City University of New York
Hunter College, Department of
Mathematics and Statistics

Fitting time-varying parameters to Multivariate time series

Abrar Imon

SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ARTS IN STATISTICS

Advisor : Dr. Giovanni Motta
Columbia University, USA

December 2023

City University of New York, Hunter College
DEPARTMENT OF MATHEMATICS AND STATISTICS

The undersigned hereby certify that they have read and recommend to the Faculty of Mathematics for acceptance a thesis entitled **Fitting time-varying parameters to Multivariate time series** by **Abrar Imon** in partial fulfillment of the requirements for the degree of **Master in Statistics**.

Dated: December the 4th, 2023

Research Supervisor — Giovanni Motta
Columbia University

Academic Advisor — Sandra Clarkson
City University of New York

City University of New York

Date: December 2023

Author	:	Abrar Imon
Title	:	Fitting time-varying parameters to Multivariate Financial Time Series
Department	:	Mathematics and Statistics
Degree	:	Master in Statistics
Convocation	:	December
Year	:	2023

Permission is herewith granted to CUNY to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

_____

Signature of Author

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Time Series in Finance

In the field of finance, there is an abundance of data as there are many different assets, financial derivatives, and macroeconomic indicators. These data are typically analyzed as time series data with the use of various statistical methods. Modelling these data allows us to understand underlying patterns in the financial indicators. These patterns include: temporal dependencies (the effects of past and present on the future), and associations between different time series, both of which will be studied in this thesis. We will study these by modelling various financial time series. These include: the price of index funds, the federal funds effective rate (interest rate), and the inflation rate. In the next section, we briefly explain some traditional statistical methods and show how these methods can be used to model the data. Then, we introduce the locally stationary vector auto-regression (LS-VAR) estimation method and use it to gain further insights on the different time series.

One of the time series data we are using for this thesis is the price of the Vanguard 500 Index Fund. This index fund has the ticker symbol: VOO. It is made up of 500 of the largest publicly traded companies in the United States. Therefore, the performance of this fund tells us about the health of the U.S. economy. We will investigate the price of this

fund in the ten years from May, 2013 to April, 2023. This data is structured in monthly increments, where each observation is the price (of the fund) from the first business day of a month, at the time of market close (4:00PM ET). Hence, our time series data has a total of 120 observations and is sourced from Yahoo Finance.

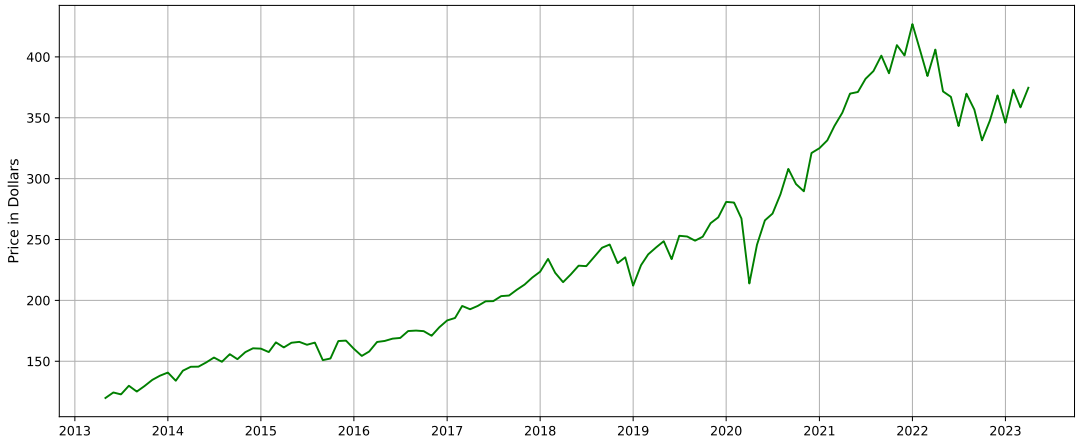


Figure 1.1 Vanguard 500 Index Fund (ticker symbol: VOO)

Another time series data we are using is the Federal Funds Effective Rate (FFER). The FFER is an important component of the U.S. economy. The FFER is an interest rate that is controlled directly by the Fed. When the FFER is lowered, it stimulates the economy by encouraging borrowing and spending of money. This tends to speed up inflation, which can weaken the U.S. currency. So, controlling the FFER is a way to balance between economic productivity and inflation (among other things). The time series data of FFER that we model in this thesis is from May, 2005 to April, 2023. Each observation is made at the first day of each month in which the stock market is open. The observations are made near 4:21PM ET.

