The LIBOR Market Model Calibration with Separated Approach

María Teresa Casparri(✉), Martín Ezequiel Masci, and Javier García-Fronti

Centro de Investigación en Métodos Cuantitativos Aplicados a la Economía y la Gestión, IADCOM, Facultad de Ciencias Económicas, Universidad de Buenos Aires, Av. Córdoba 2122, C1120AAQ CABA, Argentina
casparri@econ.uba.ar,
{martinmasci,javier.garciafronti}@economicas.uba.ar

Abstract. From an economic perspective, interest rates constitute key tools for decision making in the financial sector as they have micro and macro impacts, making its risk management a crucial matter. The LIBOR Market Model (LMM) uses the yield curve of the British interbank rate LIBOR (forward) as its basic input. Unlike models that use instantaneous rates, those involved in the LMM are observable in the market. Furthermore, the model is consistent with and adjusts its parameters according to the option valuation on futures formula in the Black '76 fashion. This allows for efficient calibration and can be used to value various derivative financial instruments. While there are several approaches for calibration, this work uses the *separated* approach with *optimization*. It is implemented using a routine in MATLAB with data of european swaptions. This work concludes that the proposed algorithm is computationally efficient and the fit is satisfactory.

Keywords: Interest rate model · Optimization algorithm · Financial derivatives valuation · Swaptions

1 Introduction

In order to illustrate the application of theoretical concepts developed in the LMM model[1], it is interesting to use swaptions on LIBOR [1]. This paper uses LIBOR market data to calibrate the LMM model by the optimized separation approach [2] and implement it in MATLAB. Model calibration consists in playing the market value of plain vanilla options [3].

Implementation poses a challenge which consists in transforming the unobservable values (such as forward rates and the rates volatility) into dynamics on observables [4]. Firstly, the parameters are set by minimizing the quadratic residues of the difference between the theoretical model and the pre-valuation of the derivatives in question. This procedure requires from the analyst to have discretion regarding the data to be used and, therefore, market information needs to be available [5]. The remaining parts of the model's implementation process, are post-calibration which it reflects the use of the

[1] For a more detailed development about stochastic calculus, see [1].

R. León et al. (Eds.): MS 2016, LNBIP 254, pp. 12–21, 2016.
DOI: 10.1007/978-3-319-40506-3_2

results, either for valuation of derivatives [6], or the associated cash flow, and in a context of less uncertainty.

The objective of this work[2] is to calibrate the LMM model using the separated approach and it is divided into two sections. The first one explains the LMM model and how to calibrate it. The latter applies this methodology to a database and calculates the parameters of the model in this context. After this, a conclusion is given, and future research is proposed.

2 The Model

Calibration of the LMM model implies finding parameters σ_i, $i = 1, \ldots, N$. This parameters represent the volatility of certain derivative. In this research paper, swaptions on LIBOR rate agreements are used [6, 7]. For MATLAB, this procedure can take between less than a minute and 15 min, depending on both the functional forms, and the number of iterations of the process. It is important to note that the process of calibrating is not something that should be done only once. An investor who uses these techniques to enhance their portfolio, should additionally think about how often it needs to re-calibrate the model [8, 9].

This section will be exposing some theoretical guidelines that justify the use of this approach for the calibration of the LMM model. This guidelines are taken from the book of Gatarek et al. [10] in which the model is studied in great detail. First, it is necessary to define the approach and the steps to outline the algorithm, which is then translated into computer language using MATLAB (2007a version).

As indicated by the authors mentioned, the algorithm belongs to the class of non-parametric calibration. The use of a matrix of volatilities allows for a straightforward implementation. The aim is to construct a covariance matrix forward LIBOR rate. To do this it will be necessary to compute different variants of the same approach depending on the set of parameters to be estimated: Λ_i. The relevant parameters are an array of lambdas associated with swaptions with various maturities [11]. An intermediate step is the construction of the covariance matrix (VCV) forward LIBOR rate by use of eigenvalues and eigenvectors. Here, sub-routines are necessary to correct data errors, so that we can assure the VCV is a positive definite matrix form. After this, one may proceed to minimize the mean square error between the theoretical valuation and market values.

