Affine term structure models: forecasting the Colombian yield curve

Research practice II: Final report

Mateo Velásquez-Giraldo
Mathematical Engineering
EAFIT University

Diego A. Restrepo-Tobón
Tutor, Department of Finance
EAFIT University

I. Introduction

The term structure of interest rates (TS for short) shows the relationship between interest rates and investment time horizons at a given time. Approximating and forecasting the TS is useful for pricing financial instruments and managing risk.

The most common way of representing the TS is using yield curves. A yield curve is obtained by plotting the yields of different bonds with similar credit risk (e.g. issued by the same institution) against their maturities. An example of a yield curve constructed with the data we use in this project is presented in Fig 1.

Affine term structure models (ATSMs) approximate the yield curve as a function of N factors which can be observable (macroeconomic variables) or latent. We will denote the yield of a zero-coupon bond with maturity τ by γ_τ. ATSMs assume [1] that the short rate is an affine function[1] of a state vector $X(t)$ which contains the underlying factors:

$$r = \lim_{\tau \to 0} \gamma_\tau = \delta_0 + \delta_1^\top X(t) \tag{1}$$

with $\delta_0 \in \mathbb{R}$ and $\delta_1 \in \mathbb{R}^N$.

[1]A function $F : R^N \to R^M$ is said to be affine if $F(X) = A + B * X$ for some vector A and matrix B.

The state vector is assumed to follow an affine diffusion process under the risk-neutral measure Q:

$$dX(t) = \widetilde{K}\left(\widetilde{\Theta} - X(t)\right)dt + \Sigma\sqrt{S(t)}d\widetilde{W}t \tag{2}$$

where $\widetilde{K}$, $\Sigma \in \mathbb{R}^{N\times N}$, $\widetilde{\Theta} \in \mathbb{R}^N$, $\widetilde{W}$ is an N-dimentional independent brownian motion and $S(t)$ is a $N \times N$ diagonal matrix with entries:

$$[S(t)]_{i,i} = \alpha_i + \beta_i^\top X(t) \tag{3}$$

with $\alpha_i \in \mathbb{R}$ and $\beta_i \in \mathbb{R}^N$.

The market price of risk $\Lambda(X) \in \mathbb{R}^N$ must also be specified in order to obtain the physical measure (P) dynamics. We assume, as in [2], that $\Lambda(t) = \sqrt{S(t)}\lambda$, where λ is a vector of constants. Under these assumptions, the state process is also affine under the physical measure P:

$$dX(t) = K(\Theta - X(t))dt + \Sigma\sqrt{S(t)}dWt \tag{4}$$

Under this structure, [1] shows that the yield for any maturity τ can be obtained as an affine function of the state vector:

$$\gamma_\tau(t) = A(\tau) + B(\tau)^\top X(t) \tag{5}$$

where coefficients $B(\tau)$ and constants $A(\tau)$ are found by solving the following system of differential equations:

$$\begin{aligned} a'(\tau) &= -\delta_0 + b(\tau)^\top \widetilde{K}\widetilde{\Theta} + \frac{1}{2}\sum_{i=1}^{N}[b(\tau)^\top \Sigma]_i^2 \alpha_i \\ b'(\tau) &= \delta_1 - \widetilde{K}^\top b(\tau) + \frac{1}{2}\sum_{i=1}^{N}[b(\tau)^\top \Sigma]_i^2 \beta_i \end{aligned} \qquad (6)$$

with $a(0) = 0$, $b(0) = \vec{0}$, $A(\tau) = -a(\tau)/\tau$ and $B(\tau) = -b(\tau)/\tau$. These equations are consequences of the no-arbitrage hypothesis, which are derived in [3].

In this project we follow the methodology presented in [4] to estimate nine ATSMs using Colombian data.

II. Methodology

A. Data

As the Colombian market is still in its development phase, it is not possible to obtain observations of zero-coupon yields for fixed maturities over long time periods. We therefore use the Nelson-Siegel model to approximate yield curves. Curves' parameters are those published daily by *Infoval* in the time period between 1/08/2002 and 03/02/2015.

Taking out non-bursatile days, our sample amounts to 3051 days. Estimations are conducted using the first 2000 observations, reserving the rest for out of sample forecasting validations.

We use this data to calibrate the nine ATSMs presented in [4], which range from one to three factors. We adopt the notation proposed in [2], which denotes models as $A_M(N)$, where N is their number of factors and $M \leq N$ is the number of them which appear in the volatility term of Eq 2 and Eq 4.

To calibrate the models, we follow the methodology presented in [4]. We assume that the 3 year, 6 year and 9 year yields are observed with independent Gaussian errors. For a model with N factors we also assume that N yields are observed without errors:

- 1 year yield for one-factor models.
- 1 year and 10 year yields for two-factor models.
- 1 year, 5 year and 10 year yields for three-factor models.

B. The loglikelihood function

We use the yield loglikelihood function presented in [4]. This approach is summarized in Fig 2.

For a given set of parameters, a system of linear equations is obtained by using Eq 5 for every maturity observed without error. This system can be solved for every point of time, as it has N equations and N variables (the value of every state). A time series of the the value of each state is thus obtained.

Once the state is calculated, we obtain estimators for yields observed with errors using Eq 5. We calculate errors as the differences between estimated and observed values. The loglikelihood of errors can then be found using the Gaussian density function.

