CALLABLE SWAPS, SNOWBALLS AND VIDEOGAMES

Albanese, Claudio

Independent Consultant

09. September 2007

CALLABLE SWAPS, SNOWBALLS AND VIDEOGAMES

CLAUDIO ALBANESE

Abstract. Although economically more meaningful than the alternatives, short rate models have been dismissed for financial engineering applications in favor of market models as the latter are more flexible and best suited to cluster computing implementations. In this paper, we argue that the paradigm shift toward GPU architectures currently taking place in the high performance computing world can potentially change the situation and tilt the balance back in favor of a new generation of short rate models. We find that operator methods provide a natural mathematical framework for the implementation of realistic short rate models that match features of the historical process such as stochastic monetary policy, calibrate well to liquid derivatives and provide new insights on complex structures. In this paper, we show that callable swaps, callable range accruals, target redemption notes (TARNs) and various flavors of snowballs and snowblades can be priced with methods numerically as precise, fast and stable as the ones based on analytic closed form solutions by means of BLAS level-3 methods on massively parallel GPU architectures.

1. Introduction

Fixed income derivatives have had a story spanning three decades and represent the playground where many of the innovations in Mathematical Finance have been first introduced. One factor short rate models are described in (Vasicek 1977), (Ho and Lee 1986), (Hull and White 1993), (Cox *et al.* 1985) and (Black and Karasinski 1991). Multi-factor short rate models are in (Cheyette 1992), (Longstaff and Schwartz 2001), (Chen 1996) and general affine models are in (Duffie and Kan 1996) and (Dai and Singleton 2003). All these models are characterized by at least partial analytic solvability for the discount curve and caplets. The constraint of analytic solvability derives from engineering considerations and applicability to pricing purposes. However, requiring solvability lessens the main advantage of such models which is the potential proximity between the pricing measure as is revealed by derivative prices and the real world econometric estimates based on historical time series.

Market models based on an over-complete set of stochastic processes constrained by drift restrictions are in (Heath *et al.* 1992) and in (Brace *et al.* 1997), (Jamshidian 1997) and (Sandmann and Sondermann 1997). Market models are more flexible and calibrate better to swaption prices than the traditional short rate models. The implied dynamics for swaption prices and the term structure itself however do not reflect econometric evidence. This is often compensated by using a market model to decompose an exotic in elementary contracts and then applying a model such as SABR in (Hagan *et al.* 2002) to risk manage at the book level.

Date: first version: `September 9th, 2007`. Revised October 1, 2007.

I am deeply indebted with Manlio Trovato for the many discussions and work in collaboration on interest rate modeling.

The Markov functional model in (Hunt *et al.* 2000) can be regarded as a light version of a BGM model which offers a number of computational advantages as it is implementable on lattices, see (Bennett and Kennedy 2005). The technique of recombining bushy trees discussed in (Tang and Lange 2001) also offers a systematic way of building a recombining lattice models out of a BGM specification. These lattice models however have limitations as they face the challenge of having to substantially prune the number of nodes to control dimensional explosion. The industry standard for path dependent options such as target redemption notes (TARNs), callable swaps such as CMS spread range accruals, callable snowballs and TARN snowblades is currently to use market models and Montecarlo methods, combined the variance reduction techniques in (Longstaff and Schwartz 2001).

Let us recall that a CMS is a swap paying LIBOR or a fixed rate against a floating rate given by a fixed tenor swap rate prevailing at the time the coupon is paid. A TARN is a structure which terminates whenever either a maturity is reached or the total amount paid on the structured coupon leg exceeds a certain threshold. A snowball is a structured swap with a funding leg and a coupon stream whereby the coupon paid on a given date is given by the sum of a fraction of the coupon paid in the previous period plus an amount determined by the realization of the rate process in the coupon period itself. A snowblade is similar to a snowball except that it is not callable and termination is triggered by a TARN condition.

From a computational viewpoint, short rate lattice models typically execute on single threaded CPUs. Montecarlo schemes for market models are naturally parallelizable and are well suited to cluster computing implementations. The predominance of cluster computing in the last decade was one of the leading factors sustaining the predominance of market models.

In this paper, we introduce a new and flexible modeling framework for short rate models which does not rely on analytic solvability in any sense at all. As opposed to evaluating hypergeometric functions, as our main numerical engine we make use of matrix-matrix multiplication routines implemented on massively parallel multi-core architectures typical of emerging graphic processing units (GPUs). GPUs are particularly interesting from an engineering viewpoint as they are mass-produced and low cost due to the applications to video-games and computer aided design. We show that consideration of this more general class of models allows one to overcome the limitations of traditional analytically solvable models, can be made to resemble to a greater degree the econometric evidence, can be calibrated well and can used as a basis for efficient pricing of exotic structures such as callable swaps, TARNs, snowballs and snowblades.

GPU chipsets are based on arrays of single-instruction-multiple-data (SIMD) processors running at lower frequency than current CPUs but capable of processing hundreds of threads per clock cycle. Unlike CPU clusters, GPU multi-core chipsets have large blocks of fast shared memory and fast interconnects. Also unlike CPUs, GPUs are optimized for single precision arithmetics, while double precision is implemented in software and typically comes with a performance impact of about a factor 10. These architectures are particularly suitable to execute tasks that can leverage on a high degree of parallelism, fast shared memory and broad communication bandwidth such as BLAS level-3 methods and in particular large matrix-matrix multiplications. The level-3 methods we use involve time discretization steps as small as one-hour and because of this reason exhibit internal smoothing mechanisms which are not present in sparse BLAS level-2 methods or Montecarlo methods. Thanks to the internal smoothing properties, calculations can be safely carried out in single precision in their entirety. One can even safely numerically differentiate probability kernels to find sensitivities and price path-dependent options within single precision floating point arithmetics.

In this paper, we show how to structure pricing algorithms based on level-3 BLAS which execute particularly efficiently on GPUs, are robust in the native single precision and allow one to price a variety of fixed income exotics including European and Bermuda swaptions, callable constant maturity swaps (CMSs), callable CMS spread range accruals, CMS and CMS spread target redemption notes (TARNs), callable and TARN snowballs. The model can be freely specified as long as it is a Markov model for the short rate and possibly other factors under the risk neutral measure. Specification strategies can vary, one could proceed non-parametrically as well as semi-parametrically. The numerical scheme we use is based on a lattice which on current hardware can realistically admit up to 1500 sites (we use 672 in our example) and an elementary time discretization step not longer than a few hours at most. Partial explanations for the smoothing phenomenon in terms of rigorous results are in (Albanese and Mijatovic 2006) and (Albanese 2006). We find that the smallness of the time discretization step is crucial for the existence of internal smoothing and the usability of a single precision engine. Pade' approximants and implicit methods can be implemented on such architectures using mixed precision iterative methods as discussed in (Buttari *et al.* 2007). From the mathematical viewpoint, instead of analytic closed form solutions we make use of operator methods, functional calculus and algebraic manipulations, see (Albanese 2006).

The paper is structured as follows. In Section 2, we describe our working model, a 2-factor stochastic monetary policy model. This model could easily be extended by introducing a stochastic volatility term, but for simplicity's sake we don't pursue this possibility. In Section 3, we review the basic concepts behind operator methods such as path-ordered exponentials, the fast-exponentiation algorithm, the Feynman path-integral expansion, Dyson's formula, pointwise kernel convergence and diffusion smoothing. Section 4 is about the theory of Abelian processes and how this applies to path-dependent structured products. Section 5 covers moment methods. Section 6 contains a discussion of scaling laws and numerical complexity on massively parallel GPU architectures. We present strategies to price portfolios of exotics collectively and valuation strategies for callable and TARN snowball structures. Lattice models for TARNs are discussed in (Albanese and Trovato 2006) and that method is based on the theory of Abelian processes and first implemented on a CPU architecture. To our knowledge, this is the first article in the public literature where snowballs are priced by means of a lattice model implemented on a massively parallel GPU architecture.

2. A working example: A stochastic monetary policy short rate model

The numerical methods described in this paper apply in great generality to all multi-factor interest rate models based on a Markov process where one of the factor is the short rate and the representation corresponds to the risk neutral measure whereby one uses the money market account as the numeraire asset. No analytic solvability assumption is made, not even for the term structure of interest rates.

The only limitation is given by the number of sites in the discretization lattice for the short rate itself and the additional factors. To illustrate the methodology, we discuss in detail a relatively simple example with 84 discretization points for the short rate and 8 regimes, for a total of 672 lattice sites. However, lattices with up to 1500 sites are still quite realistic from the engineering viewpoint given current hardware.

Although short rate models are within range of our approach, the lattice size limitation makes it impractical to implement market models describing an over-complete set of risk factors subject to drift restrictions. The problem is not so much with the drift restriction, rather with the sheer number of forward rates in a market model which should be discretized and fit on a lattice. In this case, the technique of non-exploding bushy trees in (Tang and Lange 2001) provides an effective strategy for lattice discretization which is however quite distinct from our approach.

To take advantage of GPU platforms, we use operator methods from the start and express our model by means of a Markov generator given by a time dependent matrix of the form

$$\mathcal{L}^{\alpha}(x, a; x', a'; t) = \mathcal{D}(x, x'|a, t)\delta_{aa'} + \mathcal{T}_{\texttt{mp}}(a, a'|x, t)\delta_{xx'} \tag{1}$$

A state variable is given by a couple $y = (x, a)$ whereby x is related to the short rate by a function of the form

$$\bar{r}(x, t) \equiv r(x)\lambda(t). \tag{2}$$

The function $r(x)$ is specified initially as a regular grid extending from 0 to 50%. The variable a is a monetary policy regime indicator. In our example, $x = 0, ...83$. The function $\lambda(t)$ is piecewise constant in time and is determined iteratively starting from current time $t = 0$ in such a way to fit the term structure of interest rates, as explained below. The function $\lambda(t)$ that applies to our example is plotted in figure 2. Notice that this function does not deviate from 1 by more than the 5%. This is obtained by construction we aim at explaining the general shape of the yield curve by specifying the stochastic monetary policy dynamics as opposed to doing so by tweaking the function $\lambda(t)$.

The mean reversion term $\mathcal{D}(x, x'; |a, t)$ is a tri-diagonal matrix such that $\mathcal{D}(x, x'; |a, t) = 0$ if $|x - x'| >= 2$ and its first and second moments are given by

$$\kappa_1(\theta_1 - \bar{r}(x, t)) + \mu_a(x) \equiv \sum_{x'} \mathcal{D}_{\texttt{out}}(x; x'|a, t)(r(x') - r(x)) \tag{3}$$

$$\sigma_a^2 r(x)^{2\beta(x)} \equiv \sum_{x'} \mathcal{D}_{\texttt{out}}(x; x'|a)(r(x') - r(x))^2 \tag{4}$$

Here, $\kappa_1 = 6.25\%$ and $\theta_1 = 13\%$. The variable a takes eight values and distinguishes between 8 different monetary policies: $a = 0$ corresponds to a deflation scenario and the other states for $a > 0$ correspond to a normal monetary policy regime with rates falling, stable or rising at various rates. More precisely, the diffusion term $\mathcal{D}(x, x'; |a, t)$ is a tri-diagonal matrix such that $\mathcal{D}(x, x'; |a, t) = 0$ if $|x - x'| >= 2$ and its first and second moments are given by

$$\mu_a(x) \equiv \sum_{x'} \mathcal{D}_{\texttt{out}}(x; x'|a, t)(r(x') - r(x)) \tag{5}$$

$$\sigma^2(t)r(x)^{2\beta(x)} \equiv \sum_{x'} \mathcal{D}_{\mathtt{out}}(x;x'|a,t)(r(x')-r(x))^2 \tag{6}$$

Here, $\sigma(t) = 11.5\% + 6.5\%(1-e^{-t})$ and the functions $\mu_a(x)$ are given by the following table:

a	a=0	a=1	a=2	a=3	a=4	a=5	a=6	a=7
$\mu_a(x)$	$\kappa_0(\theta_0 - r(x))$	-75bp	-50bp	-25bp	0bp	25bp	50bp	100bp

TABLE 1.

Here, $\kappa_0 = 38\%$ and $\theta_0 = 0.25\%$. The monetary policy dynamics is chosen in such a way to be in qualitative agreement with the historical process in 1. A more refined, quantitative econometric analysis would be useful and possible here, but venturing in this direction is not the objective of this article. See (Dai and Singleton 2003) and (Ait-Sahalia *et al.* 2005) and references therein for a review of econometric studies. Finally, the function $\beta(x)$ is calibrated to the implied volatility backbone, as explained below.

The time dependent matrix $\mathcal{T}_{\mathtt{mp}}(a,a'|x,t)$ gives the transition probability rates between monetary policy regimes. This matrix is parametrized in terms of a transition probability intensity

$$p(t) = p_0 + (p_1 - p_0)e^{-\tau_p t}, \tag{7}$$

with $p_0 = 50\%$, $p_1 = 40\%$, $\tau_p = 3Y$. A mean reversion rate

$$k(t) = k_0 + (k_1 - k_0)e^{-\tau_k t}, \tag{8}$$

with $k_0 = 12\%$, $k_1 = 4\%$ and $\tau_k = 15Y$. And a monetary policy drift term

$$m(t) = m_0 + (m_1 - m_0)e^{-\tau_m t}, \tag{9}$$

with $m_0 = -15\%$, $m_1 = 1\%$ and $\tau_m = 0.5Y$. If $a > 0$, we set

$$\mathcal{T}_{\mathtt{mp}}(a,a+1|x,t) = \frac{1}{2}p(t) + (4-a)k(t) + m(t). \tag{10}$$

If $a-1>0$ then

$$\mathcal{T}_{\mathtt{mp}}(a,a-1|x,t) = \frac{1}{3}p(t) - \frac{2}{3}(4-a)k(t) - \frac{2}{3}m(t), \tag{11}$$

and if $a-2>0$

$$\mathcal{T}_{\mathtt{mp}}(a,a-2|x,t) = \frac{1}{6}p(t) - \frac{1}{3}(4-a)k(t) - \frac{1}{3}m(t). \tag{12}$$

The deflation regime is a bit special and for it we set

$$\mathcal{T}_{\mathtt{mp}}(0,3|x,t) = \mathcal{T}_{\mathtt{mp}}(0,4|x,t) = 5\% \tag{13}$$

and, if $a = 1,2,3$, we set

$$\mathcal{T}_{\mathtt{mp}}(a,0|x,t) = 10\%(4-a)\left(1 - \frac{r(x)}{r_0}\right)_+ . \tag{14}$$

Here $r_0 = 5\%$. All other matrix elements of $\mathcal{T}_{\mathtt{mp}}(a,a'|x,t)$ are zero.

