Dr. Sebastian Schlenkrich
Institut für Mathematik
Bereich Stochastik

April 28, 2022

Interest Rate Modelling and Derivative Pricing, SS 2022

Exercise 1

1. **Preliminaries**
   Exercises will comprise in large parts setting up, extending and modifying Python scripts or Jupyter notebooks. We will also use the open source QuantLib financial library which facilitates many financial modelling tasks. You will need a Python environment with the QuantLib package included.
   We will demonstrate the examples with *Anaconda Python* and *Visual Studio Code* IDE and Jupyter .[1] A Python environment with QuantLib (and some other useful packages) can be set up along the following steps

   (a) Download and install Anaconda Python 3.x from `www.anaconda.com/download/`. Download Visual Studio Code IDE either via Anaconda Navigator / home screen or manually from `https://code.visualstudio.com/`. Jupyter Notebook also comes with Anaconda and VS Code.

   (b) Run 'Anaconda Prompt', create the new Python 3.x environment, and install required packages:
   ```
   conda create -n MyPython Python
   conda activate MyPython
   pip install matplotlib QuantLib numpy pandas scipy tqdm
   ```

   (c) Test Python environment and QuantLib: Launch Visual Studio Code from Anaconda Navigator > Home (make sure your Python 3.x environment is still selected). Select a working directory via File > Open Folder. Create a new file `test.py`. Include Python script
   ```
   import pandas as pd
   import matplotlib.pyplot as plt
   import QuantLib as ql
   d = ql.Settings.instance().evaluationDate
   print(d)
   ```
   Execute the script via right click on `test.py` and 'Run Python File in Terminal'. You should see a terminal output like
   ```
   April 28th, 2022
   ```

   (d) Python/QuantLib examples will be provided via GitHub at
   `https://https://github.com/sschlenkrich/InterestRateModelling_examples`.
   We recommend using the version control tool `git` to maintain up-to-date versions of the example scripts and keep track of your own modifications. Git is available via `https://git-scm.com/`. An easy to use Windows client is also available at `https://tortoisegit.org/`. You can clone a copy of the repository via
   ```
   git clone https://github.com/sschlenkrich/InterestRateModelling_examples
   ```
   Updated versions can be obtained e.g. via `git pull`. A history of your own modifications can be maintained via `git add [...]` and `git commit [...]` (see git documentation for details of the commands). Git offers much more functionality with detailed documentation, see also `https://git-scm.com/book/en/v2`.

[1] QuantLib is also available with the standard Python 3.x distribution from `www.python.org`.

Alternatively, you could also download the code as ZIP file from the Github web site. However, this workflow is not recommended.
The examples will act as a basis for the exercises. They show the usage of QuantLib and provide building blocks for other modelling exercises. For further reading on QuantLib/Python we suggest e.g. L. Ballabio and G. Balaraman. *QuantLib Python Cookbook*. (`https://leanpub.com/quantlibpythoncookbook`). Some QuantLib documentation is also available at `https://quantlib-python-docs.readthedocs.io/en/latest/`.

2. **Yield curve interpolation**
The module `src/yieldcurve.py` illustrates the use of QuantLib yield curve objects based on interpolated forward rates. Alternatively, QuantLib/Python allows setting up yield curves based on interpolated zero rates using different interpolations:
`QuantLib.ZeroCurve`
`QuantLib.LogLinearZeroCurve`
`QuantLib.CubicZeroCurve`
`QuantLib.NaturalCubicZeroCurve`
`QuantLib.LogCubicZeroCurve`
`QuantLib.MonotonicCubicZeroCurve`
Further details on the interface can be found in the SWIG interface file
`https://github.com/lballabio/QuantLib-SWIG/blob/master/SWIG/zerocurve.i`

   (a) Set up a yield curve class that takes as inputs zero rates and interpolates as indicated in above class names.

   (b) Analyse and compare forward rates, continuous compounded zero rates and annually compounded zero rates for the various interpolation methods. As example data use zero rates calculated from the example curve in `examples/YieldCurve.ipynb`. Also try other input data (e.g. flat curve, descending curve data, negative interest rates).

   (c) Explain the basic principes of the interpolation methods tested. Which interpolation method do you reccomend for interest rate modelling? Why?

3. **Fixed leg pricing with QuantLib**
Use the documentation at `https://quantlib-python-docs.readthedocs.io/en/latest/` and replicate the fixed leg pricing from spread sheet YieldCurvesAndLegs.xlsx, "Dates & Fixed Leg" in QuantLib/Python.