The state loglikelihood is obtained with the approximations proposed by Aït-Sahalia & Kimmel in [5]. Using the change of variable theorem, the implied state loglikelihood multiplied by a jacobian determinant to obtain the loglikelihood of yields observed without errors.

The total loglikelihood of the observed yields is the sum of the loglikelihood of the observation errors and the loglikelihood of yields observed without error. Evaluating the loglikelihood of a time series of yields thus involves: solving systems of differential equations, multiple calculations involving inverse matrices and various callings of sub-function.

The domain of feasible parameters is also heavily restricted for various models. We apply the restrictions found in [4]. An additional restriction is that states which affect volatility have to be non-negative.

C. Estimation procedure

The preceding characteristics of the loglikelihood function make it difficult to maximize using methods with theoretical bases. We turn to heuristics looking for approximate but attainable solutions. We use the "Differential Evolution" heuristic [6] because of its capacity to search for optimal parameter values in a continuous space.

Differential Evolution's pseudocode is presented in Algorithm 1 and an implementation in Matlab® R2013A is available in Appendix B. Within the method, solutions (sets of parameter values) are treated as vectors. We represent the j-th solution of a population P as $P(j)$, and its i-th parameter value as $P(j)_i$. 'Evolution' is recreated by comparing individuals (solutions) from an initial population with new ones and preserving the better ones.

New individuals are generated as a linear combination of individuals from the initial population. Before being compared with the initial individual, they can 'mutate' by changing some of their parameter values with a given probability. This 'evolutionary' process is repeated over numerous generation and the best (according to a given objective function) individual from the last population is taken as the final solution.

As every other heuristic method, there is no guarantee that the obtained solution will be a global maximum

```
Data: # Generations:$ng$, Population size:$np$,
      $F \in [0,2]$, $CR \in [0,1]$
$P1 \leftarrow$ Random initial population
for $i = 1$ to $ng$ do
    $P0 \leftarrow P1$
    for $j = 1$ to $np$ do
        $\{a, b, c\} \leftarrow$ random individuals from $P0$
        $V \leftarrow a + F * (b - c)$
        for $k = 1$ to $\#Params$ do
            if $rand \leq CR$ then
                $U_k \leftarrow V_k$
            else
                $U_k \leftarrow P0(j)_k$
            end
        end
        if $fobj(U) \leq P0(j)$ then
            $P1(j) \leftarrow U$
        end
    end
end
```

Algorithm 1: Differential evolution.

or minimum. However given the complexity of the likelihood function that is being maximized, we accept solutions as approximations that meet our requirements. Another problem that emerges from the use of heuristics (and numerical methods) is that the parameter standard errors can not be computed in a straight forward manner (as is the case in ordinary least squares, for example).

We estimate parameters of simulated affine diffusion processes in order to test method's pertinence to our problem and verify the accuracy of Aït-Sahalia & Kimmel's approximations. Simulations are carried out using Euler's numerical scheme.

III. Results

In this section we discuss the main products obtained from this research practice.

A. Implementation

The likelihood function was implemented in Matlab® R2013A. The system of ordinary differential equations is solved using its *ode45(...)* solver. However, the solver needs to be executed with a timeout because some parameter values can produce unbounded solutions, which make it very slow.

The Differential Evolution algorithm was also implemented. Matlab's *Parallel computing toolbox* was used in order to speed up the algorithm's execution. The source code is in Appendix B.

B. Simulation tests

We simulated diffusion processes with each of the models presented in Appendix A using Euler's numeric scheme and tried to estimate the data-generating parameters using Aït-Sahalia & Kimmel's loglikelihood functions. Estimation was carried out using Differential Evolution and Matlab's *fminserach(...)*. Results for one simulation with each model are reported in Table I.

From the results, it is evident that models present identification problems. Both methods often achieve loglikelihoods close to the values obtained with real parameters. However, estimated parameter values differ greatly from those used in the simulations. This issue is reported in [4].

Nevertheless, both methods almost always manage to obtain loglikelihood values which are even greater than the ones produced by real parameter values. This tests show that both optimization procedures are able to find solutions which are very close to a good value. This is an important characteristic to check when using heuristic methods.

The most important results from Table I are the enormous loglikelihood values obtained for the $A_1(1)$ and the $A_1(3)$ models using Differential Evolution. These values show that Aït-Sahalia & Kimmel's approximation sometimes return huge, wrong values. Repeated tests pointed to state variables which affect volatility as the cause of the problem. Models $A_0(1)$, $A_0(2)$ and $A_0(3)$ weren't affected by this issue in any of the tests.

C. Estimation with Colombian data

The parameters of each model were estimated using the data and maturities described in Section II-A. Matlab's *fminsearch(...)* performed poorly when fitting yields: its solutions remained very close to the initial solution and thus weren't very good. This behavior might emerge from the complexity of the feasible region, which makes it hard for search methods to succeed.

When the observed variable is the state vector $X(t)$ (as in our simulation tests), it doesn't have to be estimated, which makes the procedure simpler. When fitting yields, $X(t)$ is dependent on parameter values (as shown in Fig.2) and it changes with every solution. Many of the solutions obtained in the estimation must be discarded as they produce non-feasible values for $X(t)$, which raises the complexity of the feasible region.