We stress that this is just a working example that we choose here to show the methods and a calibration example. The mathematics and numerical analysis in the following sections would apply and extend disregard of the specific model specification. The yield curves produced by this model are in Figures 3, 4, 5, 6, 7. The projected monetary policy scenarios are given in Figure 10 and the state dependent correlations are plotted in Figures 9 and 8.

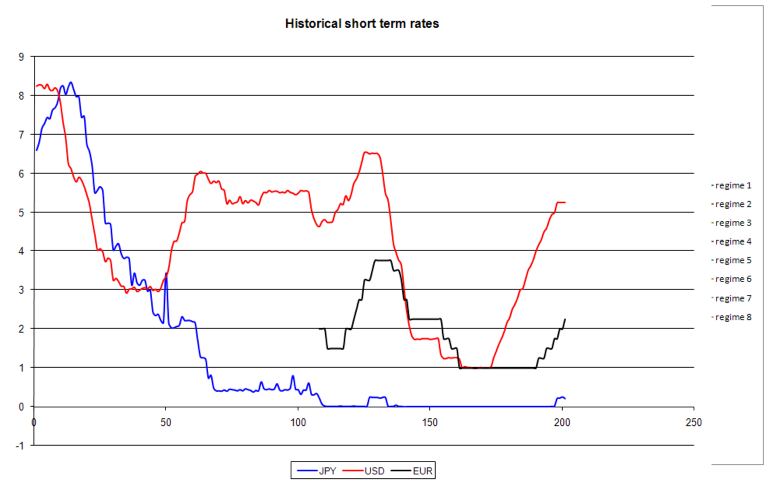

FIGURE 1. History of short rates (fund rates) for US dollar, the Euro and the Japanese Yen.

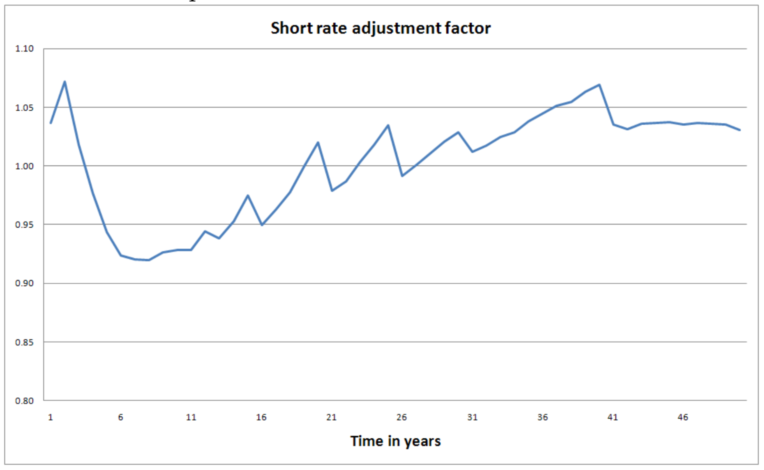

FIGURE 2. Time dependent adjustment function $\lambda(t)$ to fit the term structure of rates. Notice that this function is very close to 1. This is achieved by calibrating the drift of the monetary policy process.

3. NUMERICAL METHODS

To price interest rate derivatives under a short rate model we need to accomplish a few basic tasks. The first is to evaluate numerically the discounted propagator satisfying the equation

$$\frac{d}{dt_1}U(t_1,t_2) + (\mathcal{L}(t_1) - \lambda(t_1)r)U(t_1,t_2) = 0, \tag{15}$$

where r is the multiplication operator by the short rate grid function $r(x)$, with final time condition $U(t_2, t_2) = \mathbb{I}$, the identity operator, for all time ordered pairs $t_1 \leq t_2$. This is the propagator which is used to discount cash flows under the risk neutral measure. Having done this, we need to evaluate more general propagators carrying path information for path-dependent options.

The single-name propagator satisfying equation (15) is given by the path-ordered exponential

$$U(t_1, t_2) = \mathcal{P} \exp\left(\int_{t_1}^{t_2} \big(\mathcal{L}(s) - \lambda(s) r\big) ds \right). \tag{16}$$

There are two useful methods to express a path-ordered exponential. One is the Feynman path-integral expansion

$$\mathcal{P} \exp\left(\int_{t_1}^{t_2} \mathcal{L}(s) ds \right) = \lim_{N \to \infty} \big(\mathbb{I} + \delta t_N \mathcal{L}(t_1)\big)\big(\mathbb{I} + \delta t_N \mathcal{L}(t_1 + \delta t_2)\big) ... \big(\mathbb{I} + \delta t_N \mathcal{L}(t_N)\big) \tag{17}$$

where $\delta t_N = \frac{t_2 - t_1}{N}$. The second is Dyson's formula

$$\mathcal{P} \exp\left(\int_{t_1}^{t_2} \mathcal{L}(s) ds \right) = \sum_{n=0}^{\infty} \frac{1}{n!} \mathcal{P} \bigg(\int_{t_1}^{t_2} \mathcal{L}(s) ds \bigg)^n . \tag{18}$$

where

$$\mathcal{P} \bigg(\int_{t_1}^{t_2} \mathcal{L}(s) ds \bigg)^n = n! \int_{t_1}^{t_2} ds_1 \int_{s_1}^{t_2} ds_2 .... \int_{s_{n-1}}^{t_2} ds_n \mathcal{L}(s_1) \mathcal{L}(s_2) ... \mathcal{L}(s_n). \tag{19}$$

Dyson's formula is useful from a theoretical viewpoint. In the next section we provide examples of its use by deriving a numerical moment method to price a number of Abelian path-dependent options, ranging from risky annuities to volatility swaps and cliquets.

The Feynman path integral representation is interesting in this context as this formula can be implemented numerically very efficiently on GPU architectures by the method of fast exponentiation. The method works as follows. Assume that the dynamic generators $\mathcal{L}(t)$ are piecewise constant as a function of time. Suppose $\mathcal{L}(t) = \mathcal{L}_i$ in the time interval $[t_i, t_i + (\Delta t)_i]$. Assume δt be chosen so small that the following two conditions hold:

$$\min_{y \in \Lambda} (1 + \delta t \mathcal{L}_i(y, y)) \geq 1/2 \tag{FE1}$$

$$\log_2 \frac{(\Delta t)_i}{\delta t} = n \in \mathbb{N}. \tag{FE2}$$

This condition leads to intervals δt of the order of one hour of calendar time and this is indeed the choice we make. To compute $e^{(\Delta t)_i \mathcal{L}_i}(x, y)$, we first define the elementary propagator

$$u_{\delta t}(x, y) = \delta_{xy} + \delta t \mathcal{L}_i(y, y) \tag{20}$$

and then evaluate in sequence $u_{2\delta t} = u_{\delta t} \cdot u_{\delta t}$, $u_{4\delta t} = u_{2\delta t} \cdot u_{2\delta t}$, ... $u_{2^n \delta t} = u_{2^{n-1}\delta t} \cdot u_{2^{n-1}\delta t}$. Matrix multiplication is accomplished numerically by invoking either the single precision routine `sgemm` or the double precision routine `dgemm`. Since GPUs floating point units are natively single precision and double precision is slower by a factor 10, our advice is to use `sgemm`. Interestingly enough, we find that single precision is robust for all practical applications we considered, including the working example in this paper. In fact, the algorithm of fast-exponentiation as described above with a δt satisfying the above bound has self-smoothing properties which lessen the impact of floating point errors to be far less than what one would naively expect with a back-of-the envelope worst case estimate. The mathematical reasons behind this phenomenon appear to be deep and not well understood; see (Albanese and Mijatovic 2006) and (Albanese 2006) for some results in this direction.

Fast exponentiation per se suffices to evaluate all discount functions and to price European and Bermudan swaptions and to find the CMS convexity convections. We find that model parameters can be calibrated against at-the-money options is quite feasible by using a Levenberg-Marquardt algorithm with box bounds as the sensitivities are sharp and stable even if single precision arithmetics is used. The calibration results for at-the-money European swaptions are showed in Figures 11, 12 and 13. The skews we obtain are in 14, 15 and 16. Notice that the persistent skews are controlled by the occupancy probability of the deflation state, as the other time dependencies in the model are concentrated on short maturities. The function $\beta(r)$ is chosen in a time homogeneous way in such a way to fit the implied volatility backbone in Figures 17, 18 and 19. Notice that the function $\beta(r)$ is time homonegenous and does not help to create a persistent implied volatility skew. The Bermuda backbone in figures 20 and 21 is also an important calibration target. Finally, the CMS convexity corrections, i.e. the difference between an equilibrium CMS swap spread and the plain vanilla swap of matching maturity is given in Figures 22 and 23.

4. Abelian Processes

In (Albanese 2006) and (Albanese and Vidler 2007), we introduce the notion of Abelian processes as an extension of the notion of stochastic integral over a diffusion process, see also (Albanese and Trovato 2006) and (Albanese *et al.* 2006). Abelian processes are characterized by an algebraic commutativity condition which allows one to extend the key results of stochastic calculus such as Girsanov's theorem, Ito's lemma, the Cameron-Martin theorem and the Feynman-Kac formula.

To explain the notion of Abelian process, consider a lattice process $y_t = (x_t, a_t)$ as the one introduced in the previous section and consider a discretely valued Markov chain process m_t dependent on the path followed by y_t such that the joint process (y_t, m_t) is Markov and its generator is given by the lifted operator

$$\tilde{\mathcal{L}}(y, m; y', m'; t) = \mathcal{L}(y, y'; t)\delta_{mm'} + \mathcal{A}(m; m'|y)\delta_{yy'}. \tag{21}$$

Consider the algebra spanned by sums and products of the operators $\mathcal{A}(y)$ of matrix elements $\mathcal{A}(y)_{mm'} = \mathcal{A}(m; m'|y)$. If this algebra is commutative, i.e. if the commutators $[A(y), A(y')] \equiv A(y)A(y') - A(y')A(y)$ vanish for all pairs y, y', then the process m_t is called Abelian.

It is simple to prove that if two matrices A and B have distinct eigenvalues and commute, then they can be diagonalized simultaneously. Let u be an eigenvector of A of eigenvalue λ. Then we have that

$$BAu = \lambda Bu = ABu. \tag{22}$$

Hence, also Bu is an eigenvector of A of eigenvalue λ. Since the eigenvalues of A are assumed to be mutually distinct, Bu needs to be proportional to u, i.e. $Bu = \mu u$ for some μ. This implies that u is also an eigenvector of B of eigenvalue μ.

The set of matrices with at least two coinciding eigenvalues has zero measure in the space of all matrices. This set is a codimension-1 surface. Any neighborood around each point on this surface cointains a matrix whose eigenvalues are all distinct. Hence, if a matrix has degenerate eigenvalues, there are arbitrarily small perturbations which lift the degeneracy. One would conclude that degeneracies are a fairly unusual occurrence. However, from the numerical viewpoint one has to pay attention to the phenomenon as near degeneracies can translate in ill-conditioning. In this case, floating point errors can be amplified to give rise to numerical instabilities.

If the Abelian property is satisfied and unless near-degeneracies and ill-conditioning manifest themselves, then there exist a numerically usable non-singular linear transformation $S(m;i)$ which diagonalizes all the operators $\mathcal{A}(m;m'|y)$ simultaneously. Hence, the transformed lifted generator $S^{-1}\tilde{\mathcal{L}}S$ is a complex, block-diagonal matrix which can be fast-exponentiated by means of the BLAS routine `cgemm` which implements a complex matrix-matrix multiplication in single precision. This exponential gives the joint distribution of the pairs (y_t, m_t) for all time horizons t.

Possibly capped and floored range accruals provide an example for the application of this theory. The time a given rate spends within a certain range is in fact an Abelian process. Also the maximum achieved by a given rate over a certain period of time is an example of an Abelian process and lookback options can thus be priced by block-diagonalization.

A discrete version of the notion of Abelian process is particularly useful to price interest rate derivatives such as target redemption notes (TARNs). Consider a coupon period ΔT and a state dependent payoff which is payed at times $T_i = i\Delta T$ and is of the following general form

$$C_s(y_{i-1}, y_i, T_i) - C_f(y_{i-1}, y_i, T_i). \tag{23}$$

where $C_s(x, T_i)$ is the coupon paid on the so-called *structured leg* of the TARN and $C_f(x, T_i)$ is the coupon paid instead on the so-called *funding leg.* A TARN is characterized by a maturity $T = T_n$ for some integer $n > 0$ and a target H. One says that the target is reached on the $k-$th coupon date T_k if

$$\sum_{i=1}^{k} C_s(y_{i-1}, y_i, T_i) > H. \tag{24}$$

A TARN is structured so that cash flows continue either until maturity or at the first coupon date at which the target is reached, whichever comes first.

To price a TARN, one can consider the following lifted propagator

$$U_i(y_{i-1}, m_{i-1}; y_i, m_i) = P\exp\left(\int_{T_{i-1}}^{T_i} \mathcal{L}(t)dt\right)(y_{i-1}; y_i)K(y_{i-1}, m_{i-1}; y_i, m_i) \tag{25}$$

where

$$\begin{aligned} K(y_{i-1}, m_{i-1}; y_i, m_i) \equiv{}& p(y_{i-1}, y_i, T_i)\delta\left(m' - m - \left[\frac{C_s(y_{i-1}, y_i, T_i)}{\Delta C_s}\right]_-\right) \\ &+ \big(1 - p(y_{i-1}, y_i, T_i)\big)\delta\left(m' - m - \left[\frac{C_s(y_{i-1}, y_i, T_i)}{\Delta C_s}\right]_+\right). \end{aligned} \tag{26}$$

Here, (ΔC_s) is a discretization interval for the structured coupon dimension. If $\xi \in \mathbb{R}$, we denote with $[\xi]_+$ the smallest integer larger than ξ and with $[\xi]_-$ the largest integer smaller than ξ. Furthermore, $p(y_{i-1}, y_i, T_i)$ is chosen so that

(27)
$$p(y_{i-1}, y_i, T_i) \left[\frac{C_s(y_{i-1}, y_i, T_i)}{\Delta C_s} \right]_- + \big(1 - p(y_{i-1}, y_i, T_i)\big) \left[\frac{C_s(y_{i-1}, y_i, T_i)}{\Delta C_s} \right]_+ = \frac{C_s(y_{i-1}, y_i, T_i)}{\Delta C_s}.$$

Let $K(y_{i-1}, y_i)$ be the operators of matrix elements

$$K(y_{i-1}, y_i)_{m_{i-1} m_i} = K(y_{i-1}, m_{i-1}; y_i, m_i). \tag{28}$$

The operators $K(y_{i-1}, y_i)$ commute with each other for all choices of pairs (y_{i-1}, y_i). Hence, in most cases and assuming no ill-conditioning situations arise, they can be simultaneously diagonalized.

