(t)} \right] = EPE(t)$$

One would therefore expect a flat exposure evolution up to option expiry. The deviation from this in ORE's simulation is due to the pricing approach chosen here under scenarios. A Black FX option pricer is used with deterministic Black volatility derived from today's volatility structure (pushed or rolled forward, see section 6.4.3). The deviation can be removed by extending the volatility modelling, e.g. implying model consistent Black volatilities in each simulation step on each path.

4.8 Cross Currency Swap Exposure and FX Reset

The case in `Examples/Example_8` is a vanilla cross currency Swap. It shows the typical blend of an Interest Rate Swap's saw tooth exposure evolution with an FX Forward's exposure which increases monotonically to final maturity, see figure 11.
The effect of the FX resetting feature, common in Cross Currency Swaps nowadays, is shown in `Examples/Example_9`. The example shows the exposure evolution of a EUR/USD cross currency basis Swap with FX reset at each interest period start, see figure 12. As expected, the notional reset causes an exposure collapse at each period start when the EUR leg's notional is reset to match the USD notional.

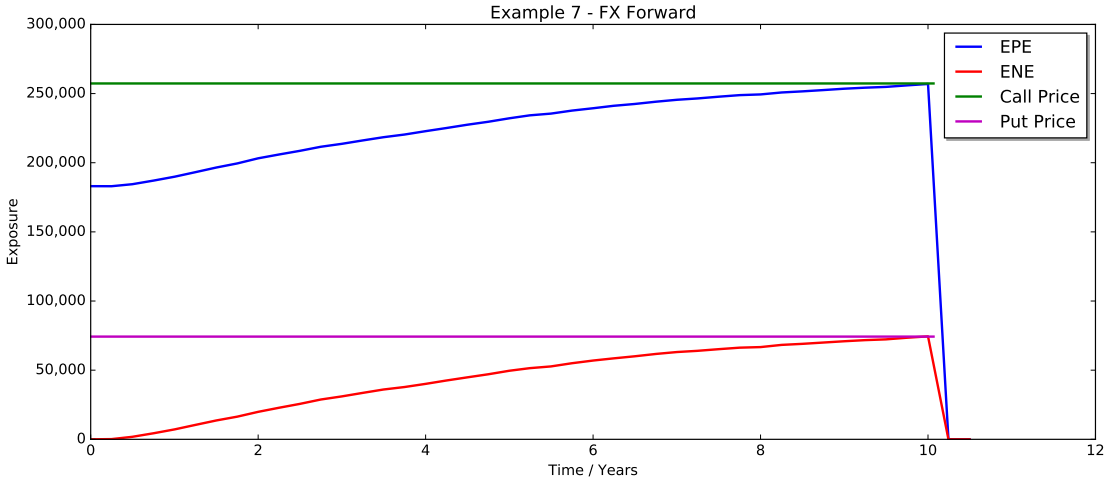

Figure 9: EUR/USD FX Forward expected exposure in a realistic market environment as of 26/02/2016 from both parties' perspectives. Value date is obviously in 10Y. The flat lines are FX Option prices which coincide with EPE and ENE, respectively, on the value date. Simulation with 5000 paths and quarterly time steps.

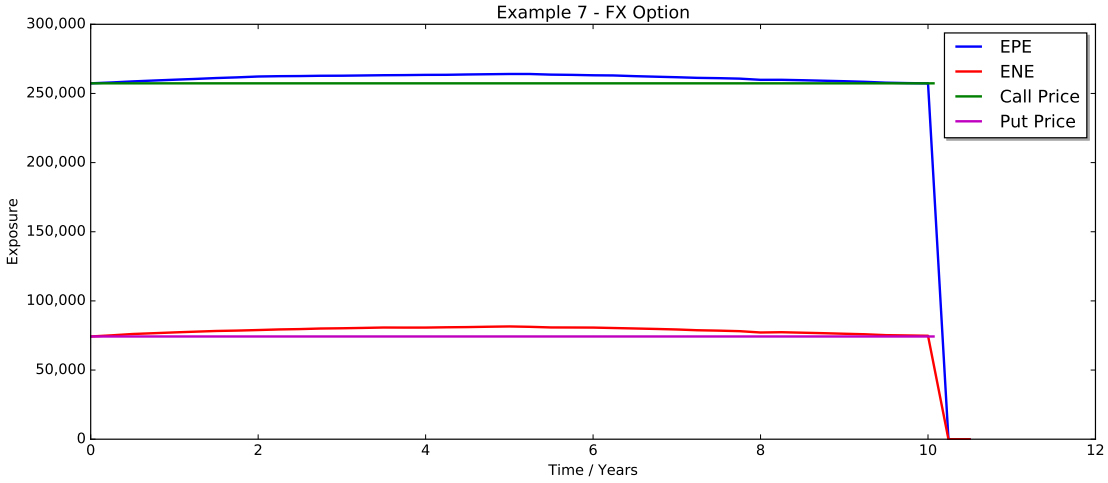

Figure 10: EUR/USD FX Call and Put Option exposure evolution, same underlying and market data as in section 4.6, compared to the call and put option price as of today (flat line). Simulation with 5000 paths and quarterly time steps.

4.9 Netting and Collateral

In this example (see folder `Examples/Example_10`) we showcase a small netting set consisting of three Swaps in different currencies, with different collateral choices

- no collateral - figure 13,
- collateral with threshold (THR) 1m EUR, minimum transfer amount (MTA) 100k EUR, margin period of risk (MPOR) 2 weeks - figure 14
- collateral with zero THR and MTA, and MPOR 2w - figure 15

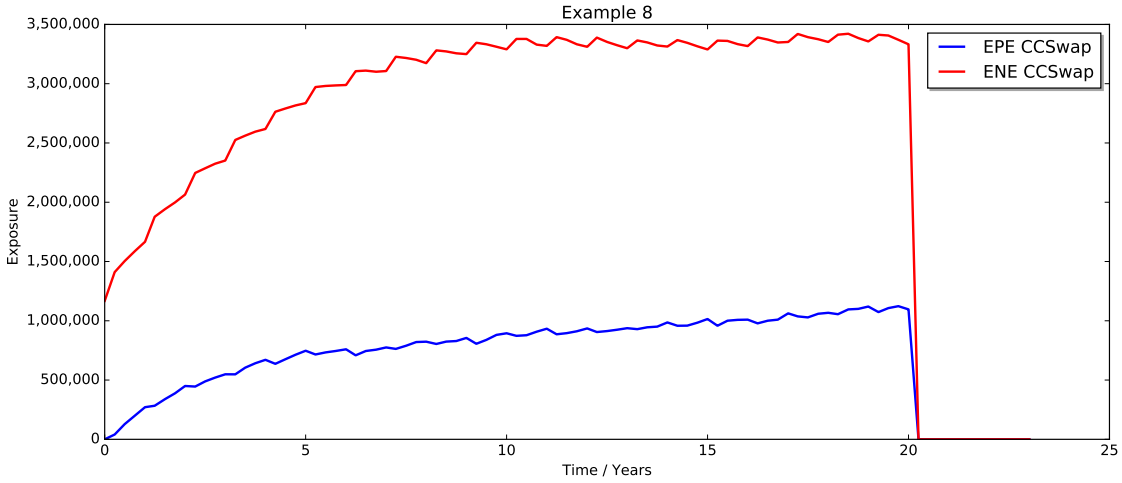

Figure 11: Cross Currency Swap exposure evolution without mark-to-market notional reset. Simulation with 1000 paths and quarterly time steps.

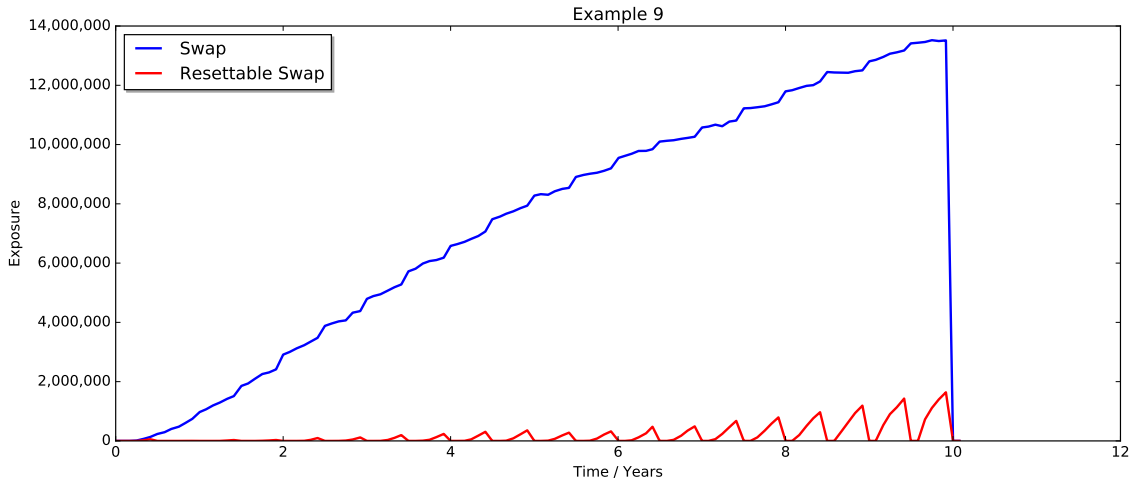

Figure 12: Cross Currency Basis Swap exposure evolution with and without mark-to-market notional reset. Simulation with 1000 paths and quarterly time steps.

The exposure graphs with collateral and positive margin period of risk show typical spikes. What is causing these? As sketched in appendix A.8, ORE uses a *classical collateral model* that applies collateral amounts to offset exposure with a time delay that corresponds to the margin period of risk. The spikes are then caused by instrument cash flows falling between exposure measurement dates d_1 and d_2 (an MPOR apart), so that a collateral delivery amount determined at d_1 but settled at d_2 differs significantly from the closeout amount at d_2 causing a significant residual exposure for a short period of time. See for example [19] for a recent detailed discussion of collateral modelling. The approach currently implemented in ORE corresponds to *Classical+* in [19], the more conservative approach of the classical methods. The less conservative alternative, *Classical-*, would assume that both parties stop paying trade flows at the beginning of the MPOR, so that the P&L over the MPOR does not contain the cash flow effect, and

exposure spikes are avoided. Note that the size and position of the largest spike in figure 14 is consistent with a cash flow of the 40 million GBP Swap in the example's portfolio that rolls over the 3rd of March and has a cash flow on 3 March 2020, a bit more than four years from the evaluation date.

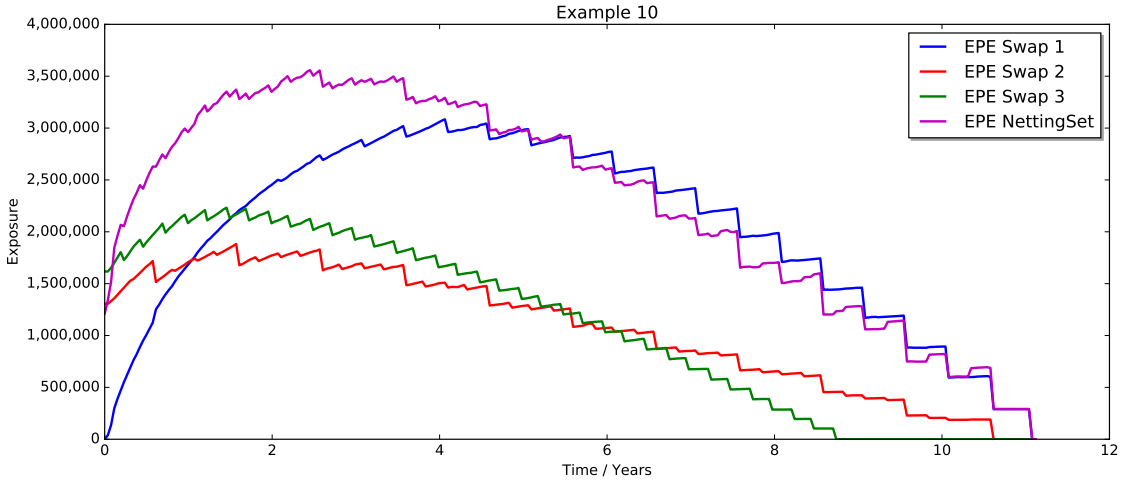

Figure 13: Three Swaps netting set, no collateral. Simulation with 5000 paths and bi-weekly time steps.

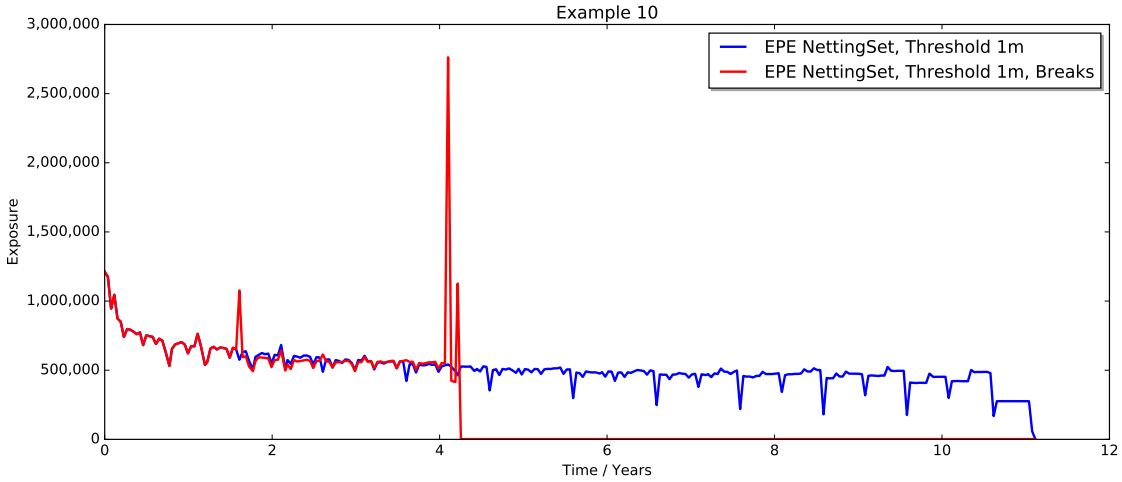

Figure 14: Three Swaps netting set, THR=1m EUR, MTA=100k EUR, MPOR=2w. The red evolution assumes that the each trade is terminated at the next break date. The blue evolution ignores break dates. Simulation with 5000 paths and bi-weekly time steps.

4.10 CVA, DVA, FVA, COLVA, MVA, Collateral Floor

We use one of the cases in `Examples/Example_10` to demonstrate the XVA outputs, see folder `Examples/Example_10/Output/collateral_threshold_dim`.

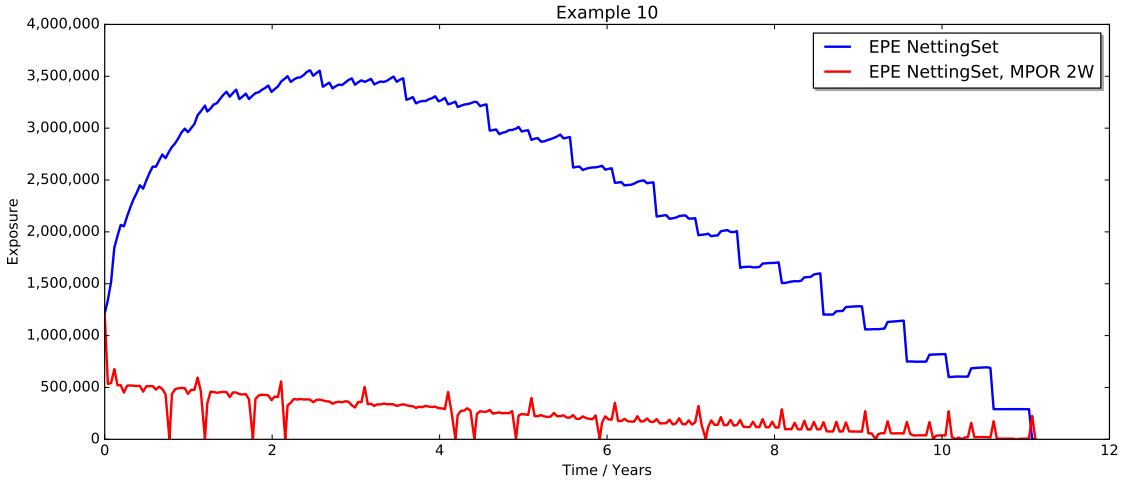

Figure 15: Three Swaps, THR=MTA=0, MPOR=2w. Simulation with 5000 paths and bi-weekly time steps.

The summary of all value adjustments (CVA, DVA, FVA, COLVA, MVA, as well as the Collateral Floor) is provided in file `xva.csv`. The file includes the allocated CVA and DVA numbers to individual trades as introduced in the next section. The following table illustrates the file's layout, omitting the three columns containing allocated data.

TradeId	NettingSetId	CVA	DVA	FBA	FCA	COLVA	MVA	CollateralFloor	BaselEPE	BaselEEPE
	CPTY_A	6,521	151,193	-946	72,103	2,769	-14,203	189,936	113,260	1,211,770
Swap_1	CPTY_A	127,688	211,936	-19,624	100,584	n/a	n/a	n/a	2,022,590	2,727,010
Swap_3	CPTY_A	71,315	91,222	-11,270	43,370	n/a	n/a	n/a	1,403,320	2,183,860
Swap_2	CPTY_A	68,763	100,347	-10,755	47,311	n/a	n/a	n/a	1,126,520	1,839,590

The line(s) with empty TradeId column contain values at netting set level, the others contain uncollateralised single-trade VAs. Note that COLVA, MVA and Collateral Floor are only available at netting set level at which collateral is posted.

Detailed output is written for COLVA and Collateral Floor to file `colva_nettingset_*.csv` which shows the incremental contributions to these two VAs through time.

4.11 Exposure Reports & XVA Allocation to Trades

Using the example in folder `Examples/Example_10` we illustrate here the layout of an exposure report produced by ORE. The report shows the exposure evolution of Swap_1 without collateral which - after running Example_10 - is found in folder `Examples/Example_10/Output/collateral_none/exposure_trade_Swap_1.csv`:

TradeId	Date	Time	EPE	ENE	AllocEPE	AllocENE	PFE	BaselEE	BaselEEE
Swap_1	05/02/16	0.0000	0	1,711,850	0	0	0	0	0
Swap_1	19/02/16	0.0383	40,219	1,744,080	-1,190,830	513,032	263,973	40,217	40,217
Swap_1	04/03/16	0.0765	137,552	1,840,760	-914,468	788,739	1,053,940	137,535	137,535
Swap_1	18/03/16	0.1148	299,155	1,742,450	-650,225	793,067	1,914,150	299,091	299,091
Swap_1	01/04/16	0.1530	390,178	1,834,810	-552,029	892,604	2,373,560	390,058	390,058
Swap_1	15/04/16	0.1913	471,849	1,918,600	-465,580	981,171	2,765,710	471,659	471,659
Swap_1	29/04/16	0.2295	550,301	2,000,640	-330,578	1,119,760	3,106,810	550,016	550,016
Swap_1	13/05/16	0.2678	620,279	2,074,880	-266,042	1,188,560	3,427,080	619,888	619,888
Swap_1	27/05/16	0.3060	690,018	2,140,320	-190,419	1,259,880	3,778,570	689,509	689,509
Swap_1	10/06/16	0.3443	763,207	2,206,020	-137,681	1,305,130	4,052,870	762,560	762,560
Swap_1	...	...	...	...	...	...	...	...	...

The exposure measures EPE, ENE and PFE, and the Basel exposure measures EE_B and EEE_B, are defined in appendix A.2. Allocated exposures are defined in appendix A.9. The PFE quantile and allocation method are chosen as described in section 6.1.3. In addition to single trade exposure files, ORE produces an exposure file per netting set. The example from the same folder as above is:

NettingSet	Date	Time	EPE	ENE	PFE	ExpectedCollateral	BaselEE	BaselEEE
CPTY_A	05/02/16	0.0000	1,211,770	0	1,211,770	0	1,211,770	1,211,770
CPTY_A	19/02/16	0.0383	1,344,220	137,776	3,414,000	0	1,344,160	1,344,160
CPTY_A	04/03/16	0.0765	1,518,610	308,381	4,354,060	0	1,518,410	1,518,410
CPTY_A	18/03/16	0.1148	1,846,900	382,068	5,200,730	0	1,846,500	1,846,500
CPTY_A	01/04/16	0.1530	1,961,290	494,416	5,869,470	0	1,960,690	1,960,690
CPTY_A	15/04/16	0.1913	2,067,240	598,283	6,384,140	0	2,066,400	2,066,400
CPTY_A	29/04/16	0.2295	2,053,670	745,960	6,740,070	0	2,052,610	2,066,400
CPTY_A	13/05/16	0.2678	2,149,190	845,507	6,930,230	0	2,147,840	2,147,840
CPTY_A	27/05/16	0.3060	2,235,630	930,218	7,295,440	0	2,233,980	2,233,980
CPTY_A	10/06/16	0.3443	2,314,470	1,014,690	7,753,190	0	2,312,510	2,312,510
CPTY_A	...	...	...	...	...	...	...	...
CPTY_A	07/07/17	1.4167	3,320,430	2,423,890	12,787,900	0	3,304,650	3,304,650
CPTY_A	21/07/17	1.4551	3,351,780	2,452,640	12,964,200	0	3,335,420	3,335,420
CPTY_A	04/08/17	1.4934	3,302,820	2,511,500	12,796,100	0	3,286,260	3,335,420
CPTY_A	18/08/17	1.5318	3,339,840	2,545,850	13,120,000	0	3,322,640	3,335,420
CPTY_A	01/09/17	1.5701	3,371,300	2,576,100	13,238,700	0	3,353,480	3,353,480
CPTY_A	15/09/17	1.6085	3,279,670	2,555,370	13,041,300	0	3,261,880	3,353,480
CPTY_A	29/09/17	1.6468	3,305,060	2,579,200	13,072,800	0	3,286,680	3,353,480
CPTY_A	13/10/17	1.6852	3,332,830	2,604,200	13,225,600	0	3,313,850	3,353,480
CPTY_A	27/10/17	1.7236	3,280,280	2,661,770	13,034,600	0	3,261,150	3,353,480
CPTY_A	13/11/17	1.7701	3,316,800	2,701,060	13,331,600	0	3,296,880	3,353,480
CPTY_A	24/11/17	1.8003	3,337,760	2,720,870	13,402,400	0	3,317,280	3,353,480
CPTY_A	...	...	...	...	...	...	...	...

Allocated exposures are missing here, as they make sense at the trade level only, and the expected collateral balance is added for information (in this case zero as collateralisation is deactivated in this example).

The allocation of netting set exposure and XVA to the trade level is frequently required by finance departments. This allocation is also featured in `Examples/Example_10`. We start again with the uncollateralised case in figure 16, followed by the case with threshold 1m EUR in figure 17. In both cases we apply the *marginal* (Euler) allocation method as published by Pykhtin and Rosen in 2010, hence we see the typical negative EPE for one of the trades at times when it reduces the netting set exposure. The case with collateral moreover shows the typical spikes in the allocated exposures. The analytics results also feature allocated XVAs in file `xva.csv` which are derived from the allocated exposure profiles. Note that ORE also offers alternative allocation methods to the marginal method by Pykhtin/Rosen, which can be explored with `Examples/Example_10`.

4.12 Basel Exposure Measures

Example `Example_11` demonstrates the relation between the evolution of the expected exposure (EPE in our notation) to the 'Basel' exposure measures EE_B, EEE_B, EPE_B and EEPE_B as defined in appendix A.2. In particular the latter is used in internal model methods for counterparty credit risk as a measure for the exposure at default. It is a 'derivative' of the expected exposure evolution and defined as a time average over the running maximum of EE_B up to the horizon of one year.

