A Work Project presented as part of the requirements for the Award of a Master Degree in Economics from the NOVA – School of Business and Economics.

Empirical Analysis of the Impact of Monetary Policy in the Euro Area

How do monetary policy shocks affect financial markets and economic activity in the Euro Area?

Nazeerah Balogun, 33095

A Project carried out on the Master in Economics program, under the supervision of:

Luis Catela Nunes

03.01.2020

Abstract

This paper studies the effects of monetary policy in the aggregate Euro Area. Contrary to traditional money shock analysis, this paper uses a vector autoregressive model and estimates the structural shocks through an external instrument identification approach, employing high-frequency financial data as instrument. The model inhibits economic as well as financial variables and uses the movement of Eurozone overnight index swaps around monetary policy meetings as proxy for unexpected monetary policy shocks. The results show, that a contractionary monetary policy shock behaves contrary to theory, indicating a bias in high-frequency identification. Apart from the application of high-frequency identification, this paper contributes to the literature by using Python for the estimation and identification of the model.

JEL Classification: E44, E52

Keywords: Proxy SVAR, high-frequency identification, monetary policy, Euro Area

This work used infrastructure and resources funded by Fundação para a Ciência e a Tecnologia (UID/ECO/00124/2013, UID/ECO/00124/2019 and Social Sciences DataLab, Project 22209), POR Lisboa (LISBOA-01-0145-FEDER-007722 and Social Sciences DataLab, Project 22209) and POR Norte (Social Sciences DataLab, Project 22209).

1. Introduction

„Within our mandate, the ECB is ready to do whatever it takes to preserve the euro. And believe me, it will be enough.“ After ECB President Mario Draghi made this statement during the Global Investment Conference on the 26th of July, 2012, financial markets rallied. The three words “whatever it takes” marked a historical turning point for the euro-zone. Back then, Europe was in the depth of the sovereign debt crisis: Several euro-zone countries, among them Greece, Spain, Ireland and Portugal, accumulated debt, uncertain if they were able to repay it; rating companies downgraded the countries’ government bonds, yields were inversed resulting in high interest rates and the survival of the euro as a currency was threatened. In the hours following Draghi’s “whatever it takes” speech, Spanish and Italian bond yields sharply fell: The Spanish 2-year bond dropped by 74 basis points while Italy’s 2-year bond even dropped by 89 basis points. The Euro strengthened against the dollar and European stock markets jumped (Financial Times, 2012).

The “whatever it takes” speech and its immediate effects on financial markets underline the power of central bank’s monetary policy communication and the importance of the tool of forward guidance, that was introduced one year later. Forward guidance refers to the central bank’s communication to the public about its future monetary policy course and its outlook and perceptions about the economy’s development. It mitigates monetary policy surprises and thus market volatility following monetary policy announcements. It puts emphasis on the link between monetary policy and financial markets and the need to accurately measure monetary policy effects, also in context of financial markets. Measuring monetary policy has become difficult, especially since the use of unconventional monetary policy actions. Prior popular econometric approaches such as vector autoregressive models (VAR) with recursive identification schemes show up limitations in their application and also impose strong economic

restrictions. Due to the simultaneous change of financial variables, VAR analysis is also limited in estimating the effects of monetary policy on financial markets.

Precisely for this reason, this paper employs a standard monetary VAR with a mix of financial and economic variables and uses high-frequency data in an external instrument approach to identify unexpected monetary policy shocks. The goal of this paper is to provide further evidence of the effects of monetary policy in the Euro Area as this field needs to be further researched. Moreover, through the use of high-frequency data, this paper aims to estimate a more accurate response to monetary policy in the Euro Area and to infer results that are more closer to reality.

This paper is structured as follows: In chapter 2, a review of the strand of literature investigating monetary policy transmission and the literature focussing on high-frequency methodologies and external instrument identification is given. Chapter 3 outlines the empirical framework used and explains the procedure with its underlying assumptions. A discussion about the data used for the estimation is in chapter 4 and the results and robustness checks are presented in chapter 5 and 6.

2. Literature Review

Monetary policy is an important tool to stabilize a country's economy, provide liquidity and foster economic growth. Recent years brought attention to central bank's actions all around the world and emphasised its impact. As this paper deals with measuring monetary policy effects (2.1) offers an introduction to traditional monetary policy analysis and discusses its limitations. In (2.2) a relatively new approach to analyse monetary policy effects is presented and (2.3) focusses on the transmission mechanisms in the Euro Area.

2.1 Traditional Monetary Policy Analysis

Monetary policy not only affects the economy but also bases its policy decisions on the state of the economy. There is an intrinsic link between monetary policy, financial markets and

economic activity – a simultaneity problem which needs to be solved in order to interpret the effects of monetary policy. Since the publication of Sims (1980) in which he presents VAR models as solution and alternative approach for empirical macroeconomics, VAR analysis became a popular econometric approach to measure the effects of monetary policy shocks and analyse how monetary policy affects macroeconomic variables. However, VAR analysis serves to study the transmission mechanisms of monetary policy rather than serving evidence-based recommendations for ideal monetary policy decisions. Identification of the structural innovations is an important step in the VAR based approach and empirical literature shows discrepancies in inference and how policy shocks affect the economy due to different identification schemes and the inherent restrictions/assumptions.

In the following influential papers using VAR analysis and their findings of different channels of monetary transmission are presented. The empirical research on monetary policy transmission is huge – especially on the U.S. economy – and paints a picture on the response of economic variables of different sectors to a monetary policy shock. Influential papers in traditional money shock analysis, such as Bernanke and Blinder (1992) use structural VAR analysis, to analyse how a monetary policy shock transmits to banks activities and finds that a tight shock to the federal funds rate, hence an increase, has an impact on the selling-off bank securities in the short-run and the decline of loans in the long-run. Evidently bank loans are an important component in monetary policy transmission. Christiano, Eichenbaum & Evans (1996) examine the transmission effects on economic activity and show that a contractionary monetary policy leads to a decline of real GDP, retail sales, corporate profits and non-corporate profits and an increase in unemployment. In a different paper, Eichenbaum and Evans (1995) investigate the effects of U.S. monetary policy on exchange rates and find that that a tightening of monetary policy leads to an appreciation of the U.S. nominal and real exchange rates.

These papers are only a small excerpt of the widely researched field of monetary transmission mechanisms. Nevertheless, a general consensus on the qualitative effect of

monetary policy on the economy prevails. However, empirical results also show anomalies from basic concepts of monetary policy such as the price puzzle which refers to a phenomenon in empirical analysis in which an increase in the federal funds rate has led to an increase in inflation. Moreover, traditional VAR analysis shows estimations limits regarding the link between the target rate and other market interest rates and asset prices as these change simultaneously and also react to other factors (Rigobon & Sack, 2002).

Thus the VAR approach arouses criticisms and the assumptions of different identification schemes are questionable. Additionally, Rudebusch (1998) highlights four weaknesses of the VAR approach: 1) The linear specification and the fixed time component as VARs do not incorporate the change of strategy of a central bank. 2) As it assumes that the economy can be summarized by only a few variables, important factors for decision-making are excluded, and omitted variable bias can arise. 3) Misspecifications arise due to the use of revised data to which the central bank does not have access at the time of formulating the policy decisions. 4) Spurious results could arise due to too many lags in interest rate equations. Cochrane and Piazzesi (2002) also agree and state three problems of VAR analysis which are quite similar to Rudebusch: The omitted variable bias problem, the orthogonalization problem and the time-varying parameter problem. They argue that these problems can be solved by employing high-frequency identification in VAR analysis.

In conclusion, prior literature on monetary policy transmission reveals that it is challenging to model the complex relationships within an economy. VAR analysis served as a strong tool but empirical puzzles are evidence for its limitations. Thus, in the next chapter, high-frequency identification is presented as an alternative approach for measuring monetary policy effects.

2.2 High Frequency Identification in Monetary Policy Analysis

Besides the limitation of VAR analysis, the events of the global financial crisis gave rise to the close investigation of the relationship of target rates, interest rates and asset prices. The effect between monetary policy actions and asset prices is more immediate and direct. Therefore, asset

prices are important in understanding monetary policy transmission. The general consensus is that the target rate affects market interest rates which then further affect the economy. Thus, an increase in the federal funds rate leads to an increase of other market interest rates and the fall of bond prices (Kuttner, 2000).

However, empirical studies such as Kuttner (2000) find that the relationship between the target rate and other interest rates is not statistically significant due to the anticipation of target rate changes of forward-looking financial market participants. Hence, the relationship between target rates and other interest rates and asset prices must be evaluated only by the surprise component of monetary policy. Kuttner is considered a pioneer in the estimation of monetary policy effects on interest rates through high frequency data, in this case fed funds futures contracts. Since then literature using financial market data to analyse monetary policy shocks has increased. Fed funds futures data are used as a proxy for market expectations and thus, the effects of unexpected monetary policy actions or surprises can be independently measured (Kuttner, 2000). High-frequency data is collected by an event-study approach meaning the data is collected around the periods of policy changes, e.g. the day of FOMC policy announcements. Henceforth, the changes in data is only driven by the policy shock (Rigobon & Sack, 2002). Faust, Swanson, and Wright (2004) develop this approach further by using high-frequency data for structural identification of a standard monetary policy VAR and find that contractionary monetary policy results in a stronger decline of GDP. Contrary to the estimation results of Christiano, Eichenbaum, and Evans (1998), the price puzzle does not appear. Bernanke and Kuttner (2004) investigate the relationship of monetary policy and stock prices and find that a contractionary monetary policy shock leads to a decline in stock prices. Even though in recent years the analysis of monetary policy and its reaction to financial markets became more popular, there is still little evidence about the effects on corporate bond spreads. In standard economic models a tightening of monetary policy leads to a tightening of financial conditions and thus, an increase in the corporate bond spread. Furthermore, Gürkaynak and Sack (2004) make use

of an event-study approach with high-frequency data and evaluate the effects of monetary policy on asset prices via a two-factor model, which emphasizes the importance of forward guidance. Forward guidance was especially important during the financial crisis when interest rates were already at the zero lower bound, it was the only way how central banks could influence financial markets. Gertler and Karadi (2013) use high-frequency data along with an external instrument approach to identify a mixed VAR of financial and economic variables. They confirm that high-frequency identification yields results consistent with economic theory; a contractionary monetary policy shock leads to a decline in economic activity and a tightening of financial conditions.

In conclusion, the emergence of high-frequency data in applied monetary policy analysis as well as recent papers suggest, that limitations of traditional VAR analysis can be dissolved. Therefore, this paper's approach builds on the approach used in Gertler & Karadi (2013).

2.3 Monetary Transmission in the Euro Area

The previous sections showed that monetary policy has been a focus of macroeconomic research – especially the U.S. economy has been thoroughly studied. In contrast, there is some uncertainty when it comes to measuring monetary policy in the Euro Area as it is a complex construct of different national institutions with heterogenous domestic data. There are still various aspects of monetary policy and transmission, which need further investigative research. Angeloni et al. (2003) summarize facts about monetary transmission mechanisms in the Euro Area and find, that also in the Euro Area a tightening of monetary policy leads to a decline of output and inflation in all countries. Especially investment changes influence the fall in output in the Euro Area, rather than a change in consumption behaviour. Additionally, the Monetary Transmission Networks confirm that the interest rate channel is the most important channel in the Euro Area and empirical evidence confirms the existence of a credit channel (European Central Bank, 2011). Furthermore, general patterns in transmission mechanisms apply not only for the Euro Area as a whole but also to the country-level. Nevertheless, selected transmission

mechanisms can have stronger or weaker effects depending on the individual country. Recent literature has focussed on the transmission effects of unconventional monetary policy such as the effects of the asset purchase programme of the ECB or the impact of interest rates at the zero lower bound. Elbourne, Ji, and Duijndam (2018) investigate the effects of unconventional expansionary monetary policy and find, that the effects on output and inflation are relatively small: output increases slightly while the effects on inflation are statistically insignificant.

However, when studying the country-level effects, they find large differences across countries, e.g. output effects in crisis countries are smaller. Corsetti, Duarte, and Mann (2018) pick up this topic and study the heterogeneity of the transmission mechanism within the Euro Area. Following Gertler and Karadi (2013) and Gürkaynak and Sack (2004), they employ a dynamic factor model with external instrument identification to investigate the monetary policy shocks in the Euro Area. Their results for the Euro Area consequently show, that a contractionary monetary policy shock does not have a significant impact on the harmonized index of consumer prices (HICP) but leads to a significant fall in consumer prices, which is in line with theory. Moreover, GDP and consumption fall as well as imports, exports and investments – whereas these time series react stronger. Furthermore, the contractionary monetary policy shock causes government spending to increase and unemployment to rise while wages fall. Corsetti, Duarte, and Mann (2018) also investigate the housing market and confirm the economic theory that tight monetary policy leads to more expensive mortgages and thus to less demand for houses, which consequently leads to a fall in real estate prices. Jarociński and Karadi (2018) investigate the effects of US as well as Euro Area monetary policy with high frequency data as identification method among other identification schemes. They confirm that a contractionary monetary policy shock in standard theory leads to a fall in the Euro Stoxx50 Index.

The most recent paper contributing to the strand of literature on external instruments is Altavilla et al. (2019), who measure the effects of Euro Area monetary policy on different

classes of asset prices by using a FAVAR model. Through their event study database, they are able to extract three different factors of policy surprises – Target, Timing and Forward Guidance and evaluate the reaction of asset prices to these different kind of surprises.

This paper aims to provide insights in the transmission mechanisms of monetary policy in the Euro Area. It contributes to the strand of literature, which focusses on employing new econometric approaches such as the use of high-frequency financial data and VAR analysis with instruments such as in Gertler and Karadi (2013). Also, it is inspired by Altavilla et al. (2019) concluding note to further research the effects of monetary policy in the Euro Area. Additionally, this paper's contribution lies in the use of Python. Prior authors have used Matlab, which is a prominent software in economics. However, Python is becoming the most popular programming language and convinces through its generic and transparent code. In contrast to Matlab, it is a free software and the corresponding jupyter notebooks can be used as an open source, which facilitates the spread of knowledge. This paper's jupyter notebooks aim to set a base for further macroeconometric analysis in Python.

3. Empirical Framework

The methodology of this paper focusses on using traditional VAR analysis combined with an external instrument identification scheme to analyse the effects of monetary policy. The high-frequency instrument is used with the objective to separate unexpected monetary policy shocks from expected monetary policy shocks due to market expectations.

The structural VAR usually follows the general set-up

$$AY_t = \sum_{j=1}^{p} C_j Y_{t-j} + \epsilon_t \,,$$

where ϵ_t are the structural shocks. By multiplying each side with A^{-1}, we obtain the reduced form VAR:

$$Y_t = \sum_{j=1}^{p} B_j Y_{t-j} + u_t \,,$$

where u_t are the reduced form residuals and $B_j = A^{-1}C_j$. The reduced form VAR can be estimated from the data via ordinary least squares, whereas the structural shocks are unobserved. However, we assume that there is a linear relationship between the structural shocks and the VAR innovations as is shown in:

$$u_t = H\epsilon_t \,,$$

with $H = A^{-1}$, thus $\epsilon_t = H^{-1}u_t$. Moreover, we assume that the structural model is invertible and stationary and that the structural shocks are serially and mutually uncorrelated such that:

(1) $E[\epsilon_t] = 0$

(2) $\Sigma_{\epsilon\epsilon} = E[\epsilon_t \epsilon_t'] = I$

(3) $E[\epsilon_t \epsilon_s] = 0$, for $t \neq s$

(4) $E[u_t u_t'] = H\Sigma_{\epsilon\epsilon}H' = HH'$.

As the objective is to find the effect of monetary policy shocks and analyse the impulse response functions we need to recover the matrix H. However, the assumptions from the structural shocks only provide $(N+1)N/2$ moment conditions, thus we need further restrictions to recover H. These will be provided by using an external instrument identification scheme.

3.1 Identification via External Instruments

Contrary to microeconometrics, in which instrumental variables are used to mitigate omitted variable bias, an external instrument in macroeconomics is used outside of the VAR and serves as a proxy for the target shock, as ϵ_t is unknown. This approach was pioneered by Stock & Watson (2012), Mertens & Ravn (2013), and further developed by Gertler & Karadi (2013). However, literature shows that there a several strategies incorporating external instruments as an identification scheme; the basic set-up will be explained in the following:

Let $\epsilon_t^{(1)}$ be the policy structural shock and $\epsilon_t^{(2)}$ be a non-policy structural shock. For a variable z_t to be a valid instrument for the policy shock it has to obey the following two conditions:

(1) $E\left[z_t \epsilon_t^{(1)}\right] = \alpha \neq 0 \Rightarrow$ relevance criterium

(2) $E\left[z_t \epsilon_t^{(2)}\right] = 0 \Rightarrow$ exogeneity criterium

The relevance criterium says that the target shock is correlated with the instrument, while the exogeneity criterium states that the instrument is uncorrelated with all other structural shocks. Set in context, the instrument is only correlated with the policy shock and uncorrelated with the non-policy shocks. The choice of an instrument can follow several approaches. This paper will focus on the high-frequency approach.

