Distributional Approximation of Asset Returns with Nonparametric Markovian Trees

Gaetano Iaquinta,[†] *and Sergio Ortobelli Lozza*[††]*,*

University of Bergamo, Department MSIA, Italy

Summary
The paper proposes the use a Markov chain to model and predict the distributional behaviour of a portfolio of returns. In particular, it describes an algorithm to compute the distribution of returns that follow a markovian tree. This approach reduces the computational complexity as compared to the classic markovian approach, since the tree recombines at each temporal step. Furthermore, the paper compares *ex-post* the assumption that returns follow either a geometric Brownian motion or a Markov chain. Finally, it discusses some possible financial applications of the proposed approach.

Key words:
Computational complexity, Markov chain, return distribution, financial applications.

Introduction

In this paper, we propose an algorithm to compute the return distribution at a fixed future time assuming that the return follows a Markov chain. Thus, we discuss the computational complexity of the algorithm and we compare its forecasting power with the classic assumption that continuous compounded returns follow a Brownian motion.

It is well known that asset returns are not Gaussian distributed. However, most of the financial models used by institutional investors and market operators are based on this distributional assumption. In order to overcome the limits of this assumption several alternative financial models have been widely studied in past and recent literature. Most of them are based on Markov processes or Semi-Markov processes (see [1], [3], [4], [7], [8]). As a matter of fact, if we test the null hypothesis that the intervals of the distributional support of a given portfolio are independent against the hypothesis that the intervals follow a Markov chain, we cannot reject the markovian hypothesis (see [2]). Among markovian models we can distinguish parametric and nonparametric approaches. The first ones capture the markovian behavior of asset prices assuming that they follow a particular diffusion process. While nonparametric approaches use historical time series to predict the probability distribution [9]. Here, we propose a nonparametric markovian approach to model asset returns.

In the paper we assume that the interval dependence of portfolios of returns can be characterized by a Markov chain. In particular, we divide the support of the portfolio into N intervals each of which represents a state of the Markov chain. Following this procedure, we build up the transition matrix. In order to minimize the computational complexity of the model, we choose opportunely the states and we get that the gross returns follow a tree, whose number of nodes grows linearly with the time. On the other hand, recombining trees are typically used in option pricing theory. Thus we propose an algorithm with polynomial complexity that permits us to compute the distribution of the returns after k periods of time. Then we *ex-post* compare the return distributional approximation with the hypothesis that gross return are log normal distributed. (i.e., continuous compounded returns follow a Brownian motion). In particular, we first test the ex-post distributions and then we analyze the tails of the return distributions.

The paper is organized as follows: in Section 2 we formalize the markovian nonparametric approach. Section 3 deals with the algorithm to approximate the return distribution. In Section 4 we test the algorithm as compared to the classical geometric Brownian motion. Finally, we briefly summarize the paper and we describe further financial applications of this approach.

2. Nonparametric markovian trees

Let us assume that the gross return of a portfolio of assets z_p has support on the interval $(\min_t z_{p,t}, \max_t z_{p,t})$, where $z_{p,t} = S_{p,t+1} / S_{p,t}$ is the t-th return observation and $S_{p,t}$ is the value of the portfolio of securities at time t. By convention, through all the paper, we assume that the return follows a Markov chain with N states and we count the states beginning from that with the greatest value. Then we build the transition matrix as follows:

1. we share in N intervals $I_i = (a_i, a_{i-1})$ (small enough) the return support $(\min_t z_{p,t}, \max_t z_{p,t})$ where $a_0 = \max_t z_{p,t}$, $a_i = u^i \max_t z_{p,t}$, $u = \sqrt[N]{\min_t z_{p,t} / \max_t z_{p,t}}$ and i=1,...,N;

Manuscript received November 5, 2006.
Manuscript revised November 25, 2006.

2. we assume that inside the interval $I_i = (a_i, a_{i-1})$ the return is given by the geometric mean of the extremes, i.e., $z^{(i)} := \sqrt{a_{i-1}a_i} = u^{i-0.5} \max_t z_{p,t}$;

3. we build the transition matrix $P_s = [p_{ij,(s)}]_{1\le i,j\le N}$ where the probability $p_{ij,(s)}$ points out the probability valued at time s to transit from the state $z^{(i)}$ to the state $z^{(j)}$ conditional of being in the i-th state.

