MARKOV FUNCTIONAL ONE FACTOR INTEREST RATE MODEL IMPLEMENTATION IN QUANTLIB

PETER CASPERS

First Version October 21, 2012 - This Version April 14, 2013

Abstract. We describe the implementation of a Markov functional one factor interest rate model in the QuantLib [4] framework.

1. Model description

The presentation of the theoretical background is mainly following [2].

1.1. **Basic setup.** The dynamics in a Markov functional one factor model is given by a one dimensional markovian state variable x. Specifically we use the following process for the evolution of x

$$dx = \sigma(t)e^{at}dW(t) \tag{1.1}$$

with a time dependent volatility σ and a constant mean reversion a. At time $t = 0$ we assume $x(0) = 0$. We choose σ to be piecewise constant. This process is implemented in the class `MfStateProcess`.

Different than in other models, where the driving process is related directly to an at least theoretically observable market parameter like a short rate, a discrete Libor rate etc., in the Markov approach the state variable does not have any direct interpretation in terms of a financial variable. Instead this relation is established by fixing a numeraire N and writing

$$N(t) = N(t, x(t)) \tag{1.2}$$

We will use $N(t) := P(t, T)$, the zerobond with maturity T chosen such that all payments to be priced in our model occur on times $t \leq T$, i.e. our model evolves in the T forward measure. Payment means not necessarily a physical payment here but can also refer to a payment in an underlying, e.g. the payments on the fixed side of an swap underlying a cms coupon payment of the deal. If for example a deal has its last payment on $t = 10y$ paying a cms 10y rate fixed in arrears, then we will need to chose $T = 20y$, because the last payment of the underlying 10y swap will be 10y after the cms coupon fixing.

The function N relates the zerobond price at time t to the model state $x(t)$. We decide to follow a nonparameteric approach, i.e. we do not put any restrictions on the shape of the numeraire surface N. This ensures maximal calibration capability of our model.

Date: April 14, 2013.

1.2. **Calibration instrument sets.** If we fix σ and a we are in position to calibrate $N(t,\cdot)$ for given t such that a given continuum of option prices with expiry t and with strikes $K \in [0,\infty)$ (possibly $[-c,\infty)$ for some $c>0$ even) is matched by the model. In other words, the numeraire function can be used to match volatility smiles of caplets or swaptions on an arbitrary set of expiries. Note that we can however only match one smile on a given expiry, i.e. we have to choose an underlying. For example, for a fixed expiry t, we can calibrate to swaptions on a 10y underlying swap rate for all strikes, but not at the same time to swaptions on a 2y underyling. As a consequnence the one factor model will be suitable for the pricing of cms coupons, but too narrow for cms spread coupons. Also if options on cms spread coupons are part of the payoff it is likely that the model will underestimate the value of such an option because we only have one factor and therefore perfect correlation between rates of different maturity.

The function σ can be used to calibrate to a second set of instruments, but obviously only for one strike per expiry.

This can be used to calibrate the numeraire N for example to 10y cms swaption smiles on the fixing dates of cms coupons of a deal to correctly price these cms coupons in the model and at the same calibrate the volatility function σ to atm coterminal swaptions in order to match european (plain vanilla) call rights. The model will then be a good candidate to price multicallable cms swaps. However if the underlying cms swap exchanges the cms rate with a libor rate the same comments as above for options on cms spread coupons apply.

Another use case would be the pricing of a multicallable reverse capped floored Libor swap, for which we can calibrate to caplet smiles and coterminal swaptions.

1.3. **The mean reversion.** We will not calibrate the mean reversion a, but let it fixed, e.g. feeded from some global calibration outside the specific pricing in the markov model. The role of this model parameter is to control intertemporal correlations of the state variable x. Indeed if we assume σ constant for a moment we get

$$< dx, dx >_T = \frac{\sigma^2}{2}\left(e^{2aT}-1\right) \tag{1.3}$$

and therefore

$$\operatorname{corr}(x(T_1),x(T_2)) = \sqrt{\frac{e^{2aT_2}-1}{e^{2aT_1}-1}} = e^{-a(T_2-T_1)}\sqrt{\frac{1-e^{-2aT_1}}{1-e^{-2aT_2}}} \tag{1.4}$$

which shows that for $a=0$ the correlation is $\sqrt{T_1/T_2}$ and goes to zero if $a\to\infty$ and to one if $a\to-\infty$. Formula 1.4 is the same as in the Hull White one factor model.

1.4. **Multicurve modeling.** For the time being we set up the model in a single curve world. In particular the discount curve coincides with all forward curves.

1.5. **Floater approximation.** We use a simplified formula for the npv at time t of a floating rate leg (without nominal payment), namely

$$P(t,t_s) - P(t,t_e) \tag{1.5}$$

where t_s is the start date and t_e is the last payment date of the schedule. This ignores the possible mismatches between the accrual, payment and index schedules occuring in reality.

2. The numerical representation

2.1. **Discretization of the numeraire.** The numeraire surface $N(\cdot,\cdot)$ is represented as follows. First we normalize $x(t)$ by setting

$$y(t) := \left(\int_0^t \left[e^{at}\sigma(t)\right]^2 dt\right)^{-1/2} x(t) =: \Sigma(t)^{-1}x(t) \tag{2.1}$$

Obviously $y(t)$ is standard normal distributed for each t. From now on we write N in (t,y) coordinates instead of (t,x).