The objective is to find the effect of the monetary policy shock, thus we partition the structural shocks such that $\epsilon_t = \left(\epsilon_t^{(1)\prime}, \epsilon_t^{(2)\prime}\right)'$, where $\epsilon_t^{(1)}$ refers to the target shock and $\epsilon_t^{(2)}$ to all other (non-policy) structural shocks. We then further partition H in such a way that, $H = \left(H^{(1)}, H^{(2)}\right)$, with the objective to estimate $H^{(1)}$- the matrix, which responds to the target shocks – in order to recover the impulse response functions. By further partitioning the matrix H, we obtain

$$\begin{pmatrix} u_t^{(1)} \\ u_t^{(2)} \end{pmatrix} = \begin{pmatrix} H^{(1,1)} & H^{(1,2)} \\ H^{(2,1)} & H^{(2,2)} \end{pmatrix} \begin{pmatrix} \epsilon_t^{(1)} \\ \epsilon_t^{(2)} \end{pmatrix},$$

$$\Rightarrow u_t^{(1)} = H^{(1,1)}\epsilon_t^{(1)} + H^{(1,2)}\epsilon_t^{(2)}$$

$$\Rightarrow u_t^{(2)} = H^{(2,1)}\epsilon_t^{(1)} + H^{(2,2)}\epsilon_t^{(2)}$$

with the objective to identify the first column $H^{(1)} = \left(H^{(1,1)}, H^{(2,1)}\right)$. By taking the instrument conditions and the partitioned matrix, we obtain $E\left[z_t u_t^{(1)}\right] = z_t\left(H^{(1,1)}\epsilon^{(1)} + H^{(1,2)}\epsilon^{(2)}\right)$ and

$E\left[z_t u_t^{(2)}\right] = z_t\left(H^{(2,1)}\epsilon^{(1)} + H^{(2,2)}\epsilon^{(2)}\right)$, which simplifies to: $E\left[z_t u_t^{(1)}\right] = \alpha H^{(1,1)}$ and $E\left[z_t u_t^{(2)}\right] = \alpha H^{(2,1)}$. Together, the equations yield:

$$H^{(2,1)}H^{(1,1)^{-1}} = E\left[z_t u_t^{(1)}\right]^{-1} E\left[z_t u_t^{(2)}\right],$$

which can be then estimated from the data by first estimating the reduced VAR and then using the two-stage-least-square approach (2SLS). In the first stage $u_t^{(1)}$ is regressed on z_t and in the second stage, the non-policy residuals are regressed on the predicted value of the policy residuals (from stage 1), which yields a consistent estimator $E\left[z_t u_t^{(1)}\right]^{-1} E\left[z_t u_t^{(2)}\right]$, which equals to $H^{(2,1)}H^{(1,1)^{-1}}$. Hence, the fitted value of the regression of the instrument helps to identify the structural shocks. Via the variance-covariance matrix the relevant columns of matrix H can be estimated (Appendix A) (Jentsch & Lunsford, 2016; Lakdawala, 2016; Lunsford, 2016). Once $H^{(1)}$ is estimated, we can proceed with computing the IRFs. As we only have one policy shock, this econometric framework is sufficient to identify the coefficients up to sign and scale (Dias, Daniel A. & Duarte, 2019). Mertens and Ravn (2013) provide an extended identification strategy in case of more than one target shock – a detailed explanation would push the limits of this thesis.

3.2 Instrumental Variable

The following section gives an overview on how high-frequency data can be collected and used as an instrument. Firstly, different methods from previous papers are presented before the method of this paper is thoroughly described.

Following the event-study methodology of Gürkaynak & Sack (2004) and based on the instrument approach of Gertler & Karadi (2013), high-frequency data is collected around the FOMC policy announcements and used as an instrument for VAR identification in the latter case. By selecting a narrow time window of 30 minutes, monetary policy surprises can be isolated and the changes in high-frequency data circulating the FOMC meetings can serve as a

proxy for the monetary policy shock. However, the ECB communicates its policy decisions in a different manner. The governing council is the decision-making body of the ECB and meets every two weeks in the ECB headquarters in Frankfurt Germany. Within the fortnightly meetings, the governing council discusses the economic and monetary developments within the Euro Area and assesses possible risks to price stability. Based on this analysis it forms its monetary policy decisions. The monetary policy decisions are announced every six weeks and are published at 13:45 CET as a summarized list of the changes without any underlying explanations. The president of the ECB explains these decisions in the press conference, which follows at 14:30 CET and lasts about an hour. The president announces the monetary policy decisions followed by explaining the reasons, which motivated the governing council to come to the specific monetary policy decisions and also gives insights into the further economic developments expected by the ECB. Afterwards, journalists have the opportunity to ask the president questions within a Q&A session (ECB, n.d.-a)(ECB, n.d.-b).

Therefore, high-frequency data for the Euro Area can be either collected in separate windows following Altavilla et al. (2019), who collect data from several classes of assets in their monetary policy event-study database around the so-called policy decision window, the press conference release window and when not distinguishing between them, the whole monetary policy decision window. They use a time frame of 10 minutes prior to the event and 10 minutes afterwards to compute changes in the intraday data, e.g. overnight index swaps and German bond rate changes at different maturities. Corsetti, Duarte & Mann (2018) do not distinguish between the two release windows and choose to observe a 6-hour window from 13:00 to 19:00 CET, as these times correspond to the closing of the stock exchange market in London and Tokyo. Through this technique they construct an external intraday series and overcome the problem of missing data. Hafemann & Tillmann (2017) use changes in the 10-year German government bond on meeting days; An increase in the German bond, hence a positive surprise, is associated with a tightening of monetary policy.

All in all, the high-frequency literature focussing on the economy in the Euro Area experiments with several instruments, while the research conducted for the U.S. economy, usually uses fed funds futures to identify unexpected monetary policy shocks. Lloyd (2018) contributes to this topic with his research on the usefulness of OIS rates as instruments: He comes to the conclusion that among others, the 1-24-month Eurozone OIS rates can be used as a measure for market expectations and thus are an applicable instrument for monetary policy analysis.

Therefore, I will use the changes in the 1-year OIS rate as an instrument for monetary policy shocks. Due to restricted data availability on Thomas Reuter's Eikon I will use the EA-MPD database from Altavilla et al. (2019) for the instrument data collection. The dataset includes Eurozone OIS rates at different maturities and several other asset prices in the three different time windows mentioned above. I will use the 1-year OIS rate of the monetary event window in which the change in the median quote from 13.25-13:35 and from 15:40-15:50 is collected, hence before the press release and after the press conference.

In the following, the validity of the changes in the 1-year-OIS rate around ECB monetary decision days is shown with two illustrative examples. Figure 1 shows the instrument's minutely development on the 12th September, 2019. This date is out of the sample range, however it serves as an example for the minutely development of the Eurozone OIS rate on meeting days. From noon onwards the OIS rate gradually increases. At 13:45 the monetary policy decisions are released; Mario Draghi announced a huge stimulus package including a cut of the deposit facility rate by 10 basis points and the revival of the asset purchase programme (APP) for an unlimited amount of time. Although, markets expected a rate cut and even saw the relaunch of the APP as very likely due to earlier comments of ECB representatives, the OIS rate displayed a negative surprise component due to the new information of an unlimited time frame for the APP and thus shifted slightly downwards, which is associated with a loosening of monetary policy. During the press conference, Mario Draghi announced that the ECB expects

from now to leave the key interest rates at their current level. Moreover, he alerted governments and stressed the importance of fiscal policy to avert a new crisis: With fiscal policy in place, the monetary policy stance would not need to be that expansionary. During the introductory statement, the OIS rate slightly drops but then increases again with the Q&A session (ECB, 2019; Financial Times, 2019a, 2019b; Szalay, 2019). The movement of the Eurozone OIS rate emphasises the importance of the subsequent press conference and the direct reaction of financial market participants to the ECB's economic outlooks.

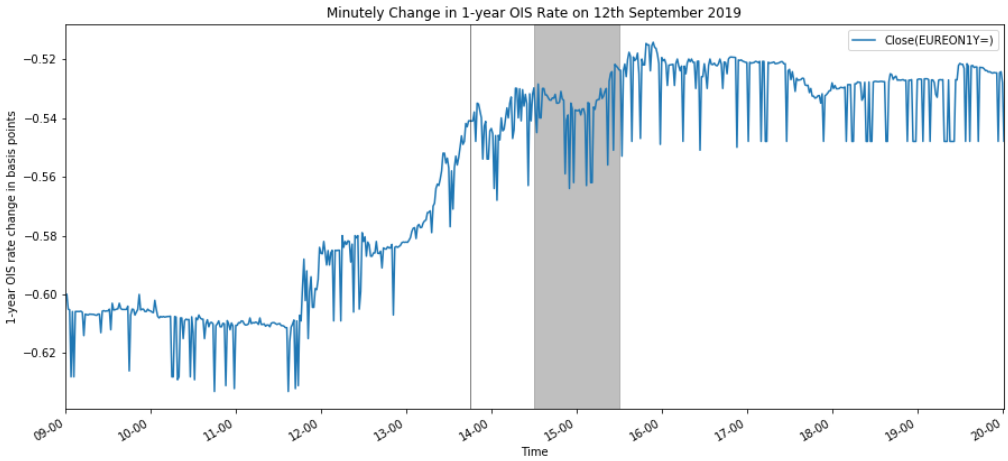


Figure 1: Minutely change in 1-year OIS rate on 12th September, 2019.

The EA-MPD dataset includes all dates of ECB governing council meetings: In total there are 264 monetary policy announcements from 7th January, 1999 til 6th June, 2019. Hereinafter, a monthly VAR will be estimated, thus the instrument time series will be transformed into a monthly series, by cumulating the changes within a given month.

Figure 2 shows the development of the monthly instrument time series. The time series fluctuates around zero but displays some large positive and also negative spikes associated with huge surprise components. The largest spikes are found in the early years of the ECB, around 2001, and then during the financial crisis and the subsequent years. The recent years til 2017 do not show any large fluctuations at all. One of the large spikes occurred in August 2001, the

1-year OIS rate changed with a value of -16.7. In that month, the ECB announced to cut all three interest rates by 0.25 basis points as inflationary pressures seemed to ease. The corresponding significant change, mirrors the market's surprise.

This short descriptive analysis of the instrument emphasises its close relationship to the target rate and illustrates the volatility of financial markets.

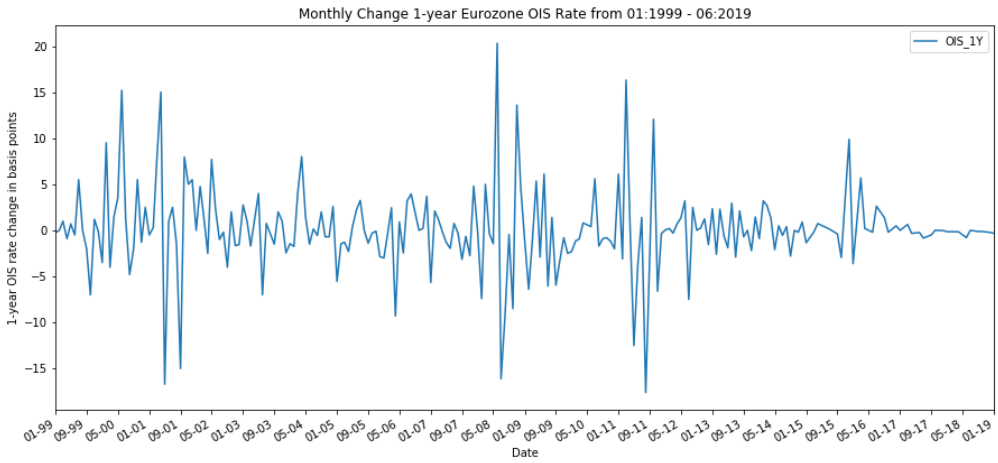


Figure 2: Monthly change in the 1-year Eurozone OIS rate.

4. Data and Estimation

This chapter describes the data used for the baseline VAR and it explains all relevant steps in the estimation as basis for the discussion of the results in chapter 5.

4.1 The baseline VAR

The baseline VAR includes 5 monthly time-series, which will be a mix of financial and macroeconomic variables over the sample period, from 01:1999 to 06:2019. The start date of the sample corresponds to the introduction of the Euro – and henceforth the start of the ECB's monetary activities and influence in the Euro Area – and dates up to this year. The sample includes the financial crisis in 2008 and the subsequent introduction of unconventional monetary policy, such as the APP in 2015, and the current zero lower bound phase. The data is obtained from the ECB Statistical Warehouse, Bloomberg, FRED and Thomas Reuters Eikon.

As a measure of output and price level, the log of industrial production and the log of the harmonized index of consumer prices (HICP) are included in the baseline VAR. The industrial production index excludes construction and is seasonally and working day adjusted. For the HICP, the overall index is used, seasonally and working day adjusted. For both indices the reference year is 2015. As financial variables, the Euro Stoxx50 Index and a corporate bond spread are included to incorporate financial market conditions and credit risk. The Euro Stoxx50 Index is a weighted blue-chip index and incorporates 50 stocks from 11 Eurozone countries. As BBB corporate bond spread, the Euro High Yield Option-Adjusted Spread is used analogously to (Jarociński & Karadi, 2018). Additionally, the 1-year German Government Bill Index serves as the policy indicator. Both units are in percent.

The baseline VAR is estimated in levels via ordinary least squares with two lags according to the minimum value of -31.62, of the Bayes information criterion (BIC) in the lag length test. According to Dolado & Lütkepohl (1996), estimating a VAR with an order of integration equal to $I(1)$ with a number of lags $d \geq 2$ still provides asymptotically normal t-ratios, thus inference can be made when estimating a VAR in levels; inference on impulse response functions also remains valid if the VAR has a lag length greater than one (Luetkepohl, 2011). Additionally, the autocorrelation function of the residuals will be consulted to confirm the choice of lag length. All autocorrelation plots (Figure 5), which are reported with 95% confidence bands, lie within the confidence bands indicating that there is no serial autocorrelation within the residuals. Except for the corporate bond spread which shows one significant spike at lag five. However, in accordance with the BIC, an optimal lag length of two is confirmed, as all other information criteria, such as the AIC, FPE and HQIC, recommend a lag length of 2 as well. Moreover, the rest of the variables do not exhibit any serial autocorrelation and a higher order would only inflate the coefficients.

Prior literature in the proxy SVAR method emphasises the importance of a fitting instrument and policy indicator choice. Gertler & Karadi (2013) try several different

combinations of policy indicator and instrument within a regression exercise while Hafemann & Tillmann (2017) use an event-study-regression to study this issue. However, within the monthly VAR both use the first stage F-statistic to confirm the adequacy of the instrument as proxy for the policy indicator. Stock, Wright, & Yogo (2002) recommend a high threshold for the first-stage F-statistic, with a value higher than 10, to reject a weak instrument problem and assure reliable inference. After estimating the reduced VAR, 2SLS estimation is performed with the reduced VAR residuals and the instrument. In the first stage regression of the policy residuals on a constant and the instrument, I obtain a F-statistic with the value of 33.08 (Figure 6), which is quite above the threshold of 10 and thus assures the adequacy of the instrument. With an accurate instrument choice, the estimation continues by calculating the 2SLS estimator β_{IV}, and identifying the matrix $H^{(1)}$ to further estimate the impulse response functions with 90% bootstrapping confidence bands (Appendix A).

5. Results

The impulse response functions span over a horizon of 40 periods. Figure 3 shows the impulse response functions for a 100 basis points contractionary monetary policy shock. This shock leads the 1-year German Government Bond to increase for two periods until it is 1.35 percentage points above its previous value, that is, relative to the situation when there was no policy shock. After that, the response of the government bond decreases and starts to slowly decay towards zero. After 25 periods the response reaches zero and the response turns negative for the subsequent periods. However, the response is only significant for the first twelve periods. Notably, the results suffer from the price puzzle: a statistically significant increase of the price level. However, the output puzzle does not appear: industrial production has a negative response in the initial period and drops by 0.01% against its initial value. The response increases until it turns positive, decays towards zero and turns negative again after 14 periods; however it is insignificant for almost all periods. Contrary to theory that a monetary tightening leads to

a decline of stock indices, the Euro Stoxx50 Index has a positive reaction to a tightening of monetary policy and increases up to 0.17%. The response gets smaller after two periods and goes towards zero (until period 15), and then turns negative; but is only significant in the short-run (first 5 periods). Moreover, the BBB corporate bond spread initially decreases with a tightening of monetary policy indicating improving financial conditions. The response to the monetary policy shock turns positive after 7 periods and then slowly decays to zero; but the impulse response function is only significant in the short-run (till period 4) and again in the medium-run (period 10-30).

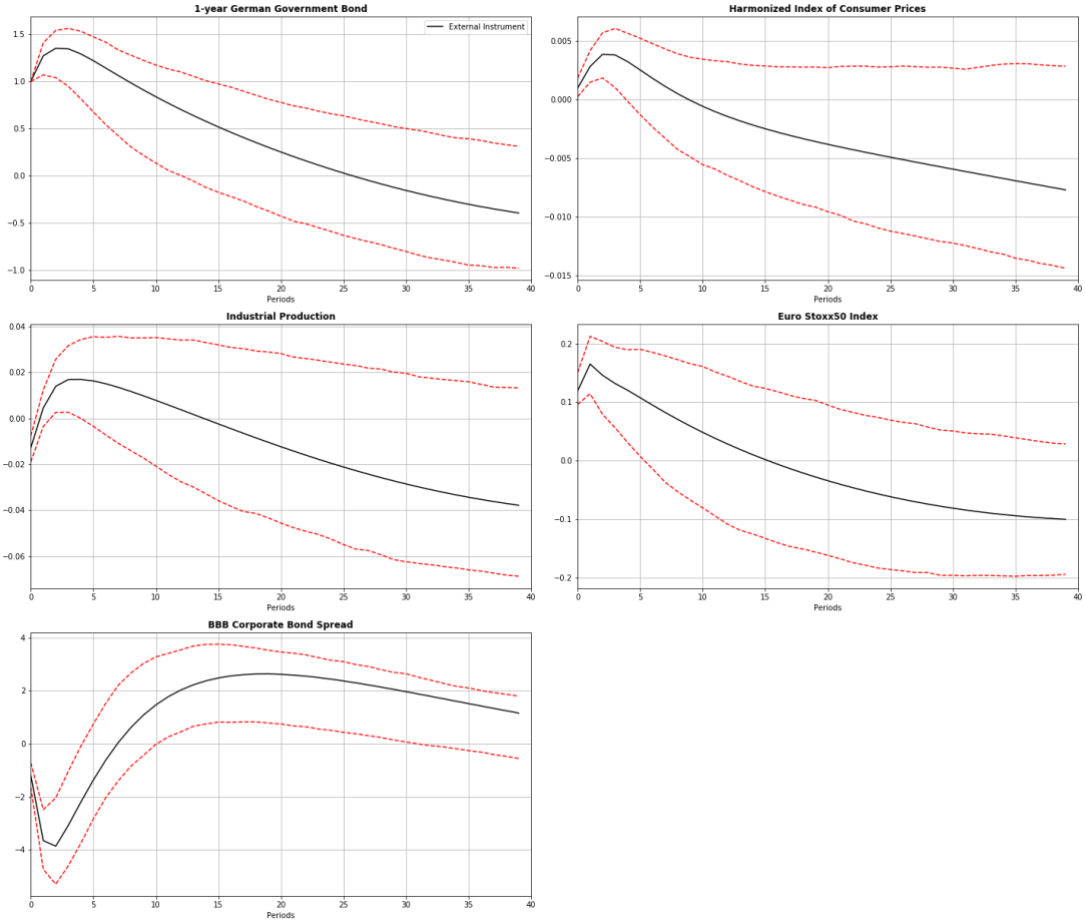


Figure 3: Impulse response to a monetary policy shock.

