

# Hedging Dividend Futures

*by* Majd Agoumi

---

**Submission date:** 10-Sep-2019 12:33AM (UTC+0100)

**Submission ID:** 110619404

**File name:** AGOUMI\_MAJD\_00834447.pdf (50.06M)

**Word count:** 20669

**Character count:** 94216

Imperial College  
London

Janus Henderson  
INVESTORS

# HEDGING DIVIDEND FUTURES

by

Majd Agoumi (CID: 00834447)

Department of Mathematics  
Imperial College London  
London SW7 2AZ  
United Kingdom

Thesis submitted as part of the requirements for the award of the  
MSc in Mathematics and Finance, Imperial College London, 2018-2019

# Declaration

The work contained in this thesis is my own work unless otherwise stated.

Signature and date: **09/09/2019**

A handwritten signature in black ink, appearing to be 'M.A.' followed by a flourish.

## Acknowledgements

Despite this thesis being an individual piece of work, I honestly feel that my submission would not be complete without having a word for the people who provided me constant support, help and motivation throughout the whole process and without whom this whole research could not have been possible.

First, I want to thank Dr. Antoine Jacquier for his full support, encouragements as well as his insights, that helped me shape my thesis.

Also, I would like to have a special thought for Natasha Sibley, Mark Richardson and David Elms, for never letting me down as they always took the time to give me invaluable advices and share with me their deep expertise that helped me overcome the challenges I had to face during my research. Without a shadow of doubt, their commitment and constant help have played a key role in my work.

Finally, I would like to express a special thanks to each member of the team who welcomed me and always made sure I was at ease so I could reach my best potential during the last months and made my time at Janus Henderson absolutely fantastic.

# Contents

<b>1</b>	<b>Introduction</b>	<b>6</b>
<b>2</b>	<b>Stochastic Dividend Model</b>	<b>7</b>
2.1	General Framework . . . . .	7
2.2	Estimation of $\delta$ . . . . .	8
2.3	Resolution and Calibration . . . . .	11
<b>3</b>	<b>Dividend Futures</b>	<b>11</b>
3.1	Rolled Series . . . . .	13
3.2	Constant Maturities Futures . . . . .	13
3.3	Pricing . . . . .	15
<b>4</b>	<b>PCA Term Structure</b>	<b>16</b>
4.1	General theory on PCA . . . . .	16
4.2	Stochastic model with PCA . . . . .	18
4.2.1	Model . . . . .	18
4.2.2	Test of Mean-reversion . . . . .	19
4.2.3	Calibration . . . . .	20
4.2.4	Results . . . . .	26
4.3	Fractional Vasicek Model . . . . .	29
4.3.1	Hurst Exponent . . . . .	29
4.3.2	Fractional Brownian Motion and Model . . . . .	30
4.4	Hedging with PCA . . . . .	36
<b>5</b>	<b>Beta Hedging</b>	<b>40</b>
5.1	Historical model . . . . .	41
5.2	Parametric model . . . . .	42
5.2.1	Gompertz function . . . . .	44
5.2.2	Generalized Logistic Function . . . . .	44
5.3	Machine Learning Models . . . . .	46
5.3.1	Neural Network . . . . .	50
5.3.2	SVM . . . . .	52
5.3.3	XGB . . . . .	54
5.3.4	Stacking . . . . .	55
5.3.5	Smoothing with EWMA . . . . .	56
5.3.6	Results of Machine Learning models . . . . .	57

6	Back-Testing Beta Hedges	61
7	Regime Shift Detection	67
8	Conclusion	71

## 1 Introduction

In this paper, we will be studying Dividend Futures. Since the dividend dynamic can be slightly different in Europe, USA or Japan, we will focus on European Dividend Futures. More precisely, we will focus on the EuroStoxx 50 Index Dividend Futures (FEXD) [1]. These contracts pay the gross cash dividends paid by the companies (or the sum of the dividends if its on the index SX5E) during one year (reset date is set to be the third Friday of December of each year). At each time, there is 10 series listed. SX5E is the ticker used for the EuroStoxx 50, SX5GT is the ticker used for the EuroStoxx 50 Gross Returns.

Let's give an example. Currently, the front series (the series that is reaching expiry this year) is DEDZ9. It was first listed on the 21/12/2009 (it was then the 10th Tenor or series - we use the two terms in this paper, depending on the context). It will reach expiry on the 20/12/2019 at market close. It will then pay the sum of all dividends paid by the SX5E companies between 24/12/2018 (last reset date was 21/12/2018 at market close) and 20/12/2019. Similarly, DEDZ2 will pay the sum of dividends between the 20/12/2021 and the 16/12/2022. Each year one contract reaches expiry, and one contract is added.

One of the most notable stylised facts about these contracts is that their volatility is very low for short maturities, and similar to the underlying's volatility for long maturities. This is understandable since the uncertainty for the first Tenors is lower - in particular for the front series, the dividends for the ongoing year are actually very well predicted by corporate models. Bloomberg - among others - has a function BDVD that gives a forecast of the dividends over 3 years for each company. On the contrary, there is no reliable techniques that can give a precise forecast for the dividends that will be paid in 8, 9 or 10 years from now, so the value of these contracts is more volatile.

So why do the Dividend Futures make an interesting investment option? First, the interest is growing for these contracts, which makes them easier to trade. Then, if we have a closer look at the dividends term structure, we can see that Dividends are heavily discounted. For example, let's assume that the Dividends will not grow over the next 10 years - that is actually not a completely unreasonable assumption since over the past 10 years the Dividends were flat in Europe paying in average roughly 117 index points per year. As of 04/01/2019, the contract DEDZ8 was priced at 84.3 points, DEDZ7 at 87.0 points, and DEDZ1 at 110.3 points. That represents an annualized return of roughly 3.3% for the contracts DEDZ8 and DEDZ7 (respectively 10<sup>th</sup> and 9<sup>th</sup> Tenors), and 2% for DEDZ1 (3<sup>rd</sup> Tenor). And that is assuming that Dividends in Europe will stay flat for the next 10 years, which is unlikely (even if the past 10 years were flat, the 2008 crisis played an important part in this absence of growth). If we assume a growth of dividends of 1.7% per year - which is merely the average inflation in Europe for the past 22 years - we reach returns of roughly 5%, 5.1% and 3.7% per year for respectively the 10<sup>th</sup> Tenor, 9<sup>th</sup> Tenor, 3<sup>rd</sup> Tenor.

So, having an exposure to Dividend Futures might be interesting. However, and this is particularly true for the higher Tenors, we don't want to be exposed to market fluctuations. So, to gain exposure to Dividends without gaining exposure to the underlying market, we want to find a way to hedge the beta of the Dividend Futures. In part 2, we are going to introduce a stochastic model for dividends. In part 3, we are going to use this model to deduce a pricing formula for Dividend Futures. Then, in part 4, we are going to develop a Term Structure for Dividend Futures based on a PCA decomposition. We will also show that it is possible to hedge using PCA. In part 5, we will focus on Beta Hedging. To do so, we will estimate Betas using a statistical model, then approximating the Beta term structure using parametric model, and finally we are going to use Machine Learning to forecast the prices movements conditional to SX5GT movements and deduce Betas from that forecast. We back-test all these Beta Hedges in part 6. Finally, in part 7 we will use Unsupervised Machine Learning techniques to spot different regimes in the term structure.

## 2 Stochastic Dividend Model

In this part, we are going to develop a stochastic model for Dividends based on Büehler's theory [2]. The objective is to define a mechanic for Dividends moves, to later use it for our PCA framework. Büehler developed his framework to fit a very important property of Dividends: short term Dividends have a - very - low volatility, because it is possible to have a precise idea of the companies performance on a short term horizon, long term Dividends have a high volatility, close to the discounted stock's volatility, because they are very uncertain.

### 2.1 General Framework

First we introduce our notations. We denote by  $S_t$  the stock or index paying the dividends at time  $t$ ,  $X_t$  represents the discounted stock price at time  $t$  ( $X$  is a martingale),  $\Delta_t$  is the estimation, at time 0, of the dividends that will be paid at time  $t$ .  $\delta$  is the dividend profile (more on this later).

We can write:

$$dX_t = X_t \sigma_t dW_t \quad (2.1)$$

Following Büehler's model [2], we write dividends as the product between the expected dividend at time  $t$  (dividend profile) and a stochastic process denoted in the rest of this paper by  $Y$ . Then we have:

$$\Delta_t = \delta_t Y_t, \quad t \geq 0 \quad (2.2)$$

The process  $Y$  is mean-reverting and follows the Stochastic Differential Equation:

$$Y_t = \kappa(X_t^\alpha - Y_t)dt + \Sigma(t, Y_t, X_t)dB_t, \quad t \geq 0 \quad (2.3)$$



with  $\kappa$  the speed of mean-reversion, and  $X_t^\alpha = \alpha X_t + (1 - \alpha)$ , for  $\alpha$  a constant such that  $\alpha \in [0, 1]$ . Note that for  $\alpha = 1$  the process  $Y$  reverts around the martingale  $X^\alpha$ , and for  $\alpha = 0$  the two processes are related only through their correlation coefficient.  $\Sigma$  is a deterministic function that depends on  $t, Y_t$  (to ensure that  $\Sigma(t, Y_t, X_t) > 0$  for  $t \geq 0$ ) and  $X_t$  so we can eventually explicitly write a link between Dividends volatility and discounted stock/index volatility. For example, we can take  $\Sigma(t, Y_t, X_t) = \sqrt{Y_t} \tilde{\Sigma}$  where  $\tilde{\Sigma}$  is a constant, to find a CIR process. Finally, we can set  $[dW_t, dB_t] = \rho dt$ ,  $\rho \in [-1, 1]$ , to correlate the two Brownian motions. We have to remember, when choosing the volatility for these two processes, that conditional on a future state of the world, short term dividend futures have a low volatility and long term dividend futures have a high volatility (short term dividends are said cash-like, long term dividend are said proportional, i.e. correlated to stock price). This is a well verified empirical fact.

## 2.2 Estimation of $\delta$

One of the most important challenges of Buehler's model - along with finding a good volatility profile - is to estimate a dividend profile  $\delta_t$ , for  $t > 0$ . Indeed, that is the parameter that embed one's views on Dividends evolution in the next years (flat, growth...). In Europe we can currently assume a flat structure for Dividends paid.

After an observation of the last 10 years of paid dividends, we could conclude that each year the dividends paid have the same profile - and same magnitude! We can see that in average, SX5E will pay 117 Index Points (the currency is Euro) per year, as shown in Figure 1.

From now on, we are going to use only the last 10 years of Dividends, in order to be closer to the current reality. Once normalised (meaning we divide each year by the total dividend paid during the year, and we take the time to maturity between 0 and 1 as time unit), we can see in Figure 2 that each year the Dividends paid are following the same profile.

We have to keep in mind that the data represented in Figure 1 is the **cumulative** Dividends paid, when  $\delta_t$  in Equation (2.2) is actually the dividend paid at time  $t$ . What is actually represented in Figure 1 is the sum of  $\delta_s$  for  $s \in [0, t]$ , where 0 is the beginning of the current year and  $t$  the normalised current time.

So, to compute a dividend profile  $\delta$ , we are going to use the profile represented in Figure 2, and the fact that the sum of all dividends (in discrete time the integral of  $\delta$ ) should give 117 Index Points over one year. Using the black curve in Figure 2 and differentiating it, then normalising it so the integral between 0 and 1 is worth 1, we obtain the profile shown in Figure 3. Or, if we are interested in the value in euros paid each day over a year, we have to normalise to get the profile Figure 4.

We then use the function represented in Figure 4 as a proxy for  $\delta$ .

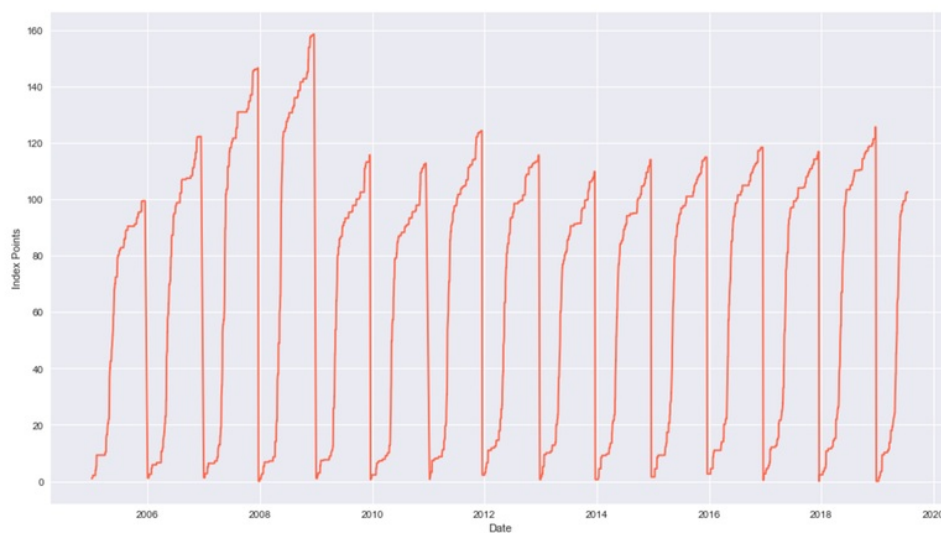


Figure 1: Cumulative Dividend paid each year between 03/01/2005 and 17/07/2019. We can see that before the 2008 crisis, Dividends paid grew each years. Since then, the Dividend growth is flat, around 117 euros per year.

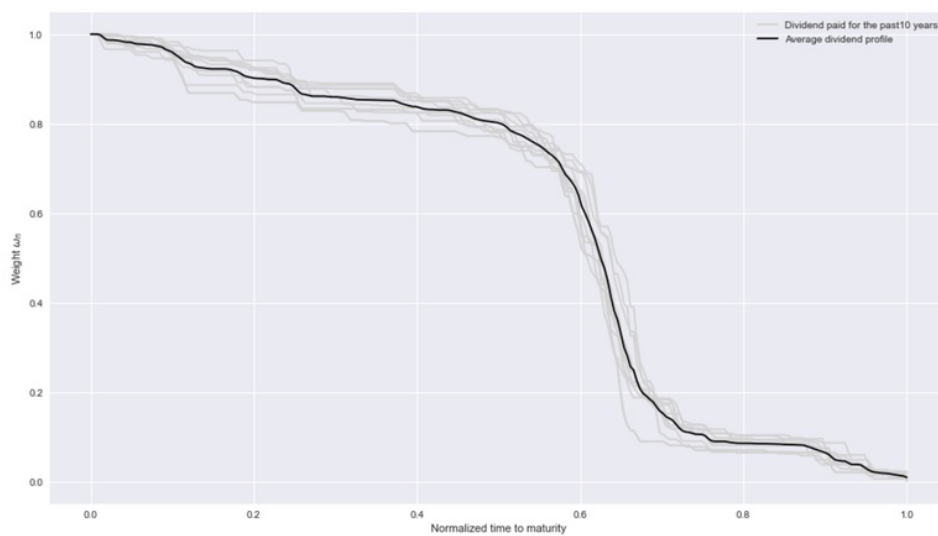


Figure 2: Cumulative Dividend paid over one year (value normalised, time to maturity between 0 - end of the year - and 1 - beginning of the year). Each grey line represent the actual dividends paid in one of the last 10 years, the black line represents the average profile.

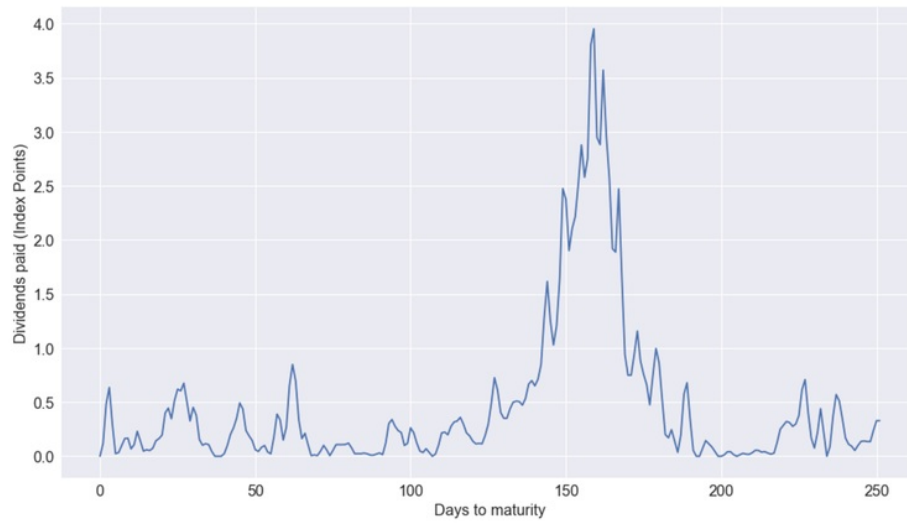


Figure 3: Dividend paid each day over one year.

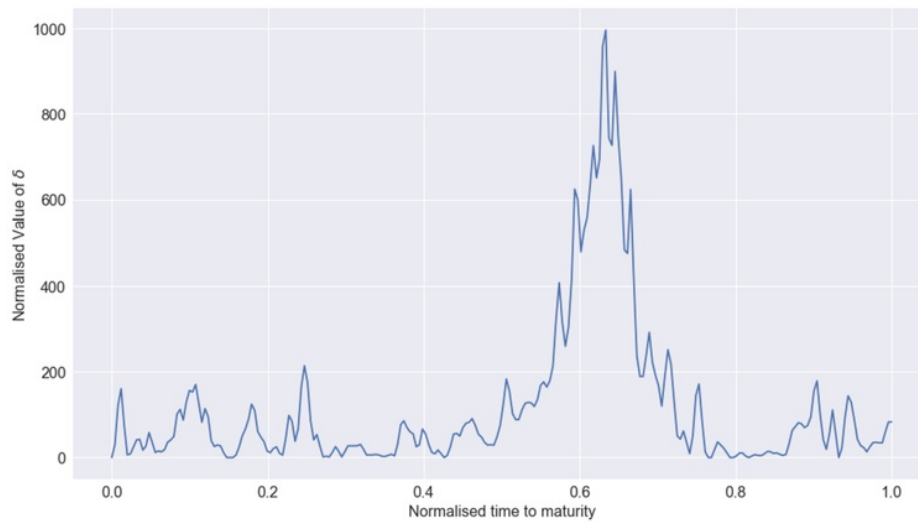


Figure 4: Normalised Dividend paid each day over one year. The area below this curve is roughly equal to 117.

## 2.3 Resolution and Calibration

We can easily find a solution for Equation (2.3) using the change of variable  $Z_t = e^{-\kappa t} Y_t$  and computing  $dZ_t$ . We find:

$$Y_s = e^{-\kappa(s-t)} Y_t + \kappa \int_t^s e^{-\kappa(s-u)} X_u^\alpha du + \int_t^s e^{-\kappa(s-u)} \Sigma(u, Y_u, X_u) dB_u, \quad s \geq t \geq 0. \quad (2.4)$$

Then, we have the following expression for the conditional expectation:

$$E_t[Y_s] = E[Y_s | (X_t, Y_t)] = X_t^\alpha + (Y_t - X_t^\alpha) e^{-\kappa(s-t)}, \quad s \geq t. \quad (2.5)$$

Finally, using Equation (2.2) and the fact that  $\delta_t$  is deterministic, we can write:

$$E_t[\Delta_s] = E_t[\delta_s Y_s] = \delta_s E_t[Y_s] = \delta_s (X_t^\alpha + (Y_t - X_t^\alpha) e^{-\kappa(s-t)}), \quad s \geq t. \quad (2.6)$$

For  $\kappa(s-t) \ll 1$ , we have  $E_t[\Delta_s] \approx \delta_s Y_t$ , which correspond to the cash-like mode, and for  $\kappa(s-t) \gg 1$  we have  $E_t[\Delta_s] \approx \delta_s X_t^\alpha$ , which correspond to the mode proportional.

To calibrate  $Y$ 's volatility, we then have to create a process with a low volatility for low values of  $t$ , high volatility for high values of  $t$ . Even if for high values of  $t$  the volatility is dominated by the process  $X^\alpha$ , we could observe that the 10<sup>th</sup> traded Dividend Future series (currently DEDZ8) has actually a higher volatility than the discounted underlying, so we use  $Y$ 's volatility to try to have a better match with the long-term Tenors.

To create  $Y$ 's volatility, we computed the ratio between  $X$ 's volatility and each one of the traded series, then used a linear interpolation to define  $\Sigma(t, Y_t, X_t)$ . We then write  $\Sigma$  (defined in Equation (2.3)) as:

$$\Sigma(t, Y_t, X_t) = \beta(t, X_t) \sigma_t \sqrt{Y_t} \quad (2.7)$$

with  $\sigma_t$  the discounted stock's volatility at time  $t$  and  $\beta$  the function represented in Figure 5.

Finally, we obtain for  $E_t[Y_s]$ ,  $t < s$ , the profile in Figure 6.

We can see in Figure 6 that for small value of  $s$ , the dividends are cash-like, i.e. have low volatility. For high values of  $s$ , dividends are proportional, i.e. their volatility is close to the underlying's volatility.

In some cases, we can find a closed form solution for Equation (2.3). For example, if  $\Sigma(t, Y_t) = \sqrt{Y_t} \tilde{\Sigma}$  for  $\tilde{\Sigma}$  a constant, we have that  $Y$  follows a non-central Chi-Square distribution.

## 3 Dividend Futures

Now that we have a stochastic model to model Dividends, we can introduce the Dividend Futures and some firsts results. We are going to introduce the difference between Rolled Series and Constant Maturity Futures (CMF). Then we will focus on pricing the futures.

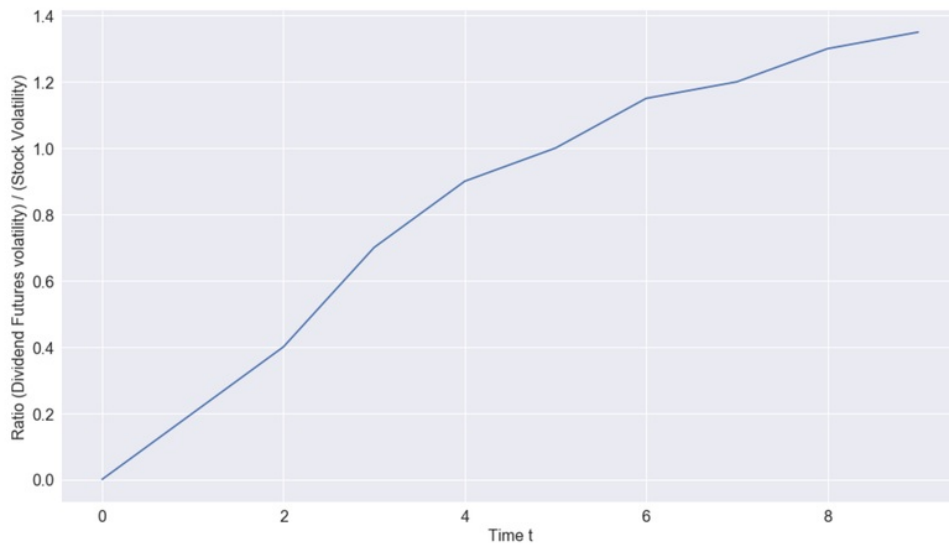


Figure 5: Shape of function  $\beta$  as defined in Equation (2.7) -  $t$  is in years.

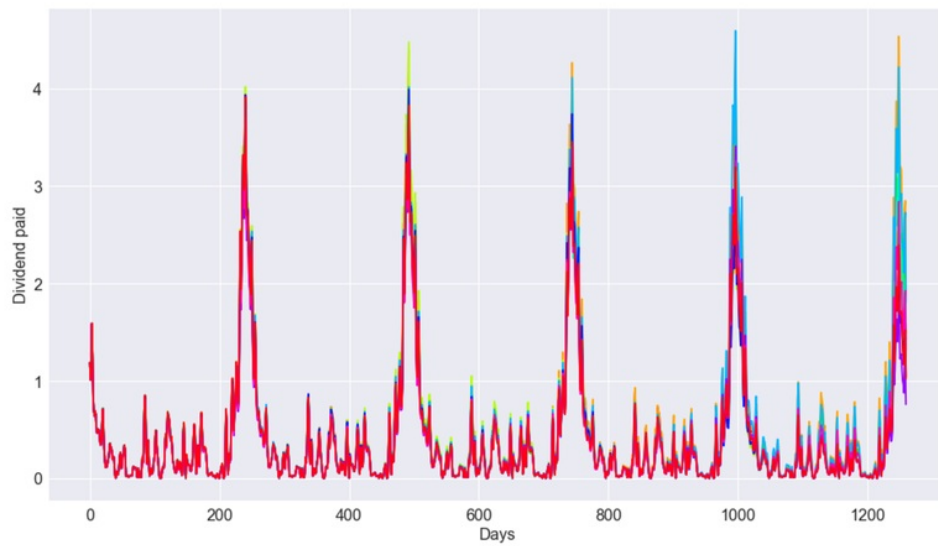


Figure 6: Shape of  $E_t[Y_s]$ ,  $t < s$ , for 10 different simulations - each colour represents one simulation. For small value of  $s$ , the dividend are cash-like, i.e. have low volatility. For high values of  $s$ , dividends are proportional, i.e. their volatility is close to the underlying's volatility.