Differential evolution performed better as it greatly improved feasible solutions (when found). The heuristic's parameters were set trying to cover a big part of the feasible region (numerous populations) and making small jumps when generating new individuals (small

		fminsearch(...)		**Differential Evolution**	
Model	**Loglikelihood with real params**	**Loglikelihood**	**Mean parameter relative error**	**Loglikelihood**	**Mean parameter relative error**
$A_0(1)$	4049	4049	1%	4049	1%
$A_1(1)$	$4967,6$	$4967,6$	60348%	6.1×10^{20}	109710%
$A_0(2)$	8023	$8023,5$	43%	$8023,5$	43%
$A_1(2)$	$6440,5$	$6449,9$	82%	$6449,9$	80%
$A_2(2)$	$5185,7$	$5185,3$	151%	$5187,3$	26%
$A_0(3)$	12181	12186	213%	12186	203%
$A_1(3)$	12533	12534	228%	$1,4 \times 10^{35}$	825%
$A_2(3)$	12885	12830	914%	12891	140%
$A_3(3)$	$6951,9$	$6954,6$	400%	$6960,2$	634%

Table I: Estimation with simulated data. The loglikelihood function evaluated with the simulation parameters is presented.

differential step F) in order to retain feasibility. The parameter values are presented in Table II.

Parameter	**Value**
ng	400
np	200
F	0.4
CR	0.9

Table II: Differential Evolution parameter values used for estimation.

The nine models can be categorized into three groups by the result of its estimation:

1) $A_0(1)$, $A_1(1)$, $A_0(2)$, $A_1(2)$ and $A_0(3)$ were adjusted succesfully.
2) Feasible solutions were found for $A_1(3)$ and $A_2(3)$. However, loglikelihoods take big values and the modeled yields don't adjust to observations.
3) The algorithm wasn't able to find solutions which met all restrictions for $A_2(2)$ and $A_3(3)$.

In the second case, the big loglikelihood values are produced by Aït-Sahalia & Kimmel's approximations. The state loglikelihood seems to be able to get arbitrarily big, which makes the error loglikelihood lose relevance. This results in fits as the one presented in Fig 3, where modeled yiels are very different from observed ones and the yield loglikelihood was 5.59×10^{38}.

As a test, we estimated an $A_1(3)$ model omitting solutions with state loglikelihoods greater that 10^{10} and results improved greatly. This makes us believe that big values are indeed an error.

The third case consists of two model in which all the states affect volatility. This makes all states need to be non-negative in order to be feasible. States obtained from our data seem to not be able to fulfill non-negativity and existence restrictions [4]. We made tests dropping existence restrictions and feasible solutions were quickly found.

Plots of the fit of every model in case 1 are presented in Figs 4-8. The achieved fit is overall very good. Mean errors for each maturity (omitting the ones observed without error) are presented in Table III.

	3 years	**6 years**	**9 years**
$A_0(1)$	$0,5334\%$	$0,0992\%$	$-0,6731\%$
$A_1(1)$	$0,0586\%$	$-0,0970\%$	$0,0521\%$
$A_0(2)$	$0,0634\%$	$0,0051\%$	$-0,0083\%$
$A_1(2)$	$-0,5367\%$	$-0,1075\%$	$0,0024\%$
$A_0(3)$	$0,3378\%$	$-0,0279\%$	$-0,0178\%$

Table III: Mean errors for each maturity and model.

As it was expected, fit gets better as the number of factors increase. However, the number of factors that affect volatility can make the fit better or worse.

One-factor models manage to capture the mean of longer maturities using the 1-year yield as observed without error. In two factor models, the inclusion of a longer maturity yield (10 years) as observed without error makes the models flexible enough to capture most of the deviations from the mean. Finally, the three-factor model is able to represent yields very closely.

To check the accuracy of the $A_0(3)$ model, we estimated it using yields from 1 to 10 years. The obtained fit is presented in Fig.9. The model can adjust all yields with great precision.

IV. Conclusions and further work

In this project we intended to make a first step towards the application of ATSMs in Colombia. We implemented an estimation methodology based on the one proposed by

Aït-Sahalia & Kimmel in [4], which is already working well for five models ranging from one to three factors.

Through simulation we managed to identify a problem that can emerge from using the loglikelihood expansions from [5] when estimating diffusion processes. We intend to contact the author to ask him about what might be causing it.

Two different methods were also tested for optimizing the yield loglikelihoods. Matlab's *fminsearch(...)* was found to be stagnant because of the complexity of the feasible space. An evolutionary algorithm, *Differential evolution*, obtained better results.

Out of the five models for which the estimation procedure is working, $A_0(2)$ and $A_0(3)$ obtained the best in-sample fit, deviating only a few basis points from observed yields. It was also shown that the $A_0(3)$ model is able to adjust a greater number of yields very closely.

In upcoming months we intend to test the forecast accuracy of the estimated models using data from 2012 to 2015. Forecasts and confidence intervals for different horizons will be obtained using Montecarlo simulation and Euler's numeric scheme.

We expect to be able to determine the cause of the problem with the state loglikelihood approximations. If we were to solve it, our estimation procedure would work for new models which we would include in further tests.

Another possible research matter could be an analysis of the confiability of our estimation. As heuristic methods and loglikelihood approximations are used, there is no straight forward way of determining confidence intervals for estimated parameters. Numerical or simulation-based methods could be explored.

References

[5] Ait-Sahalia, Y. (2008). Closed-form likelihood expansions for multivariate diffusions. *The Annals of Statistics*, 36(2):906–937.

[4] Ait-Sahalia, Y. and Kimmel, R. L. (2010). Estimating affine multifactor term structure models using closed-form likelihood expansions. *Journal of Financial Economics*, 98(1):113–144.