The times $t_i, i = 1,...,n$ on which the numeraire is calibrated is given externally by the fixing times of the first set of calibration instruments. For each of these t_i we discretize $y(t)$ by

$$y_j(t_i) = jh \tag{2.2}$$

with

$$j = -m, -m+1, ..., -1, 0, 1, ..., m-1, m \tag{2.3}$$

and h chosen as

$$h = s_{\max}/m \tag{2.4}$$

The idea behind is to cover $s_{\max}$ standard deviations of the original state variable x in the grid. The number of discretization steps m is given by the parameter `yGridPoints` in the model constructor `MarkovFunctional` and defaulted to 64. The number of standard deviations $s_{\max}$ is given by the parameter `yStdDevs` and defaulted to 7.0. The discretized variable y is stored as the member variable `y_` in the `MarkovFunctional` class.

The numeraire is stored in `numeraireDiscretization_` as a vector of a vector: The inner vector contains $\tilde{N}(t_i, y_j)$ with fixed i. The outer vector stores the different $\ln \tilde{N}(t_i,\cdot)$ for all i. As indicated by the tilde we actually do not store the numeraire directly, but a normalized numeraire defined by

$$\tilde{N}(t,y) = N(t,y)\frac{P(0,t)}{P(0,T)} \tag{2.5}$$

as also proposed in [2].

2.2. **Interpolation and Extrapolation.** The numeraire is interpolated in the normalized state variable y with monotonic cubic splines with Lagrange boundary condition. Extrapolation outside the boundaries for y is done flat. The interpolation in time direction is done linearly in the reciprocal of the normalized numeraire. Extrapolation in time direction is also done flat. Note that extrapolating in time direction should never be necessary when the calibration instrument set is specified correctly. It is merely allowed in the implementation for technical reasons, e.g. if

a discount factor is required for a maturity only a few days after the numeraire date. Still the extrapolation in time direction is robust in the sense that the input yield term structure is reproduced, this is a basic consequence of the normalization formula 2.5. However there is of course no dynamics out there because $N(\equiv 1)$ is constant in y.

3. Calibration

3.1. **Determination of calibration times and the numeraire maturity date.** The calibration times are determined by an algorithm which starts with the fixing dates of the calibration instruments given in the model constructor. Initially the numeraire is known on the maximum payment date of all underlying instruments which is also the inital numeraire maturity date (the numeraire is 1 there by definition). Working backward through the calibration expiries it is checked whether the numeraire is known on all payment dates of the current options underlying. If not the set of calibration instruments is enriched such that the numeraire can be bootstrapped on the date in question. Since new payment dates come into existence by this the whole procedure has to be restarted. If the backward loop through all calibration expiries ends without the need for new expiry dates the algorithm stops.

3.2. **Numeraire Bootstrapping.** We begin with the observation

$$\left.\frac{P(t,u)}{P(t,T)}\right|_{y(t)} = E\left(\frac{1}{P(u,T)}\middle| y(t)\right) \tag{3.1}$$

for $t \leq u$. This is just the general pricing formula applied to the asset $P(\cdot,u)$. With this relation we are able to compute the deflated zerobond price on the left hand side by an gaussian integral over the reciprocal of the numeraire on the right hand side, the latter evaluated on the maturity of the required zerobond. This is done in the function `deflatedZerobond` using a Gauss Hermite Integration. The number of points used in the numerical integration is given by the parameter `gaussHermitePoints` in the model constructor `MarkovFunctional` and is defaulted to 32. The evaluation of 3.1 is done as follows: Conditional on $y(t) = x(t)/\Sigma(t)$ the variable $x(u)$ is distributed normally with mean $x(t)$ and variance $\Sigma(t,u)^2 := \Sigma(u)^2 - \Sigma(t)^2$. Therefore 3.1 can be written

$$\frac{1}{\Sigma(t,u)}\int_{-\infty}^{\infty}\frac{1}{N(u,x(u)/\Sigma(u))}\phi\left(\frac{x(u)-x(t)}{\Sigma(t,u)}\right)dx(u) \tag{3.2}$$

We can rewrite this integral as

$$\int_{-\infty}^{\infty}\frac{1}{N\left(u,\frac{z(u)\Sigma(t,u)+\Sigma(t)y(t)}{\Sigma(u)}\right)}\phi(z(u))dz(u) \tag{3.3}$$

which is the formula that is actually implemented. At this point we describe a related function which is not used here but will be useful in the implementation of pricing engines later on. For pricing we will need to evaluate expressions like

$$E\left(\frac{f(x(u)/\Sigma(u))}{P(t,u)}\middle| y(t)\right) \tag{3.4}$$

for some function f and $u > t$. Just as in 3.2 we can write this expectation as

$$\frac{1}{\Sigma(t,u)} \int_{-\infty}^{\infty} \frac{f(x(u)/\Sigma(u))}{N(u, x(u)/\Sigma(u))} \phi\left(\frac{x(u) - x(t)}{\Sigma(t,u)}\right) dx(u) \tag{3.5}$$

and again rewriting in $z(u)$

$$\int_{-\infty}^{\infty} \frac{f\left(\frac{z(u)\Sigma(t,u)+\Sigma(t)y(t)}{\Sigma(u)}\right)}{N\left(u, \frac{z(u)\Sigma(t,u)+\Sigma(t)y(t)}{\Sigma(u)}\right)} \phi(z(u))dz(u) \tag{3.6}$$

Obviously it is useful to have a discretization of the rather complicated expression for $y(u)$ in the argument of f respectively $N(u, \cdot)$. This can be retrieved from the function `yGrid` taking the number of standard deviations and a number n of points resulting in a grid consisting of $2n + 1$ equidistant points as well as t, $y(t)$ and u as arguments. The corresponding discretization for $z(u)$ can be retrieved from the same function without specifying the latter three parameters.