### 3.1 Rolled Series

The Rolled series are the series such that the first series is always the shortest maturity quoted on the market, and the  $n^{\text{th}}$  series correspond to the  $n^{\text{th}}$  Tenor currently quoted on the market. For example, on the 17/05/2019, the first Rolled Series' price is DEDZ9's price, the tenth is DEDZ8's price. On the 17/05/2018, the first Rolled Series' price was DEDZ18's price, the tenth is DEDZ7's price. To plot these series, we rolled the contracts at each reset date (third Friday of December of each year at market close). This way, the front series always represents the contract that will reset on the next 3<sup>rd</sup> Friday of December.

For example, the Series from 04/01/2016 17/05/2019 are represented in Figure 7.



Figure 7: Dividend Futures - Rolled Series from 04/01/2016 to 17/05/2019.

These series will be particularly important for part 5 on Beta Hedging.

### 3.2 Constant Maturities Futures

The Constant Maturities Futures (CMF) are synthetic time series whose objective is to offer a way to approximate the following: at each time, these contracts have the value of a Dividend Swap that will reset in exactly one year for the front series,  $n$  years for the  $n^{\text{th}}$  series. These contracts are not traded, they are merely a tool to calibrate our models.

To compute these CMFs, we use the following idea: the  $n^{\text{th}}$  CMF is a weighted average between the  $n^{\text{th}}$  Rolled Series and the  $n + 1^{\text{th}}$  traded series. The objective is now to select the weights. We are going to use the average dividend profile (represented in Figure 2) as weights. We can see that at time 0, all the weight is on the current year, which is coherent. Moreover, we see that the

weight is worth 0.5 for  $t \approx 1 - \tau \approx 1 - 0.6 \approx 0.4$ , which means approximately in May, at the time where most of the companies are paying their dividends, that makes sense once again.

If we call  $\tilde{F}_T$  the  $T^{\text{th}}$  CMF's price, and  $F_T$  the  $T^{\text{th}}$  Rolled Series price, we have the following relation between the two prices [3]:

$$\tilde{F}_T(t) = [1 - \omega_n(T - t)]F_T(t) + \omega_n(T - t)F_{T+1}(t) \quad (3.1)$$

The CMF from 04/01/2016 to 17/05/2019 are represented in Figure 8. We can see, when we compare with Figure 7, that most of the discontinuities coming from seasonality effects have been cancelled. The curves are smoother, so we expect them to be easier to use for model calibration.

Then, to invert the relation (meaning going from CMFs to Rolled Series), we just need the value of one Tenor (the value of the Rolled Series). Indeed, we then have:

$$F_{T+1}(t) = \frac{\tilde{F}_T(t) - [1 - \omega_n(T - t)]F_T(t)}{\omega_n(T - t)}. \quad (3.2)$$

Since we know that the front year series (first series) has a very low volatility, we can use its price to invert all the contracts. Even if we make a mistake in our forecast for the first Tenor, it is unlikely to have a huge impact on the rest of the prices.



Figure 8: Dividend Futures - Constant Maturity Futures (CMF) from 04/01/2016 to 17/05/2019

### 3.3 Pricing

Now, we can start to model the Dividend Futures Term Structure. First, we have to compute their prices. The Dividend Futures pays, at expiry, the sum of the Dividends paid by the underlying stock/index during the elapsed year (here, year is from last third Friday of December to next third Friday of December). Then, the price of a Dividend Future with maturity  $T$  is the sum of non-discounted expected dividends paid during the period  $[T - 1, T]$ . So we have, following Buehler's theory [2]:

$$\begin{aligned}
 F_t^T &= \int_{T-1}^T \Delta_s ds \\
 &= \int_{T-1}^T \delta_s E_t[Y_s] ds \\
 &= \int_{T-1}^T \delta_s \left[ X_t^\alpha + (Y_t - X_t^\alpha) e^{-\kappa(s-t)} \right] ds \\
 &= Y_t e^{\kappa t} D_T^\kappa + X_t^\alpha (D_T^0 - e^{\kappa t} D_T^\kappa)
 \end{aligned} \tag{3.3}$$

with

$$D_T^\kappa = \int_{T-1}^T e^{-\kappa s} \delta_s ds \tag{3.4}$$

We now deal with the particular case of the front-year series. At time  $t$ , the dividends paid between time 0 and time  $t$  presents no uncertainty. So we shouldn't integrate the stochastic process on the whole year. Instead, we can split the integral this way ( $t \in [0, 1]$  is the fraction of the year elapsed):

$$\begin{aligned}
 F_t^1 &= \int_{0Y}^{1Y} \Delta_s ds \\
 &= \int_0^t \Delta_s ds + \int_t^{1Y} \Delta_s ds \\
 &= D(t) + \int_t^1 \delta_s \left[ X_t^\alpha + (Y_t - X_t^\alpha) e^{-\kappa(s-t)} \right] ds
 \end{aligned} \tag{3.5}$$

with

$$D(t) = \int_0^t \Delta_s ds \tag{3.6}$$

So, if we redefine (3.4) by

$$D_T^\kappa = \int_{\max(T-1, t)}^T e^{-\kappa s} \delta_s ds \tag{3.7}$$

we can write, using (3.3) and (3.5), an expression that will be correct for all maturities [3]:

$$F_t^T = D(t)H[t - (T - 1)] + Y_t e^{\kappa t} D_T^\kappa(t) + X_t^\alpha (D_T^0(t) - e^{\kappa t} D_T^\kappa(t)) \tag{3.8}$$



with

$$H(t) = \begin{cases} 0 & \text{if } t < 0 \\ 1 & \text{if } t \geq 0 \end{cases} \quad (3.9)$$

## 4 PCA Term Structure

Now, we want to use a model based on PCA [3]. PCA has many advantages. First, it allows to represent the term structure in lower dimension (we will take 3 Principal Components). Then, the first three Principal Components explain almost all the model's variation. Thus we do not lose a significant amount of information when using only 3 dimensions. And finally, we will see that it is a powerful tool for hedging purposes. The idea is to express the series in term of Level, Slope and Curvature (the name usually given to the three first Principal Components in this framework, in reference to their shape) and then use a linear regression to recover the prices. In this section, to make all the notations easier for the reader, we assume that we are studying an underlying with 10 series (DEDZx). This means we have 10 Rolled Time series and 9 CMFs.

### 4.1 General theory on PCA

First, we start by a reminder on PCA's theory [4]. This part is also useful to establish all the notations, since the terms employed often refers to different values according to the authors.

Assume we have  $N$  Time series. We denote by  $\Sigma$  their covariance matrix. This matrix is real and symmetric. So the Spectral Theorem says that  $\Sigma$  can be diagonalized. Moreover, we know that  $\Sigma$  is semi-positive Definite. So we can write<sup>1</sup>:

$$\Sigma = \Omega \mathcal{S}^2 \Omega^\dagger \quad (4.1)$$

Moreover, we know that  $\Omega$  obtained in Equation (4.1) is orthonormal. So  $\Omega \Omega^\dagger = \Omega^\dagger \Omega = I$  the identity matrix.

Assuming that the eigenvalues in  $\mathcal{S}$  are in decreasing order, we have the following interpretation: each column of  $\Omega$  is an eigenvector linked to the eigenvalue in  $\mathcal{S}^2$ , and represents a basis-vector of an orthonormal base. In the construction of  $\Omega$ , each new vector "explains" the highest model variance as possible. Thus, taking the  $k^{th}$  first columns in  $\Omega$ , and projecting our Time Series in this new space of lower dimension, we have a projection in lower dimension, and the variations in this new subspace explains  $\frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^N \lambda_i} \%$  of the original Time Series's variance ( $N$  is the dimension of  $\Sigma$ ,  $(\lambda_i)_{i=1, \dots, N}$  are the eigenvalues of  $\Sigma$ ).

<sup>1</sup>In this paper, we use  $\dagger$  to denote the transpose matrix in order to avoid any confusion with the times to expiry or maturities. This is actually the symbol for the Hermitian conjugate, but since all our coefficient are real, this is strictly equivalent to taking the transpose of the matrix.

This method allows us to work in a lower dimension space (we denote by  $k$  this dimension), so it is easier to calibrate our parameters, and a small number of vectors explains a high percentage of variance. In our case, the first  $k = 3$  vectors (called Principal Components) usually explains over 98% of the variance. Because of their shapes, these 3 vectors are called Level, Slope and Curvature (see Figure 9).

We denote by  $\bar{\Omega}$  the projection matrix to the new space, meaning that

$$\bar{\Omega} = \begin{bmatrix} \Omega_{1,1} & \dots & \Omega_{1,k} \\ \vdots & \ddots & \vdots \\ \Omega_{N,1} & \dots & \Omega_{N,k} \end{bmatrix} \quad (4.2)$$

Also, we denote by  $\tilde{\mathcal{F}}$  the diagonal matrix

$$\tilde{\mathcal{F}} = \begin{bmatrix} \mathcal{S}_{1,1} & & \\ & \ddots & \\ & & \mathcal{S}_{k,k} \end{bmatrix} \quad (4.3)$$

We select 3 Principal Components, and we work on 9 (or 10, according to which Time Series are used for the calibration, the CMFs or the Rolled Series) Time series. So the matrix  $\bar{\Omega}$  is a  $9 \times 3$  matrix, and  $\tilde{\mathcal{F}}$  is a  $3 \times 3$  matrix.

We define the Scores by:

$$x_t = \bar{\Omega}^\dagger F_t \quad (4.4)$$

where  $F_t$  is the vector containing the prices at time  $t$  ( $F_t = [F_t^1, \dots, F_t^N]^\dagger$ ). In other words,  $x_t$  is the projection in the new basis of the Dividend Futures prices. Note that, in our case,  $x_t$  has a dimension  $3 \times 1$  ( $x_t$  is represented in Figure 10).

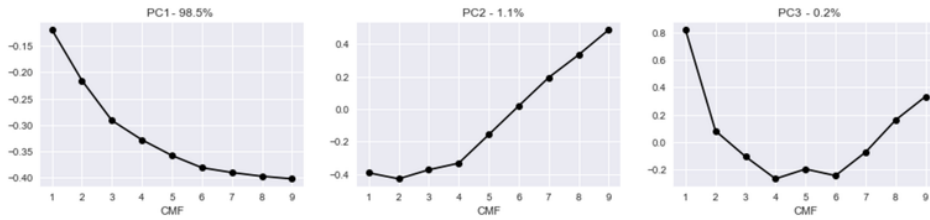


Figure 9: Principal Components - Data from 04/01/2016 to 17/05/2019. The value indicated in the titles represent the % of variance explained. PC1 is the Level, PC2 the Slope, and PC3 the curvature.

Now that we have all our notations, we can start to elaborate the model. The idea behind a PCA term structure is to assume that prices can be expressed as a Linear Regression of the scores.

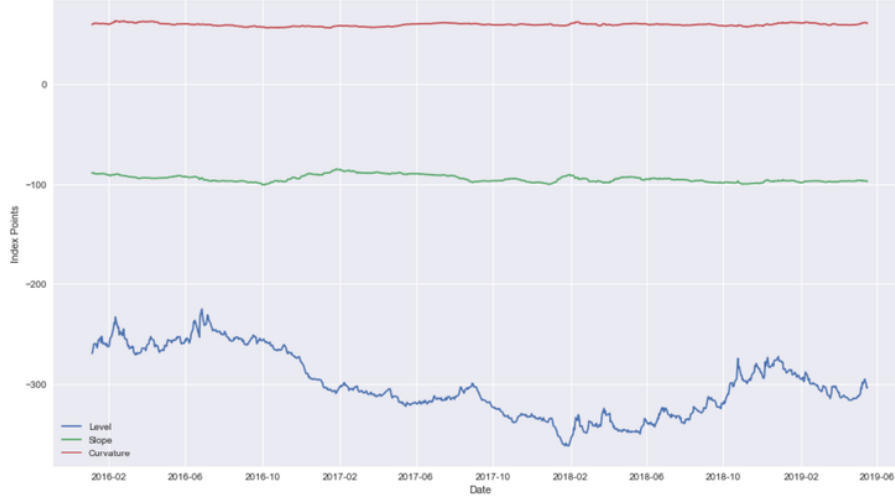


Figure 10: Scores as defined in Equation (4.4) - Data from 04/01/2016 to 17/05/2019.

Thus we write:

$$F_t = \omega + \bar{\Omega}x_t \quad (4.5)$$

with  $\omega$  an intercept (dimension  $9 \times 1$ ).

## 4.2 Stochastic model with PCA

### 4.2.1 Model

The basis of our model is Equation (4.5). This means that, to forecast prices, we need to compute the intercept  $\omega$ , a  $9 \times 1$  matrix in our case (done using a linear regression of  $F_t$  against  $x_t$ ), the parameter  $\Omega$  which is obtained performing a PCA using three components, a  $9 \times 3$  matrix in our case, and we need to forecast  $x_t$ . This is the main challenge of the model.

We are going to assume that  $x_t$  is a mean reverting process following the SDE:

$$dx_t = \kappa(\theta - x_t)dt + \mathcal{S}\rho dW_t \quad (4.6)$$

where  $dW_t = [dW_t^{(1)} \ dW_t^{(2)} \ dW_t^{(3)}]^\dagger$  is a vector of three uncorrelated Brownian Motions. We can write that  $[dW_t, dW_t] = \rho dt$ , with  $\rho$  a  $k \times k$  matrix (here a  $3 \times 3$  matrix), and that introduces one more parameter in our model. However, we will see that the three scores present no sign of correlation so we can fix  $\rho = I_3$ .

We are assuming that the process  $x_t$  is mean-reverting. See part 4.2.2 for more details on this assumption. Then our model is entirely defined by the parameters  $\omega, \Omega, \kappa, \theta, \mathcal{S}$ . As we previously

said,  $\omega$  is obtained using a linear regression,  $\Omega$  is obtained performing a PCA.  $\mathcal{S}$  is also obtained in the PCA (we select the eigenvalues corresponding to the eigenvectors in  $\Omega$ ). There is only  $\kappa$  and  $\theta$  left. We will see that  $\theta$  can be calibrated using a linear regression as well, and we will calibrate  $\kappa$  such that the model recovers the market volatility.

#### 4.2.2 Test of Mean-reversion

In the previous parts, we have made a strong assumption on the Scores Time Series: the three processes are mean-reverting. We want to show here that this assumption is actually in line with the experimental results.

We are going to use an Augmented Dickey-Fuller test to test for mean-reversion for each one of the three Scores.

The Augmented Dickey-Fuller test tests the null hypothesis "a unit root is present in the sample". Let  $(y_i)_{1 \leq i \leq N}$  be our Time Series (composed of  $N$  observations). We test the model:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 y_{t-1} + \dots + \delta_{p-1} y_{t-p+1} + \epsilon_t$$

with the hypothesis:

$$H_0: \gamma = 0 \text{ against } H_1: \gamma < 0$$

We want to show that the three scores presents no units roots. That means we want to reject the null hypothesis. Thus we are looking for p-values as low as possible. If the Time Series has no unit roots, it is mean-reverting. The results for the three scores are presented in Table 1.

Score	Critical Value	1%	5%	10 %	p-values
Level	-3.247	-3.433	-2.567	-2.863	0.01742
Slope	-1.926	-3.433	-2.567	-2.863	0.3198
Curvature	-2.465	-3.433	-2.567	-2.863	0.1242

Table 1: Result of the Augmented Dickey-Fuller Test. We calibrated using data from 05/05/2019 to 17/05/2009, so we could conclude on the mean-reverting behaviour on a long time period.

We can see that  $p - \text{value}^{\text{level}} < 0.05$  and Level's Critical value is lower than the 5% and the 10% values. So we can conclude that the Level is mean-reverting. Slope and Curvature do not seem to be mean-reverting (or at least not at a confidence level better than 90%). However, Level accounts for almost 90% of model variation on this period, so we will consider that the mean-reverting model we choose is relevant.



Figure 11: Scores Scaled - Data from 04/01/2016 to 17/05/2019. We call  $x_1$ ,  $x_2$ ,  $x_3$  the three scaled processes.

#### 4.2.3 Calibration

Now that we have evidences that the data are compatible with a mean-reverting behaviour model, we can start to think about the model calibration.

The objective of this part is to find a way to numerically approximate the parameters used in Equations (4.5) and (4.6), in order to be able to predict the Term Structure evolution.

##### Parameter $\rho$

For parameter  $\rho$ , we want to see if the three scores are correlated. However, we can note from Figure 10 that the three signals have an important difference in mean (close to a factor 10), and even visually it is easy to check that they don't have a comparable volatility. So, in order to capture the true correlation between these signals, we first scale them, i.e. we subtract to each one of them their mean and we divide them by the square root of their variance. A quick remark on that operation: the sklearn.preprocessing package in Python has a function StandardScaler that automatically compute this operation. We then have the three curves represented in Figure 11.

Now, we check the correlation matrix between these three scaled signals:

$$\rho = \begin{bmatrix} 1 & -2.9\text{e-}15 & -2.7\text{e-}15 \\ -2.9\text{e-}15 & 1 & -5.9\text{e-}16 \\ -2.7\text{e-}15 & -5.9\text{e-}16 & 1 \end{bmatrix} \approx I_3 \quad (4.7)$$

So we can set  $\rho = I_3$ .

**Parameter  $\omega$** 

To calibrate parameter  $\omega$ , we run the linear regression  $Y = A + BX$ , with  $Y = F_t$  and  $X = x_t$ , and we find  $\omega = A$ . We used Python's `sklearn.linear_model` [5] module to perform the linear regression.

**Parameter  $\theta$** 

In Section 4.2.2, we showed that the scores  $x_t$  indeed show a mean-reverting behavior. To calibrate  $\theta$ , the mean-reversion level, we are going to show that we can use a linear regression [6].

Since we have no signs of correlation, we can write (4.6) as:

$$dx_t = \kappa(\theta - x_t)dt + \mathcal{S}dW_t$$

So in discrete time, using  $\Delta t = 1$  Business day  $\approx 1/252$ :

$$x_{t+\Delta t} - x_t = \kappa(\theta - x_t)\Delta t + \mathcal{S}\sqrt{\Delta t}\epsilon \quad \text{with } \epsilon \text{ following a normal distribution.} \quad (4.8)$$

Let  $k = 1, 2, 3$ . We denote by  $r_t^k = x_{t+\Delta t}^k - x_t^k$  the linear returns of the  $k^{\text{th}}$  Series. We are going to calibrate each one of the three scores Time Series. We have that:

$$r_t^k = \kappa^k \theta^k \Delta t - \kappa^k \Delta t x_t^k + \xi^k \quad (4.9)$$

where  $\xi^k$  are Normal (non-standardized) errors. We can read this as  $Y = B + AX$  with  $Y = r_t^k$ ,  $X = x_t^k$ ,  $B = \kappa^k \theta^k \Delta t$ ,  $A = -\kappa^k \Delta t$ . Then we have

$$\theta^k = -\frac{B}{A} \quad (4.10)$$

**Parameter  $\kappa$** 

We know that, in an Ornstein-Uhlenbeck process  $du_t = \kappa(\theta - u_t)dt + \sigma dW_t$ ,  $\kappa$  and  $\sigma$  have an opposite effect on volatility. Indeed, increasing  $\sigma$  increases the process' volatility, and increasing  $\kappa$  lowers the volatility.

In our model, the volatility matrix is fixed after performing the PCA. So the only free parameter that we can modify to adjust the volatility is  $\kappa$ . Thus, to calibrate  $\kappa$ , we are going to adjust our model's volatility to the market volatility [3].

In order to make the computations easier to read, we will give the dimensions we used for calibration during the proofs. We used the CMFs to calibrate our models, the underlying was the SX5E. So we had 10 series, 9 CMFs, and as previously explained we will use a representation in three dimensions for our data (Level, Slope, Curvature). However, for some underlyings, only 5 series are traded, and we can compute 4 CMFs. We call  $\Omega$  and  $\mathcal{S}$  the result of the PCA (that means  $\Omega$  contains three columns, the eigenvectors associated to the three largest eigenvalues of the covariance matrix.  $\mathcal{S}$  contains the squared roots of these three largest eigenvalues). That means  $\Omega$  is a  $9 \times 3$  matrix, and  $\mathcal{S}$  is a  $3 \times 3$  diagonal matrix. We denote by  $N$  the number of times series modeled (i.e. here  $N = 9$ ).

We will calibrate  $\kappa$  assuming that the matrix is diagonalisable. Thus the matrix is completely defined by its three eigenvalues (R. Rebonato, I. Saroka and V. Putyatin [7] proved that if  $\kappa$  can be assumed diagonalisable, it cannot be a diagonal matrix). Since we have three degrees of freedom, we will select three series to compute the calibrations. We will call these three series 'reference series', and denote them by  $T_1^*$ ,  $T_2^*$ ,  $T_3^*$ .

We denote by

$$F_t = \begin{bmatrix} F_t^{T_1} \\ F_t^{T_2} \\ \vdots \\ F_t^{T_N} \end{bmatrix}, \text{ with } N = 9 \text{ here.} \quad (4.11)$$

the vector of - all - prices.

Also, we call

$$\tilde{F}_t = \begin{bmatrix} F_t^{T_1^*} \\ F_t^{T_2^*} \\ F_t^{T_3^*} \end{bmatrix} \quad (4.12)$$

the vector describing the evolution of the three references series.

We set:

$$\mathbf{e} = \underbrace{\begin{bmatrix} 0 & \dots & \underbrace{1}_{T_1^*} & 0 & \dots & \dots & \dots & 0 \\ 0 & 0 & \dots & \underbrace{1}_{T_2^*} & 0 & \dots & \dots & 0 \\ 0 & 0 & 0 & \dots & \underbrace{1}_{T_3^*} & 0 & \dots & 0 \end{bmatrix}}_{N=9} \quad (4.13)$$

and we notice that:

$$\tilde{F}_t = \mathbf{e}F_t \quad (4.14)$$

Or, according to our model we have:

$$F_t = \omega + \Omega x_t \quad (4.15)$$

So, using (4.14) in (4.15), we have:

$$\tilde{F}_t = \mathbf{e}F_t = \mathbf{e}\omega + \mathbf{e}\Omega x_t = \tilde{\omega} + \tilde{\Omega}x_t \quad (4.16)$$

where  $\tilde{\Omega}$  is a  $3 \times 3$  matrix, and  $\tilde{\omega}$  is a  $3 \times 1$  matrix.

Now, recall that, for any maturity T:

$$F_t^T = \int_{\max(t, T-1)}^T \delta_s E_t[Y_s] ds + D(t)H[t - (T-1)] \quad (4.17)$$

with  $D$  and  $H$  defined as in (3.6) and (3.9). We then set ( $u$  is a  $3 \times 1$  matrix,  $U$  a scalar):

$$Y_t = U + u^\dagger x_t \quad (4.18)$$

which leads to:

$$\begin{aligned} F_t^T &= D(t)H[t - (T - 1)] + \int_{\max(t, T-1)}^T \delta_s (U + u^\dagger E_t[x_s]) ds \\ &= D(t)H[t - (T - 1)] + D_T^0(t)U + u^\dagger \int_{\max(t, T-1)}^T \delta_s (e^{\kappa(t-s)}x_t + (I_3 - e^{\kappa(t-s)}\theta)) ds \\ &= D(t)H[t - (T - 1)] + D_T^0(t)U + u^\dagger (D_T^0(t) - e^{\kappa t}D_t^\kappa(t))\theta + u^\dagger e^{\kappa t}D_T^\kappa x_t \end{aligned} \quad (4.19)$$

where  $D_t^\kappa(t)$  is defined using the matrix exponential function, i.e.

$$D_T^\kappa(t) = \int_{\max(t, T-1)}^T \delta_s \underbrace{e^{-\kappa s}}_{\text{Matrix Exponential}} ds.$$

A quick word on the Matrix Exponential function. We have that  $e^X = \sum_{i=0}^{\infty} \frac{X^i}{i!}$  for  $X$  a square matrix. It is then easy to show that for a diagonalisable matrix  $X$  such that  $X = P^{-1}DP$ , we have  $e^X = P^{-1}e^D P$ . We will apply this property to both the matrices  $\kappa t$  and  $D_T^\kappa(t)$ .

