

Non-recombining trees for the pricing of interest rate derivatives in the BGM/J framework

Peter Jäckel

Quantitative Research Centre, The Royal Bank of Scotland
135 Bishopsgate, London EC2M 3UR

17th October 2000

Abstract

In the Libor market model framework of Brace-Gatarek-Musiela and Jamshidian (BGM/J) for the pricing of interest rate derivatives, the drifts of the underlying forward rates are state dependent. Wherever Monte Carlo methods can be used for the numerical calculation of discounted expectations, this poses no major difficulty since the computational effort necessary in order to achieve acceptable accuracy depends only weakly on the dimensionality of the sampling space. For the valuation of options that involve finding the optimal exercise strategy, however, this means that any finite-differencing method enabling us to make the pointwise comparison directly between intrinsic value and discounted expectation will have to cope with “the curse of dimensionality” whereby the number of evaluations explodes exponentially. This document is about the implementation of a non-recombining multi-factor tree algorithm with a minimal number of branches out of each node for the representation of the desired number of factors. This method can serve as a benchmark for simple test cases for the development of other approximations such as exercise-strategy parametrisations in a Monte Carlo setup.

1 Introduction

Traditionally, implementations of option pricing models tended to use some form of lattice method. In most cases, this meant an explicit finite-differencing approach was chosen. In fact, many of the early quantitative analysts would describe this as ‘having been brought up on *trees*’. This tendency towards the use of tree methods is also reflected in the option pricing literature. Cox, Ross and Rubinstein [CRR79] described the option pricing procedure on a binomial tree in 1979.

And some of the breakthrough publications in derivatives modelling were first formulated as an algorithm for a tree node construction matching a market given set of security prices and Black implied volatilities. These include the lognormal interest rate model by Black, Derman, and Toy [BDT90] and the deterministic but spot dependent instantaneous volatility model by Derman and Kani [DK98]. The great advantages of recombining tree methods are their comparative ease of implementation, equally easy applicability to the calculation of Greeks, and fast performance.

Alas, we cannot always use recombining tree methods. This is typically so when the stochastic process chosen to model the evolution of the underlying quantities is strongly path-dependent. The state-dependence of the drift term of forward rates in the BGM/J framework is one such case. This makes it a prime application of Monte Carlo methods. However, when we wish to price options of American style, we really need to compare the expected payoff as seen from any one node with the intrinsic value. This means, the only method that can in principle give an unbiased result is a non-recombining tree. Whilst there are many publications on recombining tree methods and how to construct them for optimal performance, very little is in the literature on the construction of non-recombining trees. What's more, the few descriptions of the construction of non-recombining tree methods and analysis of their performance [JW00, MW99, Rad98a, Rad98b] focus on no more than three factors. In this article, I present a generic method to construct a non-recombining tree for any given number of factors and provide the algebraic equations needed to calculate the coefficients that determine the branches. McCarthy and Webber [MW99] and Radhakrishnan [Rad98a] discuss the question of *the clustering of nodes* and suggest methods to overcome it such as varying the step size, for instance in a linearly increasing or decreasing fashion, or changing both the length of some of the branches and their associated probabilities. For realistic applications, however, one tends to use a noticeably time-varying term structure of volatility which effectively changes the width of the branches over different time steps sufficiently to remove most of the harmful effect of clustering, and therefore I don't consider this issue of major importance.

The remainder of this article is organised as follows. First, I briefly summarise the setting of the BGM/J model and discuss its factorisation. Then, in section 3, I explain how the evolution of forward rates can be modelled in a non-recombining tree method. Next, I discuss in more detail some of the aspects of the high-dimensional geometry of the branching scheme in section 4. Following this, I elaborate a few points on the efficient implementation of the algorithm. The main results on the performance and applicability of the method are then presented in section 6. Next, I explain possible improvements that can be done to match the variance as it would result from a continuous description in section 7. Furthermore, I discuss a different technique to account for the state-dependent drift of the underlying forward rates in section 8 such that all martingale conditions are met exactly. Following that, I demonstrate how the clustering effect that can be

observed for flat volatility structures is broken up by the use of a time-varying term structure of instantaneous volatility in section 9. Finally, I conclude.

2 Factorisation of the BGM/J Libor market model

Given the assumption of instantaneously correlated lognormal evolution of n individual forward rates in the Forward Rate Agreement (FRA) based Libor market BGM/J model [Reb98, Reb99], the dynamics of the rates are determined by the stochastic differential equation

$$\frac{df_i}{f_i} = \mu_i(\mathbf{f}, t)dt + \sigma_i(t)d\widetilde{W}_i . \quad (1)$$

The correlation is incorporated by the fact that the n standard Wiener processes in equation (1) satisfy

$$\mathbb{E} \left[d\widetilde{W}_i d\widetilde{W}_j \right] = \varrho_{ij}dt . \quad (2)$$

Given the instantaneous covariance matrix $C(t)$, with its elements being

$$c_{ij}(t) = \sigma_i(t)\sigma_j(t)\varrho_{ij} , \quad (3)$$

and a decomposition of $C(t)$ into a pseudo-square root \tilde{A} such that

$$C = \tilde{A}\tilde{A}^T \quad (4)$$

we can transform equation (1) to

$$\frac{df_i}{f_i} = \mu_i dt + \sum_j \tilde{a}_{ij} dW_j \quad (5)$$

with dW_j being n independent standard Wiener processes where dependence on time has been omitted for clarity. It is also possible to drive the evolution of the n forward rates with fewer underlying independent standard Wiener processes than there are forward rates, say only m of them, which is particularly desirable when a non-recombining tree method is to be employed. In this case, the coefficient matrix $\tilde{A} \in \mathbb{R}^{n \times n}$ is to be replaced by $A \in \mathbb{R}^{n \times m}$ which must satisfy

$$\sum_{j=1}^m a_{ij}^2 = c_{ii} \quad (6)$$

in order to retain the calibration of the options on the FRAs, i.e. the caplets. In practice, this can be done very easily by calculating the decomposition as in equation (4) as before and rescaling according to

$$a_{ij} = \tilde{a}_{ij} \sqrt{\frac{c_{ii}}{\sum_{k=1}^m \tilde{a}_{ik}^2}} . \quad (7)$$

The effect of this procedure is that the individual variances of each of the rates are still correct, even if we have reduced the number of driving factors to one, but the effective covariances will be higher. In fact, for a single factor model, all correlations will be unity and the covariances just the products of the pairs of associated volatilities.