[2] Dai, Q. and Singleton, K. J. (2000). Specification analysis of affine term structure models. *The Journal of Finance*, 55(5):1943–1978.

[3] Duffie, D. and Kan, R. (1996). A yield-factor model of interest rates. *Mathematical Finance*, 6(4):379–406.

[1] Piazzesi, M. (2010). Affine term structure models. *Handbook of financial econometrics*, 1:691–766.

[6] Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

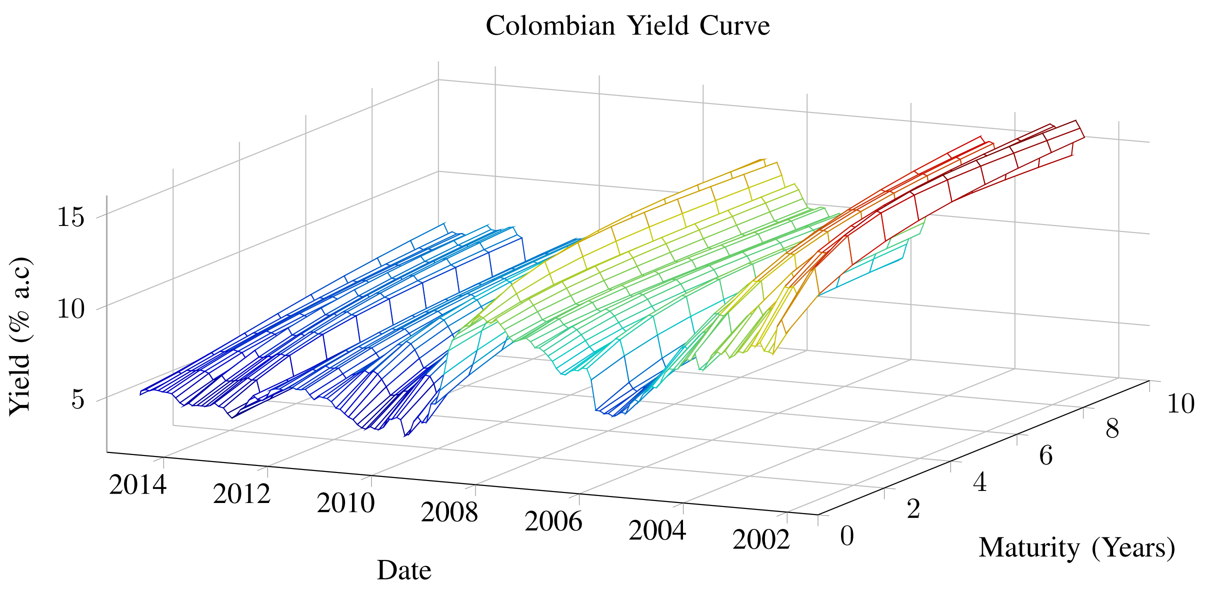

Figure 1: Data used throught the proyect.

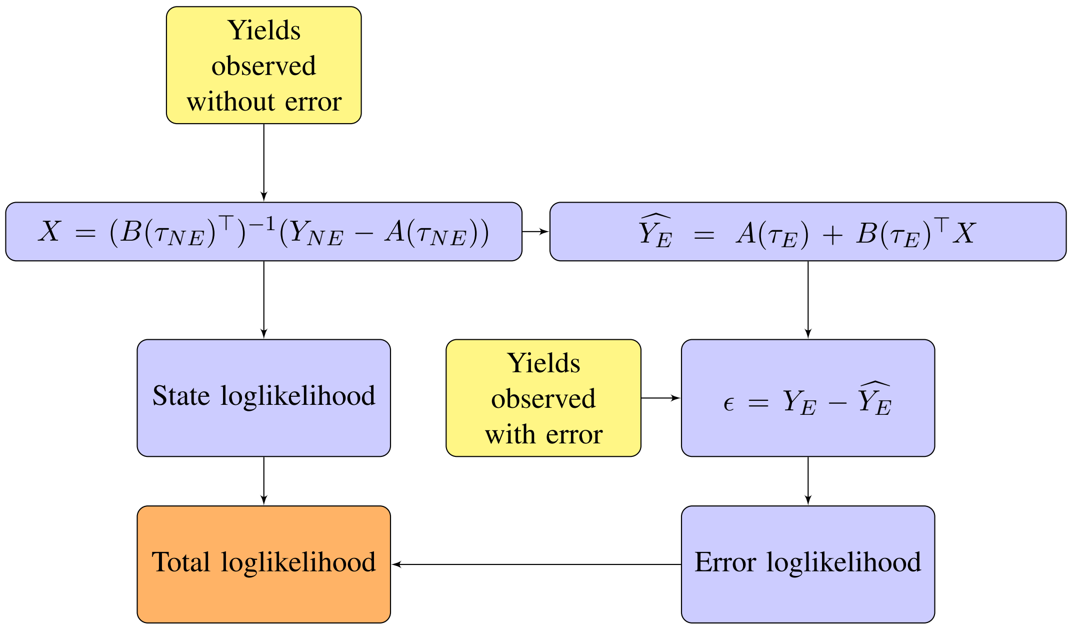
Yields observed without error
$X = (B(\tau_{NE})^\top)^{-1}(Y_{NE} - A(\tau_{NE}))$
$\widehat{Y_E} = A(\tau_E) + B(\tau_E)^\top X$
State loglikelihood
Yields observed with error
$\epsilon = Y_E - \widehat{Y_E}$
Total loglikelihood
Error loglikelihood

Figure 2: Yield loglikelihood.

