ASW Spread on BTPs

Comandini Leonardo, Nicoletti Antea, Schiavon Andrea

24 June 2016

Contents

List of Figures

List of Algorithms

Abstract

In this brief report we present and discuss three main topics: a small example of basic data mining, EONIA curve bootstrapping and the construction of the Asset Swap Spread curve over EONIA. A segmented regression technique is introduced to fit data under consideration. Financial interpretations and numerical results are finally presented

Keywords: EONIA swap rate, Asset Swap Spread, segmented regression, time-to-slope-change (ttsc)

1 Introduction

In this little work we describe the steps needed to construct the so-called Asset Swap Spread Curve on BTPs (ASWS curve henceforth) both from a theoretical and numerical viewpoint. The analysis has been carried out in a time lag starting from 1st January 2007 to 31st December 2015. It is worth mentioning from the beginning that the ASWS curve is an asset spread term structure computed for every business day in the considered data-set. The whole analysis hinges on the empirical evidence that the ASWS curve can be modeled by a segmented line broken in one point (a elbow-shaped curve). This fact is heavily exploited throughout the study to construct financial stress indexes (FSI henceforth) able to detect stress in the financial market at county level. Another important fact is that these FSI are extremely powerful when considered in relation with the others and not individually, that is a comparative analysis of the three carries the most information. The FSI presented are :

- **time**: time (in years) when the spread term structure changes slope
- **slope**: slope of the first regression line approximating the data
- **spread**: value of the asset spread connected to the bond with maturity 10 years

The final part of the report is devoted to present the relation between previous indexes and financial shocks in the euro zone to show how well this machinery describes the financial reality[1].

[1] all the ideas and results presented in this brief report are taken from [2]

2 Methodology

In this section we present the theoretical body of the work. It can be divided in four main parts, for each of them we provide the methodologies we followed and the assumptions we made.

2.1 Data mining

2.1.1 Data description

The given data set (provided by the info-provider *Bloomberg*) consists of :

- Overnight Index Swap rate ($R^{OIS}(t_0, t)$) market quotes for every business day t_0 in $[t_1, t_N^\star]$ where $t_1 = 01/01/2007, \quad t_N^\star = 05/01/2016$
- BTP main info, last clean and dirty mid prices

An overnight index (e.g. EONIA) is the interest rate that a bank is willing to pay for a cash deposit lasting no more than 24 hours to another financial institution belonging to the interbank market. This kind of interest rate is the benchmark for the computation of swap rates $R^{OIS}(t_0, t)$ (fixed rate paid by the long position in an interest rate swap) for different expiries. As far as the bonds are concerned, the following information is available for every BTP under consideration:

- bond identification name (e.g ED282995 Corp)
- bond settlement date t_s
- bond maturity date t_e
- bond first coupon date
- coupon type (FIXED or ZERO COUPON)
- coupon value c (expressed in percentage on a 1 €notional)
- coupon frequency f (e.g. $f = 2$ in the semiannual case)
- inflation-linked indicator (YES, NO)
- issue size [€]

Moreover, the following bond price quotes are given:

- $P^{clean}_{LAST}(0)$ = bond clean price at value date [% on a 1 €notional]
- $P^{dirty}_{MID}(0)$ = bond dirty price at value date [% on a 1 €notional]

The subscript $LAST$ stands for last quoted price in the considered trading day while MID indicates that the price is the average between the bid and ask price[2]. The difference between clean and dirty price is explained by the following relation:

$$\boxed{P^{dirty}(t) = P^{clean}(t) + accrual(t)} \tag{1}$$

where $accrual(t) = c \cdot \delta(t_{-1}, t)$ and t_{-1} = last coupon date. Nevertheless the dirty MID were at hand, we decided to based our analysis on the clean last and compute the accrual term[3].

2.1.2 Data filtering

In this section we discuss how we processed the row data-set and the filters activated for the data mining. The first filter we applied is the selection of days in the time interval of interest $[t_1, t_N]$ where $t_1 = 01/01/2007, \quad t_N = 31/12/2015$. It might then happen that for a fixed business day t the OIS rate is not given for every expiry, if this happens for at least one expiry we filtered it out (this happened 21 times out of 2349 days). The third filter takes care of deleting not-business day (33 not-business days have been filtered). We then moved to check if all bonds satisfied the following requirements:

- issue size > 500 MIO €
- inflation-linked indicator = NO
- coupon-bearing bond
- $01/01/1999 < t_s < t_N$

107 bonds fall within these constraints, 34 do not.

[2]Since the treasury bond market is extremely liquid, the MID average is close to the actual BID or ASK price

[3]As a matter of fact, Python code has both the implementation

2.2 Bootstrap over EONIA

We can now present the risk-free curve construction via bootstrap. This technique makes use of liquid instruments in the market (e.g. Interest Rate Swap, FRAs, Eurodollar Futures) to build in a recursive way the risk-free interest rate term structure and the corresponding discount factors. The liquid contracts mentioned above are based on a reference interest rate for instance LIBOR or EURIBOR. However, as explained in [2] these benchmark rates present some criticalities:

1. after Lehman's default EURIBOR is not considered purely risk-free anymore, in the sense that it reflects the banks risk aversion towards money lending over a 1-year period

2. a recent scandal involving LIBOR came to light, showing how its quotes were manipulated by big financial institutions (`http://www.economist.com/node/21558281`)

As a consequence, *Euro OverNight Index Average* (EONIA) has been chosen as the reference interest rate for curve construction. Since it is an overnight rate (lending period within the day) it can be regarded as truly risk free being the counterparty risk in such a short period nearly negligible. Following [1], the curve has been built just using EONIA Interest Rate Swaps up to 10 years and managing differently swaps with expiry within a year. The recursive relations for discount factors up to one year and beyond one year are respectively:

$$B(t_0, t_e) = \frac{1}{1 + \delta(t_0, t_e) \cdot R^{OIS}(t_0, t_e)} \tag{2}$$

$$B(t_0, t_i) = \frac{1 - R^{OIS}(t_0, t_i) \sum_{k=1}^{i-1} \delta_k \cdot B(t_0, t_k)}{1 + \delta_i \cdot R^{OIS}(t_0, t_i)} \quad i = 1, \ldots, n \tag{3}$$

where δ_k is the year fraction between t_{k-1} and t_k.
We computed the EONIA discount curve for every value date t in the dataset (t_0 is the settlement day, two business days after t, follow) using an ACT/360 day count convention. When a discount factor $B(t_0, t)$ is needed ($t \neq t_i \quad i = 1, \ldots, n$) linear interpolation on zero-rates (ACT/365 day count convention) is used, if $t < 1$ month then we imposed that $z(t_0) = z_{1m}$ since

$R^{OIS}(t, t_{1m})$ is the first market quote available in our data-set.
Here we present two examples, in one of them negative rates are present:

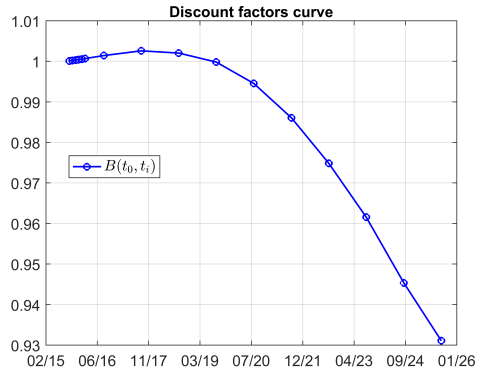


Figure 1: Discount factor curve on August 19th 2015

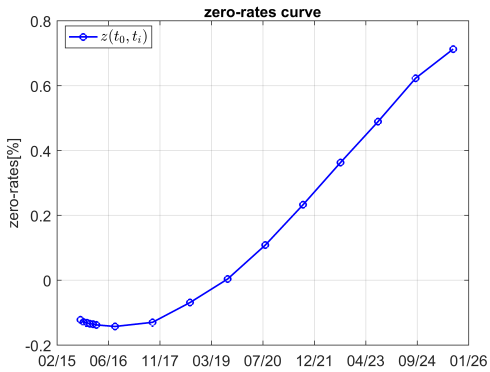


Figure 2: Zero-rates curve on August 19th 2015

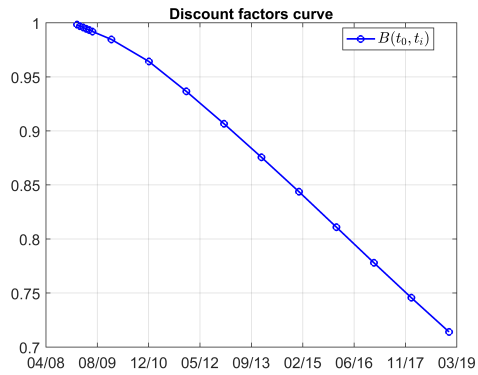


Figure 3: Discount factor curve on December 29th 2008

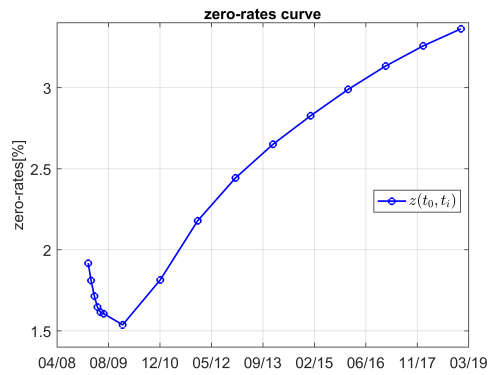


Figure 4: Zero-rates curve on December 29th 2008

2.3 ASW spread over EONIA

In this section we briefly discuss how to construct an ASWS curve: first a little digression on how an asset swap works and then results are presented.

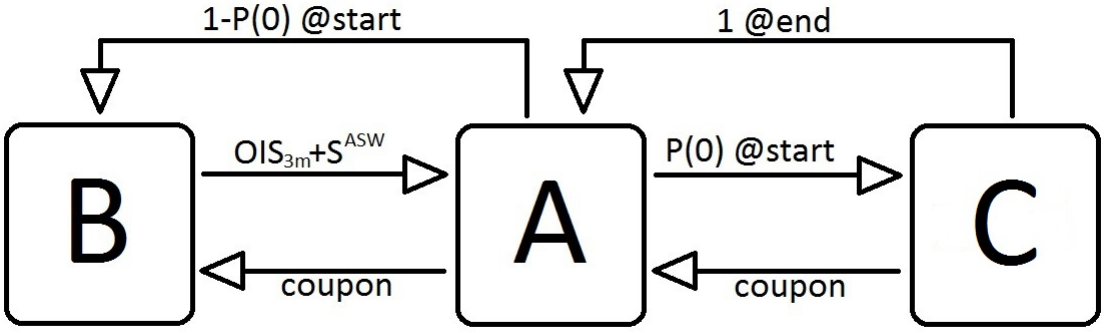


Figure 5: Asset swap mechanics

2.3.1 Assest Swap

The first step is to briefly explain the mechanics of an asset swap: it is a contract between two counterparties, say, A and B that agree to exchange the following cash flows over the time interval $[t_0, t_N]$:

- B (asset swap seller) gives A a coupon bond issued by C (credit reference) receiving the difference between the bond notional and the dirty price $(1 - P^{dirty}(0))$ at t_0 (contract settlement date). B agrees to pay a floating leg plus a spread $(OIS_{3m} + s^{asw})$ at a given frequency (quarterly)

- A (asset swap buyer) gets the bond in t_0 and pays to B coupons c at a given frequency (semiannually)

This kind of contract meets the needs of A who wants a floaters but cannot find it in the market as well as the needs of B who seeks protection against the default of the credit reference C. Indeed if C were to default A would have still to pay coupons c to B. The asset swap spread s^{asw} is determined by requiring the NPV of contract cash flows to be 0 at value date. The higher s^{asw} the riskier the bond, in other words if the spread keeps increasing over a period of time the market thinks that the credit reference might not be able to pay some coupon or even the notional amount back.
The floating payment dates are set in arrears, that is starting from T (maturity) and going back in time. In this way, in general, the first leg is a **short stub** whose rate is established using the Eonia discount curve.

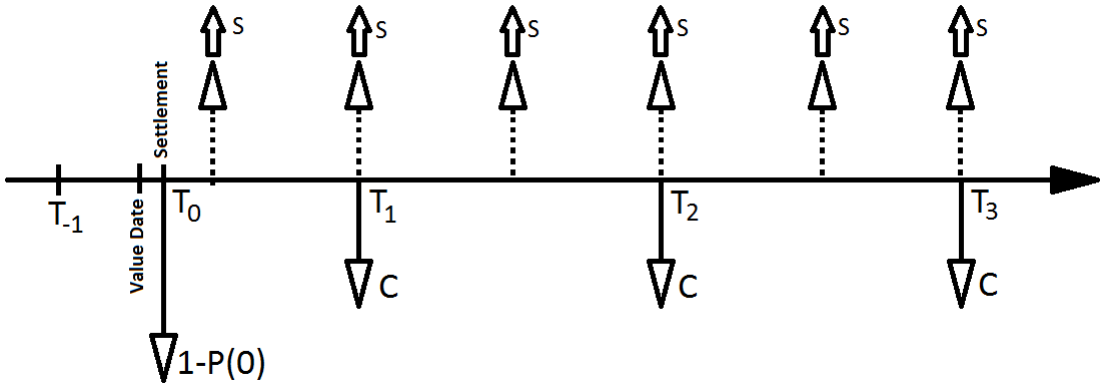


Figure 6: Asset swap cash flows

We now move to show the actual computation in order to derive a formula for the asset swap spread s^{asw}: to begin with we compute the net present value at value date of the fixed and floating leg.
Set:
t_0 = settlement date (two business days after value date)
T = maturity of the bond
$P^{fx} = \{t_{-1}^{fx}, t_1^{fx}, ..., t_N^{fx} = T\}$ = dates of the fixed payments
t_{-1}^{fx} = last coupon date before t_0, if there is not settlement of the bond
$P^{fl} = \{t_0, t_1^{fl}, ..., t_M^{fl} = T\}$ = dates of the floating payments