If the forward yield curve is given by n spanning forward rates f_i , whereby the payoff of forward rate agreement i is $f_i\tau_i$ and paid at time t_{i+1} , and we use a zero coupon bond that pays one currency unit at t_N as numeraire as schematically illustrated in figure 1, then the drifts μ_i in

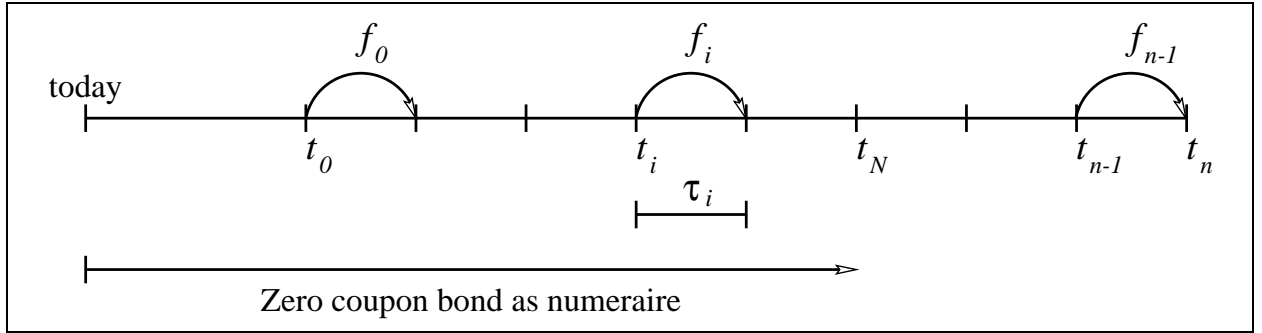


Figure 1: The yield curve is specified by a set of spanning forward rates.

equations (1) and (5) can be calculated by the aid of Itô's lemma to be:

$$\mu_i = \begin{cases} -\sigma_i \sum_{k=i+1}^{N-1} \frac{f_k \tau_k}{1+f_k \tau_k} \sigma_k \rho_{ik} & \text{for } i < N-1 \\ 0 & \text{for } i = N-1 \\ \sigma_i \sum_{k=N}^i \frac{f_k \tau_k}{1+f_k \tau_k} \sigma_k \rho_{ik} & \text{for } i \geq N \end{cases} \quad (8)$$

3 Evolving the forward rates

Clearly, equation (8) means that the drifts are state-dependent and thus indirectly stochastic. For the purpose of derivatives pricing, we need to sample the space of all possible evolutions of the yield curve into the future. If we approximate the drift coefficients μ_i as constant over a small time step Δt , we can represent the evolved forward rates by

$$f_i(t + \Delta t) = f_i \cdot e^{\bar{\mu}_i(t, t+\Delta t)\Delta t - \frac{1}{2}c_{ii} + \sum_{j=1}^m \bar{a}_{ij} z_j} \quad (9)$$

with z_j being independent normal variates. The coefficients \bar{a}_{ij} are calculated from the entries in the covariance matrix \bar{C}' as before, only the elements of the covariance matrix now have to be

integrals over the small time step Δt :

$$\bar{c}'_{ij} = \int_{t'=t}^{t+\Delta t} \sigma_i(t')\sigma_j(t')\varrho_{ij}dt' \quad (10)$$

To summarise, the steps that have to be carried out for the construction of Δt -evolved forward rates as in equation (9) are as follows:-

1. Populate the marginal covariance matrix $\bar{C}'(t, t + \Delta t)$ using equation (10).
2. Decompose (e.g. using the Cholesky method or by spectral decomposition) such that

$$\bar{A}' \bar{A}'^T = \bar{C}' . \quad (11)$$

3. Form the m -factor truncated coefficient matrix \bar{A} using

$$\bar{a}_{ij} = \bar{a}'_{ij} \sqrt{\frac{\bar{c}'_{ii}}{\sum_{k=1}^m \bar{a}'_{ik}{}^2}} . \quad (12)$$

4. Build the m -factor approximation covariance matrix \bar{C} :

$$\bar{C} = \bar{A}\bar{A}^T \quad (13)$$

which will in general, for $m < n$ not be identical to \bar{C}' (apart from the diagonal elements which were preserved by construction).

Given the above definitions, we can now specify $\bar{\mu}_i$ in equation (9):

$$\bar{\mu}_i(t, t + \Delta t)\Delta t = \begin{cases} - \sum_{k=i+1}^{N-1} \frac{f_k(t)\tau_k}{1+f_k(t)\tau_k} \bar{c}_{ik}(t, t + \Delta t) & \text{for } i < N - 1 \\ 0 & \text{for } i = N - 1 \\ \sum_{k=N}^i \frac{f_k(t)\tau_k}{1+f_k(t)\tau_k} \bar{c}_{ik}(t, t + \Delta t) & \text{for } i \geq N \end{cases} \quad (14)$$

In a Monte Carlo framework, we would now construct Δt -forward yield curves by drawing many independent m -dimensional normal variate vectors \mathbf{z} and applying them to equation (9). In order to build a tree for the pricing of derivatives that require the comparison between expectation and intrinsic value such as Bermudan swaptions, we now wish to use the minimal number of such vectors necessary. Clearly, in a 1-factor model, we could simply use the set $\{(+1), (-1)\}$ and thus construct a non-recombining binomial tree. In this case, it can be shown that the Δt -step evolution

equation (9) produces a set of evolved forward rates that is accurate up to order $\mathcal{O}\left((\sigma\sqrt{\Delta t})^3\right)$ (inclusive) both in the expected value and in variance. However, changing equation (9) to

$$f_i(t + \Delta t) = f_i \cdot e^{\bar{\mu}_i(t, t + \Delta t)\Delta t - \frac{1}{2}c_{ii} + \frac{1}{12}c_{ii}^2 - \frac{1}{45}c_{ii}^3 + \frac{17}{2520}c_{ii}^4 + \sum_{j=1}^m \bar{a}_{ij}z_j} \quad (15)$$

will correct the expected value up to order $\mathcal{O}\left((\sigma\sqrt{\Delta t})^9\right)$ (inclusive) and reduce the coefficient in front of the order $\mathcal{O}\left((\sigma\sqrt{\Delta t})^4\right)$ for the variance from $-5/6$ to $-2/3$. In practice, though, I found very little difference in the convergence behaviour when replacing equation (9) with equation (15). This indicates that other factors dominate the convergence behaviour.

In order to see how to construct variate vector sets $\{z\}$ for any given m , it is conducive to state clearly the requirements on the elements of the matrix $S \in \mathbb{R}^{m' \times m}$ whose rows comprise the vectors z to be used for each realisation of the evolved yield curve as given by equation (9). Assuming that we wish to assign equal probability to each of the m' realisations, we thus have¹:

$$\sum_{i=1}^{m'} s_{ij} = 0 \quad (\text{Mean}) \quad (16)$$

$$\frac{1}{m'} \sum_{i=1}^{m'} s_{ij}s_{ik} = \delta_{jk} \quad (\text{Covariance}) \quad (17)$$

$$\sum_{j=1}^m s_{ij}s_{kj} = \begin{cases} m & \text{for } i = k \\ -1 & \text{for } i \neq k \end{cases} \quad (\text{Equal probability}) \quad (18)$$

The smallest m' for which it is possible to construct S satisfying the above equation is $m + 1$. In other words, for an m -factor tree model, we need a minimum of $m + 1$ branches out of each node.

The elements of the above matrix S describe the Cartesian coordinates of a perfect simplex in m dimensions. Equation (18) can best be understood by the geometrical interpretation that in order to define equally probable tree branches, all the angles in the simplex must be equal (which makes it a perfect simplex). Note that I have made no statements about the alignment of this simplex in our coordinate system yet.

¹This set of equations is not strictly independent. Stating all of them, however, aids the clarification of the simplex concept.