The process creating a matrix of volatilities of swaptions [10] begins:

$$\sum^{SWPT} = \begin{bmatrix} \sigma^{swpt}_{1,2} & \sigma^{swpt}_{1,3} & \sigma^{swpt}_{1,4} & \cdots & \sigma^{swpt}_{1,m+1} \\ \sigma^{swpt}_{2,2} & \sigma^{swpt}_{2,4} & \sigma^{swpt}_{2,5} & \cdots & \sigma^{swpt}_{2,m+2} \\ \sigma^{swpt}_{3,4} & \sigma^{swpt}_{3,5} & \sigma^{swpt}_{3,6} & \cdots & \sigma^{swpt}_{3,m+3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \sigma^{swpt}_{m,m+1} & \sigma^{swpt}_{m,m+2} & \sigma^{swpt}_{m,m+3} & \cdots & \sigma^{swpt}_{m,M} \end{bmatrix}_{m \times m} \tag{1}$$

[2] We appreciate the comments and help from Pablo Matías Herrera and Joaquín Bosano.

Each matrix component, $\sigma_{i,j}^{swpt} = \sigma^{swpt}(t, T_i, T_j)$ is the volatility of a *swaption* with *maturity* T_i (assuming an underling swap T_i, T_j).

After that, it is possible to define the covariance matrix for the forward LIBOR as:

$$\Phi^i = \begin{bmatrix} \varphi_{1,1}^i & \varphi_{1,2}^i & \varphi_{1,3}^i & \cdots & \varphi_{1,m}^i \\ \varphi_{2,1}^i & \varphi_{2,2}^i & \varphi_{2,3}^i & \cdots & \varphi_{2,m}^i \\ \varphi_{3,1}^i & \varphi_{3,2}^i & \varphi_{3,3}^i & \cdots & \varphi_{3,m}^i \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \varphi_{m,1}^i & \varphi_{m,2}^i & \varphi_{m,3}^i & \cdots & \varphi_{m,m}^i \end{bmatrix}_{m \times m} \tag{2}$$

Source: Gatarek et al. [10].

Where

$$\varphi_{kl}^i = \int_0^{T_i} \sigma^{inst}(t, T_{l-1}, T_l)\sigma^{inst}(t, T_{k-1}, T_k)dt; \quad i < k \,\Lambda\, i < l \tag{3}$$

And $\sigma^{inst}(t, T_{l-1}, T_l)$ is the instant volatility of the LIBOR rate $L_l(t, T_{l-1}, T_l)$.

It is assumed that parameters Λ_i exist, such that:

$$\varphi_{kl}^i = \Lambda_i \varphi_{kl} \tag{4}$$

Assuming that $\Lambda_i = \delta_{0,k} \quad \forall k = 1, \ldots, m$, it is possible to calculate parameters of the principal diagonal of $\Phi_{m \times m}$:

$$\varphi_{kk} = \frac{\delta_{0,k} \cdot \sigma^{swpt}(t, T_k, T_{k+1})^2}{\Lambda_k} \tag{5}$$

The next step is to define parameters $R_{i,j}^k(t)$

$$R_{i,j}^k(t) = \frac{B(0, T_{k-1}) - B(0, T_k)}{B(0, T_i) - B(0, T_j)} \tag{6}$$

Where $B(0, T_n)$ $n = 1, \ldots, M$ are the LIBOR discount factors, and vector B is defined as:

$$B = \begin{pmatrix} B(0, T_1) \\ \cdots \\ B(0, T_M) \end{pmatrix} \tag{7}$$

Please note that $R_{i,j}^k(t)$ depends on k, i, j and on the maturities selected in the calibration.

With these new definitions it is now possible to calculate the rest of matrix:

$$\varphi_{k,N-1} = \\ = \frac{\delta_k \cdot \sigma^2_{k,N} - \Lambda_k\left(\sum_{l=k+1}^{N}\sum_{i=k+1}^{N} R^i_{k,N}(0) \cdot \varphi_{i-1,l-1} \cdot R^l_{k,N}(0) - 2 \cdot R^{k+1}_{k,N}(0) \cdot \varphi_{k,N-1} \cdot R^N_{k,N}(0)\right)}{2 \cdot \Lambda_k \cdot R^{k+1}_{k,N}(0) \cdot R^N_{k,N}(0)}$$

Gatarek et al. [10] use the model proposed by Longstaff et al. [12] to give a numerical solution. Initially the authors proposed the creation of a new covariance matrix called Φ^M, similar to the original matrix but without the eigenvectors associated with negative eigenvalues.

This procedure consists in multiplying each eigenvectors (e) by the square root of the associated eigenvalue $(\sqrt{\lambda_i})$. This is only done for positive λ_i. Then, a matrix is built with this changes. The matrix Φ^M is then, the result of the product between the modificated Φ^i by its transpose. This procedure fixes any problem related to negative values.