To complete the definition of the operator $K(y_{i-1}, m_{i-1}; y_i, m_i)$ in a usable way, we need to limit the range of variability of the variable m to a finite interval $m = 0, 1, ...M$ for some $M > 0$ and specify boundary conditions for the process as it reaches the site $m = M$. A useful strategy is to impose periodic boundary conditions so that upon surpassing M, the process winds back and re-enters from the point $m = 0$, as if the interval $[0, M]$ was wrapped around a circle. With this choice, the situation is simple as the block-diagonalization can be accomplished in all cases by means of an expicit partial Fourier transform whose implementation does not give rise to ill-conditioning. Let $\mathcal{F}$ be the operator such that

$$\hat{u}(y, p) = (\mathcal{F}u)(y, p) = \sum_{m=0}^{M} e^{-ipm} u(y, m), \tag{29}$$

where $p = \frac{2\pi j}{M+1}, j = 0, 1, ...M$. Then we have that the transformed operator

$$\hat{U}_i = \mathcal{F} U_i \mathcal{F}^{-1} \tag{30}$$

is block-diagonal and has matrix elements

(31)
$$\hat{U}_i(y_{i-1}, p_{i-1}; y_i, p_i) = \delta_{p_{i-1} p_i} P \exp\left(\int_{T_{i-1}}^{T_i} \mathcal{L}(t) dt \right)(y_{i-1}; y_i) \sum_{m=0}^{M} K(y_{i-1}, 0; y_i, m) e^{i p_i m}.$$

We conclude that the joint distribution of the discrete time process m_{T_i} and the underlying process y_{T_i} at time T_i conditioned to starting at y_0 and $m_0 = 0$ at time 0 is given by

$$U_i(y_{i-1}, m_{i-1}; y_i, m_i) = \frac{1}{M+1} \sum_{j=0}^{M} e^{\frac{2i\pi j (m_i - m_{i-1})}{M+1}} \left(\prod_{j=0}^{i} \hat{U}_j(p) \right)(y_0, y_{T_i}) \tag{32}$$

where $\hat{U}_j(p)$ is the operator of matrix elements $\hat{U}_j(p)(y, y') = U(y, p; y', p)$.

Notice here one of the differences between probability theory expressed via stochastic calculus as opposed to our formalism based on operator methods: in the former one focuses on measure changes which are given by one parameter families of positive linear transformations, while in our case we can make full use of general complex valued linear tansformations as the applicability of operator methods does not hinge on the existence of a probabilistic interpretation for the dynamic generators.

5. Moment Methods

In addition to applying block-diagonalization techniques, for most Abelian payoffs one can also use a moment method based on Dyson's expansion. The method applies whenever one needs to evaluate moments of stochastic integrals over a bridge, such as

$$J(y,y') \equiv E_0\left[\left(\int_0^T \phi(y_t,t)dt\right)^m \delta(y_{T'}-y')\middle| y_0=y\right]. \tag{33}$$

Here m is an integer and $[0,T]$ is a given interval. In this case we have that

$$J(y,y') = \left(\frac{d^m}{d\epsilon^m}\right)\bigg|_{\epsilon=0} P\exp\left(\int_0^T \mathcal{L}_\epsilon(t)dt\right)(y,y') \tag{34}$$

where

$$\tilde{\mathcal{L}}_\epsilon(t)(y,y') = \mathcal{L}(y,y';t) + \epsilon\phi(y;t)\delta_{yy'}. \tag{35}$$

For instance, to evaluate a range accrual where each coupon payment is proportional to the time a given swap spread $s_t^1 - s_t^2$ is in the interval $[L,U]$, one can use the formula above with

$$\phi(y;t) = 1\big(s^1(y;t) - s^2(y;t) \in [L,U]\big). \tag{36}$$

If the range accrual is capped and/or floored the problem is complicated by the fact that in general knowledge of the first moment is not sufficient and all moments are required to reconstruct the probability distribution function for the time elapsed in the given range. However, also in this case one can take advantage of the simplicity of moment method by using an ansatz to extrapolate the form of the distribution from the knowledge of its first two or four moments. As an ansatz distribution one can use for instance either the log-normal or the chi-squared distribution. In general, it is possible to verify the quality of the ansatz by using the block-diagonalization method in the previous section, as range accruals are based on a continuous time Abelian process.

Similarly, discrete payoffs such as TARNs can also be priced approximately by means of moment methods. Consider a moment of a stochastic sum over a bridge of the form

$$F(y,y') \equiv E_0\left[\left(\sum_{j=1}^{i} \phi_j(y_{T_{j-1}},y_{T_j})\right)^m \delta(y_{T_i}-y')\middle| y_0=y\right] \tag{37}$$

where m is an integer and $[0,T_1,....T_i]$ is a sequence of time points. In this case we have that

$$F(y,y') = \left(\frac{d^m}{d\epsilon^m}\right)\bigg|_{\epsilon=0}\left(\prod_{j=1}^{i} U_\epsilon(T_{j-1},T_j)\right)(y,y') \tag{38}$$

where the matrix elements of the operators $U_\epsilon(T_{j-1},T_j)$ are given by
(39)

$$U_\epsilon(y,T_{j-1};y',T_j) = P\exp\left(\int_{T_{j-1}}^{T_j} \mathcal{L}(t)dt\right)(y,y')\exp\left(\epsilon\phi_j(y_{T_{j-1}},y_{T_j})\right)(y,y').$$

When pricing a TARN instead, one needs to proceed by iteration going forward in time. In this case, at every step one needs to apply the kernel to a probability distribution function $\psi(y,T_{j-1})$ which is initialized to have unit mass at the initial point $y = y_0$. The function $\phi_j(y_{T_{j-1}}, y_{T_j})$ in the kernel is the $j-$th coupon function. At each date, one then numerically differentiates with respect to ϵ at least twice and evaluates at $\epsilon = 0$. This provides the two first moments of the distribution of the total accrued by the structured leg on a give bridge originating from the spot and allows one to reconstruct the probability distribution function and the survival probability conditioned on a bridge. This survival probability needs to be applied to each coupon amount which then has to be discounted to current time using the discounted transition probability kernel.

An application of moment methods is given in Figures 25, 26, 27, 28, 29, 30, 31 where we show the pricing functions of callable CMS spread range accruals.

6. Scaling of algorithmic complexity, portfolios and snowballs

When using GPUs it is important to familiarize ourselves with fairly counter-intuitive scaling properties for the run time required by various tasks, as this is not proportional to the traditional concept of algorithmic complexity intended as the total number of floating point operations. Since matrix-matrix multiplications are distributed across a large number of cores, the larger are the matrices one multiplies, the better is the hardware performance. In this section we show how these concepts apply to portfolio valuations and pricing of snowballs and snowblades.

Consider pricing a large portfolio of European or callable instruments and suppose one has available in memory all the required discounted transition probability kernels and data structures that describe cash-flows. In this case the best strategy is to organize the pricing functions for all the instruments in the portfolio in a large matrix, attributing one column vector to each instrument, and perform the backward induction collectively on the entire portfolio. The alternative strategy of iterating on each instrument separately is extremely more time expensive.

Task	**GPU-O**	**GPU-D**	**Host-O**	**Ratio**
Initialization	4.79	5.16	3.77	0.79
Calibration to term structure of rates	4.76	6.14	61.88	13.00
585 European swaptions	7.62	10.17	85.30	11.19
Portfolio of CMS spread range accruals: 9 callable swaps and 3 callable snowballs	21.03	23.74	134.59	6.40
30240 ATM European swaptions	11.34	16.16	138.68	12.23
13725 ATM European swaptions and 13725 ATM Bermuda swaptions	56.56	59.02	272.19	4.81

Table 2. Execution times in seconds for various portfolios under various configurations. **GPU-O**: using the GPU with host side optimized code. **GPU-D**: using the GPU with host side debug code. **Host-O**: using the host only with optimized code and Intel MKL libraries. **Ratio**: column 3 versus column 1.

Table 2 illustrates performance with examples of portfolios, also comparing the performance of optimized code versus debug code. The code for this example is written almost entirely in VB.NET, the CPU is a 2 GHz Xeon and the GPU is a card Nvidia 8800 GTX originally intended for the games market, the API to the GPU being given by the CUDA-BLAS library. One can notice that the debug code runs only 5% slower than optimized code. The code is entirely managed but by extrapolation one would conclude that the speed up achievable by using native C code as opposed to VB.NET would also be negligible. The computational bottleneck is in fact captured by calls to `sgemm` executing on the GPU.

One can notice that while a portfolio of 585 European swaption takes 7.62 seconds to price, a portfolio with 30,240 swaptions takes only 11.34 seconds. Execution time thus scales in a very non-linear way with respect to the number of floating point operations. This is achieved by organizing all pricing functions of the portfolio as column vectors in a matrix and applying backward induction to the portfolio in aggregate. The alternative strategy of pricing each swaption individually would give rise to linear scaling and be much more time expensive. However, by multiplying a large kernel matrix with a large payoff matrix, the load is partitioned more evenly across the cores, each one handling the task of multiplying sub-blocks, and as a consequence the performance is greatly enhanced.

Notice that the performance gain with respect to an application running on the host varies. The initialization is slightly slower when using a GPU due to memory allocation on the card. The calibration to the term structure of interest rates shows a gain by a factor 15. The pricing of European swaptions shows speedups by a factor 11-12, with a slightly better performance for the larger portfolio. The speedup for the portfolios containing callables is not as great and is only 4.81 for the portfolio with 13,725 Bermuda swaptions. The reason for this is that the implementation was designed in such a way that the exercise condition was checked on the host and at each exercise date a portion of the portfolio was moved across the bus. This can easily be avoided by checking the exercise condition with a special function written in C and executing on the GPU side. By eliminating the bus overhead, the performance gain is again in the range 11-12.

Another instance where counter-intuitive scaling is revealed and can be taken advantage of is the pricing of snowballs and snowblades. A snowball is a callable swap whose structured coupon at time T_i has the following form

$$C_{T_i} = kC_{T_{i-1}} + \Phi_i(y_{T_{i-1}}, y_{T_i}) \tag{40}$$

where k is a fixed parameter. The difficulty with snowballs is that the coupon process is not Abelian, in fact it is path-dependent. Hence, the methods in the two previous sections are not applicable to reduce dimensionality. However, using multi-core scaling it is possible to setup a very efficient pricing scheme for snowballs.

Callable snowballs are priced by backward induction while snowblades are priced by forward induction using either block-diagonalizations or, for greater efficiency, the moment method. Consider first callable snowballs. In this case, one needs to discretize the coupon dimension in intervals ΔC, so that a generic coupon can be approximated as follows $C_{T_i} = (\Delta C)n_{T_i}$ where $n_{T_i} = 0, 1, ...N - 1$ is a discrete time, integer value process. In the working example we consider here we choose $N = 100$. A strategy to implement the backward induction scheme for a single callable snowball is to organize the payoff function at maturity in a matrix with N columns, each one indexed by the state variable y and representing the price conditional to the discretized value of the previous coupon paid. One can then iterate backward by applying the pricing kernel to this matrix of conditional pricing functions. After each step in the iteration, one needs to reshuffle the pricing functions in such a way that the conditioning relation on each column is satisfied. When pricing a portfolio of European or callable snowballs there is obviously a great advantage in pricing them all together.

Pricing functions for callable snowballs are given in Figures 32, 33, 34.

When pricing a TARN snowball instead one needs to go forward in time and obtain the survival probability distributions $\psi_j(y, n)$ on a bridge starting from the spot and ending at y at time T_j. This requires obtaining the joint probability that at time T_j the site y is occupied and the coupon paid at time T_{j-1} is approximately $(\Delta C)n$. In this case, at every step one needs to multiply the $n - th$ column by $e^{kn(\Delta C)}$ and then apply the kernel to all columns collectively where the function $\phi_j(y_{T_{j-1}}, y_{T_j})$ is the $j-$th coupon function. At each date, one then numerically differentiates with respect to ϵ at least twice and based on the outcome, reconstruct the probability distribution function for the total accrued on the structured leg and the survival probability conditioned on a bridge. This survival probability needs to be applied to the coupon amount which then has to be discounted to current time. Also in this case, on massively parallel multicore architectures one obtains major speedup ratios by pricing portfolios of TARN snowballs in one single iteration.

7. Conclusions

We have shows that short rate models specified semi-parametrically or even non-parametrically are viable from the engineering viewpoint as frameworks within which to price exotic single name interest rate derivatives. We find that the recently introduced GPU platforms which are being mass-produced because of applications to the video-game and computer-aided-design markets are an ideal hardware environment for such modeling approaches. We find that BLAS level-3 methods for which the speed-up ratios are more significant are stable under single precision arithmetics and that sensitivities are also smooth. The mathematical framework that supports such analysis is based on operator methods, fast exponentiation and the theory of Abelian processes. It is also based on techniques to exploit the highly non-linear scaling behavior observed on massively parallel multi-core architectures between the nominal algorithmic complexity and execution time. The methods presented here can possibly be extended to cross-currency derivatives and hybrids by modeling correlations by means of dynamic conditioning, as we have already demonstrated in the case of unmanaged CDOs in (Albanese and Vidler 2007), but we reserve to discuss these extensions in a future article.