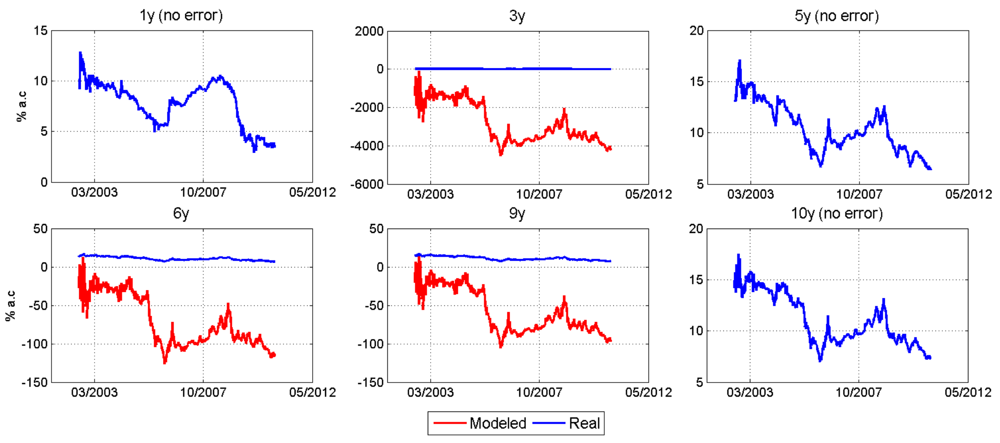

Figure 3: $A_1(3)$ results.

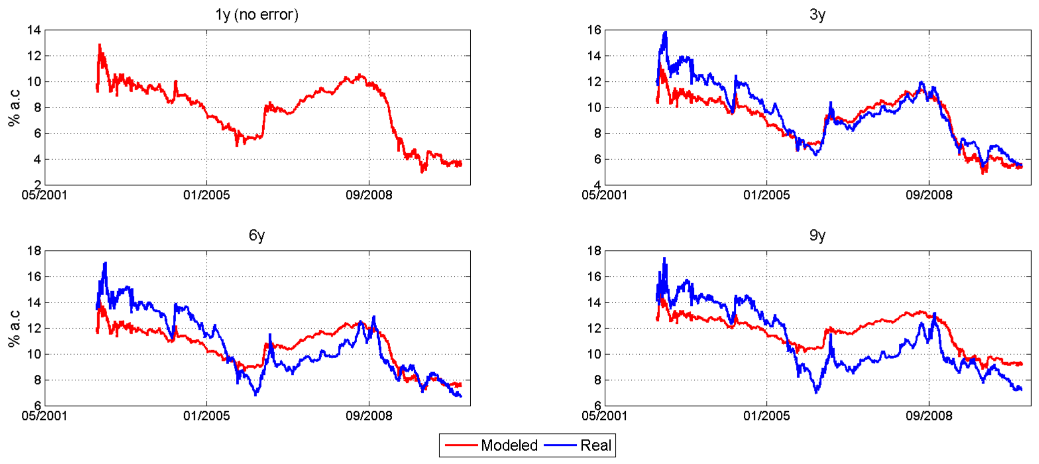

Figure 4: $A_0(1)$ results.

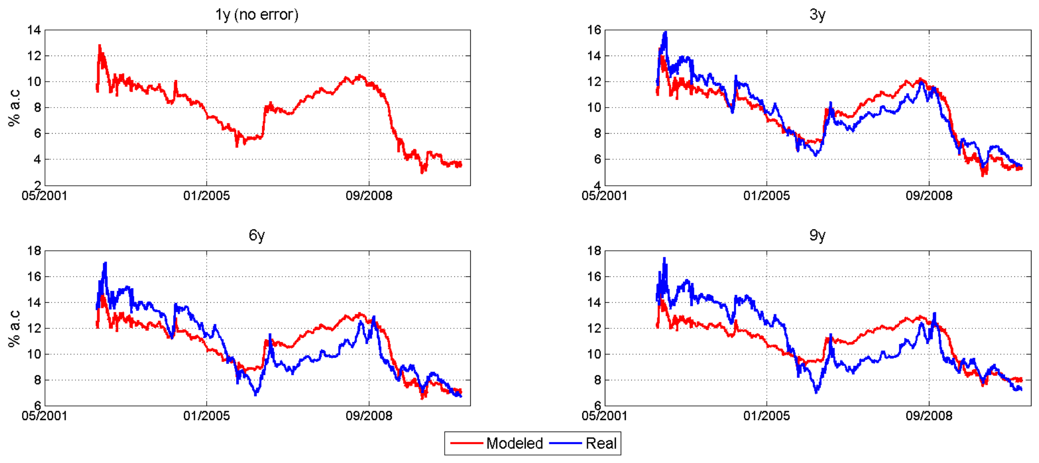

Figure 5: $A_1(1)$ results.

