QFGB8960 Advanced C++ for Finance
Homework 7

Spring 2025

Problem 1 (50 points) Cap-Floor Parity

a) Create a function `capFloorletBS` to be declared in the file

`qflib/pricers/simplepricers.hpp`

and implemented in the file `qflib/pricers/simplepricers.cpp`.

The function signature should be

```
double capFloorletBS(int payoffType,
                     SPtrYieldCurve spyc,
                     double strikeRate,
                     double timeToReset,
                     double tenor,
                     double fwdRateVol);
```

The function uses eqs. (12-14) in the lecture notes to compute the price of a caplet or a floorlet.

`payoffType` is 1 for cap and -1 for floor

`spyc` is a shared pointer to the yield curve

`strikeRate` is the fixed strike rate, annualized and with simple compounding

`timeToReset` is the time to the reset (fixing) of the future rate, in years

`tenor` is the time span between reset and payment, in years

`fwdRateVol` is the annualized volatility of the forward rate

In the notation of eq. (12) of the lecture notes, current time is $t = 0$, and $\phi = 1/-1$ means caplet/floorlet. For a given caplet or floorlet i, T_{i-1} is `timeToReset`, $T_i - T_{i-1}$ is `tenor`, and $\bar{R}$ is `strikeRate`. The currently observed forward rate $F(t = 0, T_{i-1}, T_i)$ is to be obtained from the yield curve pointed to by `spyc`, and converted to simple compounding within each accrual period. Finally $\sigma_F(t = 0; T_{i-1}, T_i)$ is `fwdRateVol`.

Expose this function to Python as

```
qf.capFloorletBS(payType, ycName, strikeRate, timeToReset,
                 tenor, fwdRateVol)
```

Use the file `pyqflib/pyfunctions2.hpp` for implementation of the C++ function.

b) In Python notebook, create a yield curve with name "USD-LIBOR" using the function `qf.ycCreate` and the maturities (in years) and spot rates (cont. cmpd) below.

Maturity	SpotRate
1/12	1.00%
1/4	2.00%
1/2	3.00%
3/4	3.50%
1	4.00%
2	4.75%
3	5.25%
5	6.00%

Use a 5% rate (annualized with semiannual compounding) as the strike rate for all caplets/floorlets.

Use the same 5% rate as the coupon for the fixed leg of an interest rate swap (IRS).

Assume a forward rate volatility of 20% and a notional of 1,000,000 USD, to be used as the notional for all caplets/floorlets and the IRS.

Then, do the following:

1. Compute the total price of a sequence of five back-to-back caplets, all with a half year tenor. Caplet one resets half a year from now and pays one year from now. Caplet two resets one year from now and pays one and a half years from now, end so on until the final payment of the fifth caplet three years from now. Use the function `qf.capFloorletBS` that you coded in the previous part.

2. Compute the total price of the same sequence of five floorlets, with same strike rate and volatility.

3. Price an IRS on the "USD-LIBOR" yield curve that has a maturity of three years, paying semiannually. The fixed coupon is 5% p.a. with semiannual compounding. The first reset time is half a year from now and the first payment time is one year from now. The swap continues for three years, i.e. it has five reset times and five payment times, matching the caplets/floorlets above.

4. Verify the parity relation for the present values CAP-FLOOR = SWAP.

5. Make sure you use simple compounding for the forward rate in each semiannual accrual period. Use the function `qf.fromContCmpd` to convert the cc forward rate given by the yield curve, to a semiannual rate, to be used for computing the floating payments of the IRS.

Submit your source code and the notebook.

Problem 2 (50 points) CDS Valuation

a) Create a function `cdsPV` to be declared in the file `qflib/pricers/simplepricers.hpp` and implemented in the file `qflib/pricers/simplepricers.cpp`.

The function signature should be

```
qf::Vector cdsPV(SPtrYieldCurve sprfyc,
                 double credSprd,
                 double cdsRate,
                 double recov,
                 double timeToMat,
                 double payFreq);
```

The function uses eqs. (21, 22) in the lecture notes to compute the PV of the default leg and the premium leg of a CDS per one unit of notional ($F = 1$).

The returned vector should contain two values, PV(DefaultLeg) and PV(PremiumLeg) in that order.

`sprfyc` is a shared pointer to the risk free yield curve

`credSprd` is the annualized constant credit spread with **continuous compounding**

`cdsRate` is the CDS premium rate, annualized, with **payfreq compounding**

`recov` is the recovery rate

`timeToMat` is the time to maturity in years

`payFreq` is the annual premium pay frequency, i.e. number of times per year that premium is paid or default payment is made.

- The number of payments should be `timetoMat*payFreq`. If `timeToMat` is not an integer, the number of payments should be `ceiling(timeToMat*payFreq)`. Place the stub (shorter) payment at the beginning of the CDS schedule.
- Make sure the survival probabilities are always non-negative for non-zero recovery.
- Validate all input parameters.

Expose this function to Python as

```
qf.cdsPV(rfreeYC, credSpread, cdsRate, recov, timeToMat, payFreq)
```

Use the file `pyqflib/pyfunctions2.hpp` for implementation of the C++ function.

The function should return an 1×2 vector with the PV of the default and premium legs in that order.

b) Reuse the notebook of the previous problem, and in particular the yield curve "USD-LIBOR".

Using a constant credit spread of 1% (cont. cmpd), a CDS rate of 1% (cont. cmpd.) and **zero recovery**, price a sequence of semiannual CDS with maturity of 1, 2, 3, 4 and 5 years, applying the ORF.CDSPV function. All CDS should have a notional of 1,000,000.

Verify that the PV(DefaultLeg) - PV(PremiumLeg) = 0 for every maturity.

Do not forget to convert the CDS rate from continuous to semiannual compounding before passing it to the `qf.cdsPV` function.