Now suppose we have already bootstrapped the numeraire up to time t_i, starting with $i = n$, i.e. $t_n = T$ where the numeraire is of course known to be 1, independently of y, and then moving backward in i. We wish to compute

$$N(t_{i-1}, y_j) \tag{3.7}$$

for $j = m, ..., -m$ and start as indicated with $j = m$ moving down to $j = -m$. We either want to calibrate to caplets or swaptions. Without loss of generality we can restrict ourselves to the swaption case since caplets are recovered as a special case, at least in theory. In the actual implementation we have to distinguish between caplets and swaptions to a certain degree since they are represented as different instruments with different volaility structure implementations. Consider now the (time zero) price of a digital swaption with strike K and expiry $t := t_{i-1}$

$$\mathrm{dig}(K) = P(0,T)E\left(\frac{A(t)1_{S(t)\geq K}}{P(t,T)}\right) \tag{3.8}$$

Note that digital swaption here means an option paying out the Annuity

$$A(t) = \sum_{k=1}^{n} \tau_k P(t, t_k) \tag{3.9}$$

exactly on the expiry date (i.e. without settlement days) if the underlying swap rate $S(t)$ is on or above strike level K. The swap rates conventions are slightly adjusted in that the first accrual period on the fixed and float leg starts on the expiry date, i.e. there is no start delay. In particular τ_1 is not exactly consistent to the market convention, which implies e.g. two settlement days for Euribor swaps. The reason for zeroing the delay will become clear shortly.

We will now first imply the swap rate $S(t_{i-1})$ belonging to the node $y^* := y_j$. We assume that $S(t)$ is strictly monotonic in y. We can compute the digital with strike $S(t_{i-1})$ in the model by

$$\text{dig}_{\text{model}} = P(0,T)\int_{y^*}^{\infty} \frac{A(t,y)}{P(t,T)}\phi(y)dy \tag{3.10}$$

where $\phi(\cdot)$ is the standard normal density. Using 3.1 each term in 3.10 can be computed by an gaussian integral involving the numeraire $N(s)$ only for $s > t$, which by assumption is already known. This is done in the function `deflatedZerobond` by mean of a Gauss Hermite integration. The number of points is set in the parameter `gaussHermitePoints` and it is defaulted to 32.

As mentioned above we start at $j = m$ and work down to $j = -m$. Assuming we are at node j, we already have computed the digital price with the strike corresponding to y_{j+1}. To compute the digital for the strike corresponding to y_j we use the previously computed price and need only to compute the differential

$$P(0,T)\int_{y_j}^{y_{j+1}} \frac{A(t,y)}{P(t,T)}\phi(y)dy \tag{3.11}$$

For the first computation we formally set $y_{m+1} := y_\infty$, where y_∞ represents an upper cut off point approximating the integration to plus infinity.

The integrand is represented by interpolating the annuity $A(t,y)$ which is computed on the discretized grid for y with gauss hermite integration as decribed above. The interpolation is chosen to be a monotonic cubic spline with lagrange boundary condition, extrapolation to $+\infty$ is done using the last cubic polynomial in this case. Flat extrapolation can be enforced by specifying the option `ExtrapolatePayoffFlat`. The integration can also be cut to the rightmost point y_m by `NoPayoffExtrapolation`. See 3.3.3 for more details on these options.

The integral 3.11 can be computed by the following general identity for definite integrals against the gaussian density. Of course for the current interpolation with cubic spline a formula for degree 3 polynomials would be sufficient.

$$\frac{1}{\sqrt{\pi}}\int_{x_0}^{x_1} (aX^4 + bX^3 + cX^2 + dX + e)e^{-X^2}dX = \tag{3.12}$$

$$\frac{a + 2c + 4e}{8}\text{erf}(x_1) - \frac{3a + 2c + 4e}{8}\text{erf}(x_0) \tag{3.13}$$

$$-\frac{1}{4\sqrt{\pi}}e^{-x1^2}(2ax_1^3 + 3ax_1 + 2b(x_1^2 + 1) + 2cx_1 + 2d) \tag{3.14}$$

$$-\frac{1}{4\sqrt{\pi}}e^{-x_0^2}(2ax_0^3 + 3ax_0 + 2b(x_0^2 + 1) + 2cx_0 + 2d) \tag{3.15}$$

Of course we will set $X = y/\sqrt{2}$ in our applications thereby getting

$$\frac{1}{\sqrt{2\pi}}\int_{y_0}^{y_1} (a'y^4 + b'y^3 + c'y^2 + d'y + e')e^{-\frac{y^2}{2}}dy \tag{3.16}$$

This latter integral is implemented with input parameters a', b', c', d', e' which are then transformed to $a = 4a', b = 2\sqrt{2}b', c = 2c', d = \sqrt{2}d', e = e'$ internally to make use of 3.12. The name of the function is `gaussianPolynomialIntegral`. In addition it is useful to specify alternatively coefficients $a'', ..., e''$ of a polynomial in $(y - y_0)^i$ rather than in y^i, because when using e.g. cubic interpolation these coefficients can directly be retrieved from the interpolation object. The calculation

is straightforward: $a' = a'', b' = -4a''y_0 + b'', c' = 6a''y_0^2 - 3b''y_0 + c'', d' = -4a''y_0^3 + 3b''y_0^2 - 2c''y_0 + d'', e' = a''y_0^4 - b''y_0^3 + c''y_0^2 - d''y_0 + e''$. The respective function name integrating such a polynomial is `guassianShiftedPolynomialIntegral`.

Having computed the model digital price we seek the swap rate $S(t_{i-1})$ by inverting the market price of the same swaption. The digital market price is computed by replication with a gap given by the parameter `digitalGap` and defaulted to $1E-5$. The strike $S(t_{i-1})$ is then computed using a numerical zero search, more specifically the `Brent` solver using an accuracy given by the parameter `marketRateAccuracy` which is defaulted to $1E-7$.