References

Ait-Sahalia, Y., Hansen L.P. and J. A. Scheinkman (2005). Operator Methods for Continuous Time Markov Processes. *Journal of Economic Theory.*

Albanese, C. (2006). Operator Methods, Abelian Processes and Dynamic Conditioning. *preprint, available at www.level3finance.com.*

Albanese, C. and A. Mijatovic (2006). Convergence Rates for Diffusions on Continuous-Time Lattices. *preprint, available at www.level3finance.com.*

Albanese, C. and A. Vidler (2007). A Structural Model for Bespoke CDOs. *Willmott Magazine.*

Albanese, C. and M. Trovato (2006). A Stochastic Volatility Model for Callable CMS Swaps and Translation Invariant Path Dependent Derivatives. *preprint, available at www.level3finance.com.*

Albanese, C., H. Lo and A. Mijatovic (2006). Spectral Methods for Volatility Derivatives. *preprint.*

Bennett, M.N. and J.E. Kennedy (2005). Common Interests. *Finance and Stochastics.*

Black, F. and P. Karasinski (1991). Bond and Option Pricing when Short Rates are Lognormal. *Financial Analysts Journal* pp. 52–59.

Brace, A., D. Gatarek and M. Musiela (1997). The Market Model of Interest Rate Dynamics. *Mathematical Finance* **7**, 127–154.

Buttari, A., J. Dongarra, J. Kurzak, J. Langou, P. Lusczek and S. Tomov (2007). Exploiting Mixed Precision Floating Point Hardware in Scientific Computations. *preprint, available at www.netlib.org.*

Chen, Lin (1996). *Interest Rate Dynamics, Derivatives Pricing, and Risk Management.* Springer.

Cheyette, O. (1992). Term Structure Dynamics and Mortgage Valuation. *Journal of Fixed Income* **1**, 28–41.

Cox, J.C., J.E. Ingersoll and S.A. Ross (1985). A theory of the term structure of interest rates. *Econometrica.* **53**, 385–407.

Dai, Q. and K. Singleton (2003). Term Structure Dynamics in Theory and Reality. *Review of Financial Studies* **16**, 631–678.

Duffie, D. and R. Kan (1996). A Yield-Factor Model of Interest Rates. *Mathematical Finance* **6**, 379–406.

Hagan, P., D. Kumar, A. Lesniewski and D. Woodward (2002). Managing Smile Risk. *Willmott Magazine* pp. 84–108.

Heath, D., R. Jarrow and A. Morton (1992). Bond Pricing and the Term Structure of Interest Rates: A New Methodology. *Econometrica* **60**, 77–105.

Ho, T.S.Y. and S.B. Lee (1986). Term Structure Movements and Pricing Interest Rate Contingent Claims. *Journal of Finance.*

Hull, J. and A. White (1993). One-factor Interest Rate Models and the Valuation of Interest Rate Derivative Securities. *Journal of Financial and Quantitative Analysis* **28**, 235–54.

Hunt, P.J., J.E. Kennedy and A. Pelsser (2000). Markov Functional Interest Rate Models. *Finance and Stochastics* **4**, 391–408.

Jamshidian, F. (1997). Libor and Swap Market Models and Measures. *Finance and Stochastics* **1**, 293–330.

Longstaff, F.A. and E.S. Schwartz (2001). Valuing American Options by Simulation: a Simple Least Squares Approach. *Rev. Financial Stud.* **14**, 113–148.

Sandmann, K. and D. Sondermann (1997). A Note on the Stability of Lognormal Interest Rate Models and the Pricing of Eurodollar Futures. *Mathematical Finance* **7**, 119–125.

Tang, Y. and J. Lange (2001). A Non-Exploding Bushy Tree Technique and its Application to the Multifactor Interest Rate Market Model. *Computational Finance.*

Vasicek, O. A. (1977). An Equilibrium Characterization of the Term Structure. *Journal of Financial Economics* **5**, 177–88.

Claudio Albanese
E-mail address: claudio@level3finance.com

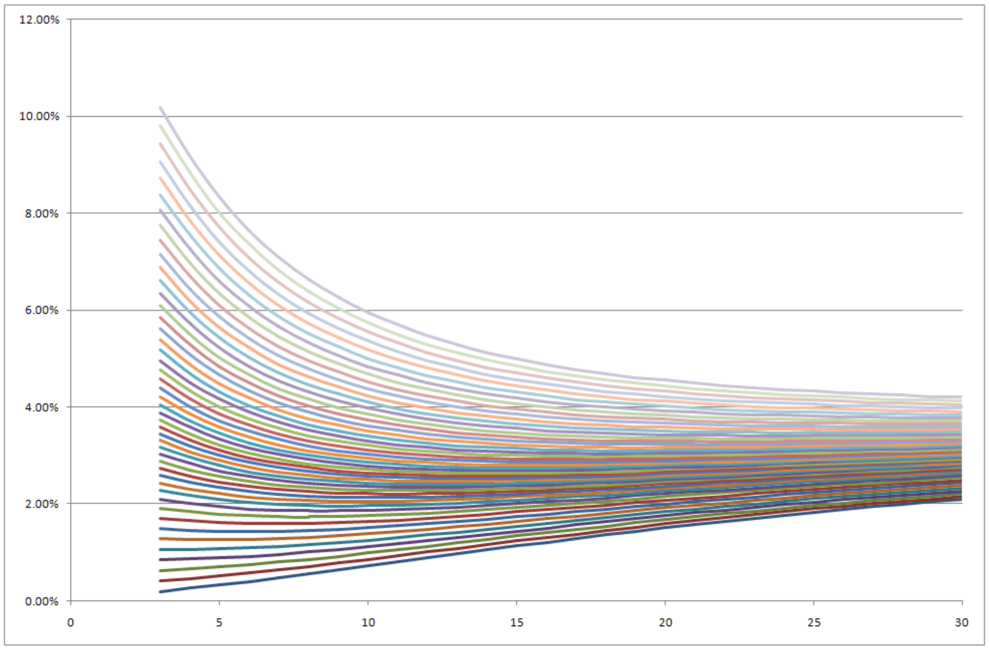

Figure 3. Zero curves in the deflation regime.

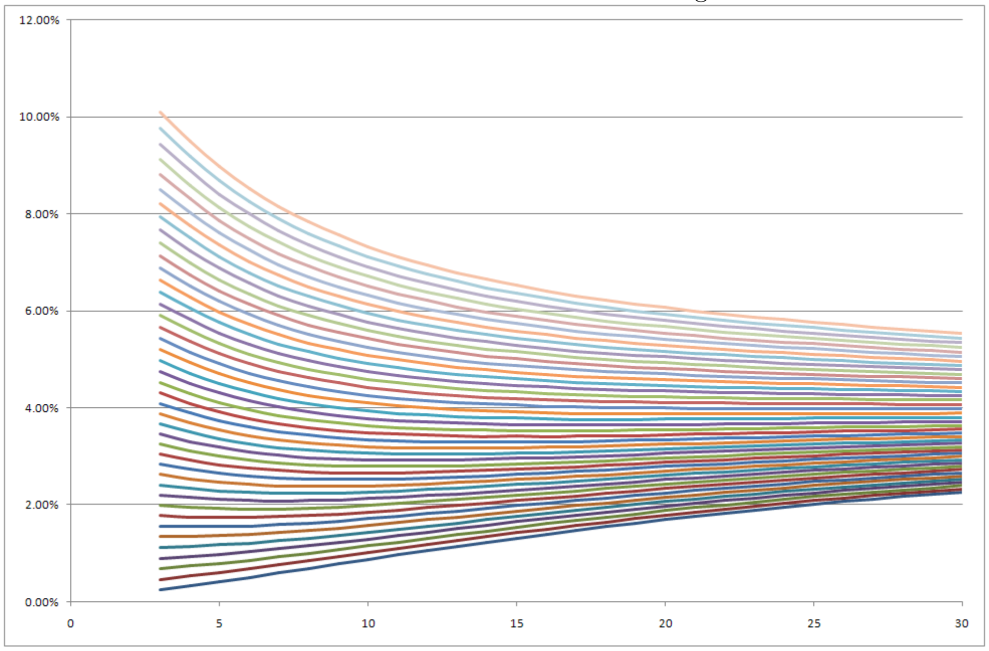

Figure 4. Zero curves in the regime with drift -75 bp/year.

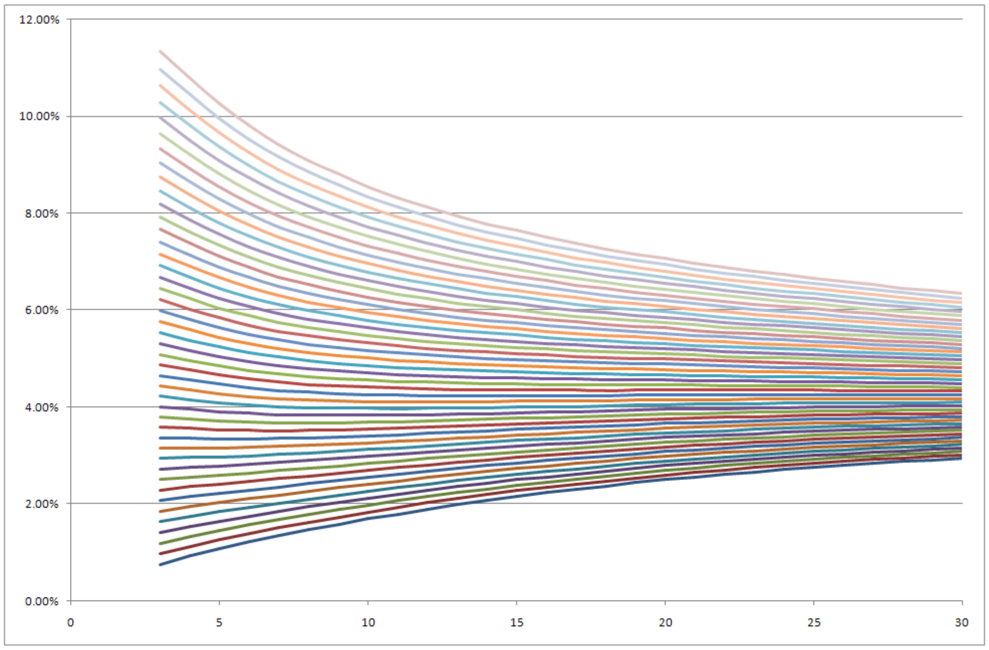

Figure 5. Zero curves in the regime with drift -25 bp/year.

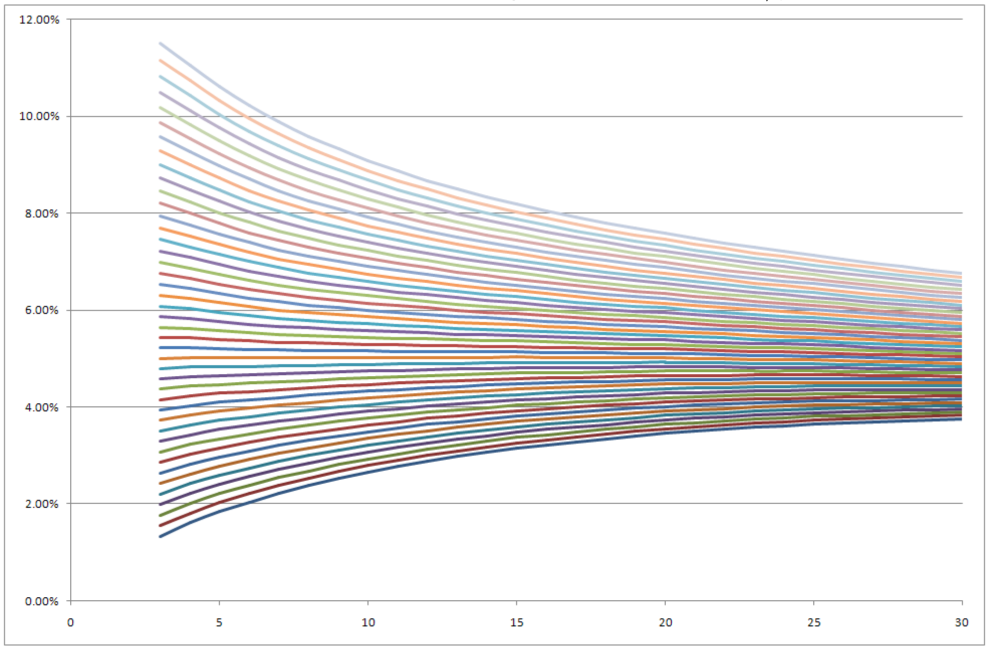

Figure 6. Zero curves in the regime with drift +25 bp/year.

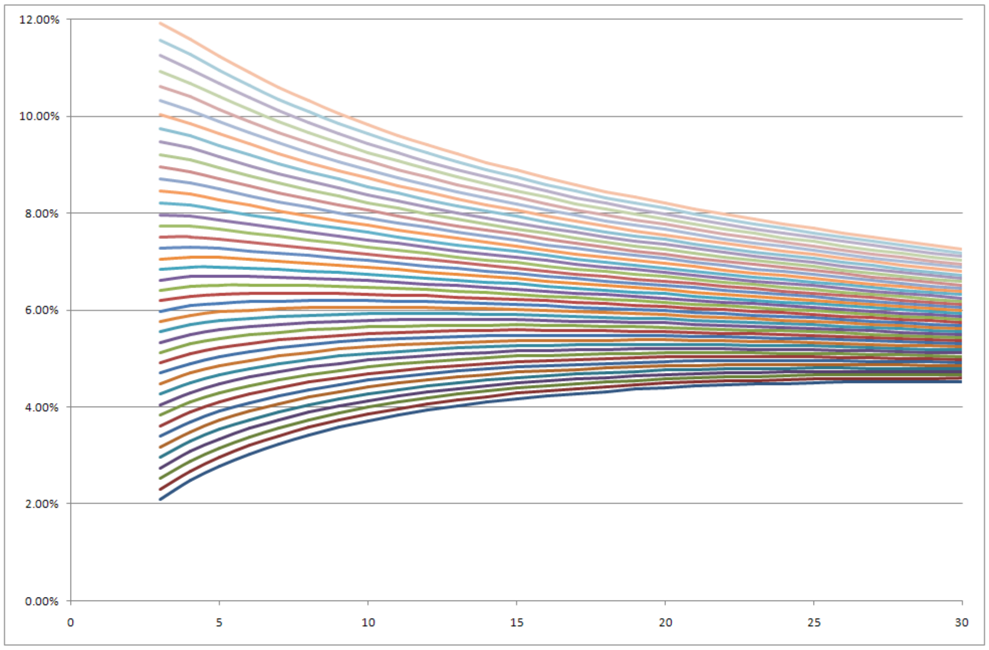

FIGURE 7. Zero curves in the regime with drift +100 bp/year.