The development of this sub algorithm allows us to build the swaptions' theoretical values matrix by approaching their volatilities using, initial values of Φ^M.

Each component follows Gatarek et al. [10]:

$$\begin{gathered} \varphi^{M^i}_{kl} = \Lambda_i \varphi^M_{kl} \\ \delta_k \sigma^2_{k,N} \cong \Lambda_k \sum_{l=k+1}^{N}\sum_{i=k+1}^{N} R^i_{k,N}(0) \cdot \varphi^M_{i-1,l-1} \cdot R^l_{k,N}(0) \end{gathered} \tag{8}$$

Since $\delta_k = \Lambda_k$ (Logstaff-Schwartz-Santa Clara model), then:

$$\sigma^2_{k,N} \cong \sum_{l=k+1}^{N}\sum_{i=k+1}^{N} R^i_{k,N}(0) \cdot \varphi^M_{i-1,l-1} \cdot R^l_{k,N}(0) \tag{9}$$

Towards this assumptions, it is possible to build market volatilities and address the accuracy by the root mean square error between theoretical assumptions and market values:

$$RMSE = \sum_{i,j=1}^{m}\left(\sigma^{Theoretical}_{ij} - \sigma^{Market}_{ij}\right)^2 \tag{10}$$

With this expression described above, the separated approach can be optimized by nonlinear functions. RSME admits minimum only if VCV is defined positive. The theoretical model explains better the reality of the market if minimization is made. In the following section, the developed algorithm is shown with real data using MATLAB.

3 Calibration for European Swaptions

First of all, in order to implement LMM with real data it should be taken into consideration three steps: calibration, derivative payoff and cash flows related to the option. This work examines the procedure to obtain the parameters needed in the first step.

In the MATLAB routine it is possible to program a recursive process in six steps. The recursive character is configured by the fact that each step needs information of previous steps. The next figure shows all the process, data input and output required (Fig. 1):

Step	*Inputs*	*Outputs*
1	Vector of Discount Factors **[B]**	The Matrix of Parameters **[R]**
2	i. **[R]** ii. Vector of Date **[T_num]** iii. Matrix of market swaption volatilities **[Sig]** iv. Vector of initial parameters **[Lambda]**	The matrix of covariances [**VCV]** as a function of parameters **[Lambda]**
3	**[VCV]**	i. The vector of EigenValues **[L]** as a function of parameters [Lambda] ii. The parix of eigenvestors **[E]** as a function of parameters [Lambda]
4	i. **[L]** ii. **[E]**	The modified covariance matrix **[VCV_M]**
5	i. **[R]** ii. **[VCV_M]**	Theoretical swaption volatilities **[Sig_theo]**
6	i. **[Sig_theo]** ii. **[Sig]**	RSME between theoretical and market swaption volatilities

Fig. 1. Algorithm – separated approach. (Source: compiled by authors based on Gatarek et al. [10])

As it was mentioned above, the recursive character is illustrated by the dotted arrow. The output data in each step is treated as input data of the next one.

This process is known as the separated approach and some initial data is necessary in order to execute the algorithm[3]: vector of dates, swaption market volatilities matrix and parameters vector (all belonging to step 2). Regarding to the first one, it is a column vector with schedule data according to the instruments tenors. If those *maturities* are expressed in years, vector has *M* dates of the same initial date but within a year difference. This paper is based on Gatarek et al. [10] data (European Swaptions from January 2005).

[3] A discount factor vector needs to be specified as well, this is easily calculated from interest rates and tenors.

As initial data it is necessary to put on a parameter vector: Λ_i. Taking into consideration this performing issues:

i. If an arbitrary vector is imposed, the MATLAB code will return an RSME that matches with this initial condition. Hardly this RSME will be the local minimum for all the problem.
ii. Rather than including an arbitrary vector *[Lambda]*, an optimization in which each component of the vector are control variables of the mentioned problem should be performed.

This is why the separated approach with optimization is clearly superior than specifying an arbitrary vector. An initial condition is used as a vector "Lambda0" which components goes between 1 to 10 (in this case). In MATLAB it is allowed a function fminsearch to solve the mean square error as a nonlinear optimization:

```
options=optimset('MaxIter',1000);
Lambda0=[1 2 3 4 5 6 7 8 9 10];
[Lambda,f]=fminsearch(@FunciónObjetivo,Lambda0,opti
ons);
```

Then, routine will run an optimization performance with some objective values. The output it yields is the minimized RSME and an initial parameters vector which matches with this minimized error. Syntax, in this case, has RSME as objective function in its arguments. Initial parameters vector (Λ_i) is a constraint.

