CQF Final Project Report

Rustam Guseynov

January 17, 2013

Abstract

In this report I present results of my CQF Final Project. A short description of models used is provided with the reasons for selecting those particular models. I also discuss some issues arised during implementation of the project and present findings which I consider to be most significant for me.

1 Introduction

When I implemented this project, my primary goal was to understand the models better and answer some questions which were left unanswered (or even not formulated) during the course of study. Hence, I tried to avoid reproducing all formulas already provided and proven in textbooks or going into great details of well-known mathematical methods I used.

I have chosen Matlab as programming environment. The choice was determined by the unique features of Matlab: easy vector operations, lots of built-in linear algebra algorithms, convinient programming language. Matlab allows to concentrate on the task first, and then go to details as deep as it's necessary.

Matlab also has a very convinient feature of storing data in its native ".mat" files, which were like Excel spreadsheets for me. I have once downloaded, structured and imported source data, and then stored them into .mat file, which is reloaded each time script is run.

2 HJM Model

2.1 Comparing to short-rate models

HJM model is the first interest rate model which took advantage of modeling whole yield curve instead of modeling only one point of that curve like it was in short-rate models.

Why do I say "advantage"? Generally speaking, there's nothing wrong with short-rate models, but there's something unnatural in the way they treat bond pricing and yield curve modeling. Here's what I mean.
Let's take spot yield curve r(t). Instantaneous forward yield curve is then

$$f(t) = \frac{d}{dt}(rt) = r(t) + r'(t)t \tag{1}$$

and, conversely,

$$r(t) = \frac{1}{t}\int_0^t f(s)ds \tag{2}$$

Price of zero-coupon bond is

$$Z(0,T) = e^{-r(T)T} = exp\left(-T\frac{1}{T}\int_0^T f(s)ds\right) = exp\left(-\int_0^T f(s)ds\right) \tag{3}$$

These formulas are very intuitive and could be used directly should we have behaviour of spot or forward yield curve modeled. But in case of short-rate models one could not do it like this. Without any understanding of yield curve movements, one has to consider ZCB to be a derivative on short rate and looking for solution of bond pricing equation in form

$$Z(t,T) = e^{A(t,T)r+B(t,T)} \tag{4}$$

Comparing this equation to "true" bond pricing formula (**??**), I conclude that short-rate models approximate longer interest rate with linear function of r in form $A(t,T)r + B(t,T)$. One could call it a "first-order approximation". Later on one could estimate full yield curve $r(t,T)$ and forward curve $f(t,T)$ from these bond prices

$$r(t,T) = -\frac{lnZ(t,T)}{T-t}, \tag{5}$$

$$f(t,T) = -\frac{dlnZ(t,T)}{dT} \tag{6}$$

Thus short rate models are a "double trouble": a headache to implement and deduce bond prices, and only a first-order approximation of true dynamics of yield curve.

2.2 HJM setting

As I have already told before, HJM (in theory) models dynamic of whole yield curve. Or, more specifically, of an instantaneous forward yield curve.

[**?**, ch. 37] starts with premise that ZCB price follows some general lognormal random walk

$$dZ(t,T) = \mu(t,T)Z(t,T)dt + \sigma(t,T)Z(t,T)dW_t \tag{7}$$
$$\forall t\ Z(t,t) = 1$$

Integrating (**??**) into

$$lnZ(t,T) = \left(\mu(t,T) - \frac{1}{2}\sigma^2(t,T)\right)t + \sigma(t,T)W_t \tag{8}$$

and substituting into (**??**) he obtains

$$f(t,T) = \frac{\partial}{\partial T}\left(\frac{1}{2}\sigma^2(t,T) - \mu(t,T)\right)t - \frac{\partial}{\partial T}\sigma(t,T)W_t \tag{9}$$

It can be shown ([**?**, par. 10.3.2] for example) that in risk-free case $\mu(t,T) = r(t)$. Thus

$$f(t,T) = \sigma(t,T)\frac{\partial\sigma(t,T)}{\partial T}t - \frac{\partial}{\partial T}\sigma(t,T)W_t \tag{10}$$

Introducing $\nu(t,T) = -\frac{\partial}{\partial T}\sigma(t,T)$ and differentiating we obtain almost final version of forward rate SDE

$$df(t,T) = \nu(t,T)\left(\int_t^T \nu(t,s)ds\right)dt + \nu(t,T)dW_t$$

Finally, applying Musiela parametrization $\nu(t,T) = \bar{\nu}(t,T-t)$ and SDE becomes

$$d\bar{f}(t,\tau) = \left[\bar{\nu}(t,\tau)\int_0^\tau \bar{\nu}(t,s)ds + \frac{\partial}{\partial\tau}\bar{f}(t,\tau)\right]dt + \bar{\nu}(t,\tau)dW_t \tag{11}$$

where $\tau = T - t$.