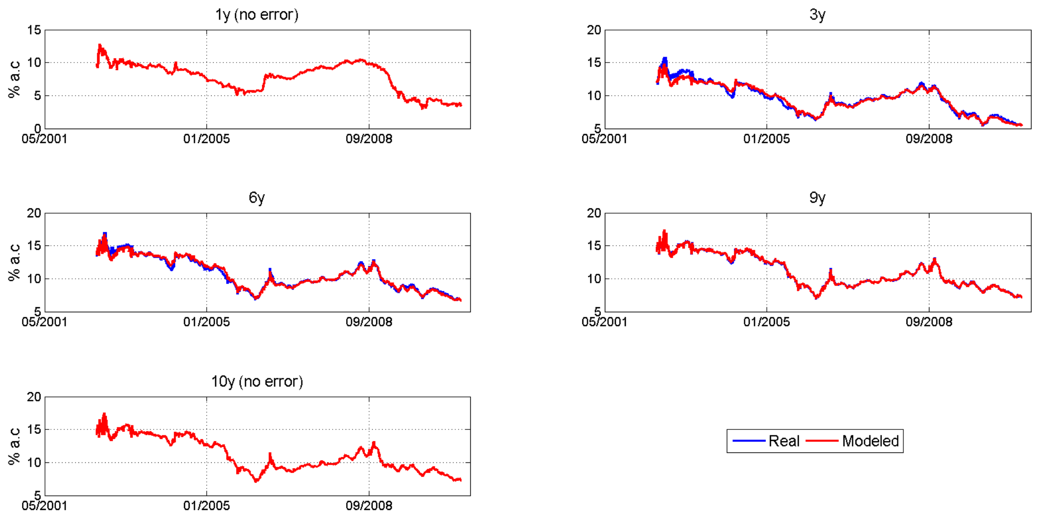

Figure 6: $A_0(2)$ results.

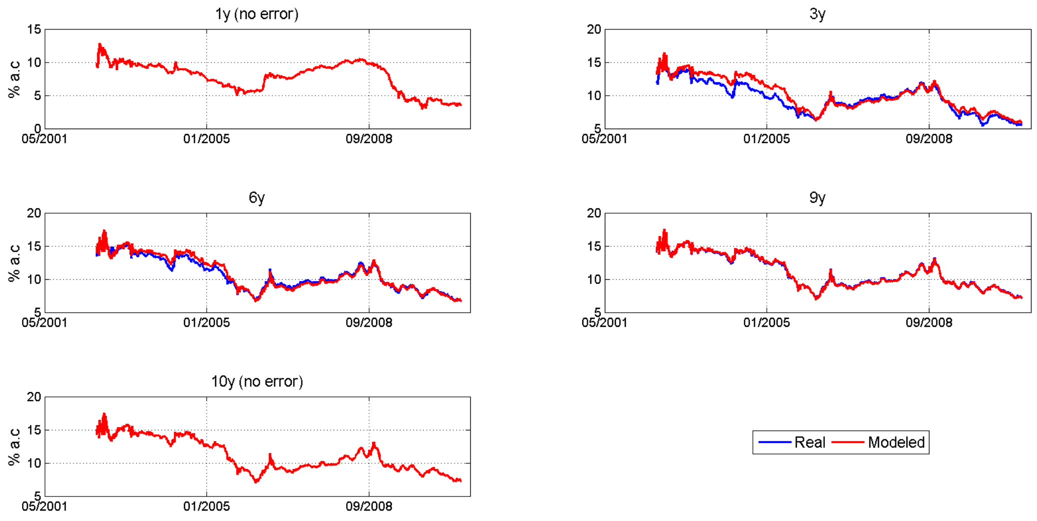

Figure 7: $A_1(2)$ results.

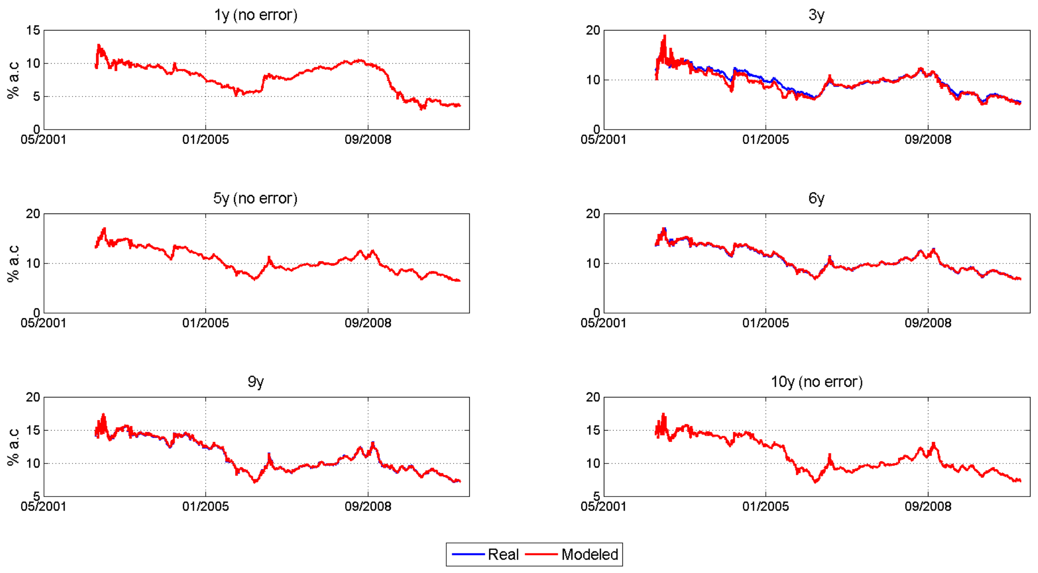

Figure 8: $A_0(3)$ results.