How do we get the numeraire now from the swap rate? Write

$$S(t)A(t) + P(t, t^*) = 1 \tag{3.17}$$

where t^* denotes the time of the final payment in the swap. Remember that we assume zero start delay for the swap here, the assumption we made earlier and which is now used, because otherwise instead of 1 on the right hand side we would have $P(t, t')$ with $t' - t$ representing the delay days of the swap. This extra term would destroy the following derivation. Dividing this equation by $N(t) = P(t, T)$ gives

$$N(t) = \frac{1}{S(t)\frac{A(t)}{N(t)} + \frac{P(t,t^*)}{N(t)}} \tag{3.18}$$

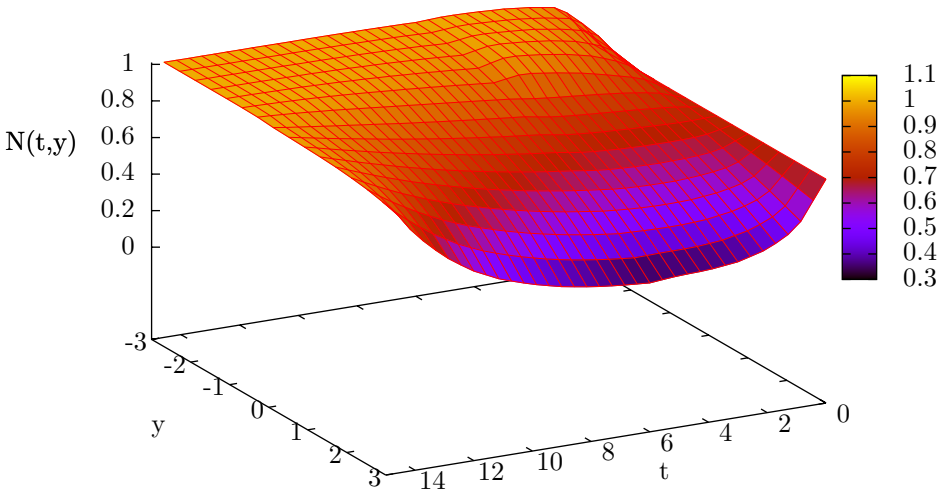
The right hand side again only contains terms which we actually already have computed above in the derivation of the swap rate. We end up with the numeraire value $N(t = t_{i-1}, y_j)$.

Iterating in an inner loop over j starting with m and going to $-m$ for each time t_i and iterating i from $i = n-1$ to $i = 1$ (because for $i = n$ and $i = 0$ the numeraire is known to be 1 and $P(0, T)$ respectively) in an outer loop gives the whole discretized numeraire surface $\{N(t_i, y_j)\}_{i,j}$. 3.2 shows a graphical representation of a numeraire surface in (y, t) space. The calibration is done using market data as of 14-11-2012 with yearly option expiries on $1y, ..., 5y$ with 10y swap underlying, no calibration to a secondary instrument set, mean reversion 0.01. In state variable direction y we display 3 standard deviations only, but merely for aesthetical reasons. The actual calibration is done with 7 standard deviations to achieve a good level of accuracy.

3.3. **Bootstrapping options.** There are some options that can be set and which influence the numeraire bootstrapping procedure. The adjusters are more for experimental purposes, it is not recommended to use them at the moment. The extrapolation options may stabilize the calibration in cases where a payoff extrapolation produces non reasonable results, although up to now no test case requiring these options is known yet to the author.

3.3.1. *Digitals adjusters.* By specifying `AdjustDigitals` the expression 3.11 is multiplied by a correction factor d. This correction factor d is computed such that the digital for the lowest strike, i.e. the strike corresponding to y_{-m}, is exactly equal to the annuity. For this two loops through j must be run, the first one to determine the correction factor and the second one to do the final computation of the swap rates belonging to the nodes y_j.

FIGURE 1. Numeraire surface for market data as of 14-11-2012



3.3.2. *Yield adjusters.* By specifying `AdjustYts` todays zero rates on the calibration instruments expiry dates are forced to be equal to the market zero rates. This is achieved by multiplying the calibrated discrete numeraire with a correction factor:

$$N(t_i, y_j) \to N(t_i, y_j) \frac{P^{\text{model}}(0, t_i)}{P^{\text{market}}(0, t_i)} \tag{3.19}$$

with $P^x, x = \text{model}, \text{market}$, denoting the zero bond implied by the model resp. the market. This option should be used with care because it may lower the accuracy of the volatility smile match. In most situations the adjustment factors are moderate though. For the same example as explored later in section 6, table 1 displays the factors needed to force the model zero rates equal to the input market zero rates.

3.3.3. *Payoff extrapolation.* The digital options priced during the numeraire calibration as well as the pricing support functions in the model class use a spline interpolated payoff (where payoff always means deflated payoff in this context). By specifiying `ExtrapolatePayoffFlat` the extrapolation of these payoffs is made flat outside the specified standard deviations for y instead of doing an extrapolation using the last polynomial of the spline at the boundaries (note that the extrapolation is done only at the relevant boundary for calls and puts in any case). The setting `NoPayoffExtrapolation` implies that the payoffs are integrated only within the specified standard deviations for y and no extrapolation at all is done.

3.4. **Second instrument set calibration.** The calibration to a second instrument set is done by setting up standard QuantLib calibration helpers and invoking the `calibrate` method on the model. In this sense, the calibration to a second instrument set is what is actually regarded as a calibration in QuantLib usually while the calibration to the first instrument set is rather a model internal construction of a numeraire surface while initialization. This is redone in every step of the calibration to a seond instrument set.