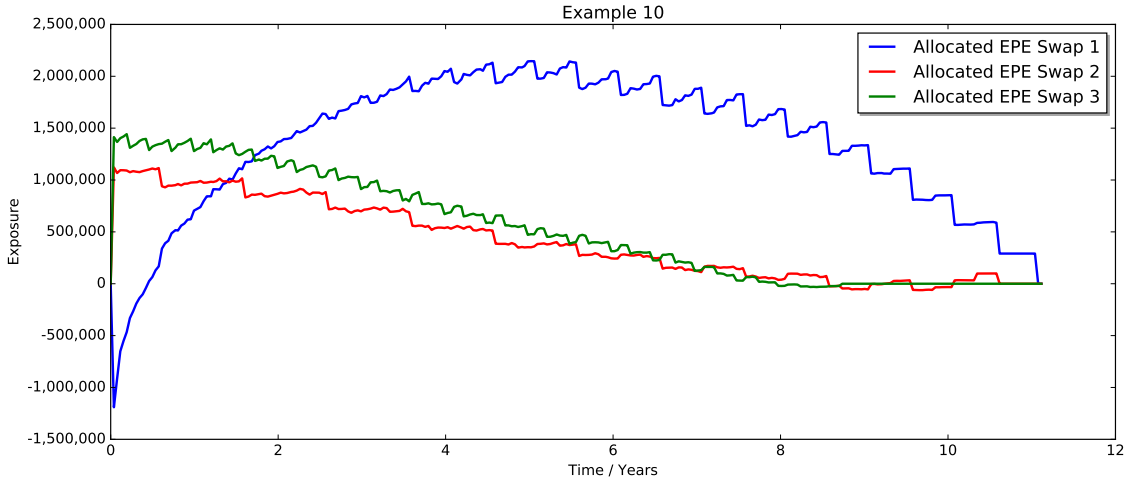

Figure 16: Exposure allocation without collateral. Simulation with 5000 paths and bi-weekly time steps.

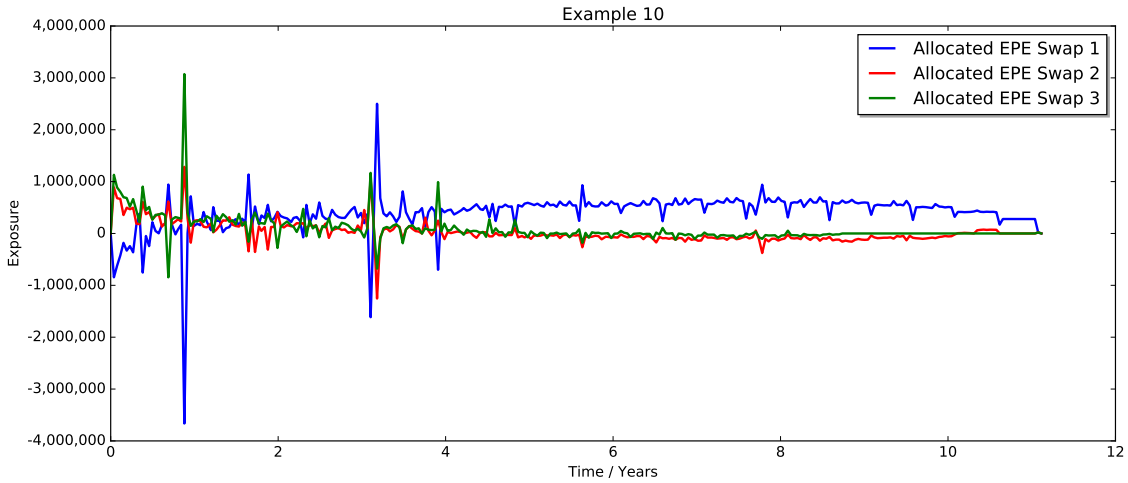

Figure 17: Exposure allocation with collateral and threshold 1m EUR. Simulation with 5000 paths and bi-weekly time steps.

4.13 Long Term Simulation with Horizon Shift

The example in folder `Example_12` finally demonstrates an effect that, at first glance, seems to cause a serious issue with long term simulations. Fortunately this can be avoided quite easily in the Linear Gauss Markov model setting that is used here.
In the example we consider a Swap with maturity in 50 years in a flat yield curve environment. If we simulate this naively as in all previous cases, we obtain a particularly noisy EPE profile that does not nearly reconcile with the known exposure (analytical Swaption prices). This is shown in figure 19 ('no horizon shift'). The origin of this issue is the width of the risk-neutral NPV distribution at long time horizons which can turn out to be quite small so that the Monte Carlo simulation with finite number of samples does not reach far enough into the positive or negative NPV range to adequately sample the distribution, and estimate both EPE and ENE in a single run. Increasing the number

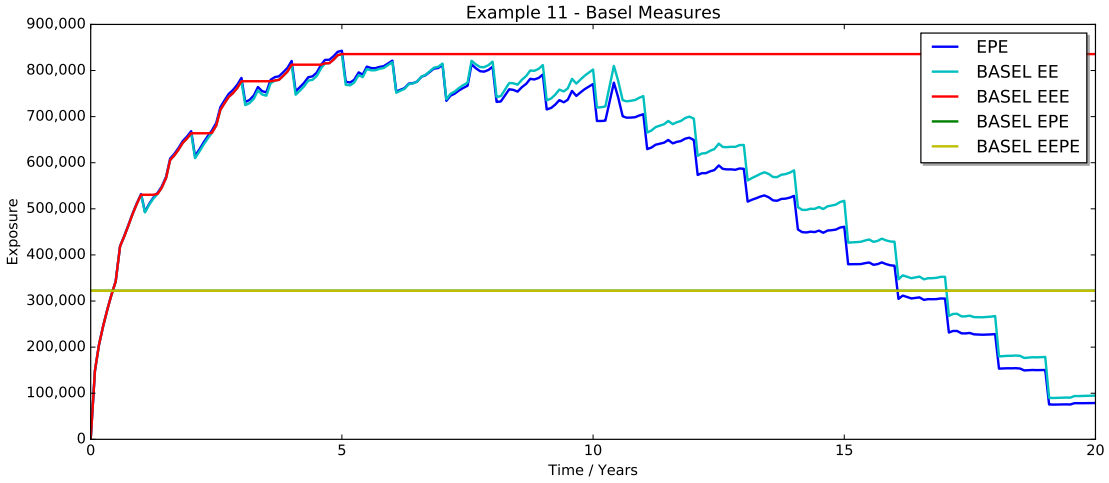

Figure 18: Evolution of the expected exposure of Vanilla Swap, comparison to the 'Basel' exposure measures EEE_B, EPE_B and EEPE_B.

of samples may not solve the problem, and may not even be feasible in a realistic setting. The way out is applying a 'shift transformation' to the Linear Gauss Markov model, see `Example_12/Input/simulation2.xml` in lines 92-95:

```
<ParameterTransformation>
  <ShiftHorizon>30.0</ShiftHorizon>
  <Scaling>1.0</Scaling>
</ParameterTransformation>
```

The effect of the 'ShiftHorizon' parameter T is to apply a shift to the Linear Gauss Markov model's $H(t)$ parameter (see appendix A.1) *after* the model has been calibrated, i.e. to replace:

$$H(t) \to H(t) - H(T)$$

It can be shown that this leaves all expectations computed in the model (such as EPE and ENE) invariant. As explained in [17], subtracting a H shift effectively means performing a change of measure from the 'native' LGM measure to a T-Forward measure with horizon T, here 30 years. Both negative and positive shifts are permissible, but only negative shifts are connected with a T-Forward measure and improve numerical stability. In our experience it is helpful to place the horizon in the middle of the portfolio duration to significantly improve the quality of long term expectations. The effect of this change (only) is shown in the same figure 19 ('shifted horizon'). Figure 20 further illustrates the origin of the problem and its resolution: The rate distribution's mean (without horizon shift or change of measure) drifts upwards due to convexity effects (note that the yield curve is flat in this example), and the distribution's width is then too narrow at long horizons to yield a sufficient number of low rate scenarios with contributions to the Swap's *EPE* (it is a floating rate payer). With the horizon shift (change of measure), the distribution's mean is pulled 'back' at long horizons, because the convexity effect is effectively wiped out at the chosen horizon, and the expected rate matches the forward rate.

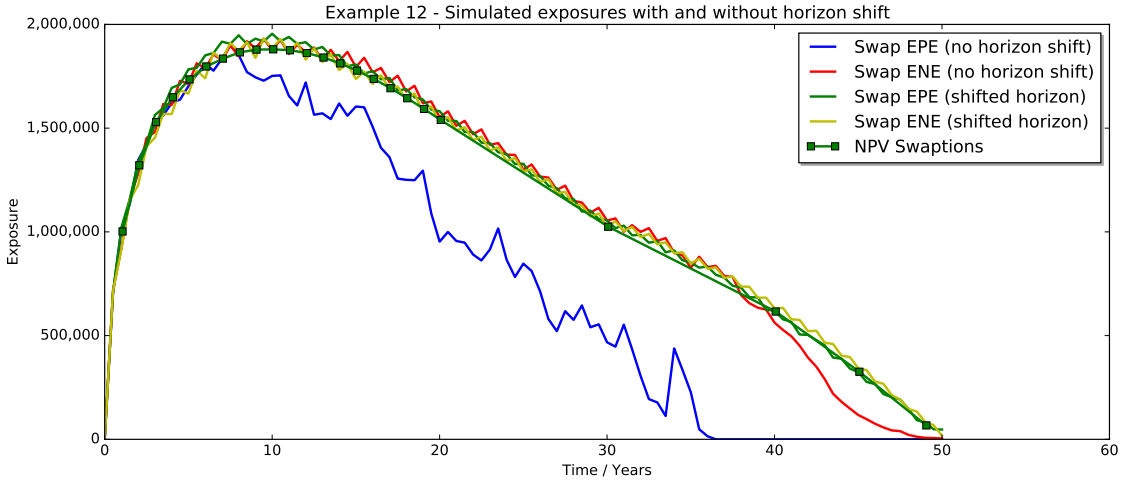

Figure 19: Long term Swap exposure simulation with and without horizon shift.

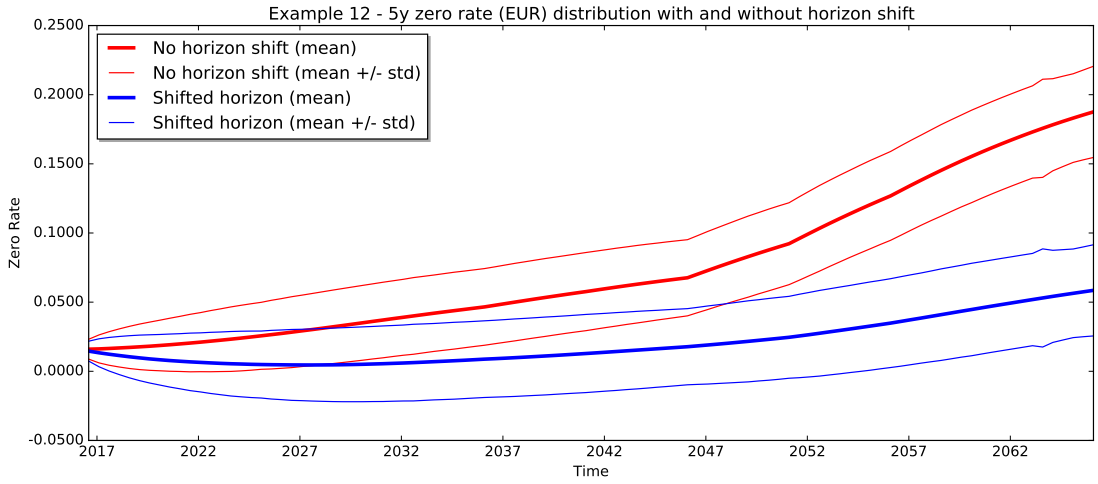

Figure 20: Evolution of rate distributions with and without horizon shift (change of measure). Thick lines indicate mean values, thin lines are contours of the rate distribution at ± one standard devation.

4.14 Dynamic Initial Margin and MVA

This example in folder `Examples/Example_13` demonstrates Dynamic Initial Margin calculations (see also appendix A.7) for a number of elementary products:

- A single currency Swap in EUR (case A),
- a European Swaption in EUR with physical delivery (case B),
- a single currency Swap in USD (case C), and
- a EUR/USD cross currency Swap (case D).

The examples can be run as before with

```
python run_A.py
```

and likewise for cases B, C and D. The essential results of each run are are visualised in the form of

- evolution of expected DIM
- regression plots at selected future times

illustrated for cases A and B in figures 21 - 24. In all cases the zero order regression estimate of DIM differs noticeably from the higher orders (one and two). In the three swap cases we moreover see that first and second order polynomial choice makes hardly any difference; note that case C and D use up to three-dimensional regressors (simulated USD and EUR rate fixings and the EUR/USD FX rate). In the Swaption case B, first and second order polynomial choice makes a difference before option expiry.

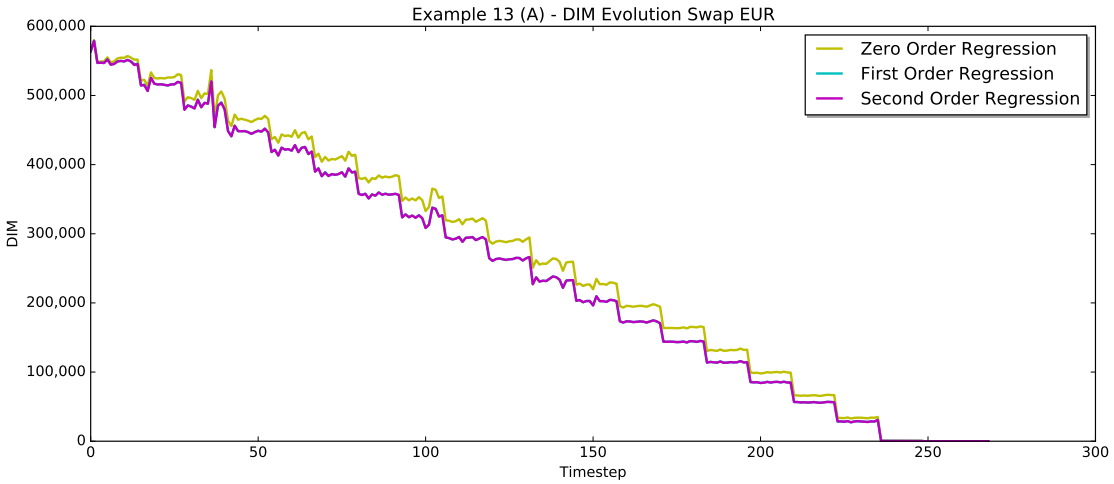

Figure 21: Evolution of expected Dynamic Initial Margin (DIM) for the EUR Swap of Example 13 A. DIM is evaluated using regression of NPV change variances versus the simulated 3M Euribor fixing; regression polynomials are zero, first and second order (first and second order curves are not distinguishable here). The simulation uses 1000 samples and a time grid with bi-weekly steps in line with the Margin Period of Risk.

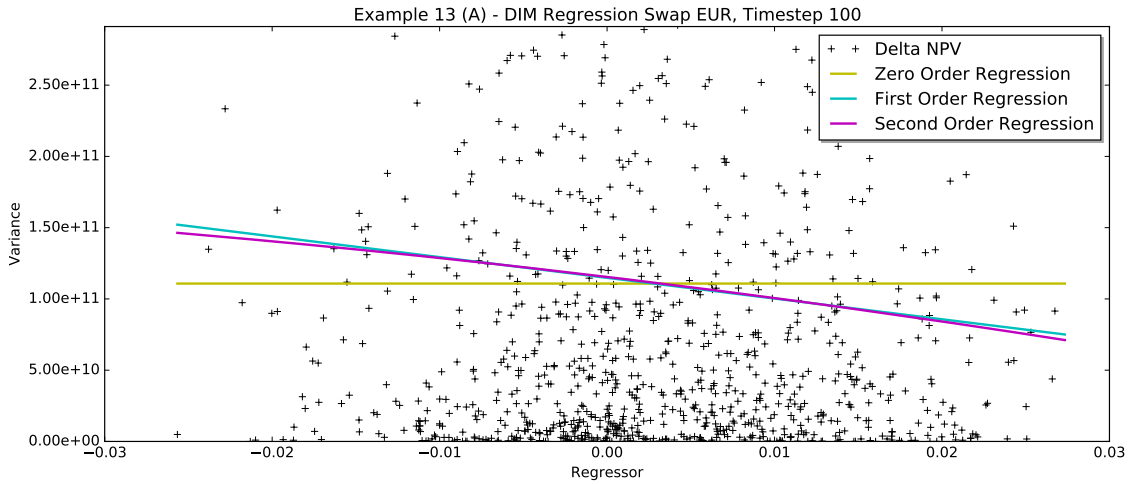

Figure 22: Regression snapshot at time step 100 for the EUR Swap of Example 13 A.

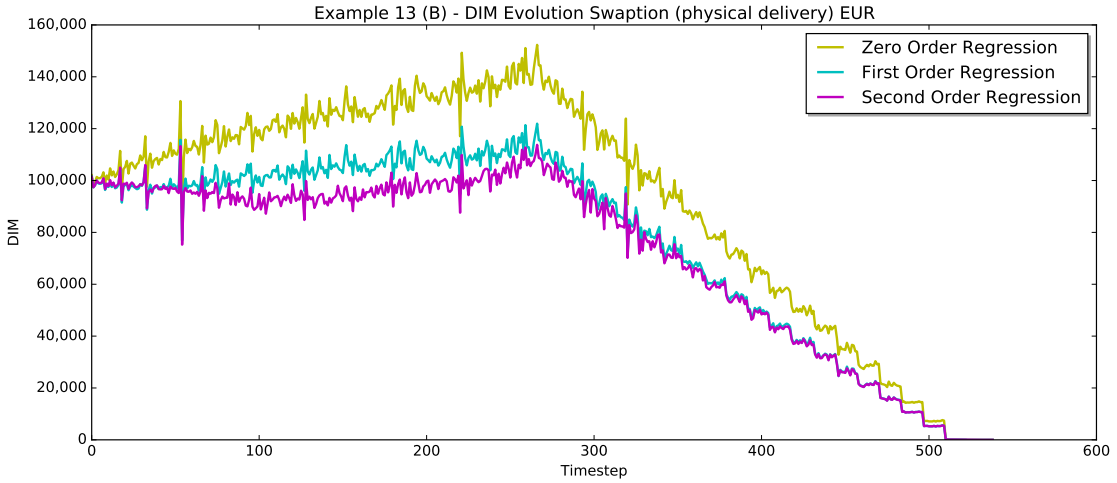

Figure 23: Evolution of expected Dynamic Initial Margin (DIM) for the EUR Swaption of Example 13 B with expiry in 10Y around time step 100. DIM is evaluated using regression of NPV change variances versus the simulated 3M Euribor fixing; regression polynomials are zero, first and second order. The simulation uses 1000 samples and a time grid with bi-weekly steps in line with the Margin Period of Risk.

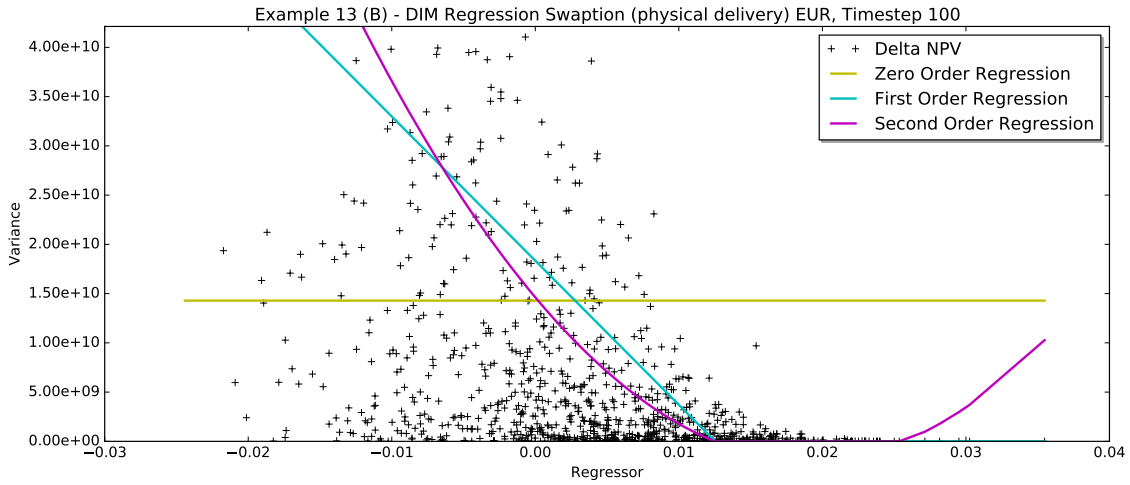

Figure 24: Regression snapshot at time step 100 (before expiry) for the EUR Swaption of Example 13 B.

5 Launchers and Visualisation

5.1 Jupyter

ORE comes with an experimental Jupyter notebook for launching ORE batches and in particular for drilling into NPV cube data. The notebook is located in directory `FrontEnd/Python/Visualization/npvcube`. To launch the notebook, change to this directory and follow instructions in the `Readme.txt`. In a nutshell, type[3]

```
jupyter notebook
```

to start the ipyton console and open a browser window. From the list of files displayed in the browser then click

```
ore_jupyter_dashboard.ipynb
```

to open the ORE notebook. The notebook offers

- launching an ORE job
- selecting an NPV cube file and netting sets or trades therein
- plotting a 3d exposure probability density surface
- viewing exposure probability density function at a selected future time
- viewing expected exposure evolution through time

The cube file loaded here by default when processing all cells of the notebook (without changing it or launching a ORE batch) is taken from `Example_7` (FX Forwards and FX Options).

5.2 Calc

ORE comes with a simple LibreOffice Calc [10] sheet as an ORE launcher and basic result viewer. This is demonstrated on the example in section 4.1. It is currently based on the stable LibreOffice version 5.0.6 and tested on OS X.
To launch Calc, open a terminal, change to directory `Examples/Example_1`, and run

```
./launchCalc.sh
```

The user can choose a configuration (one of the `ore*.xml` files in Example_1's subfolder Input) by hitting the 'Select' button. Initially Input/ore.xml is pre-selected. The ORE process is then kicked off by hitting 'Run'. Once completed, standard ORE reports (NPV, Cashflow, XVA) are loaded into several sheets. Moreover, exposure evolutions can then be viewed by hitting 'View' which shows the result in figure 25.
This demo uses simple Libre Office Basic macros which call Python scripts to execute ORE. The Libre Office Python uno module (which comes with Libre Office) is used to communicate between Python and Calc to upload results into the sheets.

[3] With Mac OS X, you may need to set the environment variable `LANG` to `en_US.UTF-8` before running jupyter, as mentioned in the installation section 3.3.