Considering that after k periods the gross return should be $S_{p,k} / S_{p,0} = \prod_{i=1}^{k} z_{p,i}$ then we can represent the price after a given number of periods with a markovian tree that recombines at each step (see Figure 1).

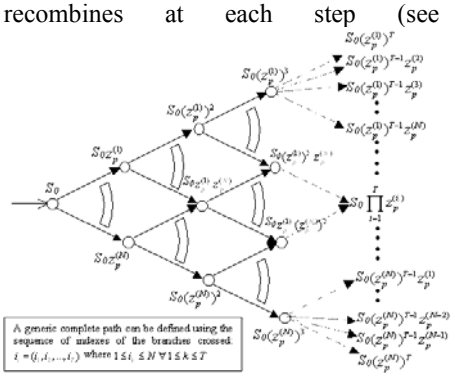

Fig. 1. A generic recombining markovian tree.

Therefore, the number of nodes increases linearly with the number of the time steps. This is the main reason according to which we can simplify the computation of the probability at each node of the tree. As a matter of fact, with a Markov chain that does not recombine at each step the number of nodes of the markovian tree after k periods should be N^{2k}. Thus the computation of the return distribution after k periods could be very complex for a large number of states N. With our proceeding we clearly reduce and control the computational complexity in comparison with the classic markovian approach. In particular, after $k\Delta t$ intervals of time we have $(N\text{-}1)k+1$ nodes (i.e., the multinomial tree grows linearly with the time). Starting to count from the highest node, after k steps the *j-th* node has:

a) value in the interval:

$$I_j^{(k)} = (u^{j+(k-1)/2}(\max_t z_{p,t})^k, u^{j+(k-3)/2}(\max_t z_{p,t})^k)$$

b) gross return:

$$z_k^{(j)} = u^{j-1+k/2}(\max_t z_{p,t})^k;$$

c) stock price:

$$S_0 z_k^{(j)} \quad j=1,\ldots,(N\text{-}1)k+1.$$

Next we discuss an algorithm with polynomial complexity to get the probability at each node of the markovian tree.

3. An algorithm to approximate the return distribution with a markovian tree

One of the fundamental aspects in any financial valuation is the estimation of the return distribution at a given future time. In this section we create a recursive algorithm that gives us the distribution of returns after k periods of time with a computational complexity of $O(N^3k^2)$.

The procedure to compute the distribution function of the future returns is strictly connected to the Markovian hypothesis and the recombining feature of the tree. It allows us to maintain distinguishable the state of the provenance for each node. Clearly, once we distinguish the provenance state, we can easily compute the probabilities for the other steps using the properties that characterize any Markov chain. In particular, we build a sequence of matrixes Q_k of dimension $((N\text{-}1)k+1)\times N$ such that, after k periods of time, the return probabilities in the $(N\text{-}1)k+1$ nodes of the tree are given by the vector

$$Q_k \square 1_N \qquad (1)$$

where 1_N is the unity vector column. Note that each node of the tree is simultaneously achievable from different states. Thus, each node could be in different states and this depends on the provenance state. In this process we have that $Q_k = [q_{(k),j,i}]_{\substack{1\le j\le (N-1)k+1 \\ 1\le i\le N}}$, where $q_{(k),j,i}$ is the probability of being in the *i-th* state and in the *j-th* node (counting from the highest node) after k periods of time. Therefore in order to obtain the probability to move in other states, we have to multiply Q_k by the transition matrix. However, since each final node is not achievable from all the states, we have to take into account the null probability of a given node to be in a particular state. This step of the algorithm is performed by a "diagonalization process" that we explain here in the following. Suppose we know the initial state, say the *i-th* one, then the probabilities of each node after one period are given by the *i-th* row of the first transition matrix. Therefore, the first matrix is the $N\times N$ matrix with these probabilities on the diagonal, i.e., $Q_1 = diag(p_{i1,(1)},\ldots,p_{iN,(1)})$. In order to derive the Q_{k+1} matrix, we multiply the matrix Q_k by the transition matrix P_{k+1} and then we apply the following "*diagonalization process*" to the resulting matrix $Q_k \square P_{k+1}$:

1. we shift below the *s-th* column of s-1 spaces for $s=2,\ldots,N$, creating the new matrix $((N\text{-}1)(k+1)+1)\times N$;
2. we fill all the new spaces with zeros.