4 Optimal simplex alignment

Given a Cartesian coordinate system, we can write the coordinates of the corner points defining a perfect simplex in m dimensions as:-

$$s_{ij}^{(m)} = \begin{cases} -\sqrt{\frac{m+1}{(j+1)j}} & \text{for } j \geq i \\ \sqrt{\frac{(m+1)j}{j+1}} & \text{for } j = i - 1 \\ 0 & \text{for } j < i - 1 \end{cases} \quad (19)$$

Examples:

$$S^{(1)} = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \quad (20)$$

$$S^{(2)} = \begin{pmatrix} -\sqrt{\frac{3}{2}} & -\sqrt{\frac{1}{2}} \\ \sqrt{\frac{3}{2}} & -\sqrt{\frac{1}{2}} \\ 0 & \sqrt{2} \end{pmatrix} \quad (21)$$

$$S^{(3)} = \begin{pmatrix} -\sqrt{2} & -\sqrt{\frac{2}{3}} & -\sqrt{\frac{1}{3}} \\ \sqrt{2} & -\sqrt{\frac{2}{3}} & -\sqrt{\frac{1}{3}} \\ 0 & \sqrt{\frac{8}{3}} & -\sqrt{\frac{1}{3}} \\ 0 & 0 & \sqrt{3} \end{pmatrix} \quad (22)$$

$$S^{(4)} = \begin{pmatrix} -\sqrt{\frac{5}{2}} & -\sqrt{\frac{5}{6}} & -\sqrt{\frac{5}{12}} & -\sqrt{\frac{1}{4}} \\ \sqrt{\frac{5}{2}} & -\sqrt{\frac{5}{6}} & -\sqrt{\frac{5}{12}} & -\sqrt{\frac{1}{4}} \\ 0 & \sqrt{\frac{10}{3}} & -\sqrt{\frac{5}{12}} & -\sqrt{\frac{1}{4}} \\ 0 & 0 & \sqrt{\frac{15}{4}} & -\sqrt{\frac{1}{4}} \\ 0 & 0 & 0 & \sqrt{4} \end{pmatrix} \quad (23)$$

Using the definition of S , we can now specify a branch coefficient matrix B as

$$\bar{B} = \bar{A} \cdot S^T. \quad (24)$$

The tree construction algorithm is thus as follows. At each node with its associated yield curve given by the set of n forward rates $\{f_i(t)\}$, construct a set of $n \cdot (m + 1)$ forward rates to represent all possible evolutions over a time interval Δt according to

$$f_{ik}(t + \Delta t) = f_i(t) \cdot e^{\bar{\mu}_i(t)\Delta t - \frac{1}{2}\bar{c}_{ii}(t) + \bar{b}_{ik}(t)} \quad i = 1 \dots n, k = 1 \dots m + 1. \quad (25)$$

There may be situations when we would like to have more than $m + 1$ branches. An example is the pricing of a path-dependent derivative on a single FRA. In this case, we have $m = 1$ but we might want to construct a non-recombining trinomial tree because of the inherently higher convergence rate and stability in comparison to the binomial tree. Examples are not only standard payoff derivatives like a caplet but also barrier options, trigger derivatives, etc. This can be achieved very easily in the above framework by using only the first m columns of the matrix describing a perfect simplex in $(N_{\text{Branches}} - 1)$ dimensions instead of S in equation (24), i.e.

$$S^{(m, N_{\text{Branches}})} = S^{(N_{\text{Branches}} - 1)} \cdot \begin{pmatrix} \mathbf{1}_m \\ \mathbf{0}_{(N_{\text{Branches}} - 1 - m) \times m} \end{pmatrix} \quad (26)$$

with

$$\mathbf{1}_m \in \mathbb{R}^{m \times m}$$

being the m -dimensional identity matrix, and

$$\mathbf{0}_{(N_{\text{Branches}} - 1 - m) \times m} \in \mathbb{R}^{(N_{\text{Branches}} - 1 - m) \times m}$$

being a matrix whose elements are all zero.

In general, there are no limitations to how many branches one may use out of each node. In fact, many recombining tree-like methods or PDE solvers use effectively² more than three nodes for improved convergence. Examples include fast convolution methods such as the ones using Fourier [CM99] or Laplace transformations [FMW98] but also the willow tree method [Cur94]. However, using the simplex coordinates as given by equation (19) will quickly result in redundant, i.e. identical branch coefficients. For instance, if we were to choose a 4-branch construction for a single factor model, we would probably want the four branches to end in four different realisations of the evolved forward rate. As we can see in equation (22), however, two of the branch coefficients in the first column are identical, namely 0. In fact, if we look at the branch coefficients of the first modes in the higher dimensional simplices, i.e. the entries in the first columns of the $S^{(\cdot)}$ matrices, we realise that there are never more than three different values. In geometrical terms, this is a consequence of our particular choice of alignment of the simplex as specified by equation (19). In order to obtain the maximum benefit out of the additional effort in using more branches, we may want them to spread as much and as evenly as possible. In each column, we may wish to have the entries to be symmetrically distributed around zero, to whatever extent this can be achieved. It turns out, for any m -dimensional perfect simplex, it is possible to find a rotation $R^{(m)}$ of the

²In a general sense, even implicit finite differencing methods can be seen as a technique to use many nodes at a future time slice to infer the values at an earlier time slice.

simplex $S \xrightarrow{R^{(m)}} S'$ such that

$$\begin{aligned} s'_{ij}{}^{(m)} &= -s'_{m+2-i,j}{}^{(m)} \quad \text{for } m \text{ even and } j = 1 \dots \frac{m}{2} \\ s'_{ij}{}^{(m)} &= -s'_{m+2-i,j}{}^{(m)} \quad \text{for } m \text{ odd and } j = 1 \dots \frac{m+1}{2}. \end{aligned} \quad (27)$$

An appropriate rotation for the m -dimensional simplex can be found by specifying a rotation matrix

$$R^{(m)} = \prod_{\substack{l=m \\ k=m-1 \\ k=1 \\ l=k+1}} R_{kl}^{(m)}(\theta_{kl}) \quad (28)$$

with $R_{kl}^{(m)}(\theta_{kl}) \in \mathbb{R}^{m \times m}$ being the rotation matrix in the (k, l) plane by an angle θ_{kl} , i.e. $R_{kl}^{(m)}(\theta_{kl})$ is equal to the m -dimensional identity matrix apart from the elements $r_{kk}^{(m)} = r_{ll}^{(m)} = \cos \theta_{kl}$ and $r_{kl}^{(m)} = -r_{lk}^{(m)} = \sin \theta_{kl}$. The rotated simplex is then given by

$$S' = S \cdot R^{(m)}. \quad (29)$$

Allowing all of the $\frac{m(m-1)}{2}$ angles to vary, a simple iterative fitting procedure then very quickly finds a suitable rotation to minimise the χ^2 -error in the conditions given by equation (27). To give a specific example, one alignment of the simplex for $m = 4$ that satisfies equation (27) is given by:

$$S^{(4)} = \begin{pmatrix} -1.1588 & -0.91287 & -0.6455 & -0.5 \\ 1.1588 & -0.91287 & -0.6455 & -0.5 \\ 0 & 1.8257 & -0.6455 & -0.5 \\ 0 & 0 & 1.9365 & -0.5 \\ 0 & 0 & 0 & 2 \end{pmatrix} \xrightarrow{R^{(m)}} S'^{(4)} = \begin{pmatrix} -1.1306 & -1.1053 & 0 & 1.22474 \\ 1.1053 & -1.1306 & 0.91287 & -0.8165 \\ 0 & 0 & -1.8257 & -0.8165 \\ -1.1053 & 1.1306 & 0.91287 & -0.8165 \\ 1.1306 & 1.1053 & 0 & 1.22474 \end{pmatrix}. \quad (30)$$