In comparison to the findings of other papers that applied high frequency identification, this paper's results slightly differ. Corsetti, Duarte and Mann (2018)'s results do not show the prize puzzle due to the adoption of the high-frequency identification and a dynamic factor model, which incorporates information about several price indices in the economy. As the price puzzle

is a common problem in VAR analysis, this result might suggest that using solely the HICP as indicator for inflation in the model is not enough to model the price dynamics in the economy correctly. Moreover, the results for the stock index and the corporate bond spread do not align with economic theory but are consistent with the results of Jarociński and Karadi (2018), who find as well that stock prices increase and corporate bond spreads decrease following a tightening monetary policy shock with standard high-frequency identification. Jarociński and Karadi (2018) delve deeper into the cause of a positive response of the stock prices and conclude that a positive co-movement of monetary policy and stock prices are due to a central bank information shock, which biases the results of standard high-frequency identification which in turn should indicate a negative co-movement according to theory.

These central bank information shocks are transmitted through central banks' announcements, in which private information of the central bank and their beliefs of the development of the economy are revealed to the public; a positive central bank information shock is associated with good news about the economy. Hence, it would translate into a situation in which the central bank tightens monetary policy but would communicate a positive perception of the economic outlook in order to counteract the effects of a monetary policy tightening on the economy. The positive co-movement of monetary policy and stock prices would also result in improving financial conditions, consistent with a decline of the corporate bond spread. Although Altavilla et al. (2019) obtain results about the stock prices according to standard economic theory, they re-estimate their variables in a small exercise finding evidence of information shocks, so called Delphic surprises. Thus, this paper's results could be decomposed further to confirm the theory of different macroeconomic effects of so-called Delphic (information shock) and Odyssean (monetary policy) surprises.

Furthermore, it should be noticed, that the model is estimating the response of the aggregate Euro Area. However, heterogeneity issues within the member countries could also lead to results not complying to standard economic theory, as the transmission channel with weaker

member countries could be partially broken. Hafemann and Tillmann (2017) attribute the different responses of stock prices on the country-level to an impaired monetary policy transmission in structurally aggrieved countries.

6. Robustness Checks

To confirm the structural validity of the results, several robustness checks are carried out. All results are reported with 90% bootstrapping confidence bands.

6.1 Cholesky Identification

As alternative identification scheme, the Cholesky identification is used to compare the results. The Cholesky ordering is the following: Log of industrial production, log of HICP, the policy indicator, and then followed by the log of the Euro Stoxx50 Index and last, the BBB corporate bond spread. This ordering assumes that the stock index and the corporate bond spread react contemporaneously to monetary policy, while inflation and output only react within a period. Figure 7 shows the impulse response functions for the Cholesky identification. Strikingly, the results obtained via the Cholesky identification not only display the price puzzle but also the output puzzle underlining the more accurate measuring of monetary policy effects through high-frequency identification. Industrial production has a positive reaction over the whole time horizon but is only significant for the first ten periods. The reaction of consumer prices to a monetary policy shock is also positive but quickly decays to zero. The impulse response is only significant until period 4. The curve for the Euro Stoxx50 Index behaves similar to the external instrument case but the effect of a tightening of monetary policy is stronger: While in the external instrument case the 1% shock to monetary policy led the Euro Stoxx50 Index initially to increase by 0.12% from its previous value, the Cholesky case shows an increase of 0.46% in the initial period. Moreover, the response is positive at all times and does not reach zero in the entire time horizon; the response is significant for 20 periods. Also, the effect on the corporate bond spread is similar to the external instrument case but again stronger in the initial period.

6.2 The Post-2008 Sample

To investigate the effects of unconventional monetary policy the sample is split and only the period after the financial crisis will be analysed: from 10:2008 till the end of the sample in 06:2019. October 2008 is marked as the beginning of the crisis sample as the ECB sharply cut their interest rates by 50 bp that month, in coordination with the FED as the effects of the financial crisis intensified. The VAR is estimated with only one lag according to the Bayesian information criterion. In the crisis sample, the first stage regression F-test has a value of 7.99, indicating a weak-instrument problem. However, the impulse response functions in Figure 8 do not show any puzzles. The HICP decreases with a 1% monetary policy shock but the response is insignificant; industrial production also has a negative reaction with a monetary policy tightening but the response is only significant for the first two periods. However, the response for the Euro Stoxx50 Index and the BBB Corporate Bond spread still not obey to economic theory: The Euro Stoxx50 Index reacts positively but then the response get constantly smaller and goes to zero. The impulse response mirroring the financial conditions behaves similar to the external instrument case, the reactions are only prolonged.

6.3 Alternative Instrument

The robustness of the results is also verified by employing an alternative instrument, the Eurozone overnight index swap, but with a shorter maturity of six months. According to the first stage F-statistic, which has a value of 29.3, the 6-month OIS rate is also an adequate instrument. The impulse response functions with the 6-month rate do not differ greatly from the ones with the 1-year rate, underlining the validity of the method.

7. Conclusion

In using a vector autoregressive model with external instrument identification through high-frequency data, this paper analysed the effects of monetary policy shocks on economic activity and financial markets in the aggregate Euro Area. This approach not only makes it feasible to

jointly analyse economic as well as financial variables but also to evaluate unexpected monetary policy shocks separately and incorporate market expectations. The findings show that monetary policy is transmitted through all channels under consideration – real economic activity, the credit channel and stock markets. Moreover, the results confirm that monetary policy analysis through high-frequency identification shows more accurate and precise effects of monetary policy shocks on economic activity. However, the findings also suggest that the effects of monetary policy on financial markets need to be decomposed further and separating expected from unexpected monetary policy shocks is not enough to fully grasp the effects on financial markets. In alignment with Jarociński & Karadi (2018), the findings emphasise that high-frequency identification can be biased and that monetary policy shocks to financial markets should be further separated, e.g. into shocks to the central bank's policy instrument and shocks to their communication.

However, the results also need to be evaluated within the limits of this paper. For one thing the data used for the instrument was based on the EA-MPD database due to license restrictions with other financial databases. Therefore, the robustness of the instrument could only be analysed by using a shorter maturity. A different financial intraday time series could lead to different results. On the other hand, the VAR only included five variables which aimed at modelling economic as well as financial markets activity. A model including more variables might be able to capture the transmission mechanisms more accurately.

The findings as well as the limitations offer new directions of research, which could be especially interesting for the unconventional monetary policy phase. Additionally, this paper used Python as programming language and if further research would also resort to more open software packages, a benchmark library for empirical macroeconomics could be built in Python.

Bibliography

Altavilla, C., Brugnolini, L., Gürkaynak, R. S., Motto, R., & Ragusa, G. (2019). Measuring euro area monetary policy. *European Central Bank*, (2281), 57.

Angeloni, I., Kashyap, A. K., Mojon, B., & Terlizzese, D. (2003). *Monetary Transmission in the Euro Area: Does the Interest Rate Channel Explain it All?* (Working Paper No. 9984). https://doi.org/10.3386/w9984

Bernanke, B., & Blinder, A. (1992). The Federal Funds Rate and the Channels of Monetary Transmission. *American Economic Review*, *82*, 57.

Bernanke, B., & Kuttner, K. N. (2004). *What Explains the Stock Market's Reaction to Federal Reserve Policy?* 56.

Christiano, L., Eichenbaum, M., & Evans, C. (1998). *Monetary Policy Shocks: What Have We Learned and to What End?* (No. w6400; p. w6400). https://doi.org/10.3386/w6400

Christiano, L. J., Eichenbaum, M., & Evans, C. (1996). The Effects of Monetary Policy Shocks: Evidence from the Flow of Funds. *The Review of Economics and Statistics*, *78*(1), 16–34. https://doi.org/10.2307/2109845

Cochrane, J. H., & Piazzesi, M. (2002). *The Fed and Interest Rates: A High-Frequency Identification* (Working Paper No. 8839). https://doi.org/10.3386/w8839

Corsetti, G., Duarte, J. B., & Mann, S. (2018, February). One money, many markets: A factor model approach to monetary policy in the Euro Area with high-frequency identification [Monograph]. Retrieved September 26, 2019, from http://www.centreformacroeconomics.ac.uk/Discussion-Papers/Home.aspx

Dias, Daniel A., & Duarte, J. B. (2019). Monetary Policy, Housing Rents and Inflation Dynamics. *International Finance Discussion Paper*, *2019*(1248), 1–26. https://doi.org/10.17016/IFDP.2019.1248

Dolado, J. J., & Lütkepohl, H. (1996). Making wald tests work for cointegrated VAR systems. *Econometric Reviews*, *15*(4), 369–386. https://doi.org/10.1080/07474939608800362

ECB. (2019, December 9). Introductory statement to the press conference (with Q&A). Retrieved November 5, 2019, from European Central Bank website: https://www.ecb.europa.eu/press/pressconf/2019/html/ecb.is190912~658eb51d68.en.html

ECB, E. C. B. (n.d.-a). Governing Council. Retrieved September 26, 2019, from European Central Bank website: https://www.ecb.europa.eu/ecb/orga/decisions/govc/html/index.en.html

ECB, E. C. B. (n.d.-b). Governing Council decisions. Retrieved September 26, 2019, from European Central Bank website: https://www.ecb.europa.eu/press/govcdec/html/index.en.html

Eichenbaum, M., & Evans, C. L. (1995). Some Empirical Evidence on the Effects of Shocks to Monetary Policy on Exchange Rates. *The Quarterly Journal of Economics*, *110*(4), 975–1009. https://doi.org/10.2307/2946646

Elbourne, A., Ji, K., & Duijndam, S. (2018). *The effects of unconventional monetary policy in the euro area* (No. 371). Retrieved from CPB Netherlands Bureau for Economic Policy Analysis website: https://ideas.repec.org/p/cpb/discus/371.html

European Central Bank. (2011). *The monetary policy of the ECB*. Frankfurt am Main: European Central Bank.

Faust, J., Swanson, E. T., & Wright, J. H. (2004). Identifying VARS based on high frequency futures data. *Journal of Monetary Economics*, *51*(6), 1107–1131. https://doi.org/10.1016/j.jmoneco.2003.11.001

Financial Times. (2012, July 26). ECB 'ready to do whatever it takes.' Retrieved November 19, 2019, from Financial Times website: https://www.ft.com/content/6ce6b2c2-d713-11e1-8e7d-00144feabdc0

Financial Times. (2019a, November 9). Draghi must deliver his parting shot of stimulus. Retrieved November 5, 2019, from Financial Times website: https://www.ft.com/content/0e375bbc-cfdd-11e9-99a4-b5ded7a7fe3f

Financial Times. (2019b, December 9). ECB cuts rates and tells governments to act | Financial Times. Retrieved November 5, 2019, from Financial Times website: https://www.ft.com/content/9b2c29c0-d53d-11e9-a0bd-ab8ec6435630

Gertler, M., & Karadi, P. (2013). *Monetary Policy Surprises, Credit Costs and Economic Activity*. 44.

Gürkaynak, R. S., & Sack, B. (2004). *Do Actions Speak Louder Than Words? * The Response of Asset Prices to Monetary Policy Actions and Statements*. 43.

Hafemann, L., & Tillmann, P. (2017). *The Aggregate and Country-Specific Effectiveness of ECB Policy: Evidence from an External Instruments (VAR) Approach*. 40.

Jarociński, M., & Karadi, P. (2018). Deconstructing monetary policy surprises: The role of information shocks. *European Central Bank*, (2133), 64.

Jentsch, C., & Lunsford, K. G. (2016, July). *Proxy SVARs: Asymptotic theory, bootstrap inference, and the effects of income tax changes in the United States*. https://doi.org/10.26509/frbc-wp-201619

Kuttner, K. (2000). *Monetary Policy Surprises and Interest Rates: Evidence from the Fed Funds Futures Market*. 24.

Lakdawala, A. (2016). *Decomposing the Effects of Monetary Policy Using an External Instruments SVAR*. 47.

Lloyd, S. (2018). *Overnight Index Swap Market-Based Measures of Monetary Policy Expectations* (SSRN Scholarly Paper No. ID 3135278). Retrieved from Social Science Research Network website: https://papers.ssrn.com/abstract=3135278

Luetkepohl, H. (2011). *Vector autoregressive models* [Working Paper]. Retrieved from http://cadmus.eui.eu//handle/1814/19354

Lunsford, K. G. (2016). *Identifying Structural VARs with a Proxy Variable and a Test for a Weak Proxy*. 35.

Mertens, K., & Ravn, M. O. (2013). The Dynamic Effects of Personal and Corporate Income Tax Changes in the United States. *American Economic Review*, *103*(4), 1212–1247. https://doi.org/10.1257/aer.103.4.1212

Rigobon, R., & Sack, B. P. (2002). *The Impact of Monetary Policy on Asset Prices* (Working Paper No. 8794). https://doi.org/10.3386/w8794

Rudebusch, G. D. (1998). Do Measures of Monetary Policy in a Var Make Sense? *International Economic Review*, *39*(4), 907–931. https://doi.org/10.2307/2527344

Sims, C. A. (1980). Macroeconomics and Reality. *Econometrica*, *48*(1), 1. https://doi.org/10.2307/1912017

Stock, J. H., & Watson, M. W. (2012). *Disentangling the Channels of the 2007-2009 Recession* (Working Paper No. 18094). https://doi.org/10.3386/w18094

Stock, J. H., Wright, J. H., & Yogo, M. (2002). A Survey of Weak Instruments and Weak Identification in Generalized Method of Moments. *Journal of Business & Economic Statistics*, *20*(4), 518–529. https://doi.org/10.1198/073500102288618658

Szalay, E. (2019, September 11). Investors bet ECB's next steps will strengthen euro. Retrieved November 5, 2019, from Financial Times website: https://www.ft.com/content/7fae3356-d3e8-11e9-a0bd-ab8ec6435630

Appendix A: Estimation & Impulse Response Function

A.1 Estimation

The estimation and identification of the structural VAR follows the following procedure: First the reduced form VAR is estimated trough the data. Then the two-stage least square approach is used to yield a result for $H^{(2,1)}H^{(1,1)^{-1}}$. From this the relevant columns of matrix H can be identified.

In the first stage of the 2SLS approach, the policy residuals $u_t^{(1)}$ are regressed on the instrument z_t, such that $u_t^{(1)} = \gamma z_t + \eta_t$, the fitted value of $\hat{u}_t^{(1)}$ is then used in the second stage in which the non-policy residuals are regressed on the policy residuals, $u_t^{(2)} = \beta_{2SLS}\, \hat{u}_t^{(1)} + v_t$. The two-stage-least-square approach then yields then a consistent estimator $\beta_{2SLS} = E\left[z_t u_t^{(1)}\right]^{-1} E\left[z_t u_t^{(2)}\right]$, which in turn yields an estimate of the relationship $H^{(2,1)}H^{(1,1)^{-1}}$. Additionally, the first stage F-statistic provides a weak instrument test. Following Stock & Watson a F-statistic > 10 rejects the presence of a weak instrument.

With the aid of the reduced VAR variance-covariance matrix, $E[u_t u_t'] = \Sigma = \begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}$, the relevant columns of the matrix H can be estimated. From the structural VAR assumptions we know that $E[u_t u_t'] = H\Sigma_{\epsilon\epsilon}H' = HH'$. Hence, we can express the columns of Σ by expressions of the matrix H:

$$\Sigma_{11} = E\left[u_t^{(1)} u_t^{(1)\prime}\right] = (H^{(1,1)}\epsilon^{(1)} + H^{(1,2)}\epsilon^{(2)}) * (H^{(1,1)}\epsilon^{(1)} + H^{(1,2)}\epsilon^{(2)})'$$

$$= H^{(1,1)}H^{(1,1)\prime} + H^{(1,2)}H^{(1,2)\prime}, \quad (1)$$

$$\Sigma_{21} = H^{(2,1)}H^{(1,1)\prime} + H^{(2,2)}H^{(1,2)\prime}, \quad (2)$$

$$\Sigma_{22} = H^{(2,1)}H^{(2,1)\prime} + H^{(2,2)}H^{(2,2)\prime}, \quad (3)$$

From (1) we know that

$H^{(1,1)^2} = \Sigma_{11} - H^{(1,2)}H^{(1,2)\prime}$ and thus $H^{(1,1)} = \sqrt{\Sigma_{11} - H^{(1,2)}H^{(1,2)\prime}}$.

From (1) to (3) we obtain an expression

$$Q = \Sigma_{22} - H^{(2,1)}H^{(1,1)^{-1}} * \Sigma'_{21} - \Sigma_{21} * H^{(2,1)}H^{(1,1)^{-1}\prime} + H^{(2,1)}H^{(1,1)^{-1}} * \Sigma_{11} * H^{(2,1)}H^{(1,1)^{-1}\prime},$$

which can be estimated from the data. Then an expression for $H^{(1,2)}H^{(1,2)}$ can be estimated:

$$H^{(1,2)}H^{(1,2)} = (\Sigma_{21} - H^{(2,1)}H^{(1,1)^{-1}} * \Sigma'_{11}) * Q^{-1} * (\Sigma_{21} - H^{(2,1)}H^{(1,1)^{-1}} * \Sigma_{11}),$$

And finally $H^{(1,1)^2} = \Sigma_{11} - H^{(1,2)}H^{(1,2)'}$ can also be estimated.