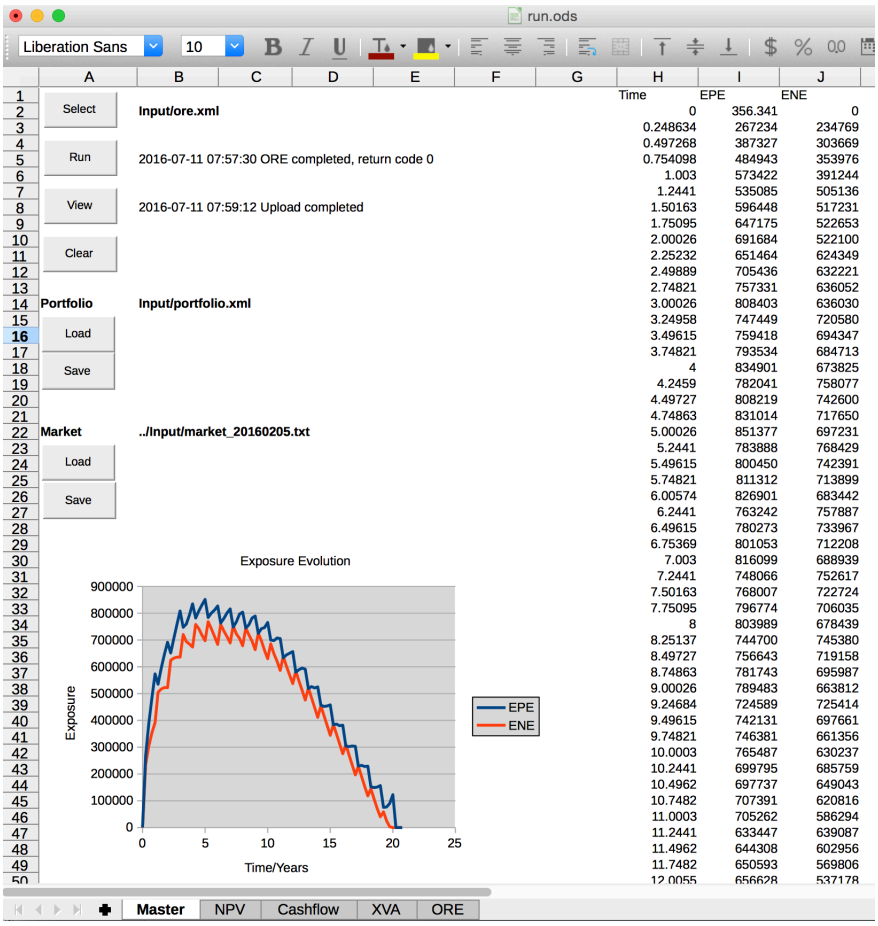

Figure 25: Calc sheet after hitting 'Run'.

5.3 Excel

ORE also comes with a basic Excel sheet to demonstrate launching ORE and presenting results in Excel. This demo is more self-contained than the Calc demo in the previous section, as it uses VBA only rather than calls to external Python scripts. The Excel demo is available in Example_1. Launch `Example_1.xlsm`. Then modify the paths on the first sheet, and kick off the ORE process.

6 Parametrisation

ORE's batch version is kicked off with a single command line parameter

```
ore[.exe] ore.xml
```

which points to the 'master input file' referred to as `ore.xml` subsequently. This file is the starting point of the engine's configuration explained in the following sub section.

6.1 Master Input File: `ore.xml`

The master input file contains general setup information (paths to configuration, trade data and market data), as well as the selection and configuration of analytics. The file has an opening and closing root element `<ORE>`, `</ORE>` with three sections

- Setup
- Markets
- Analytics

which we will explain in the following.

6.1.1 Setup

This subset of data is easiest explained using an example, see listing 1.

Listing 1: ORE setup example

```
<Setup>
  <Parameter name="asofDate">2016-02-05</Parameter>
  <Parameter name="inputPath">Input</Parameter>
  <Parameter name="outputPath">Output</Parameter>
  <Parameter name="logFile">log.txt</Parameter>
  <Parameter name="marketDataFile">../../Input/market_20160205.txt</Parameter>
  <Parameter name="fixingDataFile">../../Input/fixings_20160205.txt</Parameter>
  <Parameter name="implyTodaysFixings">Y</Parameter>
  <Parameter name="curveConfigFile">../../Input/curveconfig.xml</Parameter>
  <Parameter name="conventionsFile">../../Input/conventions.xml</Parameter>
  <Parameter name="marketConfigFile">../../Input/todaysmarket.xml</Parameter>
  <Parameter name="pricingEnginesFile">../../Input/pricingengine.xml</Parameter>
  <Parameter name="portfolioFile">portfolio.xml</Parameter>
  <!-- None, Unregister, Defer or Disable -->
  <Parameter name="observationModel">Disable</Parameter>
</Setup>
```

Parameter names are self explanatory: Input and output path are interpreted relative from the directory where the ORE script is started, but can also be specified using absolute paths. All file names are then interpreted relative to the 'inputPath' and 'outputPath', respectively. The files starting with `../../Input/` then point to files in the global Example input directory `Example/Input/*`, whereas files such as `portfolio.xml` are local inputs in `Example/Example_#/Input/`.
When ORE starts, it will initialise today's market, i.e. load market data and fixings, and build all term structures as specified in `todaysmarket.xml`. Moreover, ORE will load the trades in `portfolio.xml` and link them with pricing engines as specified in `pricingengine.xml`. When parameter `implyTodaysFixings` is set to Y, today's fixings would not be loaded but implied, relevant when pricing/bootstrapping off hypothetical market data as e.g. in scenario analysis and stress testing.

The last parameter `observationModel` can be used to control ORE performance during simulation. The choices *Disable* and *Unregister* yield similarly improved performance relative to choice *None*. For users familiar with the QuantLib design - the parameter controls to which extent *QuantLib observer notifications* are used when market and fixing data is updated and the evaluation date is shifted:

- The 'Unregister' option limits the amount of notifications by unregistering floating rate coupons from indices;

- Option 'Defer' disables all notifications during market data and fixing updates with `ObservableSettings::instance().disableUpdates(true)` and kicks off updates afterwards when enabled again
- The 'Disable' option goes one step further and disables all notifications during market data and fixing updates, and in particular when the evaluation date is changed along a path, with
  `ObservableSettings::instance().disableUpdates(false)`
  Updates are not deferred here. Required term structure and instrument recalculations are triggered explicitly.

6.1.2 Markets

The `Markets` section (see listing 2) is used to choose market configurations for calibrating the IR and FX simulation model components, pricing and simulation, respectively. These configurations have to be defined in `todaysmarket.xml` (see section 6.2).

Listing 2: ORE markets

```
<Markets>
  <Parameter name="lgmcalibration">collateral_inccy</Parameter>
  <Parameter name="fxcalibration">collateral_eur</Parameter>
  <Parameter name="pricing">collateral_eur</Parameter>
  <Parameter name="simulation">collateral_eur</Parameter>
</Markets>
```

For example, the calibration of the simulation model's interest rate components requires local OIS discounting whereas the simulation phase requires cross currency adjusted discount curves to get FX product pricing right. So far, the market configurations are used only to distinguish discount curve sets, but the market configuration concept in ORE applies to all term structure types.

6.1.3 Analytics

The `Analytics` section lists all permissible analytics using tags `<Analytic type="..."> ... </Analytic>` where type can be (so far) in

- npv
- cashflow
- curves
- simulation
- xva
- initialMargin

Each `Analytic` section contains a list of key/value pairs to parameterise the analysis of the form `<Parameter name="key">value</Parameter>`. Each analysis must have one key `active` set to Y or N to activate/deactivate this analysis. The following listing 3 shows the parametrisation of the first three basic analytics in the list above.

Listing 3: ORE analytics: npv, cashflow and curves

```
<Analytics>
  <Analytic type="npv">
    <Parameter name="active">Y</Parameter>
    <Parameter name="baseCurrency">EUR</Parameter>
    <Parameter name="outputFileName">npv.csv</Parameter>
  </Analytic>
  <Analytic type="cashflow">
    <Parameter name="active">Y</Parameter>
    <Parameter name="outputFileName">flows.csv</Parameter>
  </Analytic>
  <Analytic type="curves">
    <Parameter name="active">Y</Parameter>
    <Parameter name="configuration">default</Parameter>
    <Parameter name="grid">240,1M</Parameter>
    <Parameter name="outputFileName">curves.csv</Parameter>
  </Analytic>
  <Analytic type="...">
    <!-- ... -->
  </Analytic>
</Analytics>
```

The curves analytic exports all yield curves that have been built according to the specification in `todaysmarket.xml`. Key `configuration` selects the curve set to be used (see explanation in the previous Markets section). Key `grid` defines the time grid on which the yield curves are evaluated, in the example above a grid of 240 monthly time steps from today. The discount factors for all curves with configuration default will be exported on this monthly grid into the csv file specified by key `outputFileName`. The grid can also be specified explicitly by a comma separated list of tenor points such as `1W, 1M, 2M, 3M, ....`

The purpose of the `simulation` 'analytics' is to run a Monte Carlo simulation which evolves the market as specified in the simulation config file. The primary result is an NPV cube file, i.e. valuations of all trades in the portfolio file (see section Setup), for all future points in time on the simulation grid and for all paths. Apart from the NPV cube, additional scenario data (such as simulated overnight rates etc) are stored in this process which are needed for subsequent XVA analytics.

Listing 4: ORE analytic: simulation

```
<Analytics>
  <Analytic type="simulation">
    <Parameter name="active">Y</Parameter>
    <Parameter name="simulationConfigFile">simulation.xml</Parameter>
    <Parameter name="pricingEnginesFile">../../Input/pricingengine.xml</Parameter>
    <Parameter name="baseCurrency">EUR</Parameter>
    <Parameter name="storeScenarios">N</Parameter>
    <Parameter name="storeFlows">Y</Parameter>
    <Parameter name="cubeFile">cube_A.dat</Parameter>
    <Parameter name="additionalScenarioDataFileName">scenariodata.dat</Parameter>
    <Parameter name="scenariodump">scenariodump.csv</Parameter>
  </Analytic>
</Analytics>
```

The pricing engines file specifies how trades are priced under future scenarios which can differ from pricing as of today (specified in section Setup). Key base currency determines into which currency all NPVs will be converted. Key store scenarios (Y or N) determines whether the market scenarios are written to a file for later reuse. And finally, the key 'store flows' (Y or N) controls whether cumulative cash flows between simulation dates are stored in the (hyper-) cube for post processing in the context of Dynamic Initial Margin and Variation Margin calculations. The additional scenario data (written to the specified file here) is likewise required in the post processor step. These data comprise simulated index fixing e.g. for collateral compounding and simulated FX rates for cash collateral conversion into base currency. The scenario dump file, if specified here, causes ORE to write simulated market data to a human-readable csv file.

The XVA analytic section offers CVA, DVA, FVA and COLVA calculations which can be selected/deselected here individually. All XVA calculations depend on a previously generated NPV cube (see above) which is referenced here via the `cubeFile` parameter. This means one can re-run the XVA analytics without regenerating the cube each time. The XVA reports depend in particular on the settings in the `csaFile` which determines CSA details such as margining frequency, collateral thresholds, minimum transfer amounts, margin period of risk. By splitting the processing into pre-processing (cube generation) and post-processing (aggregation and XVA analysis) it is possible to vary these CSA details and analyse their impact on XVAs quickly without re-generating the NPV cube.

Listing 5: ORE analytic: xva

```
<Analytics>
  <Analytic type="xva">
    <Parameter name="active">Y</Parameter>
    <Parameter name="csaFile">netting.xml</Parameter>
    <Parameter name="cubeFile">cube.dat</Parameter>
    <Parameter name="hyperCube">Y</Parameter>
    <Parameter name="scenarioFile">scenariodata.dat</Parameter>
    <Parameter name="baseCurrency">EUR</Parameter>
    <Parameter name="exposureProfiles">Y</Parameter>
    <Parameter name="quantile">0.95</Parameter>
    <Parameter name="calculationType">Symmetric</Parameter>
    <Parameter name="allocationMethod">None</Parameter>
    <Parameter name="marginalAllocationLimit">1.0</Parameter>
    <Parameter name="exerciseNextBreak">N</Parameter>
    <Parameter name="cva">Y</Parameter>
    <Parameter name="dva">N</Parameter>
    <Parameter name="dvaName">BANK</Parameter>
    <Parameter name="fva">N</Parameter>
    <Parameter name="fvaBorrowingCurve">BANK_EUR_BORROW</Parameter>
    <Parameter name="fvaLendingCurve">BANK_EUR_LEND</Parameter>
    <Parameter name="colva">Y</Parameter>
    <Parameter name="collateralSpread">0.0010</Parameter>
    <Parameter name="collateralFloor">Y</Parameter>
    <Parameter name="dim">Y</Parameter>
    <Parameter name="mva">Y</Parameter>
    <Parameter name="dimQuantile">0.99</Parameter>
    <Parameter name="dimHorizonCalendarDays">14</Parameter>
    <Parameter name="dimRegressionOrder">1</Parameter>
    <Parameter name="dimRegressors">EUR-EURIBOR-3M,USD-LIBOR-3M,USD</Parameter>
    <Parameter name="dimLocalRegressionEvaluations">100</Parameter>
    <Parameter name="dimLocalRegressionBandwidth">0.25</Parameter>
    <Parameter name="dimScaling">1.0</Parameter>
    <Parameter name="dimEvolutionFile">dim_evolution.txt</Parameter>
    <Parameter name="dimRegressionFile">dim_regression.txt</Parameter>
    <Parameter name="dimOutputNettingSet">CPTY_A</Parameter>
    <Parameter name="dimOutputGridPoint">0</Parameter>
    <Parameter name="rawCubeOutputFile">rawcube.csv</Parameter>
    <Parameter name="netCubeOutputFile">netcube.csv</Parameter>
  </Analytic>
</Analytics>
```

Parameters:

- `csaFile`: Netting set definitions file covering CSA details such as margining frequency, thresholds, minimum transfer amounts, margin period of risk
- `cubeFile`: NPV cube file previously generated and to be post-processed here
- `hyperCube`: If set to N, the cube file is expected to have depth 1 (storing NPV data only), if set to Y it is expected to have depth ¿ 1 (e.g. storing NPVs and cumulative flows)
- `scenarioFile`: Scenario data previously generated and used in the post-processor (simulated index fixings and FX rates)

- `baseCurrency`: Expression currency for all NPVs, value adjustments, exposures
- `exposureProfiles`: Flag to enable/disable exposure output
- `quantile` Confidence level for Potential Future Exposure (PFE) reporting
- `calculationType` Determines the settlement of margin calls: Symmetric - margin for both counterparties settled after the margin period of risk; AsymmetricCVA - margin requested from the counterparty settles with delay, margin requested from us settles immediately; AsymmetricDVA - vice versa).
- `allocationMethod`: XVA allocation method, choices are *None, Marginal, RelativeXVA*
- `marginalAllocationLimit`: The marginal allocation method a la Pykhtin/Rosen breaks down when the netting set value vanishes while the exposure does not. This parameter acts as a cutoff for the marginal allocation when the absolute netting set value falls below this limit and switches to equal distribution of the exposure in this case.
- `exerciseNextBreak`: Flag to terminate all trades at their next break date before aggregation and the subsequent analytics
- `cva, dva, fva, colva, collateralFloor, dim, mva`: Flags to enable/disable these analytics.
- `dvaName`: Credit name to look up the own default probability curve and recovery rate for DVA calculation
- `fvaBorrowingCurve`: Identifier of the borrowing yield curve
- `fvaLendingCurve`: Identifier of the lending yield curve
- `collateralSpread`: Deviation between collateral rate and overnight rate, expressed in absolute terms (one basis point is 0.0001) assuming the day count convention of the collateral rate.
- `dimQuantile`: Quantile for Dynamic Initial Margin (DIM) calculation
- `dimHorizonCalendarDays`: Horizon for DIM calculation, 14 calendar days for 2 weeks, etc.
- `dimRegressionOrder`: Order of the regression polynomial (netting set clean NPV move over the simulation period versus netting set NPV at period start)
- `dimRegressors`: Variables used as regressors in a single- or multi-dimensional regression; these variable names need to match entries in the `simulation.xml`'s AggregationScenarioDataCurrencies and AggregationScenarioDataIndices sections (only these scenario data are passed on to the post processor)
- `dimLocalRegressionEvaluations`: Nadaraya-Watson local regression evaluated at the given number of points to validate polynomial regression. Note that Nadaraya-Watson needs a large number of samples for meaningful results. Evaluating the local regression at many points (samples) has a significant performance impact, hence the option here to limit the number of evaluations.

- `dimLocalRegressionBandwidth`: Nadaraya-Watson local regression bandwidth in standard deviations of the independent variable (NPV)
- `dimScaling`: Scaling factor applied to all DIM values used, e.g. to reconcile simulated DIM with actual IM at t_0
- `dimEvolutionFile`: Output file name to store the evolution of zero order DIM and average of nth order DIM through time
- `dimRegressionFile`: Output file name for a DIM regression snapshot
- `dimOutputNettingSet`: Netting set for the DIM regression snapshot
- `dimOutputGridPoint`: Grid point (in time) for the DIM regression snapshot
- `rawCubeOutputFile`: File name for the trade NPV cube in human readable csv file format (per trade, date, sample)
- `netCubeOutputFile`: File name for the aggregated NPV cube in human readable csv file format (per netting set, date, sample) *after* taking collateral into account

The latter two cube file outputs are provided for interactive analysis and visualisation purposes, see section 5.

6.2 Market: `todaysmarket.xml`

This configuration file determines the subset of the 'market' universe which is going to be built by ORE. It is the user's responsibility to make sure that this subset is sufficient to cover the portfolio to be analysed. If it is not, the application will complain at run time and exit.

We assume that the market configuration is provided in file `todaysmarket.xml`, however, the file name can be chosen by the user. The file name needs to be entered into the master configuration file `ore.xml`, see section 6.1.

The file starts and ends with the opening and closing tags `<TodaysMarket>` and `</TodaysMarket>`. The file then contains configuration blocks for

- Discounting curves
- Index curves (to project index fixings)
- Swap index curves (to project Swap rates)
- FX spot rates
- FX Volatility structures
- Swaption volatility structures
- Cap/Floor volatility structures
- Default curves

There can be alternative versions of each block each labeled with a unique identifier (e.g. Discount curve block with ID 'default', discount curve block with ID 'ois', another one with ID 'xois', etc). The purpose of these IDs will be explained at the end of this section. We now discuss each block's layout.

6.2.1 Discounting Curves

We pick one discounting curve block as an example here (see `Examples/Input/todaysmarket.xml`), the one with ID 'ois'

Listing 6: Discount curve block with ID 'ois'

```
<DiscountingCurves Id="ois">
  <DiscountingCurve Currency="EUR">Yield/EUR/EUR1D</DiscountingCurve>
  <DiscountingCurve Currency="USD">Yield/USD/USD1D</DiscountingCurve>
  <DiscountingCurve Currency="GBP">Yield/GBP/GBP1D</DiscountingCurve>
  <DiscountingCurve Currency="CHF">Yield/CHF/CHF6M</DiscountingCurve>
  <DiscountingCurve Currency="JPY">Yield/JPY/JPY6M</DiscountingCurve>
  <!-- ... -->
</DiscountingCurves>
```

This block instructs ORE to build five discount curves for the indicated currencies. The string within the tags, e.g. Yield/EUR/EUR1D, uniquely identifies the curve to be built. Curve Yield/EUR/EUR1D is defined in the curve configuration file explained in section 6.5 below. In this case ORE is instructed to build an Eonia Swap curve made of Overnight Deposit and Eonia Swap quotes. The right most token of the string Yield/EUR/EUR1D (EUR1D) is user defined, the first two tokens Yield/EUR have to be used to point to a yield curve in currency EUR.

6.2.2 Index Curves

See an excerpt of the index curve block with ID 'default' from the same example file:

Listing 7: Index curve block with ID 'default'

```
<IndexForwardingCurves Id="default">
  <Index Name="EUR-EURIBOR-3M">Yield/EUR/EUR3M</Index>
  <Index Name="EUR-EURIBOR-6M">Yield/EUR/EUR6M</Index>
  <Index Name="EUR-EURIBOR-12M">Yield/EUR/EUR6M</Index>
  <Index Name="EUR-EONIA">Yield/EUR/EUR1D</Index>
  <Index Name="USD-LIBOR-3M">Yield/USD/USD3M</Index>
  <!-- ... -->
</IndexForwardingCurves>
```

This block of curve specifications instructs ORE to build another set of yield curves, unique strings (e.g. Yield/EUR/EUR6M etc.) point to the `curveconfig.xml` file where these curves are defined. Each curve is then associated with an index name (of format Ccy-IndexName-Tenor, e.g. EUR-EURIBOR-6M) so that ORE will project the respective index using the selected curve (e.g. Yield/EUR/EUR6M).

6.2.3 Swap Index Curves

The following is an excerpt of the swap index curve block with ID 'default' from the same example file:

Listing 8: Swap index curve block with ID 'default'

```
<SwapIndexCurves Id="default">
  <SwapIndex Name="EUR-CMS-1Y">
    <Index>EUR-EURIBOR-6M</Index>
    <Discounting>EUR-EONIA</Discounting>
  </SwapIndex>
  <SwapIndex Name="EUR-CMS-30Y">
    <Index>EUR-EURIBOR-6M</Index>
    <Discounting>EUR-EONIA</Discounting>
  </SwapIndex>
  <!-- ... -->
</SwapIndexCurves>
```

These instructions do not build any additional curves. They only build the respective swap index objects and associate them with the required index forwarding and discounting curves already built above. This enables a swap index to project the fair rate of forward starting Swaps. Swap indices are also containers for conventions. Swaption volatility surfaces require two swap indices each available in the market object, a long term and a short term swap index. The curve configuration file below will show that in particular the required short term index has term 1Y, and the required long term index has 30Y term. This is why we build these two indices at this point.

6.2.4 FX Spot

The following is an excerpt of the FX spot block with ID 'default' from the same example file:

Listing 9: FX spot block with ID 'default'

```
<FxSpots Id="default">
  <FxSpot Pair="EURUSD">FX/EUR/USD</FxSpot>
  <FxSpot Pair="EURGBP">FX/EUR/GBP</FxSpot>
  <FxSpot Pair="EURCHF">FX/EUR/CHF</FxSpot>
  <FxSpot Pair="EURJPY">FX/EUR/JPY</FxSpot>
  <!-- ... -->
</FxSpots>
```

This block instructs ORE to provide four FX quote objects in the market object, all quoted with target currency EUR so that foreign currency amounts can be converted into EUR via multiplication with that rate.

6.2.5 FX Volatilities

The following is an excerpt of the FX Volatilities block with ID 'default' from the same example file:

Listing 10: FX volatility block with ID 'default'

```
<FxVolatilities Id="default">
  <FxVolatility Pair="EURUSD">FXVolatility/EUR/USD/EURUSD</FxVolatility>
  <FxVolatility Pair="EURGBP">FXVolatility/EUR/GBP/EURGBP</FxVolatility>
  <FxVolatility Pair="EURCHF">FXVolatility/EUR/CHF/EURCHF</FxVolatility>
  <FxVolatility Pair="EURJPY">FXVolatility/EUR/JPY/EURJPY</FxVolatility>
  <!-- ... -->
</FxVolatilities>
```

This instructs ORE to build four FX volatility structures for all FX pairs with target currency EUR, see curve configuration file for the definition of the volatility structure.

6.2.6 Swaption Volatilities

The following is an excerpt of the Swaption Volatilities block with ID 'default' from the same example file:

Listing 11: Swaption volatility block with ID 'default'

```
<SwaptionVolatilities Id="default">
  <SwaptionVolatility Currency="EUR">SwaptionVolatility/EUR/EUR_SW_N</SwaptionVolatility>
  <SwaptionVolatility Currency="USD">SwaptionVolatility/USD/USD_SW_N</SwaptionVolatility>
  <SwaptionVolatility Currency="GBP">SwaptionVolatility/GBP/GBP_SW_N</SwaptionVolatility>
  <SwaptionVolatility Currency="CHF">SwaptionVolatility/CHF/CHF_SW_N</SwaptionVolatility>
  <SwaptionVolatility Currency="JPY">SwaptionVolatility/CHF/JPY_SW_N</SwaptionVolatility>
</SwaptionVolatilities>
```

This instructs ORE to build five Swaption volatility structures, see the curve configuration file for the definition of the volatility structure. The latter token (e.g. EUR_SW_N) is user defined and will be found in the curve configuration's CurveId tag.

6.2.7 Cap/Floor Volatilities

The following is an excerpt of the Cap/Floor Volatilities block with ID 'default' from the same example file:

Listing 12: Cap/Floor volatility block with ID 'default'

```
<CapFloorVolatilities id="default">
  <CapFloorVolatility currency="EUR">CapFloorVolatility/EUR/EUR_CF_N</CapFloorVolatility>
  <CapFloorVolatility currency="USD">CapFloorVolatility/USD/USD_CF_N</CapFloorVolatility>
  <CapFloorVolatility currency="GBP">CapFloorVolatility/GBP/GBP_CF_N</CapFloorVolatility>
</CapFloorVolatilities>
```

This instructs ORE to build three Cap/Floor volatility structures, see the curve configuration file for the definition of the volatility structure. The latter token (e.g. EUR_CF_N) is user defined and will be found in the curve configuration's CurveId tag.