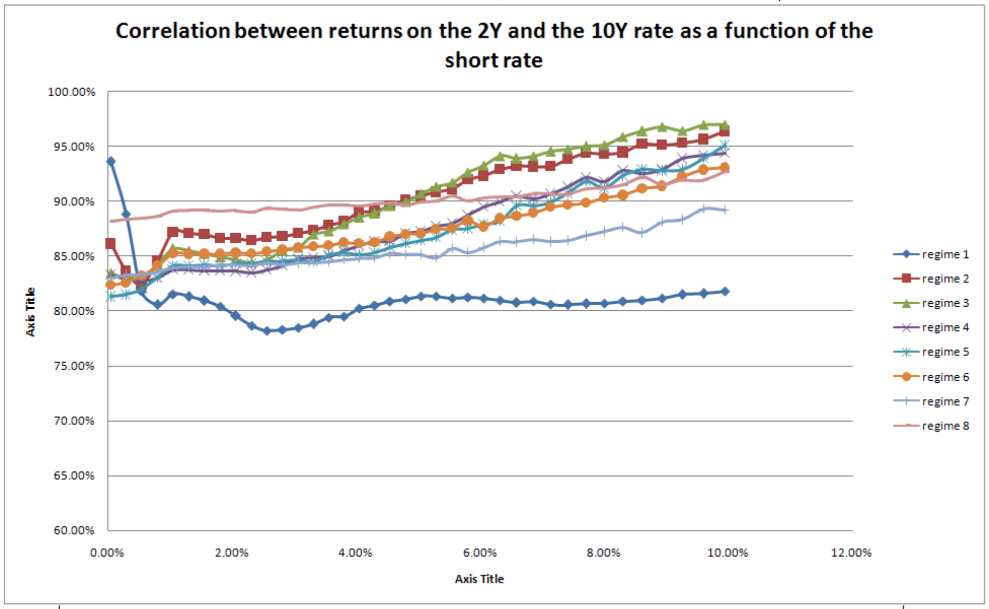

FIGURE 8. Backbone of 2y-10Y correlation, i.e. correlation between daily returns of the 2Y swap rate versus the 10Y swap rate as a function of the short rate.

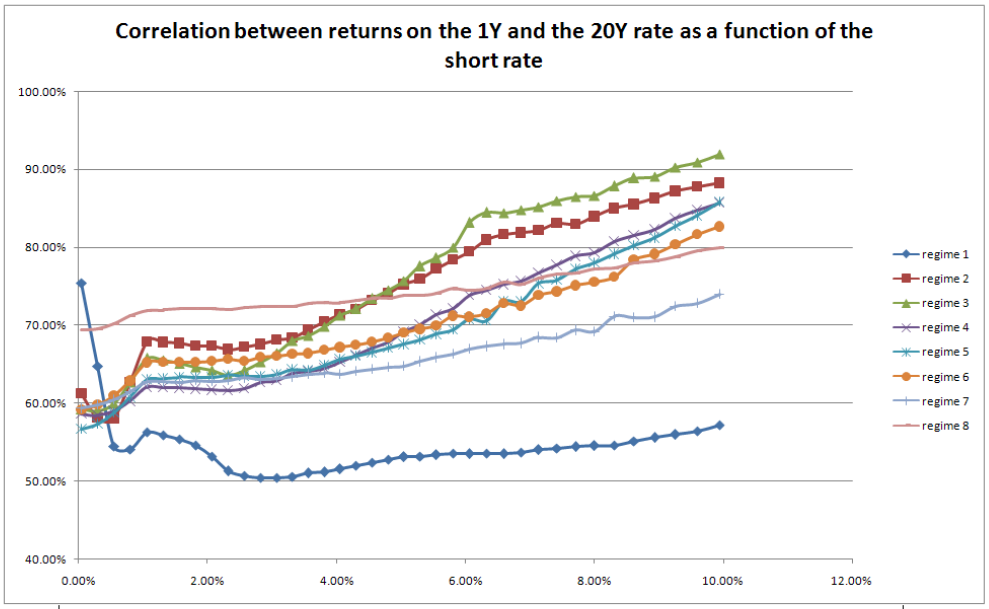

FIGURE 9. Backbone of 1y-20Y correlation, i.e. correlation between daily returns of the 1Y swap rate versus the 20Y swap rate as a function of the short rate.

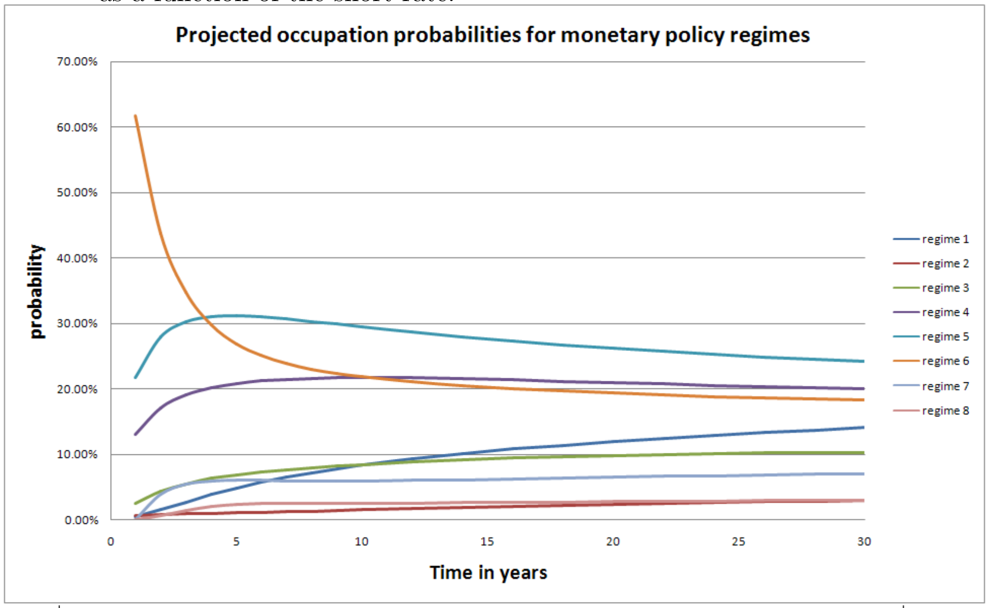

FIGURE 10. Projected occupancy probabilities of monetary policy regimes.

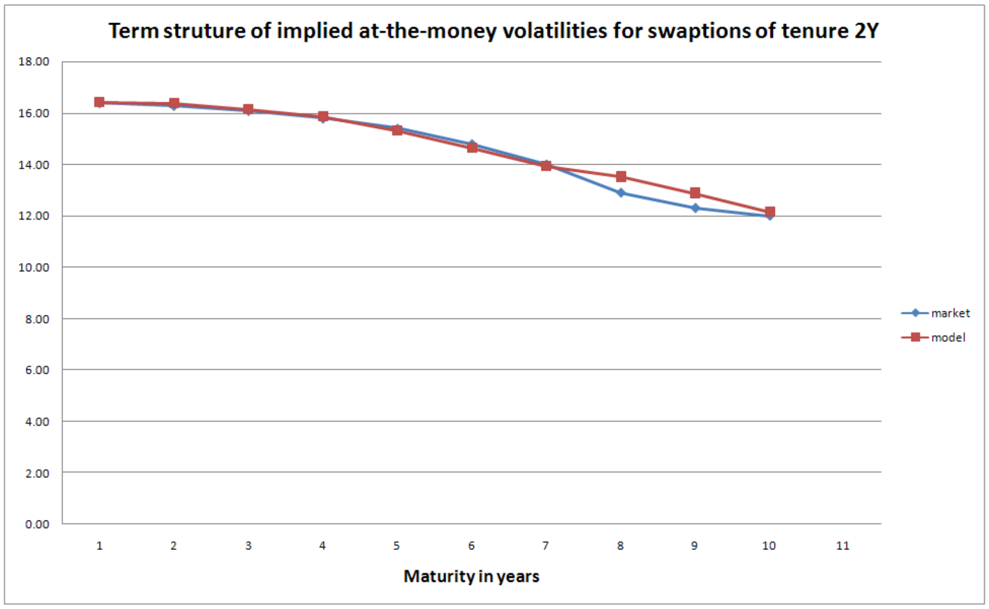

FIGURE 11. Implied volatilities for at-the-money European swaptions of tenor 2Y compared to market data.

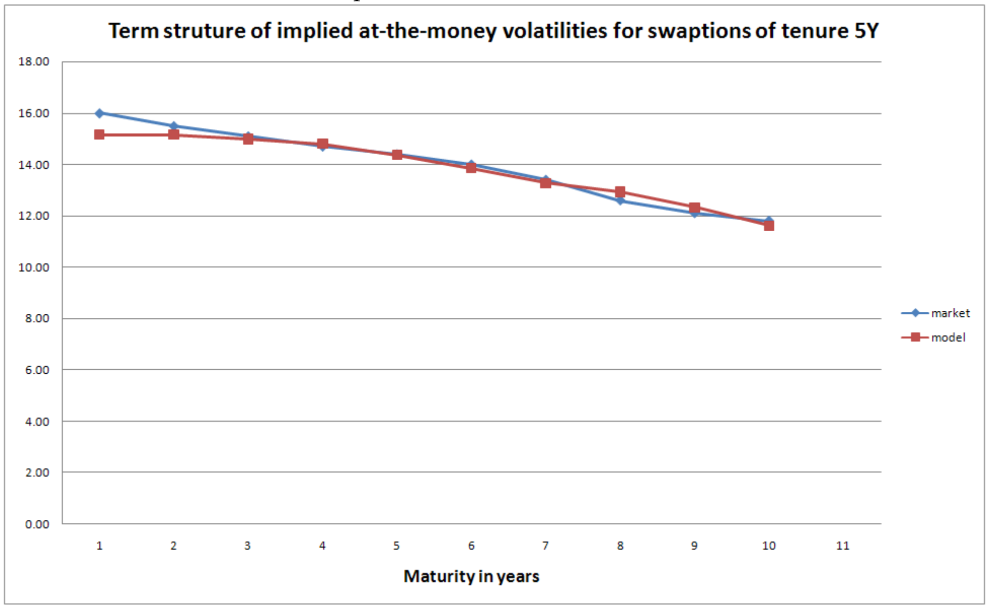

FIGURE 12. Implied volatilities for at-the-money European swaptions of tenor 5Y compared to market data.

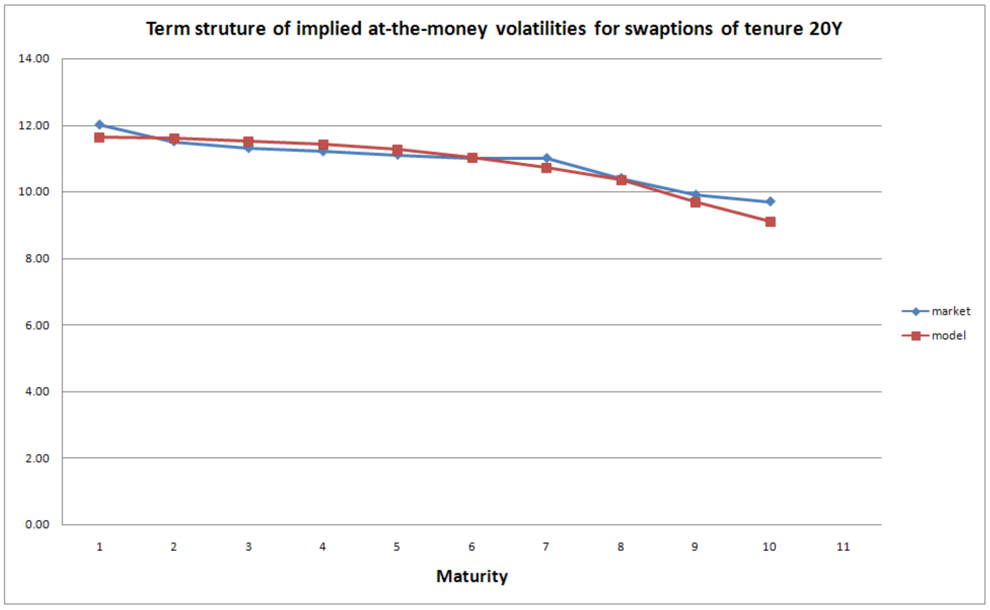

FIGURE 13. Implied volatilities for at-the-money European swaptions of tenor 20Y compared to market data.

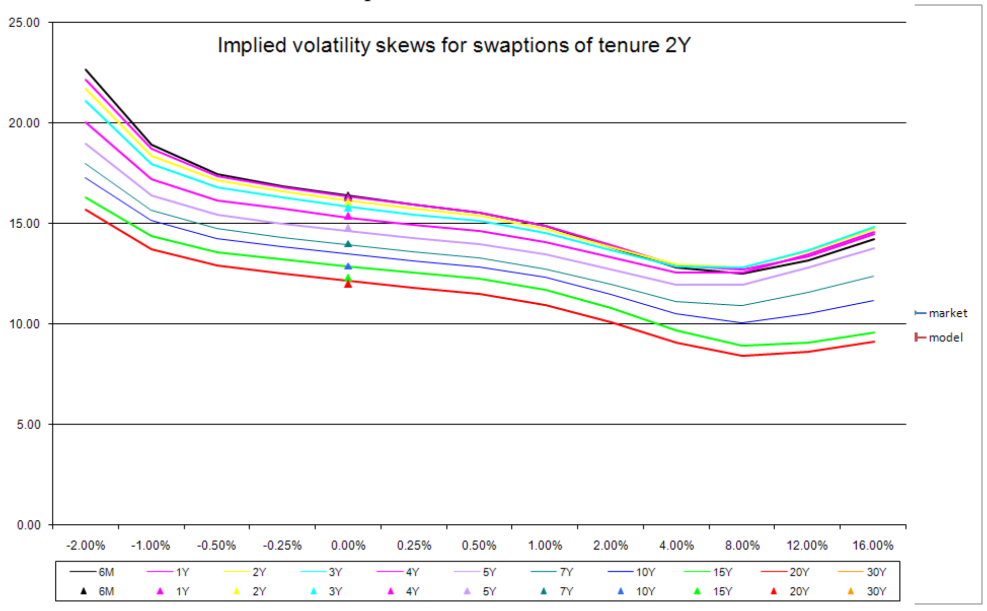

FIGURE 14. Implied volatility skews for European swaptions of tenor 2Y compared to market data.