One of the sectors that exhibits significant associations with the FFER, is the financial sector. There are many ways to obtain a value that describes the performance of the financial sector. In this thesis, we use the Financial Select Sector SPDR Fund, which goes by the ticker symbol: XLF. This is a diversified portfolio of stocks primarily from the financial sector. This fund includes financial institutions, such as banks, insurance com-

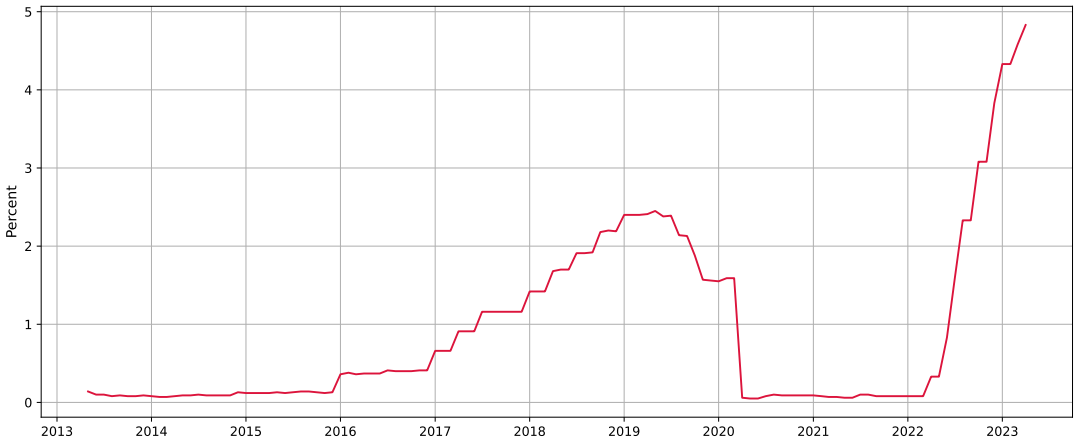


Figure 1.2 Monthly Federal Funds Effective Rate

panies, real estate firms, etc. The time series data of XLF is structured in the same way as the FFER data. The only difference is that the observations are made near market close (4:00PM ET). Analyzing the time series data of FFER and the XLF fund, can offer insights on the associations between these two variables as well as insights on how this association changes over time.

Figure 1.3 Financial Select Sector SPDR Fund (ticker symbol: XLF)

1.2 Time series models with time-invariant parameters

1.2.1 Univariate AR(p) processes

In this section, we introduce simpler versions of the Locally Stationary VAR method. In this particular subsection, we introduce the auto-regression (AR) method and apply it to model the time series of Vanguard 500 Index Fund. A general AR process of order p takes the form:

$$\begin{aligned} X_t &= c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \varepsilon_t \\ &= \varepsilon_t + c + \sum_{i=1}^{p} \phi_i X_{t-i}, \end{aligned} \tag{1.1}$$

where X_t is the value of the time series at time $t = 1, 2, \ldots, n$, , the ϕ_i's are the coefficients, c is the intercept, and ε_t is the error at time t, and $\varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ such that $\forall t \neq s, \mathbb{C}\text{ov}(\varepsilon_t, \varepsilon_s) = 0$. Notice that the coefficients ϕ_i's and the intercept c do not depend on time. An AR(p) process has first and second moments that do not depend on time. We refer to this as the stationary condition. The condition of the first moment being independent of time implies we can define $\mathbb{E}[X_t] := \mu$. Now, taking the expectation of equation (1.1), we obtain $\mu = c + \sum \phi_i \mu \implies c = (1 - \sum \phi_i)\mu$. Substituting this for c in equation (1.1), we can rewrite equation (1.1) as:

$$(X_t - \mu) = \varepsilon_t + \sum_{i=1}^{p} \phi_i (X_{t-i} - \mu). \tag{1.2}$$

ACVF and ACF of an AR(p) process

Consider the covariance, $\mathbb{C}\text{ov}(X_t, X_{t-h})$, between two points in the time series. The condition of the second moments being independent of time means that this covariance is only a function of h. This justifies the notation $\gamma(h) := \mathbb{C}\text{ov}(X_{t+h}, X_t)$ and we refer to this as the autocovariance function (ACVF). An important property of this function is that $\gamma(h) = \gamma(-h)$. To prove this, we use the time invariant and commutative property of $\mathbb{C}\text{ov}(X_{t+h}, X_t)$; $\gamma(-h) = \mathbb{C}\text{ov}(X_{t-h}, X_t) = \mathbb{C}\text{ov}(X_t, X_{t+h}) = \mathbb{C}\text{ov}(X_{t+h}, X_t) = \gamma(h)$.

Also, notice from the definition that $\gamma(0) = \mathbb{V}\mathrm{ar}(X_t)$. Hence, we are able to define the autocorrelation $\rho(h) := \gamma(h)/\gamma(0)$ in terms of the ACVF.