Now, we select  $\bar{N}$  series (for us, it will be first  $\bar{N} = 3$  to express  $\kappa$  and then  $\bar{N} = 9$  to perform the calibration). If we write:

$$\bar{\mathcal{B}}_t^\dagger = \begin{bmatrix} u^\dagger e^{\kappa t} D_{T_1}^\kappa(t) \\ \vdots \\ u^\dagger e^{\kappa t} D_{T_{\bar{N}}}^\kappa(t) \end{bmatrix} \quad (4.20)$$

then we have ( $\bar{F}_t$  is the price column of the  $\bar{N}$  selected Time Series):

$$\bar{F}_t = \bar{\mathcal{A}}_t + \bar{\mathcal{B}}_t x_t \quad (4.21)$$

Let consider for now that we are in the calibration space, i.e. we are using only our 3 references series. As discussed previously, we assume that  $\kappa$  is diagonalisable. So we write:

$$\kappa = k\Lambda k^{-1} = k \begin{bmatrix} \kappa_1 & 0 & 0 \\ 0 & \kappa_2 & 0 \\ 0 & 0 & \kappa_3 \end{bmatrix} k^{-1} \quad (4.22)$$

Then we have, using (4.20) with  $\bar{N} = 3$  and (4.22):

$$\begin{aligned} \tilde{\mathcal{B}}_t &= \begin{bmatrix} D_{T_1}^{\kappa_1^\dagger}(t)e^{\kappa_1^\dagger t}u & D_{T_2}^{\kappa_2^\dagger}(t)e^{\kappa_2^\dagger t}u & D_{T_3}^{\kappa_3^\dagger}(t)e^{\kappa_3^\dagger t}u \end{bmatrix} \\ &= \begin{bmatrix} (k^{-1})^\dagger D_{T_1}^\Lambda(t)e^{\Lambda t}k^\dagger u & (k^{-1})^\dagger D_{T_2}^\Lambda(t)e^{\Lambda t}k^\dagger u & (k^{-1})^\dagger D_{T_3}^\Lambda(t)e^{\Lambda t}k^\dagger u \end{bmatrix} \\ &= (k^{-1})^\dagger \begin{bmatrix} \text{diag}(k^\dagger u)D_{T_1}^\Lambda(t)e^{\Lambda t} & \text{diag}(k^\dagger u)D_{T_2}^\Lambda(t)e^{\Lambda t} & \text{diag}(k^\dagger u)D_{T_3}^\Lambda(t)e^{\Lambda t} \end{bmatrix} \\ &= (k^{-1})^\dagger \text{diag}(k^\dagger u)\mathcal{F}^\dagger \end{aligned} \quad (4.23)$$



with

$$\mathcal{F}^\dagger = \begin{bmatrix} D_{T_1}^{\kappa_1}(t)e^{\kappa_1 t} & D_{T_2}^{\kappa_1}(t)e^{\kappa_1 t} & D_{T_3}^{\kappa_1}(t)e^{\kappa_1 t} \\ D_{T_1}^{\kappa_2}(t)e^{\kappa_2 t} & D_{T_2}^{\kappa_2}(t)e^{\kappa_2 t} & D_{T_3}^{\kappa_2}(t)e^{\kappa_2 t} \\ D_{T_1}^{\kappa_3}(t)e^{\kappa_3 t} & D_{T_2}^{\kappa_3}(t)e^{\kappa_3 t} & D_{T_3}^{\kappa_3}(t)e^{\kappa_3 t} \end{bmatrix} \quad (4.24)$$

Note that  $k^\dagger u$  is a  $3 \times 1$  matrix. The operator  $\text{diag}$  creates a  $3 \times 3$  diagonal matrix, whose coefficients are the three coefficients in  $k^\dagger u$ . Also, it is clear that  $\tilde{\mathcal{B}}_t$  is a  $3 \times 3$  matrix.

So, now we have these two equations for  $\tilde{F}_t$ :

$$\begin{cases} \tilde{F}_t = \tilde{\omega} + \tilde{\Omega}x_t \\ \tilde{F}_t = \tilde{\mathcal{A}}_t + \tilde{\mathcal{B}}_t^\dagger x_t \end{cases} \quad (4.25)$$

By identification in (4.25), we have  $\tilde{\mathcal{B}}_t^\dagger = \tilde{\Omega}$ . Then using (4.23) it follows:

$$\begin{aligned} \tilde{\Omega} &= (k^{-1})^\dagger \text{diag}(k^\dagger u) \mathcal{F}^\dagger \\ (k)^{-1} &= \text{diag}(k^\dagger u)^{-1} \mathcal{F}^{-1} \tilde{\Omega} \end{aligned} \quad (4.26)$$

Once we have an expression for  $k$ , we can deduce an expression for  $\kappa$  using the model's parameters. We wrote  $\kappa$  as a function of  $k$  in (4.22). We then have:

$$\begin{aligned} \kappa &= k \Lambda k^{-1} \\ &= \tilde{\Omega}^{-1} \mathcal{F} \text{diag}(k^\dagger u) \Lambda \text{diag}(k^\dagger u) \mathcal{F}^{-1} \tilde{\Omega} \\ &\text{and since diagonal matrices commute} \\ \kappa &= \tilde{\Omega}^{-1} \mathcal{F} \Lambda \mathcal{F}^{-1} \tilde{\Omega} \end{aligned} \quad (4.27)$$

Then by identification, we have an easier way to write  $k$ :

$$k = \tilde{\Omega}^{-1} \mathcal{F} \quad (4.28)$$

We now have all the parameters we need to find  $u$ .

Equation (4.28) gives us that:

$$k^{-1} = \mathcal{F}^{-1} \tilde{\Omega}$$

Or, equation (4.26) gave us:

$$k^{-1} = \text{diag}(k^\dagger u)^{-1} \mathcal{F}^{-1} \tilde{\Omega}$$

So we want  $u$  such that:

$$\begin{aligned} \text{diag}(k^\dagger u)^{-1} &= I_3 \\ k^\dagger u &= \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^\dagger \\ \mathcal{F}^\dagger (\tilde{\Omega}^{-1})^\dagger u &= \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^\dagger \end{aligned}$$

We set  $u = \tilde{\Omega}^\dagger v$ , for  $v$  a matrix. Then

$$\mathcal{F}^\dagger(\tilde{\Omega}^{-1})^\dagger u = \mathcal{F}^\dagger v$$

Then we set  $v = (\mathcal{F}^\dagger)^{-1} \mathbf{1}$ , with  $\mathbf{1}^\dagger = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ . So

$$\mathcal{F}^\dagger(\tilde{\Omega}^{-1})^\dagger u = \mathcal{F}^\dagger(\mathcal{F}^\dagger)^{-1} \mathbf{1} = \mathbf{1} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^\dagger$$

Finally, we have the following expression for  $u$ :

$$u = \tilde{\Omega}^\dagger (\mathcal{F}^\dagger)^{-1} \mathbf{1} \quad (4.29)$$

We quickly verify if all the matrices dimensions are coherent:  $\tilde{\Omega}$  and  $\mathcal{F}$  are  $3 \times 3$  matrices, and  $\mathbf{1}$  is a  $3 \times 1$  matrix. So we indeed have that  $u$  is a  $3 \times 1$  matrix.

Now, let's use all the Time Series (nine in our case). We are going to use equation (4.20) with  $\bar{N} = N$  ( $= 9$ ), knowing that equations (4.28) and (4.29) give us the expression of  $u$  and  $k$  as functions of the problem (determined with the references). Note that  $\mathcal{B}_t^\dagger$  is a  $\bar{N} \times 3$  ( $= 9 \times 3$ ) matrix. We call  $\mathcal{B}_t$  the matrix  $\mathcal{B}_t^{\bar{N}=N=9}$

$$\begin{aligned} \mathcal{B}_t^\dagger &= \begin{bmatrix} u^\dagger e^{\kappa t} D_{T_1}^\kappa(t) \\ \vdots \\ u^\dagger e^{\kappa t} D_{T_N}^\kappa(t) \end{bmatrix} = \begin{bmatrix} \mathbf{1}^\dagger \mathcal{F}^{-1} \tilde{\Omega} k e^{\Lambda t} D_{T_1}^\Lambda(t) k^{-1} \\ \vdots \\ \underbrace{\mathbf{1}^\dagger}_{1 \times 3} \underbrace{\mathcal{F}^{-1}}_{3 \times 3} \underbrace{\tilde{\Omega}}_{3 \times 3} \underbrace{k}_{3 \times 3} \underbrace{e^{\Lambda t}}_{3 \times 3} \underbrace{D_{T_N}^\Lambda(t)}_{3 \times 3} \underbrace{k^{-1}}_{3 \times 3} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{1}^\dagger \mathcal{F}^{-1} \tilde{\Omega} \tilde{\Omega}^{-1} \mathcal{F} e^{\Lambda t} D_{T_1}^\Lambda(t) \mathcal{F}^{-1} \tilde{\Omega} \\ \vdots \\ \mathbf{1}^\dagger \mathcal{F}^{-1} \tilde{\Omega} \tilde{\Omega}^{-1} \mathcal{F} e^{\Lambda t} D_{T_N}^\Lambda(t) \mathcal{F}^{-1} \tilde{\Omega} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{1}^\dagger e^{\Lambda t} D_{T_1}^\Lambda(t) \\ \vdots \\ \mathbf{1}^\dagger e^{\Lambda t} D_{T_N}^\Lambda(t) \end{bmatrix} \mathcal{F}^{-1} \tilde{\Omega} \\ &= \begin{bmatrix} e^{\kappa_1 t} D_{T_1}^{\kappa_1}(t) & e^{\kappa_2 t} D_{T_1}^{\kappa_2}(t) & e^{\kappa_3 t} D_{T_1}^{\kappa_3}(t) \\ \vdots & \vdots & \vdots \\ e^{\kappa_1 t} D_{T_N}^{\kappa_1}(t) & e^{\kappa_2 t} D_{T_N}^{\kappa_2}(t) & e^{\kappa_3 t} D_{T_N}^{\kappa_3}(t) \end{bmatrix} \mathcal{F}^{-1} \tilde{\Omega} \end{aligned}$$

Finally, we can write:

$$\mathcal{B}_t^\dagger = \bar{\mathcal{F}} \mathcal{F}^{-1} \tilde{\Omega} \quad (4.30)$$

with

$$\bar{\mathcal{F}} = \begin{bmatrix} e^{\kappa_1 t} D_{T_1}^{\kappa_1}(t) & e^{\kappa_2 t} D_{T_1}^{\kappa_2}(t) & e^{\kappa_3 t} D_{T_1}^{\kappa_3}(t) \\ \vdots & \vdots & \vdots \\ e^{\kappa_1 t} D_{T_N}^{\kappa_1}(t) & e^{\kappa_2 t} D_{T_N}^{\kappa_2}(t) & e^{\kappa_3 t} D_{T_N}^{\kappa_3}(t) \end{bmatrix} \quad (4.31)$$

Or, equation (4.21) gives us (since we took  $\bar{N}$  all the series available, in our case, 9):

$$F_t = \mathcal{A} + \mathcal{B}_t x_t$$

And we know that:

$$F_t = \omega + \Omega x_t$$

So we can approximate our model's covariance by:

$$\bar{\Sigma} = \mathcal{B}_t^\dagger \mathcal{S} \mathcal{B}_t \quad (4.32)$$

Then, calibrating  $\kappa$  is equivalent to solving the following minimisation-under-constraint problem presented in Equation (4.33) (we denote by  $\Sigma$  the true covariance matrix). We denote by  $\kappa_1^*$ ,  $\kappa_2^*$ ,  $\kappa_3^*$  the result of the optimisation problem, and by  $\|\cdot\|$  the Frobenius norm.

$$(\kappa_1^*, \kappa_2^*, \kappa_3^*) = \underset{\kappa_1, \kappa_2, \kappa_3}{\operatorname{argmin}} (\|\bar{\Sigma} - \Sigma\| \mid \kappa_1 \neq \kappa_2 \neq \kappa_3, \kappa_1 > 0, \kappa_2 > 0, \kappa_3 > 0) \quad (4.33)$$

Then, using Equation (4.27), we find:

$$\kappa = \tilde{\Omega}^{-1} \bar{\mathcal{F}} \begin{bmatrix} \kappa_1^* & 0 & 0 \\ 0 & \kappa_2^* & 0 \\ 0 & 0 & \kappa_3^* \end{bmatrix} \bar{\mathcal{F}}^{-1} \tilde{\Omega} \quad (4.34)$$

#### 4.2.4 Results

We are going to present the results at two different times. The first date - corresponding to 15/04/2019 - corresponds to a period where the instantaneous volatility is close to the global volatility over the calibration period. The second date - corresponding to 17/05/2019 - corresponds to a period where the instantaneous volatility is different from the global volatility over the calibration period.

It will be observed that for periods where the instantaneous volatility is not close enough from the volatility in the calibration period, the expected prices don't match well the realised prices. It is one of the property of a stochastic model: it gives a trend given a calibration period, not a step-by-step price evolution based on instantaneous properties.

First, we present the result of the calibration using data between 05/01/2016 and 17/05/2019. We can see in Figure 12 that the model covariance matrix and the true covariance matrix are close (less than 1.5% of relative difference in norm, using Frobenius Norm). So the calibration of

parameter  $\kappa$  seems coherent. In Figure 13 we can see a matrix of the relative errors between the model covariance and the true covariance matrices. The values of  $\kappa_1^*$ ,  $\kappa_2^*$  and  $\kappa_3^*$  are given in Table 2.

$\kappa_1^*$	$\kappa_2^*$	$\kappa_3^*$
0.131	0.315	1e-8

Table 2: Values of  $\kappa_1^*$ ,  $\kappa_2^*$  and  $\kappa_3^*$ .

The corresponding  $\kappa$  matrix is - rounding to only 3 decimals:

$$\kappa = \begin{bmatrix} -0.0753 & 0.366 & 0.0855 \\ -0.128 & -0.243 & -0.770 \\ 0.113 & 0.297 & 0.765 \end{bmatrix} \quad (4.35)$$

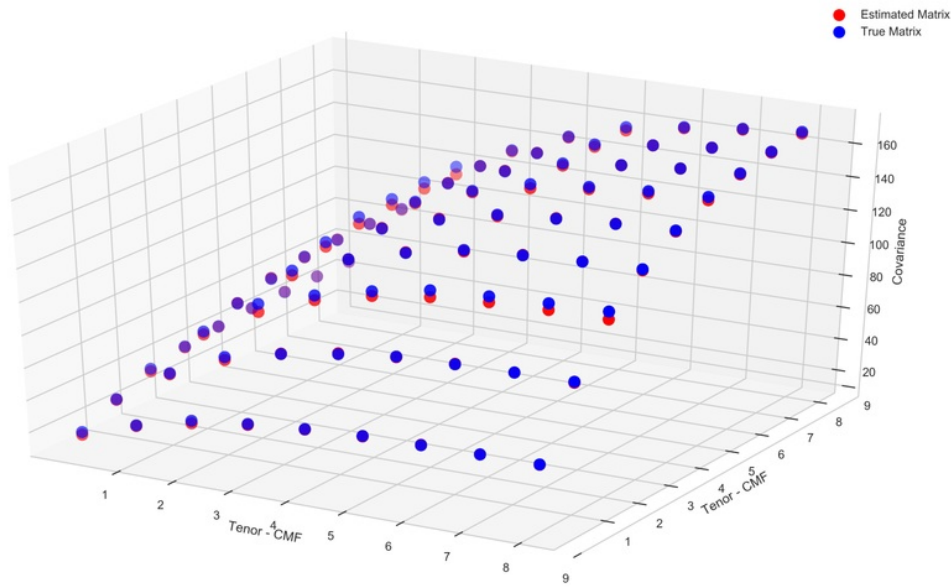


Figure 12: Comparison between True Covariance Matrix and Model Covariance Matrix - Time Frame for Calibration: 05/01/16 - 17/05/2019. We can see that the two matrices are very close. In Figure 13 we can see a matrix representing the relative error between these two matrices.

Now, we want to 'graphically' verify if we have a good calibration for  $\kappa$ . We generate one path using an Euler Scheme. In Figure 14, we use the date 15/04/2019 as starting point of the simulation. In Figure 15 we use the date 17/05/2019 as starting point of the simulation.

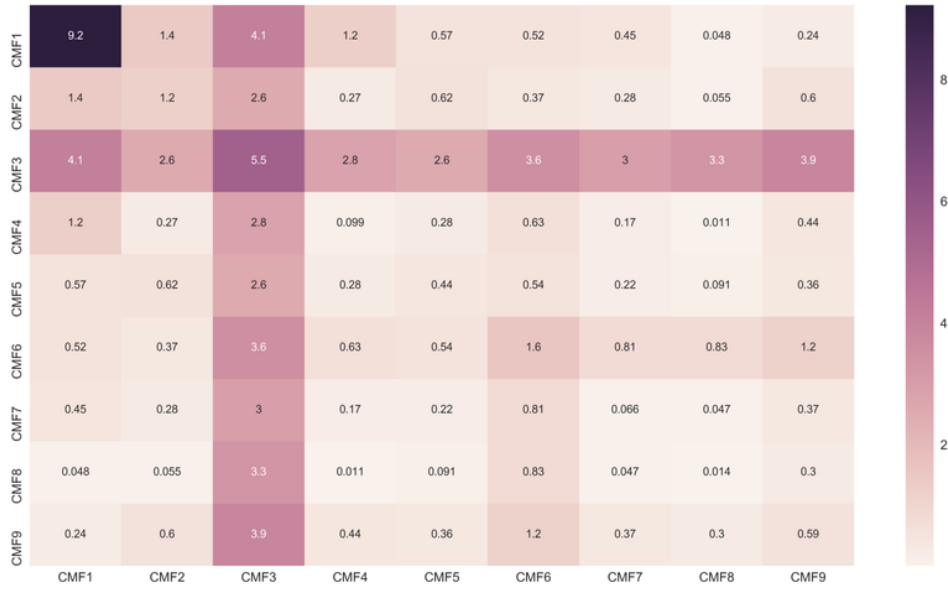


Figure 13: Relative error (%) between true Covariance matrix and estimated Covariance matrix - Three references Series for Calibration: CMF 2, CMF 4 and CMF 8. Time Frame for Calibration: 05/01/16 - 17/05/19. We can see that the two matrices are close, with the highest relative error achieved for the Constant Maturity Future 1. Its low volatility makes this Tenor hard to calibrate in all the models studied in this paper.

Finally, we can look at our model's predictions. We used a Vasicek model (Equation (4.6)) to describe the Term Structure's evolution. So we know that we have a closed form solution to express the conditional expectations of the Scores (and then the Prices, since they are a linear combination of the Scores). We have that:

$$\mathbb{E}_t[x_s] = e^{\kappa(t-s)}x_t + (1 - e^{\kappa(t-s)}) \cdot \theta \text{ for } t < s. \quad (4.36)$$

In our case,  $t$  is either 15/04/2019 or 17/05/2019. We used Equation (4.36) to predict the prices over 10 days, starting the 15/04/2019 in Figure 16 and starting the 17/05/2019 in Figure 17. We can see that the results obtained starting the 15/04/2019 are better. This might be because the instantaneous volatility for this period is closer to the volatility over the entire calibration period, whereas volatility the 17/05/2019 was a bit higher.

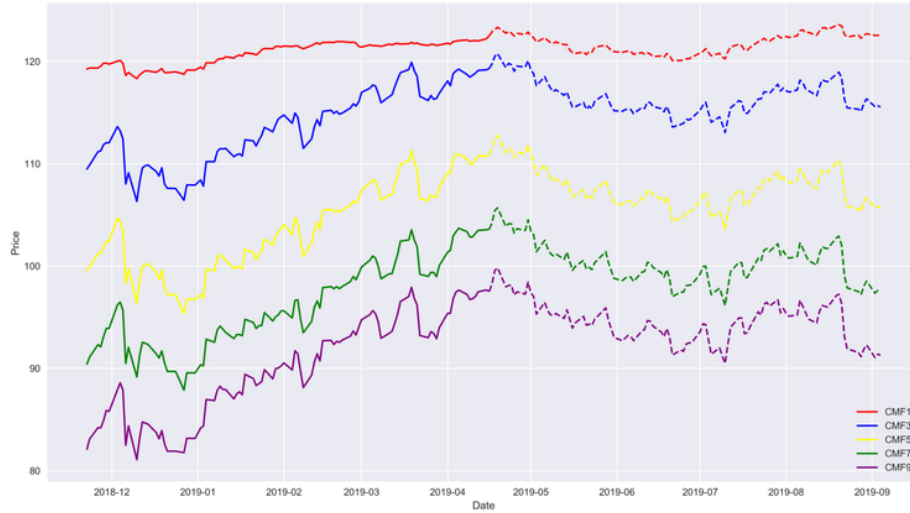


Figure 14: Path generated with Euler Scheme for odd CMFs over 100 days - Calibration period: 05/01/16 - 15/04/19. The continuous lines represent realised prices, the dashed lines represents the simulations. We can see that the amplitude of the movements predicted are in line with the historical one. An important behaviour well represented here is the strong link between the CMFs: usually (and particularly for important movements in price) all the CMFs move in the same direction.

### 4.3 Fractional Vasicek Model

An idea to try to improve the previous model is to use fractional Brownian Motions. The increments in the random shocks are not independent anymore. The first step to use this tool (and the fVasicek model) is to determine the Hurst exponent of the three scores.

#### 4.3.1 Hurst Exponent

The Hurst exponent can be computed using the R/S analysis method [8] - we are studying a Time Series  $X = X_1, X_2, \dots, X_n$  with  $n$  the number of points:

1. Calculate  $m = \frac{1}{n} \sum_{i=1}^n X_i$
2. Calculate  $Y_t = X_t - m$  for  $t = 1, 2, \dots, n$ .
3. Calculate  $Z_t = \sum_{i=1}^t Y_i$ ,  $t = 1, 2, \dots, n$
4. Calculate  $R_t = \max(Z_1, Z_2, \dots, Z_t) - \min(Z_1, Z_2, \dots, Z_t)$  for  $t = 1, 2, \dots, n$ .
5. Calculate  $S_t = \sqrt{\frac{1}{t} \sum_{i=1}^t (X_i - u)^2}$  for  $t = 1, 2, \dots, n$  where  $u$  is the mean value of  $X_1, X_2, \dots, X_t$ .

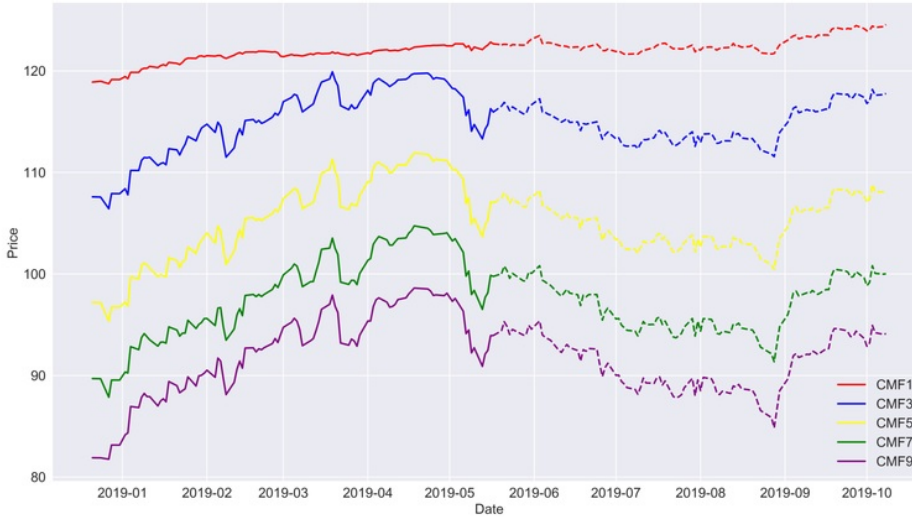


Figure 15: Path generated with Euler Scheme for odd CMFs over 100 days - Calibration period: 05/01/16 - 17/05/19. The continuous lines represent realised prices, the dashed lines represents the simulations. Once again, we can see that the amplitude of the movements predicted are in line with the previously realised ones.

6. Calculate the series  $(R/S)$ :  $(R/S)_t = R_t/S_t$  for  $t = 1, 2, \dots, n$ .

Hurst found that:

$$(R/S)_t = c \cdot t^H \quad \text{with } c \text{ a constant.} \quad (4.37)$$

So, we can then write, for  $t \in [1, n]$  - since  $R_t$  and  $S_t$  are positive for each  $t$ , we have that  $(R/S)_t$  is positive and  $c$  too, which make the following correct:

$$\log(R/S)_t = \tilde{c} + H \cdot \log(t) \quad \text{with } \tilde{c} \text{ a constant.} \quad (4.38)$$

Then, we can compute the Hurst exponent using Equation (4.38) and a linear regression, with  $Y = \log(R/S)_t$ ,  $X = \log(t)$ , and  $Y = \tilde{c} + H \cdot X$ .

We plotted the regression for each of the three series (Level, Slope, Curvature) in Figure 18 and 19.