This diagonalization process is performed by a matrix operator that we call *diagM* operator, i.e.,

$$Q_{k+1} = diagM(Q_k \square P_{k+1}). \qquad (2)$$

Observe that each zero inserted in the diagonalizaton process represents the null probability of a given node to

be in a particular state. Let us consider the following representative example.

Table 1: This table summarizes the complexity of computing $Q_{k+1} = diagM(Q_k P)$

Computational complexity of $diagM(A)$, where A is a $((N-1)i+1)\times N$ matrix and i is the number of steps.

Micro operations	Step 1	…	Step k	Total
Memorize dimensions	1	…	1	*k*
Add N rows of zeros	$2N^2$	…	$(N-1)Nk+N+N^2$	$k(\frac{k+1}{2}N(N-1)+$ $+N^2+N)$
Recompute dimensions	1	…	1	*k*
Vectorial reshape	1	…	1	*k*
Deletion last N elements	1	…	1	*k*
Matrix reshape	1	…	1	*k*
Total	$2N^2+5$	…	$(N-1)Nk+N+N^2+5$	$k(\frac{k+1}{2}N(N-1)+$ $+N^2+N+5)$
Total O(.)				$O(k^2N^2)$

Computational complexity of $Q_{k+1} = diagM(Q_k P)$

	Number of multiplications	Number of additions	Number micro oper. of diagM
Step 1	N^3	$N^2(N-1)$	$2N^2+5$
Step 2	$2N^3-N^2$	$N(N-1)(2N-1)$	$3N^2-N+5$
…	…	…	…
Step k	$N^2((N-1)k+1)$	$N(k(N-1)^2+(N-1))$	$N((N-1)k+1)+N^2$
Total O(.)	$O(k^2N^3)$	$O(k^2N^3)$	$O(k^2N^2)$

Example. Assume the return evolves following a simple homogeneous trinomial tree (i.e., the Markov chain is homogeneous and the number of states is 3). Let P be the transition matrix of dimension 3x3. Suppose at time zero the initial state is the first one. Then the first matrix is given by $Q_1 = diag(p_{(1,1)}, p_{(1,2)}, p_{(1,3)})$ and the probability at each node after one period is given by the vector $Q_1 \square 1_3$. The second matrix of the recursive algorithm is given by $Q_2 = diagM(Q_1 \square P) =$

$$= diagM\left(\begin{bmatrix} p_{(1,1)} & 0 & 0 \\ 0 & p_{(1,2)} & 0 \\ 0 & 0 & p_{(1,3)} \end{bmatrix} \square \begin{bmatrix} p_{(1,1)} & p_{(1,2)} & p_{(1,3)} \\ p_{(2,1)} & p_{(2,2)} & p_{(2,3)} \\ p_{(3,1)} & p_{(3,2)} & p_{(3,3)} \end{bmatrix}\right),$$

and applying the diagonalization process we get:

$$Q_2 = \begin{bmatrix} p_{(1,1)}p_{(1,1)} & 0 & 0 \\ p_{(1,2)}p_{(2,1)} & p_{(1,1)}p_{(1,2)} & 0 \\ p_{(1,3)}p_{(3,1)} & p_{(1,2)}p_{(2,2)} & p_{(1,1)}p_{(1,3)} \\ 0 & p_{(1,3)}p_{(3,2)} & p_{(1,2)}p_{(2,3)} \\ 0 & 0 & p_{(1,3)}p_{(3,3)} \end{bmatrix}.$$

Thus, the probabilities after two periods are given by the vector:

$$Q_2 \square 1_3 = \begin{bmatrix} p_{(1,1)}p_{(1,1)} \\ p_{(1,2)}p_{(2,1)} + p_{(1,1)}p_{(1,2)} \\ p_{(1,3)}p_{(3,1)} + p_{(1,2)}p_{(2,2)} + p_{(1,1)}p_{(1,3)} \\ p_{(1,3)}p_{(3,2)} + p_{(1,2)}p_{(2,3)} \\ p_{(1,3)}p_{(3,3)} \end{bmatrix}.$$

The following Figure 2 summarizes these first two steps of the algorithm of this simple homogeneous trinomial tree.