Now the relevant matrix $H^{(1)}$ is identified and can be used for the impulse response analysis.

A.2 Impulse Response Functions

The structural model is given by

$$Y_t = \sum_{j=1}^{p} B_j Y_{t-j} + H\epsilon_t.$$

The impulse refers to the change in the innovations of the model. By inverting the VAR into a moving average representation, the impulse response can be studied.

The model can be rewritten in lag notation to $B(L)Y_t = H\epsilon_t$, where $B(L) = I - B_1L - \cdots - B_pL^p$. And it can be turned into the moving average representation:

$$Y_t = \sum_{j=1}^{\infty} \varphi_j u_{t-j},$$

where $\varphi_j = \varphi(L) = B(L)^{-1}$. The impulse response to a monetary policy shock is then given by $\frac{\partial Y_t}{\partial \epsilon_1} = B(L)^{-1}H$. The coefficient matrix, can directly be estimated via the reduced form. The matrix H was identified previously.

A.3 Recursive Residual-Based Wild Bootstrapping

The wild bootstrapping algorithm was chosen for this estimation as Gertler & Karadi (2013) and Mertens & Ravn (2013) also use it for their proxy SVAR inference. For the bootstrapping procedure a random variable is needed with mean zero and variance 1. In proxy SVAR literature the Rademacher distribution is used commonly, where the draws are either 1 or -1, with a

probability of 0.5. The residuals and the instrument are multiplied with the Rademacher distribution to obtain u^*, Z^*. The regressors are left at their sample value, however, the response variable is resampled based on u^*, Z^*. Thus a new Y^* is produced. From the new bootstrap sample, the reduced residuals are estimated and structural residuals identified as in the usual proxy SVAR procedure. This algorithm is repeated several times, in this case a 1000 times. Then the confidence intervals can be computed, which are robust against conditional heteroskedacity and allow better inference under this design.

Appendix B: Data Sources

Variable	Adj.	Source
1-year German Government Bill	-	Bloomberg (GDBR1 Index)
Harmonized Index of Consumer Prices – Overall Index	Seasonally adjusted and working day adjusted	ECB Statistical Warehouse (Series-Key: ICP.M.U2.Y.000000.3.INX)
Industrial Production Index: Total excluding construction	Seasonally adjusted and working day adjusted	ECB Statistical Warehouse (Series-Key: STS.M.I8.Y.PROD.NS0020.4.000)
ICE BofAML Euro High Yield Index Option-Adjusted Spread	Not seasonally adjusted	FRED (BAMLHE00EHYIOAS)
Euro Stoxx50 Index	-	Bloomberg (SX5E Index)
1-year Overnight Index Swap	-	EAMPD-Database (Altavilla et al., 2019), based on EUREON1Y= from Thomas Reuters Eikon

Appendix C: Figures

	AIC	BIC	FPE	HQIC
0	-13.14	-13.06	1.962e-06	-13.11
1	-31.81	-31.34	1.528e-14	-31.62
2	-32.49*	-31.62*	7.789e-15*	-32.14*
3	-32.40	-31.14	8.479e-15	-31.89
4	-32.37	-30.72	8.798e-15	-31.70
5	-32.36	-30.31	8.892e-15	-31.53
6	-32.32	-29.88	9.316e-15	-31.33
7	-32.30	-29.47	9.532e-15	-31.16
8	-32.22	-29.00	1.038e-14	-30.92
9	-32.14	-28.52	1.139e-14	-30.68
10	-32.08	-28.07	1.224e-14	-30.46
11	-32.06	-27.66	1.267e-14	-30.28
12	-32.02	-27.22	1.351e-14	-30.08

Figure 4: Results of lag length test.

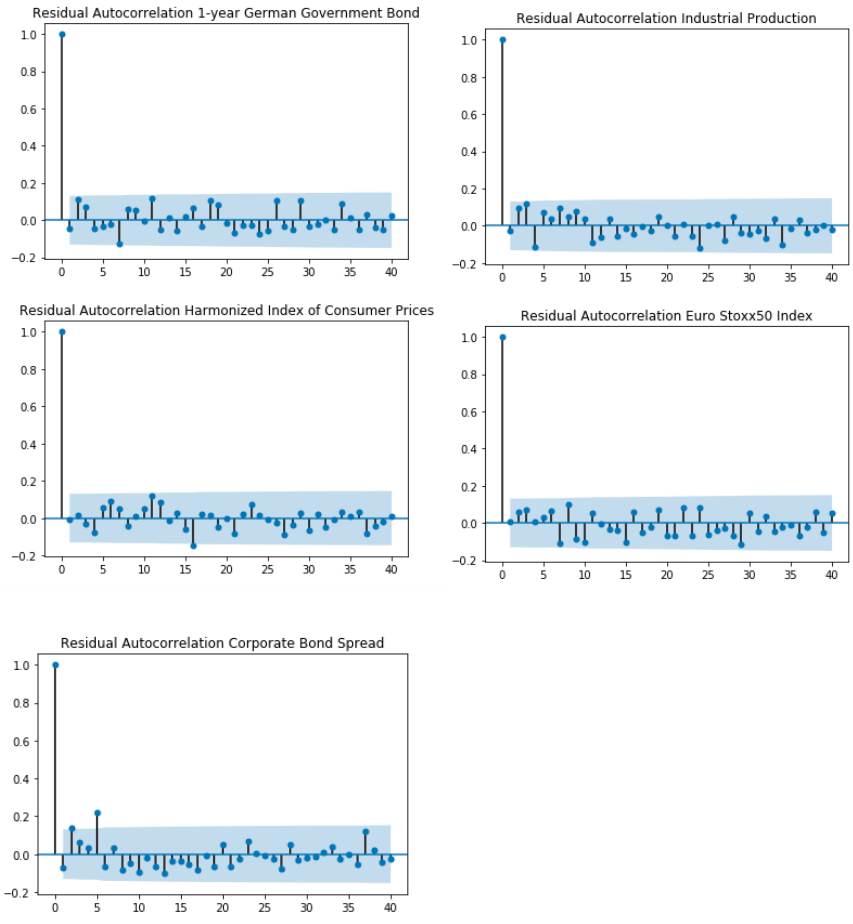


Figure 5: Residual autocorrelation plots.

OLS Regression Results

Dep. Variable:	1ygovbondindx	R-squared:	0.130
Model:	OLS	Adj. R-squared:	0.126
Method:	Least Squares	F-statistic:	33.08
Date:	Thu, 19 Dec 2019	Prob (F-statistic):	2.91e-08
Time:	22:12:38	Log-Likelihood:	108.68
No. Observations:	224	AIC:	-213.4
Df Residuals:	222	BIC:	-206.5
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0008	0.010	-0.077	0.938	-0.020	0.019
OIS_1Y	0.0125	0.002	5.752	0.000	0.008	0.017

Omnibus:	27.529	Durbin-Watson:	2.041
Prob(Omnibus):	0.000	Jarque-Bera (JB):	63.671
Skew:	-0.562	Prob(JB):	1.49e-14
Kurtosis:	5.358	Cond. No.	4.61

Figure 6:First stage regression results.

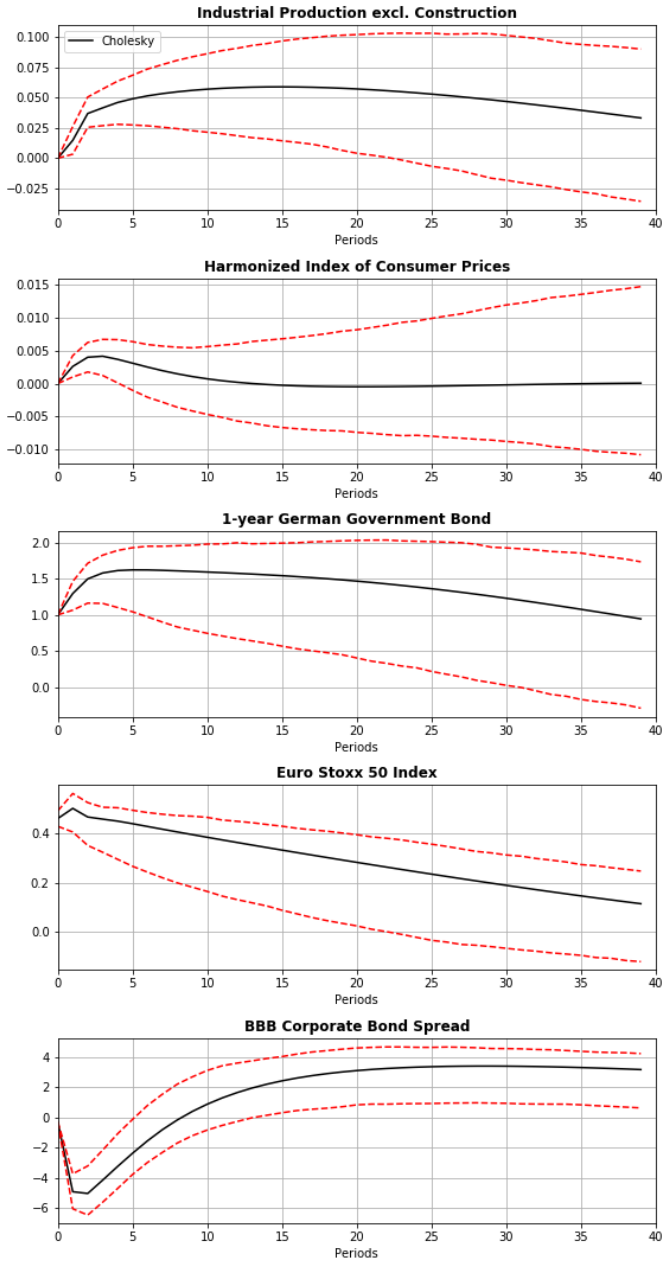


Figure 7: Impulse response to a monetary policy shock with Cholesky identification.

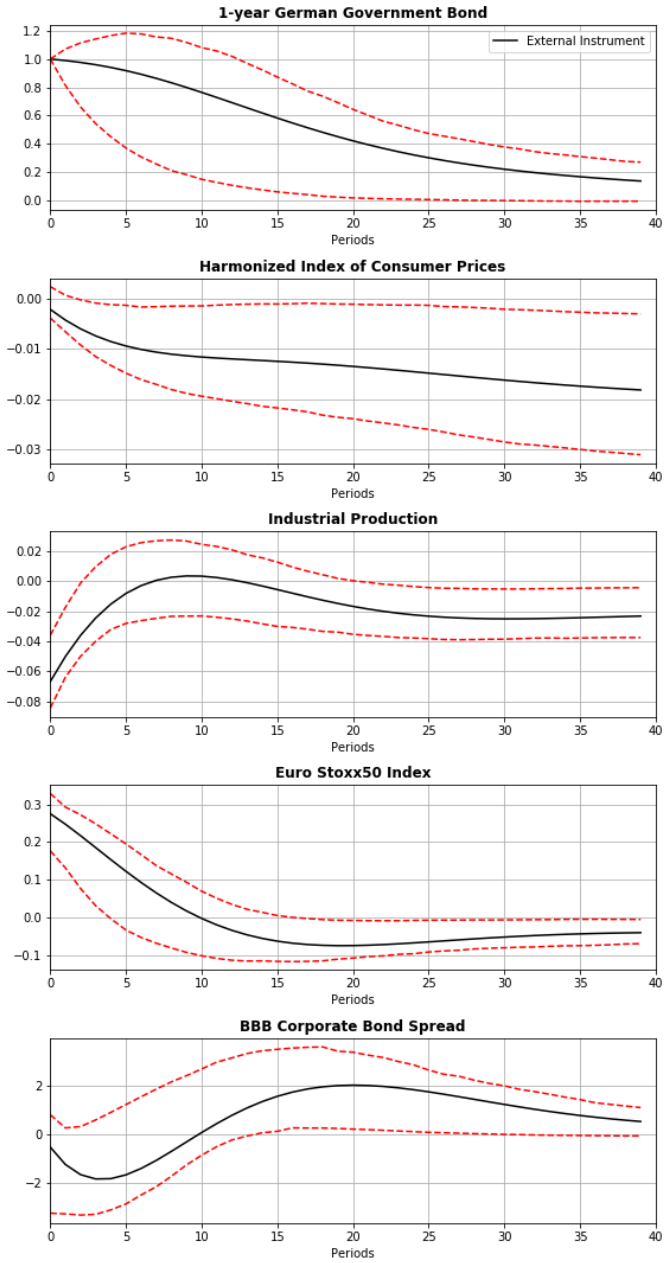


Figure 8: Impulse response for the crisis sample from (10:2008 – 06:2019).

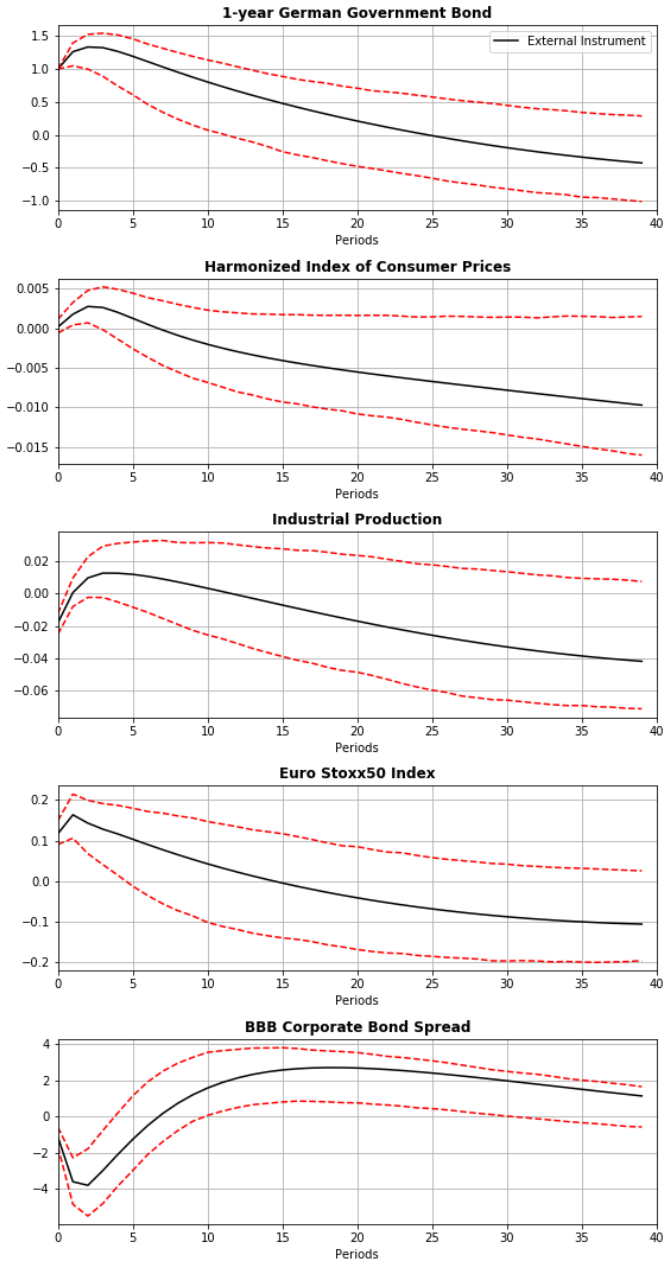


Figure 9: Impulse response for alternative instrument.

Appendix D: Python Jupyter Notebook

1. Data Transformation

December 30, 2019

*this jupyter notebook was created specifically for the pdf version presented in the appendix of the work project document, not all tables and graphics are displayed.