#### 4.3.2 Fractional Brownian Motion and Model

Now that we have computed the Hurst Exponent, we can introduce the Fractional Brownian Motion. A fractional Brownian Motion (denoted  $B^H$ ) is, quoting K. Tanaka, W. Xiao and J. Yun

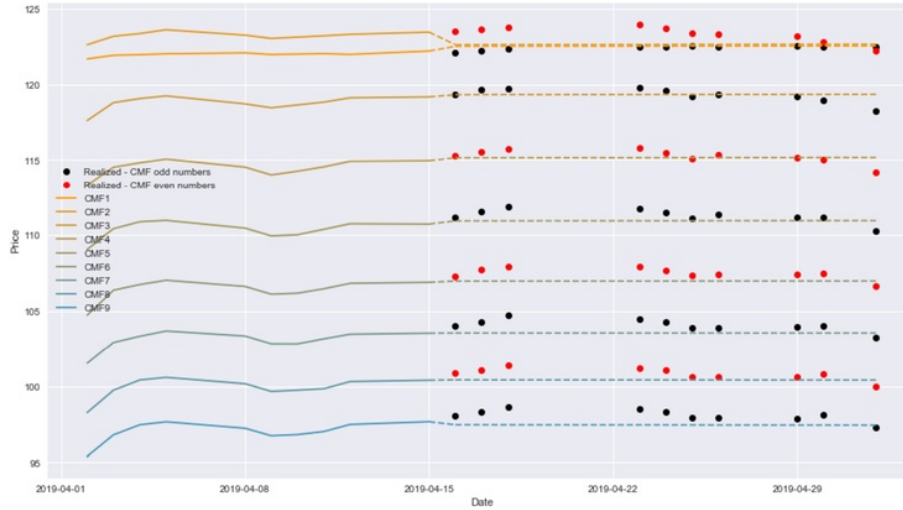


Figure 16: 10 days forecast starting from 15/04/2019, using the closed form solution of a Vasicek Model. The continuous lines represent realised prices, the dashed lines represents the simulations. The dots represent the market prices after the 15/04/2019. We can see that the predicted evolution is similar to the realised evolution.

[9], "a zero-mean Gaussian process, defined on a complete probability space  $(\Omega, \mathcal{F}, \mathbb{P})$ " with the covariance function defined in Equation (4.39).

$$\mathbb{E}[B_t^H B_s^H] = \frac{1}{2}(|t|^{2H} + |s|^{2H} - |t - s|^{2H}) \quad \text{with } t, s \in \mathbb{R} \quad (4.39)$$

The process  $B^H$  is self similar, meaning that for a positive real number  $a$ , a real number  $t$ , we have  $B_{at}^H \stackrel{d}{=} a^H B_t^H$ . For  $H = \frac{1}{2}$ , it becomes a Brownian Motion. There is different way to simulate a fractional Brownian Motion, we briefly describe the Cholesky method. See [10] for more precision.

The Cholesky method consists of roughly three steps:

1. Construct a matrix  $\Gamma$  such that  $\Gamma_{i,j} = \mathbb{E}[B_{t_i}^H B_{t_j}^H]$
2. Perform a Cholesky decomposition on  $\Gamma$ , and call  $\Sigma$  the resulting matrix,
3. Set  $u = \Sigma v$ , where  $v$  is vector of independent standard Gaussian variables.

Then  $u$  yields a fBM path.

Now, we will focus on the model, following W. Xiao and J. Yu [11]. Since we already showed that the Scores don't present any signs of correlation, we are going to model each Score independently.



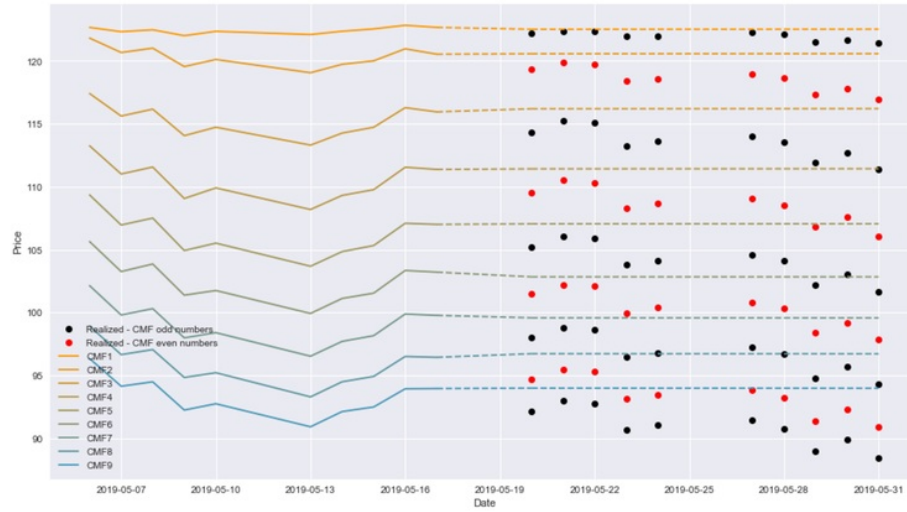


Figure 17: 10 days forecast starting from 17/05/2019, using the closed form solution of a Vasicek Model. The continuous lines represent realised prices, the dashed lines represents the simulations. The dots represent the market prices after the 17/05/2019. We can see that there is some differences between the realised and forecasted price evolution.

Each Score will follow this Equation:

$$dx_t = \kappa(\theta - x_t)dt + \sigma dB_t^H \quad (4.40)$$

where  $x_t$  represents the Score at time t.

Now, we want to estimate  $\kappa$  and  $\theta$ , so we can launch some numerical simulations. We are going to use the Least Squares Estimator, as presented by W. Xiao and J. Yu [11]:

$$\hat{\kappa}_{LS} = \frac{(X_T - X_0) \int_0^T x_t dt - T \int_0^T x_t dx_t}{T \int_0^T x_t^2 dt - \left( \int_0^T x_t dt \right)^2} \quad (4.41)$$

$$\hat{\theta}_{LS} = \frac{(X_T - X_0) \int_0^T x_t^2 dt - \int_0^T x_t dx_t \int_0^T x_t dt}{(X_T - X_0) \int_0^T x_t dt - T \int_0^T x_t dx_t}$$

So we generated the fractional Brownian motions, and we plotted in Figure 20 and 21 one simulation (using Euler Scheme) of the Prices Evolution using a fVasicek Model. We can see that the results of the simulation are in line with the previous realisation. The signals might seem a bit rougher than previously, which was the objective when we introduced fractional Brownian Motion. Finally, the first Tenor seems a bit more volatile than previously, which is not necessary one of its characteristics (but once again, the first Tenor is the most complicated to calibrate in all the different models presented in this paper).

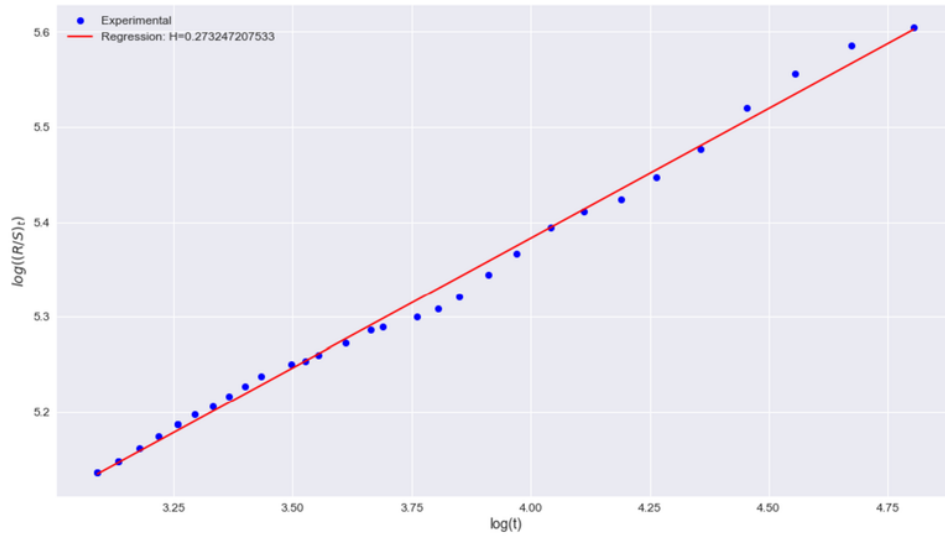


Figure 18: Hurst Exponent for the Level ( $H \approx 0.27$  and  $R^2 = 0.9946$ ).

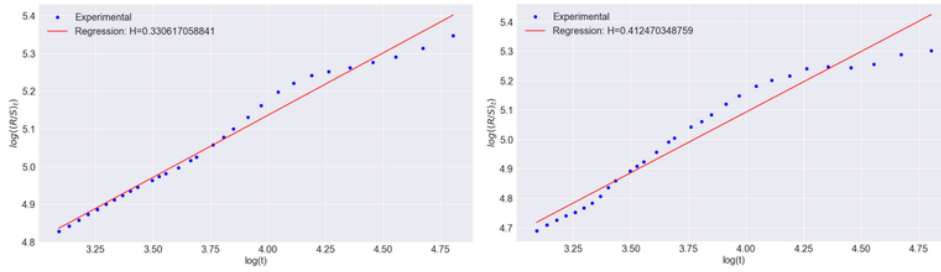


Figure 19: Hurst Exponent for the Slope and the Curvature ( $H \approx 0.33$  and  $H \approx 0.41$  respectively).  $R^2$  is slightly better for the Level ( $R^2 \approx 0.9817$  and  $R^2 \approx 0.9500$  for Slope and Curvature respectively).

Then, in Figure 22 and 23 we plotted the results of a Monte-Carlo simulation for the two dates. The results are very close to the previous ones (we checked numerically, there is not a lot of difference between the two results).

So, the fractional Vasicek Model might bring a bit of precision to the global Term Structure. However, we still didn't managed to find a Hedging tool. This is the objective of the next part.

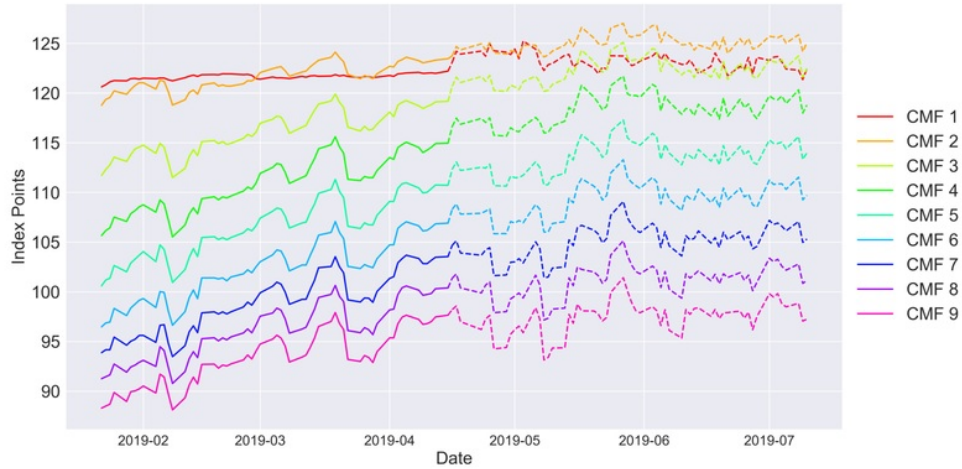


Figure 20: One simulation launched on the 15/04/2019, using the fVasicek model for each one of the three scores. The continuous line is the past prices, the dashed line the result of the simulation.

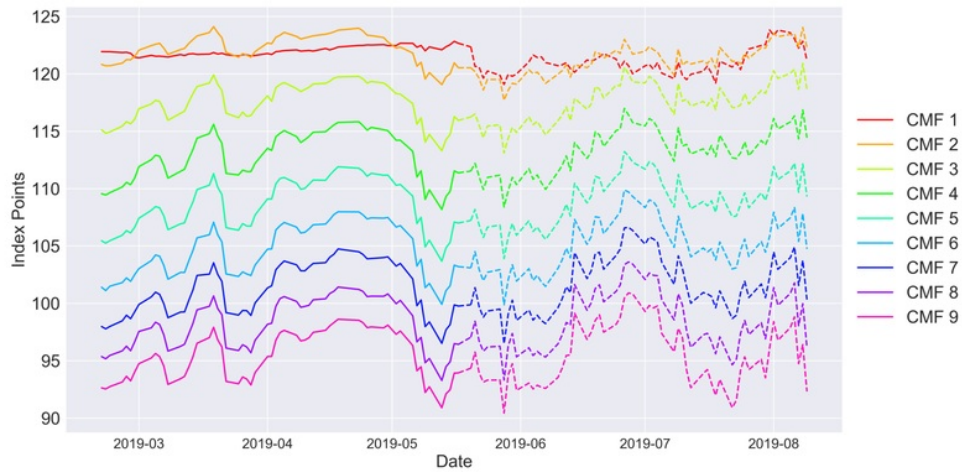


Figure 21: One simulation launched on the 17/05/2019, using the fVasicek model for each one of the three scores. The continuous line is the past prices, the dashed line the result of the simulation.

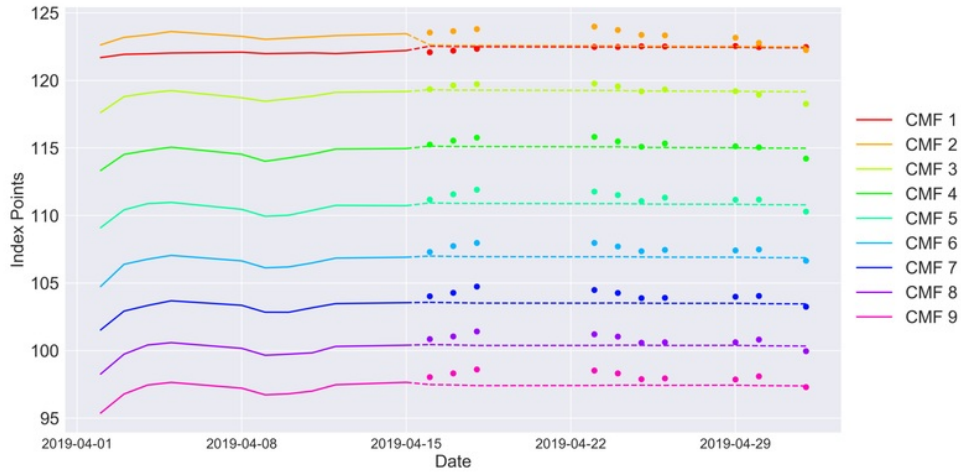


Figure 22: Monte-Carlo simulation launched on the 15/04/2019, using the fVasicek model for each one of the three scores. The continuous line is the past prices, the dashed line the result of the simulation, the dots are the observed prices at the predicted days.

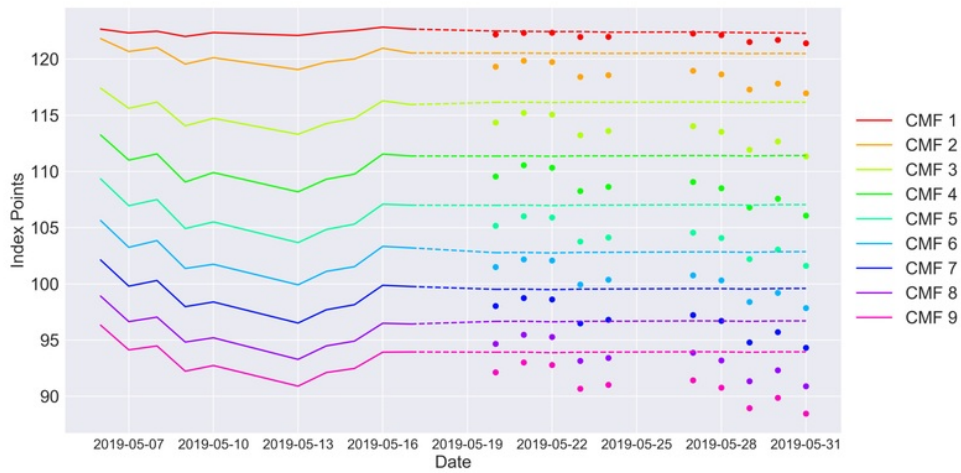


Figure 23: Monte-Carlo simulation launched on the 17/05/2019, using the fVasicek model for each one of the three scores. The continuous line is the past prices, the dashed line the result of the simulation, the dots are the observed prices at the predicted days.

## 4.4 Hedging with PCA

The previous parts provide interesting insights about the trend of the Dividend Futures - solving analytically or numerically the SDEs (4.6) or (4.40) gives us the equilibrium price of the Dividend Futures.

However, this information is not precise enough to hedge a portfolio's exposure to the market. We need to be able to lower the risk arising from market exposure in a more efficient way. We previously saw that the three first Principal Components explained 99.8% of the Dividend Futures price variance. So, if we can identify which Score (Figure 10) are correlated to the market, we will be able to cut the portfolio's exposure to these Scores using a Butterfly trade. A Butterfly trade means taking two long positions and one short position (or two short and one long) in three different maturities such that the resulting portfolio is not exposed to two of the Scores. We can take a 1Y-3Y-8Y butterfly to present some result and examples of Butterfly trades in this part. We are going to follow D. Huggins and C. Schaller [12] in this section.

For this part, we are going to assume that we can trade CMFs. It is an approximation but as we mentioned before, it is very easy to go from CMF to Rolled Series. Moreover, all the tools used in this part can be easily applied to Rolled Series instead of CMF (PCA does not require any assumption that is true with CMFs and is false with Rolled Series). And finally, CMFs are a weighted sum of 2 tradable Rolled Series, so 3 positions in CMFs are equivalent to at most 6 positions in tradable Rolled Series.

### Scores Dependency to the market:

The first step in a trade based on PCA is to identify the Scores' dependency to external factors we would be interested to hedge. In our case, it is the Market price move (SX5GT). So we plotted regressions of the three scores log-returns vs SX5GT log-returns. We can see the results in Figure 24.

So, using the results presented in Figure 24, we now have all the parameters to compute the hedge ratios: We are going to use T1, T3 and T8 to cut the dependency to the first and the third Score. Then we will back-test the portfolio obtained using these hedge ratio to see if it is indeed independent from SX5GT moves. Note that the back-test does not focus on obtaining the most profitable portfolio, but the less correlated to the market portfolio.

### Structure of the 1Y-3Y-8Y butterfly:

We call  $e_{j,i}$  the value of the  $j^{\text{th}}$  PC for the  $i^{\text{th}}$  Tenor (Figure 9), and  $n_i$  the notional of the  $i^{\text{th}}$  Tenor for  $i \in [1, 9]$  and  $j \in [1, 3]$ . We are going to adapt D. Huggins and C. Schaller method to hedge with PCA to our model (they developed this framework in Fixed Income). In what follows, the Scores represented in Figure 10 will be called Level (for Score 1), Slope (for Score 2) and Curvature (for Score 3).

For a Tenor  $T_i$ ,  $i \in [1, 9]$ , we have that  $T_i = e_{1,i} \cdot \text{Level} + e_{2,i} \cdot \text{Slope} + e_{3,i} \cdot \text{Curvature}$ . Then,



Figure 24: Three regressions of the Scores vs SX5GT - Score 1 on the left, Score 2 at the centre, Score 3 on the right. We can see that the first Score and third Score are significantly correlated to the market ( $R^2 = 0.80$  and  $R^2 = 0.21$  respectively). They are not correlated in the same direction (positive correlation for the first score, negative correlation for the third). However, the second Score is not correlated to SX5GT ( $R^2 = 0.017$ ). So this is the Score we want an exposure to in our Butterfly Trade.

we want to do a 1Y-3Y-8Y Butterfly hedge against the first and third factor. So we are going to write:

$$\begin{cases} n_1 \cdot T_1 &= n_1(e_{1,1} \cdot \text{Level} + e_{2,1} \cdot \text{Slope} + e_{3,1} \cdot \text{Curvature}) \\ n_3 \cdot T_3 &= n_3(e_{1,3} \cdot \text{Level} + e_{2,3} \cdot \text{Slope} + e_{3,3} \cdot \text{Curvature}) \\ n_8 \cdot T_8 &= n_8(e_{1,8} \cdot \text{Level} + e_{2,8} \cdot \text{Slope} + e_{3,8} \cdot \text{Curvature}) \end{cases} \quad (4.42)$$

Then our portfolio value is:

$$\begin{aligned} \Pi &= \text{Level} \cdot (n_1 e_{1,1} + n_3 e_{1,3} + n_8 e_{1,8}) \\ &+ \text{Slope} \cdot (n_1 e_{2,1} + n_3 e_{2,3} + n_8 e_{2,8}) \\ &+ \text{Curvature} \cdot (n_1 e_{3,1} + n_3 e_{3,3} + n_8 e_{3,8}) \end{aligned} \quad (4.43)$$

Since we want to make our portfolio independent from the first (Level) and third (Curvature) factors, we actually want to solve the following problem:

$$\begin{cases} n_1 e_{1,1} + n_3 e_{1,3} + n_8 e_{1,8} &= 0 \\ n_1 e_{3,1} + n_3 e_{3,3} + n_8 e_{3,8} &= 0 \end{cases} \quad (4.44)$$

or, writing it with matrices:

$$\begin{bmatrix} e_{1,1} & e_{1,8} \\ e_{3,1} & e_{3,8} \end{bmatrix} \cdot \begin{bmatrix} n_1 \\ n_8 \end{bmatrix} = \begin{bmatrix} -n_3 e_{1,3} \\ -n_3 e_{3,3} \end{bmatrix} \quad (4.45)$$

We choose to isolate  $n_3$ , and since we are not interested in the return of the portfolio here but its relation with the market (SX5GT), we can arbitrarily set the short position  $n_3 = -1$  unit. We

can then find  $n_1$  and  $n_8$  solving Equation (4.46).

$$\begin{bmatrix} n_1 \\ n_8 \end{bmatrix} = \begin{bmatrix} e_{1,1} & e_{1,8} \\ e_{3,1} & e_{3,8} \end{bmatrix}^{-1} \cdot \begin{bmatrix} -n_3 \cdot e_{1,3} \\ -n_3 \cdot e_{3,3} \end{bmatrix} \quad (4.46)$$

We note that Equation (4.46) gives us  $n_1$  and  $n_8$  given a notional  $n_3$ . This parameter  $n_3$  is the degree of freedom left to determine our degree of exposure to the factors uncorrelated to the market (they are not necessarily explicitly defined, it depends if we could find them using the same kind of methods than the one we previously performed).

Also, it is very interesting to notice that the hedging ratios do not depend on the value of the three Scores. Indeed, the mean-reverting property of the three Scores can give us indications on the direction of the exposure we should take (is the signal above or under its mean-reverting level?), but does not affect the hedge.

#### Results:

First, we want to check if using Equation (4.46) really produce a portfolio that is uncorrelated to Scores 1 and 3 (Level and Slope) and correlated to Score 2 (Slope). To do so, we computed the PCA at a given date, used Equation (4.46) to compute the Hedge ratios, and constituted a portfolio using these ratios, starting from the first day of calibration to last day of calibration. We can see on Figure 25 and Figure 26 that this portfolio is not significantly exposed to Level and Curvature, but is well correlated to the Slope.

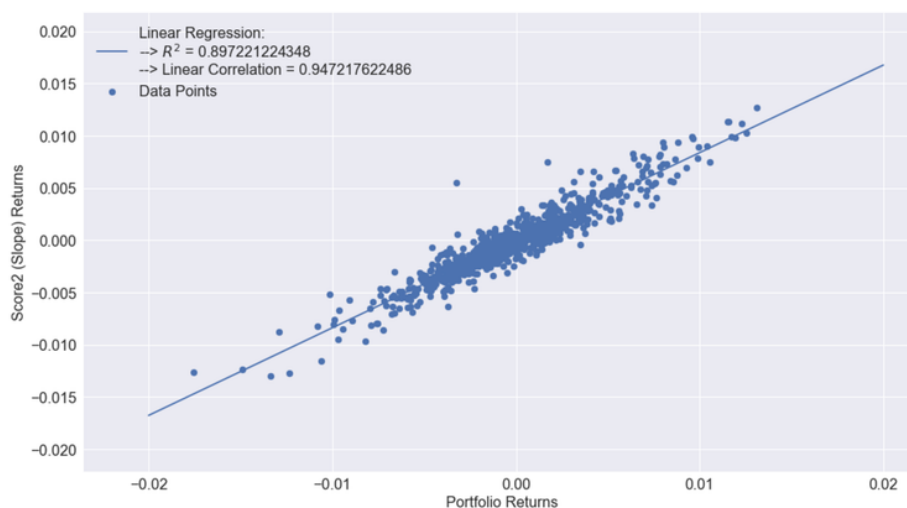


Figure 25: Exposition of a Portfolio created to Hedge the Level and the Curvature to the Second Score (Slope). We can see that there is indeed a high Correlation between the two Time Series (High  $R^2$  and high linear correlation).

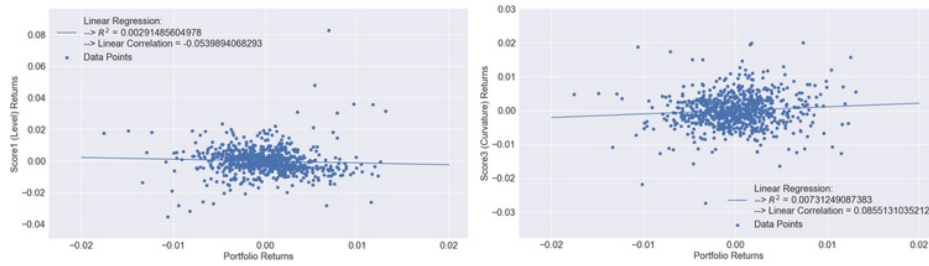


Figure 26: Exposition of a Portfolio created to Hedge the Level and the Curvature to the First and Third Score (Level and Curvature). We can see that there is a low Correlation between the two Time Series (low  $R^2$  and linear correlation close to 0).