$$\begin{aligned} NPV_{fix}(0) &= 1 - P(0) + c \sum_{t_i \in P^{fx} \setminus \{t_{-1}^{fx}\}} \delta(t_{i-1}, t_i) \cdot B(t_0, t_i) \\ &= 1 - P(0) + c \cdot BPV^{fx}(t_0). \end{aligned}$$

$$
\begin{aligned}
NPV_{flt}(0) &= \sum_{t_i \in P^{fl} \setminus \{t_0\}} \mathbb{E}_0 \left[\delta(t_{i-1}, t_i)[OIS_{3m}(t_{i-1}, t_i) + s^{asw}] D(t_0, t_i) \right] \\
&= \underbrace{\delta(t_0, t_1^{fl})[OIS_{3m}(t_0, t_1^{fl}) + s^{asw}]}_{\text{short stub}} B(t_0, t_1^{fl}) + \\
&+ \sum_{t_i \in P^{fl} \setminus \{t_0, t_1^{fl}\}} \mathbb{E}_0 \left[\delta(t_{i-1}, t_i)[OIS_{3m}(t_{i-1}, t_i) + s^{asw}] D(t_0, t_i) \right].
\end{aligned}
$$

We make use of the flows equivalence[4]

$$
\mathbb{E}_0 \left[\delta(t_{i-1}, t_i) OIS_{3m}(t_{i-1}, t_i) D(t_0, t_i) \right] = B(t_0, t_{i-1}) - B(t_0, t_i)
$$

to rewrite each floating leg within the sum. In this way we obtained a telescopic sum:

$$
\sum_{t_i \in P^{fl} \setminus \{t_0, t_1^{fl}\}} \mathbb{E}_0 \left[\delta(t_{i-1}, t_i) OIS_{3m}(t_{i-1}, t_i) D(t_0, t_i) \right] = B(t_0, t_1^{fl}) - B(t_0, T)
$$

Substituting this result in the NPV of the floating leg, discounting it in t_0 and paying attention to the initial short stub

$$
\begin{aligned}
NPV_{flt}(0) = OIS(t_0, t_1^{fl}) \delta(t_0, t_1^{fl}) B(t_0, t_1^{fl}) + B(t_0, t_1^{fl}) - B(t_0, T) + \\
+ s^{asw} \sum_{t_i \in P^{fl} \setminus \{t_0, t_1^{fl}\}} \delta(t_{i-1}, t_i) \cdot B(t_0, t_i)
\end{aligned}
$$

but the rate corresponding to the short stub (that is fixed and hence known in t_0) is established with the usual relation with the EONIA discount curve (as requested), so:

$$
B(t_0, t_1^{fl}) = \frac{1}{1 + \delta(t_0, t_1^{fl}) \cdot OIS(t_0, t_1^{fl})}
$$

$$
\implies \delta(t_0, t_1^{fl}) \cdot OIS(t_0, t_1^{fl}) \cdot B(t_0, t_1^{fl}) = 1 - B(t_0, t_1^{fl}).
$$

Hence,

$$
NPV_{flt}(0) = 1 - B(t_0, T) + s^{asw} BPV^{fl}(t_0).
$$

[4] see [3] pag 4

Solving

$$NPV_{fix}(t_0) = NPV_{flt}(t_0)$$

for s^{asw} we get

$$\boxed{s^{asw} = \frac{B(t_0, t_N) + c \cdot BPV^{fx} - P(0)}{BPV^{fl}}} \tag{4}$$

Remark 1 *The bond price in the above formula is the dirty $P^{dirty}(0)$ at value date (see (1)). The accrual term[5] has been computed with an 30/360 day count convention over the period $[t^{fl}_{-1}, t_0]$.*

Remark 2 *The numerator of (4) can be seen as $\overline{P}(0) - P(0)$, where $\overline{P}(0)$ is the price of a risk free coupon bond. This equation highlights the fact that that if the asset spread is negative the bond is considered less risky than a risk-less bond.*

2.3.2 Second bond filtering

We spend here a few words explaining the second bond filtration (the first filtration took place while reading the data (see subsection (2.1)). The filters applied to bond data can be summarized as follow:

Filter1: consider only bonds whose maturity satisfies $t + 2months < t_e < t + 10years$ for every business day t in the filtered data-set.

Filter2: some bonds have a price before their settlement so we require $t_s < t$.

Filter3: some bonds lack a price quote for a certain business day t. For those bonds it is not possible to compute the asset swap spread so they are filtered out.

Filter4: we noted that the bond price data-set included some outliers, for example bond *EJ000941 Corp* is quoted $1717,988€$ on the 21st September 2012 hence we activated a filter to discard unreasonable values.

[5] if a coupon date coincides with the settlement t_0 we checked that the clean last is very close to the dirty MID, this means that the accrual is maximum one day before a coupon date and zero on a coupon date. From a modeling point of view we can suppose the coupon payment happens at $t_i - dt$ where t_i is a coupon date

On average, an asset swap spread term structure consists of 34 spreads, this means that 34 bonds on average survive the second filtration process out of 107.

Examples In this section a few examples of ASWS curves are reported for different business days. The first two curves are computed in a period of financial stress for the Republic of Italy while the third refers to a period of relative quiet(before Lehman's fall). Financial interpretation of these evidences is postponed to the last section (see (3)).

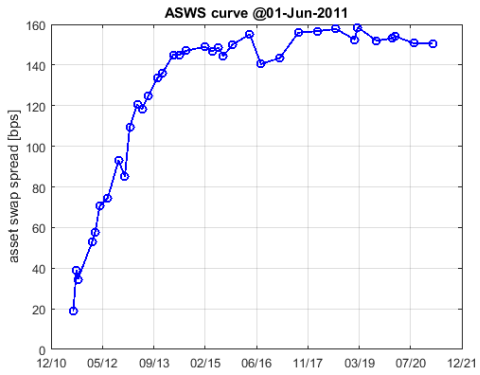


Figure 7: ASWS curve presents an elbow shape

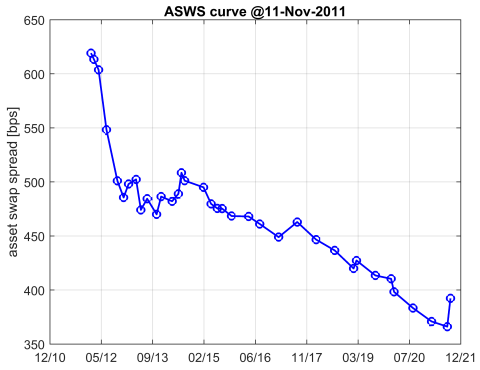


Figure 8: ASWS curve "inverted"