The coefficients ϕ_i's of the model can be written in terms of the autocovariance. To do this, we start with equation (1.2), which is one representation of the AR(p) model. Multiply both sides of this equation by $(X_{t-j} - \mu)$ and take the expectation to obtain the equation: $\mathbb{E}[(X_t-\mu)(X_{t-j}-\mu)] = \sum_{i=1}^{p} \phi_i \mathbb{E}[(X_{t-i}-\mu)(X_{t-j}-\mu)]$. This can be rewritten with the γ notation: $\gamma(j) = \sum_{i=1}^{p} \phi_i \gamma(j-i)$. Substituting $j = 1, 2, \ldots, p$ into this, we obtain p different linear equation. This system of equations is known as the Yule-Walker equations and is written below in matrix notation:

$$\begin{bmatrix} \gamma(1) \\ \gamma(2) \\ \vdots \\ \gamma(p) \end{bmatrix} = \begin{bmatrix} \gamma(0) & \gamma(-1) & \ldots & \gamma(1-p) \\ \gamma(1) & \gamma(0) & \ldots & \gamma(2-p) \\ \vdots & \vdots & \ddots & \vdots \\ \gamma(p-1) & \gamma(p-2) & \ldots & \gamma(0) \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_p \end{bmatrix}. \tag{1.3}$$

We can rewrite the above equation as $\boldsymbol{\gamma} = \boldsymbol{\Gamma}_p \boldsymbol{\phi}$, where $\boldsymbol{\gamma} = [\gamma(1), \gamma(2), \ldots, \gamma(p)]'$, $\boldsymbol{\Gamma}_p$ is the $p \times p$ matrix of covariances, and $\boldsymbol{\phi} = [\phi_1, \phi_2, \ldots, \phi_p]'$. The $\boldsymbol{\Gamma}_p$ matrix is symmetric, because $\gamma(h) = \gamma(-h)$. From the data, $\{X_t | t = 1, 2, \ldots, n\}$, we obtain the estimates $\hat{\gamma}(0), \hat{\gamma}(1), \ldots, \hat{\gamma}(p)$, with the following formula:

$$\hat{\gamma}(h) = \tfrac{1}{n} \sum_{t=1}^{n-|h|} (X_t - \hat{\mu})(X_{t+|h|} - \hat{\mu}), \qquad \text{with} \qquad \hat{\mu} = \tfrac{1}{n} \sum_{t=1}^{n} X_t. \tag{1.4}$$

We can use these equations to estimate $\hat{\boldsymbol{\gamma}}$ and $\hat{\boldsymbol{\Gamma}}_p$. In the formula for $\hat{\gamma}(h)$, we divide by n and not $n - |h|$ to make sure that $\hat{\boldsymbol{\Gamma}}$ is positive semi-definite. This choice to divide by n also causes our estimates to be more biased for larger values of h. Using the estimates $\hat{\boldsymbol{\gamma}}$ and $\hat{\boldsymbol{\Gamma}}_p$, we write the equation for the estimated vector of coefficients: $\hat{\boldsymbol{\gamma}} = \hat{\boldsymbol{\Gamma}}_p \hat{\boldsymbol{\phi}}$. If $\hat{\boldsymbol{\Gamma}}_p$ is invertible, then $\hat{\boldsymbol{\phi}} = \hat{\boldsymbol{\Gamma}}_p^{-1} \hat{\boldsymbol{\gamma}}$. We can represent $\hat{\boldsymbol{\phi}}$ in terms of the vector of autocorrelation, $\hat{\boldsymbol{\rho}} = \hat{\boldsymbol{\gamma}}/\hat{\gamma}(0)$, by defining $\hat{\mathbf{R}}_p := \hat{\boldsymbol{\Gamma}}_p/\hat{\gamma}(0)$. This definition implies that $\hat{\mathbf{R}}_p^{-1} = \hat{\gamma}(0)\hat{\boldsymbol{\Gamma}}_p^{-1}$. Now, we have $\hat{\boldsymbol{\phi}} = \hat{\boldsymbol{\Gamma}}_p^{-1}\hat{\boldsymbol{\gamma}} = (\hat{\gamma}(0)\hat{\boldsymbol{\Gamma}}_p^{-1})(\hat{\boldsymbol{\gamma}}/\hat{\gamma}(0)) = \hat{\mathbf{R}}_p^{-1}\hat{\boldsymbol{\rho}}$.

An AR(p) model of the form (1.2) implies that the first p coefficients $\phi_1, \phi_2, \ldots, \phi_p$ are

non-zero and the coefficients after that are 0. Therefore, the equation $\hat{\boldsymbol{\phi}} = \hat{\mathbf{R}}_p^{-1}\hat{\boldsymbol{\rho}}$ implies that the elements of the estimated $\hat{\mathbf{R}}_p^{-1}\hat{\boldsymbol{\rho}}$ are significantly different from 0. Each element of $\hat{\mathbf{R}}_p^{-1}\hat{\boldsymbol{\rho}}$ is not directly related to a specific autocorrelation $\hat{\rho}(i)$. Therefore, testing whether each estimated autocorrelation $\hat{\rho}(i)$ is significantly different from 0, does not directly inform us of the order of an AR model.