4. **Bond yield and spread calculation**
Fixed rate bonds are instruments that pay deterministic cash flows. The present value $V^{\text{Bond}}(0)$ of a bond is often expressed in terms of *yields* and *spreads*.
Consider a fixed rate bond paying $K = 3\%$ coupon rate (30/360 day count convention for year fractions τ_i in coupon calculation) at future pay dates in $1y, , 2y, \ldots, 10y$ and the notional amount at final maturity in $10y$. All pay dates are assumed based on TARGET business day calendar and following business day convention. The *yield* of a fixed rate bond is defined as the flat annual compounded rate y that solves

$$V^{\text{Bond}}(0) = \frac{1}{(1+y)^{\tau(d_0,d_N)}} + \sum_{i=1}^{N} \frac{K \cdot \tau_i}{(1+y)^{\tau(d_0,d_i)}}$$

for given present value (or dirty market price) $V^{\text{Bond}}(0)$ and fixed rate K. Year fractions $\tau(d_0, d_i)$ are assumed also based on 30/360 day count convention.
The *z-spread* (or zero-rate spread) s is defined as the flat spread which needs to be added to a given zero rate curve $z(T)$ such that discounted cash flows equal a given present value. For example, for continuous compounded zero rates the z-spread is defined via

$$V^{\text{Bond}}(0) = e^{-[z(T_N)+s]\tilde{\tau}(d_0,d_N)} + \sum_{i=1}^{N} K \cdot \tau_i \cdot e^{-[z(T_i)+s]\tilde{\tau}(d_0,d_i)}.$$

Here, year fractions $\tilde{\tau}(d_0, d_i)$ of the yield curve are typically based on Act/365 (Fixed) day count convention.

(a) Set up a Python function that calculates the bond yield for above bond example given a present value. Plot yield versus present value for $V^{\text{Bond}}(0) \in [0.5, 1.5]$. How does the bond yield y relate to the bond's fixed rate K?

(b) Set up a Python function that calculates the z-spread for above bond example given a present value and an input zero rate yield curve. Use the yield curve in `testYieldCurve.py` as input yield curve. Plot z-spread versus present value for $V^{\text{Bond}}(0) \in [0.5, 1.5]$.

(c) Discuss pro's and con's of using yield or z-spread as a sensible representation of the bond's market price.

Hint: Use yield curve objects based on interplated zero rates to efficiently calculate the discount factors $1/\left(1+y\right)^{\tau(d_0,d_i)}$ and $e^{-[z(T_i)+s]\tau(d_0,d_i)}$. The corresponding equations can be solved e.g. using Brent solver from `scipy.optimize.brentq`. See also module `QuantLibWrapper.Helpers`.

5. **Ito isometry**
Consider the scalar Ito integral

$$X(T) = \int_0^T \sigma(t) dW(t)$$

with deterministic and piece-wise constant volatility function $\sigma(t) = \sigma_i$ for $t \in [t_{i-1}, t_i)$ for $i = 1, \ldots n$, $t_0 = 0$, $t_n = T$ and scalar Brownian motion $W(t)$. Show that

$$\mathbb{E}^{\mathbb{P}}\left[X(T)^2\right] = \int_0^T \sigma(t)^2 dt.$$

6. **Market price of risk**
Consider a market with two assets $X_1(t)$ and $X_2(t)$. Assume that both assets follow geometric Brownian motions

$$\frac{dX_{1,2}(t)}{X_{1,2}} = \mu_{1,2} \cdot dt + \sigma_{1,2} \cdot dW(t)$$

with scalar drifts μ_1 and μ_2 and scalar volatilities σ_1 and σ_2. Both assets are driven by a unique Brownian motion $W(t)$ under the real-world meassure $\mathbb{P}$. Moreover, assume a bank account $B(t) = \exp\left\{\int_0^t r \cdot ds\right\}$ as numeraire with risk-free rate r and equivalent martingale measure $\mathbb{Q}$.

(a) Show that

$$\frac{dX_{1,2}(t)}{X_{1,2}} = r \cdot dt + \sigma_{1,2} \cdot d\tilde{W}(t)$$

where $\tilde{W}(t)$ is a Brownian motion under $\mathbb{Q}$.

(b) Moreover, show that the drifts and volatilities are related via

$$\lambda = \frac{\mu_1 - r}{\sigma_1} = \frac{\mu_2 - r}{\sigma_2}.$$

Note: λ is called the market price of risk. Provided the existence of an equivalent martingale meassure (i.e. absence of arbitrage) you cannot get more return (i.e. drift) without taking more risk (i.e. volatility).