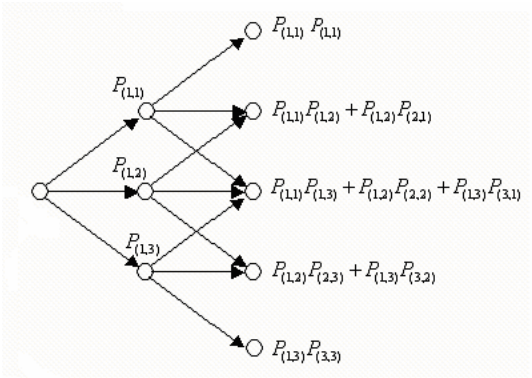

Fig. 2.. Final probabilities of the nodes at the 2nd step.

In the Appendix one can find the algorithm written in MATLAB language while in the next subsection there is the analysis of the computational complexity for such procedure.

3.1 Computational complexity

In order to calculate the computational complexity of the algorithm we count the number of primitive operations which has to be performed for a tree with N states and k steps long. First of all we observe that the maximum space memory used in one single step is of $O(N^2k)$ order. Therefore, we derive the complexity of the computation procedure distinguishing three kinds of operations

1. the number of the multiplications;
2. the number of the additions;
3. the number of operations due to the matrix manipulation performed by the *diagM* operator.

Table 1 reports the order of total complexity. In particular, it shows the number of operations the algorithm needs to perform the computation for each step. In the first part of the table is reported the number of operations obtained by the diagM operator (using the algorithm given in the Appendix). Observe that the "Memorize dimensions" and "Recompute dimensions" are assignment operations and they are valued one. A matrix is allocated at the physical level in the memory as a vector containing the sequence of its columns. Then we compute the complexity of the operation "Add N rows of zeros" as equal to a complete rewriting of the old matrix and the new portion of zeros. The "Vectorial reshape" operation values 1 because it changes only the addressing rule to the physical structure. The operation "deletion last N elements" values 1 since it decreases only the dimensions of the data structure. Even the last "Matrix Reshape" has value 1 since it changes another time only the addressing rule without affecting the data.

Therefore, we obtain a computational complexity of the $O(N^3k^2)$ order. In order to compare this complexity with the distribution valuation of a nonrecombining markovian tree, we use MATLAB algorithms and a notebook with intel centrino processor 1500 MHz, 512 MB of RAM. We observe that after one hour we couldn't finish the computation of a 20 days return distribution with a nonrecombining markovian tree based on a homogeneous Markov chain with 50 states. Instead with our algorithm we can compute in few seconds the 20, 40, 60, 90, 120 days return distributions of markovian trees based on Markov chains with 50, 60, 70, 80 states (see Table 2).

Table 2: This table summarizes the number of seconds necessary to compute 20, 40, 60, 90, 120 days return distributions with markovian trees based on Markov chains with 50, 60, 70, 80 states.

	States			
Days	***50***	***60***	***70***	***80***
20	0,13	0,22	0,37	0,51
40	0,58	1,00	1,54	2,19
60	1,40	2,35	3,57	5,00
90	3,32	5,47	8,33	11,59
120	6,81	10,41	15,42	22,49

Thus the complexity of our algorithm is much lower than that presented by any nonrecombining markovian tree.

4. Model backtesting

In this section we compare the empirical distribution of the gross returns and the *ex-post* distributions forecasted with:

a) the nonparametric markovian approach;
b) the log normal assumption typical of the Black and Scholes option pricing model.

Our data consist of gross returns on the S&P500, Dow Jones Industrials and Nasdaq from January 1996 to January, 2006. From a preliminary analysis of our data we observe that daily log-returns are negatively skewed and present kurtosis significantly different from the Gaussian one. Therefore, a first naïve analysis of data suggests that there exist better approximations of gross return distributions.

Table 3: This table summarizes Kolmogorov-Smirnov test (K-S) and Anderson-Darling test (A-D) for S&P500, Nasdaq, and Dow Jones Industrials (20, 40, 60 days) return series whose distributions are forecasted either with the markovian approach (Markov) or assuming log normal distributed gross returns (B&S).