PACF of an AR(p) process

Since we are modelling the time series with an AR(p), we want to analyze the partial autocorrelation function (PACF). The PACF between X_t and X_{t-h} for some positive integer h is the remaining correlation between X_t and X_{t-h} after accounting for the effects of $X_{t-1}, X_{t-2}, \ldots, X_{t-h+1}$. The PACF $\alpha(h)$ tells us whether estimating a ϕ_h coefficient significantly improves the fit of an AR($h-1$) fit. This is the same as checking whether the last coefficient ϕ_{hh} of an AR(h) fit is non-zero. It can actually be shown that $\alpha(h) = \phi_{hh}$. We estimate $\hat{\alpha}(h) = \hat{\phi}_{hh}$ by estimating the coefficients of an AR(h) fit on the data and then our estimate $\hat{\phi}_{hh}$ is the last coefficient of that fit.

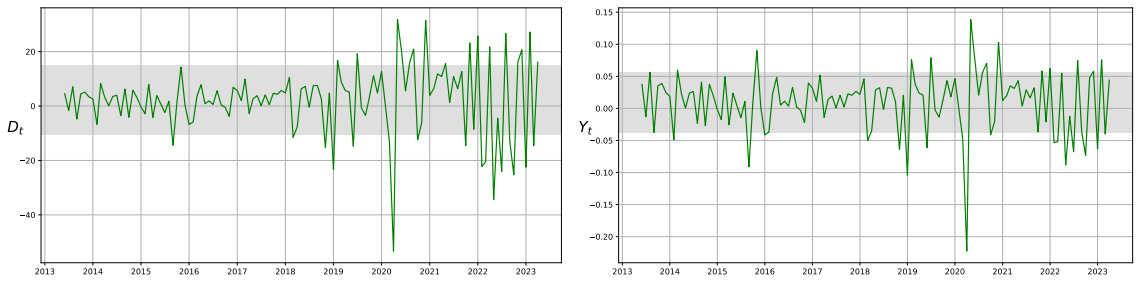
AIC

Another way to select the best order for an AR model is with the Akaike information criterion (AIC), which is a version of the likelihood that is penalized by the number of parameters. To understand the reasoning behind this criterion, we use the fact that the likelihood of a model increases by estimating more coefficients even if the additional coefficients do not contain any information about the response variable. The AIC is equal to $2(p+1) - 2\ell$, where p is the order of the model and ℓ is the log likelihood of the model. Models with higher ℓ have lower AICs. Two models with similar ℓ but different p will have different AICs; the model with higher p will have a higher AIC. The way we select the correct order of an AR model with AIC is we try fitting AR models of different orders and estimate the AIC for each of the models. Then, the model with the lowest AIC is considered to be the best AR model.

Data Analysis

We can hypothesize from looking at figure 1.1 that the Vanguard time series is not stationary as its mean and variance are growing over time. We conduct an augmented Dickey–Fuller (ADF) test, with $\alpha = 0.1$ to confirm this. The null hypothesis of the ADF test is that the time series is non-stationary. The test gives a p-value of 0.912274, which means we fail to reject the null. In other words, the Vanguard time series is non-stationary. Hence, we need to transform the data.

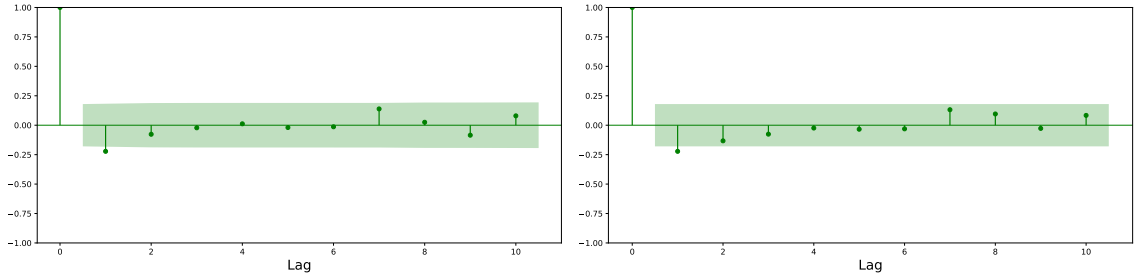
We refer to the price of the Vanguard 500 Index Fund at time t as V_t. In general, non-stationary time series can be transformed into stationary time series by differencing enough times. The time series $\{V_t | t = 1, 2, \ldots, 120\}$ can be transformed into a stationary process by differencing once. The differenced time series is $\{D_t = V_t - V_{t-1} | t = 2, 3, \ldots, 120\}$. Another way to transform $\{V_t\}$ into a stationary time series is by applying a log difference transformation. This will result in the stationary time series, $\{Y_t = \log(V_t) - \log(V_{t-1}) | t = 2, 3, \ldots, 120\}$. Both transformed time series $\{D_t\}$, and $\{Y_t\}$ are stationary according to the ADF test. We choose to model $\{Y_t\}$, because it leads to more linear temporal dependencies. Log difference transformations tend to have this effect on financial time series, because these time series tend to change exponentially (this means their rate of change is proportional to their value). Furthermore, from figure 1.4, it is apparent that $\{D_t\}$ has higher variance after 2021 than before 2020. The log difference transformation, $\{Y_t\}$, does not have this problem to the same degree.