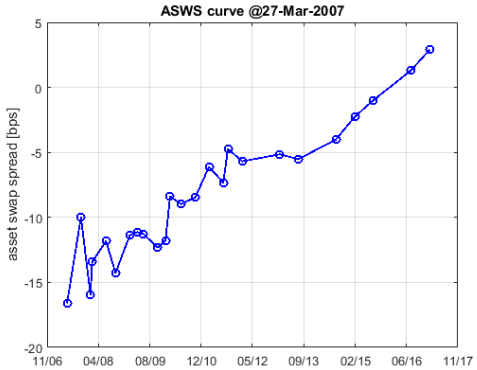


Figure 9: ASWS curve with negative spreads

Remark 3 *It can happen that on two consecutive days the spread for a fixed bond has a jump of more than 50 bps (in absolute value) and on the next day another jump but in the opposite direction. As explained in [2] these phenomena seem to be related to market illiquidity and the mid value is set equal to the average between the other two spread values.*

2.4 Segmented Regression

2.4.1 Algorithm description

Once again following [2], in this section we present the technique for fitting a constrained linear regression suitable for data showing an elbow-shaped pattern. Firstly a brief description of the algorithm (see section (5) for further details) is discussed and then some examples on Asset Swap Spread data are shown. Let us first set the notation

- $\{s\}_{i=1,\dots,n}$ = spreads
- $\{T\}_{i=1,\dots,n}$ = bond expiries
- τ_k = time-to-slope-change (ttsc) at step k
- L_k = least residual sum of squares at step k
- $\tau^\star$ = time-to-slope-change
- $L^\star$ = least residual sum of squares

The algorithm idea is to iteratively split the data-set into two sets, fit independent or constrained linear regressions and select the one giving best performances. Let's suppose we are at step k, the two sets are $\{T\}_{i=1,\dots,k} \quad \{s\}_{i=1,\dots,k}$ and $\{T\}_{i=k+1,\dots,n} \quad \{s\}_{i=k+1,\dots,n}$. Fitting two independent linear regressions we get the lines parameters (a_1, b_1, a_2, b_2) and the residual sum of squares, $L_k = L_{k,1} + L_{k,2}$, at the current step . If a straight line fits perfectly the data $(a_1 = a_2, b_1 = b_2)$ then we deduce there is no slope change at this step and we set $\tau = \infty$ otherwise we simply compute the intercept $\tau = \frac{b_2 - b_1}{a_1 - a_2}$. If τ belongs to the interval $[T_k, T_{k+1}]$ we select it as the k-th ttsc otherwise (if L_k deserves the effort, that is $L_k < \min_{j<k} \quad L_j$) we compute two constrained linear regressions. This means that a continuity constraint is imposed, binding the lines to have the same value in $\tau = T_k$ for the left case and $\tau = T_{k+1}$ for the

right one. The constrained regression that produces the lowest L is chosen. These instructions are repeated until ending up with just three point to fit the right regression, then the lowest L_K is picked with the corresponding τ_k. Finally, the goodness of fit of the regression model is measured by means of the coefficient of determination R^2 that is defined as follow:

$$\begin{aligned} R^2 &= \frac{\sum_{i=1}^{n}(\widehat{s}(T_i) - \overline{s})^2}{\sum_{i=1}^{n}(s_i - \overline{s})^2} \\ &= \frac{\sum_{i=1}^{n}(s_i - \overline{s})^2 - \sum_{i=1}^{n}(s_i - \widehat{s}(T_i))^2}{\sum_{i=1}^{n}(s_i - \overline{s})^2} \\ &= \frac{SS_{tot} - SS_{res}}{SS_{tot}} \\ &= 1 - \frac{SS_{res}}{SS_{tot}} \\ &= 1 - \frac{L^\star}{SS_{tot}}. \end{aligned}$$

It can be proved that $R^2 \in [0, 1]$ (see [2]). R^2 is used as a measure of how well the model describes the data since it is the percentage of variability explained by the model (e.g. if $R^2 = 1$ there is a perfect fit and the line perfectly interpolates the data).

2.4.2 Constrained regression

Here we are going to present how we solved the least squares problem. The problem can be stated as follow

$$\begin{aligned} \underset{p}{\text{minimize}} \quad & \sum_{i=1}^{n}\Big(s_i - f(p, T_i, \tau)\Big)^2 \\ \text{subject to} \quad & A \cdot p = d \end{aligned} \tag{5}$$

where

$$f(p, T_i, \tau) = \begin{cases} a_1 T_i + b_1 & \text{if } T_i \le \tau \\ a_2 T_i + b_2 & \text{if } T_i > \tau \end{cases}, \quad i = 1, \dots, n$$

$$A = \begin{bmatrix} 1 & \tau & -1 & -\tau \end{bmatrix}, \qquad p = \begin{bmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \end{bmatrix}, \qquad d = 0$$

This is a constrained optimization problem in a four-dimensional space so it could be quite slow to solve numerically (see section (4)) but we can simplify it by re-stating it as follow:

$$\underset{x}{\text{minimize}} \quad \left\| Cx - y \right\|^2 \tag{6}$$

where

$$C = \begin{bmatrix} T_1 & 1 & 0 \\ T_2 & 1 & 0 \\ \vdots & \vdots & \vdots \\ T_k & 1 & 0 \\ \tau & 1 & T_{k+1} - \tau \\ \vdots & \vdots & \vdots \\ \tau & 1 & T_n - \tau \end{bmatrix}, \qquad y = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix}, \qquad x = \begin{bmatrix} a_1 \\ b_1 \\ a_2 \end{bmatrix}$$

and the continuity constrain is expressed by the following relation

$$b_2 = b_1 + a_1\tau - a_2\tau. \tag{7}$$

This is an overdetermined system since the number of equations is greater then the number of unknowns. In numerical analysis this is a well-known problem and a new definition of solution has been introduced in this cases: $x^\star$ is a solution for the problem (6) iff

$$\left\| Cx^\star - y \right\| \leq \left\| Cx - y \right\| \qquad \forall x \in \mathbb{R}^3.$$

To deterine $x^\star$ it is needed to solve:

$$C^T C x = C^T y$$

Efficient numerical routines are available to solve this kind of problems (see section (4) for further details), the only thing to check is that $rank(C) = 3$, it is enough to show that C has 3 independent rows:

$$det \begin{bmatrix} T_1 & 1 & 0 \\ T_2 & 1 & 0 \\ T_k & 1 & T_{k+1} - \tau \end{bmatrix} = (T_{k+1} - \tau)(T_1 - T_2) \neq 0$$

Examples We report here the same plots of subsection (2.3) just adding the linear fit. The results are the following:

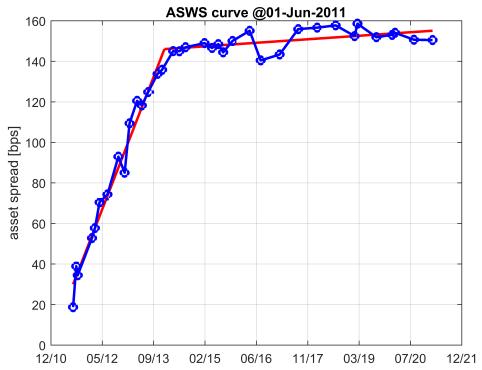


Figure 10: segmented regression on 1 Jun 2011

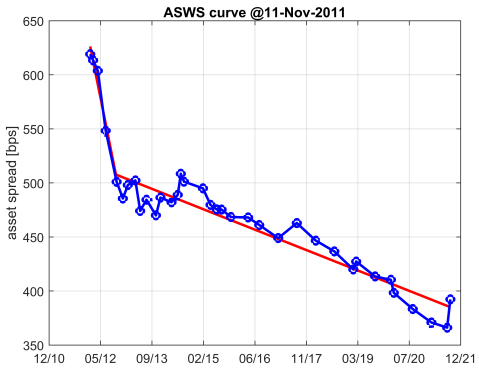


Figure 11: segmented regression on 11 Nov 2011

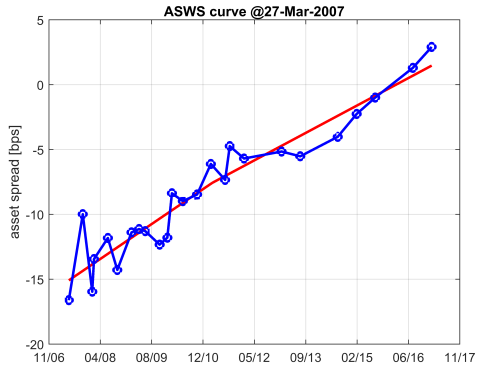


Figure 12: segmented regression on 27 Mar 2007

3 Results & Financial Interpretations

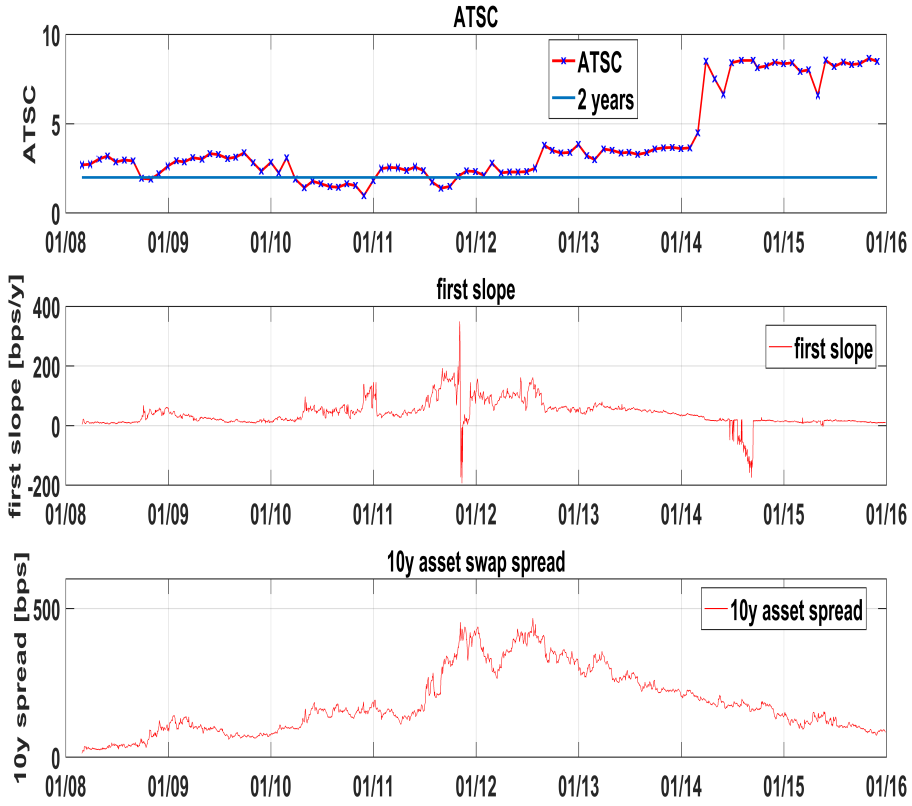
In this section (following [2]) we give a financial interpretation of three quantities deriving from ASWS curves : **time**, **slope**, **spread**. As stated at the beginning, the most financial information is carried by the three quantities together and not when considered individually. By **spread** we mean the

value of the 10-year asset swap spread [6] on BTPs which is a measure of how risky the market is assessing that particular bond. In our analysis we consider just spreads whose monthly average is above the threshold of 20 bps (14 months don't satisfy the requirement, up to the beginning of 2008). The **slope** is nothing else that the angular coefficient of the first regression line, it gives an idea of how fast the spread is growing on the short period. In general **slope** is smaller for short maturities and usually positive but "inverted" ASWS curves are possible during extremely stressed financial period (see fig (11)). Finally, **time** has been defined as the difference (in years) between the time the slope of the ASWS curve changes and settlement date.
Since we are interested in a quite long time interval we considered the monthly average[7] of **time** and set its critical threshold to 2 years: every time ttsc goes below 2 years, the bond issuer (in this case Republic of Italy) is experiencing some difficulties in placing its obligations in the market since investors believe it is not going to honor its debt. What is interesting is that during stressed financial periods these three measures move accordingly as can be seen in the following figure:

[6] We picked the latest available that precisely corresponds to a bond whose maturity is $\leq$ 10y.

[7] the mean has been computed only for those business days that have **slope** positive

Figure 13: **time**, **slope**, **spread** over the period under consideration.

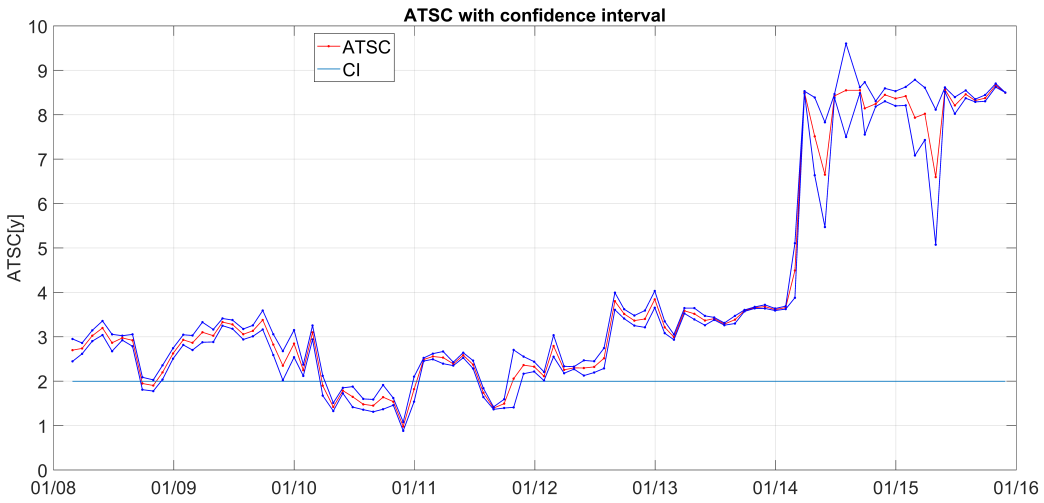


The first time ATSC falls below the threshold is immediately after Lehman's default (September 2008), correspondingly **slope** and **spread** start increasing. **Time** goes below 2 again on May 2010 when the Greek crisis struck, as in the previous case a decrease of **time** under the 2-year threshold is accompanied by a boost in both **spread** and **slope**. Finally, on August 2011 Italy faced a severe debt crisis. This was the beginning of a period of financial instability that culminated with the resignation of the Italian prime minister in favour of a technocratic government led by Mario Monti. Once again, these financial events are detected by the three FSI measures. What is more, on November 2011 an asset spread curve inversion was reported (see figure (11)), that is not only Italy found it difficult to place its obligation on

the market but also a large number of bond-holders tried to close out their long positions causing the short-term spreads to skyrocket. To sum up, we can think of **time** as an alarm bell: if it goes below a critical threshold the bond issuer is experiencing some problems accessing the capital market; if in addition to that there is also an inversion of **slope** then the bond issuer is perceived to be close to default.