In [19]:
```
#libraries
import numpy as np
import pandas as pd
import statsmodels.api as sm
from statsmodels.tsa.api import VAR
import matplotlib.pyplot as plt
%matplotlib inline
import datetime
import warnings
warnings.filterwarnings("ignore")
```

0.1 Part I: VAR Data

In [20]:
```
#load raw data
data = pd.read_excel('VAR_data.xlsx').sort_values(by = 'Date', ascending = True)
data1 = data.set_index('Date')
data1.head()
```

Out[20]:

Date	Year	Month	IP excl. Construction	HICP	eurostoxx50	bbb_spread	1ygovbondindx
1999-01-01	1999	1	89.5	74.08	3547.15	7.410476	2.883
1999-02-01	1999	2	88.4	74.10	3484.24	7.526190	3.016
1999-03-01	1999	3	88.6	74.25	3559.86	7.347826	2.871
1999-04-01	1999	4	89.0	74.52	3757.87	6.990909	2.616
1999-05-01	1999	5	89.2	74.50	3629.46	6.830000	2.667

In [21]:
```
#create seperate dataframe for the dates
date = data1[['Year', 'Month']]
```

```
In [22]: #load instrument data
         instrument = pd.read_excel('EA_instrument.xlsx')
         instrument1 = instrument.set_index('Date')
         instrument1.head()
```

Out[22]:

Date	Year	Month	OIS_1Y
1999-01-07	1999	1	-0.25
1999-01-21	1999	1	0.00
1999-02-18	1999	2	0.00
1999-03-04	1999	3	0.00
1999-03-18	1999	3	1.00

```
In [23]: #cumulate the changes in the 1-year OIS rate by month
         instrument2 = instrument1.groupby(['Year', 'Month']).sum().reset_index()
         instrument2['Days'] = np.ones((len(instrument2['Month'])))

In [24]: #create dataframe with datetime index
         instrument2['Date'] = pd.to_datetime((instrument2.Year*10000+instrument2.Month*100+ins
         instrument3 = instrument2.drop(columns = ['Year', 'Month', 'Days']).set_index('Date')
         instrument3.head()
```

Out[24]:

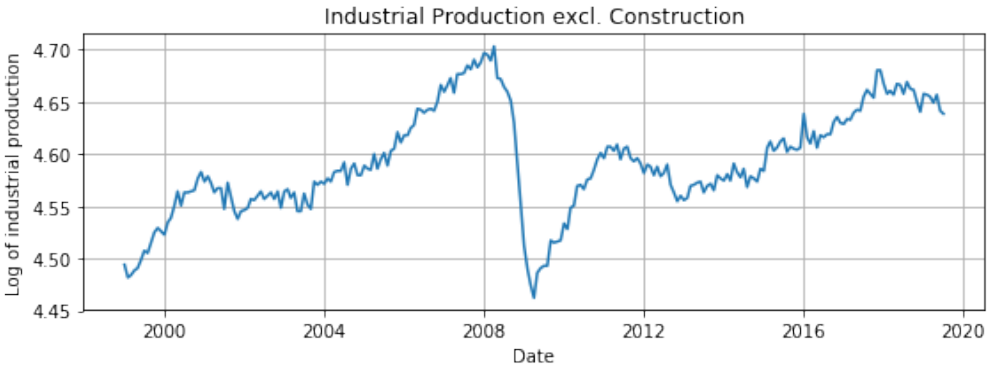
Date	OIS_1Y
1999-01-01	-0.25
1999-02-01	0.00
1999-03-01	1.00
1999-04-01	-0.90
1999-05-01	0.70

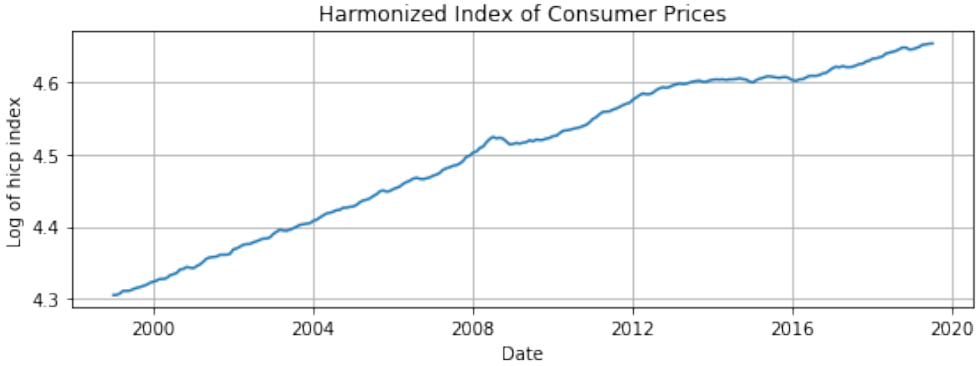
```
In [25]: #seperate time series for transformation
         ip = pd.DataFrame(data1['IP excl. Construction'])
         hicp = pd.DataFrame(data1['HICP'])
         stoxx = pd.DataFrame(data1['eurostoxx50'])
         bbb_spread = pd.DataFrame(data1['bbb_spread'])
         gov_bond = pd.DataFrame(data1['1ygovbondindx'])

In [26]: #transform time series with logs
         log_ip = np.log(ip)
         log_ip.rename(columns = {'IP excl. Construction': 'IP'}, inplace = True)
         log_hicp = np.log(hicp)
         log_stoxx = np.log(stoxx)
```

0.2 Part II: Visualise VAR Data

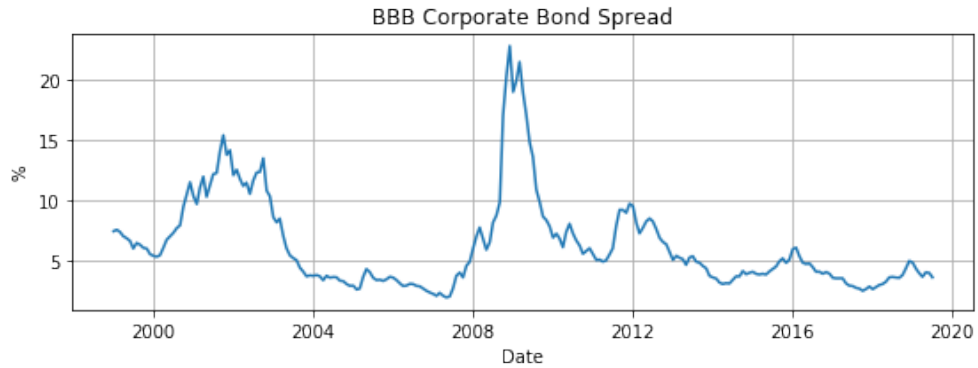
```
In [27]: plt.figure(figsize=(8,15))

         plt.subplot(511)
         plt.plot(log_ip)
```

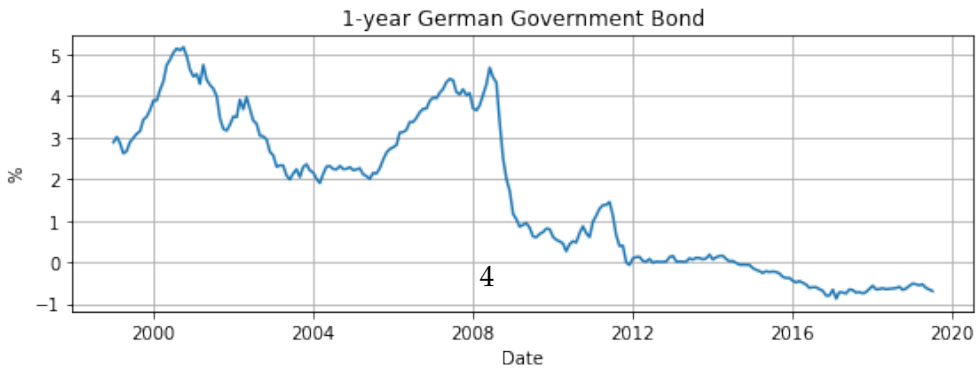
```
plt.title('Industrial Production excl. Construction')
plt.xlabel('Date')
plt.ylabel('Log of industrial production')
plt.grid()

plt.subplot(512)
plt.plot(log_hicp)
plt.title('Harmonized Index of Consumer Prices')
plt.xlabel('Date')
plt.ylabel('Log of hicp index')
plt.grid()

plt.subplot(513)
plt.plot(log_stoxx)
plt.title('Euro Stoxx50 Index')
plt.xlabel('Date')
plt.ylabel('Log of Euro Stoxx50 Index')
plt.grid()

plt.subplot(514)
plt.plot(bbb_spread)
plt.title('BBB Corporate Bond Spread')
plt.xlabel('Date')
plt.ylabel('%')
plt.grid()

plt.subplot(515)
plt.plot(gov_bond)
plt.title('1-year German Government Bond')
plt.xlabel('Date')
plt.ylabel('%')
plt.grid()

plt.tight_layout()
```


Industrial Production excl. Construction
Log of industrial production
4.70
4.65
4.60
4.55
4.50
4.45
2000
2004
2008
2012
2016
2020
Date


Harmonized Index of Consumer Prices
Log of hicp index
4.6
4.5
4.4
4.3
2000
2004
2008
2012
2016
2020
Date


Euro Stoxx50 Index
Log of Euro Stoxx50 Index
8.6
8.4
8.2
8.0
7.8
7.6
2000
2004
2008
2012
2016
2020
Date


BBB Corporate Bond Spread
%
20
15
10
5
2000
2004
2008
2012
2016
2020
Date


1-year German Government Bond
%
5
4
3
2
1
0
−1
2000
2004
2008
2012
2016
2020
Date

0.3 Part III: Prepare VAR Data in Levels

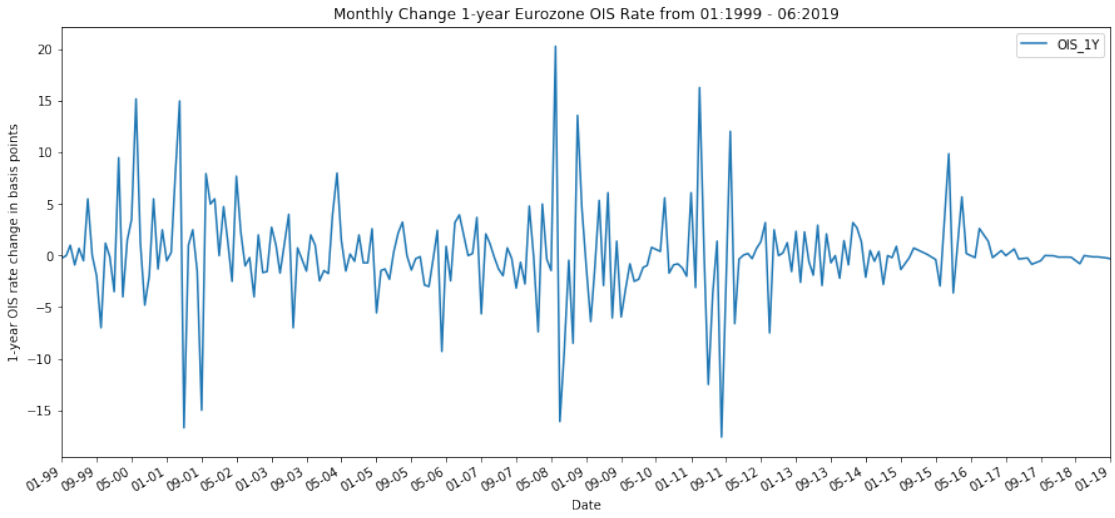
In [28]:
```
#join transformed time series in one dataframe
df = date.join([gov_bond,log_hicp, log_ip ], how = 'outer')
df2 = df.join([log_stoxx, bbb_spread], how = 'outer')

#join instrument
data_final = pd.merge(df2, instrument3, on = ['Date'], how = 'outer').dropna()
data_final.head() #final data for subsequent estimation
```

Out[28]:

Date	Year	Month	1ygovbondindx	HICP	IP	eurostoxx50	bbb_spread	OIS_1Y
1999-01-01	1999	1	2.883	4.305146	4.494239	8.173900	7.410476	-0.25
1999-02-01	1999	2	3.016	4.305416	4.481872	8.156005	7.526190	0.00
1999-03-01	1999	3	2.871	4.307438	4.484132	8.177476	7.347826	1.00
1999-04-01	1999	4	2.616	4.311068	4.488636	8.231608	6.990909	-0.90
1999-05-01	1999	5	2.667	4.310799	4.490881	8.196839	6.830000	0.70

0.4 Part IV: Example 1-year Eurozone OIS Rate

In [29]:
```
#create plot of monthly OIS rate
import matplotlib.dates as mdates

plt.ioff()
fig, ax = plt.subplots(figsize=(15,7))
instrument3.plot(ax=ax)
ax.xaxis.set_major_locator(mdates.MonthLocator(interval = 8))
ax.xaxis.set_major_formatter(mdates.DateFormatter('%m-%y'))

plt.title('Monthly Change 1-year Eurozone OIS Rate from 01:1999 - 06:2019')
plt.xlabel('Date')
plt.ylabel('1-year OIS rate change in basis points')
plt.xlim('1999','2019')
plt.ioff()
```

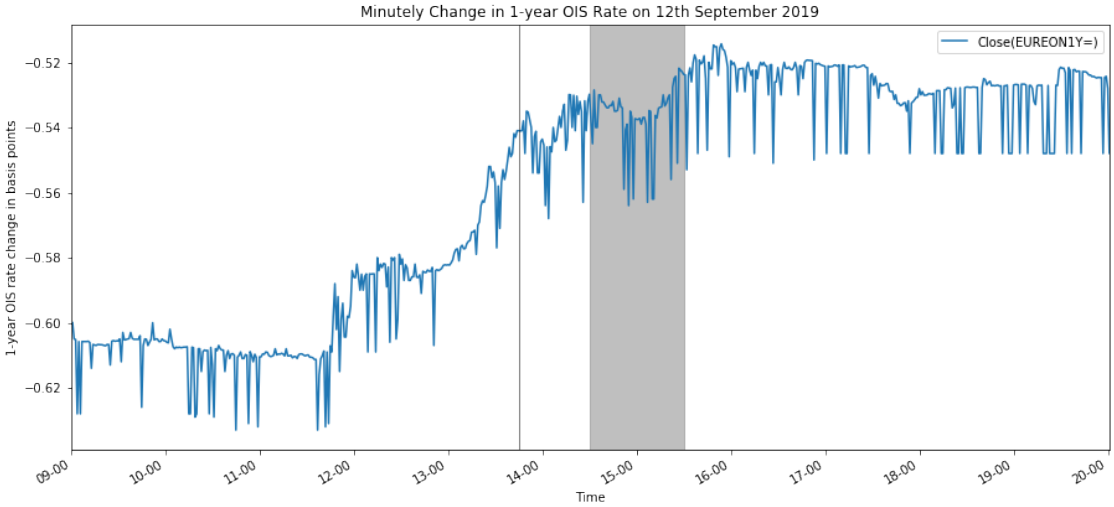


In [30]:
```
#read data
df = pd.read_excel('eureon1y_minutely.xlsx')

#select time range
df2 = df[(df['Date'] > '2019-09-12 09:00:00') & (df['Date'] < '2019-09-13 00:01:00')]
df2.head()
```

Out[30]:

Date	Close(EUREON1Y=)
2019-09-13 00:00:00	-0.548
2019-09-12 23:59:00	-0.548
2019-09-12 23:58:00	-0.548
2019-09-12 23:57:00	-0.548
2019-09-12 23:56:00	-0.548

In [31]:
```
#create plot
plt.ioff()
fig, ax = plt.subplots(figsize=(15,7))

df2.plot(ax=ax)

plt.title('Minutely Change in 1-year OIS Rate on 12th September 2019')
plt.xlabel('Time')
plt.ylabel('1-year OIS rate change in basis points')
plt.xlim('2019-09-12 09:00:00', '2019-09-12 20:01:00')

ax.xaxis.set_major_locator(mdates.HourLocator(interval = 1))
ax.xaxis.set_major_formatter(mdates.DateFormatter('%H-%M'))
ax.axvline(x = '2019-09-12 13:45:00', color = 'black', linewidth = 0.5)
ax.axvspan('2019-09-12 14:30:00','2019-09-12 15:30:00', color = 'grey', alpha = 0.5)
plt.ioff()
```


Minutely Change in 1-year OIS Rate on 12th September 2019
Close(EUREON1Y=)
1-year OIS rate change in basis points
−0.52
−0.54
−0.56
−0.58
−0.60
−0.62
09-00
10-00
11-00
12-00
13-00
14-00
15-00
16-00
17-00
18-00
19-00
20-00
Time

2. Baseline VAR Estimation and Identification

December 30, 2019

In [1]:
```
#libraries
import numpy as np
import pandas as pd
import statsmodels.api as sm
from statsmodels.tsa.api import VAR
import matplotlib.pyplot as plt
%matplotlib inline
import datetime
from numpy.linalg import inv
from scipy import linalg
import warnings
warnings.filterwarnings("ignore")
```

0.1 Part I: Load Data

In [2]:
```
#load data and set datetime as index
df = pd.read_excel('data_levels.xlsx')
df2 = df.set_index('Date')
df2.head()
```

Out[2]:

Date	Year	Month	1ygovbondindx	HICP	IP	eurostoxx50	bbb_spread	OIS_1Y
1999-01-01	1999	1	2.883	4.305146	4.494239	8.173900	7.410476	-0.25
1999-02-01	1999	2	3.016	4.305416	4.481872	8.156005	7.526190	0.00
1999-03-01	1999	3	2.871	4.307438	4.484132	8.177476	7.347826	1.00
1999-04-01	1999	4	2.616	4.311068	4.488636	8.231608	6.990909	-0.90
1999-05-01	1999	5	2.667	4.310799	4.490881	8.196839	6.830000	0.70

In [3]:
```
#select variables for reduced var model
X = df2[['1ygovbondindx','HICP','IP','eurostoxx50','bbb_spread']]
```

0.2 Part II: Preliminary Tests

In [4]:
```
#lag length test
import statsmodels.api as sm
from statsmodels.tsa.api import VAR

model   = VAR(X)
model.select_order(12).summary()
```

Out[4]:
```
<class 'statsmodels.iolib.table.SimpleTable'>
```

In [5]:
```
from statsmodels.graphics.tsaplots import plot_acf

results = model.fit(2, trend = 'nc')
#compute residuals
residuals = results.resid

#recheck lag selection via autocorrelation function of residuals
plt.ioff()
plot_acf(residuals[['1ygovbondindx']], lags=40, title = 'Residual Autocorrelation 1-yea
plot_acf(residuals[['HICP']], lags=40, title = 'Residual Autocorrelation Harmonized Inc
plot_acf(residuals[['IP']], lags=40, title = 'Residual Autocorrelation Industrial Produ
plot_acf(residuals[['eurostoxx50']], lags=40, title = 'Residual Autocorrelation Euro St
plot_acf(residuals[['bbb_spread']], lags=40, title = 'Residual Autocorrelation Corporat
plt.ioff()
```

0.3 Part III: Estimation Reduced VAR

In [6]:
```
#manual estimation of reduced var

#create dataframe with lags
XLAG     = pd.DataFrame()
num_lags = 2 #number of lags according to BIC
for i in range(1,num_lags+1):
    XLAG = pd.concat([XLAG,X.shift(i).add_suffix("-"+str(i))],axis=1)

#
X2       = X.iloc[num_lags:,:]
XLAG2    = XLAG.iloc[num_lags:,:]
num_vars = X2.shape[1]
num_obs  = XLAG2.shape[0]

#turn dataframe into arrays
X3       = np.array(X2)
XLAG3    = np.array(XLAG2)

#calculate beta coefficient
Bhat     = inv(XLAG3.T@XLAG3)@XLAG3.T@X3
```

```
#print beta coefficient results
col_names = list(X2.columns)
index = list(XLAG2.columns)
coefficient_matrix = pd.DataFrame(Bhat, index = index, columns = col_names)
coefficient_matrix
```

Out[6]:

	1ygovbondindx	HICP	IP	eurostoxx50	bbb_spread
1ygovbondindx-1	1.206220	0.001479	0.007181	0.059325	-1.882308
HICP-1	12.326375	1.143944	0.222482	0.041549	45.282049
IP-1	-0.772842	0.019828	0.688560	0.026600	-4.739764
eurostoxx50-1	0.183317	0.002299	0.013894	0.965809	-5.918025
bbb_spread-1	-0.018017	-0.000184	-0.003683	0.007399	0.988982
1ygovbondindx-2	-0.264382	-0.001726	-0.007373	-0.070943	2.165812
HICP-2	-13.176063	-0.153208	-0.213005	-0.190087	-40.974560
IP-2	1.290158	-0.008866	0.278898	0.145637	-0.506159
eurostoxx50-2	0.007398	-0.003173	0.000026	0.021246	6.508368
bbb_spread-2	0.014252	0.000206	0.002969	-0.006951	-0.090552