(a) First Difference of V_t **(b)** Log difference of V_t

Figure 1.4 Note: the region within one standard deviation of mean is shaded.

Now that $\{Y_t\}$ is known to be stationary, we can model this with as an AR process. First, we want to figure out the correct order p for the optimal AR model. We will do this by finding p such that for all $h > p$, $\alpha(h)$ is close enough to 0 to be considered negligible. To test $\alpha(h)$, we construct a hypothesis test for each h, using the fact that for all $h > p$ in an AR(p) process, $\hat{\alpha}(h) \sim \mathcal{N}(0, 1/(n-1-h))$, where $n = 119$ for the time series $\{Y_t\}$. For each h, we conclude that $\alpha(h)$ is significantly different from 0, if $|\hat{\alpha}(h)| > 1.96/\sqrt{n-1-h}$. Figure 1.4 shows that the only significant PACF is at lag 1, which suggests that the best AR model will be of order 1.



(a) ACF of Y_t **(b)** PACF of Y_t

Figure 1.5 Note: significance level is at 0.05.

We fit AR(0), AR(1), and AR(2) and compute their AICs. An AR(0) is just a model where we estimate the mean of the time series. The computed AICs are shown below. The AR(1) model seems to have a slightly lower AIC than AR(0) model does. The AR(2) model has a much higher AIC than the other two models. According to AIC, our best model is the AR(1) model.

Order	AIC
0	-385.6178
1	-385.7001
2	-381.5670

Table 1.1

Both the PACF hypothesis test and the AIC model selection method suggests that the

best AR model is of order 1. The estimated AR(1) model is:

$$Y_t = 0.0114 - 0.2230 Y_{t-1} + \varepsilon_t. \tag{1.5}$$

1.2.2 Multivariate AR processes

In this section, we introduce the vector auto-regression (VAR) model and the concept of stationarity for multivariate time series. A general VAR(p) for r different time series takes the form:

$$\underset{r\times 1}{\boldsymbol{X}_t} = \underset{r\times 1}{\boldsymbol{\varepsilon}_t} + \underset{r\times 1}{\boldsymbol{c}} + \sum_{i=1}^{p} \underset{r\times r}{\boldsymbol{\Phi}_i} \underset{r\times 1}{\boldsymbol{X}_{t-i}}, \tag{1.6}$$

where $\boldsymbol{X}_t$ is the $(r \times 1)$ vector of values of the r different time series at time t, the $\boldsymbol{\Phi}_i$'s are the $(r \times r)$ matrices of coefficients, $\boldsymbol{c}$ is the $(r \times 1)$ vector of intercepts for the r time series, and $\boldsymbol{\varepsilon}_t$ is the vector of errors at time t such that $\mathbb{C}\mathrm{ov}(\boldsymbol{\varepsilon}_t, \boldsymbol{\varepsilon}_t) = \boldsymbol{\Sigma}$ and when $s \neq t$, $\mathbb{C}\mathrm{ov}(\boldsymbol{\varepsilon}_s, \boldsymbol{\varepsilon}_t) = \mathbf{0}$. Notice that the coefficients ($\boldsymbol{\Phi}_i$) and the intercept ($\boldsymbol{c}$), do not depend on time. We refer to this as the stationarity assumption. Similar to the univariate time series case (discussed in section 1.2.1), stationarity means that all the first and second moments of the stochastic process are the same for all t. For a multivariate time series, the first moment is described with a $(r \times 1)$ vector $\boldsymbol{\mu}$ where each entry is the mean of a single time series. The second moments of a multivariate time series is best described with a matrix. The autocovariance matrix of a stationary multivariate time series $\{\boldsymbol{X}_t, t = 1, 2, \dots\}$ is:

$$\boldsymbol{\Gamma}(h) = \begin{bmatrix} \gamma_{11}(h) & \gamma_{12}(h) & \dots & \gamma_{1r}(h) \\ \gamma_{21}(h) & \gamma_{22}(h) & \dots & \gamma_{2r}(h) \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{r1}(h) & \gamma_{r2}(h) & \dots & \gamma_{rr}(h) \end{bmatrix}, \tag{1.7}$$

where $\gamma_{ij}(h) = \mathbb{C}\mathrm{ov}(\boldsymbol{X}_{i,t+h}, \boldsymbol{X}_{j,t})$ is the cross-covariance between the i-th time series and the h lagged value of the j-th time series. Similar to the case of a stationary univariate time series, the autocovariance of a stationary multivariate time series also does not depend on time. The autocorrelation matrix $\boldsymbol{\rho}(h)$ is a matrix in which the i, j-th element is $\rho_{ij}(h) =$