At the same time, since the process consists of a minimization with a searching objective, it is possible to set a specific number of iterations to generate a faster solution. This paper concludes that 10.000 iterations are enough to find a local minimum in our problem. To prove this affirmation, the program was run without any iterations imposed. Results were the same in both, the restricted and the unrestricted cases.

Since Fig. 2, step 3 generates the eigenvalues vector and eigenvectors matrix as Λ_i functions (Fig. 3):

Eigenvectors (L):

$$
\begin{array}{l}
L = \\
\quad [-0.0147 \\
\quad 0.0006 \\
\quad 0.0179 \\
\quad 0.0251 \\
\quad 0.0339 \\
\quad 0.0506 \\
\quad 0.0727 \\
\quad 0.1107 \\
\quad 0.1547 \\
\quad 0.2447]
\end{array}
$$

VCV=

k/l (years)	1	2	3	4	5	6	7	8	9	10
1	0,0312	0,0109	0,0044	0,0022	0,0012	0,0007	0,0005	0,0003	0,0002	0,0002
2	0,0109	0,0454	0,0256	0,0102	0,0051	0,0029	0,0018	0,0012	0,0008	0,0006
3	0,0044	0,0256	0,0528	0,0377	0,0149	0,0075	0,0042	0,0027	0,0018	0,0012
4	0,0022	0,0102	0,0377	0,0545	0,0489	0,0194	0,0097	0,0056	0,0035	0,0024
5	0,0012	0,0051	0,0149	0,0489	0,0679	0,0462	0,0183	0,0095	0,0053	0,0035
6	0,0007	0,0029	0,0075	0,0194	0,0462	0,0656	0,0587	0,0236	0,0119	0,0068
7	0,0005	0,0018	0,0042	0,0097	0,0183	0,0587	0,0638	0,0695	0,0274	0,014
8	0,0003	0,0012	0,0027	0,0056	0,0095	0,0236	0,0695	0,0835	0,0582	0,0234
9	0,0002	0,0008	0,0018	0,0035	0,0053	0,0119	0,0274	0,0582	0,0959	0,0535
10	0,0002	0,0006	0,0012	0,0024	0,0035	0,0068	0,014	0,0234	0,0535	0,1359

Fig. 2. The matrix of covariances [VCV] as a function of parameters (Logstaff-Schwartz-Santa Clara model)

E=

i	e_1i	e_2i	e_3i	e_4i	e_5i	e_6i	e_7i	e_8i	e_9i	e_10i
1	-0,0006	0,0213	-0,267	0,6756	-0,6152	-0,2504	-0,1615	-0,0583	0,0336	0,0107
2	0,0068	-0,1474	0,4869	-0,3956	-0,3148	-0,4639	-0,4567	-0,208	0,1296	0,0414
3	-0,0385	0,4361	-0,469	-0,0618	0,3768	-0,1297	-0,4712	-0,3531	0,2655	0,0923
4	0,1045	-0,6733	0,0558	0,2668	0,2121	0,2931	-0,1007	-0,3716	0,394	0,1646
5	-0,223	0,4878	0,3641	0,0431	-0,2504	0,3201	0,3005	-0,2474	0,4546	0,2337
6	0,4932	-0,0833	-0,371	-0,3318	-0,1948	-0,2442	0,3881	0,116	0,3601	0,3329
7	-0,6891	-0,1889	-0,064	0,0496	0,1717	-0,323	0,0691	0,3643	0,1652	0,4292
8	0,4525	0,2151	0,3562	0,3145	0,2348	0,0112	-0,2793	0,3958	-0,0495	0,4805
9	-0,1207	-0,0849	-0,2517	-0,3015	-0,3655	0,5259	-0,3248	0,0289	-0,3142	0,4569
10	0,018	0,016	0,0638	0,0882	0,1292	-0,2761	0,3226	-0,5707	-0,54	0,4162

Fig. 3. Eigenvectors

Regarding the filter described above, it is possible to take away all eigenvectors associated with negative eigenvalues. In this case, only the first eigenvalue is negative. The VCV_M matrix or Φ^M is shown in Fig. 4:

VCV_M=

(years)	1	2	3	4	5	6	7	8	9	10
1	0,0312	0,0109	0,0044	0,0022	0,0012	0,0007	0,0005	0,0003	0,0002	0,0002
2	0,0109	0,0454	0,0256	0,0102	0,0051	0,0029	0,0017	0,0012	0,0008	0,0006
3	0,0044	0,0256	0,0528	0,0376	0,015	0,0072	0,0046	0,0024	0,0018	0,0012
4	0,0022	0,0102	0,0376	0,0547	0,0485	0,0202	0,0087	0,0063	0,0033	0,0024
5	0,0012	0,0051	0,015	0,0485	0,0686	0,0446	0,0206	0,008	0,0057	0,0034
6	0,0007	0,0029	0,0072	0,0202	0,0446	0,0692	0,0537	0,0269	0,011	0,007
7	0,0005	0,0017	0,0046	0,0087	0,0206	0,0537	0,0708	0,0649	0,0287	0,0138
8	0,0003	0,0012	0,0024	0,0063	0,008	0,0269	0,0649	0,0866	0,0574	0,0235
9	0,0002	0,0008	0,0018	0,0033	0,0057	0,011	0,0287	0,0574	0,0961	0,0535
10	0,0002	0,0006	0,0012	0,0024	0,0034	0,007	0,0138	0,0235	0,0535	0,1359

Fig. 4. The modified covariance matrix [VCV_M] as a function of parameters

It is possible to generate theoretical volatilities matrix according to the last matrix and [R] matrixes calculated in the beginning of the algorithm. This is shown in formula (9).

Taking into account all the important information, the following outputs show theoretical swaption volatilities, the observables one (market information) and their differences (Figs. 5 and 6):

Sig_theo=

Theoretical	1	2	3	4	5	6	7	8	9	10
1	22,70%	20,29%	19,96%	19,81%	19,94%	19,57%	19,24%	19,17%	18,89%	18,63%
2	22,40%	20,34%	19,26%	18,83%	18,06%	17,47%	17,19%	16,78%	16,42%	
3	20,90%	19,45%	18,70%	17,55%	16,72%	16,28%	15,76%	15,32%		
4	19,53%	19,61%	18,03%	16,93%	16,36%	15,69%	15,14%			
5	18,30%	16,64%	15,57%	14,96%	14,23%	13,64%				
6	17,93%	16,95%	16,34%	15,33%	14,54%					
7	17,61%	17,74%	16,45%	15,44%						
8	16,27%	15,07%	14,07%							
9	15,26%	14,28%								
10	14,50%									

Fig. 5. Calculation of theoretical swaption volatilities [Sig_theo]

Sig=

Market	1	2	3	4	5	6	7	8	9	10
1	22,70%	23,00%	22,10%	20,90%	19,60%	18,60%	17,60%	16,90%	16,30%	15,90%
2	22,40%	21,50%	20,50%	19,40%	18,30%	17,40%	16,70%	16,20%	15,80%	
3	20,90%	20,10%	19,00%	18,00%	17,00%	16,30%	15,80%	15,50%		
4	19,50%	18,70%	17,70%	16,80%	16,00%	15,50%	15,10%			
5	18,20%	17,40%	16,50%	15,80%	15,10%	14,80%				
6	17,46%	16,74%	15,90%	15,24%	14,62%					
7	16,72%	16,08%	15,30%	14,68%						
8	15,98%	15,42%	14,70%							
9	15,24%	14,76%								
10	14,50%									

Fig. 6. Matrix of market swaption volatilities [Sig]. (Source: compiled by authors based on Gatarek et al. [10]).

Hence, RSME between theoretical and market swaption volatilities (Eq. (10)) is:

$$\boldsymbol{RSME} = \sum_{i,j=1}^{m} \left(\sigma_{ij}^{Theoretical} - \sigma_{ij}^{Market}\right)^2 = \mathbf{0.005336}$$

After running the program, initial parameters vector (control variables of the problem) are obtained:

$$\text{Lambda} = (\mathbf{1.6495} \quad \mathbf{2.2113} \quad \mathbf{2.4858} \quad \mathbf{2.7925} \quad \mathbf{2.4413} \quad \mathbf{2.7904} \quad \mathbf{3.0697} \quad \mathbf{2.4470} \quad \mathbf{2.1824} \quad \mathbf{1.5488})$$

As was described, this outcome results in an optimization that matches RSME with the corresponded parameters. This information is an input to valuate the financial derivative and its future prices curve. If the proposed algorithm is computationally efficient and the fit is satisfactory (in the RSME sense), this mentioned future prices curve will be certainly close to the actual future prices. Algorithm complexity has sensitivity to data input and, specifically, to the derivatives considered. The higher the tenor considered, and the lesser partial autocorrelation has the time series, the lesser will be the prediction power of the parameters found.