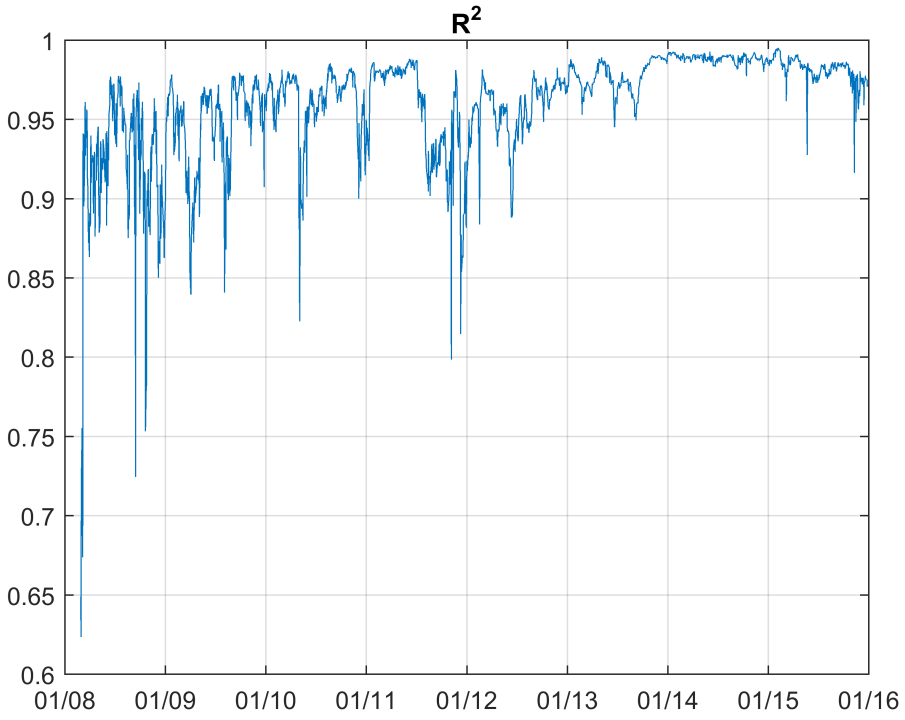
We conclude this section by checking the robustness of our results. Since ATSC is a monthly average we are interested in its variability around the mean value. The following plot tries to give an answer to it:

Figure 14: ATSC with its monthly confidence interval.



Almost every time the mean value is below the threshold also the corresponding confidence interval ($\alpha = 0.95$) is below, saying that the result is quite robust. Then we checked how well the segmented regression fits the data

Figure 15: R^2.



The answer is positive since it assumes almost always values $\in [0.8, 1]$

Final remarks and what happens next In this short report we touched the following topics:

- the bootstrapping of the EONIA risk-free curve using OIS data, justifying why it has become the reference one over the EURIBOR risk-free curve.
- the construction of the Asset Swap Spread term structure on Italian BTPs: first a brief introduction on how an asset swap works then the proof of the main formula and finally some examples were provided.
- the segmented regression methods to fit data showing elbow-like pattern: first an idea of the algorithm and the numerical strategies to perform it and then some examples were discussed.

- financial interpretation of **time**, **slope** and **spread** linking then with financial turmoils in the European financial market.
- a compendium on the two programming languages used to develop the analysis

We conclude the theoretical part of this work by proposing how the analysis could be extended further:

- by looking at figure (13) we note that immediately after the beginning of 2014 **time** grows significantly as its variance does (see figure (14)). We checked the ASWS term structures on the time span 2014/2015 and it can be seen that the slope-changing instant is far from settlement and close to the last maturity causing the ASWS curve to look more like a straight line than a segmented line. So we could have also considered bonds with longer maturities to see how the curve behaves and check if we hypothesis of elbow-shaped curve if verified also after 2014.
- Instead of just considering the R^2 as a measure of goodness of fit we could slightly modify it in order to be able to discern if the pattern is either segmented or straight. The problem is that, in the case of a nearly-straight line, it is almost impossible that the regression algorithm detects the *unique fit* since the condition is too binding. Therefore, little variations of even few spread values might cause large variation on **time** (see after 2014 in figure (14)).

  One solution could be to relax the *unique fit* condition in this way: at the beginning of the algorithm we perform a linear regression on the whole set of data on top of the two independent ones, save L_{sing}. Once having selected $L^\star$ compare it with L_{sing}, if $L_{sing} - L^\star < \epsilon$ [8] then the data don't show an elbow-like pattern so set $\tau^\star = \infty$ since it is a nearly straight line.

  While using the standard algorithm $\tau = \infty$ was almost impossible, now it has to be managed in a suitable way.
- This project was redacted immediately before the "Brexit" (that happened on the delivery day). It would be interesting to replicate the above analysis on up-to-date data to see how the three indexes would behave.

[8] ϵ has to be calibrated on data

4 MATLAB vs Python

4.1 Guide to MATLAB code

In this section, a guide to the reading of the MATLAB code is given. It provides a list of the used functions explaining with few words their main task.
The run section of the MATLAB code, ***Run_Project9***, is divided in 3 subsection, as required:

- **Run_Project9_A**: Build a database.

  A.1 *Clean_NaN*: clean a vector from Not a Number elements.

  A.2 *eurCalendar*: contains all non trading day according to Euro Calendar[9].

  A.3 *find_dates*: finds the future business days related to a given maturity, in order to find the knots (dates where bootstrap is computed). It uses A.2.

  A.4 *pay_floating*: finds dates in arrears, starting from the expiry of the bond and going back in time, with quarterly frequency. The output vector contains all floating payment dates. It uses A.2.

  A.5 *pay_fixed*: performs the same task of the previous function with semiannual frequency. However, the output vector contains the settlement date of the bond and all fixed payment dates (it is due to computational issues). In this function we approximate the stopping date to the first coupon date, neglecting the info provided from the dataset (since in some cases this date is not a business day). It uses A.2.

  A.6 *ReadXL_EONIA*: reads EONIA data from the file Excel 'INPUT_rate_curves', performs data-mining. It uses A.1, A.2.

  A.7 *BootstrapEONIA*: bootstrap the EONIA discount curve using OIS rates up to 10y. It uses A.2, A.3.