6.2.8 Default Curves

The following is an excerpt of the Default Curves block with ID 'default' from the same example file:

Listing 13: Default curves block with ID 'default'

```
<DefaultCurves Id="default">
  <DefaultCurve Name="BANK">Default/USD/BANK_SR_USD</DefaultCurve>
  <DefaultCurve Name="CPTY_A">Default/USD/CUST_A_SR_USD</DefaultCurve>
  <DefaultCurve Name="CPTY_B">Default/USD/CUST_A_SR_USD</DefaultCurve>
  <!-- ... -->
</DefaultCurves>
```

This instructs ORE to build a set of default probability curves, again defined in the curve configuration file. Each curve is then associated with a name (BANK, CUST_A) for subsequent lookup. As before, the last token (e.g. BANK_SR_USD) is user defined and will be found in the curve configuration's CurveId tag.

6.2.9 Market Configurations

Finally, representatives of each type of block (Discount Curves, Index Curves, Volatility structures, etc, up to Default curves) can be bundled into a market configuration. This is done by adding the following to the `todaysmarket.xml` file:

Listing 14: Market configurations

```
<Configuration Id="default">
  <DiscountingCurvesId> xois_eur </DiscountingCurvesId>
</Configuration>
<Configuration Id="collateral_inccy">
  <DiscountingCurvesId>ois</DiscountingCurvesId>
</Configuration>
<Configuration Id="collateral_eur">
  <DiscountingCurvesId>xois_eur</DiscountingCurvesId>
</Configuration>
<Configuration Id="libor">
  <DiscountingCurvesId>inccy_swap</DiscountingCurvesId>
</Configuration>
```

When ORE constructs the market object, all market configurations will be build and labelled using the 'Configuration Id'. This allows configuring a market setup for different alternative purposes side by side in the same `todaysmarket.xml` file. Typical use cases are

- different discount curves needed for model calibration and risk factor evolution, respectively
- different discount curves needed for collateralised and uncollateralised derivatives pricing.

The former is actually used throughout the `Examples` section. Each master input file `ore.xml` has a Markets section (see 6.1) where four market configuration IDs have to be

provided - the ones used for 'lgmcalibration', 'fxcalibration', 'pricing' and 'simulation' (i.e. risk factor evolution).

The configuration ID concept extends across all curve and volatility objects though currently used only to distinguish discounting.

6.3 Pricing Engines: `pricingengine.xml`

The pricing engine configuration file is provided to select pricing models and pricing engines by product type. The following excerpt of the Example section's `pricingengine.xml` shows the selection for Bermudan Swaption pricing:

Listing 15: Pricing engine configuration

```
<PricingEngines>
  <Product type="Swap">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingSwapEngine</Engine>
    <EngineParameters/>
  </Product>
  <Product type="CrossCurrencySwap">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingCrossCurrencySwapEngine</Engine>
    <EngineParameters/>
  </Product>
  <Product type="FxForward">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingFxForwardEngine</Engine>
    <EngineParameters/>
  </Product>
  <Product type="FxOption">
    <Model>GarmanKohlhagen</Model>
    <ModelParameters/>
    <Engine>AnalyticEuropeanEngine</Engine>
    <EngineParameters/>
  </Product>
  <Product type="EuropeanSwaption">
    <Model>BlackBachelier</Model> <!-- depends on input vol -->
    <ModelParameters/>
    <Engine>BlackBachelierSwaptionEngine</Engine>
    <EngineParameters/>
  </Product>
  <Product type="BermudanSwaption">
    <Model>LGM</Model>
    <ModelParameters>
      <Parameter name="Calibration">Bootstrap</Parameter>
      <Parameter name="BermudanStrategy">CoterminalATM</Parameter>
      <Parameter name="Reversion">0.03</Parameter>
      <Parameter name="ReversionType">HullWhite</Parameter>
      <Parameter name="Volatility">0.01</Parameter>
      <Parameter name="VolatilityType">Hagan</Parameter>
      <Parameter name="Tolerance">0.0001</Parameter>
    </ModelParameters>
    <Engine>Grid</Engine>
    <EngineParameters>
      <Parameter name="sy">3.0</Parameter>
      <Parameter name="ny">10</Parameter>
      <Parameter name="sx">3.0</Parameter>
      <Parameter name="nx">10</Parameter>
    </EngineParameters>
  </Product>
</PricingEngines>
```

These settings will be taken into account when the engine factory is asked to build a Bermudan Swaption pricing model, calibrate it and construct the pricing engine for it:

- The only model currently supported for Bermudan Swaption pricing is the LGM

selected here.

- The first block of model parameters then provides initial values for the model (Reversion, Volatility) and chooses the parametrisation of the LGM model with ReversionType and VolatilityType choices *HullWhite* and *Hagan*. Calibration and BermudanStrategy can be set to *None* in order to skip model calibration. Alternatively, Calibration is set to *Bootstrap* and BermudanStrategy to *CoterminalATM* in order to calibrate to instrument-specific co-terminal ATM Swaptions, i.e. chosen to match the instruments first expiry and final maturity.

- The second block of engine parameters specifies the Numerical Swaption engine parameters which determine the number of standard deviations covered in the probability density integrals (sy and sx), and the number of grid points used per standard deviation (ny and nx).

This file is relevant in particular for structured products which are in scope of future ORE releases. But it is also intended to allow the selection of optimised pricing engines for vanilla products such as Interest Rate Swaps.

6.4 Simulation: `simulation.xml`

This file determines the behaviour of the risk factor simulation (scenario generation) module. It is structured in three blocks of data.

Listing 16: Simulation configuration

```
<Simulation>
  <Parameters> ... </Parameters>
  <CrossAssetModel> ... </CrossAssetModel>
  <Market> ... </Market>
</Simulation>
```

Each of the three blocks is sketched in the following.

6.4.1 Parameters

Let us discuss this section using the following example

Listing 17: Simulation configuration

```
<Parameters>
  <Discretization>Exact</Discretization>
  <Grid>80,3M</Grid>
  <Calendar>EUR,USD,GBP,CHF</Calendar>
  <Sequence>SobolBrownianBridge</Sequence>
  <Seed>42</Seed>
  <Samples>1000</Samples>
</Parameters>
```

- `Discretization`: Chooses between time discretization schemes for the risk factor evolution. *Exact* means exploiting the analytcal tractability of the model to avoid any time discretization error. *Euler* uses a naive time discretization scheme which

has numerical error and requires small time steps for accurate results (useful for testing purposes)

- `Grid:` Specifies the simulation time grid, here 80 quarterly steps.
- `Calendar:` Calendar or combination of calendars used to adjust the dates of the grid. Date adjustment is required because the simulation must step over 'good' dates on which index fixings can be stored. more efficient memory usage.
- `Sequence:` Choose random sequence generator (*MersenneTwister, MersenneTwister-Antithetic, Sobol, SobolBrownianBridge*).
- `Seed:` Random number generator seed
- `Samples:` Number of Monte Carlo paths to be produced use (*Backward, Forward, BestOfForwardBackward, InterpolatedForwardBackward*), which number of forward horizon days to use if one of the *Forward* related methods is chosen.

6.4.2 Model

The `CrossAssetModel` section determines the cross asset model's number of currencies covered, composition, and each component's calibration. It is currently made of a sequence of LGM models for each currency (say n currencies), $n-1$ FX models for each exchange rate to the base currency, and finally, a specification of the correlation structure between all components.

The simulated currencies are specified as follows, with clearly identifying the domestic currency which is also the target currency for all FX models listed subsequently. If the portfolio requires more currencies to be simulated, this will lead to an exception at run time, so that it is the user's responsibility to make sure that the list of currencies here is sufficient. The list can be larger than actually required by the portfolio. This will not lead to any exceptions, but add to the run time of ORE.

Listing 18: Simulation model currencies configuration

```
<CrossAssetModel>
  <DomesticCcy>EUR</DomesticCcy>
  <Currencies>
    <Currency>EUR</Currency>
    <Currency>USD</Currency>
    <Currency>GBP</Currency>
    <Currency>CHF</Currency>
    <Currency>JPY</Currency>
  </Currencies>
  <BootstrapTolerance>0.0001</BootstrapTolerance>
  <!-- ... -->
</CrossAssetModel>
```

Bootstrap tolerance is a global parameter that applies to the calibration of all model components. If the calibration error of any component exceeds this tolerance, this will trigger an exception at runtime, early in the ORE process.

Each interest rate model is specified by a block as follows

Listing 19: Simulation model IR configuration

```
<CrossAssetModel>
  <!-- ... -->
  <InterestRateModels>
    <LGM ccy="default">
      <CalibrationType>Bootstrap</CalibrationType>
      <Volatility>
        <Calibrate>Y</Calibrate>
        <VolatilityType>Hagan</VolatilityType>
        <ParamType>Piecewise</ParamType>
        <TimeGrid>1.0,2.0,3.0,4.0,5.0,7.0,10.0</TimeGrid>
        <InitialValue>0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01<InitialValue>
      </Volatility>
      <Reversion>
        <Calibrate>N</Calibrate>
        <ReversionType>HullWhite</ReversionType>
        <ParamType>Constant</ParamType>
        <TimeGrid/>
        <InitialValue>0.03</InitialValue>
      </Reversion>
      <CalibrationSwaptions>
        <Expiries>1Y,2Y,4Y,6Y,8Y,10Y,12Y,14Y,16Y,18Y,19Y</Expiries>
        <Terms>19Y,18Y,16Y,14Y,12Y,10Y,8Y,6Y,4Y,2Y,1Y</Terms>
        <Strikes/>
      </CalibrationSwaptions>
      <ParameterTransformation>
        <ShiftHorizon>0.0</ShiftHorizon>
        <Scaling>1.0</Scaling>
      </ParameterTransformation>
    </LGM>
    <LGM ccy="EUR">
      <!-- ... -->
    </LGM>
    <LGM ccy="USD">
      <!-- ... -->
    </LGM>
  </InterestRateModels>
  <!-- ... -->
</CrossAssetModel>
```

We have LGM sections by currency, but starting with a section for currency 'default'. As the name implies, this is used as default configuration for any currency in the currency list for which we do not provide an explicit parametrisation. Within each LGM section, the interpretation of elements is as follows:

- `CalibrationType`: Choose between *Bootstrap* and *BestFit*, where Bootstrap is chosen when we expect to be able to achieve a perfect fit (as with calibration of piecewise volatility to a series of co-terminal Swaptions)
- `Volatility/Calibrate`: Flag to enable/disable calibration of this particular parameter
- `Volatility/VolatilityType`: Choose volatility parametrisation a la *Hull-White* or *Hagan*
- `Volatility/ParamType`: Choose between *Constant* and *Piecewise*

- `Volatility/TimeGrid:` Initial time grid for this parameter, can be left empty if ParamType is Constant
- `Volatility/InitialValue:` Vector of initial values, matching number of entries in time, or single value if the time grid is empty
- `Reversion/Calibrate:` Flag to enable/disable calibration of this particular parameter
- `Reversion/VolatilityType:` Choose reversion parametrisation a la *HullWhite* or *Hagan*
- `Reversion/ParamType:` Choose between *Constant* and *Piecewise*
- `Reversion/TimeGrid:` Initial time grid for this parameter, can be left empty if ParamType is Constant
- `Reversion/InitialValue:` Vector of initial values, matching number of entries in time, or single value if the time grid is empty
- `CalibrationSwaptions:` Choice of calibration instruments by expiry, underlying Swap term and strike
- `ParameterTransformation:` LGM model prices are invariant under scaling and shift transformations [17] with advantages for numerical convergence of results in long term simulations. These transformations can be chosen here. Default settings are shiftHorizon 0 (time in years) and scaling factor 1.

Each FX model is specified by a block as follows

Listing 20: Simulation model FX configuration

```
<CrossAssetModel>
  <!-- ... -->
  <ForeignExchangeModels>
    <CrossCcyLGM foreignCcy="default">
      <DomesticCcy>EUR</DomesticCcy>
      <CalibrationType>Bootstrap</CalibrationType>
      <Sigma>
        <Calibrate>Y</Calibrate>
        <ParamType>Piecewise</ParamType>
        <TimeGrid>1.0,2.0,3.0,4.0,5.0,7.0,10.0</TimeGrid>
        <InitialValue>0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1</InitialValue>
      </Sigma>
      <CalibrationOptions>
        <Expiries>1Y,2Y,3Y,4Y,5Y,10Y</Expiries>
        <Strikes/>
      </CalibrationOptions>
    </CrossCcyLGM>
    <CrossCcyLGM foreignCcy="USD">
      <!-- ... -->
    </CrossCcyLGM>
    <CrossCcyLGM foreignCcy="GBP">
      <!-- ... -->
    </CrossCcyLGM>
    <!-- ... -->
  </ForeignExchangeModels>
  <!-- ... -->
<CrossAssetModel>
```

CrossCcyLGM sections are defined by foreign currency, but we also support a default configuration as above for the IR model parametrisations. Within each CrossCcyLGM section, the interpretation of elements is as follows:

- `DomesticCcy:` Domestic currency completing the FX pair
- `CalibrationType:` Choose between *Bootstrap* and *BestFit* as in the IR section
- `Sigma/Calibrate:` Flag to enable/disable calibration of this particular parameter
- `Sigma/ParamType:` Choose between *Constant* and *Piecewise*
- `Sigma/TimeGrid:` Initial time grid for this parameter, can be left empty if ParamType is Constant
- `Sigma/InitialValue:` Vector of initial values, matching number of entries in time, or single value if the time grid is empty
- `CalibrationOptions:` Choice of calibration instruments by expiry and strike, strikes can be empty (implying the default, ATMF options), or explicitly specified (in terms of FX rates as absolute strike values, in delta notation such as $\pm 25D$, $ATMF$ for at the money)

Finally, the instantaneous correlation structure is specified as follows.

Listing 21: Simulation model correlation configuration

```
<CrossAssetModel>
  <!-- ... -->
  <InstantaneousCorrelations>
    <Correlation factor1="IR:EUR" factor2="IR:USD">0.3</Correlation>
    <Correlation factor1="IR:EUR" factor2="IR:GBP">0.3</Correlation>
    <Correlation factor1="IR:USD" factor2="IR:GBP">0.3</Correlation>
    <Correlation factor1="IR:EUR" factor2="FX:USDEUR">0</Correlation>
    <Correlation factor1="IR:EUR" factor2="FX:GBPEUR">0</Correlation>
    <Correlation factor1="IR:GBP" factor2="FX:USDEUR">0</Correlation>
    <Correlation factor1="IR:GBP" factor2="FX:GBPEUR">0</Correlation>
    <Correlation factor1="IR:USD" factor2="FX:USDEUR">0</Correlation>
    <Correlation factor1="IR:USD" factor2="FX:GBPEUR">0</Correlation>
    <Correlation factor1="FX:USDEUR" factor2="FX:GBPEUR">0</Correlation>
    <!-- ... -->
  </InstantaneousCorrelations>
</CrossAssetModel>
```

Any risk factor pair not specified explicitly here will be assumed to have zero correlation.

6.4.3 Market

The last part of the simulation configuration file covers the specification of the simulated market. Note that the simulation model will yield the evolution of risk factors such as short rates which need to be translated into entire yield curves that can be 'understood' by the instruments which we want to price under scenarios. Moreover we need to specify how volatility structures evolve even if we do not explicitly simulate volatility. This translation happens based on the information in the *simulation market* object, which is configured in the section within the enclosing tags `<Market>` and `</Market`, as shown in the following small two-currency example.

Listing 22: Simulation market configuration

```
<Market>
  <BaseCurrency>EUR</BaseCurrency>
  <Currencies>
    <Currency>EUR</Currency>
    <Currency>USD</Currency>
  </Currencies>
  <YieldCurves>
    <Configuration>
      <Tenors>3M,6M,1Y,2Y,3Y,4Y,5Y,7Y,10Y,12Y,15Y,20Y</Tenors>
      <Interpolation>LogLinear</Interpolation>
      <Extrapolation>Y</Extrapolation>
    </Configuration>
  </YieldCurves>
  <Indices>
    <Index>EUR-EURIBOR-6M</Index>
    <Index>EUR-EURIBOR-3M</Index>
    <Index>EUR-EONIA</Index>
    <Index>USD-LIBOR-3M</Index>
  </Indices>
  <SwapIndices>
    <SwapIndex>
      <Name>EUR-CMS-1Y</Name>
      <ForwardingIndex>EUR-EURIBOR-6M</ForwardingIndex>
      <DiscountingIndex>EUR-EONIA</DiscountingIndex>
    </SwapIndex>
  </SwapIndices>
  <SwaptionVolatilities>
    <ReactionToTimeDecay>ForwardVariance</ReactionToTimeDecay>
    <Currencies>
      <Currency>EUR</Currency>
      <Currency>USD</Currency>
    </Currencies>
    <Expiries>6M,1Y,2Y,3Y,5Y,10Y,12Y,15Y,20Y</Expiries>
    <Terms>1Y,2Y,3Y,4Y,5Y,7Y,10Y,15Y,20Y,30Y</Terms>
    <Strikes/>
  </SwaptionVolatilities>
  <CapFloorVolatilities>
    <ReactionToTimeDecay>ConstantVariance</ReactionToTimeDecay>
    <Currencies>
      <Currency>EUR</Currency>
      <Currency>USD</Currency>
    </Currencies>
  </CapFloorVolatilities>
  <FxVolatilities>
    <ReactionToTimeDecay>ForwardVariance</ReactionToTimeDecay>
    <CurrencyPairs>
      <CurrencyPair>EURUSD</CurrencyPair>
    </CurrencyPairs>
    <Expiries>6M,1Y,2Y,3Y,4Y,5Y,7Y,10Y</Expiries>
    <Strikes/>
  </FxVolatilities>
  <AdditionalScenarioDataCurrencies>
    <Currency>EUR</Currency>
    <Currency>USD</Currency>
  </AdditionalScenarioDataCurrencies>
  <AdditionalScenarioDataIndices>
    <Index>EUR-EURIBOR-3M</Index>
    <Index>EUR-EONIA</Index>
    <Index>USD-LIBOR-3M</Index>
  </AdditionalScenarioDataIndices>
</Market>
```

6.5 Curves: `curveconfig.xml`

The configuration of various term structures required to price a portfolio is covered in a single configuration file which we will label `curveconfig.xml` in the following though the file name can be chosen by the user. This configuration determines the composition of yield curves, default curves, Swaption and FX Option volatility structures.

6.5.1 Yield Curves

The top level XML elements for each `YieldCurve` node are shown in Listing 23.

Listing 23: Top level yield curve node

```
<YieldCurve>
  <CurveId> </CurveId>
  <CurveDescription> </CurveDescription>
  <Currency> </Currency>
  <DiscountCurve> </DiscountCurve>
  <Segments> </Segments>
  <InterpolationVariable> </InterpolationVariable>
  <InterpolationMethod> </InterpolationMethod>
  <ZeroDayCounter> </ZeroDayCounter>
  <Tolerance> </Tolerance>
  <Extrapolation> </Extrapolation>
</YieldCurve>
```

The meaning of each of the top level elements in Listing 23 is given below. If an element is labelled as 'Optional', then it may be excluded or included and left blank.

- CurveId: Unique identifier for the yield curve.
- CurveDescription: A description of the yield curve. This field may be left blank.
- Currency: The yield curve currency.
- DiscountCurve: If the yield curve is being bootstrapped from market instruments, this gives the CurveId of the yield curve used to discount cash flows during the bootstrap procedure. If this field is left blank or set equal to the current CurveId, then this yield curve itself is used to discount cash flows during the bootstrap procedure.
- Segments: This element contains child elements and is described in the following subsection.
- InterpolationVariable [Optional]: The variable on which the interpolation is performed. The allowable values are given in Table 4. If the element is omitted or left blank, then it defaults to *Discount*.
- InterpolationMethod [Optional]: The interpolation method to use. The allowable values are given in Table 5. If the element is omitted or left blank, then it defaults to *LogLinear*.
- ZeroDayCounter [Optional]: The day count basis used internally by the yield curve to calculate the time between dates. In particular, if the curve is queried for a zero rate without specifying the day count basis, the zero rate that is returned has this basis. If the element is omitted or left blank, then it defaults to *A365*.
- Tolerance [Optional]: The tolerance used by the root finding procedure in the bootstrapping algorithm. If the element is omitted or left blank, then it defaults to 1.0×10^{-12}.
- Extrapolation [Optional]: Set to *True* or *False* to enable or disable extrapolation respectively. If the element is omitted or left blank, then it defaults to *True*.

Variable	Description
Zero	The continuously compounded zero rate
Discount	The discount factor

Table 4: Allowable interpolation variables.

Method	Description
Linear	Linear interpolation
LogLinear	Linear interpolation on the natural log of the interpolation variable
NaturalCubic	Monotonic Kruger cubic interpolation with second derivative at left and right
FinancialCubic	Monotonic Kruger cubic interpolation with second derivative at left and first derivative at right

Table 5: Allowable interpolation methods.

Segments Node

The `Segments` node gives the zero rates, discount factors and instruments that comprise the yield curve. This node consists of a number of child nodes where the node name depends on the segment being described. Each node has a `Type` that determines its structure. The following sections describe the type of child nodes that are available.

Direct Segment

When the node name is `Direct`, the `Type` node has the value *Zero* or *Discount* and the node has the structure shown in Listing 24. We refer to this segment here as a direct segment because the discount factors, or equivalently the zero rates, are given explicitly and do not need to be bootstrapped. The `Quotes` node contains a list of `Quote` elements. Each `Quote` element contains an ID pointing to a line in the `market.txt` file, i.e. in this case, pointing to a particular zero rate or discount factor. The `Conventions` node contains the ID of a node in the `conventions.xml` file described in section 6.6. The `Conventions` node associates conventions with the quotes.

Listing 24: Direct yield curve segment

```
<Direct>
  <Type> </Type>
  <Quotes>
    <Quote> </Quote>
    <Quote> </Quote>
     <!--...-->
  </Quotes>
  <Conventions> </Conventions>
</Direct>
```

Simple Segment

When the node name is `Simple`, the `Type` node has the value *Deposit*, *FRA*, *Future*, *OIS* or *Swap* and the node has the structure shown in Listing 25. This segment holds

quotes for a set of deposit, FRA, Future, OIS or swap instruments corresponding to the value in the `Type` node. These quotes will be used by the bootstrap algorithm to imply a discount factor, or equivalently a zero rate, curve. The only difference between this segment and the direct segment is that there is a `ProjectionCurve` node. This node allows us to specify the CurveId of another curve to project floating rates on the instruments underlying the quotes listed in the `Quote` nodes during the bootstrap procedure. This is an optional node. If it is left blank or omitted, then the projection curve is assumed to equal the curve being bootstrapped i.e. the current CurveId.

Listing 25: Simple yield curve segment

```
<Simple>
  <Type> </Type>
  <Quotes>
    <Quote> </Quote>
    <Quote> </Quote>
    <!--...-->
  </Quotes>
  <Conventions> </Conventions>
  <ProjectionCurve> </ProjectionCurve>
</Simple>
```

Average OIS Segment

When the node name is `AverageOIS`, the `Type` node has the value *Average OIS* and the node has the structure shown in Listing 26. This segment is used to hold quotes for Average OIS swap instruments. The `Quotes` node has the structure shown in Listing 27. Each quote for an Average OIS instrument (a typical example in a USD Overnight Index Swap) consists of two quotes, a vanilla IRS quote and an OIS-LIBOR basis swap spread quote. The IDs of these two quotes are stored in the `CompositeQuote` node. The `RateQuote` node holds the ID of the vanilla IRS quote and the `SpreadQuote` node holds the ID of the OIS-LIBOR basis swap spread quote.