It is possible to run the program with 100 and 1.000 iterations. The minimum reached is better when iterations increase. In the first case, RSME is 0.091659, while in second case 0.008107.

$$\underbrace{0.091659}_{RSME_{100}} > \underbrace{0.008107}_{RSME_{1.000}} > \underbrace{0.005336}_{RSME_{10.000}}$$

A RSME closer to zero means that LMM fits better to market data. A perfect fit is hard to achieve and has no sense to several financial analysts since reality has uncertainty and risks [13]. There is a challenge in reducing uncertainty of future events and using financial models for portfolio rebalance or creating value through investment strategies.

4 Conclusion

The LMM model aims to minimize root mean squared errors between the theoretical value and market data of European swaptions. This work analysed a calibration of this model using the methodology known as *separated approach with optimization* and implemented it in MATLAB.

This is especially important to understand the dynamics of the financial markets. In this complex environment, future is unpredictable, but investors could limit their risk exposure implementing routines like the one shown in this paper. The objective from the point of view of financial engineering is to assess the discrepancy between the market value and the theory of a financial asset, and minimize it using the "root mean squared error" methodology. Moreover, an investor who uses these techniques must, additionally, reflect on how often it should re-calibrate the model.

This work concluded that the proposed algorithm is computationally efficient and the fitting is appropriate. Particularly, the results of the non-linear minimization are sensitive to the number of iterations specified. Real-time valuation compromises its accuracy. Moreover, this procedure is not appropriate for valuing complex portfolios that include exotic derivatives due to the computational power needed for this type of valuations.

Future research aims to develop a framework for investors to evaluate different financial derivatives using LMM and also expand research on how to accelerate the algorithm presented.

References

1. Masci, M.E.: Cálculo Estocástico y Calibración en Modelos de Tasa de Interés: Aplicación al Mercado LIBOR. Masters degree thesis, Facultad de Ciencias Económicas, Universidad de Buenos Aires (2015)
2. Brace, A., Gatarek, D., Musiela, M.: The market model of interest rate dynamics. Math. Financ. **7**, 127–155 (1997)
3. Löeffler, G., Posch, P.N.: Credit Risk Modeling Using Excel and VBA. Wiley, London (2007)
4. Glasserman, P.: Monte Carlo Methods in Financial Engineering, vol. 53. Springer Science & Business Media, New York (2003)

5. Korn, R., Liang, Q.: Robust and accurate Monte Carlo simulation of (cross-) Gammas For Bermudan swaptions in the LIBOR market model. CFin. **17**, 87 (2014)
6. Andersen, L.B.G.: A simple approach to the pricing of Bermudan swaptions in the multi-factor LIBOR market model. SSRN 155208 (1999)
7. Bessis, J.: Risk Management in Banking. Wiley, West Sussex (2011)
8. Andersen, L., Piterbarg, V.: Interest Rate Modelling Volume 1: Foundations and Vanilla Models. Atlantic Financial Press, London (2010)
9. Andersen, L., Piterbarg, V.: Interest Rate Modelling Volume 2: Term Structure Models. Atlantic Financial Press, London (2010)
10. Gatarek, D., Bachert, P., Maksymiuk, R.: The LIBOR Market Model in Practice. Wiley, London (2007)
11. Leippold, M., Stromberg, J.: Time-changed Lévy LIBOR market model: pricing and joint estimation of the cap surface and swaption cube. JFE **111**, 224–250 (2014)
12. Longstaff, F.A., Santa-Clara, P., Schwartz, E.S.: Throwing away a billion dollars: the cost of suboptimal exercise strategies in the swaptions market. JFE **62**, 39–66 (2001)
13. Crouhy, M., Galai, D., Mark, R.: The Essentials of Risk Management. McGraw-Hill, New York (2006)

http://www.springer.com/978-3-319-40505-6

Modeling and Simulation in Engineering, Economics and Management
International Conference, MS 2016, Teruel, Spain, July 4-5, 2016, Proceedings
León, R.; Muñoz-Torres, M.J.; Moneva, J.M. (Eds.)
2016, XII, 205 p. 49 illus., Softcover
ISBN: 978-3-319-40505-6