0.4 Part III: Two-Stage Least Square Regression

In [7]:
```
date = list(X2.index)

#estimate errors from reduced form VAR
res = X3 - XLAG3@Bhat
u = pd.DataFrame((X3 - XLAG3@Bhat),index = date, columns = col_names) #create datafram

#reduced error covariance matrix
#VAR.Sigma = (VAR.res'*VAR.res)/(VAR.T-VAR.n*VAR.p-1);
sigma = (u.T@u)/(num_obs - num_lags*num_vars - 1)

#partition errors
#policy residuals
res_p = u[['1ygovbondindx']]

#non-policy residuals
res_q = u[['HICP', 'IP','eurostoxx50', 'bbb_spread']]

#turn into array
u_p = np.array(res_p).reshape(len(X3),1)
u_q = np.array(res_q)

u.head() #show excerpt of residuals
```

Out[7]:

	1ygovbondindx	HICP	IP	eurostoxx50	bbb_spread
1999-03-01	-0.210170	0.000402	-0.004420	0.013827	0.038525
1999-04-01	-0.267440	0.001887	0.002309	0.069471	-0.525029
1999-05-01	0.021309	-0.002252	-0.000331	-0.012007	-0.393453
1999-06-01	0.169593	-0.001076	0.004220	0.040930	-0.226711
1999-07-01	-0.004069	-0.000314	0.004733	-0.049821	0.136371

In [8]:
```
sigma #show variance-covariance matrix
```

Out[8]:

	1ygovbondindx	HICP	IP	eurostoxx50	\
1ygovbondindx	0.026809	6.301566e-06	1.504118e-04	0.003799	
HICP	0.000006	2.640349e-06	4.292477e-07	-0.000006	
IP	0.000150	4.292477e-07	9.716275e-05	0.000057	
eurostoxx50	0.003799	-5.905515e-06	5.695121e-05	0.003037	
bbb_spread	-0.034949	-1.407716e-04	-1.159073e-03	-0.013639	

	bbb_spread
1ygovbondindx	-0.034949
HICP	-0.000141
IP	-0.001159
eurostoxx50	-0.013639
bbb_spread	0.498320

In [9]:
```
#get instrument
instrument = df2[['OIS_1Y']].iloc[num_lags:,:] #adapt data range according to lag leng
Z = np.array(instrument)
```

In [10]:
```
#2SLS
#First Stage: OLS with policy residual and instrument
#policy residual = constant + instrument

b_p = inv(Z.T@Z)@Z.T@u_p # beta coefficient for Z

#find constant
N = len(u_p)
c = np.ones(N)
c0 = np.mean(u_p)-(b_p*np.mean(Z))

#calculate fitted values for policy residual
u_p_hat = c0 + b_p*Z

print('The first stage coefficients are',c0, 'and', b_p)

#Produce table of actual and fitted values of dependent variable
actual = pd.DataFrame(u_p).rename(columns = {0: 'actual'})
fitted = pd.DataFrame(u_p_hat).rename(columns = {0: 'fitted'})
t1 = actual.join(fitted)
t1.head() #show excerpt
```

```
The first stage coefficients are [[-0.00077337]] and [[0.01247968]]
```

Out[10]:

	actual	fitted
0	-0.210170	0.011706
1	-0.267440	-0.012005
2	0.021309	0.007962

```
3  0.169593 -0.007013
4 -0.004069  0.067865
```

```
In [11]: #f-test for weak instruments
         k = 2
         T = len(Z)

         SSE = (u_p - c0 - b_p*Z).T@(u_p - c0 - b_p*Z) #sum of squared errors
         SST = (u_p - np.mean(u_p)).T@(u_p - np.mean(u_p)) #total sum of squares

         r_squared = 1 - (SSE/SST) #calculate R^2

         F_test = (r_squared/(k-1))/((1-r_squared)/(T-k)) #F-test


         print('R-squared:',r_squared)
         print('F-test:',F_test)
```

```
R-squared: [[0.12969016]]
F-test: [[33.08156933]]
```

```
In [12]: #Second stage
         #u_q = b * u_p_hat
         b_iv = inv(u_p_hat.T@u_p_hat)@u_p_hat.T@u_q #coefficients
         print(b_iv)
```

```
[[ 9.60695155e-04 -1.28505766e-02  1.18980757e-01 -1.18129020e+00]]
```

0.5 Part IV: Identification

```
In [13]: #turn reduced VAR variance-covariance matrix into array
         sig = np.array(sigma)

         #2SLS coefficient is estimate of H21iH11
         h21ih11 = b_iv.T

         #partitioning of the covariance matrix
         sig11 = sig[0][0].reshape(1,1)
         sig21 = sig[1:,0].reshape(-1,1)
         sig22 = sig[1:,1:5]

         #start by estimating Z
         Q = sig22 - h21ih11@sig21.T - sig21@(h21ih11.T) + h21ih11*sig11*h21ih11.T

         #next
         h12h12 = (sig21 - h21ih11*sig11).T@inv(Q)@(sig21 - h21ih11*sig11)
         h11h11 = sig11 - h12h12
```

```
        h11 = np.sqrt(h11h11)
        print('h11 is',h11)
        print('and h21 is', h21ih11*h11)

        #find H1, obtained estimates for h11 and h21
        H1 = np.vstack((h11, (h21ih11*h11)))
```

```
h11 is [[0.15584716]]
and h21 is [[ 1.49721613e-04]
 [-2.00272589e-03]
 [ 1.85428134e-02]
 [-1.84100725e-01]]
```

0.6 Part V: Impulse Response Function

```
In [14]: #impulse response function
         num_impulses = 40 #number of periods
         irs = np.zeros([num_lags+num_impulses, num_vars])
         irs[num_lags,:] = (H1.T/(H1[0]))

         for jj in range (1, num_impulses):
             lvars = irs[np.arange(start = num_lags + jj - 1, stop = jj - 1, step = -1),:].T
             irs[(num_lags + jj),:] = lvars.flatten(1).T@Bhat

         irs = irs[num_lags : num_lags + num_impulses,:]
         irs = irs
         irf_proxy = pd.DataFrame(irs, columns = col_names) #turn irf into dataframe
```

```
In [15]: irf_proxy.head() #show beginning of irf table
```

Out[15]:

	1ygovbondindx	HICP	IP	eurostoxx50	bbb_spread
0	1.000000	0.001470	0.135193	0.097597	-1.454463
1	1.290790	0.003846	0.060131	0.160887	-3.896407
2	1.421694	0.005773	0.067443	0.148352	-4.646878
3	1.486044	0.006745	0.087355	0.144625	-4.284896
4	1.505833	0.007245	0.088019	0.146016	-3.803853

0.7 Part VI: Bootstrapping

```
In [16]: #function for estimating reduced VAR
         def estimate(X):
             X = pd.DataFrame(X)
             XLAG     = pd.DataFrame()
             num_lags = 2

             for i in range(1,num_lags+1):
                 XLAG = pd.concat([XLAG,X.shift(i).add_suffix("-"+str(i))],axis=1)
             #
```

```
    X2       = X.iloc[num_lags:,:]
    XLAG2    = XLAG.iloc[num_lags:,:]
    num_vars = X2.shape[1]
    num_obs  = XLAG2.shape[0]
    #
    X3       = np.array(X2)
    XLAG3    = np.array(XLAG2)
    #
    Bhat     = inv(XLAG3.T@XLAG3)@XLAG3.T@X3
    res = X3 - XLAG3@Bhat
    return res, Bhat
```

In [17]:
```
#function for identification
def proxysvar (residual, instrument):
    sigma = (residual.T@residual)/(num_obs - num_lags*num_vars - 1)
    pshock = residual[:,0].reshape(-1,1)
    qshock = residual[:,1:,]
    #first stage
    b_fs = inv(instrument.T@instrument)@instrument.T@pshock
    constant = np.ones(len(pshock))
    constant = np.mean(pshock)-(b_fs*np.mean(instrument))
    #fitted value
    pshock_hat = constant + b_fs*instrument
    #second stage
    b_ss = inv(pshock_hat.T@pshock_hat)@pshock_hat.T@qshock
    #2SLS coefficient is estimate of H21iH11
    b21ib11 = b_ss.T
    #Columns of the covariance matrix
    sigma11 = sigma[0][0].reshape(1,1)
    sigma21 = sigma[1:,0].reshape(-1,1)
    sigma22 = sigma[1:,1:5]
    #start by estimating Q
    S = sigma22 - b21ib11@sigma21.T - sigma21@(b21ib11.T) + b21ib11*sigma11*b21ib11.T
    #next
    b12b12 = (sigma21 - b21ib11*sigma11).T@inv(S)@(sigma21 - b21ib11*sigma11)
    b11b11 = sigma11 - b12b12
    b11 = np.sqrt(b11b11)
    #find H1, obtained estimates for h11 and h21
    B1 = np.vstack((b11, b21ib11*b11))

    return B1
```

In [18]:
```
#function for impulse response
def impulse(Bhat, B1):
    num_impulses = 40
    irs = np.zeros([num_lags+num_impulses, num_vars])
    irs[num_lags,:] = (B1.T/(B1[0]))
```

```
        for jj in range (1, num_impulses):
            lvars = irs[np.arange(start = num_lags + jj - 1, stop = jj - 1, step = -1),:]
            irs[(num_lags + jj),:] = lvars.flatten(1).T@Bhat

        irsb = irs[num_lags : num_lags + num_impulses,:]
        irf_proxy = pd.DataFrame(irsb, columns = col_names)
        return irsb
```

In [19]:
```
#simulate new data and repeat in loop

jj = 0
nboot = 1000 #number of repetitions
imp  = np.zeros([(num_impulses*num_vars),nboot])

for rep in range(1,nboot):
    jj=jj+1

    rr = (1-2*(np.random.random(len(X2)) > 0.5)).reshape(-1,1) #Rademacher distributi

    resb = (res*(rr@np.ones((1, num_vars)))).T #u*
    Zb = np.vstack(((Z[0:num_lags,:]),(rr*np.ones((1,1))*Z))) #Z*
    varsb = np.zeros((len(X), num_vars))
    #initial condition
    varsb[0:num_lags,:] = X[0:num_lags]

    for j in range ((num_lags), (num_lags + len(X2))):
        lvars = (varsb[np.arange(start = j - 1, stop = j-num_lags-1, step = -1)]).T #
        varsb[j,:] = lvars.flatten(1).T@Bhat[0:(num_lags*num_vars),:] + resb[:,j-num_l

    var_j = estimate(varsb) #obtain fitted value for u_star
    H_j = proxysvar(var_j[0], Zb[num_lags:,:])
    irf_j = impulse(var_j[1], H_j)
    irf_x = np.array(irf_j)
    imp[:,jj-1]  = np.reshape(irf_x,(num_impulses*(num_vars),1)).flatten()
```

In [20]:
```
#create confidence bands
imp    = imp.reshape(num_impulses,num_vars,nboot)
imp    = np.sort(imp,axis=2)#
impci  = imp[:,:,[np.int(0.05*nboot),np.int(0.95*nboot)]]
```

In [21]:
```
#plot impulse response with bootstrapping confidence bands
plt.ioff()
plt.figure(figsize=(8,15))
periods = irf_proxy.index

plt.subplot(511)
plt.plot(periods, irf_proxy['1ygovbondindx'], 'black', label = 'External Instrument')
```

```
plt.plot(periods, impci[:,0,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci[:,0,1], 'r', linestyle = 'dashed')
plt.xlabel('Periods')
plt.title('1-year German Government Bond', weight = 'bold')
plt.xlim(0, num_impulses)
plt.legend()
plt.grid()

plt.subplot(512)
plt.plot(irf_proxy['HICP'],color = 'black')
plt.plot(periods, impci[:,1,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci[:,1,1], 'r', linestyle = 'dashed')
plt.title('Harmonized Index of Consumer Prices', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)
plt.grid()

plt.subplot(513)
plt.plot(irf_proxy['IP'], color = 'black')
plt.plot(periods, impci[:,2,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci[:,2,1], 'r', linestyle = 'dashed')
plt.title('Industrial Production', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)
plt.grid()

plt.subplot(514)
plt.plot(irf_proxy['eurostoxx50'],color = 'black')
plt.plot(periods, impci[:,3,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci[:,3,1], 'r', linestyle = 'dashed')
plt.title('Euro Stoxx50', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)
plt.grid()

plt.subplot(515)
plt.plot(irf_proxy['bbb_spread'], color = 'black')
plt.plot(periods, impci[:,4,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci[:,4,1], 'r', linestyle = 'dashed')
plt.title('BBB Corporate Bond Spread', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)
plt.grid()

plt.tight_layout()
```

3. Robustness Check Cholesky Identification

December 30, 2019

In [1]:
```
#libraries
import numpy as np
import pandas as pd
import statsmodels.api as sm
from statsmodels.tsa.api import VAR
import matplotlib.pyplot as plt
%matplotlib inline
import datetime
from numpy.linalg import inv
from scipy import linalg
from numpy import linalg as LA
import warnings
warnings.filterwarnings("ignore")
```

0.1 Part I: Load Data

In [2]:
```
#load data and set datetime as index
df = pd.read_excel('data_levels.xlsx')
df.drop(columns = 'OIS_1Y', inplace = True) #no instrument needed
df2 = df.set_index('Date')
```

In [3]:
```
#cholesky ordering
#log of industrial production, log of consumer prices, the 1-year government bond, eur
X = df2[['IP','HICP','1ygovbondindx','eurostoxx50','bbb_spread']]
X.head()
```

Out[3]:

Date	IP	HICP	1ygovbondindx	eurostoxx50	bbb_spread
1999-01-01	4.494239	4.305146	2.883	8.173900	7.410476
1999-02-01	4.481872	4.305416	3.016	8.156005	7.526190
1999-03-01	4.484132	4.307438	2.871	8.177476	7.347826
1999-04-01	4.488636	4.311068	2.616	8.231608	6.990909
1999-05-01	4.490881	4.310799	2.667	8.196839	6.830000

0.2 Part II: Reduced VAR

In [4]:
```
#lag length test
model   = VAR(X)
model.select_order(12).summary()
```

Out[4]: <class 'statsmodels.iolib.table.SimpleTable'>

In [5]:
```
#create dataframe for lags
XLAG     = pd.DataFrame()
num_lags = 2
for i in range(1,num_lags+1):
    XLAG = pd.concat([XLAG,X.shift(i).add_suffix("-"+str(i))],axis=1)

#
X2       = X.iloc[num_lags:,:]
XLAG2    = XLAG.iloc[num_lags:,:]
num_vars = X2.shape[1]
num_obs  = XLAG2.shape[0]

#turn datafram into array
X3       = np.array(X2)
XLAG3    = np.array(XLAG2)

#calculate beta coefficent
Bhat     = inv(XLAG3.T@XLAG3)@XLAG3.T@X3

#print beta coefficient results
col_names = list(X2.columns)
index = list(XLAG2.columns)
coefficient_matrix = pd.DataFrame(Bhat, index = index, columns = col_names)
coefficient_matrix
```

Out[5]:

	IP	HICP	1ygovbondindx	eurostoxx50	bbb_spread
IP-1	0.688560	0.019828	-0.772842	0.026600	-4.739764
HICP-1	0.222482	1.143944	12.326375	0.041549	45.282049
1ygovbondindx-1	0.007181	0.001479	1.206220	0.059325	-1.882308
eurostoxx50-1	0.013894	0.002299	0.183317	0.965809	-5.918025
bbb_spread-1	-0.003683	-0.000184	-0.018017	0.007399	0.988982
IP-2	0.278898	-0.008866	1.290158	0.145637	-0.506159
HICP-2	-0.213005	-0.153208	-13.176063	-0.190087	-40.974560
1ygovbondindx-2	-0.007373	-0.001726	-0.264382	-0.070943	2.165812
eurostoxx50-2	0.000026	-0.003173	0.007398	0.021246	6.508368
bbb_spread-2	0.002969	0.000206	0.014252	-0.006951	-0.090552

0.3 Part III: Cholesky Identification

In [6]:
```
#estimate errors from reduced form VAR
res = X3 - XLAG3@Bhat
sigma = res.T@res/(num_obs - num_lags*num_vars - 1) #variance-covariance matrix
#cholesky transformation
A0 = LA.cholesky(sigma)
d  = np.zeros(A0.shape)
np.fill_diagonal(d,np.diag(A0)) #scale diagonal
```

```
A0 = inv(d)@A0
A0
```

```
Out[6]: array([[ 1.        ,  0.        ,  0.        ,  0.        ,  0.        ],
               [ 0.02680918,  1.        ,  0.        ,  0.        ,  0.        ],
               [ 0.09362299,  0.02129264,  1.        ,  0.        ,  0.        ],
               [ 0.11631999, -0.07631395,  0.46002446,  1.        ,  0.        ],
               [-0.18445229, -0.131     , -0.31630658, -0.27376288,  1.        ]])
```

```
In [7]: ##impulse response function
        num_impulses = 40 #number of periods
        irs = np.zeros([num_lags+num_impulses, num_vars])
        irs[num_lags,:] = (A0[:,2].reshape(-1,1).T)
        for jj in range (1, num_impulses):
            lvars = irs[np.arange(start = num_lags + jj - 1, stop = jj - 1, step = -1),:].T
            irs[(num_lags + jj),:] = lvars.flatten(1).T@Bhat
        irs = irs[num_lags : num_lags + num_impulses,:]

        irf_chol = pd.DataFrame(irs, columns = col_names) #turn irf into dataframe
```

```
In [8]: irf_chol.head()
```

```
Out[8]:          IP      HICP  1ygovbondindx  eurostoxx50  bbb_spread
        0  0.000000  0.000000       1.000000     0.460024   -0.316307
        1  0.014737  0.002594       1.296249     0.501281   -4.917565
        2  0.036808  0.003985       1.499161     0.466188   -5.033820
        3  0.041425  0.004135       1.580896     0.457615   -4.151676
        4  0.045873  0.003665       1.614890     0.449456   -3.244012
```