$\gamma_{ij}(h)/\sqrt{\gamma_{ii}(0)\gamma_{jj}(0)}$. The autocorrelation matrix can therefore be expressed as $\boldsymbol{\rho}(h) = \mathbf{V}^{-1/2}\boldsymbol{\Gamma}(h)\mathbf{V}^{-1/2}$, where $\mathbf{V}^{-1/2} = \text{diag}\{\gamma_{11}(0)^{-1/2}, \gamma_{22}(0)^{-1/2}, \ldots, \gamma_{rr}(0)^{-1/2}\}$.

ACV Matrix a of a VAR(1)

For a stationary VAR(1) process $\boldsymbol{X}_t = \boldsymbol{\Phi}\boldsymbol{X}_{t-1} + \boldsymbol{c} + \boldsymbol{\varepsilon}_t$, if we take the expectation of both sides while keeping in mind that $\mathbb{E}[\boldsymbol{X}_t] = \mathbb{E}[\boldsymbol{X}_{t-1}] = \boldsymbol{\mu}$, we obtain $\boldsymbol{\mu} = \boldsymbol{\Phi}\boldsymbol{\mu} + \boldsymbol{c} \implies \boldsymbol{c} = (\mathbf{I}_r - \boldsymbol{\Phi})\boldsymbol{\mu}$. This implies that if the means of each of the time series in $\boldsymbol{X}_t$ are 0, then $\boldsymbol{c} = \mathbf{0}$. Now, let us assume that is the case, $\boldsymbol{\mu} = \mathbf{0}$. Our VAR(1) model can be written as $\boldsymbol{X}_t = \boldsymbol{\Phi}\boldsymbol{X}_{t-1} + \boldsymbol{\varepsilon}_t$.

The relationship between the autocovariance matrix $\boldsymbol{\Gamma}(h)$ and the coefficient matrix $\boldsymbol{\Phi}$ of this VAR(1) is given by the the recursive equation equation: $\boldsymbol{\Gamma}(h) = \boldsymbol{\Gamma}(h-1)\boldsymbol{\Phi}'$. We can express the covariance matrix of $\boldsymbol{\varepsilon}_t$ in terms of the autocovariance matrix $\boldsymbol{\Sigma} = \boldsymbol{\Gamma}(0) - \boldsymbol{\Gamma}(1)'\boldsymbol{\Phi}'$. From the recursive relation, we obtain $\boldsymbol{\Gamma}(1) = \boldsymbol{\Gamma}(0)\boldsymbol{\Phi}'$. We can use these last two equations to obtain $\boldsymbol{\Phi}'$ and $\boldsymbol{\Sigma}$ from $\boldsymbol{\Gamma}(0)$ and $\boldsymbol{\Gamma}(1)$ and vice versa.

The condition of stationarity for a VAR(1) is equivalent to the condition that all eigenvalues of $\boldsymbol{\Phi}$ have modulus less than 1. The recurrence relation $\boldsymbol{\Gamma}(h) = \boldsymbol{\Gamma}(h-1)\boldsymbol{\Phi}'$ implies that for $h \geq 1$, $\boldsymbol{\Gamma}(h) = \boldsymbol{\Gamma}(0)\boldsymbol{\Phi}'^h$. Now, we can see that the condition of stationarity implies that as $h \to \infty$, the covariances in $\boldsymbol{\Gamma}(h)$ will approach 0.

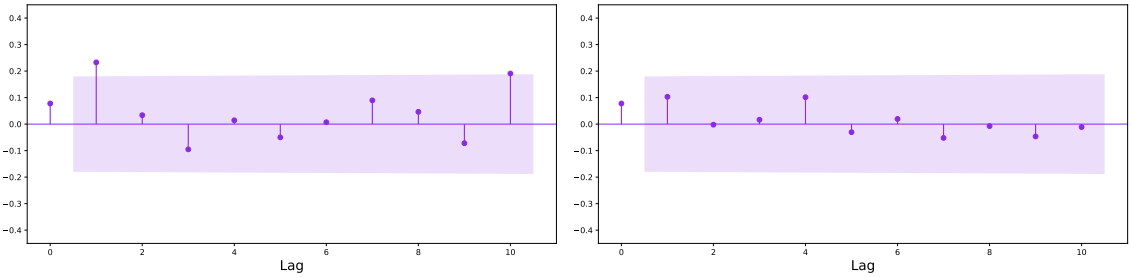
Data Analysis

In the case of multivariate time series, we can check for stationarity by checking for stability. We demonstrate this concept of stability using two of the time series that were introduced earlier. The two time series are: the Federal Funds Effective Rate (FFER) and the price of XLF fund. First, we take the log difference of the two time series. Then, we standardize the data and model the standardized data.