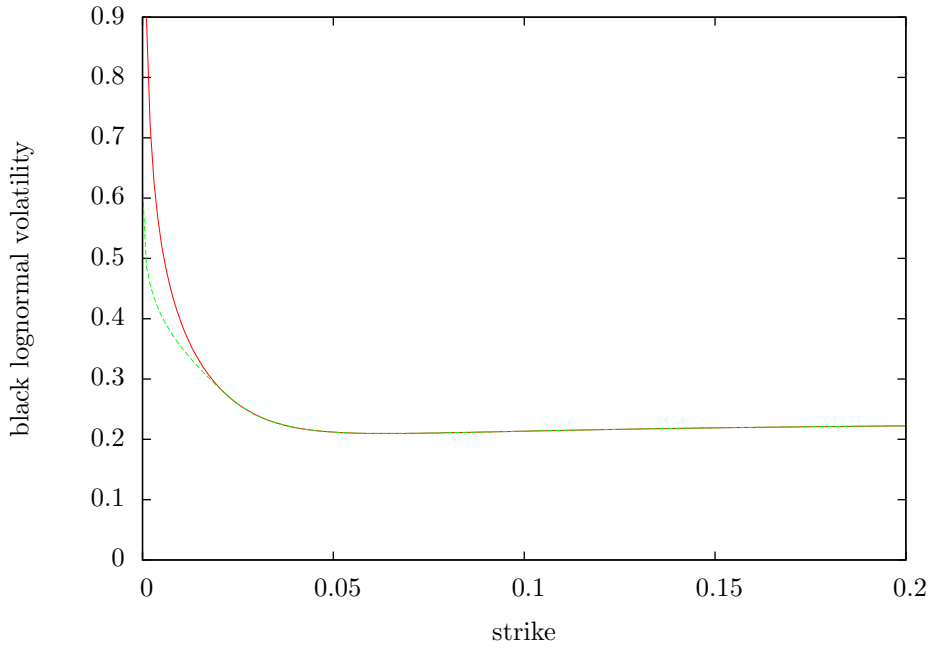
4. Smile

4.1. **Arbitrage and negative rate modelling, Smoothness.** For the numeraire calibration it is essential to have an arbitrage free smile input. In more technical terms the digital call price function must be invertible on $[0, 1]$, i.e. strictly monotonic with (non discounted) call prices of 1 for strike zero (or $-c$ with a $c > 0$, cf. below) and 0 for strike ∞. To ensure this, you have to provide a volatility structure with arbitrage free smiles for each fixed option expiry (i.e. the arbitrage condition must not necessarily be satisfied across maturities). If you can not provide such a structure you can use a standard smile pretreatment procedure within the model which checks for the maximal arbitrage free strike region of the input smile and then extrapolates the smile outside this region in a way that ensures no smile arbitrage. In the first case you have to set the model parameter `smilePretreatment` to `NoPretreatment`. In the latter case you may set this parameter to `KahaleExtrapolation`. In addition to no arbitrage it is important to have a smooth smile in the sense that it should have at least a C^1 call price function, which means continuous digital call prices, albeit not necessarily a continuous density. As a last remark note that in principle the model can adapt negative rates, if implied by the smile. The model only relies on the `digitalOptionPrice` for the numeraire bootstrap, so also technically this is possible if a suitable smile section is returned by the volatility structure. In this case the model setting `lowerRateBound` must be set to a lower rate bound $-c$ with a suitable $c > 0$. Note that if the Kahale Extrapolation is used, this implies zero probability for negative rates, cf. [1] for more details.

4.2. **Kahale Extrapolation.** The Kahale extrapolation ensures arbitrage free smiles as per fixed expiry date. To do so first the maximum arbitrage free region around atm is determined. Outside this region the arbitrage free extrapolation proposed in [1] is used. The interpolation method also proposed in the same paper is not used, instead the original smile is used. A typical scenario would be to use a standard Hagan expansion SABR smile and make it arbitrage free but cutting the arbitrageable regions (typically a region spanning $[0, k]$ for some small k) and replacing it by the arbitrage free functional forms from [1]. The implementation of this is done in `KahaleSmileSection`. See figure 2 for an example in terms of implied volatilities. The corresponding call price functions are plotted in figure 3. The adjustment to the call price to make it arbitrage free in the left strike region is rather small appearently in this example. In terms of digitals however the effect is clearly visible and this picture directly shows what goes wrong in the digital inversion when the input smile is arbitrageable, see figure 4. Some digital prices are not invertible at all while others have two different corresponding strikes. Since the Kahale extrapolation is only C^1 in the price function, the density is not continous. This can be seen in figure 5. While this kind of extrapolation is already enough to

ensure a stable calibration of the model, a smoother extrapolation would of course be desirable.

FIGURE 2. Volatility smile 14y/1y SABR input (solid) and Kahale extrapolation (dashed)



5. PRICING

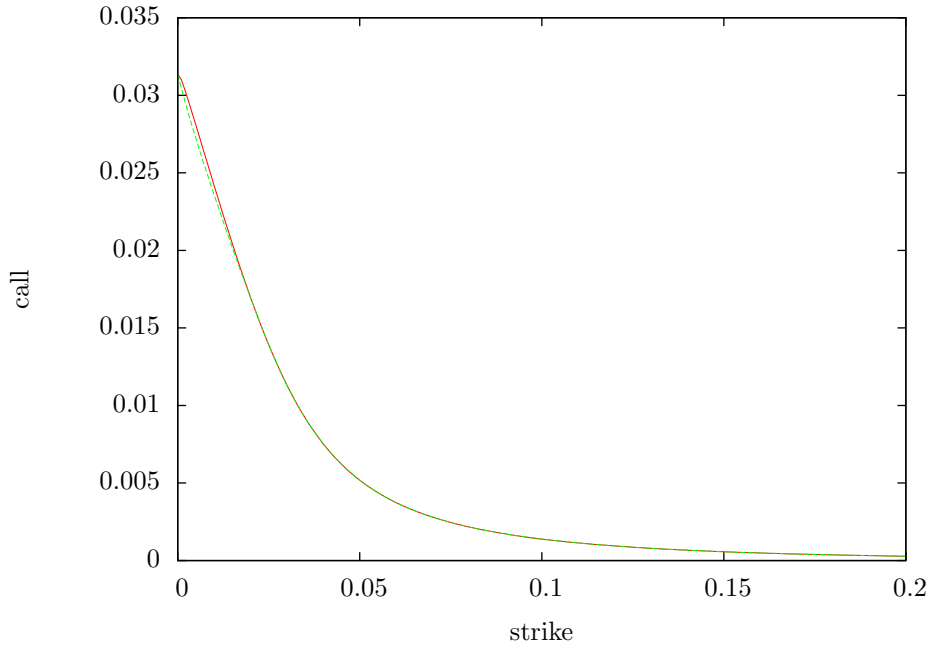
5.1. **Integral method.** It is obvious how to implement a backward pricing based on 3.4. An example is given in 5.4.1.