2.3 Dimensionality and dimension reduction

The main input into equation (**??**) is $\bar{\nu}(t,\tau)$. In this model we do not simulate or forecast volatility, the volatility is constant model, and hence $\bar{\nu}(t,\tau) = \nu(\tau)$ which is known at time zero. That volatility function is an only source of randomness in SDE. Term $\bar{\nu}(t,\tau)dW_t$ means that on each step the whole yield curve will experience a shift proportional to value of $\bar{\nu}(t,\tau)$ at particular term τ. At the same time the model can easily be extended so that to introduce several random walks. It takes form

$$d\bar{f}(t,\tau) = \left[\sum_{k=1}^{N}\bar{\nu_k}(\tau)\int_0^\tau \bar{\nu_k}(s)ds + \frac{\partial}{\partial\tau}\bar{f}(t,\tau)\right]dt + \sum_{k=1}^{N}\bar{\nu_k}(\tau)dW_{t,k} \qquad \forall k = \overline{1:N} \tag{12}$$

where N is number of components.

We calibrate the model to the historical data, so we can easily estimate interest rates covariance matrix Ω. Let us suppose we have history on N intrest rates of different terms. In this case covariation matrix components are determined by $(\Omega)_{ij} = \rho_{ij}\sigma_i\sigma_j$ and $\Omega \in R^{N\times N}$, where $\rho_{ij} = corr(r_i, r_j)$ - correlation of i^{th} and j^{th} interest rates and σ_i is standard deviation of i^{th} rate.

In this case we have simulate N correlated random walks. In appendix **??** it is shown that there's an easy way to reduce number of random walks by extracting principal components.

2.4 Implementation

2.4.1 Data

I decided to use two sources of data. First is Bank of England yield curves, as suggested in project description and another is a set of MICEX [1] OFZ[2] curves. Both datasets are

[1] Major Russian exchange

[2] Russian domestic treasury bonds

stored in Matlab files: hjm_boe_forward.mat and hjm_micex_nss.mat. Bank of England yield curves are ready to use, and after loading hjm_boe_forward.mat file I obtain main variables: terms (1D array of terms of interest rates) and rates(2D array of interest rates history).

MICEX data are a bit more contrived. The data constists of a list of daily parameters of extended Nelson-Siegel-Svennson[3] approximation of validated and bootstrapped OFZ curves. I store this list in hjm_micex_nss.mat. After loading it I create my own grid of terms and evaluate rate curves at these points. Then I recalculate them into forward rates.

Choice between data sources depends on SELECTED_MODEL variable. It can take two values, BANK_ENGLAND_FWD and MICEX_NSS.

2.4.2 Discretization

In order to simulate equation (**??**), we have to replace it with its discrete version. First I will discretize it by t. Time t starts at zero and has step of arbitrary fixed lendth δt.

$$\bar{f}_i(\tau) - \bar{f}_{i-1}(\tau) = \left[\sum_{k=1}^{N} \bar{\nu_k}(\tau) \int_0^{\tau} \bar{\nu_k}(s)ds + \frac{\partial}{\partial \tau}\bar{f}_{i-1}(\tau)\right] \delta t + \sum_{k=1}^{N} \bar{\nu_k}(\tau)\delta W_k \qquad (13)$$

Here N is number of principal components, and i ranges from 1 to artbitrary chosen period of time.