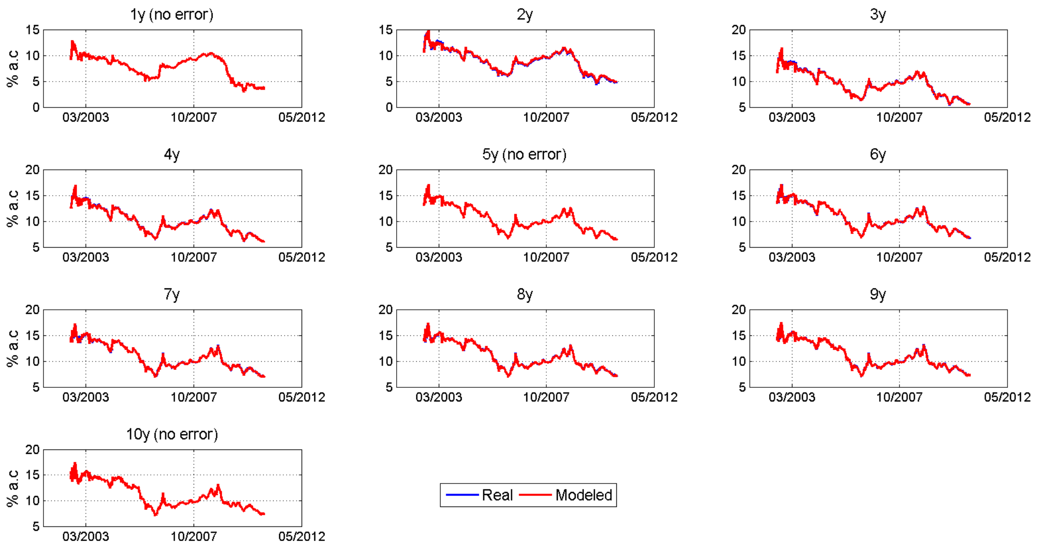

Figure 9: $A_0(3)$ results with 10 yields. Modeled and real yields are very close.

Appendix A
Model specifications

Here we present each one of the models mentioned in this study. The models' structures were taken from [4]. Each N-factor model consists of:

The short rate dynamics:

$$r = \delta_0 + \delta_1^\top X(t)$$

A diffusion process for the state under the physical measure P:

$$dX(t) = \mu_p(X(t)) + \sigma(X(t))dW^P$$

And a diffusion process for the state under the risk-neutral measure Q:

$$dX(t) = \mu_q(X(t)) + \sigma(X(t))dW^Q$$

Where δ_1, μ_p and $\mu_q \in \mathbb{R}^N$, $\sigma \in \mathbb{R}^{N\times N}$ and W^P and W^Q are N-dimensional standard brownian motions under their respective measures.

A. $A_0(1)$ model

$$\begin{aligned}\mu_p(X(t)) &= b_{11}X(t)\\ \mu_q(X(t)) &= -\lambda_1 + b_{11}X(t)\\ \sigma(X_t) &= 1\end{aligned}$$

B. $A_1(1)$ model

$$\begin{aligned}\mu_p(X(t)) &= a_1 + b_{11}X(t)\\ \mu_q(X(t)) &= a_1 + (b_{11} - \lambda_1)X(t)\\ \sigma(X(t)) &= \sqrt{X(t)}\end{aligned}$$

C. $A_0(2)$ model

$$\begin{aligned}\mu_p(X(t)) &= \begin{bmatrix} b_{11} & 0 \\ b_{21} & b_{22} \end{bmatrix} X(t)\\ \mu_q(X(t)) &= -\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} + \begin{bmatrix} b_{11} & 0 \\ b_{21} & b_{22} \end{bmatrix} X(t)\\ \sigma(X(t)) &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\end{aligned}$$

D. $A_1(2)$ model

$$\mu_p(X(t)) = \begin{bmatrix} a_1 \\ 0 \end{bmatrix} + \begin{bmatrix} b_{11} & 0 \\ b_{21} & b_{22} \end{bmatrix} X(t)$$

$$\mu_q(X(t)) = \begin{bmatrix} a_1 \\ -\lambda_2 \end{bmatrix} + \begin{bmatrix} b_{11} - \lambda_1 & 0 \\ b_{21} - \lambda_2 \beta_{21} & b_{22} \end{bmatrix}$$

$$\sigma(X(t)) = \begin{bmatrix} \sqrt{X_1(t)} & 0 \\ 0 & \sqrt{1 + \beta_{21} X_1(t)} \end{bmatrix}$$

E. $A_2(2)$ model

$$\mu_p(X(t)) = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} X(t)$$

$$\mu_q(X(t)) = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + \begin{bmatrix} b_{11} - \lambda_1 & b_{12} \\ b_{21} & b_{22} - \lambda_2 \end{bmatrix}$$

$$\sigma(X(t)) = \begin{bmatrix} \sqrt{X_1(t)} & 0 \\ 0 & \sqrt{X_2(t)} \end{bmatrix}$$