0.4 Part IV: Cholesky Bootstrapping Confidence Bands

```
In [9]: #function for estimating reduced var
        def estimate(X):
            X = pd.DataFrame(X)
            XLAG     = pd.DataFrame()
            num_lags = 2

            for i in range(1,num_lags+1):
                XLAG = pd.concat([XLAG,X.shift(i).add_suffix("-"+str(i))],axis=1)
            #
            X2       = X.iloc[num_lags:,:]
            XLAG2    = XLAG.iloc[num_lags:,:]
            num_vars = X2.shape[1]
            num_obs  = XLAG2.shape[0]
            #turn into array
            X3       = np.array(X2)
            XLAG3    = np.array(XLAG2)
            #calculate coefficients
            Bhat     = inv(XLAG3.T@XLAG3)@XLAG3.T@X3
```

```
    res = X3 - XLAG3@Bhat
    return res, Bhat
```

In [10]:
```
#function for identification
def cholsvar(sigma):
    cholmatr = LA.cholesky(sigma)
    d = np.zeros(cholmatr.shape)
    np.fill_diagonal(d,np.diag(cholmatr))
    cholmatr = inv(d)@cholmatr
    return cholmatr
```

In [11]:
```
#function for impulse response
def impulse(Bhat, cholmatr):
    num_impulses = 40
    irs = np.zeros([num_lags+num_impulses, num_vars])
    irs[num_lags,:] = (cholmatr[:,2].reshape(-1,1).T)

    for jj in range (1, num_impulses):
        lvars = irs[np.arange(start = num_lags + jj - 1, stop = jj - 1, step = -1),:]
        irs[(num_lags + jj),:] = lvars.flatten(1).T@Bhat

    irs = irs[num_lags : num_lags + num_impulses,:]
    irfbs = pd.DataFrame(irs, columns = col_names)
    return irfbs
```

In [12]:
```
#bootstrapping
#simulate new data and repeat in loop

jj = 0
nboot = 1000 #number of repetitions
imp  = np.zeros([(num_impulses*num_vars),nboot])

for rep in range(1,nboot):
    jj=jj+1

    rr = (1-2*(np.random.random(len(X2)) > 0.5)).reshape(-1,1) #Rademacher distributi

    resb = (res*(rr@np.ones((1, num_vars)))).T #u*
    varsb = np.zeros((len(X), num_vars))
    #initial condition
    varsb[0:num_lags,:] = X[0:num_lags]

    for j in range ((num_lags), (num_lags + len(X2))):
        lvars = (varsb[np.arange(start = j - 1, stop = j-num_lags-1, step = -1)]).T #
        varsb[j,:] = lvars.flatten(1).T@Bhat[0:(num_lags*num_vars),:] + resb[:,j-num_l

    var_j = estimate(varsb) #obtain fitted value for u_star
    res_j = var_j[0]
```

```
        sigma_j = (res_j.T@res_j)/(num_obs-num_lags*num_vars-1)
        A_j = cholsvar(sigma_j)

        irf_j = impulse(var_j[1], A_j)
        irf_x = np.array(irf_j)
        imp[:,jj-1]  = np.reshape(irf_x,(num_impulses*(num_vars),1)).flatten()
```

In [13]:
```
#create confidence bands
imp    = imp.reshape(num_impulses,num_vars,nboot)
imp    = np.sort(imp,axis=2)#
impci  = imp[:,:,[np.int(0.05*nboot),np.int(0.95*nboot)]]
```

In [14]:
```
#plot impulse response with bootstrapping confidence bands
plt.ioff()
plt.figure(figsize=(8,15))
periods = irf_chol.index

plt.subplot(511)
plt.plot(irf_chol['IP'], color = 'black', label = 'Cholesky')
plt.plot(periods, impci[:,0,0], 'r',linestyle='dashed' )
plt.plot(periods, impci[:,0,1], 'r',linestyle='dashed')
plt.title('Industrial Production excl. Construction', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)
plt.legend()
plt.grid()

plt.subplot(512)
plt.plot(irf_chol['HICP'], color = 'black')
plt.plot(periods, impci[:,1,0], 'r', linestyle='dashed')
plt.plot(periods, impci[:,1,1], 'r', linestyle='dashed')
plt.title('Harmonized Index of Consumer Prices', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)
plt.grid()

plt.subplot(513)
plt.plot(periods, irf_chol['1ygovbondindx'], color = 'black')
plt.plot(periods, impci[:,2,0], 'r', linestyle='dashed')
plt.plot(periods, impci[:,2,1], 'r', linestyle='dashed')
plt.title('1-year German Government Bond', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)
plt.grid()

plt.subplot(514)
plt.plot(irf_chol['eurostoxx50'], color = 'black')
plt.plot(periods, impci[:,3,0], 'r',linestyle='dashed')
```

```
plt.plot(periods, impci[:,3,1], 'r',linestyle='dashed')
plt.title('Euro Stoxx 50 Index', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)
plt.grid()

plt.subplot(515)
plt.plot(irf_chol['bbb_spread'], color = 'black')
plt.plot(periods, impci[:,4,0], 'r', linestyle='dashed')
plt.plot(periods, impci[:,4,1], 'r', linestyle='dashed')
plt.title('BBB Corporate Bond Spread', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)
plt.grid()
```

4. Robustness Check Crisis Sample

December 30, 2019

In [1]:
```
#libraries
import numpy as np
import pandas as pd
import statsmodels.api as sm
from statsmodels.tsa.api import VAR
import matplotlib.pyplot as plt
%matplotlib inline
import datetime
from numpy.linalg import inv
from scipy import linalg
import warnings
warnings.filterwarnings("ignore")
```

0.1 Part I: Data & Functions

In [2]:
```
df = pd.read_excel('data_levels.xlsx')
df.head()
```

Out[2]:

	Date	Year	Month	1ygovbondindx	HICP	IP	eurostoxx50	bbb_spread	OIS_1Y
0	1999-01-01	1999	1	2.883	4.305146	4.494239	8.173900	7.410476	-0.25
1	1999-02-01	1999	2	3.016	4.305416	4.481872	8.156005	7.526190	0.00
2	1999-03-01	1999	3	2.871	4.307438	4.484132	8.177476	7.347826	1.00
3	1999-04-01	1999	4	2.616	4.311068	4.488636	8.231608	6.990909	-0.90
4	1999-05-01	1999	5	2.667	4.310799	4.490881	8.196839	6.830000	0.70

In [3]:
```
#time span
post_crisis = df[['Date','1ygovbondindx','HICP','IP','eurostoxx50','bbb_spread', 'OIS_1
X_post = post_crisis[['Date','1ygovbondindx','HICP','IP','eurostoxx50','bbb_spread']].s
X_post.head()
```

Out[3]:

Date	1ygovbondindx	HICP	IP	eurostoxx50	bbb_spread
2008-10-01	2.501	4.521354	4.629863	7.860092	17.016957
2008-11-01	1.999	4.517213	4.590057	7.795774	20.288571
2008-12-01	1.714	4.513384	4.552824	7.802871	22.780455
2009-01-01	1.161	4.514041	4.511958	7.712882	18.964091
2009-02-01	1.029	4.515574	4.489759	7.588946	19.948095

In [4]:
```
#function for estimating reduced VAR
num_lags = 1

def estimate(X):
    X = pd.DataFrame(X)
    XLAG     = pd.DataFrame()
    for i in range(1,num_lags+1):
        XLAG = pd.concat([XLAG,X.shift(i).add_suffix("-"+str(i))],axis=1)
    #change names to frames that we modify
    X2       = X.iloc[num_lags:,:]
    XLAG2    = XLAG.iloc[num_lags:,:]
    num_vars = X2.shape[1]
    num_obs  = XLAG2.shape[0]
    #Building arrays for using OLS
    X3       = np.array(X2)
    XLAG3    = np.array(XLAG2)
    #VAR - standard OLS
    Bhat     = inv(XLAG3.T@XLAG3)@XLAG3.T@X3
    res = X3 - XLAG3@Bhat
    return res, Bhat
```

In [5]:
```
#function for identification
def proxysvar (residual, instrument):
    sigma = (residual.T@residual)/(num_obs - num_lags*num_vars - 1)
    pshock = residual[:,0].reshape(-1,1)
    qshock = residual[:,1:,]
    #first stage
    b_fs = inv(instrument.T@instrument)@instrument.T@pshock
    constant = np.ones(len(pshock))
    constant = np.mean(pshock)-(b_fs*np.mean(instrument))
    #fitted value
    pshock_hat = constant + b_fs*instrument
    #second stage
    b_ss = inv(pshock_hat.T@pshock_hat)@pshock_hat.T@qshock
    #2SLS coefficient is estimate of H21iH11
    b21ib11 = b_ss.T
    #Columns of the covariance matrix
    sigma11 = sigma[0][0].reshape(1,1)
    sigma21 = sigma[1:,0].reshape(-1,1)
    sigma22 = sigma[1:,1:5]
```

```
        #start by estimating Q
        S = sigma22 - b21ib11@sigma21.T - sigma21@(b21ib11.T) + b21ib11*sigma11*b21ib11.T
        #next
        b12b12 = (sigma21 - b21ib11*sigma11).T@inv(S)@(sigma21 - b21ib11*sigma11)
        b11b11 = sigma11 - b12b12
        b11 = np.sqrt(b11b11)
        #find H1, obtained estimates for h11 and h21
        B1 = np.vstack((b11, b21ib11*b11))

        return B1
```

```
In [6]: #function for impulse response
        def impulse(Bhat, B1):
            num_impulses = 40
            irs = np.zeros([num_lags+num_impulses, num_vars])
            irs[num_lags,:] = (B1.T/(B1[0]))

            for jj in range (1, num_impulses):
                lvars = irs[np.arange(start = num_lags + jj - 1, stop = jj - 1, step = -1),:].
                irs[(num_lags + jj),:] = lvars.flatten(1).T@Bhat

            irsb = irs[num_lags : num_lags + num_impulses,:]
            irsb_proxy = pd.DataFrame(irsb, columns = col_names)
            return irsb
```

0.2 Part II: Post-Crisis Sample

```
In [7]: model_post   = VAR(X_post)
        model_post.select_order(8).summary()
```

```
Out[7]: <class 'statsmodels.iolib.table.SimpleTable'>
```

```
In [8]: #dataframe for lags
        XLAG     = pd.DataFrame()
        num_lags = 1
        for i in range(1,num_lags+1):
            XLAG = pd.concat([XLAG,X_post.shift(i).add_suffix("-"+str(i))],axis=1)

        #
        X2       = X_post.iloc[num_lags:,:]
        XLAG2    = XLAG.iloc[num_lags:,:]
        num_vars = X2.shape[1]
        num_obs  = XLAG2.shape[0]

        #turn into array
        X3       = np.array(X2)
        XLAG3    = np.array(XLAG2)

        #calculate beta coefficient
```

```
Bhat     = inv(XLAG3.T@XLAG3)@XLAG3.T@X3

#Print coefficient results
col_names = list(X2.columns)
index = list(XLAG2.columns)
coefficient_matrix = pd.DataFrame(Bhat, index = index, columns = col_names)
coefficient_matrix
```

Out[8]:

	1ygovbondindx	HICP	IP	eurostoxx50	bbb_spread
1ygovbondindx-1	0.964851	-0.000513	0.001354	-0.015212	0.343410
HICP-1	0.581017	0.988323	0.113882	0.429521	-6.799611
IP-1	-0.498559	0.015777	0.853584	-0.251250	9.848345
eurostoxx50-1	-0.043502	-0.002177	0.019522	0.897664	-1.697296
bbb_spread-1	-0.008019	-0.000042	-0.000873	0.000664	0.896826

In [9]:
```
#estimate errors
date_post = list(X2.index)
res_post = X3 - XLAG3@Bhat
u_post = pd.DataFrame((X3 - XLAG3@Bhat),index = date_post, columns = col_names)

#reduced error covariance matrix
sigma_post = (u_post.T@u_post)/(num_obs - num_lags*num_vars - 1)

#partition errors
#policy residuals
res_post_p = u_post[['1ygovbondindx']]

#non-policy residuals
res_post_q = u_post[['HICP', 'IP','eurostoxx50', 'bbb_spread']]

#turn into array
u_post_p = np.array(res_post_p).reshape(len(X3),1)
u_post_q = np.array(res_post_q)

u_post.head()
```

Out[9]:

	1ygovbondindx	HICP	IP	eurostoxx50	bbb_spread
2008-11-01	-0.254424	-0.005286	-0.018793	-0.011970	2.656319
2008-12-01	-0.049071	-0.004655	-0.016783	0.034831	2.641227
2009-01-01	-0.323134	0.000347	-0.023008	-0.075230	-2.959358
2009-02-01	0.023153	0.001235	-0.011227	-0.134813	1.891359
2009-03-01	-0.032945	-0.001320	-0.003693	0.014446	2.579796

In [10]:
```
instrument_post = post_crisis[['OIS_1Y']].iloc[num_lags:,:] #adapt data range
Z_post = np.array(instrument_post)
```

In [11]:
```
#2SlS
#first Stage: OLS with u_p and instrument
```

```
b_post_p = inv(Z_post.T@Z_post)@Z_post.T@u_post_p # coefficient for Z

#find constant
N = len(u_post_p)
c = np.ones(N)
c0_post = np.mean(u_post_p)-(b_post_p*np.mean(Z_post))

#fitted values
u_post_p_hat = c0_post + b_post_p*Z_post

print('The first stage coefficients are',c0_post, 'and', b_post_p)

#Produce table of actual and fitted values of dependent variable
actual = pd.DataFrame(u_post_p).rename(columns = {0: 'actual'})
fitted = pd.DataFrame(u_post_p_hat).rename(columns = {0: 'fitted'})
t1 = actual.join(fitted)
t1.head()
```

```
The first stage coefficients are [[0.00014298]] and [[0.00791587]]
```

Out[11]:

	actual	fitted
0	-0.254424	0.107799
1	-0.049071	0.037348
2	-0.323134	-0.008564
3	0.023153	-0.050519
4	-0.032945	-0.010148

In [12]:
```
#f-test for weak instruments
k = 2
T = len(Z_post)

SSE = (u_post_p - c0_post - b_post_p*Z_post).T@(u_post_p - c0_post - b_post_p*Z_post)
SST = (u_post_p - np.mean(u_post_p)).T@(u_post_p - np.mean(u_post_p))

r_squared = 1 - (SSE/SST)

F_test = (r_squared/(k-1))/((1-r_squared)/(T-k))
print('R-squared:', r_squared)
print('F-test:',F_test)
```

```
R-squared: [[0.06888323]]
F-test: [[7.98974798]]
```

In [13]:
```
#Second stage
#u_q = b * u_p_hat
b_iv_post = inv(u_post_p_hat.T@u_post_p_hat)@u_post_p_hat.T@u_post_q
print(b_iv_post)
```

```
[[-0.00214137 -0.06687888  0.27507276 -0.50915359]]
```

In [14]:
```
#Reduced VAR variance-covariance matrix
sig_post = np.array(sigma_post)

#2SLS coefficient is estimate of H21iH11
h21ih11 = b_iv_post.T

#Columns of the covariance matrix
sig11 = sig_post[0][0].reshape(1,1)
sig21 = sig_post[1:,0].reshape(-1,1)
sig22 = sig_post[1:,1:5]

#start by estimating Z
Q = sig22 - h21ih11@sig21.T - sig21@(h21ih11.T) + h21ih11*sig11*h21ih11.T

#next
h12h12 = (sig21 - h21ih11*sig11).T@inv(Q)@(sig21 - h21ih11*sig11)
h11h11 = sig11 - h12h12
h11 = np.sqrt(h11h11)
print('h11 is',h11)
print('and h21 is', h21ih11*h11)

#find H1, obtained estimates for h11 and h21
H1_post = np.vstack((h11, (h21ih11*h11)))
```

```
h11 is [[0.08906932]]
and h21 is [[-0.00019073]
 [-0.00595686]
 [ 0.02450055]
 [-0.04534997]]
```

In [15]:
```
#impulse response function
num_impulses = 40
irs_post = np.zeros([num_lags+num_impulses, num_vars])
irs_post[num_lags,:] = (H1_post.T/H1_post[0])

for jj in range (1, num_impulses):
    lvars_post = irs_post[np.arange(start = num_lags + jj - 1, stop = jj - 1, step = -
    irs_post[(num_lags + jj),:] = lvars_post.flatten(1).T@Bhat

irs_post = irs_post[num_lags : num_lags + num_impulses,:]
irf_post = pd.DataFrame(irs_post, columns = col_names)
```

In [16]:
```
irf_post.head()
```

Out[16]:

	1ygovbondindx	HICP	IP	eurostoxx50	bbb_spread
0	1.000000	-0.002141	-0.066879	0.275073	-0.509154

```
1       0.989067 -0.004262 -0.050162     0.247256   -1.224178
2       0.975895 -0.005999 -0.036067     0.216867   -1.642915
3       0.959830 -0.007402 -0.024480     0.185223   -1.820779
4       0.940540 -0.008522 -0.015233     0.153429   -1.808437
```