Next step is discretization by terms τ. In my case it is either done exogenously (Bank of England data are already discretized by the Bank by terms) or I could do it myself. This means I have a system of M equations, there M is number of points on term grid.

$$\bar{f}_i^j - \bar{f}_{i-1}^j = \left[\sum_{k=1}^{N} \bar{\nu_k}(\tau_j) \int_0^{\tau_j} \bar{\nu_k}(s)ds + \frac{\bar{f}_{i-1}^j - \bar{f}_{i-1}^{j-1}}{\delta\tau_j}\right] \delta t + \sum_{k=1}^{N} \bar{\nu_k}(\tau_j)\delta W_k \qquad (14)$$

where $j = \overline{1:M}$ and $\delta\tau_j = \tau_j - \tau_{j-1}$ is a length of j^{th} term grid step.

The only undiscretized thing here is integral $\int_0^{\tau_j} \bar{\nu_k}(s)ds$. $\bar{\nu}(\tau)$ is a function with known values on grid points. Hence we can easily do a numerical integration using, for example, standard trapezoid rule:

$$\int_0^{\tau_j} \bar{\nu_k}(s)ds = \sum_{i=1}^{j-1} \frac{\bar{\nu_k}(\tau_i) + \bar{\nu_k}(\tau_{i+1})}{2}\delta\tau_i \qquad (15)$$

Another approach is to use approximated principal components. In this case integration is performed as described in appendix **??**

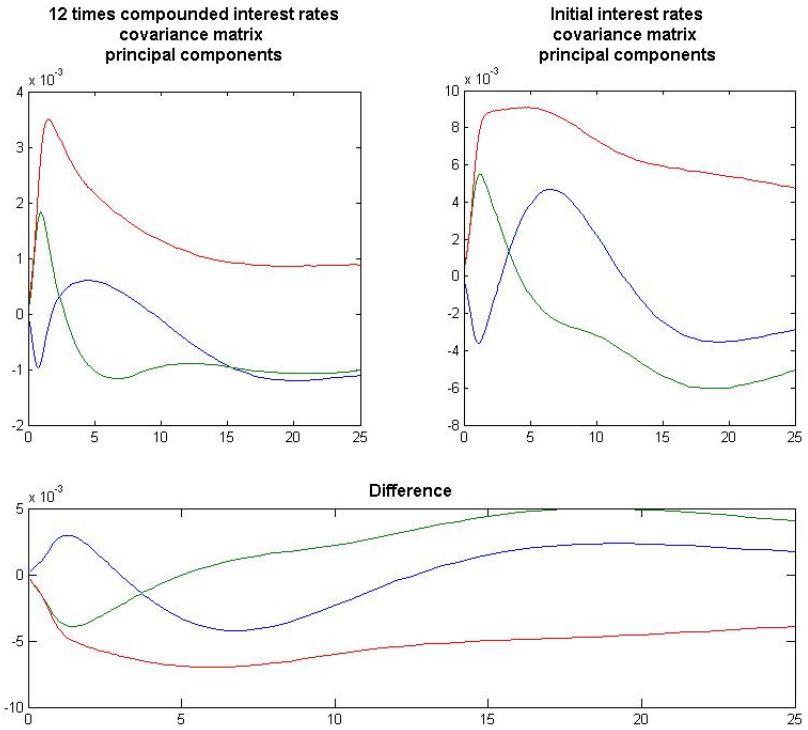
[3]See description at MICEX website

2.4.3 Volatility structure

The last issue I wanted to discuss is volatility structure. In [**?**, par. 37.15] it is told that standard HJM model might give negative interest rates. That was exactly what I have gotten, and it was a bit irritating.

To avoid it I implemented a non-infinitesimal short rate model as it is explained in there. In order to switch from standard to non-infinitesimal model it is necessary to set variable LOGNORMAL from 0 to 1.

One thing left unclear for me was if it is correct to use same principal components (e.g. same covariance matrix)? In non-infinitesimal model we work with m times compounded interest rates. In this case it might be correct to recalculate our interest rates into corresponding compounding and calculate covariance matrix for such rates. Recalculation is done using formula $r_m = m\left((1+r_1)^{1/m} - 1\right)$. There's significant difference in principal components:



In this case no negative interest rates occur.

A PCA and simulation of correlated random walks

Any matrix is simply a linear transformation in multidimensional vector space. It is well known that any matrix can be decomposed into set of rotations, shifts and scalings.