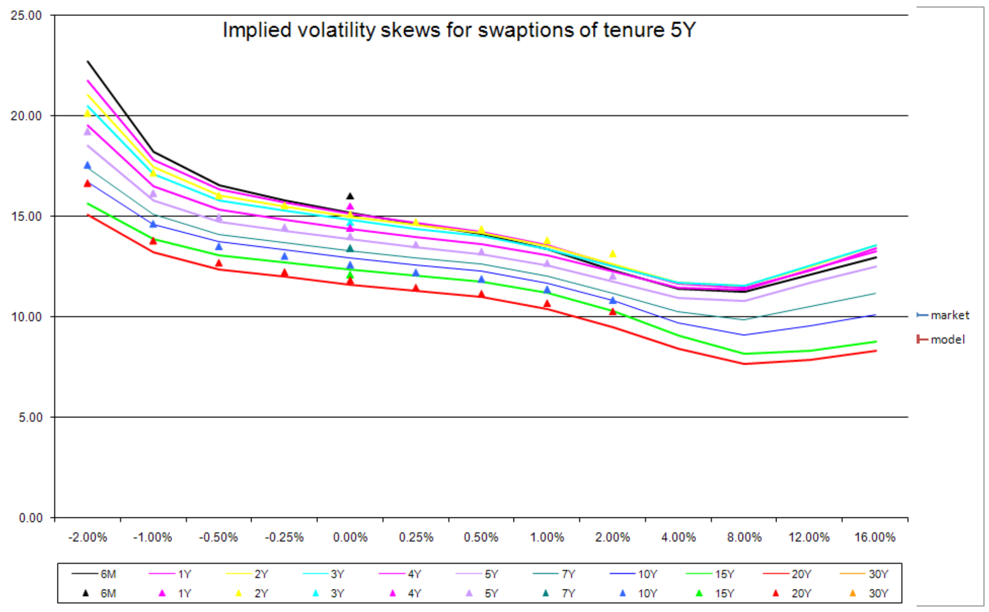

FIGURE 15. Implied volatility skews for European swaptions of tenor 5Y compared to market data.

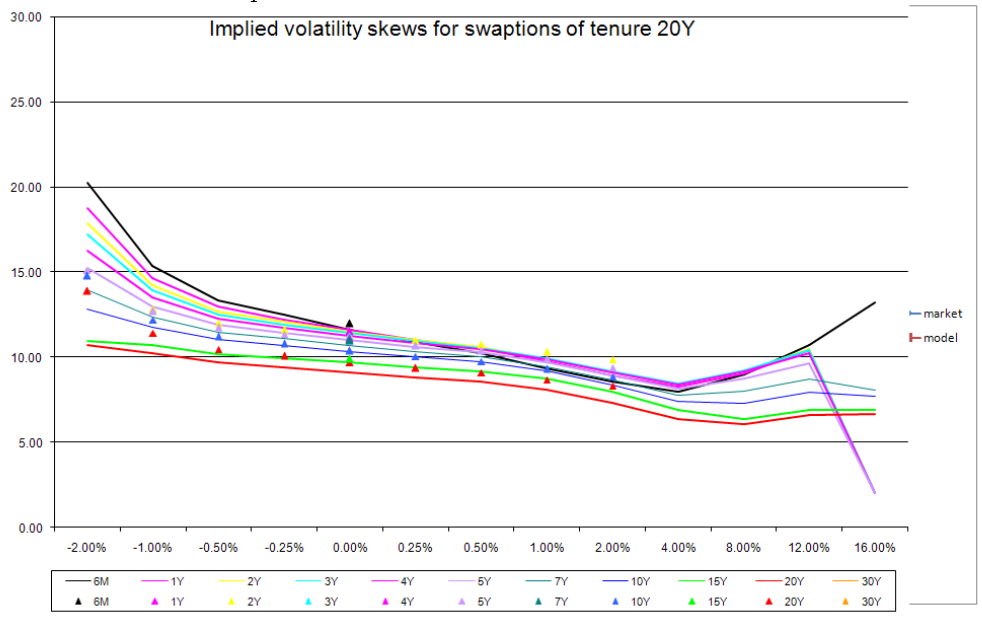

FIGURE 16. Implied volatility skews for European swaptions of tenor 20Y compared to market data. Only implied volatilites for extreme strikes 16% over the forward failed to compute, as the graph shoes.

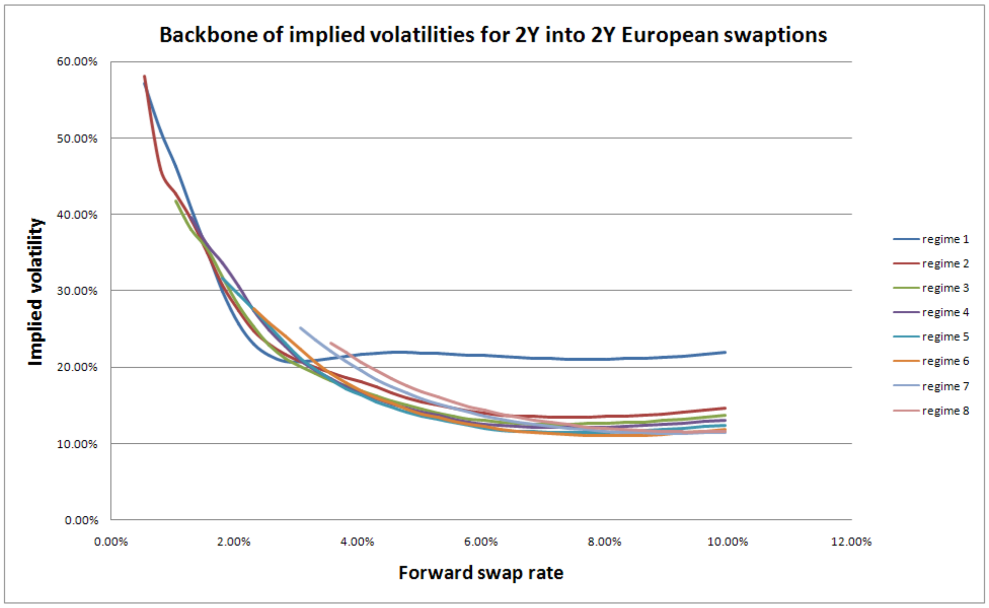

FIGURE 17. Implied volatility backbone, i.e. the scatterplot of the implied at the money volatility of 2Y into 2Y European swaptions versus the corresponding forward rate.

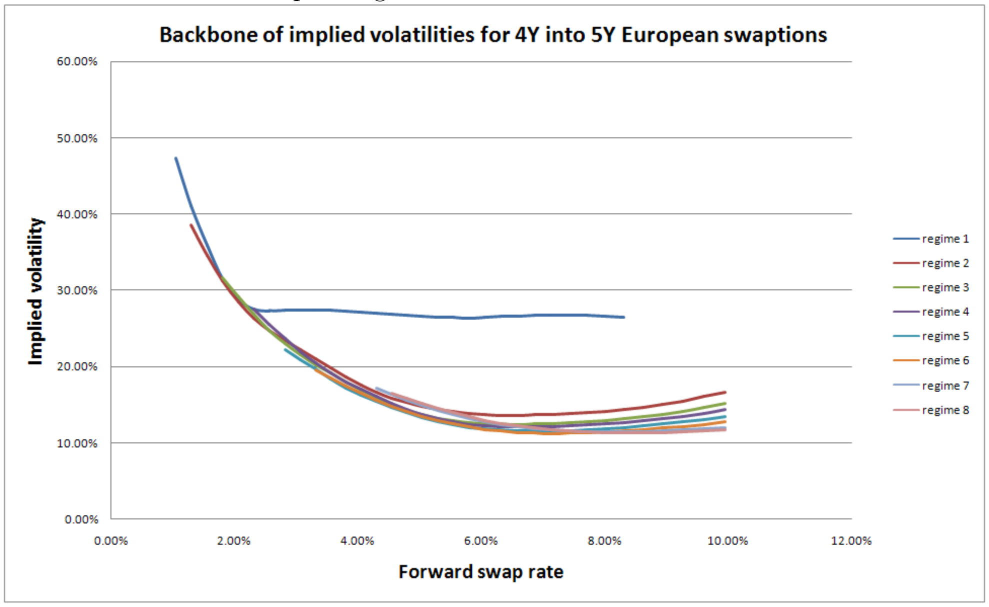

FIGURE 18. Implied volatility backbone, i.e. the scatterplot of the implied at the money volatility of 4Y into 5Y European swaptions versus the corresponding forward rate.

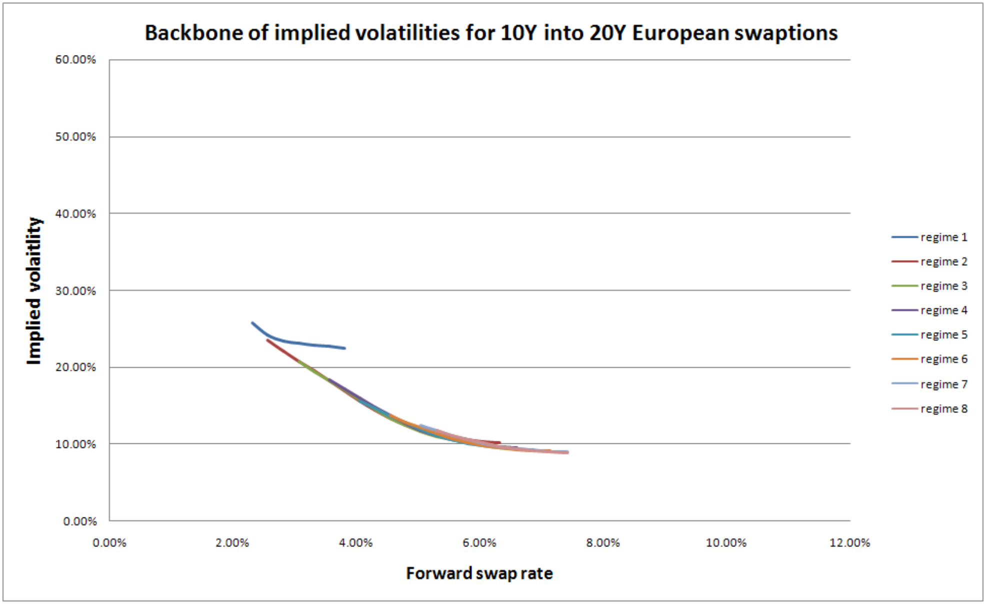

FIGURE 19. Implied volatility backbone, i.e. the scatterplot of the implied at the money volatility of 10Y into 20Y European swaptions versus the corresponding forward rate.

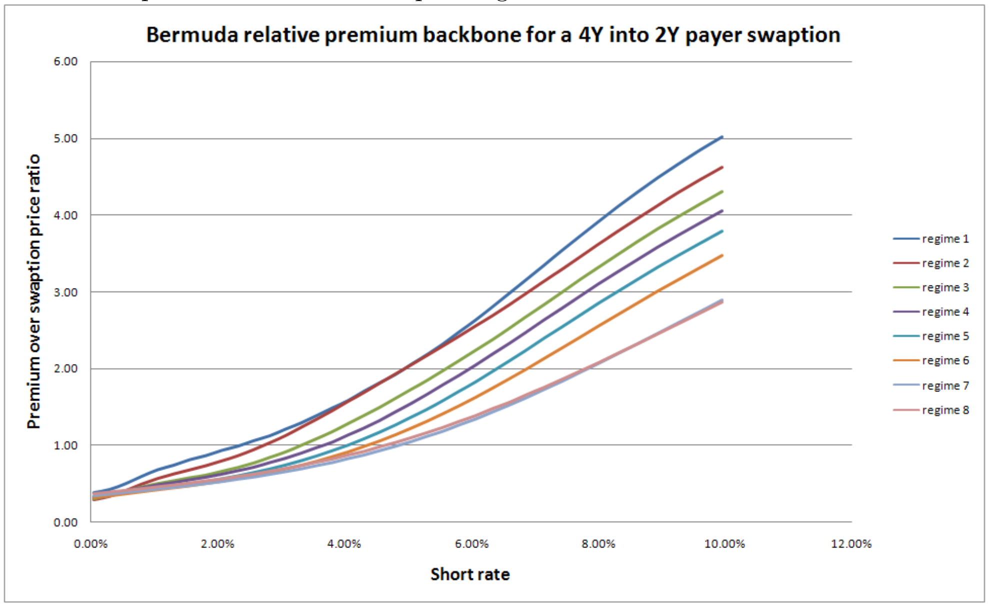

FIGURE 20. Bermuda premium backbone, i.e. the Bermuda premium of 4Y into 2Y at-the-money swaptions plotted against the corresponding forward rate.