Once we have identified a suitable alignment of the simplex, there is yet another easy method to improve the convergence behaviour of the non-recombining multi-nomial tree method. This technique is called *Alternating Simplex Direction* and entails simply switching the signs of all of the simplex coordinates in every step. How this improves convergence by increasing the overall symmetry of the procedure can be seen if we visualise the points generated by subsequent branching in the (z_1, z_2) -plane for a 2-factor, 3-branch model. This is shown in figure 2. Since we are merely adding up the coordinates of subsequent steps, the branching evolution appears to recombine. The moment we actually use the state dependent drift terms in a forward rate based yield curve model as in equation (25), this will no longer be the case. However, as we will see later, it is not unreasonable to expect that the added near-symmetry, in general, improves convergence.

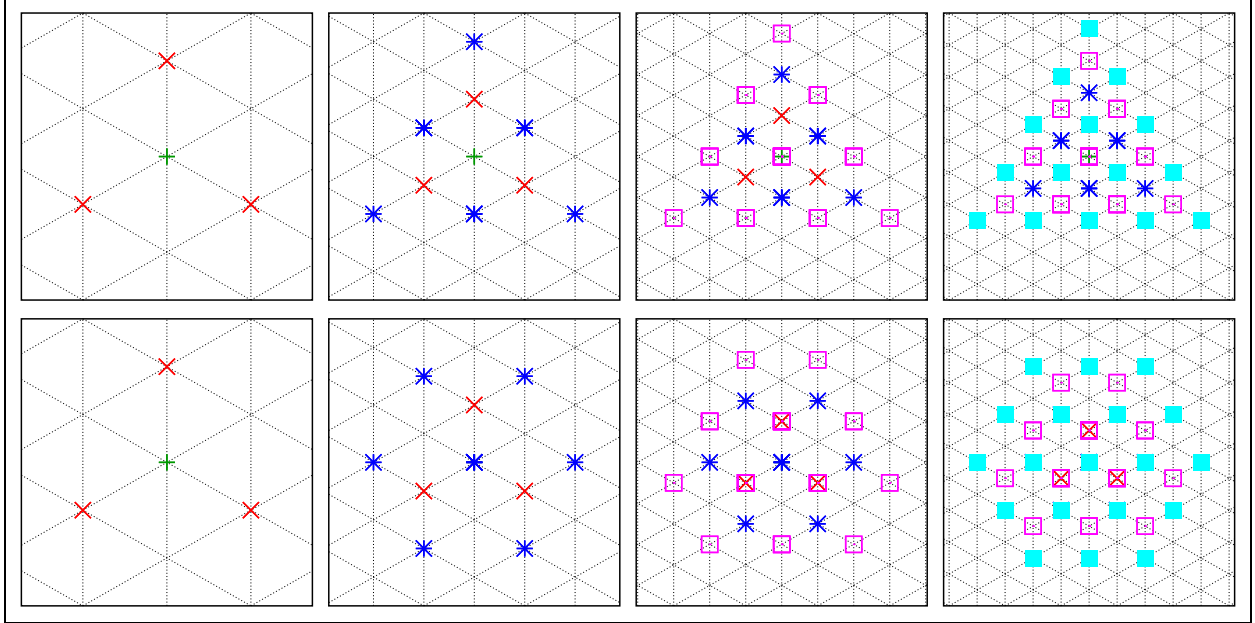


Figure 2: The placement of (z_1, z_2) nodes with (bottom) and without (top) the use of the *Alternating Simplex Direction* method. Note the increased symmetry when ASD is used.

5 Implementation

It is worth noting that neither the \bar{c}_{ii} nor the \bar{b}_{ik} in equation (25) depend on the current yield curve given by the $f_i(t)$. Therefore, they can be precalculated for all of the time steps. The only thing that needs to be calculated immediately prior to looping through all of the branches is the current set of drift terms $\{\bar{\mu}_i\}$. These, in turn, are the same for all of the branches out of each node. Taking all of the above considerations into account, we see that the non-recombining tree calculation can be implemented extremely efficiently using a recursive method since none of the evolved yield curves need to be reused after all of the branches out of any one node have been evaluated. The only storage we need to allocate is a full set of $\{\bar{\mu}_i\}$ for each time step, a full yield curve specifying FRA set $\{f_i\}$ for each time step, and of course, the \bar{c}_{ii} and \bar{b}_{ik} for each time step. In the code snippet shown in figure 1, the array element `LogShiftOfBranch[h][k][i]` contains $-\frac{1}{2}\bar{c}_{ii} + \bar{b}_{ik}$ for the time step from t_h to t_{h+1} , the array element `C[h][i][k]` holds the associated covariance matrix entry \bar{c}_{ik} for the time step, and all the other variable names should be self-explanatory. After the initial setting up, a call to the function `BushyNFactorFraBGMTree::Recurse(0)` returns with the expected value as given by the payoff specified in the function `BushyNFactorFraBGMTree::Intrinsic()`, taking into account possible early exercises. The return value of the `BushyNFactorFraBGMTree::Recurse(0)` call still has to be discounted by multiplying with the present value of the zero coupon bond chosen as numeraire.

```

double BushyNFactorFraBGMTree::Recurse(unsigned long h){
    if (h==NSteps)
        return Intrinsic(h); // Termination of the recursion.
    unsigned long i,k;
    for (i=0;i<NRates;i++){ //      Calculate the drift for all rates and store them.
        mu_dT[i] = 0.;
        for (k=NumeraireIndex;k<=i;k++)
            mu_dT[i] += C[h][i][k] * EvolvedFra[h][k] * Tau[k] / ( 1. + EvolvedFra[h][k] * Tau[k] );
        for (k=i+1;k<NumeraireIndex;k++)
            mu_dT[i] -= C[h][i][k] * EvolvedFra[h][k] * Tau[k] / ( 1. + EvolvedFra[h][k] * Tau[k] );
    }
    double tmp=0;
    for (k=0;k<NBranches;k++){ // Loop over all branches.
        for (i=0;i<NRates;i++){
            EvolvedFra[h+1][i] = EvolvedFra[h][i] * exp( mu_dT[i] + LogShiftOfBranch[h][k][i] );
        }
        tmp += Recurse(h+1); // Sum up the results from all of the branches.
    }
    // Average, unless the intrinsic value is higher.
    return CheckForEarlyExercise(h,tmp/NBranches);
}

```

Code example 1: The recursive implementation of the non-recombining tree.