We then back-tested a dynamic strategy from beginning 2017 to mid-2019. For each day, we took the past 3 years of data, computed the PCA, then used Equation (4.46) to compute the notional  $n_1$  and  $n_3$ , knowing that we set  $n_3 = -1$ . We then re-balanced the portfolio each day using the ratios computed. Recall that we are not trying to optimise a profit, but minimise a market dependency. We can see in Figure 27 that we produce a portfolio uncorrelated to the market.

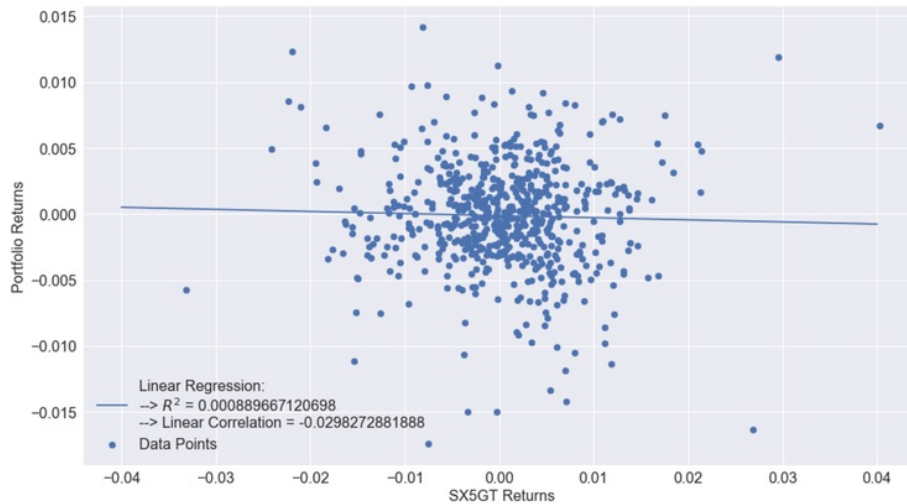


Figure 27: Dynamic strategy to back-test PCA Hedging. Portfolio is re-balanced everyday. We can see that the portfolio is not correlated to the market (low  $R^2$ , low linear correlation in absolute value).

However, even if the portfolio created using a dynamic adjustment is not correlated to the



market, it needs a daily re-balancing. For the Dividend Futures, that might be a problem. Indeed, the volumes traded don't seem to allow important changes in positions without a significant market impact. So, in practice, it would be very unlikely to see someone re-balance its portfolio every day, and if not re-balanced every day this portfolio quickly acquire a significant correlation to the market. Thus a coherent hedge would be a trade-off between market exposure and reasonable re-balancing frequency.

## 5 Beta Hedging

So far, we tried to establish a trading strategy using PCA. Assuming a mean reverting model for the Level, Slope, Curvature components, we found a way to make some predictions about the Term Structure moves. However, this was not helpful to actually trade these contracts, since we ideally want to have an exposure to dividends, but the lowest exposure to the market (SX5E/SX5GT) possible. So we defined hedge ratios using the PCA term structure.

Now we want to focus on other techniques that allows to hedge the Dividend Futures. A common way to hedge derivatives is to compute the Beta of each tenor, and then constitute a Beta-neutral portfolio. We want to compute Betas with respect to the market (SX5GT), so we can compute the total Beta of our portfolio and hedge by buying/selling SX5GT instead of re-balancing everyday our positions in Dividend Futures. This is in practice more interesting for different reasons, one of them being a consideration on market impact. Indeed, since Dividend Futures are not traded in important volume, a regular rebalancing will expose us to an important market impact that will create losses. If we use SX5GT to hedge, since the volumes are more important, the portfolio shouldn't suffer as much from market impact.

Beta is a way to measure the risk arising from exposure to a general market. For example, BT (BT/A LN) or Barclays (BARC LN) usually have a beta computed using the FTSE 100 as the market. In our case, since we are interested in the Dividends paid by the components of the EuroStoxx 50, we are going to take SX5GT (EuroStoxx 50 gross return) to compute the betas. A beta greater than 1 means that the Stock/Derivative we consider is more volatile than the market, a negative Beta means that the Stock/Derivative moves to the opposite of the market's direction (for example, market is up so Stock/Derivative is down).

Beta can have two different formulations. The first one is  $\beta = \frac{\text{cov}(r_D, r_{mkt})}{\text{var}(r_{mkt})}$ , where  $r_D$  are the returns of the Dividend Futures,  $r_{mkt}$  are the returns of the market. The second way to compute  $\beta$  is to use the following linear regression [13]:  $r_D = \alpha + \beta r_{mkt} + \epsilon$ , where  $\epsilon$  are the residuals.

A quick word on the dates for that part: when calibrating the models, we use all the data available at current time to calibrate the model. But we actually a forecast the next business day. For example, we calibrate as of Friday 17<sup>th</sup> of May. The outputs of our models are actually Beta

forecasts for Monday 20<sup>th</sup> of May. That's what is interesting for hedging purposes: trying to have a good estimator of what will be the dispersion (beta) of the stock tomorrow, given the data today.

## 5.1 Historical model

In this part, we are going to use the first definition of Beta:

$$\beta = \frac{\text{COV}(r_D, r_{mkt})}{\text{var}(r_{mkt})}. \quad (5.1)$$

The approach here is simple and intuitive: we define a window size (3 and 6 month), then for each Tenor (remember that there is 10 Tenor for DEDZ) we extract the prices Time Series and the Market Prices Time Series (corresponding to the same time frame) and we compute the log returns for these series. Then, we can define the vector of 10 beta as (1 per Tenor):

$$\beta = \left[ \frac{\text{COV}(r_{D_1}, r_{mkt})}{\text{var}(r_{mkt})} \quad \dots \quad \frac{\text{COV}(r_{D_{10}}, r_{mkt})}{\text{var}(r_{mkt})} \right]. \quad (5.2)$$

We obtain 10 Time Series representing the evolution of the different beta in time, as we can see in Figure 28.



Figure 28: Historical Beta Time Series for Tenor 2 (left) and Tenor 7 (right), using a historical Beta.

We can then plot the term structure at a given date, as in Figure 29.

What is very interesting on the Figure 28 is that we can see that the two signals are very similar, but seem shifted (it is particularly visible in the second Tenor's graph). The explanation here is that, the greater the window, the less statistical importance two points have. When we add one point and drop out one point (we roll the window to the next day), we change two points. If it is 2 points over 126 (6 month) it has less importance than 2 points over 63 points (3 months). Thus, a shock on the market (for example, if for some reason the second Tenor becomes much more volatile) would take more time to appear on the graph if the window is broader. However, a larger window produce a less noisy signal. So calibrating the window is mainly playing with a trade-off noise/reactivity.

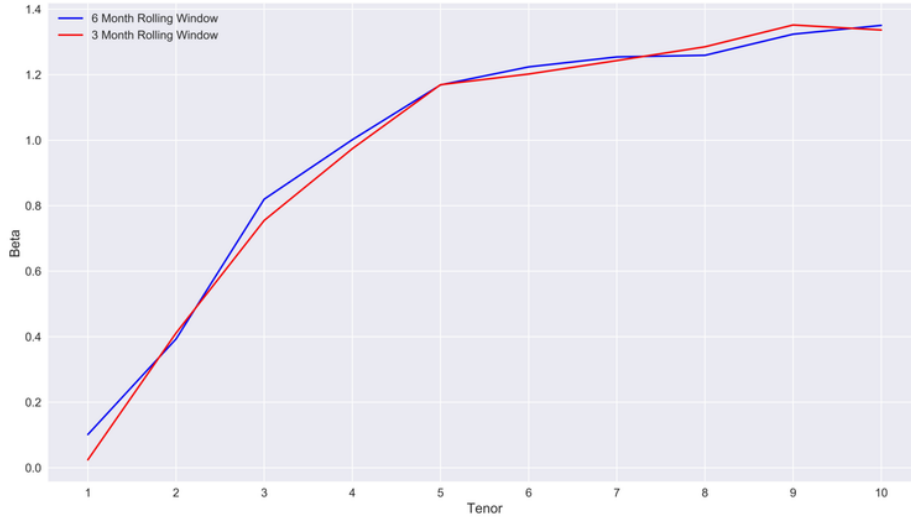


Figure 29: Term Structure computed the 17<sup>th</sup> of May 2019, to predict the historical Betas on the 20<sup>th</sup> of May (1 business day later), using a historical Beta.

## 5.2 Parametric model

Given the shape of the curves in Figure 29, a good idea would be to think of a functional form for Beta, using a set of parameters we would need to calibrate, and Beta having also a time dependency (so we can reproduce at least partially the seasonality of these contracts). We plotted Figure 30 to confirm that Betas has a characteristic shape. In abscissa, we plotted  $T - 1 + \tau$  where  $T$  is the Tenor, and  $\tau$  is the time to maturity. In ordinate we plotted the previously computed Betas (6M window calibration, but the shape is the same for 3M window calibration).

To calibrate these parametric models, we are going to use the second form of beta [13]:

$$r_D = \alpha + \beta r_{\text{mkt}} + \epsilon. \quad (5.3)$$

with  $r_{\text{mkt}}$  the market log returns,  $r_D$  the Dividend Futures log returns and  $\epsilon$  the residuals.

So, the problem we are confronted to is to find a functional form for beta, and then calibrate these parameters to reduce as much as possible the residuals. The mathematical formulation of this problem is - we call  $\theta$  the set of parameter of the function beta and  $\Theta$  their domain of definition, meaning that  $\theta \in \Theta$ :

$$\theta = \underset{\theta}{\operatorname{argmin}} \left[ \sum_{i=1}^{10} \sum_{k \in \mathcal{K}} \|r_{D_i} - \beta(\theta, t_k^*) \vec{d} \cdot \vec{r}\|^2 \right] \quad (5.4)$$

with:  $\vec{d} = [d^0 \ d^1 \ d^2 \ d^3]$  ( $d$  constant) a lag parameter (empirically, we observe that at certain time the Dividend Futures doesn't react to the SX5E/SX5GT index immediately, so we introduced

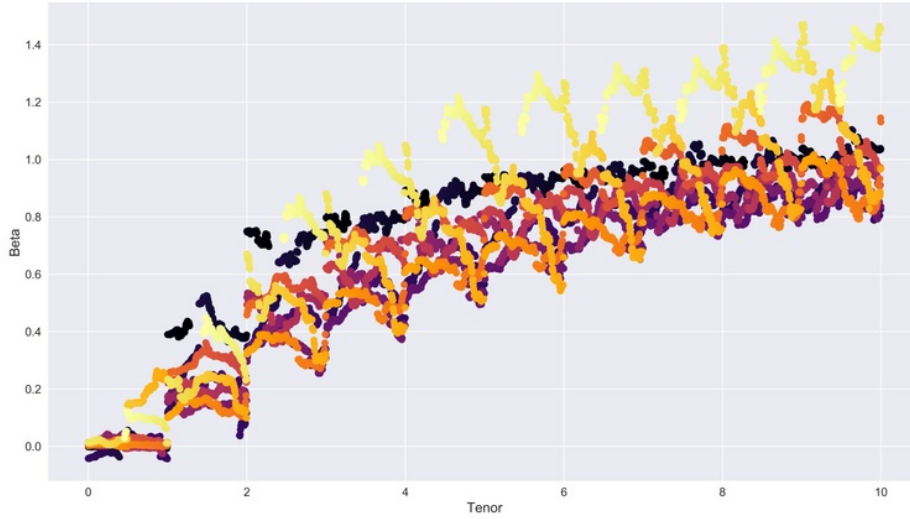


Figure 30: Scatter plot of Betas against Tenor, between 07/09/2011 (dark points) and 09/07/2019 (clear points). We can see that the shape in Figure 29 is actually representative of the structure of Betas across the 10 series.

a lag to the optimization, the movement at time  $t$  can depend on the movements up to time  $t-3$ ),  $\vec{r} = [r_t^{\text{mkt}} \ r_{t-1}^{\text{mkt}} \ r_{t-2}^{\text{mkt}} \ r_{t-3}^{\text{mkt}}]^\dagger$  and  $\mathcal{K}$  is the set of observations in the dataset. Finally,  $t_k^*$  are the 'true' Tenors of the contracts, meaning that to be as precise as possible, we consider that the Tenor price are not fixed, but depend on the time to expiry of the contracts. For example, if we are in April, the Tenor 'seen' by the front series in the optimisation problem is not 1, but  $1 - 3/12 = 0.75$ .

A quick word on parameter  $\alpha$  from Equation (5.3). We assume  $\alpha$  being constant, so its value will modify the value of the minimum in the optimization problem 4.33, but not the argmin, which is the value of interest for us. Thus, we can ignore the effects of  $\alpha$  on the optimization.

A good family function that actually has the shape of the curves in Figure 29 is the Generalized Logistic Functions [14]. Their typical form is:

$$f(x) = A + \frac{K - A}{(C + Qe^{-Bx})^{\frac{1}{\nu}}} \quad \text{for } x \in \mathbb{R}. \quad (5.5)$$

In this form,  $A$  is the lower asymptote,  $K$  the upper asymptote when  $C=1$ ,  $B$  is the growth rate,  $\nu > 0$  affects near which asymptote maximum growth occurs,  $Q$  is related to the value  $f(0)$  and typically we set  $C = 1$ , otherwise it just change the value of the upper asymptote. Thus, we reduce this to five free parameters, and we will see that there is a way to once again reduce the problem to 4 free parameters. The logistic function and the Gompertz function (3 free parameters) are special cases of the General Logistic Function.

In this part, we will solve the optimization problem (5.4) using first 3 months of data for the

calibration, and then 6 months.

### 5.2.1 Gompertz function

The Gompertz function is defined as :

$$\forall x \in \mathbb{R}, G(x) = ae^{be^{cx}} \quad (5.6)$$

Or, since we want to use the time as a parameter function, we want to map the Gompertz to change the domain of definition to  $\mathbb{R}_+^*$ . So we define, for  $x$  in  $\mathbb{R}$  the following transformation:  $x = \log(t)$ ,  $t \in \mathbb{R}_+^*$ .

Thus, the functional form for Beta becomes:

$$\forall t > 0, \beta(\theta, t) = ae^{bt^c} \text{ and } \theta = [a, b, c] \in \mathbb{R}^3. \quad (5.7)$$

The optimization (5.4) is then conducted using the package `scipy.optimize` in python. Figure 31 shows the Time Series of the three parameters. Figure 32 shows the Beta Time Series for Tenors 2 and 7, and Figure 33 shows the beta forecast the 17/05/2019.

### 5.2.2 Generalized Logistic Function

#### With 5 degrees of freedom

As previously explained, we usually choose  $C = 1$  in Equation (5.5). From a practical point of view, 5 free parameter, when considering 10 series, is already a good number. Increasing the number of free parameter makes the calibration more complicated, and we would need more data to reach a coherent calibration. That could mean sacrificing the reactivity of the model to new shocks.

Applying the same mapping than the previous part (because once again, the Generalized Logistic Function is defined on  $\mathbb{R}$  and we want to apply it on  $\mathbb{R}_+^*$ ), we get that:

$$\forall t > 0, \beta(\theta, t) = A + \frac{K - A}{(1 + Qt^{-B})^{\frac{1}{\nu}}} \text{ with } \theta = [A, B, K, Q, \nu] \in \mathbb{R}^4 \times \mathbb{R}_+^*. \quad (5.8)$$

Once again, we use the package `scipy.optimize` to solve Equation (5.4). Beta Time Series are plotted in Figure 34 and the Beta forecast of the 17/05/2019 are plotted in Figure 35.

#### With 4 degrees of freedom

With 5 degree of freedom, we could see that sometimes the first (and sometimes even the second) Tenor had a negative Beta. However, it is a well verified empirical fact that the Dividend Futures are positively correlated with the SX5E/SX5GT. We can actually force the Beta values to

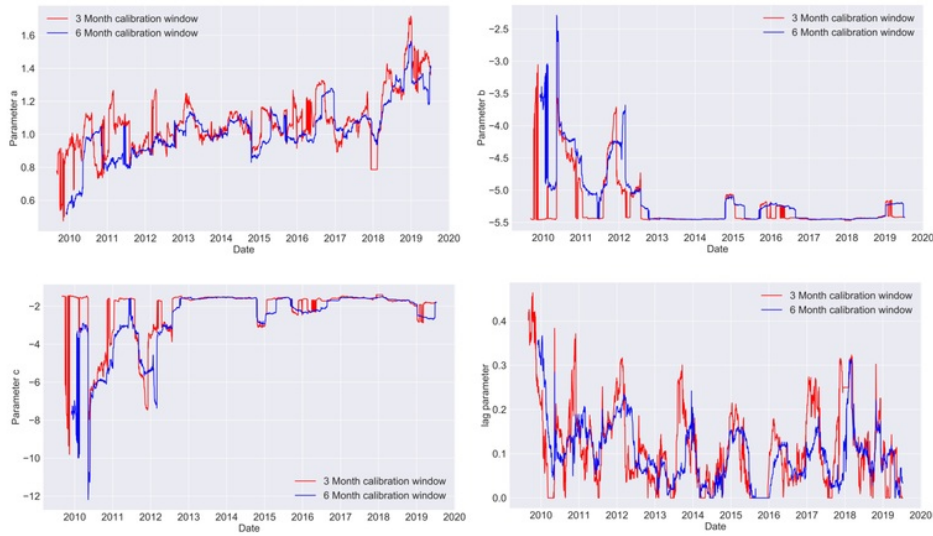


Figure 31: Time series of the parameters  $a$  (upper left),  $b$  (upper right),  $c$  (bottom left) for the Gompertz function, and  $d$  the lag (bottom right). Parameter  $a$  is the amplitude (value of the asymptote, since  $c$  is always negative),  $b$  is an offset along  $x$ -axis and  $c$  is the growth rate. We can see that for the two calibration window,  $d$  has the same shape and amplitude, which means we are indeed capturing the lag with the market and not some residuals in the parameters. Also, we can notice the same shift between 3M and 6M window calibration than in part 5.1, and once again the 6M signal is less noisy than the 3M signal.

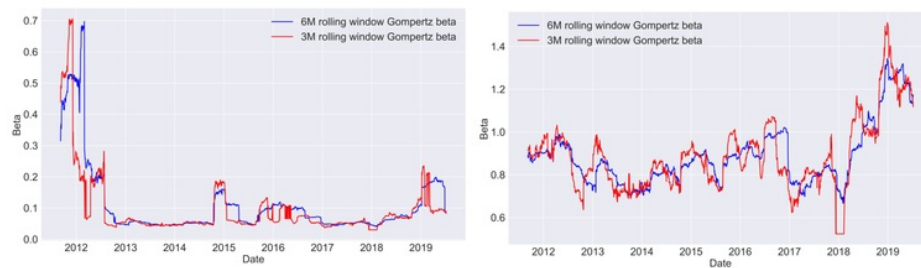


Figure 32: Time series of the Gompertz Beta for Tenor 2 (left) and Tenor 7 (right). Once again, the same comment on the shift/noise stands.

be always positive by sacrificing one degree of freedom. Indeed, we know that in Equation (5.5) the parameter  $A$  controls the value of the lower asymptote (which become, after mapping, the limit

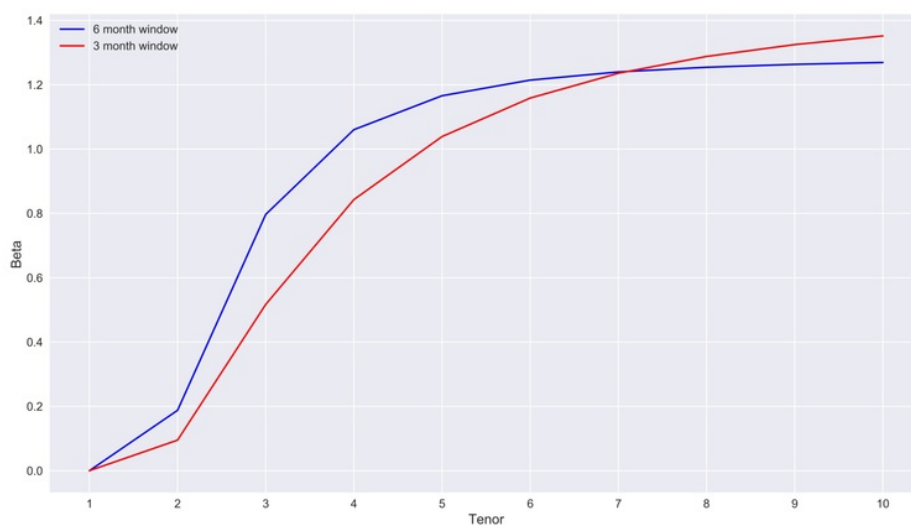


Figure 33: Term Structure computed the 17<sup>th</sup> of May 2019, to predict the Gompertz Betas on the 20<sup>th</sup> of May (1 business day later) with Beta being a Mapped Gompertz Function.



Figure 34: Time series of the GLF (5 degree of freedom) Beta for Tenor 2 (left) and Tenor 7 (right). Once again, the same comment on the shift/noise stands.

for  $t$  going to 0). So we set  $A = 0$ . We then have:

$$\forall t > 0, \beta(\theta, t) = \frac{K}{(1 + Qt - B)^{\frac{1}{\nu}}} \quad \text{with } \theta = [B, K, Q, \nu] \in \mathbb{R}^3 \times \mathbb{R}_+^*. \quad (5.9)$$

Beta Time Series are plotted in Figure 36 and the Beta forecast of the 17/05/2019 are plotted in Figure 37.

### 5.3 Machine Learning Models

Given the strong correlation between the Dividend Futures Term Structure and the underlying market, it is reasonable to try to predict the curve dynamic conditional to market moves. If we

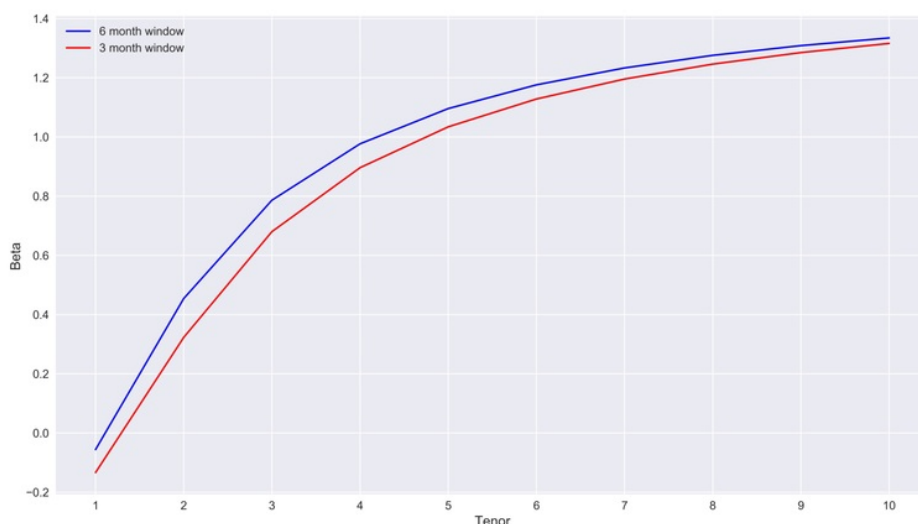


Figure 35: Term Structure computed the 17<sup>th</sup> of May 2019, to predict the Betas on the 20<sup>th</sup> of May (1 business day later) with Beta being a Mapped Generalized Logistic Function (5 degree of freedom).



Figure 36: Time series of the GLF (4 degree of freedom) Beta for Tenor 2 (left) and Tenor 7 (right).

keep in mind that  $r_D = \beta r_{\text{mkt}} + \alpha + \epsilon$ , using the notations introduced in (5.3), we can then get access to Beta by differentiating this relation. Note that, if we use non-linear regressor (which is the goal of Machine Learning), Beta will not be a constant. The value of the market return will have an impact on Beta (Beta will be a function of  $r_{\text{mkt}}$ ). However, we take the derivative in  $r_{\text{mkt}} = 0$  (tangent at the origin in Figures 51 and 52), because if we know what  $r_{\text{mkt}}$  is going to be, it is useless to hedge (we just have to take a long/short position to have interesting returns).

We are going to use three techniques to try to approximate Beta: Artificial Neural Networks (ANN), Support Vector Machine (SVM), and Gradient Boosting (XGB).