Standardizing an arbitrary time series $\{y_t\}$ with n observations means subtracting the estimated mean ($\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$) of the time series from every element of the series and then dividing by the estimated standard deviation ($\hat{\sigma}_y = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2}$). In short, the

Figure 1.6 Log Difference (Before Standardization),
note: the window cuts off when log difference of FFER is around -1.6, -1.9, -3.3, 1.4.

standardized version of $\{y_t\}$ is the time series $\{(y_t - \bar{y})/\hat{\sigma}_y\}$. We refer to the standardized log difference of XLF fund as $\{X_t\}$ and the standardized log difference of FFER as $\{F_t\}$. Since both time series are standardized, they are both at a similar scale. This will lead to the magnitude of the coefficients of VAR model being less contaminated by the scale and more representative of the strength of the associations they describe.



(a) Cross-correlations of F_t with Lags of X_t **(b)** Cross-correlations of X_t with Lags of F_t

Figure 1.7 Note: significance level is at 0.05.

The cross-correlation plots in figure 1.7 show that there is a significant correlation between F_t and X_{t-1}, which suggests a VAR(1) model. Another way to select the order for the VAR model is through AIC. We fitted and calculated the AIC of VARs of order zero through ten. Note that a VAR of order zero refers to just estimating the mean of each time

series, which is 0 since the two time series are standardized. The AICs of the first few orders are listed below, as the AIC only increases after this cutoff.

Order	0	1	2	3	4
AIC	-21.479	-21.520	-21.515	-21.516	-21.489

Table 1.2

The VAR of order one has the lowest AIC. The VAR(1) model is stated below:

$$\begin{bmatrix} F_t \\ X_t \end{bmatrix} = \begin{bmatrix} -0.1308 & 0.2441 \\ 0.1028 & 0.0025 \end{bmatrix} \begin{bmatrix} F_{t-1} \\ X_{t-1} \end{bmatrix} + \begin{bmatrix} \varepsilon_{F,t} \\ \varepsilon_{X,t} \end{bmatrix}, \tag{1.8}$$

where the vector with F_t and X_t will be referred to as the bold character $\boldsymbol{X}_t$, and the 2×2 matrix is the matrix of coefficients, which we refer to as $\boldsymbol{\Phi}$. The errors at time t associated with the two time series are $\varepsilon_{F,t}$ and $\varepsilon_{X,t}$ and the vector that includes them is $\boldsymbol{\varepsilon}_t$.

The estimated coefficients can be interpreted as the Granger causality. This is a type of causality defined though linear temporal dependence, i.e. the correlations between past values of some set of time series with the future values of some set of time series. In the case of our fitted model, we can tell from figure 1.7 (a) that the correlation $\mathbb{C}\mathrm{orr}(F_t, X_{t-1})$ is significantly different from 0. This means that the estimated coefficient that relates F_t to X_{t-1} is significant. We estimated that this coefficient is 0.2441, which means the log difference of XLF fund price from one month, causes the log difference of FFER of the next month. This relationship is positive, which implies that the Fed reacts to increases in the price of XLF fund by increasing the interest rates. It should be noted that we assumed that both our time series in figure 1.6 are stationary. In reality, this coefficient may not be representative of the true relationship between F_t and X_{t-1}, because the true relationship may be changing over time.

The model (1.8) can be written as $\boldsymbol{X}_t = \boldsymbol{\Phi}\boldsymbol{X}_{t-1} + \boldsymbol{\varepsilon}_t$. A VAR(1) process is stable, when all eigenvalues of its matrix of coefficients $\boldsymbol{\Phi}$ have modulus less than 1. To see why this is an important condition, consider the model (1.8) $\boldsymbol{X}_t = \boldsymbol{\Phi}\boldsymbol{X}_{t-1} + \boldsymbol{\varepsilon}_t$. It can be shown that $\boldsymbol{X}_t = (I + \boldsymbol{\Phi} + \boldsymbol{\Phi}^1 + \cdots + \boldsymbol{\Phi}^{t-1}) + \boldsymbol{\Phi}^t \boldsymbol{X}_0 + \sum_{i=0}^{t-1} \boldsymbol{\Phi}^i \boldsymbol{\varepsilon}_{t-i}$, by a recursive argument. As

$t \to \infty$, the right hand side of the above equation converges if all eigenvalues of $\mathbf{\Phi}$ have modulus less than 1.

The eigenvalues of $\mathbf{\Phi}$ are -0.1475 and 0.0192. Both eigenvalues have modulus less than 1. Hence, our VAR(1) is stable. Stability implies stationarity and therefore the model is also stationary.