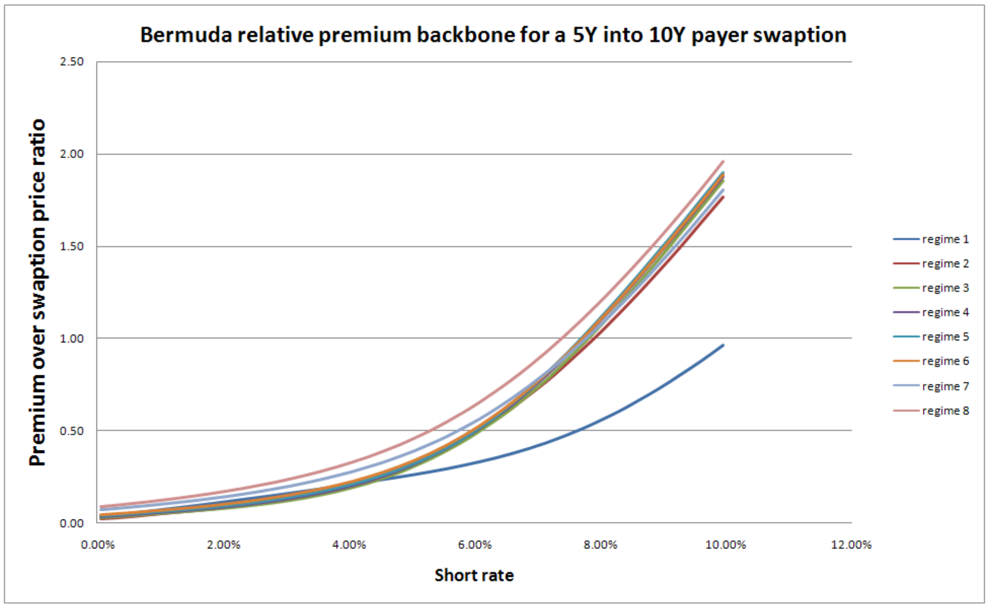

FIGURE 21. Bermuda premium backbone, i.e. the Bermuda premium of 5Y into 10Y at-the-money swaptions plotted against the corresponding forward rate.

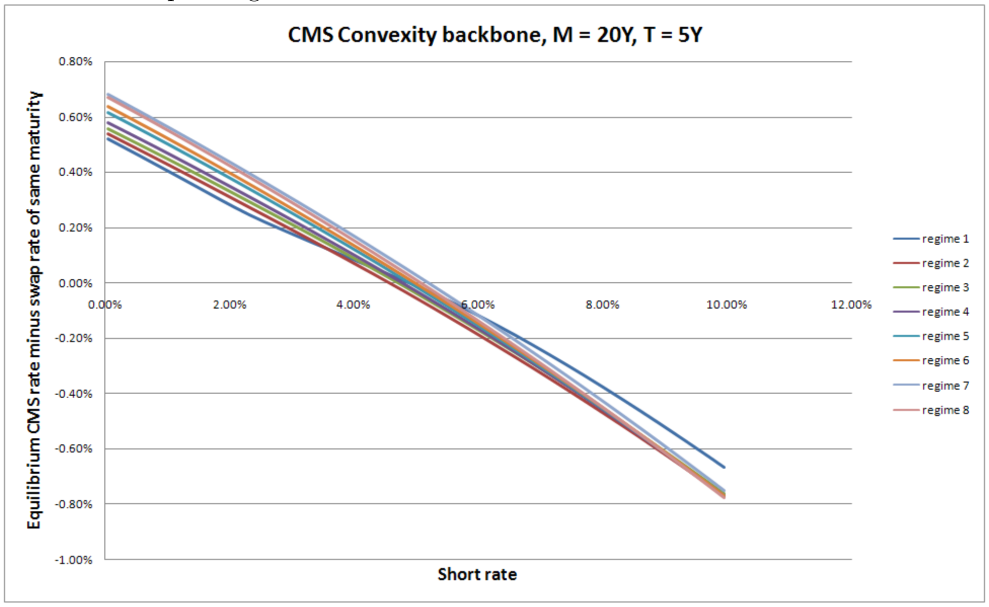

FIGURE 22. 2Y into 5Y convexity backbone, i.e. the convexity correction for of 2Y into 5Y European constant-maturity-swaps (CMS) with respect to European swaptions.

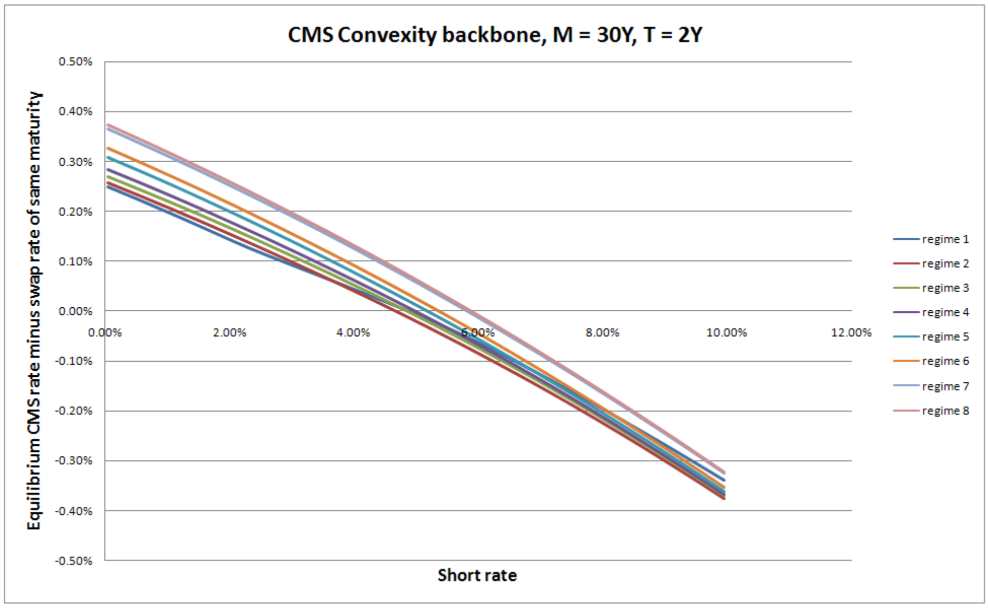

FIGURE 23. 3Y into 2Y convexity backbone, i.e. the convexity correction for of 3Y into 2Y European constant-maturity-swaps (CMS) with respect to European swaptions.

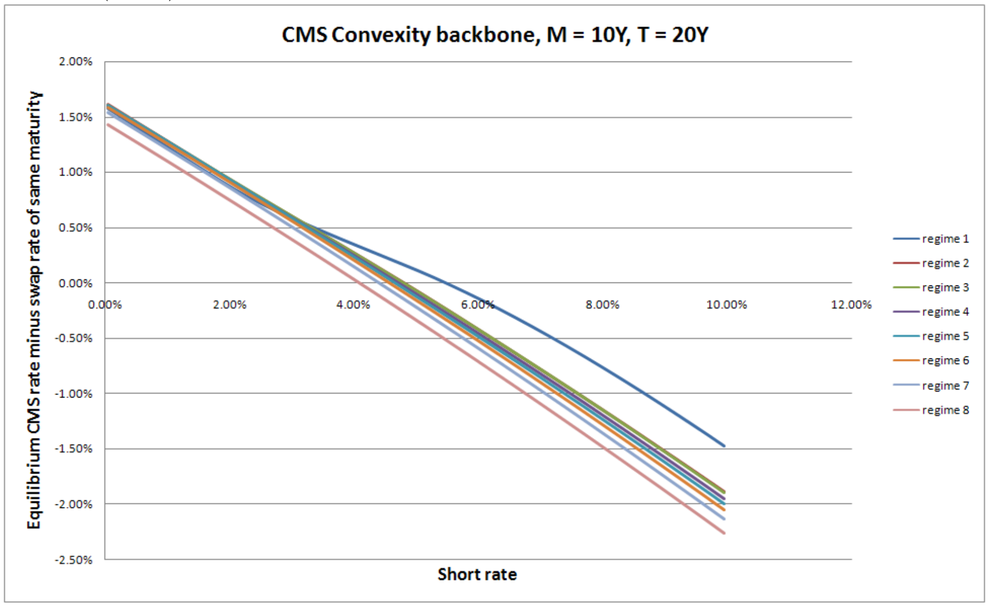

FIGURE 24. 10Y into 20Y convexity backbone, i.e. the convexity correction for of 10Y into 20Y European constant-maturity-swaps (CMS) with respect to European swaptions.

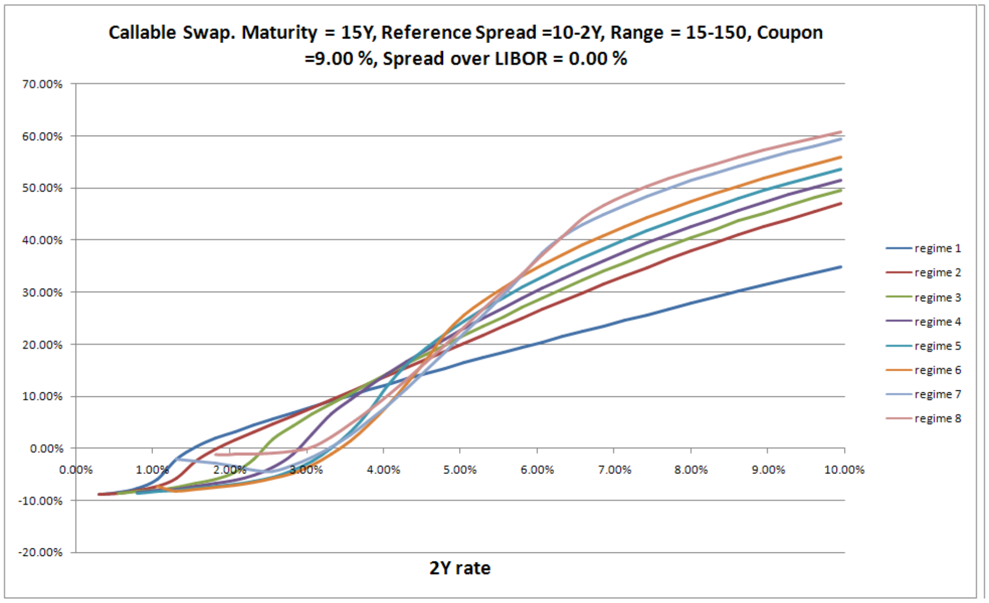

FIGURE 25. Pricing functions of callable CMS spread range accruals.

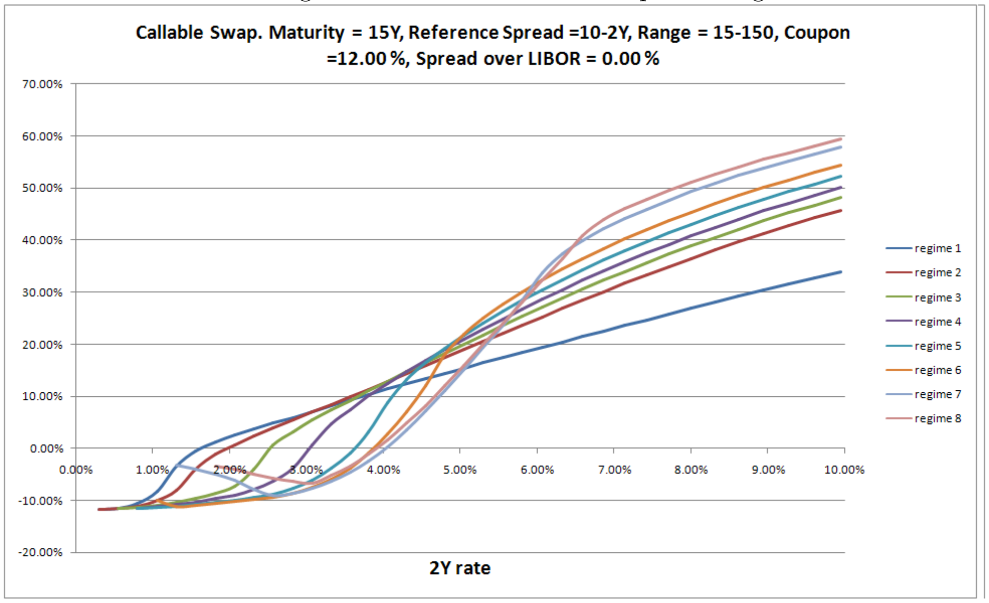

FIGURE 26. Pricing functions of callable CMS spread range accruals.

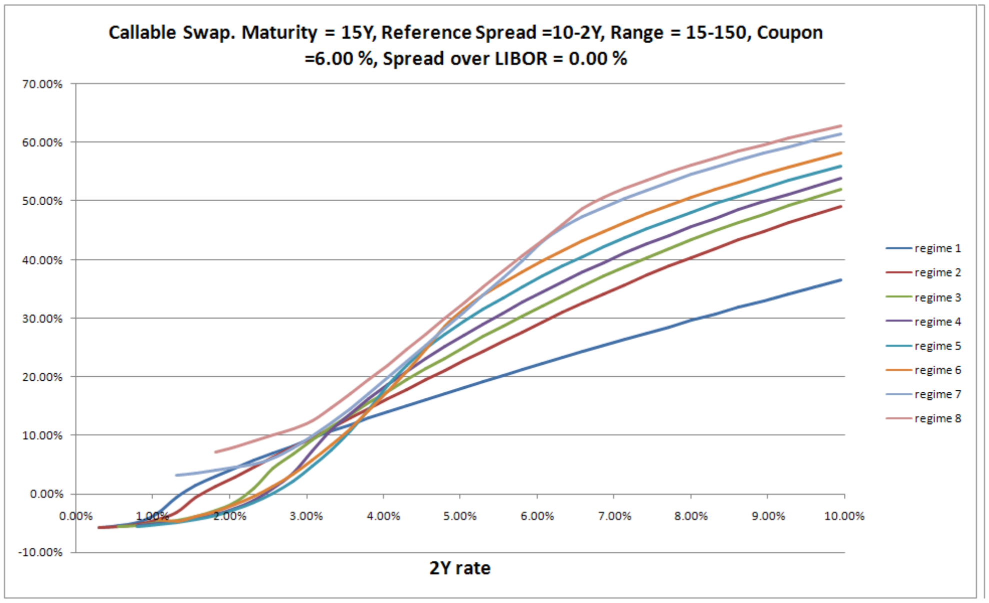

Figure 27. Pricing functions of callable CMS spread range accruals.