6 Convergence performance

In order to give the reader a feeling for the effectiveness of the methods suggested in the previous sections, we have carried out a set of numerical calculations for a 4-year payer's option on a two year semiannual European swaption. I used the yield curve and caplet implied volatilities for GBP interested rates as tabulated in table 1, and assumed an instantaneous volatility of the individual

i	t_i	discount factor	f_i	$\hat{\sigma}_i = \sqrt{\int_{t'=0}^{t_i} \sigma_i(t')^2 dt' / t_i}$
0	4	0.762757096	6.652%	21.43%
1	4.5	0.739640975	6.251%	20.67%
2	5	0.717225412	6.044%	19.98%
3	5.5	0.696187117	6.044%	19.35%
4	6	0.675765937		

Table 1: The yield curve for GBP interest rates and the caplet implied volatilities used in the examples.

forward rates as in³

$$\sigma_i(t) = [a + b(t_i - t)] e^{-c(t_i - t)} + d \quad (31)$$

³c.f. [Reb99], equation (11.4).

with $a = -2\%$, $b = 0.5$, $c = 1$, and $d = 10\%$, which is consistent with the given caplet implied volatilities. The correlation between forward rates f_i and f_j as given by ϱ_{ij} in equation (2) and (10) was assumed to be

$$\varrho_{ij} = e^{-\beta|t_i - t_j|} \quad (32)$$

with $\beta = 0.1$. The strike for the swaption was set at 7.50%. Since the forward swap rate results to 6.15% for this particular yield curve, the option under consideration is out-of-the money. I also calculated the results for the equivalent Bermudan contract, i.e. a 6-non-call-4 semiannual Bermudan swaption. In figure 3, I show how the non-recombining tree model converges as a function of the number of steps to maturity for the pricing of European swaptions, and, more interestingly, in figure 4 the convergence behaviour for Bermudan swaptions is shown. Note how the Alternating Simplex Direction method improves convergence most for two or three factors, and how the optimal alignment technique ensures convergence consistently for as little as five steps for three or more factors, especially when used in conjunction with the ASD method.

7 Variance matching

Given an enumeration $t_1 \dots t_{N_{\text{steps}}}$ of the discrete points in time over which the tree algorithm is constructed, and defining γ_{hi} to represent all drift and Itô terms over the time step $t_h \rightarrow t_{h+1}$, i.e. $\gamma_{hi} := e^{\bar{\mu}_i(t_h)(t_{h+1}-t_h) - \frac{1}{2}\bar{c}_{ii}(t_h)}$, we can rewrite equation (25) as

$$f_{(h+1)ik} = f_{hi} \gamma_{hi} e^{\bar{b}_{hik}} . \quad (33)$$

Let us now recall that the coefficients \bar{b}_{hik} were constructed such that their discrete average over all emerging branches is zero and their discrete covariances equal the elements of the given covariance matrix of the logarithms of the forward rates over the specified time step. Alas, matching the discrete covariances of logarithms means that the covariances of the forward rates themselves are not exactly matched due to the convexity of the exponential function as is known from Jensen's inequality. However, the variance of any random variate x with a continuous lognormal distribution such as

$$x = \xi e^{\omega z} \quad \text{with} \quad z \sim \text{N}(0, 1) \quad (34)$$

can be calculated to

$$\text{V}[x] = \xi^2 e^{\omega^2} (e^{\omega^2} - 1) . \quad (35)$$

In other words, if we wish to construct the tree such that the variances of the forward rates themselves have the correct value as it would result from the continuous description, we can introduce a volatility scale parameter p_{hi} to be used in the branch construction as in

$$f_{(h+1)ik} = f_{hi} \gamma_{hi} e^{p_{hi} \bar{b}_{hik}} \quad (36)$$

such that

$$\frac{1}{N_{\text{Branches}}} \left[\sum_{k=1}^{N_{\text{Branches}}} e^{2p_{hi} \bar{b}_{hik}} - \frac{1}{N_{\text{Branches}}} \left(\sum_{k=1}^{N_{\text{Branches}}} e^{p_{hi} \bar{b}_{hik}} \right)^2 \right] - e^{\bar{c}_{hii}} (e^{\bar{c}_{hii}} - 1) = 0. \quad (37)$$

In order to meet this nonlinear condition for p_{hi} , define $\phi_{hi}(p_{hi})$ as the left hand side of equation (37). Given the initial guess of $p_{hi}^{(0)} = 1$ and the partial derivative

$$\frac{\partial \phi_{hi}(p_{hi})}{\partial p_{hi}} = \frac{1}{N_{\text{Branches}}} \left[\sum_{k=1}^{N_{\text{Branches}}} 2\bar{b}_{hik} e^{2p_{hi} \bar{b}_{hik}} - \frac{2}{N_{\text{Branches}}} \left(\sum_{k=1}^{N_{\text{Branches}}} e^{p_{hi} \bar{b}_{hik}} \right) \left(\sum_{k=1}^{N_{\text{Branches}}} \bar{b}_{hik} e^{p_{hi} \bar{b}_{hik}} \right) \right], \quad (38)$$

a Newton iteration converges to the solutions of $\phi_{hi}(p_{hi}) = 0$ very fast indeed. The nonlinear root solving has to be done for each forward rate and for each time step separately. This can be done during the startup period of the tree algorithm, though, and in our tests took no measurable computing time whatsoever⁴.

The above procedure does indeed result in an exact match of the variances as given by the continuous description. I would like to remark at this point that this may not be generally desirable, though. To see this, let us consider a call option of a quantity with a standard normal distribution, and let us ignore discounting effects. For a strike of zero, the value of the option is

$$\int_0^{\infty} s \frac{e^{-\frac{1}{2}s^2}}{\sqrt{2\pi}} ds = \frac{1}{\sqrt{2\pi}}. \quad (39)$$

A single step binomial tree discretisation of this distribution that matches both the expectation and the variance of the continuous counterpart exactly is the set $\{+1, -1\}$ of equiprobable values for s . Clearly, the latter results in an option of 0.5 while the continuous description gives us a value around 0.3989. I therefore expect that products with some kind of convexity in the payoff profile will be slightly overvalued by the discretised tree when continuous variances are matched. However, comparing the values as they result from the variance matched tree construction (36) and the original scheme (25) could provide some comfort about the possible mispricing due to the approximate volatility representation in the discretised scheme. In general, I would only expect the variance matched construction to provide faster convergence for directly volatility related products such as variance or volatility swaps.

⁴The granularity of the computation time measuring function was approximately $1/100$ seconds.