  A.8 *ReadXL_bond*: reads bond info and bond prices from the file Excel 'INPUT_BTP_Dirty', applies the filters presented in section (2.1.2) and finds the payments (fixed and floating) of the Asset Swap package. It uses A.1, A.4, A.5.

[9]This function was kindly provided us by prof. Baviera.

A.9 *plot_run_A*: plots the results.

The results we drew from this section code are stored in three objects:

- *EONIA*: vector of structs, each one containing: *Dates*, knots in which the bootstrap is computed, *Rates*, OIS rates obtained from the provider, and *DiscountFactors*, obtained from the bootstrap.
  EONIA : 1x2295 struct

Fields	Dates	Rates	DiscountFactors
1	16x1 double	16x1 double	16x1 double
2	16x1 double	16x1 double	16x1 double
...	...	...	...

- *bond*: vector of structs, each one containing all information about the filtered bond and in particular the fields of the structs are: *BBGname, settleDate, expDate, firstCouponDate, couponValue, couponFrequency, pricesDates, pricesDirtyValues*.
  bond : 1x107 struct:

Fields	BBGname	settleDate	expDate	firstCouponDate	...
1	'ED282995 Corp'	731962	733057	732143	...
2	'EC449237 Corp'	731111	733102	731276	...
...	...	...	...	...	...

...	couponValue	couponFrequency	pricesDates	pricesCleanValues
...	0,0275	2	10x1 double	10x1 double
...	0,045	2	43x1 double	43x1 double
...	...	...	...	...

- *Payments*: vector of structs, each one containing floating and fixed payment dates.
  Payments : 1x107 struct:

Fields	floating	fixed
1	12x1 double	7x1 double
2	22x1 double	12x1 double
...	...	...

- **Run_Project9_B**: Compute ASW spread over EONIA.

B.1 *filter_bond*: filter bond according to section (2.3.2).

B.2 *Discount_factors*: computes a linear interpolation for the discount factors.

B.3 *Asset_spread*: computes Asset Swap Spread over EONIA for every filtered bond and for a fixed value date. It uses B.1,B.2.

B.4 *make_spreads*: computes the spread for every value date in the time interval considered and filter as mentioned in section (2.3). It uses A.2, B.3.

B.5 *plot_run_B*: plots the results.

The results we drew from this section code are stored in *ASWSpread*, that is a vector of structs, each one with two vector fields:

- *ExpiryDates*, that are the bonds' expiry dates;
- *ASWSpreads*, containing the computed spreads.
  ASWSpread : 1x2295 struct:

Fields	ASWSpreads	ExpiryDates
1	23x1 double	23x1 double
...	...	...
1700	40x1 double	40x1 double
...	...	...

- **Run_Project9_C**: construct the best fit for a straight line broken in one point, compute the monthly average of the time to slope change (ATSC).

C.1 *monthly_avg*: compute the monthly average of values.

C.2 *filter_spread*: filters out the months in which the monthly average of the ASW spreads is constantly below 20 bps. It uses C.1.

C.3 *linear_regression*: computes an independent linear fit for a set of values.

C.4 *constrained_optimization*: performs two linear regression, imposing the continuity on the time-break.

C.5 *segmented_regression*: fixing the value date, the algorithm for segmented regression (see section 5) is implemented, determining the optimal single-segmented line and the parameters of the regression. It uses C.3,C.4.

C.6 *Avg_TTSC*: computes the segmented regression for every value date, computes the ATSC and applies the filter on the first slope.

C.7 *coefficient_of_determination*: the coefficient of determination R^2 is implemented.

C.8 *plot_run_C*: plots the results.

4.1.1 Numerical problems

The numerical problems faced on the realization of the code involved two functions: *linear_regression* and *constrained_regression*.
In order to solve the least squares minimization problem in the linear case we used the MATLAB function *polyfit*. In the constrained case, we have to find the 4 optimal parameters, imposing, on one of these, the constraint on the continuity of the time-break. Firstly, we used the MATLAB function *fminsearch* to obtain the 3 independent ones. This way to proceed was not the fastest. Hence, we used the MATLAB function *lsqlin* that solves a least squares problem with constraint. Nevertheless, also this function is not the optimal one. Finally, we opt for the method explained in section 2.4.2, solving the system with \, that provides the best performance. Unfortunately, these functions are not the fastest.

4.2 Guide to the Python code

The purpose of this section is to help reading the python code, giving a sort of map of classes and functions. This guide is not enough to understand completely how the code works, for a deeper comprehension have a look directly to the heavily commented code.

4.2.1 Introduction

The structure of the Python code[10] is quite different from the MATLAB one and doesn't follow to the letter the requests for two reasons: the first is to take advantage of the object-oriented programming style and the second is to use *pandas* library to manage the data.
Our choice was to store the data in *dataframes*, that are objects provided by

[10] Actually all three members of the group were quite new to Python and even newer to *pandas* so the code will be for sure improvable

pandas reminding a $2-dim$ table with labels for rows and columns. These labels may contain complex objects such as *datetime.date* and to access the elements of the *dataframe* one has to specify the precise labels. This is a profound difference with the way MATLAB works, for instance it is not necessary to sort the data in a particular way, it is enough to sort them when needed.

4.2.2 Main Class: FSI

Class where the data are stored and manipulated.

Attributes

A1 dates : *dataframe of the dates corresponding to the given rates*

	1m	2m	...	10y
2015-12-31	2016-02-05	2016-03-07	...	2026-01-05
2015-12-30	2016-02-04	2016-03-04	...	2026-01-05
...	...	...	...	...

A2 rates : *dataframe of the given rates*

	1m	2m	...	10y
2015-12-31	-0.002235	-0.002290	...	0.007540
2015-12-30	-0.002260	-0.002250	...	0.007540
...	...	...	...	...

A3 discount : *dataframe of the discount factors corresponding to the given rates*

	1m	2m	...	10y
2015-12-31	1.000192	1.000395	...	0.925015
2015-12-30	1.000195	1.000375	...	0.924999
...	...	...	...	...

A4 zero_rates : *dataframe of the zero rates corresponding to the given rates*

	1m	2m	...	10y
2015-12-31	-0.002266	-0.002322	...	0.007788
2015-12-30	-0.002292	-0.002282	...	0.007788
...	...	...	...	...

A5 bond_info : *dataframe of the bond information*

	FIRST_SETTLE_DT	MATURITY	...	AMT_ISSUED
ED282995 Corp	2004-01-16	2007-01-15	...	1.660000e+10
EC449237 Corp	2001-09-17	2007-03-01	...	1.626250e+10
...	...	...	...	...

A6 bond_prices : *dataframe of the bond prices*[11]

	ED282995 Corp	EC449237 Corp	...	EK699416 Corp
2007-01-01	0.99972	1.00134	...	NaN
2007-01-02	0.99974	1.00132	...	NaN
...	...	...	...	...

A7 spread : *dataframe of the spreads*[12]

	EG900350 Corp	ED973742 Corp	...	EK699416 Corp
2008-03-03	5.26693	1.69936	...	NaN
2008-03-04	8.05875	2.97392	...	NaN
...	...	...	...	...