20 days returns		**B&S**	**Markov**
S&P500	**K-S**	0,060107	0,060105
	A-D	0,167752	0,157602
Nasdaq	**K-S**	0,05172	0,050409
	A-D	0,523795	0,169516
DowJones Industrials	**K-S**	0,054981	0,060929
	A-D	0,18401	0,14947
40 days returns		**B&S**	**Markov**
S&P500	**K-S**	0,075152	0,075026
	A-D	0,19475	0,18873
Nasdaq	**K-S**	0,052786	0,056384
	A-D	0,238064	0,176513
DowJones Industrials	**K-S**	0,069358	0,068377
	A-D	0,176916	0,167292
60 days returns		**B&S**	**Markov**
S&P500	**K-S**	0,095129	0,093578
	A-D	0,229406	0,222491
Nasdaq	**K-S**	0,060138	0,063968
	A-D	0,225217	0,21579
DowJones Industrials	**K-S**	0,076256	0,076172
	A-D	0,215183	0,205883

In order to verify which distributional hypothesis approximates better the return series, we propose two tests which are based on the whole empirical distribution. We consider the Kolmogorov-Smirnov (K-S) test

$$K-S=\sup\left|F_E(x)-F(x)\right|$$

and the Anderson-Darling (A-D) statistic test

$$A-D=\sup\frac{\left|F_E(x)-F(x)\right|}{\sqrt{F(x)(1-F(x))}}$$

where $F_E(x)$ is the empirical cumulative distribution and $F(x)$ is the other distribution. The A-D statistic test weights discrepancies appropriately across the whole support of the distribution and it is particularly important if one is interested in determining tails of return distributions. Therefore, we could compare the empirical

cumulative distribution $F_E(x)$ either with the markovian hypothesis or with log-normal distributed gross returns.
For the Markovian approach we assume that daily gross returns follow an homogeneous Markov chain with 60 states. In particular we test the different distributional assumptions considering the forecasted 20, 40, 60 days gross returns.
Table 3 reports the results of the two tests. From almost all the tests we obtain a better fit with the distributions approximated with the markovian trees. In particular Anderson-Darling test suggests that we have much better forecasts on the distribution tails. Therefore, the proposed tests agree upon rejecting the hypothesis of log-normality.

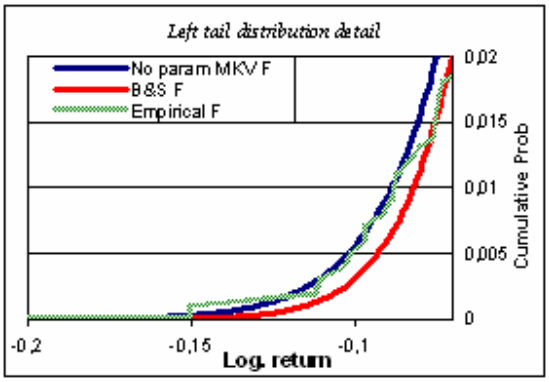

Fig. 3. *Ex-post* left tail of S&P500 20 days log-return distribution.

From a graphical point of view this results can be appreciated in Figure 3 where we show the *ex-post* comparison among the empirical, the markovian and Gaussian left tail of S&P500 20 days log-return distribution.

5. Concluding remarks

The paper proposes an algorithm to approximate long time return distributions with nonparametric markovian trees. The algorithm presents a polynomial complexity and the proposed approximation is generally better than the classical one used in financial applications. Moreover we show several further financial applications of the above algorithm that we report here in the following:

a) **in portfolio theory**: Suppose an investor with temporal horizon T and utility function u wants to invest his/her initial wealth W_0 in n risky assets. Using historical data on daily gross returns we can apply the previous algorithm assuming that each portfolio of gross returns $x'z$ follows a homogeneous Markov chain with N states, where $z=[z_1,...,z_n]'$ is the vector of gross returns and $x=[x_1,...,x_n]'$ is the vector of the positions taken in the n risky assets. Therefore, the investor should maximize his/her expected utility considering the portfolio gross return distribution at time T, i.e., he/she should choose the portfolio composition x solution of the optimization problem:

$$\max_x E(u(x'z))$$
$$x_i \geq 0;\ \sum_{i=1}^{n} x_i = W_0,$$

where the expected value is computed with the forecasted distribution at time T. Since the transition matrix P depends on portfolio composition x and it is valued on the portfolio historical series, it is not easy to solve the above optimization problem. However, with our algorithm we simplify the computational complexity of the problem and we propose a distribution strictly linked to historical series of gross returns.

b) **in risk management theory**: In order to forecast the risk position of a given portfolio, the managers generally compute the Value at Risk (VaR) of their investment positions. The VaR is the θ percentile of the future profit/loss distribution. Therefore, assuming that daily gross returns follow a Markov chain, we can easily estimate the percentile of future wealth considering the previous algorithm to compute the wealth distribution.

c) **in option pricing theory**: The value of any European contingent claim is simply given by the risk neutral expected value of a discounted contingent claim priced at the maturity. Therefore, given an evolution of the underlined with a markovian tree, we can compute the risk neutral valuation of contingent claim distribution at the maturity (see [6], [9]) and then value the option price with the discounted expected value.