5.2. **Partial differential equation method.** ToDo

5.3. **Built in pricing support.** The model class supports pricing by a number of member functions which are listed below.

(1) `numeraire`: Numeraire value at (t, y).
(2) `deflatedZerobond`: Zerobond price for maturity T divided by numeraire at (t, y).
(3) `zerobond`: Zerobond price for maturity T divided by numeraire at (t, y).
(4) `zerobondOption`: Zerobond option price for maturity T at (t, y).
(5) `forwardRate`: Forward rate w.r.t. given index fixed at t_f as seen from (t, y). If `zeroFixingDays` is true the first calculation period starts at the fixing date.
(6) `swapRate`: Swap rate w.r.t. given index and tenor fixed at t_f as seen from (t, y). If `zeroFixingDays` is true the first calculation period starts at the fixing date.

FIGURE 3. Calls 14y/1y SABR input (solid) and Kahale extrapolation (dashed)

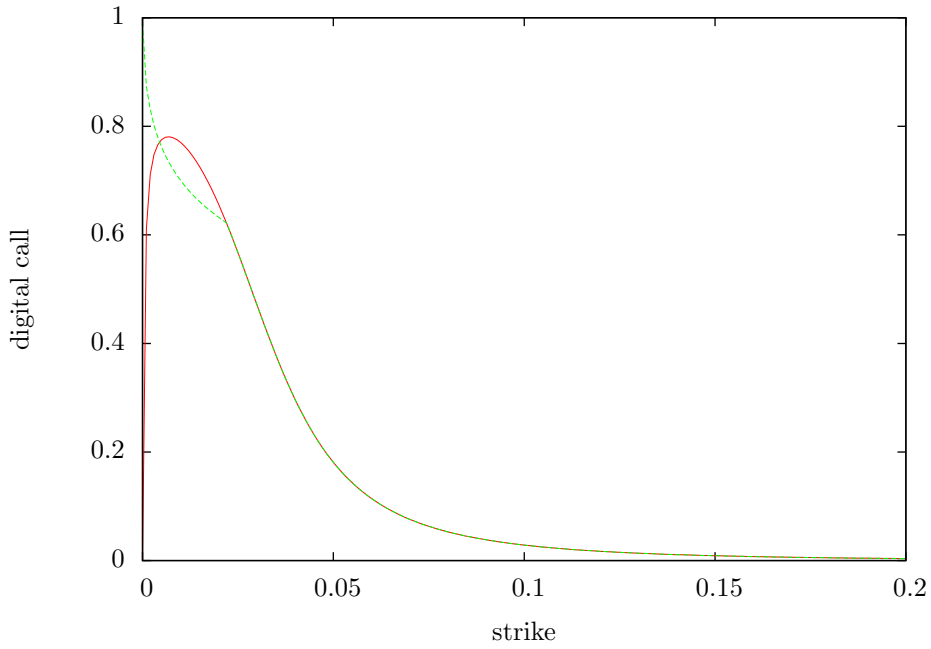


(7) `swapAnnuity`: Swap annuity w.r.t. given index and tenor fixed at t_f as seen from (t, y). If `zeroFixingDays` is true the first calculation period starts at the fixing date.
(8) `capletPrice`: Caplet price w.r.t. given index fixed at t_f as seen from (t, y). If `zeroFixingDays` is true the first calculation period of the underlying starts at the fixing date.
(9) `swaptionPrice`: Swaption price w.r.t. given index and tenor fixed at t_f as seen from (t, y). If `zeroFixingDays` is true the first calculation period of the underlying starts at the fixing date.
(10) `gaussianPolynomialIntegral`: cf. 3.12
(11) `gaussianShiftedPolynomialIntegral`: cf. the comment below 3.12
(12) `yGrid`: cf. the comment below 3.6

5.4. **Pricing Engines.**

5.4.1. *MarkovFunctionalSwaptionEngine.* This engine can handle physical settled european and bermudan swaptions and is using the integral method. No spread is allowed on the float leg at the moment. The fixed leg may however be constructed with non constant coupon rates. The engine is intended to be used in the model calibration to a second instrument calibration set, e.g. atm coterminal swaptions. The engine takes the number of standard deviations to be covered in numerical integration as well as the number of discretization points. Furthermore it may be specified that the payoff (meaning deflated payoff or more generally conditional

FIGURE 4. Digital calls 14y/1y SABR input (solid) and Kahale extrapolation (dashed)



npv in case of multiple exercise rights) is extrapolated flat or no extrapolation is done, analogous to 3.3.3. Also here in any case extrapolation is done only at the relevant boundary for calls and puts. The payoff is interpolated using monotonic cubic splines with Lagrange boundary condition.