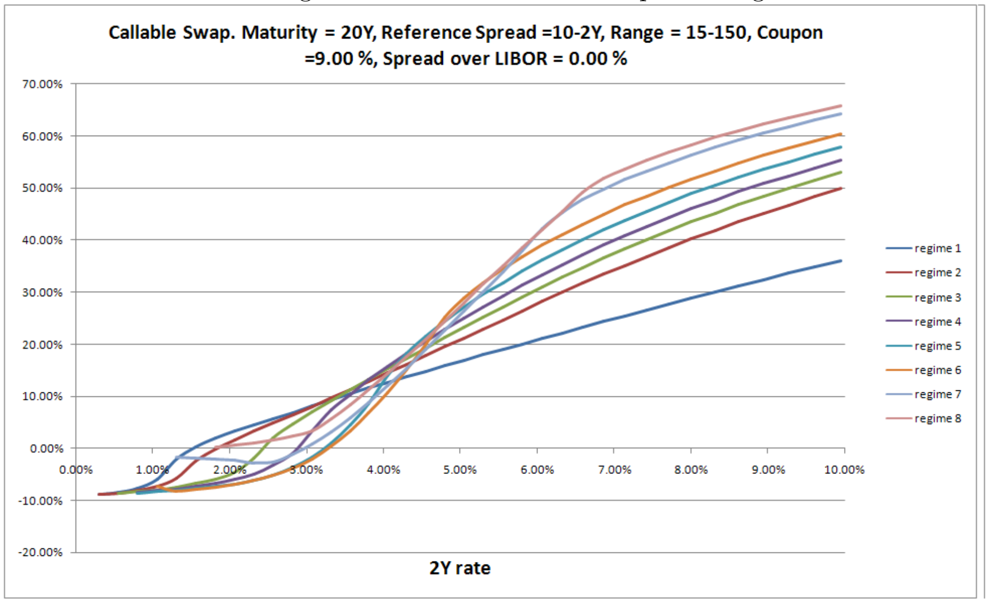

Figure 28. Pricing functions of callable CMS spread range accruals.

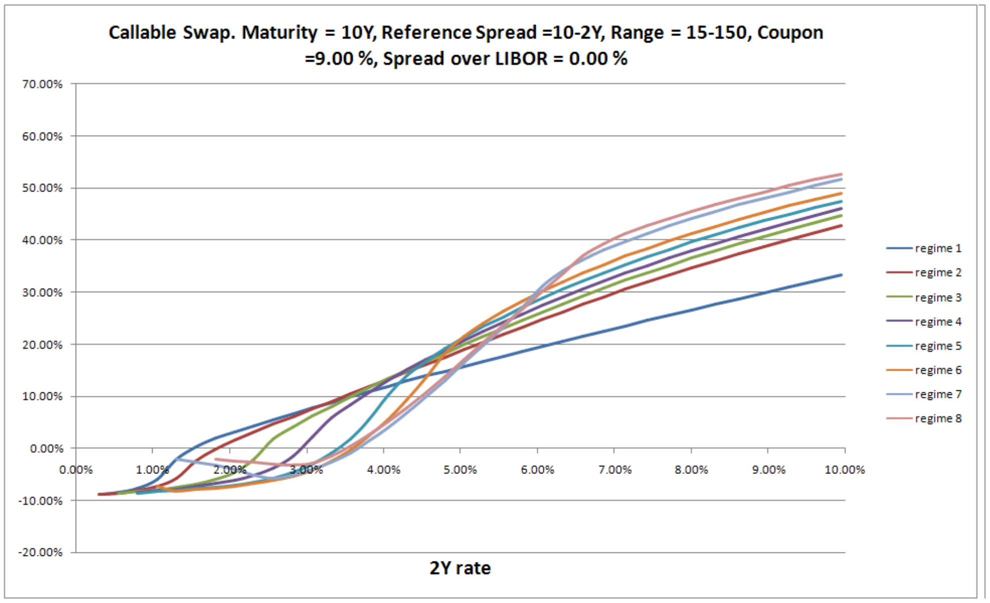

FIGURE 29. Pricing functions of callable CMS spread range accruals.

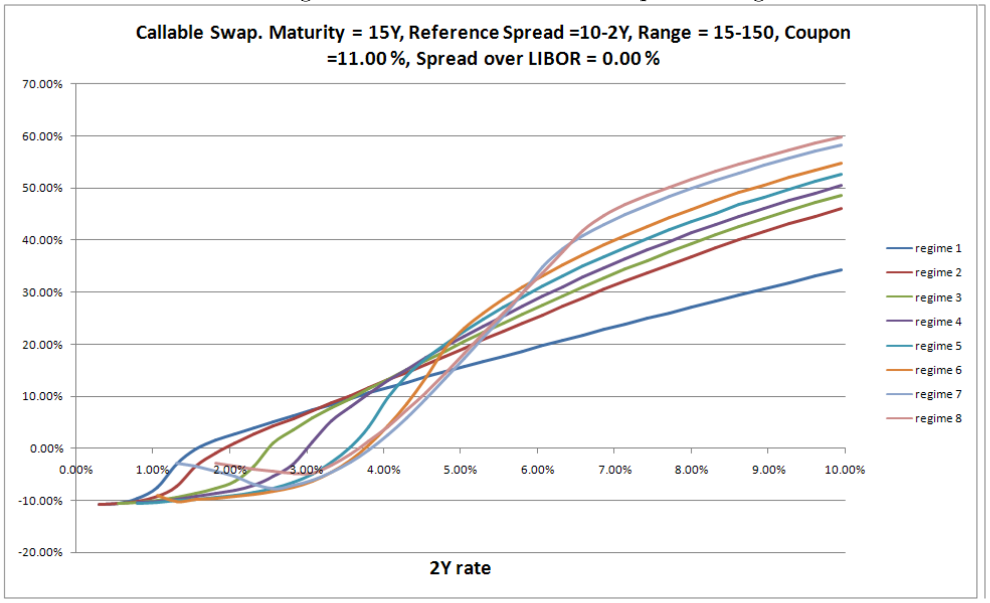

FIGURE 30. Pricing functions of callable CMS spread range accruals.

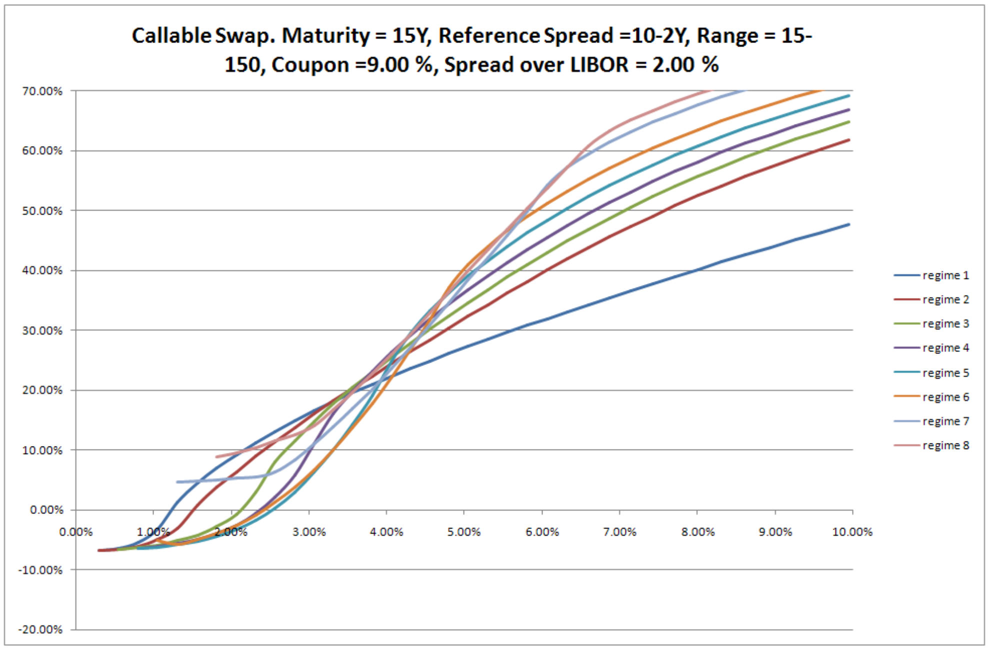

Figure 31. Pricing functions of callable CMS spread range accruals.

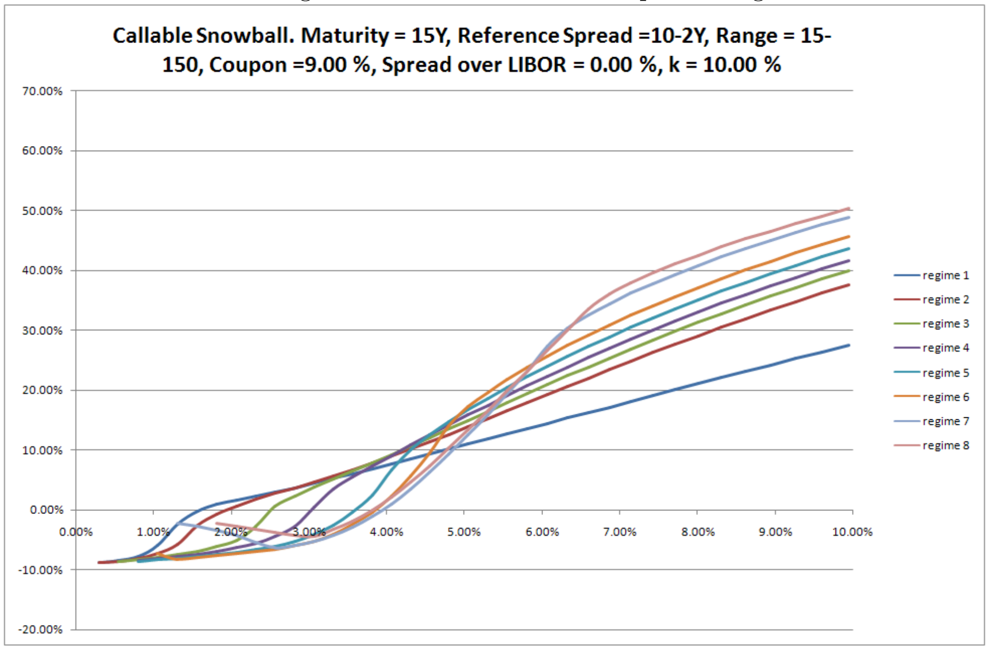

Figure 32. Pricing functions of callable snowball CMS spread range accruals.

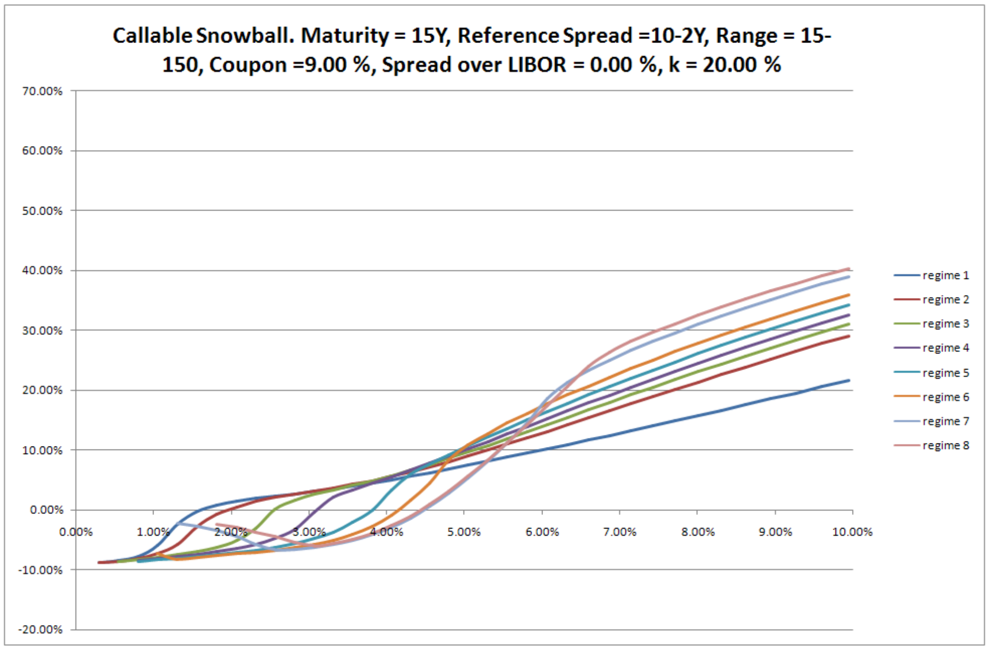

FIGURE 33. Pricing functions of callable snowball CMS spread range accruals.

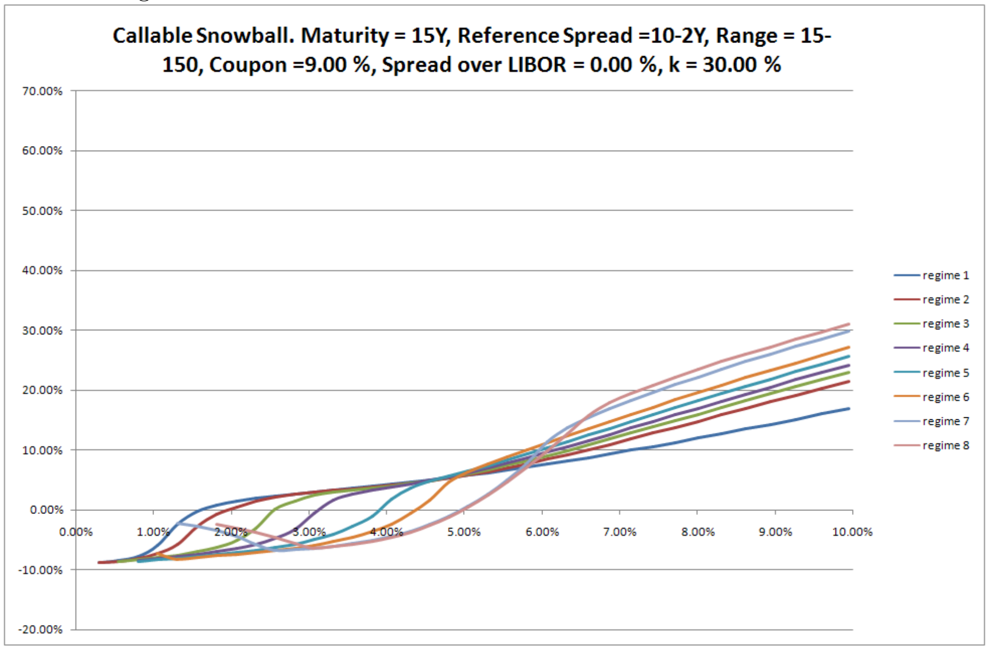

FIGURE 34. Pricing functions of callable snowball CMS spread range accruals.