In [17]:
```
#bootstrapping
#simulate new data and make loop
jj = 0
nboot = 1000
imp_post  = np.zeros([(num_impulses*num_vars),nboot])

for rep in range(1,nboot):
    jj=jj+1

    rr = (1-2*(np.random.random(len(X2)) > 0.5)).reshape(-1,1) #Rademacher distributi

    resb_post = (res_post*(rr@np.ones((1, num_vars)))).T #u*
    Zb_post = np.vstack(((Z_post[0:num_lags,:]),(rr*np.ones((1,1))*Z_post))) #Z*
    varsb_post = np.zeros((len(X_post), num_vars))
    #initial condition
    varsb_post[0:num_lags,:] = X_post[0:num_lags]

    for j in range ((num_lags), (num_lags + len(X2))):
        lvars_post = (varsb_post[np.arange(start = j - 1, stop = j-num_lags-1, step =
        varsb_post[j,:] = lvars_post.flatten(1).T@Bhat[0:(num_lags*num_vars),:] + resb

    var_post = estimate(varsb_post) #obtain fitted value for u_star
    H_post = proxysvar(var_post[0], Zb_post[num_lags:,:])
    irf_b_post = impulse(var_post[1], H_post)
    irf_bs_post = np.array(irf_b_post)
    imp_post[:,jj-1]  = np.reshape(irf_bs_post,(num_impulses*(num_vars),1)).flatten()
```

In [18]:
```
#create confidence bands
imp_post    = imp_post.reshape(num_impulses,num_vars,nboot)
imp_post    = np.sort(imp_post,axis=2)#
impci_post  = imp_post[:,:,[np.int(0.05*nboot),np.int(0.95*nboot)]]
```

In [19]:
```
#impulse response with bootstrapping confidence bands
plt.ioff()
plt.figure(figsize=(8,15))
periods = irf_post.index

plt.subplot(511)
plt.plot(periods, irf_post['1ygovbondindx'], 'black', label = 'External Instrument')
plt.plot(periods, impci_post[:,0,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci_post[:,0,1], 'r', linestyle = 'dashed')
plt.xlabel('Periods')
plt.title('1-year German Government Bond', weight = 'bold')
```

```
plt.xlim(0, num_impulses)
plt.legend()
plt.grid()

plt.subplot(512)
plt.plot(irf_post['HICP'],color = 'black')
plt.plot(periods, impci_post[:,1,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci_post[:,1,1], 'r', linestyle = 'dashed')
plt.grid()
plt.title('Harmonized Index of Consumer Prices', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)

plt.subplot(513)
plt.plot(irf_post['IP'], color = 'black')
plt.plot(periods, impci_post[:,2,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci_post[:,2,1], 'r', linestyle = 'dashed')
plt.grid()
plt.title('Industrial Production', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)

plt.subplot(514)
plt.plot(irf_post['eurostoxx50'],color = 'black',label='External Instrument')
plt.plot(periods, impci_post[:,3,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci_post[:,3,1], 'r', linestyle = 'dashed')
plt.grid()
plt.title('Euro Stoxx50 Index', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)

plt.subplot(515)
plt.plot(irf_post['bbb_spread'], color = 'black')
plt.plot(periods, impci_post[:,4,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci_post[:,4,1], 'r', linestyle = 'dashed')
plt.grid()
plt.title('BBB Corporate Bond Spread', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)

plt.tight_layout()
```

5. Robustness Check Alternative Instrument

December 30, 2019

In [1]:
```
#libraries
import numpy as np
import pandas as pd
import statsmodels.api as sm
from statsmodels.tsa.api import VAR
import matplotlib.pyplot as plt
%matplotlib inline
import datetime
from numpy.linalg import inv
from scipy import linalg
import warnings
warnings.filterwarnings("ignore")
```

0.1 Part I: Data

In [2]:
```
#load data and set datetime as index
df = pd.read_excel('data_levels.xlsx')
df.drop(columns = ['OIS_1Y'], inplace = True) #drop instrument of baseline VAR
df.head()
```

Out[2]:

	Date	Year	Month	1ygovbondindx	HICP	IP	eurostoxx50	bbb_spread
0	1999-01-01	1999	1	2.883	4.305146	4.494239	8.173900	7.410476
1	1999-02-01	1999	2	3.016	4.305416	4.481872	8.156005	7.526190
2	1999-03-01	1999	3	2.871	4.307438	4.484132	8.177476	7.347826
3	1999-04-01	1999	4	2.616	4.311068	4.488636	8.231608	6.990909
4	1999-05-01	1999	5	2.667	4.310799	4.490881	8.196839	6.830000

In [3]:
```
#load alternative instrument
instrument = pd.read_excel('EA_alternative_instrument.xlsx')
instrument.head()
```

Out[3]:

	Date	Year	Month	OIS_6M
0	1999-01-07	1999	1	-5.25
1	1999-01-21	1999	1	1.00
2	1999-02-18	1999	2	0.00
3	1999-03-04	1999	3	0.00
4	1999-03-18	1999	3	-0.50

In [4]:
```
#transform instrument to a monthly time series
instrument2 = instrument.groupby(['Year', 'Month']).sum().reset_index()
instrument2['Days'] = np.ones((len(instrument2['Month'])))
instrument2['Date'] = pd.to_datetime((instrument2.Year*10000+instrument2.Month*100+inst
instrument3 = instrument2.drop(columns = ['Year', 'Month', 'Days']).set_index('Date')
instrument3.head()
```

Out[4]:

Date	OIS_6M
1999-01-01	-4.25
1999-02-01	0.00
1999-03-01	-0.50
1999-04-01	0.10
1999-05-01	0.10

In [5]:
```
#final dataframe
data_final = pd.merge(df, instrument3, on = ['Date'], how = 'outer').dropna()
data_final.head()
```

Out[5]:

	Date	Year	Month	1ygovbondindx	HICP	IP	eurostoxx50	\
0	1999-01-01	1999	1	2.883	4.305146	4.494239	8.173900	
1	1999-02-01	1999	2	3.016	4.305416	4.481872	8.156005	
2	1999-03-01	1999	3	2.871	4.307438	4.484132	8.177476	
3	1999-04-01	1999	4	2.616	4.311068	4.488636	8.231608	
4	1999-05-01	1999	5	2.667	4.310799	4.490881	8.196839	

	bbb_spread	OIS_6M
0	7.410476	-4.25
1	7.526190	0.00
2	7.347826	-0.50
3	6.990909	0.10
4	6.830000	0.10

In [6]:
```
#set variables for reduced VAR
X = data_final[['1ygovbondindx','HICP','IP','eurostoxx50','bbb_spread']]
```

0.2 Part II: Estimation Reduced VAR

In [7]:
```
#lag length test
model    = VAR(X)
model.select_order(12).summary()
```

```
Out[7]: <class 'statsmodels.iolib.table.SimpleTable'>
```

```
In [8]: #create dataframe with lags
        XLAG     = pd.DataFrame()
        num_lags = 2 #number of lags
        for i in range(1,num_lags+1):
            XLAG = pd.concat([XLAG,X.shift(i).add_suffix("-"+str(i))],axis=1)
        #
        X2       = X.iloc[num_lags:,:]
        XLAG2    = XLAG.iloc[num_lags:,:]
        num_vars = X2.shape[1]
        num_obs  = XLAG2.shape[0]

        #turn dataframe into an array
        X3       = np.array(X2)
        XLAG3    = np.array(XLAG2)

        #calculate beta coefficient
        Bhat     = inv(XLAG3.T@XLAG3)@XLAG3.T@X3
```

0.3 Part III: Two-Stage-Least-Square Estimation

```
In [9]: col_names = list(X2.columns)
        date = list(X2.index)

        #estimated errors from reduced form
        res = X3 - XLAG3@Bhat
        u = pd.DataFrame((X3 - XLAG3@Bhat),index = date, columns = col_names)

        #reduced error covariance matrix
        sigma = (u.T@u)/(num_obs - num_lags*num_vars - 1)

        #partition residuals
        #policy residuals
        res_p = u[['1ygovbondindx']]

        #non-policy residuals
        res_q = u[['HICP', 'IP','eurostoxx50', 'bbb_spread']]

        #turn into array
        u_p = np.array(res_p).reshape(len(X3),1)
        u_q = np.array(res_q)

        u.head() #show excerpt of residuals
```

```
Out[9]:    1ygovbondindx      HICP        IP  eurostoxx50  bbb_spread
        2      -0.210170  0.000402 -0.004420     0.013827    0.038525
        3      -0.267440  0.001887  0.002309     0.069471   -0.525029
```

```
4       0.021309 -0.002252 -0.000331    -0.012007   -0.393453
5       0.169593 -0.001076  0.004220     0.040930   -0.226711
6      -0.004069 -0.000314  0.004733    -0.049821    0.136371
```

In [10]:
```
#get instrument
Z = data_final[['OIS_6M']].iloc[num_lags:,:] #adapt data range to number of lags
Z = np.array(Z)
```

In [11]:
```
#2SLS
#first stage: OLS with policy shock and instrument

b_p = inv(Z.T@Z)@Z.T@u_p # coefficient for Z

#find constant
N = len(u_p)
c = np.ones(N)
c0 = np.mean(u_p)-(b_p*np.mean(Z))

#calculate fitted values for policy shock
u_p_hat = c0 + b_p*Z

print('The first stage coefficients are',c0, 'and', b_p) #results
```

```
The first stage coefficients are [[-0.00015531]] and [[0.01373348]]
```

In [12]:
```
#F-test for weak instruments
k = 2
T = len(Z)

SSE = (u_p - c0 - b_p*Z).T@(u_p - c0 - b_p*Z) #sum of squared residuals
SST = (u_p - np.mean(u_p)).T@(u_p - np.mean(u_p)) #total sum of squares
r_squared = 1 - (SSE/SST) #r-squared

F_test = (r_squared/(k-1))/((1-r_squared)/(T-k)) #F-test
#results
print('R-squared:',r_squared)
print('F-test:', F_test)
```

```
R-squared: [[0.11659479]]
F-test: [[29.30030694]]
```

In [13]:
```
#Second stage
#u_q = b * u_p_hat
b_iv = inv(u_p_hat.T@u_p_hat)@u_p_hat.T@u_q
print(b_iv)
```

```
[[ 1.52143309e-04 -1.81087293e-02  1.17240720e-01 -1.12700210e+00]]
```

0.4 Part IV: Identification

In [14]:
```
#Reduced VAR variance-covariance matrix
sig = np.array(sigma)

#2SLS coefficient is estimate of H21iH11
h21ih11 = b_iv.T

#Columns of the covariance matrix
sig11 = sig[0][0].reshape(1,1)
sig21 = sig[1:,0].reshape(-1,1)
sig22 = sig[1:,1:5]

#start by estimating Z
Q = sig22 - h21ih11@sig21.T - sig21@(h21ih11.T) + h21ih11*sig11*h21ih11.T

#next
h12h12 = (sig21 - h21ih11*sig11).T@inv(Q)@(sig21 - h21ih11*sig11)
h11h11 = sig11 - h12h12
h11 = np.sqrt(h11h11)
print('h11 is',h11)
print('and h21 is', h21ih11*h11)

#find H1, obtained estimates for h11 and h21
H1 = np.vstack((h11, (h21ih11*h11)))
```

```
h11 is [[0.15201587]]
and h21 is [[ 2.31281970e-05]
 [-2.75281419e-03]
 [ 1.78224497e-02]
 [-1.71322202e-01]]
```

0.5 Part V: Impulse Response Function

In [15]:
```
#impulse response function
num_impulses = 40
irs = np.zeros([num_lags+num_impulses, num_vars])
irs[num_lags,:] = (H1.T/(H1[0]))

for jj in range (1, num_impulses):
    lvars = irs[np.arange(start = num_lags + jj - 1, stop = jj - 1, step = -1),:].T
    irs[(num_lags + jj),:] = lvars.flatten(1).T@Bhat

irs = irs[num_lags : num_lags + num_impulses,:]
irs = irs
irf_proxy = pd.DataFrame(irs, columns = col_names)
```

In [16]:
```
irf_proxy.head()
```

Out[16]:

	1ygovbondindx	HICP	IP	eurostoxx50	bbb_spread
0	1.000000	0.000152	-0.018109	0.117241	-1.127002
1	1.263888	0.001771	0.000525	0.163744	-3.598005
2	1.335849	0.002752	0.009558	0.143309	-3.794887
3	1.325633	0.002623	0.012525	0.128516	-2.980327
4	1.268788	0.002005	0.012455	0.116681	-2.091942

0.6 Part VI: Bootstrapping Confidence Bands

In [17]:
```
#function for estimating reduced VAR
def estimate(X):
    X = pd.DataFrame(X)
    XLAG     = pd.DataFrame()

    for i in range(1,num_lags+1):
        XLAG = pd.concat([XLAG,X.shift(i).add_suffix("-"+str(i))],axis=1)
    #
    X2       = X.iloc[num_lags:,:]
    XLAG2    = XLAG.iloc[num_lags:,:]
    num_vars = X2.shape[1]
    num_obs  = XLAG2.shape[0]
    #
    X3       = np.array(X2)
    XLAG3    = np.array(XLAG2)
    #
    Bhat     = inv(XLAG3.T@XLAG3)@XLAG3.T@X3
    res = X3 - XLAG3@Bhat
    return res, Bhat
```

In [18]:
```
#function for identification
def proxysvar (residual, instrument):
    sigma = (residual.T@residual)/(num_obs - num_lags*num_vars - 1)
    pshock = residual[:,0].reshape(-1,1)
    qshock = residual[:,1:,]
    #first stage
    b_fs = inv(instrument.T@instrument)@instrument.T@pshock
    constant = np.ones(len(pshock))
    constant = np.mean(pshock)-(b_fs*np.mean(instrument))
    #fitted value
    pshock_hat = constant + b_fs*instrument
    #second stage
    b_ss = inv(pshock_hat.T@pshock_hat)@pshock_hat.T@qshock
    #2SLS coefficient is estimate of H21iH11
    b21ib11 = b_ss.T
    #Columns of the covariance matrix
    sigma11 = sigma[0][0].reshape(1,1)
    sigma21 = sigma[1:,0].reshape(-1,1)
    sigma22 = sigma[1:,1:5]
```

```
        #start by estimating Q
        S = sigma22 - b21ib11@sigma21.T - sigma21@(b21ib11.T) + b21ib11*sigma11*b21ib11.T
        #next
        b12b12 = (sigma21 - b21ib11*sigma11).T@inv(S)@(sigma21 - b21ib11*sigma11)
        b11b11 = sigma11 - b12b12
        b11 = np.sqrt(b11b11)
        #find H1, obtained estimates for h11 and h21
        B1 = np.vstack((b11, b21ib11*b11))

        return B1
```

In [19]:
```
#function for impulse response
def impulse(Bhat, B1):
    num_impulses = 40
    irs = np.zeros([num_lags+num_impulses, num_vars])
    irs[num_lags,:] = (B1.T/(B1[0]))

    for jj in range (1, num_impulses):
        lvars = irs[np.arange(start = num_lags + jj - 1, stop = jj - 1, step = -1),:]
        irs[(num_lags + jj),:] = lvars.flatten(1).T@Bhat

    irsb = irs[num_lags : num_lags + num_impulses,:]
    return irsb
```

In [20]:
```
#bootstrapping
#simulate new data and make loop

jj = 0
nboot = 1000
imp  = np.zeros([(num_impulses*num_vars),nboot])

for rep in range(1,nboot):
    jj=jj+1

    rr = (1-2*(np.random.random(len(X2)) > 0.5)).reshape(-1,1) #Rademacher distributi

    resb = (res*(rr@np.ones((1, num_vars)))).T #u*
    Zb = np.vstack(((Z[0:num_lags,:]),(rr*np.ones((1,1))*Z))) #Z*
    varsb = np.zeros((len(X), num_vars))
    #initial condition
    varsb[0:num_lags,:] = X[0:num_lags]

    for j in range ((num_lags), (num_lags + len(X2))):
        lvars = (varsb[np.arange(start = j - 1, stop = j-num_lags-1, step = -1)]).T #
        varsb[j,:] = lvars.flatten(1).T@Bhat[0:(num_lags*num_vars),:] + resb[:,j-num_l

    var_j = estimate(varsb) #obtain fitted value for u_star
    H_j = proxysvar(var_j[0], Zb[num_lags:,:])
```

```
            irf_j = impulse(var_j[1], H_j)
            irf_x = np.array(irf_j)
            imp[:,jj-1]  = np.reshape(irf_x,(num_impulses*(num_vars),1)).flatten()
```

In [21]:
```
#create confidence bands
imp    = imp.reshape(num_impulses,num_vars,nboot)
imp    = np.sort(imp,axis=2)#
impci  = imp[:,:,[np.int(0.05*nboot),np.int(0.95*nboot)]]
```

In [22]:
```
#impulse response with bootstrapping confidence bands
plt.ioff()
plt.figure(figsize=(8,15))
periods = irf_proxy.index

plt.subplot(511)
plt.plot(periods, irf_proxy['1ygovbondindx'], 'black', label = 'External Instrument')
plt.plot(periods, impci[:,0,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci[:,0,1], 'r', linestyle = 'dashed')
plt.xlabel('Periods')
plt.title('1-year German Government Bond', weight = 'bold')
plt.xlim(0, num_impulses)
plt.legend()
plt.grid()

plt.subplot(512)
plt.plot(irf_proxy['HICP'],color = 'black')
plt.plot(periods, impci[:,1,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci[:,1,1], 'r', linestyle = 'dashed')
plt.title('Harmonized Index of Consumer Prices', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)
plt.grid()

plt.subplot(513)
plt.plot(irf_proxy['IP'], color = 'black')
plt.plot(periods, impci[:,2,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci[:,2,1], 'r', linestyle = 'dashed')
plt.title('Industrial Production', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)
plt.grid()

plt.subplot(514)
plt.plot(irf_proxy['eurostoxx50'],color = 'black')
plt.plot(periods, impci[:,3,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci[:,3,1], 'r', linestyle = 'dashed')
plt.title('Euro Stoxx50 Index', weight = 'bold')
plt.xlabel('Periods')
```

```
plt.xlim(0, num_impulses)
plt.grid()

plt.subplot(515)
plt.plot(irf_proxy['bbb_spread'], color = 'black')
plt.plot(periods, impci[:,4,0], 'r', linestyle = 'dashed')
plt.plot(periods, impci[:,4,1], 'r', linestyle = 'dashed')
plt.title('BBB Corporate Bond Spread', weight = 'bold')
plt.xlabel('Periods')
plt.xlim(0, num_impulses)
plt.grid()

plt.tight_layout()
```