F. $A_0(3)$ model

$$\mu_p(X(t)) = \begin{bmatrix} b_{11} & 0 & 0 \\ b_{21} & b_{22} & 0 \\ b_{31} & b_{32} & b_{33} \end{bmatrix} X(t)$$

$$\mu_q(X(t)) = \begin{bmatrix} -\lambda_1 \\ -\lambda_2 \\ -\lambda_3 \end{bmatrix} + \begin{bmatrix} b_{11} & 0 & 0 \\ b_{21} & b_{22} & 0 \\ b_{31} & b_{32} & b_{33} \end{bmatrix} X(t)$$

$$\sigma(X(t)) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

G. $A_1(3)$ model

$$\mu_p(X(t)) = \begin{bmatrix} a_1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} b_{11} & 0 & 0 \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} X(t)$$

$$\mu_q(X(t)) = \begin{bmatrix} a_1 \\ -\lambda_2 \\ -\lambda_3 \end{bmatrix} + \begin{bmatrix} b_{11} - \lambda_1 & 0 & 0 \\ b_{21} - \lambda_2\beta_{21} & b_{22} & b_{23} \\ b_{31} - \lambda_3\beta_{31} & b_{32} & b_{33} \end{bmatrix} X(t)$$

$$\sigma(X(t)) = \begin{bmatrix} \sqrt{X_1(t)} & 0 & 0 \\ 0 & \sqrt{1 + \beta_{21}X_1(t)} & 0 \\ 0 & 0 & \sqrt{1 + \beta_{31}X_1(t)} \end{bmatrix}$$

H. $A_2(3)$ model

$$\mu_p(X(t)) = \begin{bmatrix} a_1 \\ a_2 \\ 0 \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & 0 \\ b_{21} & b_{22} & 0 \\ b_{31} & b_{32} & b_{33} \end{bmatrix} X(t)$$

$$\mu_q(X(t)) = \begin{bmatrix} a_1 \\ a_2 \\ -\lambda_3 \end{bmatrix} + \begin{bmatrix} b_{11} - \lambda_1 & 0 & 0 \\ b_{21} & b_{22} - \lambda_2 & 0 \\ b_{31} - \lambda_3\beta_{31} & b_{32} - \lambda_3\beta_{32} & b_{33} \end{bmatrix} X(t)$$

$$\sigma(X(t)) = \begin{bmatrix} \sqrt{X_1(t)} & 0 & 0 \\ 0 & \sqrt{X_2(t)} & 0 \\ 0 & 0 & \sqrt{1 + \beta_{31}X_1(t) + \beta_{32}X_2(t)} \end{bmatrix}$$

I. $A_3(3)$ model

$$\mu_p(X(t)) = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} X(t)$$

$$\mu_q(X(t)) = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} b_{11} - \lambda_1 & b_{12} & b_{13} \\ b_{21} & b_{22} - \lambda_2 & b_{23} \\ b_{31} & b_{32} & b_{33} - \lambda_3 \end{bmatrix} X(t)$$

$$\sigma(X(t)) = \begin{bmatrix} \sqrt{X_1(t)} & 0 & 0 \\ 0 & \sqrt{X_2(t)} & 0 \\ 0 & 0 & \sqrt{X_3(t)} \end{bmatrix}$$

APPENDIX B
DIFFERENTIAL EVOLUTION IMPLEMENTATION IN MATLAB ®

```
function output = DEestimationPar(fobj,lowPar,uppPar,np,ng,f,cr)
% Mateo Velásquez-Giraldo (mvelas26@eafit.edu.co)
% Universidad EAFIT, Medellín, Colombia.
% June 2015
% Tested with Matlab(R) R2013A.

    % Parameters:

    % -fobj: objective function which evaluates the "goodness" of a set of
    %        parameters. It will be MINIMIZED.
    % -lowpar: vector with the lower bounds for the parameters in the
    %          initial population.
    % -uppPar: vector with the upper bounds for the parameters in the
    %          initial population.
    % -np: number of solutions in every population.
    % -ng: number of generations to be simulated.
    % -f: differential weight.
    % -cr: crossover probability.

    %% Create initial population
    numpar = length(lowPar);
    p1 = zeros(numpar,np);
    for i = 1:numpar;
        p1(i,:) = unifrnd(lowPar(i),uppPar(i),np,1);
    end

    % Keep objective functions in a vector
    objP1 = zeros(1,np);
    for i = 1:np
        objP1(i) = fobj(p1(:,i));
    end

    %% Display:
    disp('Diferential evolution:');
    disp('****************************************');
    disp('  Generation    Best fobj   S.D Obj');
    disp('****************************************');

    bestP1 = min(objP1);
    info = [0,bestP1,std(objP1)];
    disp(num2str(info));

    %% Evolutive cycle

    for k = 1:ng;
        p0 = p1;
        objP0 = objP1;

        % Generate solutions
        parfor i = 1:np
            pU = zeros(numpar,1);
            % Random selection of three solutions
            samp = randsample([1:i-1,i+1:np],3);
            % Differential step
            pV = p0(:,samp(1))+f*(p0(:,samp(2))-p0(:,samp(3)));
            % Crossing
            R = randi(numpar);
            for j = 1:numpar;
                if rand < cr || j == R
                    pU(j) = pV(j);
                else
                    pU(j) = p0(j,i);
                end
            end
            % "Goodness" comparison with predecesor
            objSon = fobj(pU(:));
            if(objSon < objP0(i))
                % If the new individual is better, it replaces its
                % predecesor.
                p1(:,i) = pU(:);
                objP1(i) = objSon;
            end
        end
        % Display iteration results
        bestP1 = min(objP1);
        variation = std(real(objP1));
        info = [k,bestP1,variation];
        disp(num2str(info));
    end

    %% Find the best solution in the final population
    I = find(objP1==bestP1);
    best = p1(:,I(1));
    output = best;
end
```