We started by exploring the result of a first model, following A. Kondratyev's work [15]. The



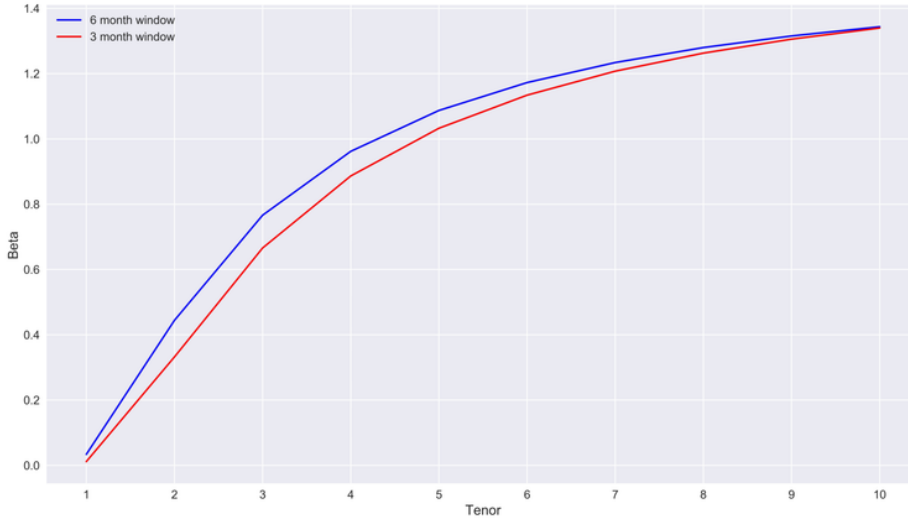


Figure 37: Term Structure computed the 17<sup>th</sup> of May 2019, to predict the Betas on the 20<sup>th</sup> of May (1 business day later) with Beta being a Mapped Generalized Logistic Function (4 degree of freedom).

input of the algorithms is  $\text{Input}(t) = [P_{D_1}(t), P_{D_2}(t), \dots, P_{D_{10}}(t), \Delta\text{SX5GT}]$ , where  $\Delta\text{SX5GT}$  is the expected shock price in the underlying market, and  $P_{D_i}(t)$  for  $i \in [1, 10]$  is the current price of the  $i^{\text{th}}$  Dividend Future Tenor. We will call this model 'Model 1' in the rest of this paper.

Then, since we know that the volatility of the dividend futures changes during the year (we know that the closer to expiry, the less volatile these products are), we had the idea to add the time remaining to reach the next reset date - the next third Friday of December. We call  $\tau$  this quantity (we divide the number of days remaining by the number of days in the period to have  $\tau \in [0, 1]$ ). The input becomes  $[P_{D_1}(t), P_{D_2}(t), \dots, P_{D_{10}}(t), \Delta\text{SX5GT}, \tau]$ . If the idea can seem natural, it is not obvious that the precision of the fit will be improved for this model. Indeed, we might take into account an important variable - particularly for 2<sup>nd</sup> Tenor - for the Term Structure's dynamic, but we also added parameters in our model, and we cannot be sure that the increased complexity of the model will not cost more in term of result than what brings the parameter  $\tau$  to the model. We will call this model 'Model 2' in the rest of this paper.

The output was slightly different according to the model. For ANNs, a multiregression is possible, so the output will be  $[\Delta P_{D_1}(t), \dots, \Delta P_{D_{10}}(t)]$ , where  $\Delta P_{D_i}(t)$  for  $i \in [1, 10]$  represents the expected price move in Tenor in Dividend Futures - i.e.  $\Delta P_{D_i}(t) = P_{D_i}(t + \Delta t) - P_i(t)$ ,  $i \in [1, 10]$ . For SVM and XGB, we are going to use 10 regressor with a one-dimension output. For  $i \in [1, 10]$ , the  $i^{\text{th}}$  regressor will have the output  $\Delta P_{D_i}(t) = P_{D_i}(t + \Delta t) - P_i(t)$ .

To create the data-set needed to train and test our models, we proceeded as follow:

1. We first define a Time Frame to calibrate the models (we determined that a 3 year period, with a 80%-20% train-test decomposition produced the best results).
2. Once we extracted all the data, we create the two input vectors  $[P_{D_1}(t), P_{D_2}(t), \dots, P_{D_{10}}(t), \Delta SX5GT]$  and  $[P_{D_1}(t), P_{D_2}(t), \dots, P_{D_{10}}(t), \Delta SX5GT, \tau]$ . In this step, note that we need the exact price move in SX5GT to feed the algorithm during the training phase, otherwise we would not learn the actual behavior of the Term Structure. So we actually check the market price one step after the current date. For example, to create the input vector the 17/05/2019, we took the Dividend Future prices, then added to the vector the quantity  $SX5GT_{t+\Delta t} - SX5GT_t$  (and  $\tau$  for the second model). To avoid any data leak, we take out the last date of the data-set (the one we want to predict) at that step, so we can train and test our algorithm, and then predict on the last date (remember we actually want the first order derivative  $\frac{dr_D}{dr_{mkt}}$  at  $r_{mkt} = 0$ ).
3. We then split the data-set in a training set and a testing set. We take care to shuffle the data-set first (meaning we don't want to have the first 550 days for training and the next 150 days for testing). This allow the models to learn from all the period we are studying, a risk of not shuffling the data-set would be to miss a regime shift. Assume we are using the last three years to calibrate our algorithms. We split in 2 years - one year. If a shift in regime occurred one year ago, our algorithms will be trained on a data-set that no longer represents today's reality. So we want to shuffle the data-set, so the algorithms are partly trained on both of these regimes.
4. Once we have a training and a testing set, we scale our inputs and outputs using ONLY the training set for the scaler's calibration (we use Standard Scaler from Sklearn, that allows us to invert easily the transformation after the prediction). Note that this action is a part of the pipe, meaning that before each new prediction we will have to scale the input, and un-scale the output using these two standard scaler (one for the input, one for the output).

A short precision concerning the splitting previously mentioned. A good strategy in Machine Learning to tune the models is to actually split the data-set in three subsets - one for training, one to tune the hyper-parameters, one to test. In these data there is a lot of regime shift, so we actually don't want to take a too long calibration window, since we need the latest points to have a high statistical significance. So we took a time frame of approximately 600 days and tuned the algorithms on it using grid searches and random grid searches. Another advantage to keep a "low" number of observation is that we reduce the risk of overfitting.

A grid search is a brute force way to find the optimal hyper-parameters. The objective is to test all the possible combinations. At each step, the program needs to calibrate then test the

regressor. So it becomes quickly too long to compute in practice. A random grid search works in a similar way, but it draws randomly a given number of combinations to test, making it more useful in practise (we can control our computation time). A good way to proceed is to compute first a few times the random grid search to find the good region for the hyper-parameters, and once we have an idea (for example, penalty coefficient is around  $5e - 5$ ) we use a grid search (since we only have a few combinations possible, it is actually possible to compute it) to find a more precise result. Once calibrated, we kept these hyper-parameter values during all the back-testing procedures.

Before presenting the result of the three algorithms, we want to add some elements on the choice of the calibration time frame (we took 3 years, roughly 750 points). As we can see in the learning curves Figure 38, at 600-650 points in the training set, the cross-validation score doesn't significantly increase anymore (except maybe for the SVM regressor). So adding more data after that would most likely present a risk of overfitting. Thus we decide to take 3 year for the calibration, and use 80% of that set for training. This is roughly 600 data points, which is in line with the previous observations. The real objective here is to have estimators that present a low bias. At that point, we can accept a little bit of variance, since there is some technique to reduce it (bagging) that works well in that case and will be presented in the following part.

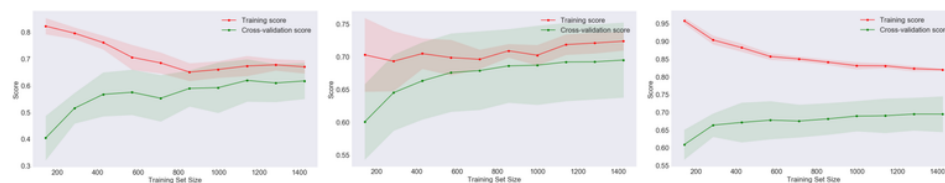


Figure 38: Learning Curves for the three algorithms: Neural Network - Multi-Layer Perceptron (left), SVM (center), XGB (right) [source: scikit-learn [5]].

Now we will present the results of each of the three methods.

### 5.3.1 Neural Network

K. Hornik, M. Stinchcombe, H. White [16] proved that "multi-layer feed Forward networks with  $[\cdot \cdot \cdot]$  a single hidden layer and an appropriately smooth hidden layer activation function are capable of arbitrarily accurate approximation to an arbitrary function and its derivatives". This property is very interesting for us, since we can not only approximate the relation (5.3), meaning  $r_{D_i}$ ,  $i \in [1, 10]$  as a function of  $r_{mkt}$ , but also the derivative, i.e. Beta, which is the quantity we are looking for.

So what is a neural network? To quote A. Kondratyev [15], an Artificial Neural Network (ANN) "is a network of interconnected activation units (neuron). Each activation unit performs three function: summation of thew input [...], non-linear transformation of the aggregated input;

sending the result to the downstream unit”. The most widely used are the MLP (Multi-Layer Perceptron), simply composed by hidden-layer interconnected together, as shown in Figure 39.

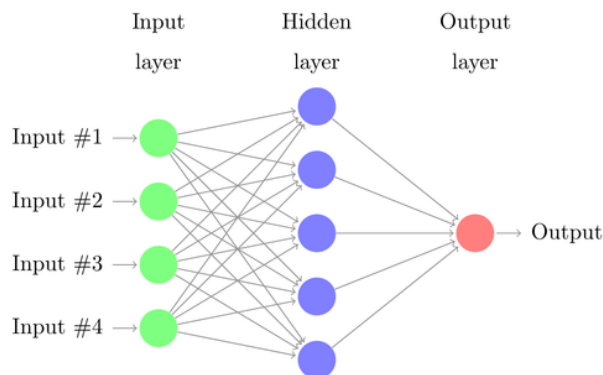


Figure 39: An example of MLP with 4 inputs (or one input in dimension 4) and 1 output (which is not the configuration we use, we have 10 outputs, one per Tenor).

After tuning our MLP, we find the optimal hyper-parameters presented in Table 3.

MLP Parameters	Value
Number of units in the input layer	11 for model 1, 12 for model 2
Number of units in the output layer	10 - 1 for each Tenor
Number of hidden layers	3
Number of units per hidden layer	10
Activation function	Hyperbolic tangent
L2 regularisation	5.8e-5
Solver	lbfgs

Table 3: Table of hyper-parameters for the MLP - using `sklearn.neural.network`.

#### Bagging:

When using a Neural Network on small data-set, we have to limit the number of parameters (hidden layers and unit per hidden layer) to a relatively small number, which can quickly exposes the model to overfitting. Overfitting is characterised by a high variance. There is a method traditionally used in Machine Learning to reduce variance - usually applied to decision trees but can be applied to any classifier. Bagging [17] (for Bootstrap aggregating) consists in calibrating an estimator on many subset of the original data-set. Samples are drawn randomly (note that it can be the entire data-set with a different order) and then one takes the average prediction. In our case, it allows us to significantly reduce the variance of the MLP Regressor.

Finally, we can compute the Beta Time Series for each Tenors (meaning that for each day over a given period, we can calibrate and compute a beta for the next business day. The result using a Neural Network to forecast is displayed in Figure 40.

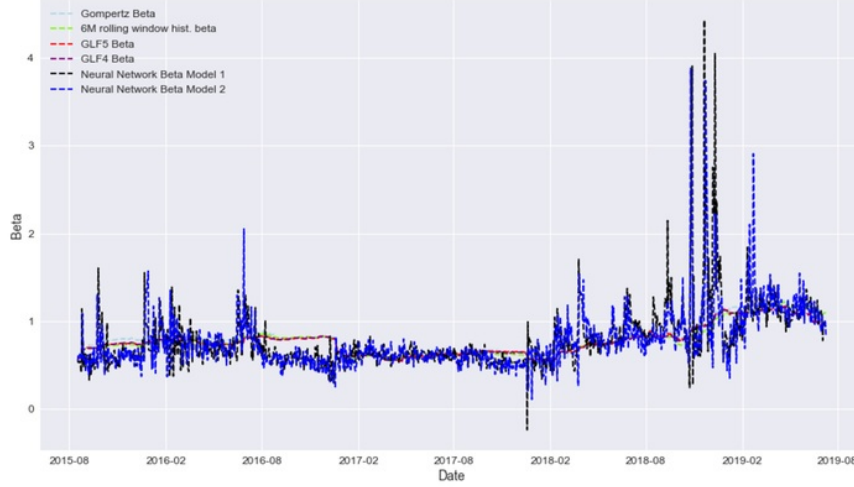


Figure 40: Beta Time series for Tenor 5, obtained using a Neural Network with the architecture in Table 3. The black line is obtained with Model 1, the blue line obtained with Model 2. We also plot the Time series obtained in the two previous parts - Historical Betas and Parametric Betas - to compare the results.

### 5.3.2 SVM

The Support Vector Machine (SVM) is another estimator. According to T. Hastie, R. Tibshirani, J. Friedman [18], the objective of the SVM is to "produces nonlinear boundaries by constructing a linear boundary in a large, transformed version of the feature space". First, we have to know how to create a linear boundary between some data. To calibrate, we have to solve the following problem (we assume here a classification problem with  $N$  data, with  $x_i$ ,  $i = 1, \dots, N$  the feature vectors and  $y_i$  is the label of the  $i^{\text{th}}$  point):

$$\operatorname{argmin}_{\beta, \beta_0} \|\beta\|, \quad \text{under the condition: } y_i(x_i^\dagger \beta + \beta_0) \geq 1 \quad i = 1, \dots, N \quad (5.10)$$

Then, to classify one more instance, we have to do -  $G(x)$  will be the label of instance  $x$ :

$$G(x) = \operatorname{sign}[x^\dagger \beta + \beta_0]. \quad (5.11)$$

The quantity  $M = \frac{1}{\|\beta\|}$ ,  $\beta$  issued from the optimization (5.10), is the optimal Margin. A graphic representation is given in Figure 41.

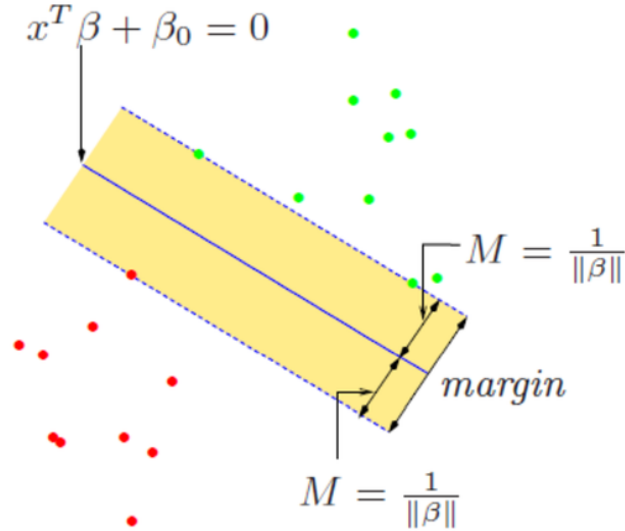


Figure 41: An illustration of a linear SVM (Illustration from T. Hastie, R. Tibshirani, J. Friedman) [18].

Once we know how to separate data using linear hyper-planes, we can map our data to compute non-linear boundaries. Let  $h$  be the function used to map the data. We write, given a data-point  $x_i$ :  $h(x_i) = [h_1(x_i) \dots h_M(x_i)]$  with  $M$  the number of features. We define the following function for any instance of the problem  $x$ :  $\hat{f}(x) = h(x)^\dagger \hat{\beta} + \hat{\beta}_0$ . The optimization problem and the classification function become:

$$\underset{\hat{\beta}, \hat{\beta}_0}{\operatorname{argmin}} \|\beta\| \quad \text{under the condition: } \hat{f}(x_i) \geq 1 \quad i = 1, \dots, N \quad (5.12)$$

and for  $x$  a new data-point:

$$\hat{G}(x) = \operatorname{sign}[\hat{f}(x)]. \quad (5.13)$$

Then, we can show that optimizing Equation (5.12) and Equation (5.13) only involves the function  $h$  through inner product, so we don't need to specify the function  $h$  but only the kernel, i.e. the function  $K$  defined as (we call  $\mathcal{D}$  the data-set):

$$\forall x, x' \in \mathcal{D}, \quad K(x, x') = \langle h(x), h(x') \rangle \quad (5.14)$$

The two most popular choices for  $K$  are:

1. Polynomial:  $\forall x, x' \in \mathcal{D}, \quad K(x, x') = (1 + \langle x, x' \rangle)$ .

2. Radial Basis:  $\forall x, x' \in \mathcal{D}, K(x, x') = \exp(-\gamma \|x - x'\|^2)$ .

We found the best fit for the Radial Basis kernel (rbf kernel in sklearn).

Finally, we can compute the Beta Time Series for each Tenors - as done in the previous part - in Figure 42

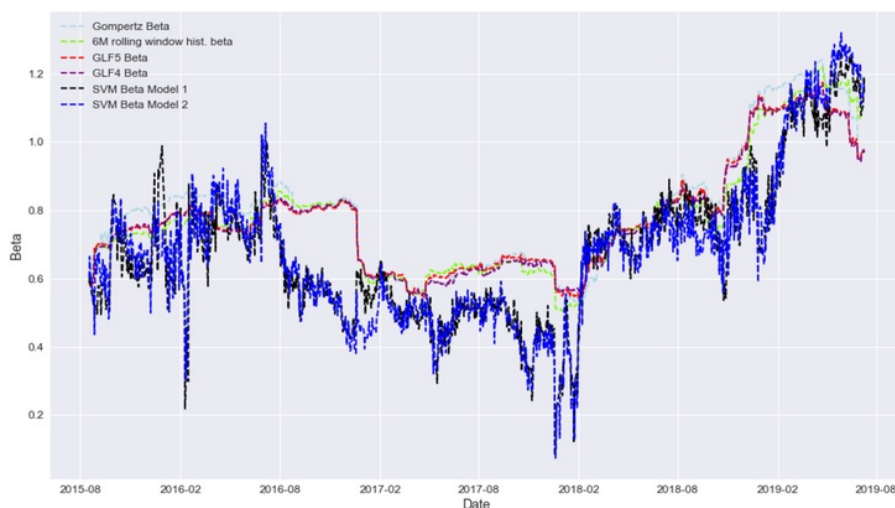


Figure 42: Beta Time series for Tenor 5, obtained using a SVM with a Radial Basis Kernel. The black line is obtained with Model 1, the blue line obtained with Model 2. We also plot the Time series obtained in the two previous parts - Historical Betas and Parametric Betas - to compare the results.

### 5.3.3 XGB

The Gradient Boosting [19] (we use the XGB library, that is why we usually use the letters XGB to refer to the Gradient Boosting Algorithm in this paper) is an ensemble method of weak predictors like decision trees.

A decision tree is a classifier that minimizes at each step an information criteria (typically GINI or entropy) by splitting the data set in two. For our model, we found these two optimal hyper-parameters: 100 trees, and a maximal depth of 3. In Figure 43 we can see an example of a Tree in the XGB algorithm.

Just to check how 'deep' would appear the first argument that was not the expected price move, we launched the calibration for other maximum depths. The result is that the first time happens for a maximum depth of 4. We plot a sub-branch of the tree in Figure 44.

The Beta Time Series for each Tenors is displayed in Figure 45.

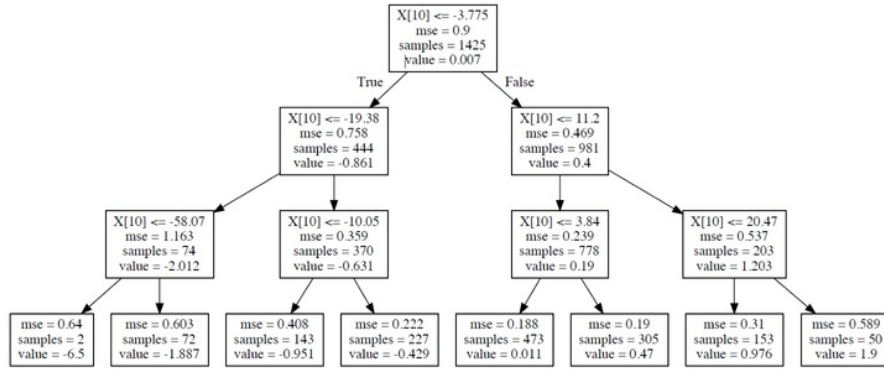


Figure 43: A Tree as our XGB Regressor creates during calibration phase - Forecasting the 5<sup>th</sup> Tenor. We can see that for a depth of 3 (i.e. 3 splits) the only parameter that is actually used by the algorithm is X[10], i.e. the expected market move. So what our XGB Regressor says (or at least one tree in this regressor, but it is coherent with the strong correlation of prices moves with the underlying market) is that the SX5GT price move alone explains the 5<sup>th</sup> Tenor's price moves.

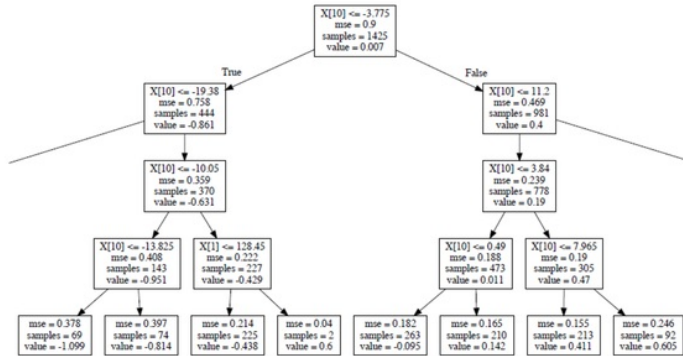


Figure 44: A tree for a maximum depth of 4 (i.e. 4 splits) - still Forecasting the 5<sup>th</sup> Tenor. We can see that one of the fourth split (after a path True-False-False) is done using the price of the second Tenor. We didn't plot the entire tree for space constraints.

### 5.3.4 Stacking

Back-testing on an important number of days, we could realise that each of the previous algorithm had periods where it was the best regressor. So we decided to combine these three models to improve the quality of the fit. To do so, we used a Stacking regression. According to L. Breiman [20], "Stacking regressions is a method for forming linear combinations of different predictors to



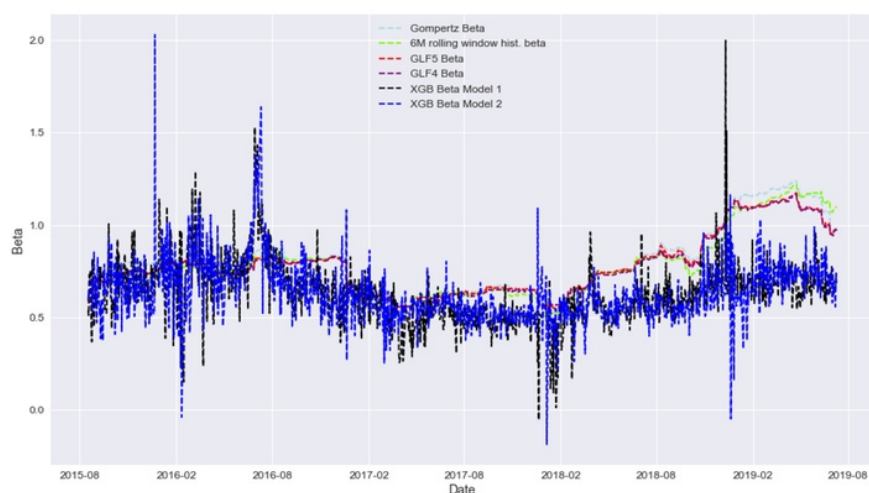


Figure 45: Beta Time series for Tenor 5, obtained using a XGB with a maximum depth of 3 and 100 estimators. The black line is obtained with Model 1, the blue line obtained with Model 2. We also plot the Time series obtained in the two previous parts - Historical Betas and Parametric Betas - to compare the results.

give improved prediction accuracy". He also recommends to use cross validation to determine the optimal parameters to create the Stacking regressor. However, given the restricted data-set we had, we decided to use equal weights for the three models so we would not take the risk to overfit the regressor. Moreover, finding the optimal weight with cross-validation also requires to know the 'true' value of Beta during the back-testing period, and computing a 'true' Beta is not that obvious (Which method? Which definition of Beta? Which time-frame to take to compute the Beta?...). Taking an equally weighted average is equivalent to a Voting regressor, hence the notation VOT in this paper. The result is plotted in Figure 46.

### 5.3.5 Smoothing with EWMA

When looking at Figure 46, we can see that there is a lot of variations in the curves of both models. However, in practice, we might want to have a smoother signal. Indeed, this kind of variations can be interpreted as noise, and hedging using a signal that unstable is in practice never done (we don't want to radically change our positions everyday when we are in high volatility regime).

A way to smooth the curve is to use an Exponentially Weighted Moving Average (EWMA). The goal is to apply exponentially decreasing weight to the past data points and make an average. The coefficients never reach zero, which means at each time all the data points are part of the average.



Figure 46: Beta Time series for Tenor 5, obtained using a Voting Regressor, equally weighted between the three models. The black line is obtained with Model 1, the blue line obtained with Model 2. We also plot the Time series obtained in the two previous parts - Historical Betas and Parametric Betas - to compare the results.

In practical terms, let  $(S_t)_{t \in \mathbb{N}}$  be a time series. We define the EWMA time series - denoted by  $(E_t)_{t \in \mathbb{N}}$  - by:

$$E_t = \begin{cases} S_t & \text{if } t = 0 \\ \alpha S_t + (1 - \alpha)E_{t-1} & \text{if } t > 0 \end{cases} \quad (5.15)$$

the parameter  $\alpha \in [0, 1]$  defines the degree of weighting. An  $\alpha$  close to 1 means that the new data point will have a higher impact on the EWMA series, because the older data points are discounted faster.