8 Exact martingale conditioning

In the recursion procedure of calculating all yield curve branches emanating out of one yield curve node, we always need to calculate the discrete time step drift approximation for each forward rate. As we know from section 3, the stepwise constant drift approximation (14) guarantees the martingale conditions that the expected value of any asset divided by the chosen numeraire asset equals its initial value only in the limit of small time steps. Choosing the numeraire to be the longest involved zero coupon bond, i.e. $N := n$ such that the payment time of the chosen zero coupon bond numeraire asset is the payment time of the last forward rate that is to be modelled, it is possible to meet the martingale conditions in each step exactly without any computational overhead. This can be seen as follows. For $N = n$, the martingale conditions are that for any time step $t_h \rightarrow t_{h+1}$ we have

$$\mathbb{E} \left[f_{(h+1)i} \prod_{j=i+1}^{N-1} (1 + \tau_j f_{(h+1)j}) \right] = f_{hi} \prod_{j=i+1}^{N-1} (1 + \tau_j f_{hj}) . \quad (40)$$

This means that in the chosen numeraire we can calculate the expectation correcting factors γ_{hi} as in equation (25) or (36) recursively starting with the last forward rate at $i = n - 1$ by

$$\gamma_{hi} = \frac{N_{\text{Branches}} \prod_{j=i+1}^{N-1} (1 + \tau_j f_{hj})}{\sum_{k=1}^{N_{\text{Branches}}} e^{p_{hi} \bar{b}_{hik}} \prod_{j=i+1}^{N-1} (1 + \tau_j f_{hj} \gamma_{hj} e^{p_{hj} \bar{b}_{hjk}})} \quad (41)$$

Clearly, it makes sense to precalculate the branching coefficients $\eta_{hjk} := e^{p_{hj} \bar{b}_{hjk}}$ and store them⁵. The above described algorithm does now exactly meet all martingale conditions. A side effect of this procedure is that it obviates the evaluation of any $\exp(\)$ function calls in the recursion procedure. For simple products, it can be easily about half of the actual computing time that is spent in the evaluation of this particular function⁶. As the above expectation correction (41) calculation does not require significantly many more floating point operations than the drift approximation (14), it is thus not surprising that the procedure presented in this section not only makes all calculations, even those with very few steps, meet the martingale conditions *exactly* but also provides a speedup by factors ranging from 1.7 to 2.8 for the tests that I conducted, depending on product type, maturities, length of the modelled yield curve, etc.

⁵For calculations without variance matching the scaling coefficients p_{hj} are, of course, all identically 1.

⁶Our benchmark tests on an i686 architecture indicate that the time taken for a single evaluation of $\exp(\)$ is in the range of 100 to 200 floating point multiplications.

9 Clustering

For most major interest rate markets, as a consequence of the prevailing rates and volatilities, the drift terms in equation (8) are comparatively small. This means that any one interest rate undergoing first an upwards and then a downwards move in two subsequent steps through the tree, appears to almost recombine at its initial level. Choosing any two forward rates on the yield curve for a two-dimensional projection on a given time slice, this produces the effect of *clustering*. This phenomenon is widely known and various methods to avoid it have been discussed in publications [MW99, Rad98a].

An example for the clustering effect is given in figure 5. Each point in the figure represents an evolved yield curve 2 years into the future. The 12-months Libor rate resetting at year 2 is along the abscissa whilst the 12-months Libor rate resetting at year 3 is given by the ordinate. In total, 4 annual forward rates were included in the modelling of the yield curve for a 6-non-call-2 annual Bermudan swaption. Using 4 factors and 6 steps until $t = 2$, there were 5 branches out of each node in the tree and a total of $5^6 = 15625$ evolved yield curves in that time slice. The initial yield curve was set at $f_i = 10\%$ for all i , and the instantaneous volatility was assumed to be 30% flat for all forward rates. As can be seen, there are only a comparatively small number of significantly different (f_0, f_1) pairs that are realised in the non-recombining tree. For the sake of brevity, I do not show any of the other projections but the reader may rest assured that the effect is just as pronounced for the remaining forward rates on the modelled yield curve.

However, if we use a more market-realistic shape for the term structure of volatility such as

$$\sigma_i(t) = k_i ([a + b(t_i - t)] e^{-c(t_i - t)} + d) \quad (42)$$

with $a = -10\%$, $b = 1$, $c = 1.5$, $d = 10\%$, and

i	k_i
0	1.179013859
1	1.319725423
2	1.458673516
3	1.57970272

(the k_i ensure that all caplets still have the same implied volatility of 30% as before), we obtain a very different diagram for the f_0 - f_1 projection at $t = 2$ as can be seen in figure 6. Therefore, for realistic applications, I do not envisage the clustering phenomenon to be an issue of foremost importance.

10 Conclusion

I have demonstrated how comparatively simple geometrical considerations can aid the construction of the branches of a non-recombining multi-factor tree model. The results show that particularly when several factors are desirable, the use of the *Alternating Simplex Direction* method in conjunction with optimal simplex alignment provides substantial benefits. In this case, the model easily converges with 5 fewer steps than needed in a plain branch construction approach. Since the computing time grows exponentially at least $\propto (N_{\text{factors}} + 1)^{N_{\text{steps}}}$, this means a speed up of, for instance, a factor of 3125 when four factors are required, 5 branches are used, and 5 fewer steps are needed due to the use of optimal alignment + ASD.

In addition to the detailed explanations of a constructive algorithm for multi-factor non-recombining trees, I also presented how the effective variance implied by the tree model can be adjusted to meet that of the analytical continuous description. Furthermore, I presented a method that guarantees that the martingale conditions are met exactly by construction. A side effect, or an added bonus, as it were, of the latter technique is an additional computation time saving of around 50%.

It should be mentioned that the methods described in this document do not resolve the problem of geometric explosion of the computational effort required for the pricing of contracts involving many exercise decisions and cashflows. However, using the techniques outlined above one can calculate the values of moderately short exercise strategy dependent contracts such as 6-non-call-2 semiannual Bermudan swaptions using many factors and achieve a comfortable level of accuracy. In fact, using multi-threading programming techniques to which the non-recombining tree algorithm is particularly amenable, I have been able to carry out overnight runs of up to ten steps for ten factors on average computing hardware (dual PII @ 300MHz). This means one can now produce benchmark results against which other numerical approximations such as exercise-strategy-parametrised Monte Carlo methods [And00] can be compared. It is mainly for this purpose that the here presented methods have been developed.

References

- [And00] L. Andersen. A simple approach to the pricing of Bermudan swaptions in the multifactor LIBOR market model. *The Journal of Computational Finance*, 3(2):5–32, Winter 1999/2000. 16
- [BDT90] F. Black, E. Derman, and W. Toy. A one-factor model of interest rates and its application to treasury bond options. *Financial Analysts Journal*, pages 33–39, Jan.–Feb. 1990. 2