A8 result : *dataframe of the results of the regression* [13]

	tau_star	TTSC	ATSC	month	L_star	first_slope	s_10y
2015-12-31	2024-05-16	8.380822	8.48394	2015-12	846.191	10.6711	84.8132
2008-03-07	2009-08-20	1.454795	2.74982	2008-03	379.845	10.4852	84.8131
...	...	...	...	...	...	...	...

[11] This dataframe contains NaN values where the price were not observed
[12] Also this dataframe has some NaN values when the spread cannot be computed
[13] s_10y is spread related to the longest maturity ($\leq$ 10 y)

Methods

M1 __init__ : *constructor*

M2 copy : *copy constructor*

M3 filter_bond : *activate the filters on* ***A5*** ***A6***

M4 filter_discount : *activate the filters on* ***A1*** ***A2*** ***A3*** ***A4*** *(uses* ***B3****)*

M5 compute_spread : *compute the spread filling* ***A7*** *(uses* ***S3-11****)*

M6 filter_jumps : *activate the filter of jumps on* ***A7***

M7 filter_too_low : *activate the filter of too low values on* ***A7***

M8 single_seg_reg : *perform the segmented regression on a slice of* ***A7***

M9 ATSC : *compute TTSC and ATSC filling* ***A8*** *(uses* ***G3****)*

4.2.3 p9_read

14

R1 readXL_EONIA : *read the rates from the Excel file filling* ***A2*** *(uses* ***S1****)*

R2 readXL_BTP_info : *read the bond information from the Excel file filling* ***A5*** *(uses* ***S2****)*

R3 readXL_BTP_data : *read the bond prices from the Excel file filling* ***A6*** *(uses* ***S1****)*

R4 bootstrap_EONIA : *bootstrap the rates filling* ***A1*** ***A3*** ***A4*** *(uses* ***B4*** ***B7*** ***S3****)*

[14] Some Excel file were not written in a way *panda.read_excel* could work easily, we decided not to modify 'by hands' the given files except for two things: in the rates and prices files we set all the dates to the Excel serial format, in the bond info file we erased one cell with an anomalous value appearing in the blank part of the sheet.

4.2.4 p9_support

S1 xldate_to_datetime : *convert dates from Excel serial number format to datetime.date*

S2 str_to_date : *convert dates from string format to datetime.date*

S3 yearfrac : *compute the year fraction*

S4 interp_bootstrap : *compute the discount factors interpolating on the bootstrap results (uses **B4**)*

S5 BPV_fixed : *compute the Basis Point Value of the fixed leg (uses **S3**)*

S6 BPV_float : *compute the Basis Point Value of the floating leg (uses **S3**)*

S7 pay_fixed : *find the dates in which the fixed payments occur (uses **B7**)*

S8 pay_float : *find the dates in which the floating payments occur (uses **B7**)*

S9 filter_pay_fix : *cut the fixed payments according to the value date*

S10 filter_pay_flo : *cut the floating payments according to the value date*

S11 find_t0 : *find the value dates t s. t. $t \in [t + 2m, t + 10y]$ (uses **B6**)*

4.2.5 p9_business

B1 EurHolidayCalendar : *class to generate the calendar*

B2 bday : *object used to manage the business day (uses **B1**)*

B3 isbday : *true if business day (uses **B2**)*

B4 busday : *advance of n business days (uses **B2**)*

B5 busdayMF : *find the next business day with the convention 'modified follow' (uses **B2**)*

B6 monthdelta : *advance of $delta$ months*

B7 monthdeltaMF : *advance of $delta$ months with the convention 'modified follow' (uses **B3** **B5**)*

4.2.6 p9_reg

G1 linear_regression : *perform linear regression*

G2 constrained_optimization : *perform a constrained optimization*

G3 segmented_regression : *perform segmented regression (uses **G1** **G2**)*

4.2.7 run_project9

The run file is divided in the following sections

U1 reading from excel *(uses **R1** **R2** **R3**)*

U2 bootstrap EONIA *(uses **R4**)*

U3 initialize the class

U4 filter bonds *(uses **M3**)*

U5 filter discount *(uses **M4**)*

U6 compute spread *(uses **M5**)*

U7 filter jumps[15] *(uses **M6**)*

U8 filter too low *(uses **M7**)*

U9 ATSC *(uses **M9**)*

U10 plot and display data and results

After having run all the sections one can display other data and perform extra computations.

4.2.8 Numerical problems

To solve the overdetermined system (see section (2.4.2) equation (6)) we tried with *np.linalg.lstsq* (that returns the least-squares solution to a linear matrix equation) and via *np.linalg.pinv* (that compute the *Moore-Penrose* pseudo-inverse matrix). The second performed better so we chose it.

[15] There are some jumps very close to the given threshold ($50bps$), therefore, due to numerical approximations, in Python the jumps detected are two more than in Matlab

4.3 Comparison between MATLAB and Python

Matlab takes a great advantage of both its matrix notation and the already implemented functions (e.g. *yearfrac()*) and in our case resulted faster.
Python turned out to be slower mainly because we lack of experience in programming with this language and marginally because some Excel files were not written in the easiest way to be read with *pd.read_excel()*. Nevertheless, after having filled all the *dataframes* it's much more easy and intuitive to access data and results and eventually perform some extra computations on these.

5 Appendix

Algorithm 1: Segmented Regression

Data: $\{s\}_{i=1,\dots,n} \quad \{T\}_{i=1,\dots,n}$
Result: $\tau^\star \quad L^\star$

1 **begin**
2 $k = 3, L_2 = \infty$;
3 split data into 2 sets: $[1, k], [k+1, 1]$;
4 fit two indipendent regressions on the two sets; compute L_k;
5 **if** *Unique fit* **then**
6 $\tau_k = \infty$;
7 **else**
8 compute lines' intercept τ;
9 **if** $\tau \in [T_k, T_{k+1})$ **then**
10 $\tau_k = \tau$;
11 **else**
12 **if** $L_k < \min_{j<k} \quad L_j$ **then**
13 constrained linear regression, $\tau = T_k$, compute L_L;
14 constrained linear regression, $\tau = T_{k+1}$, compute L_R;
15 **if** $L_L > L_R$ **then**
16 $\tau_k = T_{k+1} \quad L_k = L_R$;
17 **else**
18 $\tau_k = T_k \quad L_k = L_L$;
19 **end**
20 **else**
21 **end**
22 **end**
23 **if** $k < n - 3$ **then**
24 $k = k + 1$;
25 go to line 3
26 **else**
27 $k^\star = k : L_k = \min_{j<k} \quad L_j$;
28 $\tau^\star = \tau_{k^\star}, L^\star = L_{k^\star}$;
29 **end**
30 **end**

References

[1] Roberto Baviera e Alessandro Cassaro, *A note on Dual-Curve Construction: Mr. Crab's Bootstrap*, Applied Mathematical Finance, 2014.

[2] Roberto Baviera e Davide Lebovitz, *A soverein bond based index for financial instability in the euro zone and the role of carry trade*, 2015.

[3] Roberto Baviera, *Financial Engineering: Interest Rates' Curve*, 2016