Indeed, it is easy to show by induction that the weight of coefficient  $i \in [0, t]$  is  $\alpha(1 - \alpha)^{t-i}$ . In Figure 47, we can see an illustration of the coefficients' decay.

Choosing  $\alpha = 0.1$  smooths significantly the curve, and the variations are still in line with the one observed previously, as we can see in Figure 48.

### 5.3.6 Results of Machine Learning models

Once all our models are calibrated, we can launch predictions and deduce Beta. First, in Figure 49, we plotted the predictions of the 'Model 1', and in Figure 50 we plotted the predictions of the 'Model 2', assuming a perfect guess of the SX5GT - it is obviously not realistic, but it allows us to have a good idea of the regression fit.

Once we have a good calibration, we use relation (5.3) to compute the Betas: we compute a

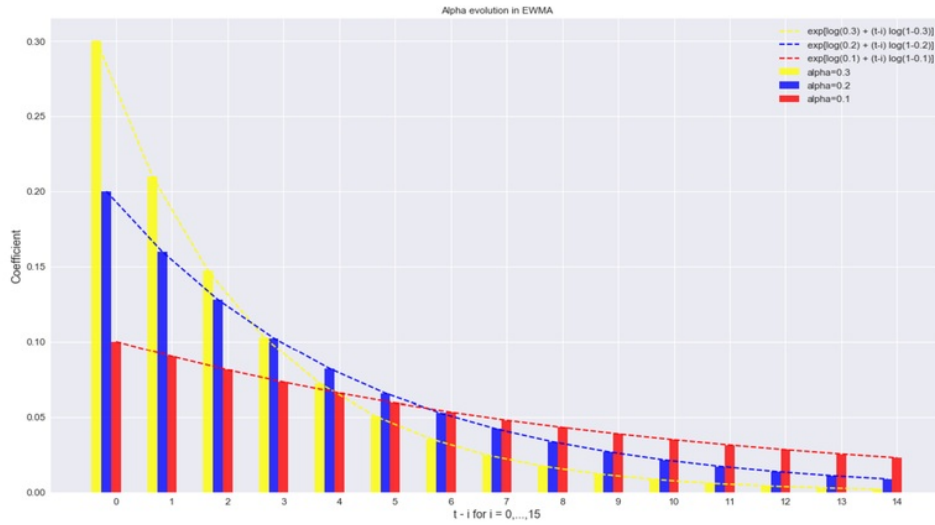


Figure 47: Illustration of the coefficients' exponential decay in EWMA.



Figure 48: Beta Time series for Tenor 5, obtained using a Voting Regressor and applying EWMA - Exponentially Weighted Moving Average - with  $\alpha = 0.1$ . The black line is obtained with Model 1, the blue line obtained with Model 2. We also plot the Time series obtained in the two previous parts - Historical Betas and Parametric Betas - to compare the results.

prediction for different values of  $r_{mkt}$  around 0%, and then take the derivative that is going to give us Beta. We do that for each of the 10 Tenors. We plot in Figure 51 and Figure 52 the curves



Figure 49: The predicted Dividend Future price moves using Model 1, assuming a perfect guess of the SX5GT move. On the left, we calibrate as of the 17/01/2019 and we forecast the 18/01/2019, on the right we calibrate as of the 17/05/2019 and we forecast the 20/05/2019 (next business day, 17/05/2019 is a Friday). The blue line represents the price at calibration time. The black line represents the prices the next business day.



Figure 50: The predicted Dividend Future price moves using Model 2, assuming a perfect guess of the SX5GT move. On the left, we calibrate as of the 17/01/2019 and we forecast the 18/01/2019, on the right we calibrate as of the 17/05/2019 and we forecast the 20/05/2019 (next business day, 17/05/2019 is a Friday). The blue line represents the price at calibration time. The black line represents the prices the next business day.

representing the predicted Dividend Futures price moves as a function of the market (SX5GT) price move.

At this point it is still needed to tackle one of the most important - and difficult! - issues in Machine Learning: estimating the quality of the fit. In our problem, we chose a walk-forward validation to assess the quality of our regression. This technique is more a back-testing technique than a pure cross-validation technique, but it is very close to the Time-Series cross validation - the calibration period is always the same for a Rolling-window analysis. It corresponds to a complete re-calibration of the model as a new data point becomes available.

The procedure works as follows:

1. Choose a start date and a calibration time frame.

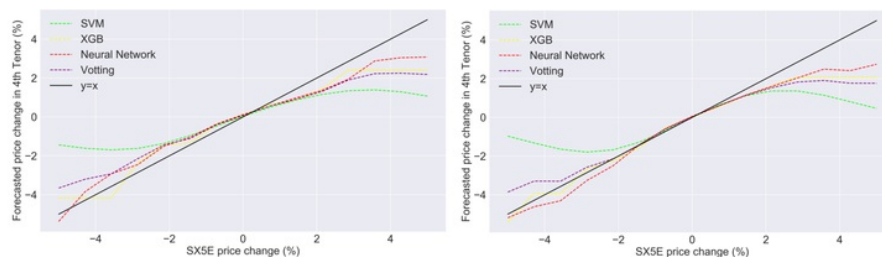


Figure 51: Predicted Dividend Futures price moves as a function of the market (SX5GT) price move using Model 1. On the left, calibration as of the 17/01/2019, on the right calibration as of the 17/05/2019.

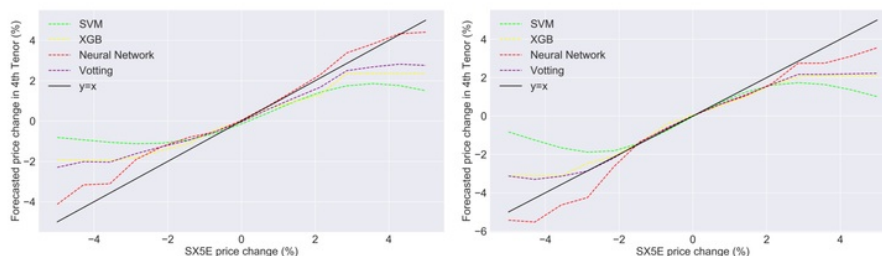


Figure 52: Predicted Dividend Futures price moves as a function of the market (SX5GT) price move using Model 2. On the left, calibration as of the 17/01/2019, on the right calibration as of the 17/05/2019.

2. Calibrate the model.
3. Make one prediction (in our models, it was a one-day-ahead prediction).
4. Record the true outcome of this prediction.
5. Slide the window from one point (in our case, we take out the first day, and add one day in the calibration period - which correspond to the previous true outcome).
6. Repeat until using all the data available or reaching a predefined end date.

For our results, we rolled the window from the 03/01/2019 to 26/06/2019 - 120 days. We then used 3 different metrics to assess the quality of the fit: the  $R^2$  score in Figure 53, the MSE in Figure 54 and the Explained Variance in Figure 55. Except for Tenor 1 - which is a particular Tenor because of its very low volatility - the regressions have good scores. However, we cannot select one model that would better than all the others for every Tenors using only the scores of the regressors. This will be the objective of part 6 later. The Voting regressor performs well on all the

Tenors (always among the most efficient models in all the 3 metrics). The increased complexity brought by the Model 2 does not bring huge improvements (on certain Tenors it actually worsen the fit). We can also see that Neural Networks (for Tenor 2 and 3 in particular) and SVMs models have a good fit on their own - SVM is frequently used when manipulating small data-sets. For the Neural Network, we can see in Figure 40 that the Beta computed for the fifth Tenor spikes at the beginning of January - until 4 approximately - which is not a behaviour one would expect (for the fifth Tenor, one would expect a beta between 0.8 and 1.2). This is one of the advantages of the Voting regressor (and the EWMA), it allows to lower the importance of that kind of outliers.

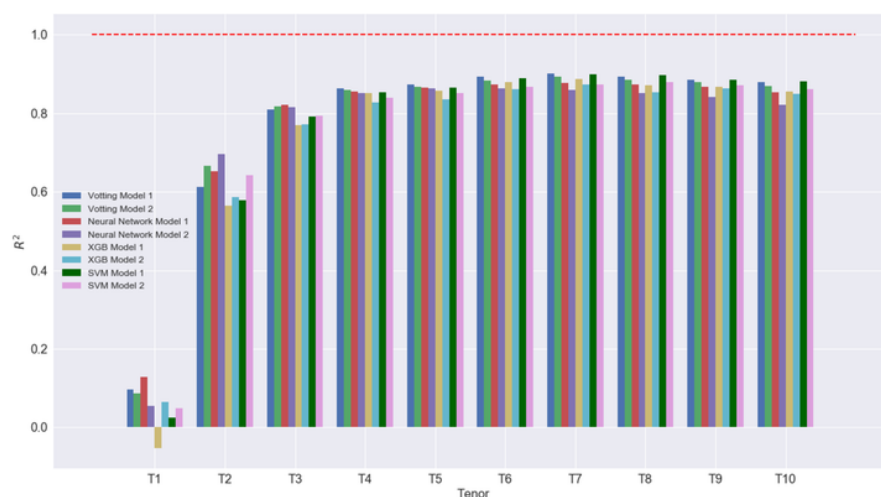


Figure 53: Coefficient  $R^2$  for all the models and Tenors. The closer to 1, the better is the regression. We can see that for Tenor 4 and higher, the Voting regressors perform better (particularly Model 1), even if SVM seems to follow very closely.

These results confirm that we cannot easily select one Beta as better than the others yet. The objective of the next part is to try to select the best beta for each of the methods we presented.

## 6 Back-Testing Beta Hedges

The objective here is to back-test the beta-hedge strategies we created in part 5 to select the best Beta for each Tenor. To do so, we want to construct a portfolio that will be uncorrelated to the SX5GT index. Recall that we are not trying to come up with an optimal portfolio in terms of returns, but we want to construct a portfolio uncorrelated to the market. In this part, we are going to use data from 21/08/2015 to 10/07/2019 to perform the back-testing.

Our strategy is the following. We will consider 10 different portfolios (one per Tenor, as all the

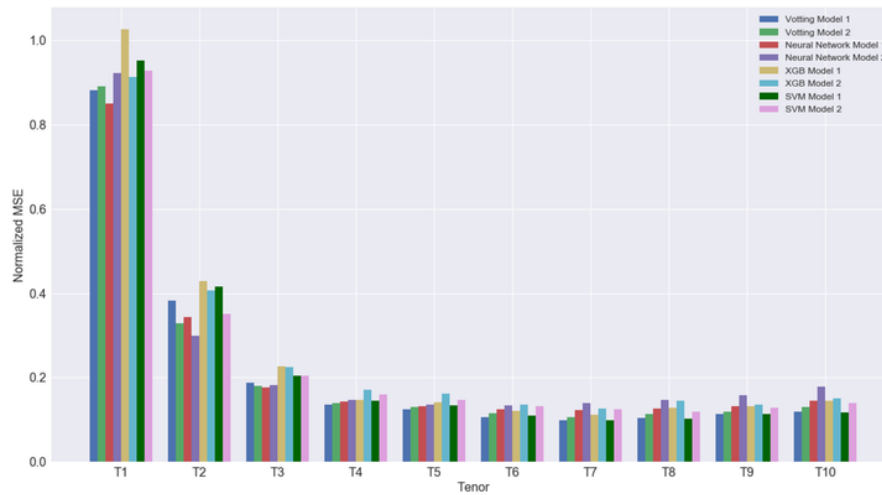


Figure 54: MSE for all the models and Tenors. We normalised by the average true value, so we can compare easily the results. The goal here is to get as close to 0 as possible. Once again, results are better for Tenors higher than 3.

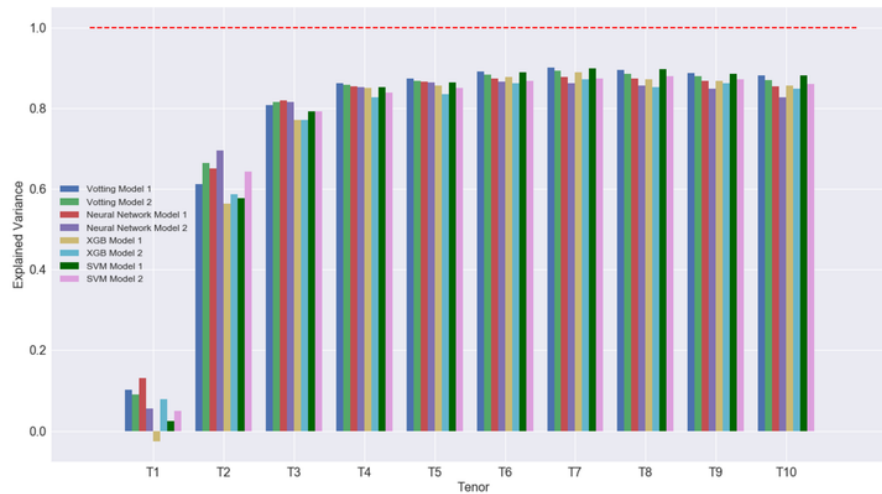


Figure 55: Explained Variance for all the models and Tenors. The closer to 1 (100% of variance explained) the better. This metric gives results very similar to  $R^2$ .

Tenors don't have the same volatility, we might think that they don't all have the same optimal way to compute beta). For each portfolio, we will assume that we have an exposure of 1 contract in Dividend Futures. Then, our objective is to determine each days how much SX5GT to buy to

uncorrelate the portfolio from the market (careful, SX5GT is an index, it is not actually traded on an exchange, but we will assume that we can replicate it perfectly at no cost).

First, we have to compute the hedging ratio as a function of  $P_t$  the price of a Tenor T at time t,  $SX5GT_t$  the price of SX5GT at time t, and  $\beta$ , for each of the previous ways to compute beta we already exposed. We start by providing a list of all the betas we used during the back-test - to the beta already presented, we added two more betas, they are described in the following list:

- Historical beta, using Equation (5.1), with a calibration time frame of 3 months,
- Historical beta, using Equation (5.1), with a calibration time frame of 6 months,
- Gompertz Beta, as in Equation (5.6), with a calibration time frame of 3 months,
- Gompertz Beta, as in Equation (5.6), with a calibration time frame of 6 months,
- GLF Beta, with 4 degree of freedom, as in Equation (5.9), with a calibration time frame of 3 months,
- GLF Beta, with 4 degree of freedom, as in Equation (5.9), with a calibration time frame of 6 months,
- GLF Beta, with 5 degree of freedom, as in Equation (5.8), with a calibration time frame of 3 months,
- GLF Beta, with 5 degree of freedom, as in Equation (5.8), with a calibration time frame of 6 months,
- SVM Beta, using what we previously called 'Model 1' in part (5.3),
- SVM Beta, using what we previously called 'Model 2' in part (5.3),
- XGB Beta, using 'Model 1',
- XGB Beta, using 'Model 2',
- Neural Network Beta, using 'Model 1',
- Neural Network Beta, using 'Model 2',
- Voting Regressor Beta, using 'Model 1',
- Voting Regressor Beta, using 'Model 2',
- Voting Regressor Beta and EWMA, using 'Model 1',
- Voting Regressor Beta and EWMA, using 'Model 2',



- An average Beta composed of the two historical beta, the parametric betas, and Voting Regressor Beta and EWMA, using 'Model 1' and 'Model 2'. We called it 'Average Beta Smooth', because we averaged the 'smooth' beta functions we had,
- An average Beta of all the models.

Now, let's focus on the hedging ratio. Assume we are at time  $t$ . We can compute  $\beta_t$  using the previous models, and we have access to  $P_t$  and  $SX5GT_t$ , the price of a Tenor  $i$  and the SX5GT price respectively at time  $t$ . So we can write our portfolio value  $\Pi$  at time  $t$  - we call  $x$  the hedging ratio (the quantity of SX5GT we want to buy):

$$\Pi_t = P_t - x \cdot SX5GT_t \quad (6.1)$$

and

$$\Pi_{t+1} = P_{t+1} - x \cdot SX5GT_{t+1}. \quad (6.2)$$

So, using Equation (6.1) - Equation (6.2), we have:

$$\Delta\Pi = \Delta P - x \cdot \Delta SX5GT \quad (6.3)$$

Or, remember that Equation (5.3) gave us  $r_t^{T_i} = \alpha + \beta \cdot r_t^{SX5GT} + \epsilon$ , for a Tenor  $T_i$  at time  $t$ . Thus we can write:

$$\begin{aligned} \Delta SX5GT &= r_{t+1}^{SX5GT} \cdot SX5GT_t \\ \Delta P &= r_{t+1}^{T_i} \cdot P_t \\ &= (\alpha + \beta \cdot r_{t+1}^{SX5GT} + \epsilon) \cdot P_t \end{aligned} \quad (6.4)$$

Finally, using Equation (6.4) in Equation (6.3), we can write  $\Delta\Pi$  as:

$$\Delta\Pi = (\beta P_t - x \cdot SX5GT_t) \cdot r_{t+1}^{SX5GT} + P_t \cdot (\alpha + \epsilon). \quad (6.5)$$

Since we want  $\Delta\Pi$  to be uncorrelated to the market, we want  $(\beta P_t - x \cdot SX5GT_t) = 0$  in Equation (6.5), which gives us the hedge ratio:

$$x = \beta \cdot \frac{P_t}{SX5GT_t}. \quad (6.6)$$

So, for each beta model, and each Tenor, we computed at time  $t$  the hedge ratio as in Equation (6.6) (we can find in Figure 56 the evolution of Hedge Ratio for some of the models). Then, at time  $t + 1$ , we computed the value of our portfolio.

The objective for each Tenor was to find the beta that would produce the hedge leading to a portfolio with the lowest correlation to the market possible. We plotted one example of portfolio returns against the market returns in Figure 57 (period back-tested: 21/08/2015 to 10/07/2019), using Gompertz Beta, with a calibration window of 6 months for the sixth Tenor.

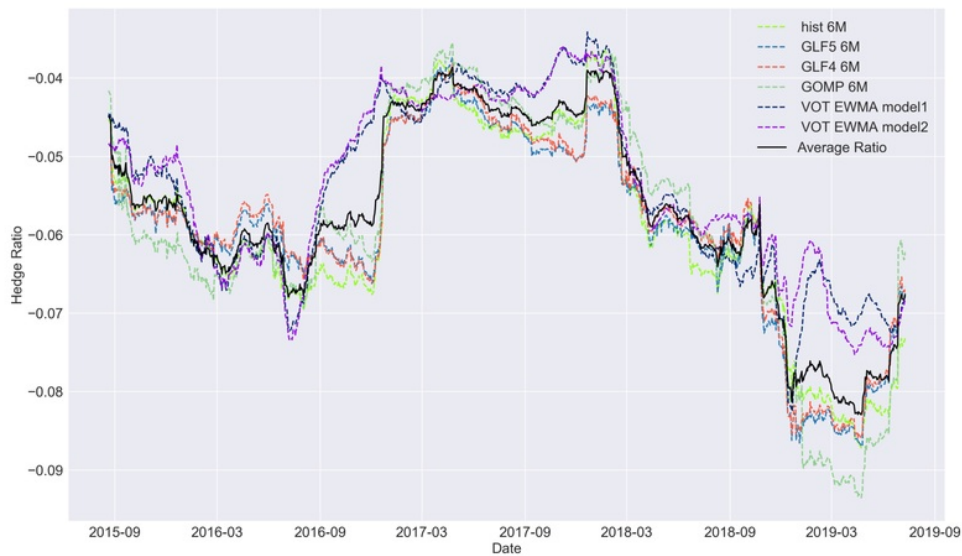


Figure 56: Hedge Ratio for Tenor 4 presented as Time Series for some Models (we selected models with a low level of noise).

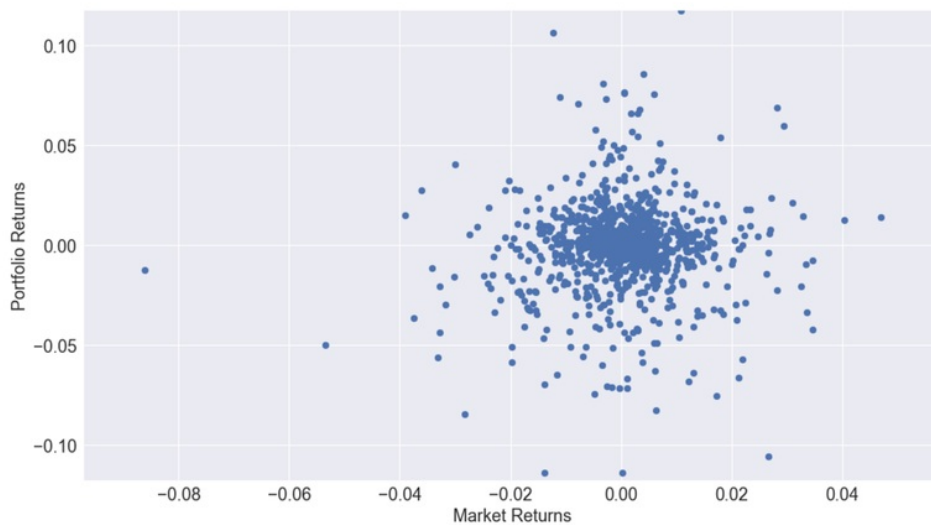


Figure 57: Scatter plot of Portfolio Returns (6<sup>th</sup> Tenor) vs Market Returns. The Hedge ratio is computed using Gompertz Beta, with a calibration window of 6 months. We have  $R^2 = 1.33e - 6$  and a Correlation Coefficient of  $1.15e-3$ . We can assume that the portfolio and the Market are uncorrelated.

When examining the results, we have one important observation: since we do not put any constraints on our portfolio, the portfolio value might sometimes change drastically from one day to another (value multiplied/divided by 2 or 3 overnight). Or it is highly unlikely for the portfolio to be exposed to that kind of variations once some constraints are added to match the profit target (which is not a parameter here). So, in a first step, we decided to disregard the models whose average return and standard deviation (non-annualised, but the two values are proportional so it leads to the same results) were above a given threshold. We want a low value in (absolute) average returns (we set 5%), because high returns in the portfolio are most likely explained by the cash injected in the portfolio to respect the hedge ratio. Also, as previously explained, we want a 'reasonable' value for the standard deviation (we set a standard deviation of 0.55). The previous values were defined experimentally: we chose values that narrow sufficiently the number of models possible for each Tenor, but we kept them high enough so that each Tenor has at least one model eligible (high tenors having a higher volatility and a higher beta, if the constraint on the standard deviation is too strong, no models qualify for these Tenors).

Then, we had to fix a measure to characterise the uncorrelation between the portfolio and the Market. We first used  $R^2$  (in Table 4), but we knew that linear regressions can be sensitive to outliers. So we used the linear correlation between the two returns Time series (in Table 5). Both results lead to the same conclusions.

Tenor	Model	Value of $R^2$
1	GLF with 5 deg. of freedom - 3M calibration	2.44e-6
2	GLF with 5 deg. of freedom - 6M calibration	1.49e-5
3	Historical - 3M calibration	5.84e-9
4	Gompertz - 6M calibration	3.79e-5
5	Historical - 6M calibration	4.35e-4
6	Gompertz - 6M calibration	1.64e-5
7	Voting Regressor + EWMA - Model 2	1.31e-3
8	Gompertz - 6M calibration	5.25e-4
9	Voting Regressor + EWMA - Model 2	4.60e-4
10	Gompertz - 6M calibration	9.42e-5

Table 4: Optimal models using  $R^2$  to measure correlation between Market and Portfolio.

During the Back-test, we could notice that there is usually several models leading to a sufficiently low  $R^2$  and/or Correlation Coefficient. Moreover, the results depends on the metric used and the constraints on the portfolio. So it will be up to the trader to choose the Beta that correspond best to his view on the market among a selection of 4/5 models that actually produce an uncorrelated portfolio. All the models are not interesting to use for every Tenor. Indeed, when

Tenor	Model	Correlation Coef.
1	GLF with 5 deg. of freedom - 3M calibration	1.56e-3
2	GLF with 5 deg. of freedom - 6M calibration	3.86e-3
3	Historical - 3M calibration	-7.64e-5
4	Gompertz - 6M calibration	6.16e-3
5	Historical - 6M calibration	2.09e-2
6	Gompertz - 6M calibration	4.05e-3
7	Voting Regressor + EWMA - Model 2	-3.62e-2
8	Gompertz - 6M calibration	2.29e-2
9	Voting Regressor + EWMA - Model 2	2.15e-2
10	Gompertz - 6M calibration	9.70e-3

Table 5: Optimal models using Correlation Coefficient to measure correlation between Market and Portfolio.

searching for the worst performers for each Tenor (exact same procedure, but instead of taking the model that minimizes the metric, we take the model that maximizes it), we could see that some models can produce a significant correlation to SX5GT for some Tenors.

One final remark on this back-testing: for the first Tenor, no model performed in a very convincing way (it is more obvious when we actually plot the regression SX5GT vs Portfolio Returns). As we already said previously, the 'strange' volatility properties of Tenor 1 make it easier to trust a macro or a corporate model to estimate the dividend paid by a company in the current year rather than trying to predict the prices' variations as a function of SX5GT variations.