- [CM99] P. Carr and D.B. Madan. Option valuation using the fast Fourier transform. *The Journal of Computational Finance*, 2(4):61–73, 1999. 8
- [CRR79] J. C. Cox, S. A. Ross, and M. Rubinstein. Option Pricing: A Simplified Approach. *Journal of Financial Economics*, 7:229–263, September 1979. 1
- [Cur94] Mike Curran. Strata gems. *RISK*, March 1994. 8
- [DK98] E. Derman and I. Kani. Stochastic Implied Trees: Arbitrage Pricing with Stochastic Term and Strike Structure of Volatility. *International Journal of Theoretical and Applied Finance*, 1(1):61–110, January 1998. 2
- [FMW98] M. C. Fu, D. B. Madan, and T. Wang. Pricing continuous Asian options: a comparison of Monte Carlo and Laplace transform inversion methods. *The Journal of Computational Finance*, 2(2):49–74, 1998. 8
- [JW00] J. James and N. Webber. *Interest rate modelling*. Financial Engineering. John Wiley and Sons, May 2000. 2
- [MW99] L. A. McCarthy and N. J. Webber. An Icosahedral Lattice Method for Three-Factor Models. Working paper, University of Warwick, 1999. 2, 2, 15
- [PTVF92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.
- [Rad98a] A. R. Radhakrishnan. An Empirical Study of the Convergence Properties of the Non-recombining HJM Forward Rate Tree in Pricing Interest Rate Derivatives. Working paper, Department of Finance, Stern School of Business, New York University, 9-190 P, 44 West 4th Street, New York, NY 10012-1126, 1998. <http://www.stern.nyu.edu/~aradhakr>. 2, 2, 15
- [Rad98b] A. R. Radhakrishnan. Does Correlation Matter in Pricing Caps and Swaptions? Working paper, Department of Finance, Stern School of Business, New York University, 9-190 P, 44 West 4th Street, New York, NY 10012-1126, 1998. <http://www.stern.nyu.edu/~aradhakr>. 2
- [Reb98] Riccardo Rebonato. *Interest Rate Option Models*. John Wiley and Sons, 1998. 3
- [Reb99] Riccardo Rebonato. *Volatility and Correlation*. John Wiley and Sons, 1999. 3, 11
- [Wil98] Paul Wilmott. *Derivatives*. John Wiley and Sons, 1998.

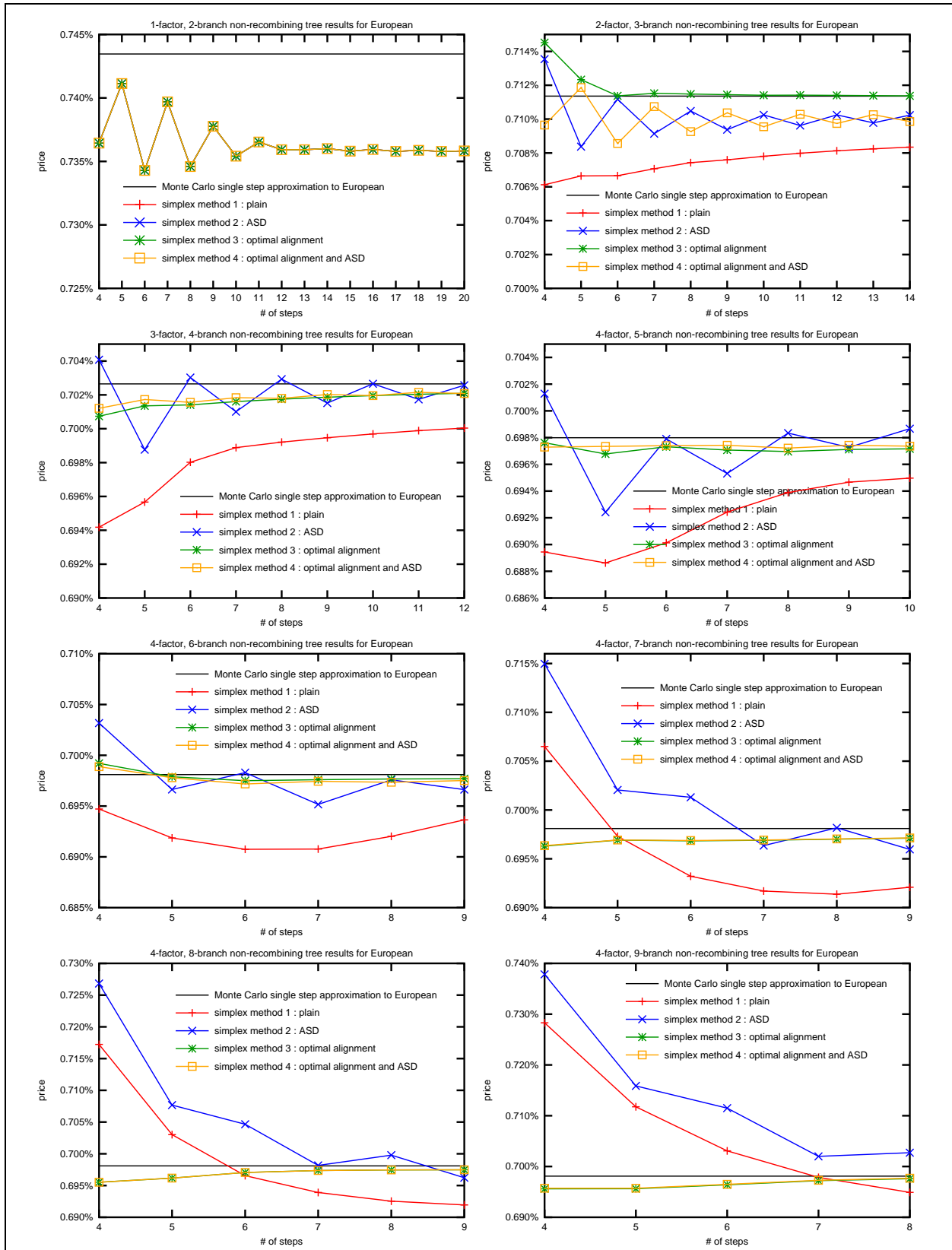


Figure 3: The convergence behaviour of the non-recombining tree for European swaptions.

