Pricing of Accreting Swaptions using QuantLib

Dr. André Miemiec, 13./14. Nov. 2013
1. Introduction
2. Model Description
3. Implementation in QuantLib
4. Pricing Quality
Introduction

- Origin of the problem:
  - Valuation of Multicallable Accreting Swaptions

- Elementary Observations:

  Picture removed
Reason must be traced back to the model choice or calibration, respectively.

Amortising swaptions are most sensitive to 'parallel' moves in the yield curve, so a single factor model is sufficient ⇒ LGM.
QuantLib does not provide a LGM implementation but possesses an unsatisfactory implementation of Hull-White.

Calibration Issue:
- Accreters are calibrated to cointial not coterminial swaptions
- HW is unable to cope with this requirement.

I had to decide between two alternatives:
- do a proper LGM implementation or
- do the calibration otherwise.

Made the second choice because
- the method selected combines the best properties of the Black and 1F-Short-Rate Models.
1. Introduction
2. Model Description
3. Implementation in QuantLib
4. Pricing Quality
• Irregular swap and its decomposition into a basket of regular swaps

- Hagan*: Want to exercise all basket swaps at the same time, i.e. put them equally far (\(\lambda\)) from ATM (\(K_i\))

\[ R_i = K_i + \lambda \]

*The corresponding reference can be found at the end of the talk.
Model Description

- Bond model of an accreting swaption:

  Fixed Leg:
  \[
  C_i = N_i \tau_i K + N_i - N_{i-1} \\
  C_n = N_n \tau_n K + N_n
  \]

  Float Leg:
  \[
  \tilde{C}_1 = N_1
  \]

- Basket of standard swaps with par-rates \( \{K_i\}_{i=1..n} \) and notionals \( \{A_i\}_{i=1..n} \)

  \[
  \begin{pmatrix}
  1 + (K_1 + \lambda) \cdot \tau \\
  0 \\
  \vdots
  \end{pmatrix}
  \begin{pmatrix}
  (K_2 + \lambda) \cdot \tau \\
  1 + (K_2 + \lambda) \cdot \tau \\
  \vdots
  \end{pmatrix}
  \begin{pmatrix}
  A_1 \\
  A_2 \\
  \vdots
  \end{pmatrix}
  =
  \begin{pmatrix}
  C_1 \\
  C_2 \\
  \vdots
  \end{pmatrix}
  \]

- Matching the floating leg:

  \[
  N_1 = \sum_{i=1}^{N} A_i(\lambda) \quad \Rightarrow \quad \lambda
  \]
Model Description

- Basket decomposition:
  \[ U_t = \sum_{i=1}^{N} A_i U_t^i(R_i) \]

- Hunt-Kennedy**: 
  
  ➢ Select \( r^* \) such that:
  \[ U_t(r^*) = 0 \]
  
  ➢ Select \( R_i \) such that:
  \[ U_t^i(R_i, r^*) = 0 \]

- Then

  \[ V_t(r) = e^{-rt} \cdot E\left[ U_t^+(r) \right] = \sum_{i=1}^{N} V_t^i(R_i, r) \]

- This decomposition works pretty well, if Hagan’s \( R_i \) are actually used.
  
  ➢ Typical deviation to a properly calibrated LGM model some $100

**The corresponding reference can be found at the end of the talk.
1. Introduction
2. Model Description
3. Implementation in QuantLib
4. Pricing Quality
Implementation in QuantLib

- Basic structure of the algorithm

**Instruments**
- IrrSwap
- IrrSwptn

**PricingEngine**
- HaganIrregularSwaptionEngine
  - void calculate() const;
  - Real HKPrice (Basket&, …) const;

**MarketData**
- SwptnVol
- YTStruct

**Basket**
- Disposable<Array> compute(Rate lambda = 0.0) const;
- Mutable Real lambda_;

**Methods**
- SVD
- Black76
- Bisection
• Final Pricing Function:

```cpp
Real HKPrice(Basket& basket, boost::shared_ptr<Exercise>& exercise) const {
    boost::shared_ptr<PricingEngine> blackSwaptionEngine = boost::shared_ptr<PricingEngine>(
        new BlackSwaptionEngine(termStructure_, volatilityStructure_));

    Disposable<Array> weights = basket.weights();

    Real npv = 0.0;

    for(Size i=0; i<weights.size(); ++i){
        boost::shared_ptr<VanillaSwap> pvSwap_ = basket.component(i);
        Swaption swaption = Swaption(pvSwap_, exercise);
        swaption.setPricingEngine(blackSwaptionEngine);
        npv += weights[i] * swaption.NPV();
    }

    return npv;
}
```
Side remark on standard QL-Classes:

• Observation:
  - Implementation of Swaption-Instrument is tightly bound to the implementation of a VanillaSwap-Instrument

• Suggestion:
  - Need for a Constructor of class VanillaSwap, who allows for all sorts of schedules

Future developments regarding this piece of code:

• automatic calibration of a bermudan swaption (get rid of tedious nested calibration)
• Full fledged LGM model with all sorts of calibrations
Agenda

1. Introduction
2. Model Description
3. Implementation in QuantLib
4. Pricing Quality
Pricing Quality: Example Callable Zerobond

IRR = 4,1055%

1. With original calibration from FO-System:

From FO-System:
PV-FO-System: € …..

Benchmarking:
PV-QuantLib: € …..

PV01: € 100k

ΔPV ≈ 0.2 bp
2. Comparison against market prices with own calibration from QuantLib:

Picture removed
● Main Result: improved fitness of prices to market

● Reason for the observed effects:

  ➢ Because the HK-Prices of Accreting Swaptions are pretty close to the corresponding LGM Prices the new calibration is more consistent than the result of a calibration based on a weighted vol (Black) approach.

~ The End ~
References

- P.S. Hagan, Methodology for Callable Swaps and Bermudan Exercise into Swaptions
- A. Miemiec, QuantLib Code on SourceForge, (2013)