Listing 26: Average OIS yield curve segment

```
<AverageOIS>
  <Type> </Type>
  <Quotes>
    <CompositeQuote> </CompositeQuote>
    <CompositeQuote> </CompositeQuote>
    <!--...-->
  </Quotes>
  <Conventions> </Conventions>
  <ProjectionCurve> </ProjectionCurve>
</AverageOIS>
```

Listing 27: Average OIS segment's quotes section

```
<Quotes>
  <CompositeQuote>
    <SpreadQuote> </SpreadQuote>
    <RateQuote> </RateQuote>
  </CompositeQuote>
  <!--...-->
</Quotes>
```

Tenor Basis Segment

When the node name is `TenorBasis`, the `Type` node has the value *Tenor Basis Swap* or *Tenor Basis Two Swaps* and the node has the structure shown in Listing 28. This segment is used to hold quotes for tenor basis swap instruments. The quotes may be for a conventional tenor basis swap where Ibor of one tenor is swapped for Ibor of another tenor plus a spread. In this case, the `Type` node has the value *Tenor Basis Swap*. The quotes may also be for the difference in fixed rates on two fair swaps where one swap is against Ibor of one tenor and the other swap is against Ibor of another tenor. In this case, the `Type` node has the value *Tenor Basis Two Swaps*. Again, the structure is similar to the simple segment in Listing 25 except that there are two projection curve nodes. There is a `ProjectionCurveShort` node for the index with the shorter tenor. This node holds the CurveId of a curve for projecting the floating rates on the short tenor index. Similarly, there is a `ProjectionCurveLong` node for the index with the longer tenor. This node holds the CurveId of a curve for projecting the floating rates on the long tenor index. These are optional nodes. If they are left blank or omitted, then the projection curve is assumed to equal the curve being bootstrapped i.e. the current CurveId. However, at least one of the nodes needs to be populated to allow the bootstrap to proceed.

Listing 28: Tenor basis yield curve segment

```
<TenorBasis>
  <Type> </Type>
  <Quotes>
    <Quote> </Quote>
    <Quote> </Quote>
    <!--...-->
  </Quotes>
  <Conventions> </Conventions>
  <ProjectionCurveLong> </ProjectionCurveLong>
  <ProjectionCurveShort> </ProjectionCurveShort>
</TenorBasis>
```

Cross Currency Segment

When the node name is `CrossCurrency`, the `Type` node has the value *FX Forward* or *Cross Currency Basis Swap*. When the `Type` node has the value *FX Forward*, the node has the structure shown in Listing 29. This segment is used to hold quotes for FX forward instruments. The `DiscountCurve` node holds the CurveId of a curve used to discount cash flows in the other currency i.e. the currency in the currency pair that is

not equal to the currency in Listing 23. The `SpotRate` node holds the ID of a spot FX quote for the currency pair that is looked up in the `market.txt` file.

Listing 29: FX forward yield curve segment

```
<CrossCurrency>
  <Type> </Type>
  <Quotes>
    <Quote> </Quote>
    <Quote> </Quote>
          ...
  </Quotes>
  <Conventions> </Conventions>
  <DiscountCurve> </DiscountCurve>
  <SpotRate> </SpotRate>
</CrossCurrency>
```

When the `Type` node has the value *Cross Currency Basis Swap* then the node has the structure shown in Listing 30. This segment is used to hold quotes for cross currency basis swap instruments. The `DiscountCurve` node holds the CurveId of a curve used to discount cash flows in the other currency i.e. the currency in the currency pair that is not equal to the currency in Listing 23. The `SpotRate` node holds the ID of a spot FX quote for the currency pair that is looked up in the `market.txt` file. The `ProjectionCurveDomestic` node holds the CurveId of a curve for projecting the floating rates on the index in this currency i.e. the currency in the currency pair that is equal to the currency in Listing 23. It is an optional node and if it is left blank or omitted, then the projection curve is assumed to equal the curve being bootstrapped i.e. the current CurveId. Similarly, the `ProjectionCurveForeign` node holds the CurveId of a curve for projecting the floating rates on the index in the other currency. If it is left blank or omitted, then it is assumed to equal the CurveId provided in the `DiscountCurve` node in this segment.

Listing 30: Cross currency basis yield curve segment

```
<CrossCurrency>
  <Type> </Type>
  <Quotes>
    <Quote> </Quote>
    <Quote> </Quote>
          ...
  </Quotes>
  <Conventions> </Conventions>
  <DiscountCurve> </DiscountCurve>
  <SpotRate> </SpotRate>
  <ProjectionCurveDomestic> </ProjectionCurveDomestic>
  <ProjectionCurveForeign> </ProjectionCurveForeign>
</CrossCurrency>
```

Zero Spread Segment

When the node name is `ZeroSpread`, the `Type` node has the only allowable value *Zero Spread*, and the node has the structure shown in Listing 31. This segment is used to build yield curves which are expressed as a spread over some reference yield curve.

Listing 31: Cross currency basis yield curve segment

```
<ZeroSpread>
      <Type>Zero Spread</Type>
      <Quotes>
        <Quote>ZERO/YIELD_SPREAD/EUR/BANK_EUR_LEND/A365/2Y</Quote>
        <Quote>ZERO/YIELD_SPREAD/EUR/BANK_EUR_LEND/A365/5Y</Quote>
        <Quote>ZERO/YIELD_SPREAD/EUR/BANK_EUR_LEND/A365/10Y</Quote>
        <Quote>ZERO/YIELD_SPREAD/EUR/BANK_EUR_LEND/A365/20Y</Quote>
      </Quotes>
      <Conventions>EUR-ZERO-CONVENTIONS-TENOR-BASED</Conventions>
      <ReferenceCurve>EUR1D</ReferenceCurve>
</ZeroSpread>
```

6.5.2 Default Curves

Listing 32: Default curve configuration

```
<DefaultCurves>
  <DefaultCurve>
    <CurveId>BANK_SR_USD</CurveId>
    <CurveDescription>BANK SR CDS USD</CurveDescription>
    <Currency>USD</Currency>
    <Type>SpreadCDS</Type>
    <DiscountCurve>Yield/USD/USD3M</DiscountCurve>
    <DayCounter>A365</DayCounter>
    <RecoveryRate>RECOVERY_RATE/RATE/BANK/SR/USD</RecoveryRate>
    <Quotes>
      <Quote>CDS/CREDIT_SPREAD/BANK/SR/USD/1Y</Quote>
      <Quote>CDS/CREDIT_SPREAD/BANK/SR/USD/2Y</Quote>
      <Quote>CDS/CREDIT_SPREAD/BANK/SR/USD/3Y</Quote>
      <Quote>CDS/CREDIT_SPREAD/BANK/SR/USD/4Y</Quote>
      <Quote>CDS/CREDIT_SPREAD/BANK/SR/USD/5Y</Quote>
      <Quote>CDS/CREDIT_SPREAD/BANK/SR/USD/7Y</Quote>
      <Quote>CDS/CREDIT_SPREAD/BANK/SR/USD/10Y</Quote>
    </Quotes>
    <Conventions>CDS-STANDARD-CONVENTIONS</Conventions>
  </DefaultCurve>
  <DefaultCurve>
    <!-- ... -->
  </DefaultCurve>
</DefaultCurves>
```

6.5.3 Swaption Volatility Structures

Listing 33: Swaption volatility configuration

```
<SwaptionVolatilities>
  <SwaptionVolatility>
    <CurveId>EUR_SW_N</CurveId>
    <CurveDescription>EUR normal swaption volatilities</CurveDescription>
    <!-- ATM (Smile not yet supported) -->
    <Dimension>ATM</Dimension>
    <!-- Normal or Lognormal or ShiftedLognormal -->
    <VolatilityType>Normal</VolatilityType>
    <!-- Flat or Linear -->
    <Extrapolation>Flat</Extrapolation>
    <!-- Day counter for date to time conversion -->
    <DayCounter>Actual/365 (Fixed)</DayCounter>
    <!--Calendar and Business day convention for option tenor to date conversion -->
    <Calendar>TARGET</Calendar>
    <BusinessDayConvention>Following</BusinessDayConvention>
    <OptionTenors>
      1M,3M,6M,1Y,2Y,3Y,4Y,5Y,7Y,10Y,15Y,20Y,25Y,30Y
    </OptionTenors>
    <SwapTenors>
      1Y,2Y,3Y,4Y,5Y,7Y,10Y,15Y,20Y,25Y,30Y
    </SwapTenors>
    <ShortSwapIndexBase>EUR-CMS-1Y</ShortSwapIndexBase>
    <SwapIndexBase>EUR-CMS-30Y</SwapIndexBase>
  </SwaptionVolatility>
    <SwaptionVolatility>
      <!-- ... -->
    </SwaptionVolatility>
</SwaptionVolatilities>
```

6.5.4 FX Volatility Structures

Listing 34: FX option volatility configuration

```
<FXVolatilities>
  <FXVolatility>
    <CurveId>EURUSD</CurveId>
    <CurveDescription />
    <Dimension>ATM</Dimension>
    <Expiries>
      1M,3M,6M,1Y,2Y,3Y,10Y
    </Expiries>
  </FXVolatility>
  <FXVolatility>
    <!-- ... -->
  </FXVolatility>
</FXVolatilities>
```

6.6 Conventions: `conventions.xml`

The conventions to associate with a set market quotes in the construction of termstructures are specified in another xml file which we will refer to as `conventions.xml` in the following though the file name can be chosen by the user. Each separate set of conventions is stored in an XML node. The type of conventions that a node holds is determined by the node name. Every node has an `Id` node that gives a unique identifier for the convention set. The following sections describe the type of conventions that can be created and the allowed values.

6.6.1 Zero Conventions

A node with name *Zero* is used to store conventions for direct zero rate quotes. Direct zero rate quotes can be given with an explicit maturity date or with a tenor and a set of conventions from which the maturity date is deduced. The node for a zero rate quote with an explicit maturity date is shown in Listing 35. The node for a tenor based zero rate is shown in Listing 36.

Listing 35: Zero conventions

```
<Zero>
  <Id> </Id>
  <TenorBased>False</TenorBased>
  <DayCounter> </DayCounter>
  <CompoundingFrequency> </CompoundingFrequency>
  <Compounding> </Compounding>
</Zero>
```

Listing 36: Zero conventions, tenor based

```
<Zero>
  <Id> </Id>
  <TenorBased>True</TenorBased>
  <DayCounter> </DayCounter>
  <CompoundingFrequency> </CompoundingFrequency>
  <Compounding> </Compounding>
  <TenorCalendar> </TenorCalendar>
  <SpotLag> </SpotLag>
  <SpotCalendar> </SpotCalendar>
  <RollConvention> </RollConvention>
  <EOM> </EOM>
</Zero>
```

The meanings of the various elements in this node are as follows:

- TenorBased: True if the conventions are for a tenor based zero quote and False if they are for a zero quote with an explicit maturity date.
- DayCounter: The day count basis associated with the zero rate quote (for choices see section 7.4)
- CompoundingFrequency: The frequency of compounding (Choices are *Once, Annual, Semiannual, Quarterly, Bimonthly, Monthly, Weekly, Daily*).

- Compounding: The type of compounding for the zero rate (Choices are *Simple, Compounded, Continuous, SimpleThenCompounded*).
- TenorCalendar: The calendar used to advance from the spot date to the maturity date by the zero rate tenor (for choices see section 7.4).
- SpotLag [Optional]: The number of business days to advance from the valuation date before applying the zero rate tenor. If not provided, this defaults to 0.
- SpotCalendar [Optional]: The calendar to use for business days when applying the `SpotLag`. If not provided, it defaults to a calendar with no holidays.
- RollConvention [Optional]: The roll convention to use when applying the zero rate tenor. If not provided, it defaults to Following (Choices are *Backward, Forward, Zero, ThirdWednesday, Twentieth, TwentiethIMM, CDS*).
- EOM [Optional]: Whether or not to use the end of month convention when applying the zero rate tenor. If not provided, it defaults to false.

6.6.2 Deposit Conventions

A node with name *Deposit* is used to store conventions for deposit or index fixing quotes. The conventions can be index based, in which case all necessary conventions are deduced from a given index family. The structure of the index based node is shown in Listing 37. Alternatively, all the necessary conventions can be given explicitly without reference to an index family. The structure of this node is shown in Listing 38.

Listing 37: Deposit conventions

```
<Deposit>
  <Id> </Id>
  <IndexBased>True</IndexBased>
  <Index> </Index>
</Deposit>
```

Listing 38: Deposit conventions

```
<Deposit>
  <Id> </Id>
  <IndexBased>False</IndexBased>
  <Calendar> </Calendar>
  <Convention> </Convention>
  <EOM> </EOM>
  <DayCounter> </DayCounter>
</Deposit>
```

The meanings of the various elements in this node are as follows:

- IndexBased: *True* if the deposit conventions are index based and *False* if the conventions are given explicitly.
- Index: The index family from which to imply the conventions for the deposit quote. For example, this could be EUR-EURIBOR, USD-LIBOR etc.

- Calendar: The business day calendar for the deposit quote.
- Convention: The roll convention for the deposit quote.
- EOM: *True* if the end of month roll convention is to be used for the deposit quote and *False* if not.
- DayCounter: The day count basis associated with the deposit quote.

6.6.3 Future Conventions

A node with name *Future* is used to store conventions for IMM Future quotes. The structure of this node is shown in Listing 39. The only piece of information needed is the underlying money market index name and this is given in the `Index` node. For example, this could be EUR-EURIBOR-3M, USD-LIBOR-3M etc.

Listing 39: Future conventions

```
<Future>
  <Id> </Id>
  <Index> </Index>
</Future>
```

6.6.4 FRA Conventions

A node with name *FRA* is used to store conventions for FRA quotes. The structure of this node is shown in Listing 40. The only piece of information needed is the underlying index name and this is given in the `Index` node. For example, this could be EUR-EURIBOR-6M, CHF-LIBOR-6M etc.

Listing 40: FRA conventions

```
<FRA>
  <Id> </Id>
  <Index> </Index>
</FRA>
```

6.6.5 OIS Conventions

A node with name *OIS* is used to store conventions for Overnight Indexed Swap (OIS) quotes. The structure of this node is shown in Listing 41.

Listing 41: OIS conventions

```
<OIS>
  <Id> </Id>
  <SpotLag> </SpotLag>
  <Index> </Index>
  <FixedDayCounter> </FixedDayCounter>
  <PaymentLag> </PaymentLag>
  <EOM> </EOM>
  <FixedFrequency> </FixedFrequency>
  <FixedConvention> </FixedConvention>
  <FixedPaymentConvention> </FixedPaymentConvention>
  <Rule> </Rule>
</OIS>
```

The meanings of the various elements in this node are as follows:

- SpotLag: The number of business days until the start of the OIS.
- Index: The name of the overnight index. For example, this could be EUR-EONIA, USD-FedFunds etc.
- FixedDayCounter: The day count basis on the fixed leg of the OIS.
- PaymentLag [Optional]: The payment lag, as a number of business days, on both legs. If not provided, this defaults to 0.
- EOM [Optional]: *True* if the end of month roll convention is to be used when generating the OIS schedule and *False* if not. If not provided, this defaults to *False*.
- FixedFrequency [Optional]: The frequency of payments on the fixed leg. If not provided, this defaults to *Annual*.
- FixedConvention [Optional]: The roll convention for accruals on the fixed leg. If not provided, this defaults to *Following*.
- FixedPaymentConvention [Optional]: The roll convention for payments on the fixed leg. If not provided, this defaults to *Following*.
- Rule [Optional]: The rule used for generating the OIS dates schedule i.e. *Backward* or *Forward*. If not provided, this defaults to *Backward*.

6.6.6 Swap Conventions

A node with name *Swap* is used to store conventions for vanilla interest rate swap (IRS) quotes. The structure of this node is shown in Listing 42.

Listing 42: Swap conventions

```
<Swap>
  <Id> </Id>
  <FixedCalendar> </FixedCalendar>
  <FixedFrequency> </FixedFrequency>
  <FixedConvention> </FixedConvention>
  <FixedDayCounter> </FixedDayCounter>
  <Index> </Index>
</Swap>
```

The meanings of the various elements in this node are as follows:

- FixedCalendar: The business day calendar on the fixed leg.
- FixedFrequency: The frequency of payments on the fixed leg.
- FixedConvention: The roll convention on the fixed leg.
- FixedDayCounter: The day count basis on the fixed leg.
- Index: The Ibor index on the floating leg.

6.6.7 Average OIS Conventions

A node with name *AverageOIS* is used to store conventions for average OIS quotes. An average OIS is a swap where a fixed rate is swapped against a daily averaged overnight index plus a spread. The structure of this node is shown in Listing 43.

Listing 43: Average OIS conventions

```
<AverageOIS>
  <Id> </Id>
  <SpotLag> </SpotLag>
  <FixedTenor> </FixedTenor>
  <FixedDayCounter> </FixedDayCounter>
  <FixedCalendar> </FixedCalendar>
  <FixedConvention> </FixedConvention>
  <FixedPaymentConvention> </FixedPaymentConvention>
  <Index> </Index>
  <OnTenor> </OnTenor>
  <RateCutoff> </RateCutoff>
</AverageOIS>
```

The meanings of the various elements in this node are as follows:

- SpotLag: Number of business days until the start of the average OIS.
- FixedTenor: The frequency of payments on the fixed leg.
- FixedDayCounter: The day count basis on the fixed leg.
- FixedCalendar: The business day calendar on the fixed leg.
- FixedFrequency: The frequency of payments on the fixed leg.

- FixedConvention: The roll convention for accruals on the fixed leg.
- FixedPaymentConvention: The roll convention for payments on the fixed leg.
- Index: The name of the overnight index.
- OnTenor: The frequency of payments on the overnight leg.
- RateCutoff: The rate cut-off on the overnight leg. Generally, the overnight fixing is only observed up to a certain number of days before the payment date and the last observed rate is applied for the remaining days in the period. This rate cut-off gives the number of days e.g. 2 for Fed Funds average OIS.

6.6.8 Tenor Basis Swap Conventions

A node with name *TenorBasisSwap* is used to store conventions for tenor basis swap quotes. The structure of this node is shown in Listing 44.

Listing 44: Tenor basis swap conventions

```
<TenorBasisSwap>
  <Id> </Id>
  <LongIndex> </LongIndex>
  <ShortIndex> </ShortIndex>
  <ShortPayTenor> </ShortPayTenor>
  <SpreadOnShort> </SpreadOnShort>
  <IncludeSpread> </IncludeSpread>
  <SubPeriodsCouponType> </SubPeriodsCouponType>
</TenorBasisSwap>
```

The meanings of the various elements in this node are as follows:

- LongIndex: The name of the long tenor Ibor index.
- ShortIndex: The name of the short tenor Ibor index.
- ShortPayTenor [Optional]: The frequency of payments on the short tenor Ibor leg. This is usually the same as the short tenor Ibor index's tenor. However, it can also be longer e.g. USD tenor basis swaps where the short tenor Ibor index is compounded and paid on the same frequency as the long tenor Ibor index. If not provided, this defaults to the short tenor Ibor index's tenor.
- SpreadOnShort [Optional]: *True* if the tenor basis swap quote has the spread on the short tenor Ibor index leg and *False* if not. If not provided, this defaults to *True*.
- IncludeSpread [Optional]: *True* if the tenor basis swap spread is to be included when compounding is performed on the short tenor Ibor index leg and *False* if not. If not provided, this defaults to *False*.
- SubPeriodsCouponType [Optional]: This field can have the value *Compounding* or *Averaging* and it only applies when the frequency of payments on the short tenor Ibor leg does not equal the short tenor Ibor index's tenor. If *Compounding* is

specified, then the short tenor Ibor index is compounded and paid on the frequency specified in the `ShortPayTenor` node. If *Averaging* is specified, then the short tenor Ibor index is averaged and paid on the frequency specified in the `ShortPayTenor` node. If not provided, this defaults to *Compounding*.

6.6.9 Tenor Basis Two Swap Conventions

A node with name *TenorBasisTwoSwap* is used to store conventions for tenor basis swap quotes where the quote is the spread between the fair fixed rate on two swaps against Ibor indices of different tenors. We call the swap against the Ibor index of longer tenor the long swap and the remaining swap the short swap. The structure of the tenor basis two swap conventions node is shown in Listing 45.

Listing 45: Tenor basis two swap conventions

```
<TenorBasisTwoSwap>
  <Id> </Id>
  <Calendar> </Calendar>
  <LongFixedFrequency> </LongFixedFrequency>
  <LongFixedConvention> </LongFixedConvention>
  <LongFixedDayCounter> </LongFixedDayCounter>
  <LongIndex> </LongIndex>
  <ShortFixedFrequency> </ShortFixedFrequency>
  <ShortFixedConvention> </ShortFixedConvention>
  <ShortFixedDayCounter> </ShortFixedDayCounter>
  <ShortIndex> </ShortIndex>
  <LongMinusShort> </LongMinusShort>
</TenorBasisTwoSwap>
```

The meanings of the various elements in this node are as follows:

- Calendar: The business day calendar on both swaps.
- LongFixedFrequency: The frequency of payments on the fixed leg of the long swap.
- LongFixedConvention: The roll convention on the fixed leg of the long swap.
- LongFixedDayCounter: The day count basis on the fixed leg of the long swap.
- LongIndex: The Ibor index on the floating leg of the long swap.
- ShortFixedFrequency: The frequency of payments on the fixed leg of the short swap.
- ShortFixedConvention: The roll convention on the fixed leg of the short swap.
- ShortFixedDayCounter: The day count basis on the fixed leg of the short swap.
- ShortIndex: The Ibor index on the floating leg of the short swap.
- LongMinusShort [Optional]: *True* if the basis swap spread is to be interpreted as the fair rate on the long swap minus the fair rate on the short swap and *False* if the basis swap spread is to be interpreted as the fair rate on the short swap minus the fair rate on the long swap. If not provided, it defaults to *True*.

6.6.10 FX Conventions

A node with name *FX* is used to store conventions for FX spot and forward quotes for a given currency pair. The structure of this node is shown in Listing 46.

Listing 46: FX conventions

```
<FX>
  <Id> </Id>
  <SpotDays> </SpotDays>
  <SourceCurrency> </SourceCurrency>
  <TargetCurrency> </TargetCurrency>
  <PointsFactor> </PointsFactor>
  <AdvanceCalendar> </AdvanceCalendar>
  <SpotRelative> </SpotRelative>
  <AdditionalSettleCalendar> </AdditionalSettleCalendar>
</FX>
```

The meanings of the various elements in this node are as follows:

- SpotDays: The number of business days to spot for the currency pair.
- SourceCurrency: The source currency of the currency pair. The FX quote is assumed to give the number of units of target currency per unit of source currency.
- TargetCurrency: The target currency of the currency pair.
- PointsFactor: The number by which a points quote for the currency pair should be divided before adding it to the spot quote to obtain the forward rate.
- AdvanceCalendar [Optional]: The business day calendar(s) used for advancing dates for both spot and forwards. If not provided, it defaults to a calendar with no holidays.
- SpotRelative [Optional]: *True* if the forward tenor is to be interpreted as being relative to the spot date. *False* if the forward tenor is to be interpreted as being relative to the valuation date. If not provided, it defaults to *True*.
- AdditionalSettleCalendar [Optional]: In some cases, when the spot date is calculated using the values in the `AdvanceCalendar` and `SpotDays` nodes, it is checked against an additional settlement calendar(s) and if it is not a good business day then it is moved forward until it is a good business day on both the additional settlement calendar(s) and the AdvanceCalendar. This additional settlement calendar(s) can be specified here. If not provided, it defaults to a calendar with no holidays.

6.6.11 Cross Currency Basis Swap Conventions

A node with name *CrossCurrencyBasis* is used to store conventions for cross currency basis swap quotes. The structure of this node is shown in Listing 47.