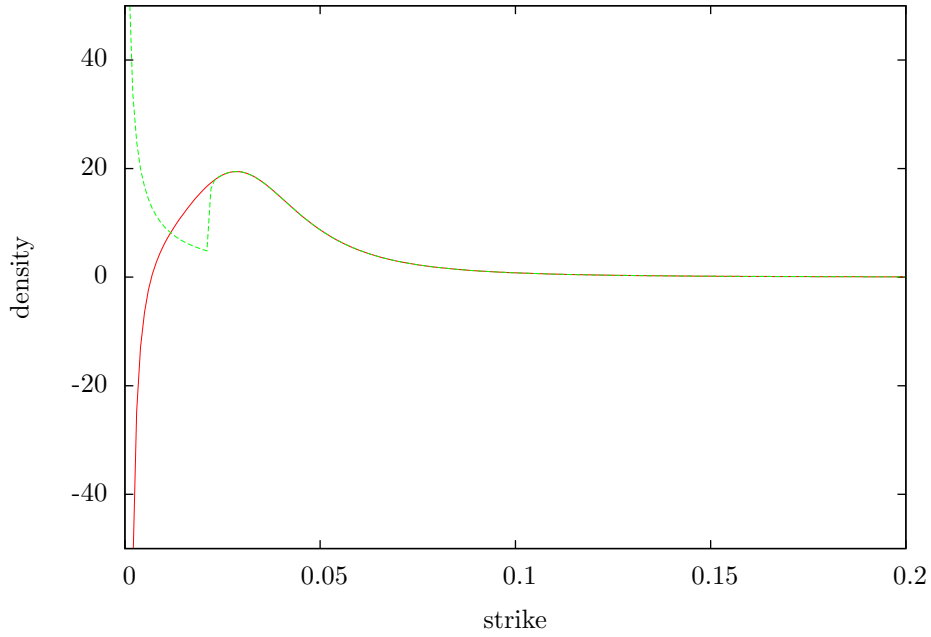
5.4.2. *MarkovFunctionalCapFloorEngine.* This engine prices plain vanilla caps, floors and collars. It may be used in case a calibration to a second instrument set with caps / floors is desired. The engine takes the number of standard deviations to be covered in numerical inttegration as well as the number of discretization points. Furthermore it may be specified that the (deflated) payoff is extrapolated flat or no extrapolation is done, analogous to 3.3.3. The payoff is interpolated using monotonic cubic splines with Lagrange boundary condition. Again extrapolation is done only at the relevant boundary for calls and puts.

5.4.3. *MarkovFunctionalFloatingSwaptionEngine.* ToDo

6. Precision and Long Term Calibration

By default `double` precision is used in all calculations within the model. For very long term calibration sets this may lead to a bad fit of the numeraire in particular for short maturities due to numerical errors. An example is given in figure 6 where a flat yield termstructure at 3% and a flat volatility surface at 20% is fitted via yearly 2y constant maturity swaptions out to 50y option expiry. As can be seen an error up to almost 10bp in the zero rate is prodcued in the short end of the model

FIGURE 5. Density 14y/1y SABR input (solid) and Kahale extrapolation (dashed)



yield term structure. The numerical parameters used for this calibrations are the default ones (32 Gauss Hermite integration points, 64 state variable grid points, 7 standard deviations, 200% upper rate bound).

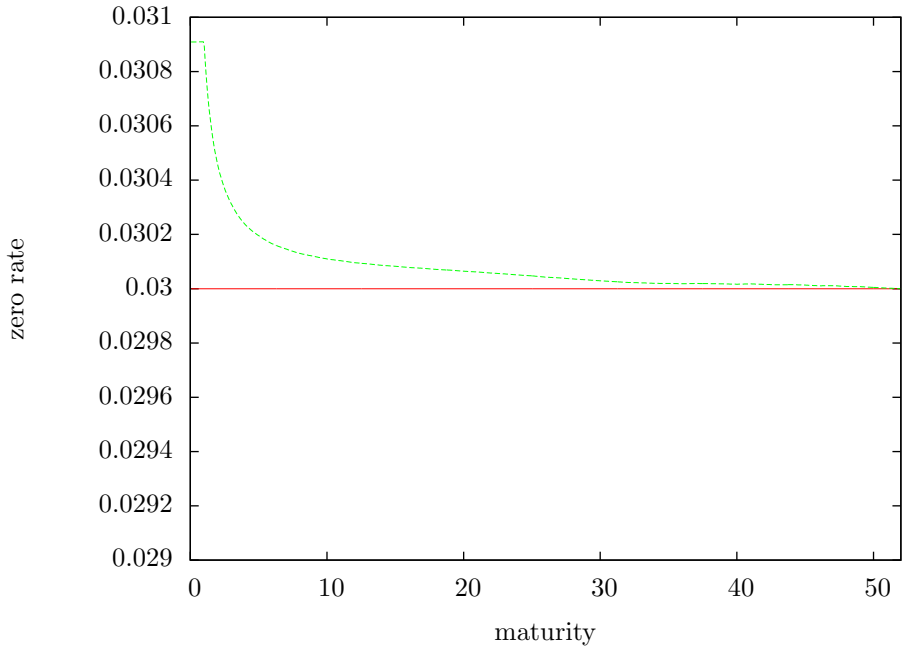
There are essentially two ways of getting a better fit: The first, more pragmatic one, is the usage of adjusters like `AdjustYts` (see 3.3.2). The second, cleaner but also computationally more demanding way is as follows:

To get a better fit one would need to increase in particular the number of standard deviations covered and the upper rate bound. However this will lead to numerical instabilities and the calibration will break down totally. It turns out that the source of the numerical problems is located mainly in the function `gaussianPolynomialIntegral`. Therefore for intermediate results within this function NTL arbitrary precision floating point numbers `RR` in connection with the boost error function implementation for `RR` can be used. For this you must enable the macro `MF_ENABLE_NTL` at the beginning of the file `markovfunctional.hpp` and provide an NTL installation on your machine, see [3] for a detailled explanation on this.