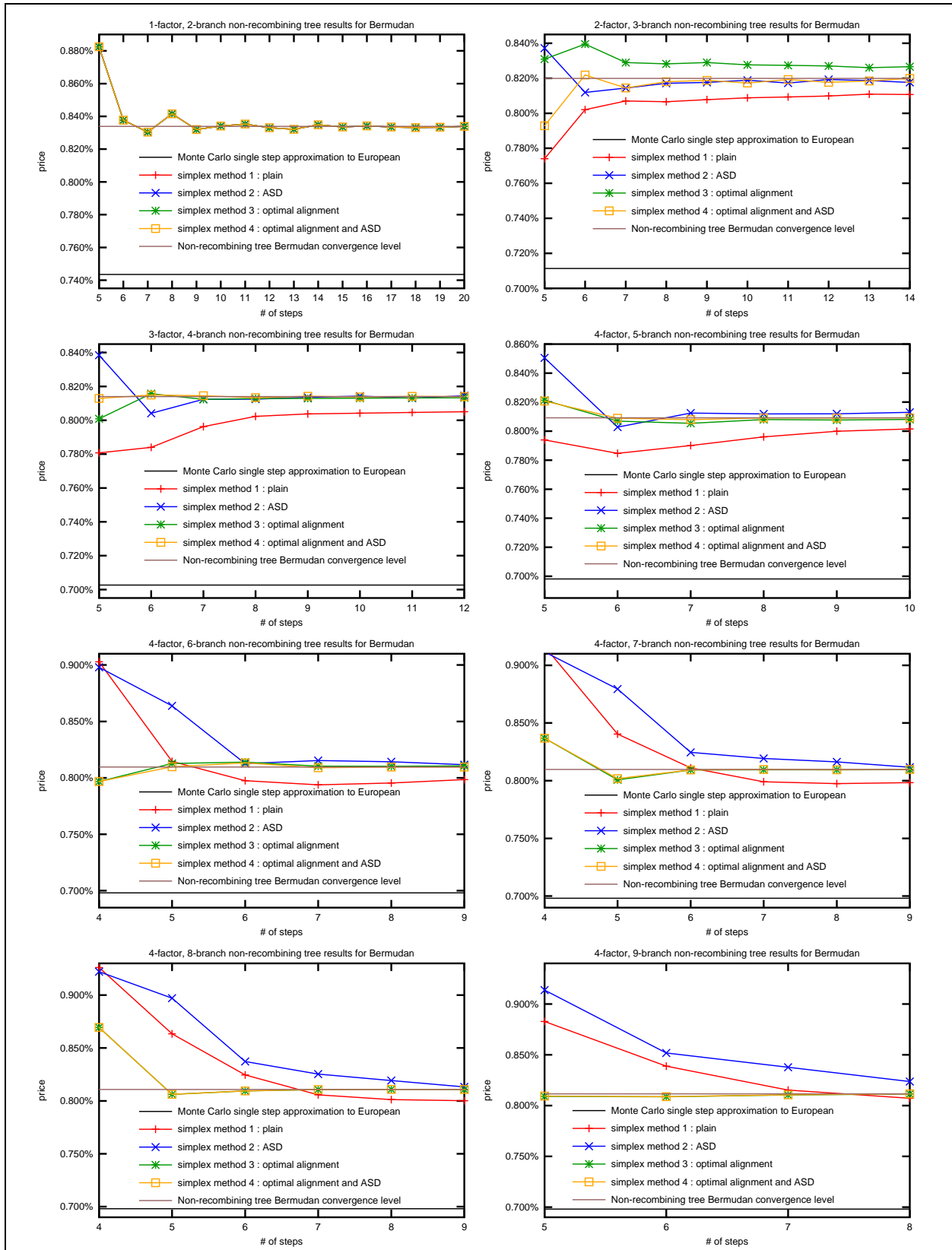


Figure 4: The convergence behaviour of the non-recombining tree for Bermudan swaptions.

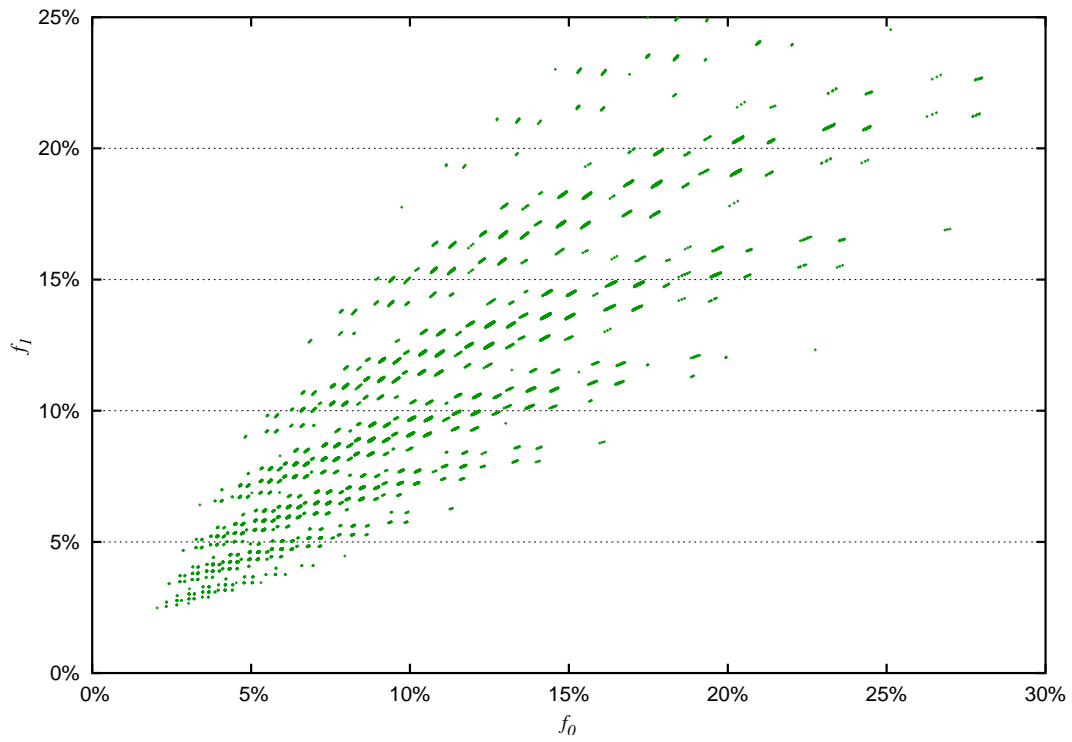


Figure 5: The clustering effect for flat volatilities.

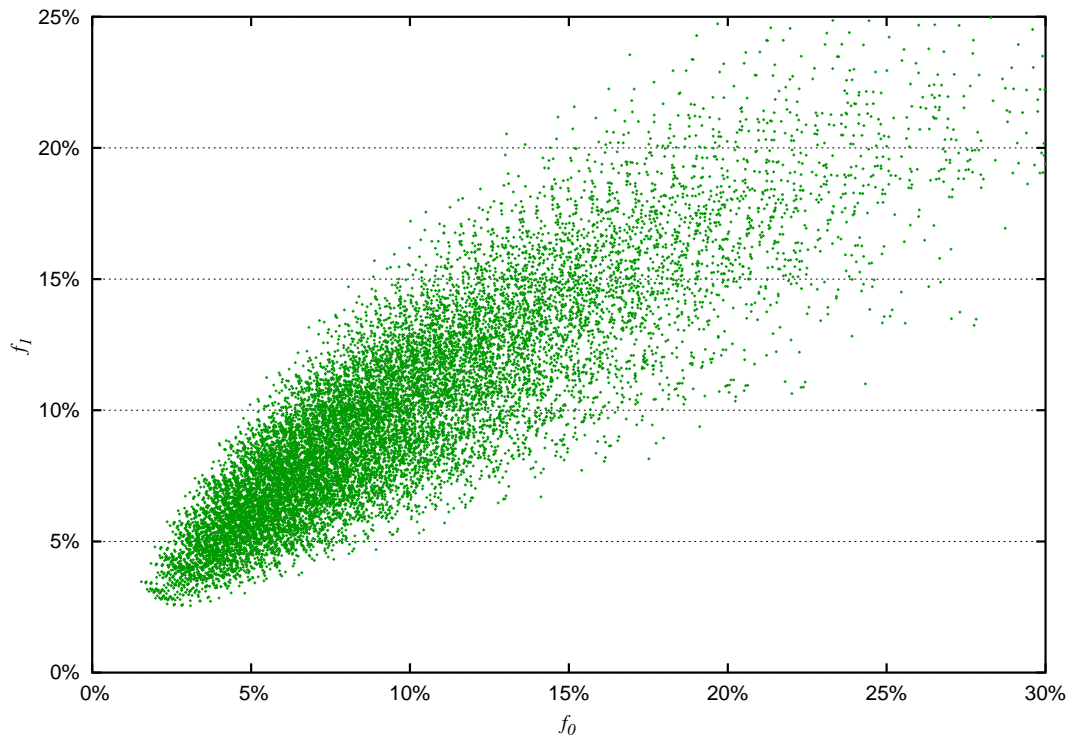


Figure 6: The clustering effect disappears for non-flat volatilities.