We refer to [5], [6], [8], [10] for further possible financial applications.

Appendix:The algorithm in MATLAB language

The following algorithm describes in MATLAB language the computation of the final probabilities of the nodes at the *T-th* step using the recursive formulas (1), (2) when we assume a homogeneous Markov chain with transition matrix P.

```
function
[Final_Prob]=NonparamMKV_Density(State0,P,T);
```

```
[N,N]=size(P);Q=diag(P(State0,:));
if T>=2
  for level=2:T
    Q=DiagM(Q*P);
  end
end
Final_Prob= (Q*ones(N,1));
end
```

The function DiagM shifts the columns of the matrix given in input by an increasing number of elements starting by the second column. It is realized using matrix manipulation operators offered by the MATLAB language that are faster in comparison to manipulations performed using loops such as *"For..."*. Very useful is the reshape operator which changes the dimensions of a matrix by refilling a new matrix of different dimensions with the same elements of the previous.

```
function [Mshifted]=DiagM(M);
  % Memorize initial dimensions
  [Mrows,N]=size(M);
  % Add N rows of zeros;
  M=[M;zeros(N,N)];
  % Recompute new dimensions
  Mrows=Mrows+N;
  % Transform the matrix in a column vector
  M=reshape(M,Mrows*N,1);
  % Delete the last N elements of the vector
  M=M(1:Mrows*N-N);
  % Reshape the matrix shifted
  Mshifted=reshape(M,Mrows-1,N);
end;
```

The following few lines of code compute the final distribution. The "cumsum operator" applied to a vector of returns gives us another vector with the cumulative sum of the elements of the original vector:

```
function
[Final_Distrib]=NonparamMKV_Density(State0,P,T);
      Final_Prob=NonparamMKV_Density(State0,P,T);
      Final_Distrib=cumsum(Final_Prob);
end
```

Acknowledgments

The authors thank for grants COFIN 60% 2005, 2006 and, for helpful comments an anonymous referee, seminar audiences at AMASES 2005 (Palermo, Italy) and at IDEAL 2006 (Burgos, Spain).

References

[1] R. Bhar, and S. Hamori, Hidden Markov models: Applications to financial economics. Kluwer Academic Publishers, Dordrecht, 2004.
[2] P. Christoffersen, "Evaluating interval forecasts", International Economic Review 39 (4), 841—862, 1998.
[3] G. D'Amico, J. Janssen, and R. Manca, "Non-homogeneous backward semi-Markov reliability approach to downward migration credit risk problem", Proc. 8th Italian Spanish Meeting on Financial Mathematics, Verbania, June 2005.
[4] R. Elliott, and J. Van der Hoek, "An application of hidden Markov models to asset allocation problems" Finance and Stochastics 1, 229-238, 1997.
[5] A. Leccadito, S. Ortobelli L., E. Russo and G. Iaquinta, "Financial Risk Modeling with Markov Chains" Lecture Notes in Computer Science 4284, 1275-1282, 2006.
[6] G. Iaquinta, and S. Ortobelli L. " Option pricing with non-parametric markovian trees", Technical Report, Department MSIA University of Bergamo, 2006.
[7] N. Limnios, and G. Oprisan, Semi-Markov processes and reliability modeling. World Scientific, Singapore, 2001.
[8] S. Rachev, and S. Mittnik, Stable Paretian Models in Finance. John Wiley & Sons, New York, 2000.
[9] M. J. Stutzer, "A Simple Nonparametric Approach to Derivative Security Valuation," Journal of Finance 51, December 1996.
[10] W. Ziemba, and J. Mulvey, World Wide Asset and Liability Modeling. Cambridge University Press, 1999.

Gaetano Iaquinta currently is Ph.D. student at University of Bergamo (Italy) in "Computational methods for financial and economic forecasting and decisions" and Visiting Ph.D. student in "Finance" at University of Lugano (Switzerland). His research interests include computational finance, option pricing theory and risk management.

Sergio Ortobelli Lozza Professor at University of Bergamo (Italy). Currently he works in "Lorenzo Mascheroni" Department of Mathematics, Statistics, Computer Science and Applications. His research interests include mathematical finance, stochastic processes and computational finance.