Let us suppose we have a covariance matrix $\Omega \in R^{N\times N}$ and vector of i.i.d. standard normal random variables $\eta \in R^N$. It is known that for any covariance matrix there exist self-adjoint matrices $U, \Lambda \in R^{N\times N}$ where U contains columns of eigenvectors and $\Lambda = diag(\lambda_1, ..., \lambda_N)$ - eigenvalues of matrix Ω. The eigenvalues are nonnegative due to nature of covariance matrix.

A.1 Generating correlated random walks using eigenvalue decomposition

I will start with vector $\eta = (\eta_1, ..., \eta_N)$ of i.i.d standard normal random variables. My goal is to obtain vector ξ of correlated standard normal random variables. Desired correlation structure is described by covariance matrix Ω.

Eigenvalue decomposition of covariance matrix is $\Omega = U\Lambda U^T = \left(U\sqrt{\Lambda}\right)\left(U\sqrt{\Lambda}\right)^T$. Let $\xi = U\sqrt{\Lambda}\eta$. In such case

$$cov(\xi_i, \xi_j) = E\left[(\xi_i - E\xi_i)(\xi_j - E\xi_j)\right] = E\left[\left(U\sqrt{\Lambda}\eta_i - EU\sqrt{\Lambda}\eta_i\right)\left(U\sqrt{\Lambda}\eta_j - EU\sqrt{\Lambda}\eta_j\right)\right]$$

Taking into account that $EU\sqrt{\Lambda}\eta_i = 0$ for any i we obtain

$$\begin{aligned} E\left[\left(\sum u_{ik}\sqrt{\lambda_k}\eta_k\right)\left(\sum u_{jk}\sqrt{\lambda_k}\eta_k\right)\right] = \\ E\sum u_{ik}\sqrt{\lambda_k}\eta_k u_{jm}\sqrt{\lambda_m}\eta_m = \\ \sum u_{ik}\sqrt{\lambda_k}u_{jm}\sqrt{\lambda_m}E(\eta_k\eta_m) = \\ \{\eta\text{'s are standard normal i.i.d}\} = \\ \sum u_{ik}\lambda_k u_{jk} = (\Omega)_{ij}\text{, q.e.d} \end{aligned} \quad (16)$$

A.2 Reducing number of dimensions

Now vector ξ can be written as $\xi = \sum U_k\sqrt{\lambda_k}\eta_k$, where U_k is k^{th} eigenvector. Evidently, ξ is a sum of vectors η weighted by their eigenvalues. This means we can select only K

largest eigenvalues for which value

$$R = \frac{\sum_{i=1}^{K} \sqrt{\lambda_i}}{\sum_{i=1}^{N} \sqrt{\lambda_i}} \tag{17}$$

is greater than some threshold (for example, 95%).

B Polynomial approximation

In order to generate smooth and nicely correlated trajectories of interest rates of adjacent terms, we have to smoothen the eigenvectors obtained by PCA as described in appendix **??**. These vector already look smooth, but polynomial approximation has an advantage that it allows easy interation and differentiation without using any numerical procedures.

B.1 Polynomial function representation

Suppose we have a polynomial $P_N(t) = \sum_{k=0}^{N} a_k t^k$ of power N. Then it can be stored as a vector of its $N+1$ components $V_{N+1} = [a_0, a_1, ..., a_N]$.

Integration of this polynomial on range from 0 to arbitrary value t would give $\int_0^t P_N(s)ds = P_{N+1}(t) = \sum_{k=0}^{N} \frac{a_k}{k+1} t^{k+1}$ which in vector representation is $V_{N+2} = \left[0, \frac{a_0}{1}, \frac{a_1}{2}, ..., \frac{a_N}{N+1}\right]$.

Similarly, differentiation would give $\frac{dP_N(t)}{dt} = P_{N-1}(t) = \sum_{k=1}^{N-1} k a_k t^k$ which in vector representation is $V_N = [a_1, 2a_2..., Na_N]$.

B.2 Solving for polynomial parameters

Now, how should one obtain polynomial approximation of arbitrary function given its values on some grid?

Suppose that we have an interval $x \in [A, B]$ which is divided into not necessarily even parts $A = x_0, x_1, ..., x_{N-1}, x_N = B$ and we have values $y_i = y(x_i)$ for $x = \overline{0:N}$. We want to find the closest approximation of that grid function y_i with polynomials of degree of N in mean-sqare sense.