Listing 47: Cross currency basis swap conventions

```
<CrossCurrencyBasis>
  <Id> </Id>
  <SettlementDays> </SettlementDays>
  <SettlementCalendar> </SettlementCalendar>
  <RollConvention> </RollConvention>
  <FlatIndex> </FlatIndex>
  <SpreadIndex> </SpreadIndex>
  <EOM> </EOM>
</CrossCurrencyBasis>
```

The meanings of the various elements in this node are as follows:

- SettlementDays: The number of business days to the start of the cross currency basis swap.
- SettlementCalendar: The business day calendar(s) for both legs and to arrive at the settlement date using the SettlementDays above.
- RollConvention: The roll convention for both legs.
- FlatIndex: The name of the index on the leg that does not have the cross currency basis spread.
- SpreadIndex: The name of the index on the leg that has the cross currency basis spread.
- EOM [Optional]: *True* if the end of month convention is to be used when generating the schedule on both legs and *False* if not. If not provided, it defaults to *False*.

7 Trade Data

The trades that make up the portfolio are specified in an XML file where the portfolio data is specified in a hierarchy of nodes and sub-nodes. The nodes containing individual trade data are referred to as elements or XML elements. These are generally the lowest level nodes.

The top level portfolio node is delimited by an opening `<Portfolio>` and a closing `</Portfolio>` tag. Within the portfolio node, each trade is defined by a starting `<Trade id="[Tradeid]">` and a closing `</Trade>` tag. Further, the trade type is set by the TradeType XML element. Each trade has an Envelope node that includes the same XML elements for all trade types (Id, Type, Counterparty, Rating, NettingSetId) plus the Additional fields node, and after that, a node containing trade specific data.

An example of a `portfolio.xml` file with one Swap trade including the full envelope node is shown in Listing 48.

Listing 48: Portfolio

```
<Portfolio>
  <Trade id="Swap#1">
    <TradeType> Swap </TradeType>
    <Envelope>
      <CounterParty> Counterparty#1 </CounterParty>
      <NettingSetId> NettingSet#2 </NettingSetId>
      <AdditionalFields>
        <Sector> SectorA </Sector>
        <Book> BookB </Book>
        <Rating> A1 </Rating>
      </AdditionalFields>
    </Envelope>
    <SwapData>
        ...
        [Trade specific data for a Swap]
        ...
    </SwapData>
  </Trade>
</Portfolio>
```

A description of all portfolio data, i.e. of each node and XML element in the portfolio file, with examples and allowable values follows below. There are two XML elements directly under the top level `Portfolio` node:

- `Trade id`: The first element of each trade is the `Trade id` and it is used to identify trades within a portfolio. Trade ids should be unique within a portfolio. The `Trade id` element is entered twice in the instrument file, firstly with an attribute to the XML element `<Trade>`, such as `<Trade id="ExampleTrade">` in the beginning closed by `</Trade>` at the end of the trade data, and secondly with a `Id` element in the envelope node. Both Trade id entries should be identical.

  Allowable values: Any alphanumeric string. The underscore (_) sign may be used as well.

- `TradeType`: The Trade type is set with the `TradeType` element, as well as with the `Type` element in the envelope. The two Trade type entries should be identical. ORE currently supports 5 trade types.

  Allowable values: *Swap, CapFloor, Swaption, FxForward, FxOption*

7.1 Envelope

The envelope node contains basic identifying details of a trade (`Id`, `Type`, `Counterparty`, `Rating`, `NettingSetId`), plus an `AdditionalFields` node where custom elements can be added for informational purposes such as `Book` or `Sector`. Beside the custom elements within the `AdditionalFields` node, the envelope contains the same elements for all Trade types. The `Id`, `Type`, `Counterparty` and `NettingSetId` elements must have non-blank entries for ORE to run, whereas ORE will run without a `Rating` element but fail to produce a CVA.
The meanings and allowable values of the various elements in the `Envelope` node follow below.

- `Id`: The `Id` element in the envelope is used to identify trades within a portfolio. It should be set to identical values as the `Trade id=" "` element.

  Allowable values: Any alphanumeric string. The underscore (_) sign may be used as well.

- `Type`: The Trade Type is in addition to being set in the `ClassName` element, also set by the `Type` element in the envelope. Both elements should have the same entry.

  Allowable values: *Swap, CapFloor, Swaption, FxForward, FxOption*

- `Counterparty`: Specifies the name of the counterparty of the trade. It is used to show exposure analytics by counterparty.

  Allowable values: Any alphanumeric string. Underscores (_) and blank spaces may be used as well.

- `Rating` [Optional]: The `Rating` element specifies the default curve that will be used in the CVA calculations. No CVA will be calculated if omitted or left blank.

  Allowable values: An alphanumeric string that matches default curve names in the market configuration file.

- `NettingSetId` [Optional]: The `NettingSetId` element specifies the identifier for a netting set. If a `NettingSetId` is specified, the trade is eligible for close-out netting under the terms of an associated ISDA agreement. The specified `NettingSetId` must be defined within the netting set definitions file (8). If left blank or omitted the trade will not belong to any netting set, and thus not be eligible for netting.

  Allowable values: Any alphanumeric string. Underscores (_) and blank spaces may be used as well.

- `AdditionalFields` [Optional]: The AdditionalFields node allows the insertion of additional trade information using custom XML elements. For example, elements such as Sector, Desk or Folder can be used. The elements within the

`AdditionalFields` node are used for informational purposes only, and do not affect pricing or exposure calculations and analytics.

Allowable values: Any custom element.

7.2 Trade Specific Data

After the envelope node, the `instruments.xml` file includes trade-specific data for each trade type supported by ORE. Each trade type has its own trade data container which is defined by an XML node containing a trade-specific configuration of individual XML tags - called elements - and trade components. The trade components are defined by XML sub-nodes that can be used within multiple trade data containers, i.e. by multiple trade types.

Details of trade-specific data for all trade types follow below.

7.2.1 Swap

The `SwapData` node is the trade data container for the Swap trade type. A Swap must have at least one leg, and can have an unlimited number of legs. Each leg is represented by a `LegData` trade component sub-node, described in section 7.3.2 Leg Data and Notionals. An example structure of a two-legged `SwapData` node is shown in Listing 49.

Listing 49: Swap data

```
<SwapData>
  <LegData>
    ...
  </LegData>
  <LegData>
    ...
  </LegData>
</SwapData>
```

7.2.2 Cap/Floor

The `CapFloorData` node is the trade data container for trade type CapFloors. It's a single floating leg that's either capped, floored or both capped and floored. The `CapFloorData` node contains a `LPILegData` sub-node where the LegType element must be set to *Floating*, plus elements for the Cap and Floor rates. An example structure with Cap rates is shown in in Listing 50. A `CapFloorData` node must have either `CapRates` or `FloorRates` elements, or both.

Listing 50: Cap/Floor data

```
<CapFloorData>
  <LegData>
    <Payer>false</Payer>
    <LegType>Floating</LegType>
     ...
  </LegData>
  <CapRates>
    <Rate>0.05</Rate>
  </CapRates>
</CapFloorData>
```

The meanings and allowable values of the elements in the `CapFloorData` node follow below.

- LegData: This is a trade component sub-node outlined in section 7.3.2 Leg Data and Notionals. Exactly one `LegData` node is allowed and the LegType element must be set to *floating*.
- CapRates: This node has child elements of type `Rate` capping the floating leg. The first rate value corresponds to the first coupon, the second rate value corresponds to the second coupon, etc. If the number of coupons exceeds the number of rate values, the rate will be kept flat at the value of last entered rate for the remaining coupons. For a fixed cap rate over all coupons, one single rate value is sufficient. The number of entered rate values cannot exceed the number of coupons.

  Allowable values for each `Rate` element: Any real number. The rate is expressed in decimal form, eg 0.05 is a rate of 0.5%
- FloorRates: This node has child elements of type `Rate` flooring the floating leg. The first rate value corresponds to the first coupon, the second rate value corresponds to the second coupon, etc. If the number of coupons exceeds the number of rate values, the rate will be kept flat at the value of last entered rate for the remaining coupons. For a fixed floor rate over all coupons, one single rate value is sufficient. The number of entered rate values cannot exceed the number of coupons.

  Allowable values for each `Rate` element: Any real number. The rate is expressed in decimal form, eg 0.05 is a rate of 0.5%

7.2.3 Swaption

The `SwaptionData` node is the trade data container for the Swaption trade type. The `SwaptionData` node has one and exactly one `OptionData` trade component sub-node, and at least one `LegData` trade component sub-node. These trade components are outlined in section 7.3.1 Option Data and section 7.3.2 Leg Data and Notionals.
Supported swaption exercise styles are European and Bermudan. European swaptions can have unlimited number of legs, with each leg represented by a `LegData` sub-node. Bermudan swaptions must have two legs, i.e. two `LegData` sub-nodes. See Table 6 for further details on requirements for Bermudan swaptions. Cross currency swaptions are not supported for either exercise style, i.e. the Currency element must have the same value for all `LegData` sub-nodes of a swaption.

The structure of an example `SwaptionData` node of a two-legged European swaption is shown in Listing 51.

Listing 51: Swaption data

```
<SwaptionData>
    <OptionData>
        <Style>European</Style>
        ...
    </OptionData>
    <LegData>
        <Currency>GBP</Currency>
        ...
    </LegData>
    <LegData>
        <Currency>GBP</Currency>
        ...
    </LegData>
</SwaptionData>
```

	A Bermudan Swaption requires:
`OptionData`	One `OptionData` sub-node
`Style`	*Bermudan*
`ExerciseDates`	At least two `ExerciseDate` child elements.
`LegData`	Two `LegData` sub-nodes
`LegType`	*Fixed* on one node and *Floating* on the other.
`Currency`	The same currency for both nodes.
`Notionals`	No accretion or amortisation, just a constant notional. Exactly one `Notional` child element for each node.
`Rates`	A constant rate. The fixed rate node should have exactly one `Rate` child element.
`Spreads`	A constant spread. The floating rate node should have exactly one `Spread` child element.

Table 6: Requirements for Bermudan Swaptions

7.2.4 FX Forward

The `FXForwardData` node is the trade data container for the FX Forward trade type. The structure - including example values - of the `FXForwardData` node is shown in Listing 52. It contains no sub-nodes.

Listing 52: FX Forward data

```
<FxForwardData>
    <ValueDate>2023-04-09</ValueDate>
    <BoughtCurrency>EUR</BoughtCurrency>
    <BoughtAmount>1000000</BoughtAmount>
    <SoldCurrency>USD</SoldCurrency>
    <SoldAmount>1500000</SoldAmount>
</FxForwardData>
```

The meanings and allowable values of the various elements in the `FXForwardData` node follow below. All elements are required.

- ValueDate: The value date of the FX Forward.
  Allowable values: See `Date` in Table 10.
- BoughtCurrency: The currency to be bought on value date.
  Allowable values: See `Currency` in Table 10.
- BoughtAmount: The amount to be sold on value date.
  Allowable values: Any positive real number.
- SoldCurrency: The currency to be sold on value date.
  Allowable values: See `Currency` in Table 10.
- SoldAmount: The amount to be sold on value date.
  Allowable values: Any positive real number.

7.2.5 FX Option

The `FXOptionData` node is the trade data container for the FX Option trade type. Vanilla FX options as well as various barrier types as specified in the SubType element are supported. FX Options can be of exercise style *European* or *American*. The strike rate of an FX option is: SoldAmount / BoughtAmount.The `FXOptionData` node includes one and only one `OptionData` trade component sub-node plus elements specific to the FX Option. The structure of an example `FXOptionData` node of a single barrier FX Option is shown in Listing 53.

Listing 53: FX Option data

```
<FxOptionData>
    <OptionData>
     ...
    </OptionData>
    <SubType>SingleBarrier</SubType>
    <BoughtCurrency>EUR</BoughtCurrency>
    <BoughtAmount>1000000</BoughtAmount>
    <SoldCurrency>USD</SoldCurrency>
    <SoldAmount>1350000</SoldAmount>
    <BarrierType>DI</BarrierType>
    <BarrierLevel>1.25</BarrierLevel>
    <BarrierRebate>0.04</BarrierRebate>
    <BarrierRebateFactor>0.1</BarrierRebateFactor>
</FxOptionData>
```

The meanings and allowable values of the elements in the `FXOptionData` node follow below.

- OptionData: This is a trade component sub-node outlined in section 7.3.1 Option Data. Note that the FX option type allows for *European* or *American* option styles, but not *Bermudan*. FX options can only have one exercise date.

- SubType: Specifies the FX option type. The allowable values are given in Table 7.

SubType	**Description**
Vanilla	A vanilla FX option with no barrier or other structured features.
SingleBarrier	Regular vanilla option payout plus a knock-in or knock-out barrier.
OneTouch	Fixed payout if barrier is breached, no payout otherwise.
NoTouch	No payout if barrier is breached, fixed payout otherwise.

Table 7: Allowable SubType Values.

- BoughtCurrency: The bought currency of the FX option.

  Allowable values: See Currency in Table 10.

- BoughtAmount: The amount in the BoughtCurrency.

  Allowable values: Any positive real number.

- SoldCurrency: The sold currency of the FX option.

  Allowable values: See Currency in Table 10.

- SoldAmount: The amount in the SoldCurrency.

  Allowable values: Any positive real number.

- BarrierType: Specifies the barrier type. It is required for `SubType` *SingleBarrier* but not used for the other subtypes. The allowable values are given in Table 8.

BarrierType	**Description**
UO	Up and Out
DO	Down and Out
UI	Up and In
DI	Down and In

Table 8: Allowable BarrierType Values.

- BarrierLevel: The barrier FX rate. Required for SubTypes *SingleBarrier, OneTouch, NoTouch.*

  Allowable values: Any positive real number.

- BarrierRebate [Optional]: When an option of type *SingleBarrier* expires without the barrier being hit, the buyer can receive a rebate as a fraction of the original premium paid. The BarrierRebate is the original premium in percentage terms, and BarrierRebateFactor is the fraction to be returned if the barrier is not hit. Only used for SubType: *SingleBarrier*. Defaults to zero if left blank or omitted.

  Allowable values: Any positive real number.

- BarrierRebateFactor [Optional]: The fraction to be returned if the barrier is not hit. Only used for SubType: *SingleBarrier*. Defaults to zero if left blank or omitted.

  Allowable values: Any positive real number.

7.3 Trade Components

Trade components are XML sub-nodes used within the trade data containers to define sets of trade data that more than one trade type can have in common, such as a leg or a schedule. A trade data container can include multiple trade components such as a swap with multiple legs, and a trade component can itself contain further trade components in a nested way.

An example of a `SwapData` trade data container, including two `LegData` trade components which in turn include further trade components such as `FixedLegData`, `ScheduleData` and `FloatingLegData` is shown in Listing 54.

Listing 54: Trade Components Example

```
<SwapData>
    <LegData>
        <Payer>true</Payer>
        <LegType>Fixed</LegType>
        <Currency>EUR</Currency>
        <PaymentConvention>Following</PaymentConvention>
        <DayCounter>30/360</DayCounter>
        <Notionals>
            <Notional>1000000</Notional>
        </Notionals>
        <ScheduleData>
        ...
        </ScheduleData>
        <FixedLegData>
            <Rates>
                <Rate>0.035</Rate>
            </Rates>
        </FixedLegData>
    </LegData>
    <LegData>
        ...
        <ScheduleData>
            ...
        </ScheduleData>
        <FloatingLegData>
            ...
        </FloatingLegData>
    </LegData>
</SwapData>
```

Descriptions of all trade components supported in ORE follow below:

7.3.1 Option Data

This trade component node is used within the `SwaptionData` and `FXOptionData` trade data containers. It contains the `ExerciseDates` sub-node which includes `ExerciseDate` child elements. An example structure of the `OptionData` trade component node is shown in Listing 55.

Listing 55: Option data

```
<OptionData>
    <LongShort>Long</LongShort>
    <OptionType>Call</OptionType>
    <Style>Bermudan</Style>
    <Settlement>Cash</Settlement>
    <PayOffAtExpiry>true</PayOffAtExpiry>
    <ExerciseDates>
        <ExerciseDate>2016-04-20</ExerciseDate>
        <ExerciseDate>2017-04-20</ExerciseDate>
    </ExerciseDates>
</OptionData>
```

The meanings and allowable values of the elements in the `OptionData` node follow below.

- LongShort: Specifies whether the option position is long or short.

  Allowable values: *Long, LONG, long, L* or *Short, SHORT, short, S*

- OptionType: Specifies whether it is a call or a put option.

  Allowable values: *Call* or *Put*

- Style: The exercise style of the option.

  Allowable values: *European* or *American* or *Bermudan*. Note that trade type Swaption can have style *European* or *Bermudan*, but not *American*. The FX Option trade type can have style *European* or *American*, but not *Bermudan*.

- Settlement: Derlivery type.

  Allowable values: *Cash* or *Physical*

- PayOffAtExpiry [Optional]: Relevant for options with early exercise, i.e. the exercise occurs before expiry; *true* indicates payoff at expiry, whereas *false* indicates payoff at exercise. Defaults to *false* if left blank or omitted.

  Allowable values: *true*, *false*

- ExerciseDates: This node contains child elements of type `ExerciseDate`. Options of style *European* or *American* require a single exercise date expressed by one single `ExerciseDate` child element. *Bermudan* style options must have two or more `ExerciseDate` child elements.

7.3.2 Leg Data and Notionals

The `LegData` trade component node is used within the `CapFloorData`, `SwapData` and `SwaptionData` trade data containers. It contains a `ScheduleData` trade component sub-node, and depending on the value of the `LegType` element, one out of the following sub-nodes: `FixedLegData`, `FloatingLegData`. The `LegData` node also includes a `Notionals` sub-node with `Notional` child elements described below. An example structure of a `LegData` node of `LegType` floating is shown in Listing 56.

Listing 56: Leg data

```
<LegData>
    <Payer>false</Payer>
    <LegType>Floating</LegType>
    <Currency>EUR</Currency>
    <PaymentConvention>Following</PaymentConvention>
    <DayCounter>30/360</DayCounter>
    <Notionals>
        <Notional>1000000</Notional>
    </Notionals>
    <ScheduleData>
    ^^I...
    </ScheduleData>
    <FloatingLegData>
    ^^I...
    </FloatingLegData>
</LegData>
```

The meanings and allowable values of the elements in the `LegData` node follow below.

- LegType: Determines which of the available sub-nodes must be used.

  Allowable values: *Fixed, Floating*

- Payer: The flows of the leg are paid to the counterparty if *true*, and received if *false*.

  Allowable values: *true, false*

- Currency: The currency of the leg.

  Allowable values: See `Currency` in Table 10.

- DayCounter: The day count convention of the leg coupons.

  Allowable values: See `DayCount Convention` in Table 12.

- PaymentConvention: The payment convention of the leg coupons.

  Allowable values: See `Roll Convention` in Table 10.

- Notionals: This node contains child elements of type `Notional`. If the notional is fixed over the life of the leg only one notional value should be entered. If the notional is amortising or accreting, this is represented by entering multiple notional values, each represented by a `Notional` child element. The first notional value corresponds to the first coupon, the second notional value corresponds to the second coupon, etc. If the number of coupons exceeds the number of notional

values, the notional will be kept flat at the value of last entered notional for the remaining coupons. The number of entered notional values cannot exceed the number of coupons.

Allowable values: Each child element can take any positive real number.

An example of a `Notionals` element for an amortising leg with four coupons is shown in Listing 57.

Listing 57: Notional list

```
<Notionals>
    <Notional>65000000</Notional>
    <Notional>65000000</Notional>
    <Notional>55000000</Notional>
    <Notional>45000000</Notional>
</Notionals>
```

Another allowable specification of the notional schedule is shown in Listing 58.

Listing 58: Notional list with dates

```
<Notionals>
    <Notional>65000000</Notional>
    <Notional start='2016-01-02'>65000000</Notional>
    <Notional start='2017-01-02'>55000000</Notional>
    <Notional start='2021-01-02'>45000000</Notional>
</Notionals>
```

The first notional must not have a start date, it will be associated with the schedule's start, The subsequent notionals can have a start date specified from which date onwards the new notional is applied. This allows specifying notionals only for dates where the notional changes.

- ScheduleData: This is a trade component sub-node outlined in section 7.3.3 Schedule Data and Dates.
- FixedLegData: This trade component sub-node is required if `LegType` is set to *fixed* It is outlined in section 7.3.4 Fixed Leg Data and Rates.
- FloatingLegData: This trade component sub-node is required if `LegType` is set to *floating* It is outlined in section 7.3.5 Floating Leg Data and Spreads.

7.3.3 Schedule Data and Dates

The `ScheduleData` trade component node is used within the `LegData` trade component. When `IsRulesBased` is set to *false*, the `ScheduleData` node includes a `Dates` sub-node where the schedule is determined directly by `Date` child elements. The schedule can also be generated from a set of rules based on the entries of the StartDate, EndDate, Tenor, Calendar, Convention, TermConvention, and Rule elements. Example structures of `ScheduleData` nodes based on rules respectively dates are shown in Listing 59 and Listing 60, respectively.

Listing 59: Schedule data, rules based

```
<ScheduleData>
      <Rules>
          <StartDate>2013-02-01</StartDate>
          <EndDate>2030-02-01</EndDate>
          <Tenor>1Y</Tenor>
          <Calendar>UK</Calendar>
          <Convention>MF</Convention>
          <TermConvention>MF</TermConvention>
          <Rule>Forward</Rule>
      </Rules>
</ScheduleData>
```

Listing 60: Schedule data, date based

```
<ScheduleData>
      <Dates>
          <Date>2012-01-06</Date>
          <Date>2012-04-10</Date>
          <Date>2012-07-06</Date>
          <Date>2012-10-08</Date>
          <Date>2013-01-07</Date>
          <Date>2013-04-08</Date>
      </Dates>
 </ScheduleData>
```

The ScheduleData section can contain any number and combination of `<Dates>` and `<Rules>` sections. The resulting schedule will then be an ordered concatenation of individual schedules.

The meanings and allowable values of the elements in the `ScheduleData` node follow below.

- StartDate: The schedule start date.

  Allowable values: See `Date` in Table 10.

- EndDate: The schedule end date.

  Allowable values: See `Date` in Table 10.

- Tenor: The tenor used to generate schedule dates.

  Allowable values: A string where the last character must be D or W or M or Y. The characters before that must be a positive integer.
  D = Day, W = Week, M = Month, Y = Year

- Calendar: The calendar used to generate schedule dates.

  Allowable values: See Table 11 Calendar.

- Convention: Determines the adjustment of the schedule dates with regards to the selected calendar.

  Allowable values: See `Roll Convention` in Table 10.

- TermConvention: Determines the adjustment of the final schedule date with regards to the selected calendar.

  Allowable values: See `Roll Convention` in Table 10.

- Rules: Rules for the generation of the schedule using given start and end dates, tenor, calendar and business day conventions.

  Allowable values and descriptions: See Table 9 Rule.

- Dates: This is a sub-node and contains child elements of type `Date`. In this case the schedule dates are determined directly by the `Date` child elements. At least two `Date` child elements must be provided.

  Allowable values: Each `Date` child element can take the allowable values listed in `Date` in Table 10.

`Rule`	
Allowable Values	**Effect**
Backward	Backward from termination date to effective date.
Forward	Forward from effective date to termination date.
Zero	No intermediate dates between effective date and termination date.
ThirdWednesday	All dates but effective date and termination date are taken to be on the third Wednesday of their month (with forward calculation.)
Twentieth	All dates but the effective date are taken to be the twentieth of their month (used for CDS schedules in emerging markets.) The termination date is also modified.
TwentiethIMM	All dates but the effective date are taken to be the twentieth of an IMM month (used for CDS schedules.) The termination date is also modified.
OldCDS	Same as TwentiethIMM with unrestricted date ends and log/short stub coupon period (old CDS convention).
CDS	Credit derivatives standard rule since 'Big Bang' changes in 2009.

Table 9: Allowable Values for Rule

7.3.4 Fixed Leg Data and Rates

The `FixedLegData` trade component node is used within the `LegData` trade component when the `LegType` element is set to *Fixed*. The `FixedLegData` node only includes the

Rates sub-node which contains the rates of the fixed leg as child elements of type Rate. An example of a FixedLegData node for a fixed leg with constant notional is shown in Listing 61.