You may then set the model flag `enableNtl` to `true` and optionally prescribe the floating point precision using `boost::math::ntl::RR::SetPrecision` (which is by default 150 bit). With that we can set the model parameters to higher values like 12 standard deviations and 400% upper rate bound (as well as adjusting to 64 Gauss Hermite integration points and 128 state variable grid points) which will lead to much better numeraire fitting as can be seen in figure 7 where the error

in the zero rate is reduced to less than 0.05 bp. This is of course at the cost of much longer computation time which scales by a factor of 4-5 approximately in this example.

FIGURE 6. Long term yield term structure fit with standard precision



REFERENCES

[1] Kahale, Nabil: An arbitrage free interpolation of volatilities, Risk May 2004, http://www.risk.net/data/Pay_per_view/risk/technical/2004/0504_tech_option2.pdf or http://nkahale.free.fr/papers/Interpolation.pdf
[2] Johnson, Simon: Numerical methods for the markov functional model, Wilmott magazine, http://www.wilmott.com/pdfs/110802_johnson.pdf
[3] Shoup, Victor: NTL A Library for doing Number theory, http://www.shoup.net/ntl/
[4] QuantLib A free/open-source library for quantitative finance, http://www.quantlib.org

E-mail address, Peter Caspers: pcaspers1973@gmail.com

FIGURE 7. Long term yield term structure fit using NTL

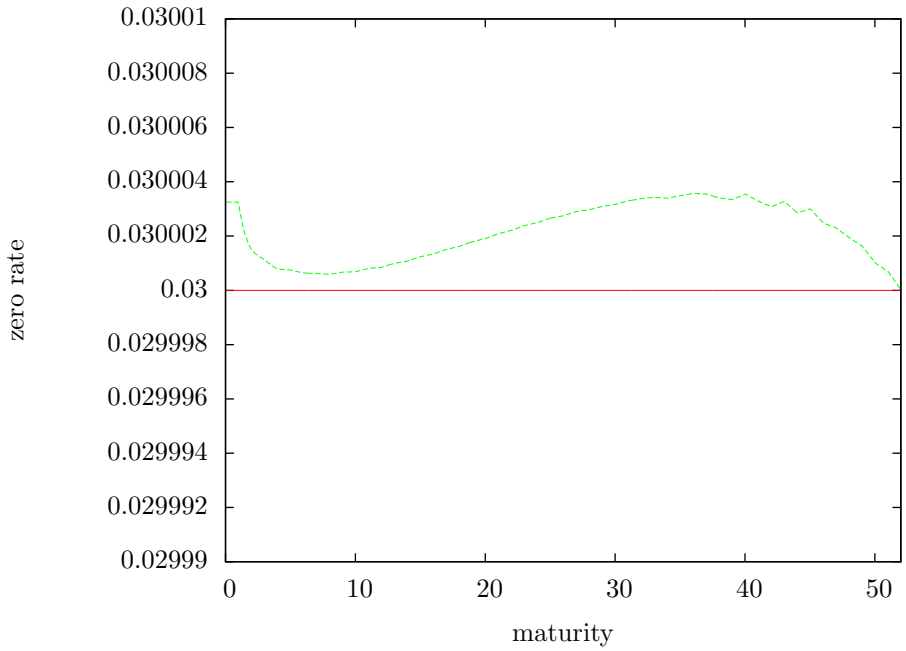


TABLE 1. Adjustment factors example

Date	Adjustment Factor
November 14th, 2013	1.00000029079227
November 14th, 2014	0.999999566861981
November 14th, 2015	0.999999720414697
November 14th, 2016	0.999999838009949
November 14th, 2017	0.999999380770489
November 14th, 2018	0.999998860452555
November 14th, 2019	0.999999397745157
November 14th, 2020	0.999999708801532
November 14th, 2021	0.99999826756268
November 14th, 2022	0.999995563493128
November 14th, 2023	0.999992108111357
November 14th, 2024	0.999988704126655
November 14th, 2025	0.999984632367841
November 14th, 2026	0.999977137924879
November 14th, 2027	0.999968677255393
November 14th, 2028	0.999959040348914
November 14th, 2029	0.999947569448016
November 14th, 2030	0.999937322893023
November 14th, 2031	0.999922851796459
November 14th, 2032	0.999903939700714
November 14th, 2033	0.999882120320225
November 14th, 2034	0.999864003916234
November 14th, 2035	0.999850090942108
November 14th, 2036	0.999850959669191
November 14th, 2037	0.99982705295748
November 14th, 2038	0.999833324843732
November 14th, 2039	0.999810600665025
November 14th, 2040	0.999854037554084
November 14th, 2041	0.999837488235502
November 14th, 2042	0.999820790446761
November 14th, 2043	0.999862504579476
November 14th, 2044	0.999899702347616
November 14th, 2045	0.999919972681322
November 14th, 2046	0.999940207031302
November 14th, 2047	1.00000230071511
November 14th, 2048	1.00000438433834
November 14th, 2049	0.999935811787155
November 14th, 2050	0.999918204985245
November 14th, 2051	0.999995605380553
November 14th, 2052	0.999965422100986
November 14th, 2053	0.999851286360178
November 14th, 2054	0.9999530097321
November 14th, 2055	0.999945359306439
November 14th, 2056	0.999804362556495
November 14th, 2057	0.999904959217797
November 14th, 2058	0.99988000473961
November 14th, 2059	0.999816723715493
November 14th, 2060	0.999830021528368
November 14th, 2061	0.999737006370401
November 14th, 2062	0.999761860227793
November 14th, 2063	0.999887598113548