## 7 Regime Shift Detection

In the previous parts, we exposed different ways to hedge the Dividend Futures - and how to select the best hedge. However, other information can be useful when trading these contracts. The objective of this part is to search some interesting facts on the Term Structure dynamic. We are going to create a tool that is able to spot changes of regimes in the curve using Unsupervised Machine learning techniques. The idea is that if we can spot different regimes (and the transition period), we will have one more input to choose the optimal hedging strategy.

As we could see in Figure 7, the Dividend Futures Term Series went through a few different 'regimes' in the last ten years. For example, we can spot the pre-2010 period, with high volatility and low spread between the curves, which is clearly different from today's Term Structure's behaviour, with a low volatility, a stable - and higher - spread.

However, how can we split the data in different regimes? And, more importantly, how can we

---

classify one new point once the available data are classified? To answer these questions, we are going to use Unsupervised Machine Learning techniques. More precisely, we are going to use three tools: two Clustering methods, DB-scan and K-means, and one classification method (which is not an Unsupervised Machine Learning method), the K-Nearest-Neighbours Classifier.

### **K-means Algorithm**

The K-means Clustering [18] is an unsupervised method (meaning no labels given to the optimisation algorithm) used to find clusters in a set of data. To start the algorithm, we have to choose a number of Clusters (the K in K-means...). This can be a tricky part if there is no easy way to visualise the data. However, in our problem it is easy to check if we are happy with a given Clustering by having a look at the curves. We can also voluntarily choose a greater number of clusters than different regimes expected, and then merge the closest Clusters. This is actually a standard technique when confronted to non-convex data sets. The K-means Algorithm works this way:

1. Choose K Clusters.
2. Initialise (usually randomly) K centroids.
3. Assign to each data point the label corresponding to the closest centroid.
4. Update the centroids. The new value for each centroid is the average of all the data points in this Cluster.
5. Repeat the two previous steps until convergence.

Once we have all the Clusters, we can decide to merge some of them if we want to.

### **DB-SCAN**

DB-SCAN [21] (for Density Based Spatial Clustering of Applications with Noise) is another way to perform the first task, which is attributing labels to the past data. One interesting feature of this algorithm is that it allows for outliers in the dataset. The intuitive idea behind DB-SCAN is that if a point has a lot of other points within a distance  $\epsilon$ , then they belong to the same cluster (high density of point means we found a cluster). So the algorithm needs two parameters: a distance  $\epsilon$  and a number  $N_{\min}$ . To describe the algorithm, we first need the following definitions:

- Core point: A point is a core point if at least  $N_{\min}$  points are within a distance  $\epsilon$ .
- Directly reachable: A point q is directly reachable from a point p if q is within distance  $\epsilon$  of p.

- Reachable: A point  $q$  is reachable from a point  $p$  if there exist a path  $p, p_1, \dots, q$  where  $p_{i+1}$  is directly reachable from  $p_i$  and all points are core points, except  $q$  (it can be or not a core point).
- Outliers: All points not reachable from any other point.
- Density-connected:  $p$  and  $q$  are density-connected if there is a point  $o$  such that both  $p$  and  $q$  are reachable from  $o$ .

Then, the clusters are creating such that each point follows two rules:

- If  $p$  and  $q$  are two points, and  $p \in C$  with  $C$  a cluster, and  $q$  is density-reachable from  $p$ , then  $q \in C$ .
- If  $p$  and  $q$  are two points of a same cluster, then  $p$  is density-connected to  $q$ .

### **K-Nearest-Neighbours**

Once our dataset labelled, we want to be able to give a label to all new point (new day of data in our problem). To do so, we use a K-Nearest-Neighbour Classifier [18]. The idea is easy: for each new point, we define a number  $K$  of neighbours points to consider. Then, we assign the label of the most represented label among these  $K$  neighbours. If there is a draw, we assign randomly the label among the draws.

### **Methodology**

Before starting to tune the three algorithms, we need to select the features and define the pre-processing to apply to our data. After a few tries, we decided to select the following features, which (understandably) are important in the Term Structure dynamic:

- The 10 prices for the 10 tenors,
- The market price (SX5GT),
- $\tau$  the time to expiry modulo 1,
- The V2X index, which gives the implied volatility for the stocks members of the SX5GT,
- The Tenors' annualised volatility - computed using a 1 month rolling window,
- The 1 month trend - computed with a simple linear regression over 1 rolling month,
- The spread between first and second tenors - interesting because the curves sometimes crosses each other,
- The spread between first and tenth tenor,

- The spread between second and tenth tenor.

Once we had our vectors for each day, we could start the pre-processing before the tuning of the algorithms.

With vector of such dimension - 36 dimensions, we know that the algorithms can loose in performance. Moreover, we expect a lot of information to be redundant (notably the 11 dimensions linked to volatility). So we decided to perform a PCA, with three dimensions. Three dimensions explained more than 99% of the model's variations here. Finally, since these algorithms are sensible to the number's scale, we scaled the vectors resulting from PCA.

Then we started tuning our parameters. We are dealing with a subject where data visualisation is relatively simple (we can visually check if the clustering match some financial explanations). So, we used a brute-force approach.

For DB-SCAN, we created a grid of  $\epsilon$  and  $N_{\min}$ , then we selected the combinations that lead to a 'reasonable' number of clusters (we checked for 5 to 10 clusters), and we selected the parameters that led to the clustering with the best financial explanation. The result for DB-SCAN are represented in Figure 58.

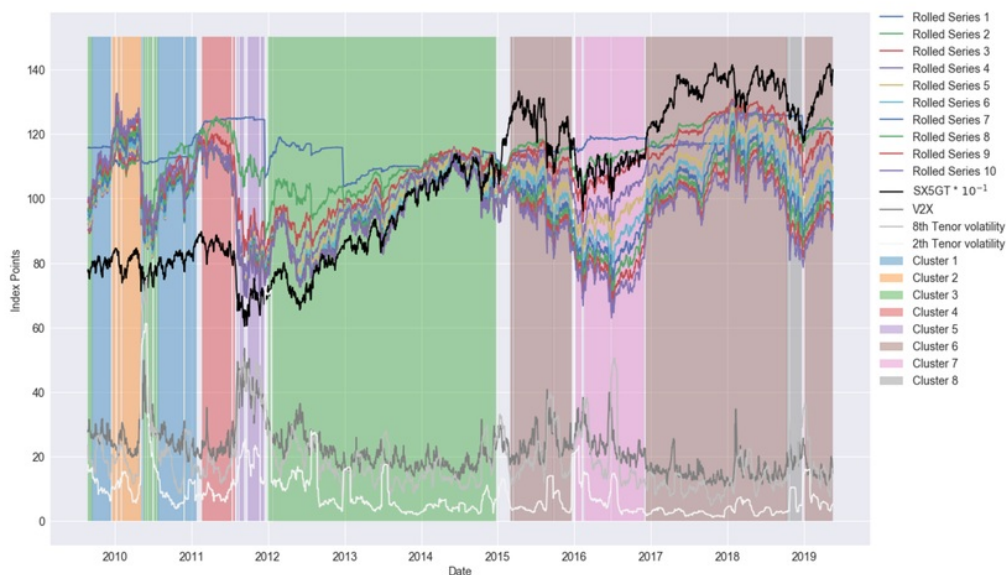


Figure 58: DB-SCAN clustering using the complete pre-processing. We can see, among the most remarkable features, that the end of 2018 has its own cluster. Moreover, we can see that Cluster 3 and Cluster 6 have similar levels of volatility and a bullish trend, but in Cluster 3 the spreads are decreasing when in Cluster 6 they are more or less constant.

For K-means, we progressively increased the number of clusters, until we were satisfied with

the conclusions we could draw from our model - we found an optimal solution of 6 clusters. Then, we realised that initialising the algorithm with 7 clusters and merging the two closest clusters lead to interesting results too. The results of the two approaches are represented in Figure 60 and Figure 59.

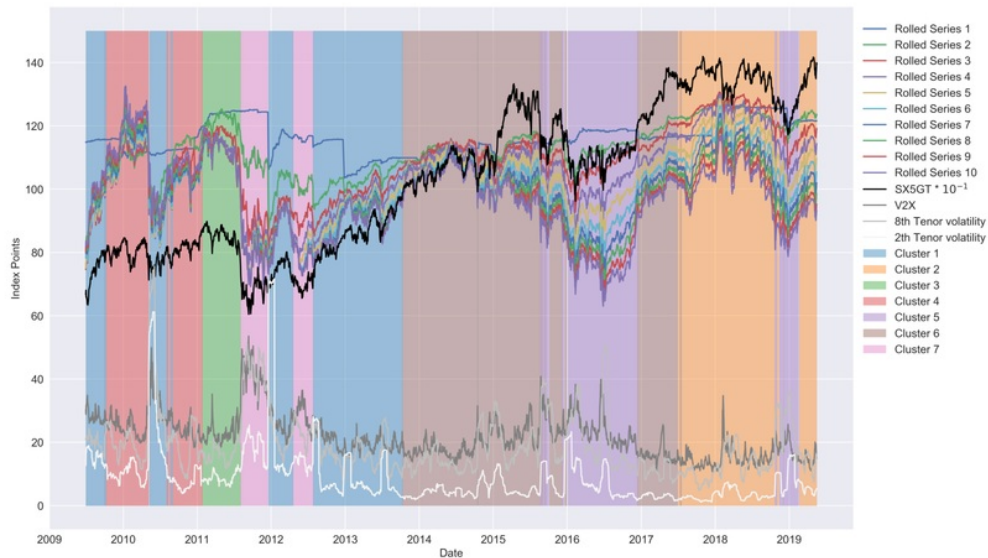


Figure 59: K-means with 7 Clusters. Clusters are slightly different from Figure 58, but still seems coherent. We can see that the period 2016-2017 has its own cluster now. The cluster corresponding to the end of 2018 is now partly on 2019. By looking at these results, we felt that some Clusters could actually be merged (in particular cluster 2 and 6).

Finally, we tried to run a DB-SCAN without applying PCA (only scaling), and if the result is not very satisfying, it has the interesting property to spot the volatility spikes as outliers. We can see that in Figure 61.

We then classify some new data points using K-Nearest-Neighbours in Figure 62 - we take the labels from DB-SCAN for that plot. As expected, we do not have a change of regime over the last two months.

## 8 Conclusion

We have explored different ways to trade Dividend Futures. However, we still didnt express a clear view on the best way to trade these contracts. In part 4, we saw that the stochastic models were interesting to identify trends in the term structure, but not effectively trade the contracts. However, we identified a way to hedge using PCA. Then, in part 5 we exposed different ways to



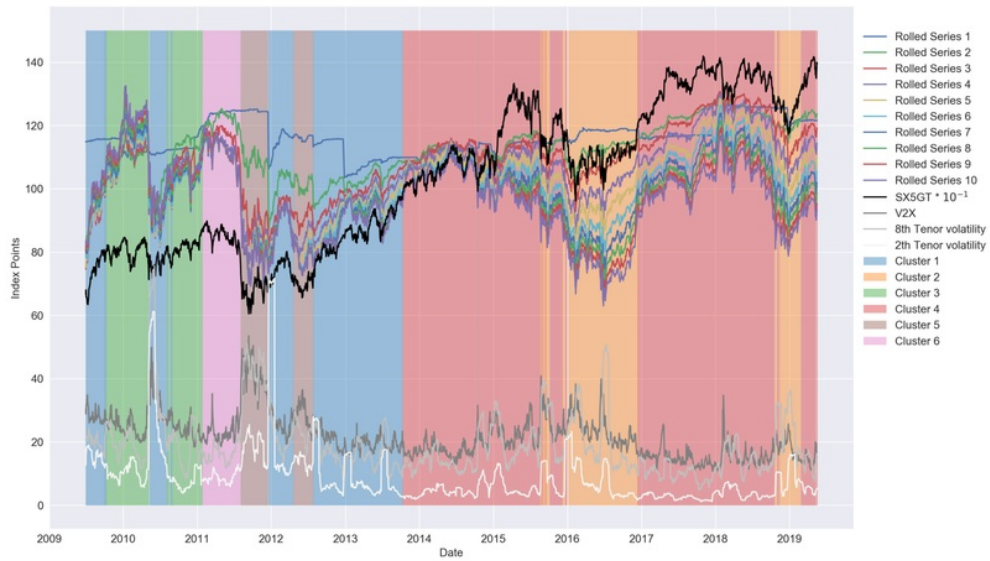


Figure 60: K-means with 7 Clusters, merged to 6. We can see that now Cluster 2 and 6 are merged (they were indeed the two closest cluster among the seven).

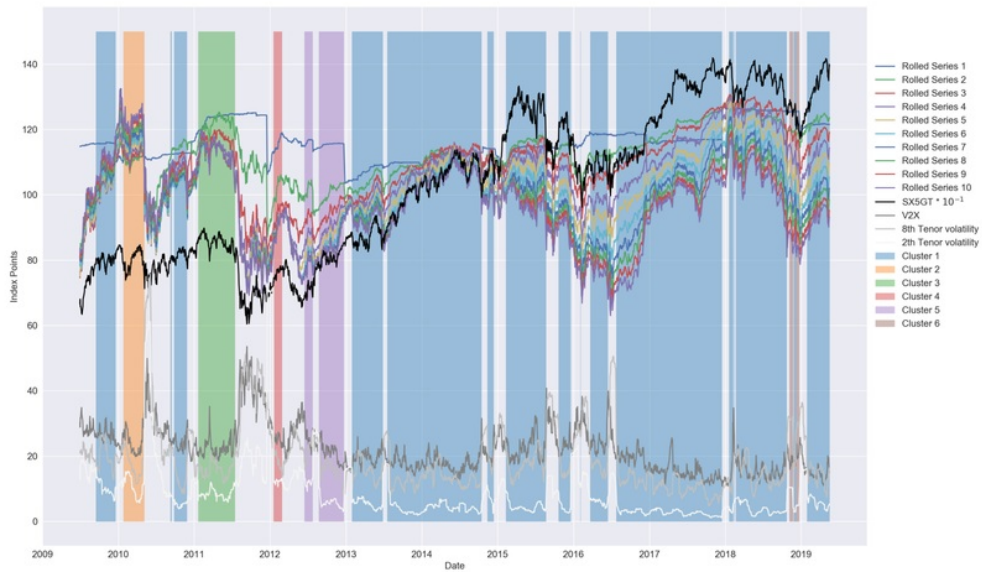


Figure 61: DB-SCAN with no PCA in pre-processing. The results are not convincing overall, but most of the volatility spikes are marked as outliers, which might be an interesting property.

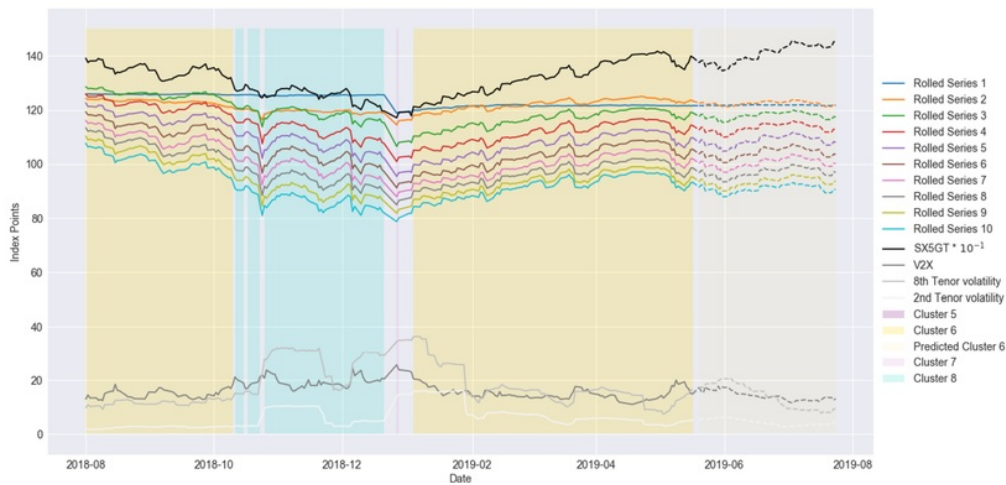


Figure 62: Example of K-Nearest-Neighbours Classifier. All points after the 17/05/2019 (dotted lines) were classified using the K-Nearest-Neighbours. We can see that we didn't experience a shift of regime over the last two months.

compute Beta for each Tenor, so we would be able to hedge our portfolio by buying/selling SX5GT (SX5GT is actually an index, so not traded, but we assume we can perfectly replicate it). In part 6, we back-tested all the Beta models to find the best one. We found that for each Tenor several models lead to portfolios uncorrelated to the market (we presented, for each Tenor, the best one).

Now, we still have to answer the following question: what is the most efficient hedging technique, PCA or Beta hedging? PCA hedging, in spite of excellent numerical results (portfolio uncorrelated to the market, portfolio's mean and volatility are 'reasonable'), doesn't seem to be the optimal solution. The first limitation to this technique is a market impact issue. Indeed, the volume of Dividend Futures traded each day is relatively low (a few thousands contracts at best for the most liquid series, a few dozens for the longer Tenors). So rebalancing a portfolio every day would expose it to important losses due to market impact - problem that disappears when trading more liquid instruments. Moreover, the interest in Dividend Futures comes from an expected high risk premium. So ideally we want to hedge using instruments with a lower risk premium (such as the underlying market).

So in practice, a Beta hedging is preferred. However, as we could see in part 5 and 6, there is different (efficient) ways to compute Beta. So the trader would have to choose the Beta that corresponds best to its view on the market (is he expecting a high/low volatility period). To help that decision, we created a tool in part 7 that can identify different regimes in the Term Structure evolution. We also used a classification method (K-Nearest-Neighbours) that can predict for each

new data point if there has been a regime shift (i.e. if we changed the cluster).

Finally, in this paper, we focused on producing portfolios that were uncorrelated to the market (SX5GT). So our portfolios don't embed any constraints to optimize the returns. This is obviously not realistic. Hedging is, in practice, a trade-off between limiting the portfolio's market exposure and optimizing returns.

## References

- [1] Eurex web site, August 2019. <https://www.eurexchange.com/exchange-en/products/did>.
- [2] Hans Büehler. "Volatility and Dividends II: Consistent Cash Dividends". January 2018. <https://ssrn.com/abstract=2639318>.
- [3] Matthias Rosenkranz. "A Principal Components-Based Dividend Term Structure Model" (Unpublished master's thesis). December 2017. University of Oxford - Christ Church College.
- [4] Mark Richardson. "Principal Component Analysis". May 2009. <http://www.dsc.ufcg.edu.br/hmg/disciplinas/posgraduacao/rn-copin-2014.3/material/SignalProcPCA.pdf>.
- [5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [6] Carlos Armando Mejía Vega. "Calibration of the Exponential Ornstein-Uhlenbeck Process when Spot Prices are Visible through the Maximum log-likelihood Method. Example with Gold Prices". 2018. <https://advancesindifferenceequations.springeropen.com/articles/10.1186/s13662-018-1718-4>.
- [7] Riccardo Rebonato, Ivan Saroka, and Vladyslav Putyatin. "A Principal-Component-Based Affine Term Structure Model". June 2014. <http://ssrn.com/abstract=2451130>.
- [8] Roel F. Ceballos, Robert Krail, and Fe F. Largo. "On The Estimation of the Hurst Exponent Using Adjusted Rescaled Range Analysis, Detrended Fluctuation Analysis and Variance Time Plot: A Case of Exponential Distribution". *Imperial Journal of Interdisciplinary Research (IJIR)*, 3(8), 2017. <https://arxiv.org/ftp/arxiv/papers/1805/1805.08931.pdf>.
- [9] Katsuto Tanaka, Weilin Xiao, and Jun Yu. "Maximum Likelihood Estimation for the Fractional Vasicek Model". *Research Collection School Of Economics*, 1(31), 3 2019. [https://ink.library.smu.edu.sg/soe\\_research/2248](https://ink.library.smu.edu.sg/soe_research/2248).

- [10] Ton Dieker. "Simulation of Fractional Brownian Motion" (Unpublished master's thesis). September 2004. <http://www.columbia.edu/~ad3217/fbm/thesis.pdf>.
- [11] Weilin Xiao and Jun Yu. "Asymptotic Theory for Rough Fractional Vasicek Models". *Research Collection School Of Economics*, 177:26–27, March 2018. [https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=3158&context=soe\\_research](https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=3158&context=soe_research).
- [12] Doug Huggins and Christian Schaller. "*Fixed Income Relative Value Analysis: A Practitioners Guide to the Theory, Tools, and Trades*". Bloomberg Financial Series. Wiley, 2013.
- [13] Clifford Asness, Robert Krail, and John Liew. "Do Hedge Funds Hedge?". *Journal of Portfolio Management*, page 5, May 2001. [https://papers.ssrn.com/sol3/papers.cfm?abstract\\_id=252810](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=252810).
- [14] Agnieszka Szparaga and Slawomir Kocira. "Generalized Logistic Functions in Modelling Emergence of Brassica napus L.". *PLoS ONE*, 13(2), 2018. <https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0201980&type=printable>.
- [15] Alexei Kondratyev. "Learning Curve Dynamics with Artificial Neural Networks". April 2018. <https://ssrn.com/abstract=3041232>.
- [16] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Universal Approximation of an Unknown Mapping and its Derivatives Using Multilayer Feedforward Networks". *Neural Networks*, 3:551–560, 1990. [http://www.inf.ufrgs.br/engel/data/media/file/cmp121/univ\\_approx.pdf](http://www.inf.ufrgs.br/engel/data/media/file/cmp121/univ_approx.pdf).
- [17] Leo Breiman. "Bagging Predictors". *Machine Learning*, 24(1):123–140, July 1996. <https://sci2s.ugr.es/keel/pdf/algorithm/articulo/1996-ML-Breiman-Bagging%20Predictors.pdf>.
- [18] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. "*The Elements of Statistical Learning: Data Mining, Inference and Prediction*". Springer, 2 edition, February 2009.
- [19] Jerome Friedman. "Greedy Function Approximation: a Gradient Boosting Machine". *The Annals of Statistics*, 29(5):1189–1232, 2001. [https://projecteuclid.org/download/pdf\\_1/euclid.aos/1013203451](https://projecteuclid.org/download/pdf_1/euclid.aos/1013203451).
- [20] Leo Breiman. "Stacked Regressions". *Machine Learning*, 24(1):49–64, July 1996. <https://link.springer.com/content/pdf/10.1007%2F00117832.pdf>.
- [21] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei XU. "A Density-Based Algorithm for Discovering Clusters". 1996. <https://www.aaii.org/Papers/KDD/1996/KDD96-037.pdf>.

**IMPERIAL COLLEGE LONDON**

**THESIS DECLARATION FORM**

(for candidates whose degree will be awarded by Imperial College)

---

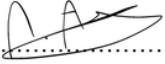
**Declaration:** I hereby confirm that the work contained in this thesis is my own work unless other  
wises stated.

**Name**.....Majd Agoumi..... **CID** .....00834447.....

**Title of Thesis**.....Hedging Dividend Futures.....

**Month and year of submission for examination**.....September 2019.....

**Date**.....09/09/2019.....

**Signature of Candidate**..........

# Hedging Dividend Futures

---

## GRADEMARK REPORT

---

FINAL GRADE

**/0**

GENERAL COMMENTS

**Instructor**

---

PAGE 1

---

PAGE 2

---

PAGE 3

---

PAGE 4

---

PAGE 5

---

PAGE 6

---

PAGE 7

---

PAGE 8

---

PAGE 9

---

PAGE 10

---

PAGE 11

---

PAGE 12

---

PAGE 13

---

PAGE 14

---

PAGE 15

---

PAGE 16

---

PAGE 17

---

PAGE 18

---

PAGE 19

---

PAGE 20

---

PAGE 21

---

PAGE 22

---

PAGE 23

---

PAGE 24

---

PAGE 25

---

PAGE 26

---

PAGE 27

---

PAGE 28

---

PAGE 29

---

PAGE 30

---

PAGE 31

---

PAGE 32

---

PAGE 33

---

PAGE 34

---

PAGE 35

---

PAGE 36

---

PAGE 37

---

PAGE 38

---

PAGE 39

---

PAGE 40

---

PAGE 41

---

PAGE 42

---

PAGE 43

---

PAGE 44

---

PAGE 45

---

PAGE 46

---

PAGE 47

---

PAGE 48

---

PAGE 49

---

PAGE 50

---

PAGE 51

---

PAGE 52

---

PAGE 53

---

PAGE 54

---

PAGE 55

---

PAGE 56

---

PAGE 57

---

PAGE 58

---

PAGE 59

---

PAGE 60

---

PAGE 61

---

PAGE 62

---

PAGE 63

---

PAGE 64

---

PAGE 65

---

PAGE 66

---

PAGE 67

---

PAGE 68

---

PAGE 69

---

PAGE 70

---



PAGE 71

---

PAGE 72

---

PAGE 73

---

PAGE 74

---

PAGE 75

---

PAGE 76

---