Listing 61: Fixed leg data

```
<FixedLegData>
      <Rates>
          <Rate>0.05</Rate>
      </Rates>
</FixedLegData>
```

The meanings and allowable values of the elements in the FixedLegData node follow below.

- Rates: This node contains child elements of type Rate. If the rate is constant over the life of the fixed leg only one rate value should be entered. If two or more coupons have different rates, multiple rate values are required, each represented by a Rate child element. The first rate value corresponds to the first coupon, the second rate value corresponds to the second coupon, etc. If the number of coupons exceeds the number of rate values, the rate will be kept flat at the value of last entered rate for the remaining coupons. The number of entered rate values cannot exceed the number of coupons.

  Allowable values: Each child element can take any real number. The rate is expressed in decimal form, e.g. 0.05 is a rate of 5%.

  As in the case of notionals, the rate schedule can be specified with dates as shown in Listing 62.

Listing 62: Fixed leg data with 'dated' rates

```
<FixedLegData>
      <Rates>
          <Rate>0.05</Rate>
          <Rate start='2016-02-04'>0.05</Rate>
          <Rate start='2019-02-05'>0.05</Rate>
      </Rates>
</FixedLegData>
```

7.3.5 Floating Leg Data and Spreads

The FloatingLegData trade component node is used within the LegData trade component when the LegType element is set to *Floating*. It is also used directly within the CapFloor trade data container. The FloatingLegData node includes elements specific to a floating leg as well as the Spreads sub-node which contains the spreads of the floating leg as child elements of type Spread.
An example of a FloatingLegData node is shown in Listing 63.

Listing 63: Floating leg data

```
<FloatingLegData>
    <Index>USD-LIBOR-3M</Index>
    <IsInArrears>false</IsInArrears>
    <FixingDays>2</FixingDays>
    <Spreads>
        <Spread>0.005</Spread>
    </Spreads>
</FloatingLegData>
```

The meanings and allowable values of the elements in the `FloatingLegData` node follow below.

- Index: The combination of currency, index and term that identifies the relevant fixings and yield curve of the floating leg.

  Allowable values: An alphanumeric string on the form CCY-INDEX-TERM, matching available Ibor indices in the `simulation.xml` file. CCY, INDEX and TERM must be separated by dashes (-). TERM must be an integer followed by D, W, M or Y. See Table 13.

- IsInArrears: *true* indicates that fixing is in arrears, i.e. the fixing gap is calculated in relation to the current period end date.
  false indicates that fixing is in advance, i.e. the fixing gap is calculated in relation to the previous period end date.

  Allowable values: *true, false*

- FixingDays: This is the fixing gap, i.e. the number of days before the period end date an index fixing is taken.

  Allowable values: Positive integers.

- Spreads: This node contains child elements of type `Spread`. If the spread is constant over the life of the floating leg only one spread value should be entered. If two or more coupons have different spreads, multiple spread values are required, each represented by a `Spread` child element. The first spread value corresponds to the first coupon, the second spread value corresponds to the second coupon, etc. If the number of coupons exceeds the number of spread values, the spread will be kept flat at the value of last entered spread for the remaining coupons. The number of entered spread values cannot exceed the number of coupons.

  Allowable values: Each child element can take any real number. The spread is expressed in decimal form, e.g. 0.005 is a spread of 0.5% or 50 bp.

  For the `<Spreads>` section, the same applies as for notionals and rates - a list of changing spreads can be specified without or with individual starte dates as shown in Listing 64.

Listing 64: 'Dated' spreads

```
<Spreads>
    <Spread>0.005</Spread>
    <Spread start='2017-03-05'>0.007</Spread>
    <Spread start='2019-03-05'>0.009</Spread>
</Spreads>
```

7.4 Allowable Values for Standard Trade Data

Trade Data	Allowable Values
`Date`	The following date formats are supported: *yyyymmdd* *yyyy-mm-dd* *yyyy/mm/dd* *yyyy.mm.dd* *dd-mm-yy* *dd/mm/yy* *dd.mm.yy* *dd-mm-yyyy* *dd/mm/yyyy* *dd.mm.yyyy* and Dates as serial numbers, comparable to Microsoft Excel dates, with a minimum of 367 for Jan 1, 1901, and a maximum of 109574 for Dec 31, 2199.
`Currency`	*ATS, AUD, BEF, BRL, CAD, CHF, CNY, CZK, DEM, DKK, EUR, ESP, FIM, FRF, GBP, GRD, HKD, HUF, IEP, ITL, INR, ISK, JPY, KRW, LUF, NLG, NOK, NZD, PLN, PTE, RON, SEK, SGD, THB, TRY, TWD, USD, ZAR, ARS, CLP, COP, IDR, ILS, KWD, PEN, MXN, SAR, RUB, TND, MYR, UAH, KZT, QAR, MXV, CLF, EGP, BHD, OMR, VND, AED, PHP, NGN, MAD*, Note: Currency codes must also match available currencies in the `simulation.xml` file.
`Roll Convention`	*F, Following, FOLLOWING* *MF, ModifiedFollowing, Modified Following, MODIFIEDF* *P, Preceding, PRECEDING* *MP, ModifiedPreceding, Modified Preceding, MODIFIEDP* *U, Unadjusted, INDIFF*

Table 10: Allowable values for standard trade data.

`Calendar`	
Allowable Values	**Resulting Calendar**
TARGET, TGT, EUR	Target Calendar
CA,TRB, CAD	Canada Calendar
TKB, JP, JPY	Japan Calendar
ZUB, CHF	Switzerland Calendar
GB, LNB, UK	UK Calendar
US, NYB, USD	US Calendar
US-SET	US Settlement Calendar
US-GOV	US Government Bond Calendar
US-NYSE	US NYSE Calendar
US-NERC	US NERC Calendar
AU, AUD	Australia Calendar
SA, ZAR	South Africa Calendar
SS, SEK	Sweden Calendar
ARS	Argentina Calendar
BRL	Brazil Calendar
CNY	China Calendar
CZK	Czech Republic Calendar
DEN, DKK	Denmark Calendar
FIN	Finland Calendar
HKD	HongKong Calendar
ISK	Iceland Calendar
INR	India Calendar
IDR	Indonesia Calendar
MXN	Mexico Calendar
NZD	New Zealand Calendar
NOK	Norway Calendar
PLN	Poland Calendar
RUB	Russia Calendar
SAR	Saudi Arabia
SGD	Singapore Calendar
KRW	South Korea Calendar
TWD	Taiwan Calendar
TRY	Turkey Calendar
UAH	Ukraine Calendar
WeekendsOnly	Weekends Only Calendar

Table 11: Allowable Values for Calendar. Combinations of up to four calendars can be provided using comma separated calendar names.

DayCount Convention	
Allowable Values	**Resulting DayCount Convention**
A360, Actual/360, ACT/360	Actual 360
A365, A365F, Actual/365, Actual/365 (fixed)	Actual 365 Fixed
T360, 30/360, 30/360 (Bond Basis), ACT/nACT	Thirty 360 (US)
30E/360, 30E/360 (Eurobond Basis)	Thirty 360 (European)
30/360 (Italian)	Thirty 360 (Italian)
ActActISDA, ActualActual (ISDA), ACT/ACT, ACT	Actual Actual (ISDA)
ActActISMA, ActualActual (ISMA)	Actual Actual (ISMA)
ActActAFB, Actual/Actual (AFB)	Actual Actual (AFB)

Table 12: Allowable Values for DayCount Convention

`Index`	
On form CCY-INDEX-TENOR, and matching available indices in the `simulation.xml` file.	
Index Component	**Allowable Values**
CCY-INDEX	*EUR-EONIA* *EUR-EURIBOR* *EUR-LIBOR* *USD-FedFunds* *USD-LIBOR* *GBP-SONIA* *GBP-LIBOR* *JPY-LIBOR* *JPY-TIBOR* *CHF-LIBOR* *AUD-LIBOR* *AUD-BBSW* *CAD-CDOR* *CAD-BA* *SEK-STIBOR* *SEK-LIBOR* *DKK-LIBOR* *DKK-CIBOR* *SGD-SIBOR* *SGD-SOR* *HKD-HIBOR* *NOK-NIBOR* *HUF-BUBOR* *IDR-IDRFIX* *INR-MIFOR* *MXN-TIIE* *PLN-WIBOR* *SKK-BRIBOR* *NZD-BKBM*
TENOR	An integer followed by *D, W, M or Y*

Table 13: Allowable values for Index.

8 Netting Set Definitions

The netting set definitions file - `NettingSetDefinitions.xml` - contains a list of definitions for various ISDA netting agreements. The file is written in XML format.

Each netting set is defined within its own `NettingSet` node. All of these `NettingSet` nodes are contained as children of a `NettingSetDefinitions` node.

There are two distinct cases to consider:

- An ISDA agreement which does not contain a *Credit Support Annex* (CSA)
- An ISDA agreement which does contain a CSA

8.1 Uncollateralised Netting Set

If an ISDA agreement does not contain a Credit Support Annex the portfolio exposures are not eligible for collateralisation. In such a case the netting set can be defined within the following XML template:

Listing 65: Uncollateralised netting set definition

```
<NettingSet>
    <NettingSetId> </NettingSetId>
    <Counterparty> </Counterparty>
    <ActiveCSAFlag> </ActiveCSAFlag>
    <CSADetails></CSADetails>
</NettingSet>
```

The meanings of the various elements are as follows:

- NettingSetId: The unique identifier for the ISDA netting set.
- Counterparty: The identifier for the counterparty to the ISDA agreement.
- ActiveCSAFlag: Boolean indicating whether the netting set is covered by a Credit Support Annex. For uncollateralised netting sets this flag should be *False*.
- CSADetails: Node containing as children details of the governing Credit Support Annex. For uncollateralised netting sets there is no need to store any information within this node.

8.2 Collateralised Netting Set

If an ISDA agreement contains a Credit Support Annex the portfolio exposures are eligible for collateralisation. In such a case the netting set can be defined within the following XML template:

Listing 66: Collateralised netting set definition

```
<NettingSet>
    <NettingSetId> </NettingSetId>
    <Counterparty> </Counterparty>
    <ActiveCSAFlag> </ActiveCSAFlag>
    <CSADetails>
        <Bilateral> </Bilateral>
        <CSACurrency> </CSACurrency>
        <Index> </Index>
        <ThresholdPay> </ThresholdPay>
        <ThresholdReceive> </ThresholdReceive>
        <MinimumTransferAmountPay> </MinimumTransferAmountPay>
        <MinimumTransferAmountReceive> </MinimumTransferAmountReceive>
        <IndependentAmount>
            <IndependentAmountHeld> </IndependentAmountHeld>
            <IndependentAmountType> </IndependentAmountType>
        </IndependentAmount>
        <MarginingFrequency>
            <CallFrequency> </CallFrequency>
            <PostFrequency> </PostFrequency>
        </MarginingFrequency>
        <MarginPeriodOfRisk> </MarginPeriodOfRisk>
        <CollateralCompoundingSpreadReceive>
        </CollateralCompoundingSpreadReceive>
        <CollateralCompoundingSpreadPay> </CollateralCompoundingSpreadPay>
        <EligibleCollaterals>
            <Currencies>
                <Currency>USD</Currency>
                <Currency>EUR</Currency>
                <Currency>CHF</Currency>
                <Currency>GBP</Currency>
                <Currency>JPY</Currency>
                <Currency>AUD</Currency>
            </Currencies>
        </EligibleCollaterals>
    </CSADetails>
</NettingSet>
```

The first few nodes are shared with the template for uncollateralised netting sets:

- NettingSetId: The unique identifier for the ISDA netting set.
- Counterparty: The identifier for the counterparty to the ISDA agreement.
- ActiveCSAFlag: Boolean indicating whether the netting set is covered by a Credit Support Annex. For collateralised netting sets this flag should be *True*.
- CSADetails: Node containing as children details of the governing Credit Support Annex.

CSADetails

The `CSADetails` node contains details of the Credit Support Annex which are relevant for the purposes of exposure calculation. The meanings of the various elements are as follows:

Bilateral There are three possible values here:

- Bilateral: Both parties to the CSA are legally entitled to request collateral to cover their counterparty credit risk exposure on the underlying portfolio
- CallOnly: Only we are entitled to hold collateral; the counterparty has no such entitlement
- PostOnly: Only the counterparty is entitled to hold collateral; we have no such entitlement

CSACurrency A three-letter ISO code specifying the master currency of the CSA. All monetary values specified within the CSA are assumed to be denominated in this currency.

Index The index is used to derive the fixing which is used for compounding cash collateral in the master currency of the CSA.

ThresholdPay A threshold amount above which the counterparty is entitled to request collateral to cover excess exposure.

ThresholdReceive A threshold amount above which we are entitled to request collateral from the counterparty to cover excess exposure.

MinimumTransferAmountPay Any margin calls issued by the counterparty must exceed this minimum transfer amount. If the collateral shortfall is less than this amount, the counterparty is not entitled to request margin.

MinimumTransferAmountReceive Any margin calls issued by us to the counterparty must exceed this minimum transfer amount. If the collateral shortfall is less than this amount, we are not entitled to request margin.

IndependentAmount This element contains two child nodes:

- IndependentAmountHeld: The netted sum of all independent amounts covered by this ISDA agreement/CSA. A negative number implies that the counterparty holds the independent amount.
- IndependentAmountType: The nature of the independent amount as defined within the Credit Support Annex. The only supported value here is *FIXED*.

This covers only the case where only one party has to post an independent amount. In a future release this will be extended to the situation prescribed by the Basel/IOSCO regulation (initial margin to be posted by both parties without netting).

MarginingFrequency This element contains two child nodes:

- CallFrequency: The frequency with which we are entitled to request additional margin from the counterparty (e.g. *1D*, *2W*, *1M*).
- PostFrequency: The frequency with which the counterparty is entitled to request additional margin from us (e.g. *1D*, *2W*, *1M*).

MarginPeriodOfRisk The length of time assumed necessary for closing out the portfolio position after a default event (e.g. *1D*, *2W*, *1M*).

CollateralCompoundingSpreadReceive The spread over the O/N interest accrual rate taken by the clearing house, when holding collateral.

CollateralCompoundingSpreadPay The spread over the O/N interest accrual rate taken by the clearing house, when collateral is held by the counterparty.

EligibleCollaterals For now the only supported type of collateral is cash. If the CSA specifies a set of currencies which are eligible as collateral, these can be listed using `Currency` nodes.

9 Market Data

Market data in the `market.txt` file is given in three columns; Date, Quote and Quote value.

- **Date**: The as of date of the market quote value.

  Allowable values: See `Date` in Table 10.

- **Quote**: A generic description that contains Instrument Type and Quote Type, followed by instrument specific descriptions (See §9.1 ff.). The base of a quote consists of InstType/QuoteType followed by instrument specific information separated by slashes "/".

  Allowable values for Instrument Types and Quote Types are given in Table 14.

- **Quote Value**: The market quote value in decimal form for the given quote on the given as of date. Quote values are assumed to be mid-market.

  Allowable values: Any real number.

Market Data Parameter	**Allowable Values**
Instrument Type	*ZERO, DISCOUNT, MM, MM_FUTURE, FRA, IR_SWAP, BASIS_SWAP, CC_BASIS_SWAP, CDS, FX_SPOT, FX_FWD, SWAPTION, CAPFLOOR, FX_OPTION, HAZARD_RATE, RECOVERY_RATE*
Quote Type	*BASIS_SPREAD, CREDIT_SPREAD, YIELD_SPREAD, RATE, RATIO, PRICE, RATE_LNVOL, RATE_NVOL, RATE_SLNVOL, SHIFT*

Table 14: Allowable values for Instrument and Quote type market data.

An excerpt from a typical `market.txt` file is shown in Listing 67.

Listing 67: Excerpt of a market data file

```
2011-01-31 MM/RATE/EUR/0D/1D 0.013750
2011-01-31 MM/RATE/EUR/1D/1D 0.010500
2011-01-31 MM/RATE/EUR/2D/1D 0.010500
2011-01-31 MM/RATE/EUR/2D/1W 0.009500
2011-01-31 MM/RATE/EUR/2D/1M 0.008700
2011-01-31 MM/RATE/EUR/2D/2M 0.009100
2011-01-31 MM/RATE/EUR/2D/3M 0.010200
2011-01-31 MM/RATE/EUR/2D/4M 0.011000

2011-01-31 FRA/RATE/EUR/3M/3M 0.013080
2011-01-31 FRA/RATE/EUR/4M/3M 0.013890
2011-01-31 FRA/RATE/EUR/5M/3M 0.014630
2011-01-31 FRA/RATE/EUR/6M/3M 0.015230

2011-01-31 IR_SWAP/RATE/EUR/2D/3M/1Y 0.014400
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/1Y3M 0.015400
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/1Y6M 0.016500
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/2Y 0.018675
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/3Y 0.022030
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/4Y 0.024670
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/5Y 0.026870
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/6Y 0.028700
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/7Y 0.030125
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/8Y 0.031340
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/9Y 0.032450
```

9.1 Zero Rate

The instrument specific information to be captured for quotes representing Zero Rates is shown in Table 15.

Property	**Allowable values**	**Description**
Currency	See `Currency` in Table 10	Currency of the Zero rate
CurveId	A CCY concatenated with a Tenor. Should match CurveIds in the `yield-curves.xml` file	Unique identifier for the yield curve associated with the zero quote
DayCounter	See `DayCount Convention` in Table 12	The day count basis associated with the zero quote
Tenor or ZeroDate	Tenor: An integer followed by D, W, M or Y, ZeroDate: See `Date` in Table 10	Either a Tenor for tenor based zero quotes, or an explicit maturity date (ZeroDate)

Table 15: Zero Rate

Examples with a Tenor and with a ZeroDate:

- ZERO/RATE/USD/USD6M/A365F/6M
- ZERO/RATE/USD/USD6M/A365F/12-05-2018

9.2 Discount Factor

The instrument specific information to be captured for quotes representing Discount Factors is shown in Table 16.

Property	Allowable values	Description
Currency	See `Currency` in Table 10	Currency of the Discount rate
CurveId	A CCY concatenated with a Tenor. Should match CurveIds in the `yield-curves.xml` file	Unique identifier for the yield curve associated with the discount quote
Term or DiscountDate	Term: An integer followed by D, W, M or Y, DiscountDate: See `Date` in Table 10	Either a Term is used to determine the maturity date, or an explicit maturity date (Discount Date) is given.

Table 16: Discount Rate

If a Term is given in the last element of the quote, it is converted to a maturity date using a weekend only calendar.

Examples with a Term and with a DiscountDate:

- DISCOUNT/RATE/EUR/EUR3M/3Y
- DISCOUNT/RATE/EUR/EUR3M/A365F/12-05-2018

9.3 FX Spot Rate

Property	Allowable values	Description
Unit currency	See `Currency` in Table 10	Unit/Source currency
Target currency	See `Currency` in Table 10	Target currency

Table 17: FX Spot Rate

Example:

- FX/RATE/EUR/USD

9.4 Deposit Rate

Property	Allowable values	Description
Currency	See `Currency` in Table 10	Currency of the Deposit rate
Forward start	An integer followed by D, W, M or Y.	Period from today to start
Term	An integer followed by D, W, M or Y.	Period from start to maturity

Table 18: Deposit Rate

Deposits are usually quoted as ON (Overnight), TN (Tomorrow Next), SN (Spot Next), SW (Spot Week), 3W (3 Weeks), 6M (6 Months), etc.
Forward start for ON is today (i.e. forward start = 0D), for TN tomorrow (forward start = 1D), for SN two days from today (forward start = 2D). For longer term Deposits, forward start is derived from conventions, see §6.6, and is between 0D and 2D, i.e. "spot days" are between 0 and 2.

Example:

- MM/RATE/EUR/2D/3M

9.5 FRA Rate

Property	Allowable values	Description
Currency	See `Currency` in Table 10	Currency of the FRA rate
Forward start	An integer followed by D, W, M or Y	Period from today to start
Term	An integer followed by D, W, M or Y	Period from start to maturity

Table 19: FRA Rate

FRAs are typically quoted as e.g. 6x9 which means forward start 6M from today, maturity 9M from today, with appropriate adjustment of dates.

Example:

- FRA/RATE/EUR/9M/3M

9.6 Money Market Futures Price

Property	Allowable values	Description
Currency	See `Currency` in Table 10	Currency of the MM Future price
Expiry	Alphanumeric string on the form MMMYY	Expiry month
Contract	String	Contract name

Table 20: Money Market Futures Price

Expiry month is typically quoted as JUN08, SEP09, DEC10, etc. The exact expiry date follows from a date rule such as 3rd Wednesday of the specified month, adjusted to the following business day. The date rule is not quoted directly, but defined in the futures contract.

Example:

- MM_FUTURE/PRICE/EUR/JUN18/LIF3ME

9.7 Swap Rate

Property	Allowable values	Description
Currency	See `Currency` in Table 10	Currency of the Swap rate
Forward start	An integer followed by D, W, M or Y	Generic period from today to start
Tenor	An integer followed by D, W, M or Y	Underlying index period
Term	An integer followed by D, W, M or Y	Swap length from start to maturity

Table 21: Swap Rate

Forward start is usually not quoted, but needs to be derived from conventions.

Example:

- IR_SWAP/RATE/EUR/2D/6M/10Y

9.8 Basis Swap Spread

Property	Allowable values	Description
Flat tenor	An integer followed by D, W, M or Y	Zero spread leg's index tenor
Tenor	An integer followed by D, W, M or Y	Non-zero spread leg's index tenor
Term	An integer followed by D, W, M or Y	Swap length from start to maturity

Table 22: Basis Swap Spread

Example:

- BASIS_SWAP/BASIS_SPREAD/6M/3M/10Y

9.9 Cross Currency Basis Swap Spread

Property	Allowable values	Description
Flat currency	See `Currency` in Table 10	Currency for zero spread leg
Flat tenor	An integer followed by D, W, M or Y	Zero spread leg's index tenor
Currency	See `Currency` in Table 10	Currency for non-zero spread leg
Tenor	An integer followed by D, W, M or Y	Non-zero spread leg's index tenor
Term	An integer followed by D, W, M or Y	Swap length from start to maturity

Table 23: Cross Currency Basis Swap Spread

Example:

- CC_BASIS_SWAP/BASIS_SPREAD/USD/3M/JPY/6M/10Y

9.10 CDS Spread

Property	Allowable values	Description
Issuer	String	Issuer name
Seniority	String	Seniority status
Currency	See `Currency` in Table 10	CDS Spread currency
Term	An integer followed by D, W, M or Y	Generic period from start to maturity

Table 24: CDS Spread

Example:

- CDS/CREDIT_SPREAD/GE/SeniorUnsec/EUR/5Y

9.11 CDS Recovery Rate

Property	Allowable values	Description
Issuer	String	Issuer name
Seniority	String	Seniority status
Currency	See `Currency` in Table 10	CDS Spread currency

Table 25: CDS Recovery Rate

Example:

- CDS/RECOVERY_RATE/GE/SeniorUnsec/EUR

9.12 Probability of Default

This quote can represent a cumulative PD for the given rating and maturity, or a Loss Given Default for the given rating.

Property	Allowable values	Description
Rating	String	Internal or external rating code
Term	An integer followed by D, W, M or Y	Generic period from start to maturity

Table 26: Probability of Default

Example:

- NAME/PD/15/3Y
- NAME/LGD/Baa3

9.13 FX Option Implied Volatility

Property	Allowable values	Description
Unit currency	See `Currency` in Table 10	Unit/Source currency
Target currency	See `Currency` in Table 10	Target currency
Expiry	An integer followed by D, W, M or Y	Period from today to expiry
Strike	*ATM, RR, BF*	ATM (Straddle), RR (Risk Reversal), BF (Butterfly)

Table 27: FX Option Implied Volatility

Volatilities are quoted in terms of strategies - at-the-money straddle, risk reversal and butterfly.