This means we have to solve optimization problem

$$J(\theta) = \frac{1}{N+1} \sum_{k=0}^{N} (y_k - f(\theta, x_k))^2 \to max \tag{18}$$

where $f(\theta_N, x) = \sum_{k=0}^{N} \theta_k x^k$. Using vector notation. we can rewite it as follows:

$$\theta = argmax \left[J(\theta) = (\mathbf{y} - \theta \cdot \mathbf{x})^T (\mathbf{y} - \theta \cdot \mathbf{x}) \right] \tag{19}$$

Here $y = (y_1, ..., y_N)^T$, $\theta = (\theta_1, ...\theta_M)$ and

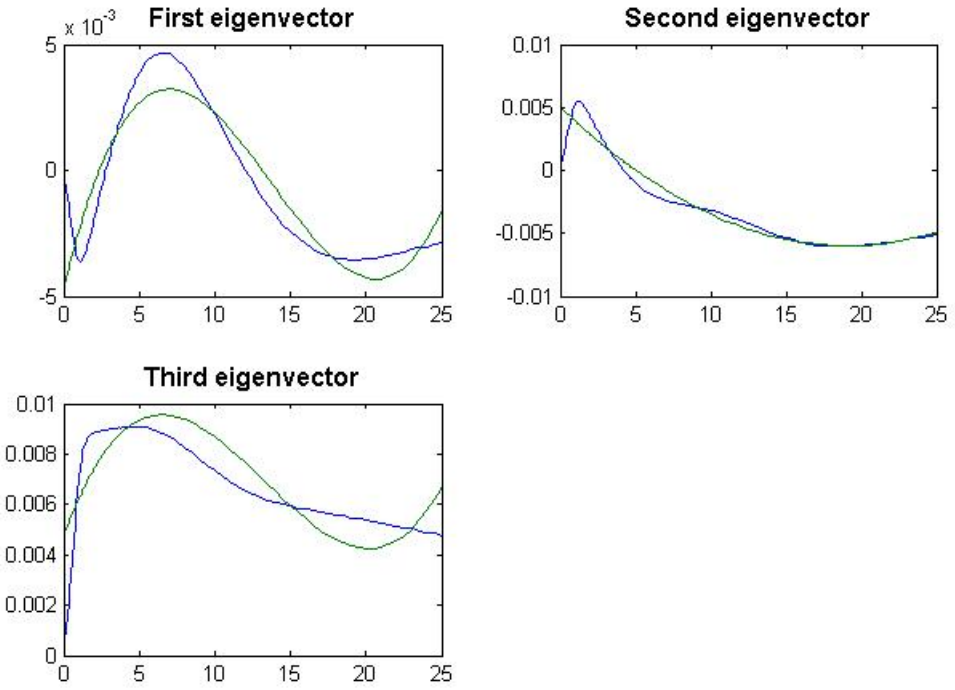
$$X = \begin{pmatrix} 1 & x_1 & \cdots & x_1^M \\ 1 & x_2 & \cdots & x_2^M \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_N & \cdots & x_N^M \end{pmatrix}$$

Using rules of matrix calculus we can solve this optimization problem by solving equation

$$\frac{dJ(\theta)}{d\theta} = 0 \iff \frac{d}{d\theta}(\mathbf{y} - \mathbf{X}\theta)^T(\mathbf{y} - \mathbf{X}\theta) = 0 \iff$$
$$\mathbf{X^T}(\mathbf{y} - \mathbf{X}\theta) = 0 \iff \theta = \left(\mathbf{X^T X}\right)^{-1}\mathbf{X^T y} \tag{20}$$

The latter is a so-called "normal equation". It is very convinient to use allowing to obtain solution in one step. This contrasts to iterative methods of solving optimization problem, where you would need to normalize independent variables, calculate gradient of cost function $J(\theta)$ and searching for optimal step size.

Example of approximation of eigenvectors with 3^{rd} order polynomials below:



References

[1] Paul Wilmott, *Paul Wilmott on Quantitative Finance*, 2nd Edition, 2006.

[2] Steven E. Shreve, *Stochastic Calculus for Finance*, 2nd Edition, 2008.

[3] Trevor Hastie, *The Elements of Statistical Learning*, 2nd Edition, 2010.