Example:

- FX_OPTION/RATE_LNVOL/EUR/USD/3M/ATM

9.14 Cap/Floor Implied Volatility

Property	Allowable values	Description
Currency	See `Currency` in Table 10	Currency of the Cap/Floor volatility
Term	An integer followed by D, W, M or Y	Period from start to expiry
IndexTenor	An integer followed by D, W, M or Y	Underlying index tenor
Atm	*1, 0*	ATM volatility quote if true (1), otherwise (0) smile quote
Relative	*1, 0*	Relative quote (to be added to atm vol) if true (1), otherwise (0) absolute quote
Strike	Real number	Strike rate

Table 28: FX Option Implied Volatility

Examples:

- CAPFLOOR/RATE_LNVOL/EUR/10Y/6M/0/0/0.0350 (smile, absolute, strike 3.5%)
- CAPFLOOR/RATE_LNVOL/EUR/10Y/6M/1/0/0.0000 (atm, absolute, irrelevant strike)

9.15 Swaption Implied Volatility

Property	Allowable values	Description
Currency	See `Currency` in Table 10	Currency of the Swaption volatility
Term	An integer followed by D, W, M or Y	Period from start to expiry
Dimension	*Smile, ATM*	Whether volatility quote is a Smile or ATM
Strike	Real number	Strike rate - (not required for ATM), as deviation from the ATM strike

Table 29: Swaption Implied Volatility

Note: The volatility quote is expected to be an absolute volatility, and not the deviation from the at-the-money volatility (the latter is e.g. the quotation convention used by BGC partners).

Examples:

- SWAPTION/RATE_LNVOL/EUR/5Y/10Y/ATM (absolute ATM vol quote)
- SWAPTION/RATE_LNVOL/EUR/5Y/10Y/SMILE/0.0050 (absolute vol quote for ATM strike plus 50bp)

10 Fixing History

Historical fixings data in the `fixings.txt` file is given in three columns; Index Name, Fixing Date and Index value. Columns are separated by semicolons ";" or blanks. Fixings are used in cases where the current coupon of a trade has been fixed in the past, or other path dependent features.

- Fixing Date: The date of the fixing.

  Allowable values: See `Date` in Table 10.

- Index Name: The name of the Index.

  Allowable values are given in Table 30.

- Index Value: The index value for the given fixing date.

  Allowable values: Any real number (not expressed as a percentage or basis points).

An excerpt of a fixings file is shown in Listing 68. Note that alternative index name formats are used (Table 30).

Listing 68: Excerpt of a fixings file

```
20150202 EUR-EONIA -0.00024
20150202 EUR-EURIBOR-1M 0.00003
20150202 EUR-EURIBOR-1W -0.00022
20150202 EUR-EURIBOR-2W -0.00017
20150202 EUR-EURIBOR-3M 0.00055
20150202 EUR-EURIBOR-3M 0.00055
20150202 EUR-EURIBOR-6M 0.00134
20150202 EUR-EURIBOR-6M 0.00271
20150202 GBP-LIBOR-12M 0.009565
20150202 GBP-LIBOR-1M 0.0050381
20150202 GBP-LIBOR-1W 0.0047938
20150202 GBP-LIBOR-3M 0.0056338
20150202 GBP-LIBOR-6M 0.006825
20150202 JPY-LIBOR-12M 0.0026471
20150202 JPY-LIBOR-1M 0.0007143
20150202 JPY-LIBOR-1W 0.0004357
20150202 JPY-LIBOR-3M 0.0010429
20150202 JPY-LIBOR-6M 0.0014357
20150202 USD-LIBOR-12M 0.006194
20150202 USD-LIBOR-1M 0.001695
20150202 USD-LIBOR-1W 0.00136
```

IR Index on form CCY-INDEX-TENOR:	
Index Component	**Allowable Values**
CCY-INDEX	*EUR-EONIA* *EUR-EURIBOR* *EUR-LIBOR* *USD-FedFunds* *USD-LIBOR* *GBP-SONIA* *GBP-LIBOR* *JPY-TONAR* *JPY-LIBOR* *CHF-LIBOR* *AUD-LIBOR* *AUD-BBSW* *CAD-CDOR* *CAD-BA* *SEK-STIBOR* *SEK-LIBOR* *DKK-LIBOR* *SGD-SIBOR* *HKD-HIBOR*
TENOR	An integer followed by *D, W, M or Y*

Table 30: Allowable values for IR indices.

A Methodology Summary

A.1 Risk Factor Evolution Model

ORE applies the cross asset model described in detail in [17] to evolve the market through time. So far the evolution model in ORE is limited to IR and FX risk factors for any number of currencies, extensions to further risk factor classes (Inflation, Credit, Equity, Commodity) will follow.
The Cross Asset Model is based on the Linear Gauss Markov model (LGM) for interest rates and lognormal FX processes. We identify a single *domestic* currency; its LGM process, which is labelled z_0; and a set of n foreign currencies with associated LGM processes that are labelled z_i, $i = 1, \ldots, n$. If we consider n foreign exchange rates for converting foreign currency amounts into the single domestic currency by multiplication, x_i, $i = 1, \ldots, n$, then the cross asset model is given by the system of SDEs

$$\begin{aligned}
dz_0 &= \alpha_0 \, dW_0^z \\
dz_i &= \gamma_i \, dt + \alpha_i \, dW_i^z, \qquad i > 0 \\
\frac{dx_i}{x_i} &= \mu_i \, dt + \sigma_i \, dW_i^x, \qquad i > 0
\end{aligned}$$

$$\begin{aligned}
\gamma_i &= -\alpha_i^2 \, H_i - \rho_{ii}^{zx} \, \sigma_i \, \alpha_i + \rho_{i0}^{zz} \, \alpha_i \, \alpha_0 \, H_0 \\
\mu_i &= r_0 - r_i + \rho_{0i}^{zx} \, \alpha_0 \, H_0 \, \sigma_i \\
r_i &= f_i(0,t) + z_i(t) \, H_i'(t) + \zeta_i(t) \, H_i(t) \, H_i'(t), \quad \zeta_i(t) = \int_0^t \alpha_i^2(s) \, ds
\end{aligned}$$

$$dW_i^a \, dW_j^b = \rho_{ij}^{ab} \, dt, \qquad a, b \in \{z, x\}$$

where we have dropped time dependencies for readability, and $f_i(0,t)$ is the instantaneous forward curve in currency i.

Parameters $H_i(t)$ and $\alpha_i(t)$ (or alternatively $\zeta_i(t)$) are LGM model parameters which determine, together with the stochastic factor $z_i(t)$, the evolution of numeraire and zero bond prices in the LGM model:

$$N(t) = \frac{1}{P(0,t)} \exp\left\{ H_t \, z_t + \frac{1}{2} H_t^2 \, \zeta_t \right\} \tag{1}$$

$$P(t,T,z_t) = \frac{P(0,T)}{P(0,t)} \exp\left\{ -(H_T - H_t) \, z_t - \frac{1}{2} \left(H_T^2 - H_t^2 \right) \zeta_t \right\}. \tag{2}$$

Note that the LGM model is closely related to the Hull-White model in T-forward measure [17].

A.2 Exposures

In ORE we use the following exposure definitions

$$EE(t) = EPE(t) = \mathbb{E}^N \left[\frac{(NPV(t) - C(t))^+}{N(t)} \right] \tag{3}$$

$$ENE(t) = \mathbb{E}^N \left[\frac{(-NPV(t) + C(t))^+}{N(t)} \right] \tag{4}$$

where $NPV(t)$ stands for the netting set NPV and C is the posted collateral. Note that these exposures are expectations of values discounted with numeraire N (in ORE the Linear Gauss Markov model's numeraire) to today, and expectations are taken in the measure associated with numeraire N. These are the exposures which enter into unilateral CVA and DVA calculation, respectively, see next section. Note that we sometimes label the expected exposure (3) EPE, not to be confused with the Basel III Expected Positive Exposure below.

Basel III defines a number of exposures each of which is a 'derivative' of Basel's Expected Exposure:

Expected Exposure

$$EE_B(t) = \mathbb{E}[\max(NPV(t) - C(t), 0)] \tag{5}$$

Expected Positive Exposure

$$EPE_B(T) = \frac{1}{T} \sum_{t<T} EE_B(t) \cdot \Delta t \tag{6}$$

Effective Expected Exposure, recursively defined as running maximum

$$EEE_B(t) = \max(EEE_B(t - \Delta t), EE_B(t)) \tag{7}$$

Effective Expected Positive Exposure

$$EEPE_B(T) = \frac{1}{T} \sum_{t<T} EEE_B(t) \cdot \Delta t \tag{8}$$

The last definition, Effective EPE, is used in Basel documents since Basel II for Exposure At Default and capital calculation. Following [11, 12] the time averages in the EPE and EEPE calculations are taken over *the first year* of the exposure evolution (or until maturity if all positions of the netting set mature before one year).

To compute $EE_B(t)$ consistently in a risk-neutral setting, we compound (3) with the deterministic discount factor $P(t)$ up to horizon t:

$$EE_B(t) = \frac{1}{P(t)} EE(t)$$

Finally, we define another common exposure measure, the *Potential Future Exposure* (PFE), as a (typically high) quantile α of the NPV distribution through time, similar to Value at Risk but at the upper end of the NPV distribution:

$$PFE_\alpha(t) = (\inf \{x | F_t(x) \geq \alpha\})^+ \tag{9}$$

where F_t is the cumulative NPV distribution function at time t. Note that we also take the positive part to ensure that PFE is a positive measure even if the quantile yields a negative value which is possible in extreme cases.

A.3 CVA and DVA

Using the expected exposures in A.2 unilateral discretised CVA and DVA are given by [17]

$$CVA = \sum_i PD(t_{i-1}, t_i) \times LGD \times EPE(t_i) \tag{10}$$

$$DVA = \sum_i PD_{Bank}(t_{i-1}, t_i) \times LGD_{Bank} \times ENE(t_i) \tag{11}$$

where

$EPE(t)$	expected exposure (3)
$ENE(t)$	expected negative exposure (4)
$PD(t_i, t_j)$	counterparty probability of default in $[t_i; t_j]$
$PD_{Bank}(t_i, t_j)$	our probability of default in $[t_i; t_j]$
LGD	counterparty loss given default
LGD_{Bank}	our loss given default

Note that the choice t_i in the arguments of $EPE(t_i)$ and $ENE(t_i)$ means we are choosing the *advanced* rather than the *postponed* discretization of the CVA/DVA integral [13]. This choice can be easily changed in the ORE source code or made configurable.
Moreover, formulas (10, 11) assume independence of credit and other market risk factors, so that PD and LGD factors are outside the expectations. With the extension of ORE to credit asset classes and in particular for wrong-way-risk analysis, CVA/DVA formulas will be generalised.

A.4 FVA

Any exposure (uncollateralised or residual after taking collateral into account) gives rise to funding cost or benefits depending on the sign of the residual position. This can be expressed as a Funding Value Adjustment (FVA). A simple definition of FVA can be given in a very similar fashion as the sum of unilateral CVA and DVA which we defined by (10,11), namely as an expectation of exposures times funding spreads:

$$\begin{aligned} FVA = & \underbrace{\sum_{i=1}^{n} f_b(t_{i-1}, t_i)\, \delta_i\, \mathbb{E}^N \left[S_C(t_{i-1})\, S_B(t_{i-1})\, (NPV(t_i))^+\, D(t_i) \right]}_{\text{Funding Benefit Adjustment (FBA)}} \\ & - \underbrace{\sum_{i=1}^{n} f_l(t_{i-1}, t_i)\, \delta_i\, \mathbb{E}^N \left[S_C(t_{i-1})\, S_B(t_{i-1})\, (-NPV(t_i))^+\, D(t_i) \right]}_{\text{Funding Cost Adjustment (FCA)}} \end{aligned} \tag{12}$$

where

$D(t_i)$ stochastic discount factor, $1/N(t_i)$ in LGM
$NPV(t_i)$ portfolio value after potential collateralization
$S_C(t_j)$ survival probability of the counterparty
$S_B(t_j)$ survival probability of the bank
$f_b(t_j)$ borrowing spread for the bank relative to the collateral compounding rate
$f_l(t_j)$ lending spread for the bank relative to the collateral compounding rate

For details see e.g. Chapter 14 in Gregory [16] and the discussion in [17].

A.5 COLVA

When the CSA defines a collateral compounding rate that deviates from the overnight rate, this gives rise to another value adjustment labeled COLVA [17]. In the simplest case the deviation is just given by a constant spread Δ:

$$COLVA = \mathbb{E}^N \left[\sum_i C(t_i) \cdot \Delta \cdot \delta_i \cdot D(t_{i+1}) \right] \tag{13}$$

where $C(t)$ is the collateral posted and $D(t)$ is the stochastic discount factor $1/N(t)$ in LGM. Both $C(t)$ and $N(t)$ are computed in ORE's Monte Carlo framework, and the expectation yields the desired adjustment.
Replacing the constant spread by a time-dependent deterministic function in ORE is straight forward.

A.6 Collateral Floor Value

A less trivial extension of the simple COLVA calculation above, also covered in ORE, is the case where the deviation between overnight rate and collateral rate is stochastic itself. A popular example is a CSA under which the collateral rate is the overnight rate *floored at zero*. To work out the value of this CSA feature one can take the difference of discounted margin cash flows with and without the floor feature. It is shown in [17] that the following formula is a good approximation to the collateral floor value

$$\Pi_{Floor} = \mathbb{E}^N \left[\sum_i C(t_i) \cdot (-r(t_i))^+ \cdot \delta_i \cdot D(t_{i+1}) \right] \tag{14}$$

where r is the stochastic overnight rate and $(-r)^+ = r^+ - r$ is the difference between floored and 'un-floored' compounding rate.
Taking both collateral spread and floor into account, the value adjustment is

$$\Pi_{Floor,\Delta} = \mathbb{E}^N \left[\sum_i C(t_i) \cdot ((r(t_i) - \Delta)^+ - r(t_i)) \cdot \delta_i \cdot D(t_{i+1}) \right] \tag{15}$$

A.7 Dynamic Initial Margin and MVA

The introduction of Initial Margin posting in non-cleared OTC derivatives business reduces residual credit exposures and the associated value adjustments, **CVA** and **DVA**.

On the other hand, it gives rise to additional funding cost. The value of the latter is referred to as Margin Value Adjustment (**MVA**).
To quantify these two effects one needs to model Initial Margin under future market scenarios, i.e. Dynamic Initial Margin (**DIM**). Potential approaches comprise

- Monte Carlo VaR embedded into the Monte Carlo simulation
- Regression-based methods
- Delta VaR under scenarios
- ISDA's Standard Initial Margin (SIMM) under scenarios

We skip the first option as too computationally expensive for ORE. In the first ORE release we focus on a relatively simple regression approach as in [18]. Consider the netting set values $NPV(t)$ and $NPV(t+\Delta)$ that are spaced one margin period of risk Δ apart. Moreover, let $F(t, t+\Delta)$ denote cumulative netting set cash flows between time t and $t+\Delta$, converted into the NPV currency. Let $X(t)$ then denote the netting set value change during the margin period of risk excluding cash flows in that period:

$$X(t) = NPV(t+\Delta) + F(t, t+\Delta) - NPV(t)$$

ignoring discounting/compounding over the margin period of risk. We actually want to determine the distribution of $X(t)$ conditional on the 'state of the world' at time t, and pick a high (99%) quantile to determine the Initial Margin amount for each time t. Instead of working out the distribution, we content ourselves with estimating the conditional variance $\mathbb{V}(t)$ or standard deviation $S(t)$ of $X(t)$, assuming a normal distribution and scaling $S(t)$ to the desired 99% quantile by multiplying with the usual factor $\alpha = 2.33$ to get an estimate of the Dynamic Initial Margin DIM:

$$\mathbb{V}(t) = \mathbb{E}_t[X^2] - \mathbb{E}_t^2[X], \qquad S(t) = \sqrt{\mathbb{V}(t)}, \qquad DIM(t) = \alpha\, S(t)$$

We further assume that $\mathbb{E}_t[X]$ is small enough to set it to the expected value of $X(t)$ across all Monte Carlo samples X at time t (rather than estimating a scenario dependent mean). The remaining task is then to estimate the conditional expectation $\mathbb{E}_t[X^2]$. We do this in the spirit of the Longstaff Schwartz method using regression of $X^2(t)$ across all Monte Carlo samples at a given time. As a regressor (in the one-dimensional case) we could use $NPV(t)$ itself. However, we rather choose to use an adequate market point (interest rate, FX spot rate) as regression variable x, because this is generalised more easily to the multi-dimensional case. As regression basis functions we use polynomials, i.e. regression functions of the form $c_0 + c_1\, x + c_2\, x^2 + ... + c_n\, x^n$ where the order n of the polynomial can be selected by the user. Choosing the lowest order $n = 0$, we obtain the simplest possible estimate, the variance of X across all samples at time t, so that we apply a single $DIM(t)$ irrespective of the 'state of the world' at time t in that case. The extension to multi-dimensional regression is also implemented in ORE. The user can choose several regressors simultaneously (e.g. a EUR rate, a USD rate, USD/EUR spot FX rate, etc.) in order order to cover complex multi-currency portfolios.

Given the DIM estimate along all paths, we can next work out the Margin Value Adjustment [17] in discrete form

$$MVA = \sum_{i=1}^{n} (f_b - s_I)\, \delta_i\, S_C(t_i)\, S_B(t_i) \times \mathbb{E}^N \left[DIM(t_i)\, D(t_i) \right]. \tag{16}$$

with borrowing spread f_b as in the FVA section A.4 and spread s_I received on initial margin, both spreads relative to the cash collateral rate.

A.8 Collateral Model

The collateral model implemented in ORE is based on the evolution of collateral account balances along each Monte Carlo path taking in to account thresholds, minimum transfer amounts and independent amounts defined in the CSA, as well as margin periods of risk. ORE computes the collateral requirement (aka *Credit Support Amount*) through time along each Monte Carlo path

$$CSA(t_m) = \begin{cases} \max(0, V_{set}(t_m) - I_A - T_{hold}), & V_{set}(t_m) - I_A \geq 0 \\ \min(0, V_{set}(t_m) - I_A + T_{hold}), & V_{set}(t_m) - I_A < 0 \end{cases} \tag{17}$$

where

- $V_{set}(t_m)$ is the value of the netting set as of time t_m
- T_{hold} is the threshold exposure below which no collateral is required (possibly asymmetric)
- I_A is the sum of all collateral independent amounts attached to the underlying portfolio of trades (positive amounts imply that the bank has received a net inflow of independent amounts from the counterparty), assumed here to be cash.

As the collateral account already has a value of $C(t_m)$ at time t_m, the collateral shortfall is simply the difference between $C(t_m)$ and $CSA(t_m)$. However, we also need to account for the possibility that margin calls issued in the past have not yet been settled (for instance, because of disputes). If $M(t_m)$ denotes the net value of all outstanding margin calls at t_m, and $\Delta(t)$ is the difference $\Delta(t) = CSA(t_m) - C(t_m) - M(t_m)$ between the *Credit Support Amount* and the current and outstanding collateral, then the actual margin *Delivery Amount* $D(t_m)$ is calculated as follows:

$$D(t_m) = \begin{cases} \Delta(t), & |\Delta(t)| \geq MTA \\ 0, & |\Delta(t)| < MTA \end{cases} \tag{18}$$

where MTA is the minimum transfer amount.

Finally, the *Delivery Amount* is settled with a delay specified by the *Margin Period of Risk* (MPoR) which leads to residual exposure and XVA even for daily margining, zero thresholds and minimum transfer amounts, see for example [14]. A more detailed framework for collateralised exposure modelling is introduced in the 2016 article [19], indicating a potential route for extending ORE.

A.9 Exposure Allocation

XVAs and exposures are typically computed at netting set level. For accounting purposes it is typically required to *allocate* XVAs from netting set to individual trade level such that the allocated XVAs add up to the netting set XVA. This distribution is not trivial, since due to netting and imperfect correlation single trade (stand-alone) XVAs hardly ever add up to the netting set XVA: XVA is sub-additive similar to VaR. ORE provides an allocation method (labeled *marginal allocation* in the following) which slightly generalises the one proposed in [15]. Allocation is done pathwise which first leads to allocated expected exposures and then to allocated CVA/DVA by inserting these exposures into equations (10,11). The allocation algorithm in ORE is as follows:

- Consider the netting set's discounted NPV after taking collateral into account, on a given path at time t:

$$E(t) = D(0,t)\left(NPV(t) - C(t)\right)$$

- On each path, compute contributions A_i of the latter to trade i as

$$A_i(t) = \begin{cases} E(t) \times NPV_i(t)/NPV(t), & |NPV(t)| > \epsilon \\ E(t)/n, & |NPV(t)| \leq \epsilon \end{cases}$$

  with number of trades n in the netting set and trade i's value $NPV_i(t)$.

- The EPE fraction allocated to trade i at time t by averaging over paths:

$$EPE_i(t) = \mathbb{E}\left[A_i^+(t)\right]$$

By construction, $\sum_i A_i(t) = E(t)$ and hence $\sum_i EPE_i(t) = EPE(t)$.
We introduced the *cutoff* parameter $\epsilon > 0$ above in order to handle the case where the netting set value $NPV(t)$ (almost) vanishes due to netting, while the netting set 'exposure' $E(t)$ does not. This is possible in a model with nonzero MTA and MPoR. Since a single scenario with vanishing $NPV(t)$ suffices to invalidate the expected exposure at this time t, the cutoff is essential. Despite introducing this cutoff, it is obvious that the marginal allocation method can lead to spikes in the allocated exposures. And generally, the marginal allocation leads to both positive and negative EPE allocations.

As a an example for a simple alternative to the marginal allocation of EPE we provide allocation based on today's single-trade CVAs

$$w_i = CVA_i / \sum_i CVA_i.$$

This yields allocated exposures proportional to the netting set exposure, avoids spikes and negative EPE, but does not distinguish the 'direction' of each trade's contribution to EPE and CVA.

References

[1] `http://www.opensourcerisk.org`

[2] `http://www.quantlib.org`

[3] `http://www.quaternion.com`

[4] `http://quantlib.org/install/vc10.shtml`

[5] `https://git-scm.com/downloads`

[6] `https://sourceforge.net/projects/boost/files/boost-binaries`

[7] `http://www.boost.org`

[8] `http://jupyter.org`

[9] https://docs.continuum.io/anaconda

[10] http://www.libreoffice.org

[11] Basel Committee on Banking Supervision, *International Convergence of Capital Measurement and Capital Standards, A Revised Framework*, http://www.bis.org/publ/bcbs128.pdf, June 2006

[12] Basel Committee on Banking Supervision, *Basel III: A global regulatory framework for more resilient banks and banking systems*, http://www.bis.org/publ/bcbs189.pdf, June 2011

[13] Damiano Brigo and Fabio Mercurio, *Interest Rate Models: Theory and Practice, 2nd Edition*, Springer, 2006.

[14] Michael Pykhtin, *Collateralized Credit Exposure*, in Counterparty Credit Risk, (E. Canabarro, ed.), Risk Books, 2010

[15] Michael Pykhtin and Dan Rosen, *Pricing Counterparty Risk at the Trade Level and CVA Allocations*, Finance and Economics Discussion Series, Divisions of Research & Statistics and Monetary Affairs, Federal Reserve Board, Washington, D.C., 2010

[16] Jon Gregory, *Counterparty Credit Risk and Credit Value Adjustment, 2nd Ed.*, Wiley Finance, 2013.

[17] Roland Lichters, Roland Stamm, Donal Gallagher, *Modern Derivatives Pricing and Credit Exposure Analysis, Theory and Practice of CSA and XVA Pricing, Exposure Simulation and Backtesting*, Palgrave Macmillan, 2015.

[18] Fabrizio Anfuso, Daniel Aziz, Paul Giltinan, Klearchos Loukopoulos, *A Sound Modelling and Backtesting Framework for Forecasting Initial Margin Requirements*, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2716279, 2016

[19] Leif B. G. Andersen, Michael Pykhtin, Alexander Sokol, *Rethinking Margin Period of Risk*, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2719964, 2016