1.3 Why time-varying parameters?

In section 1.2, the auto-regressive methods we introduced had time invariant parameters. This is to say, the coefficients that we estimated were assumed to be the same throughout time. It is often instructive to study time series data by assuming that they are non-stationary (time-varying). For example, Mikosch and Starica (2004) found an explanation for some common features of volatility found in financial time series by assuming that the time series are non-stationary. They gave the theoretical basis of a possible explanation for two stylized facts observed in return series: the long-range dependence (LRD) in volatility and the integrated GARCH (IGARCH) effect. They explained how both effects could be due to a plausible type of nonstationarity of the data: changes in the unconditional variance. Starica and Granger (2005) modelled the time series of daily returns of the S&P 500 stock over a 72 year period, by assuming non-stationarity. They used structural breaks in the time series and found that non-stationary models provided better forecasts than stationary models.

The VAR model in section 1.2.2 may not be the best way to model the data, because it assumes that the data is stationary. In section 1.1, we discussed some possible mechanism by which the two time series F_t and X_t may be related. However, there are many underlying macroeconomic factors unaccounted for in our model that may influence the auto-correlation and the cross-correlation of the two time series. Since this data spans 18 years, there are likely many underlying factors that changed over that time period which may have lead to a change in the correlation of the two time series F_t and X_t. To give a concrete example of an underlying factor, consider the fact the Fed has a lot more influ-

ence on the market during times of crisis than in times of stability. This is because part of their role is to steer the economy in a healthy direction, which they do by adjusting the interest rate F_t. During the 18 years that our data covers, there were two note-worthy times of crises. One of which is the 2007-2008 financial crisis and the other is the covid pandemic that started in 2020. Figure 1.6 shows that most of the extreme values for F_t and X_t occurred after the start of these crises. The mechanism discussed in this paragraph suggests that our model could benefit from allowing the parameters to be invariant to time.

One way to model time-varying parameters is to use piece-wise parameters. This corresponds to the assumption that there is a point in time after which the first and second order moments change and are completely unrelated to the moments from before that point in time. This method can be used to single out different time periods during which the time series can be assumed to have different characteristics than other time periods. Piece-wise parameters are useful when we can single out a point in time satisfying these assumptions. For example, it is easy to figure out during which months the drop in economic productivity due to the pandemic started. We are not always able to locate points in time like this. For example, it is not clear when exactly the economy stopped feeling the effects of the pandemic. This process of recovery is generally gradual. Piece-wise parameters do not work well when the characteristics of the time series changes gradually. When the first and second moments of our time series changes gradually, we can model it with locally stationary methods.

The are mainly two approaches to model time-varying parameters: *structural breaks* and *local stationarity*. The structural breaks models assume that the coefficients change abruptly right after the so-called *change points*. Aue and Horváth (2013) give an account of some of the recent work on structural breaks in time series models. The estimation is usually based on the popular cumulative sum, CUSUM. Both structural breaks in the mean as well as in the variance and covariance/correlation structure belong to this approach. CUSUM procedures are nonparametric by design.

To overcome the limitations discussed above, we work within the framework of local stationarity, which assumes instead that the parameters (mean, variance and autocovari-

ance) change *slowly* over time. This type of nonstationarity is the one we adopt in this thesis.

1.4 Locally stationary processes

In this section, we introduce the definition of locally stationary processes proposed by Dahlhaus (1996), which is based on the concept of evolutionary spectrum.

1.4.1 Locally Stationary Processes defined

Definition 1 *A sequence of stochastic processes* $X_n(t)$ $(t = 1, \ldots, n)$ *is called locally stationary with transfer function* A^0 *and trend* μ *if there exists a representation*

$$X_n(t) = \mu\left(\frac{t}{n}\right) + \int_{-\pi}^{\pi} \exp(i\omega t)\, A_n^0(t,\omega) dZ(\omega) \tag{1.9}$$

where the following holds.

(i) $Z(\omega)$ *is a stochastic process on* $[-\pi, \pi]$ *with* $\overline{Z(\omega)} = Z(-\omega)$ *and*

$$\operatorname{cum}\{dZ(\omega_1), \ldots, dZ(\omega_k)\} = \eta\left(\sum_{j=1}^{k} \omega_j\right) g_k(\omega_1, \ldots, \omega_{k-1})\, d\omega_1 \ldots d\omega_k,$$

where $\operatorname{cum}\{\cdots\}$ *denotes the cumulant of the* k*th order,* $g_1 = 0$, $g_2(\omega) = 1$, $|g_k(\omega_1, \ldots, \omega_{k-1})| \leq \operatorname{const}_k$ *for all* k *and* $\eta(\omega) = \sum_{j=-\infty}^{\infty} \delta(\omega + 2\pi j)$ *is the period* 2π *extension of the Dirac delta function.*

(ii) There exists a constant C *and a* $2\pi-$*periodic function* $A : [0,1] \times \mathbb{R} \to \mathbb{C}$ *with* $A(u, -\omega) = \overline{A(u,\omega)}$ *and*

$$\sup_{t,\omega} \left| A_n^0(t,\omega) - A\left(\frac{t}{n}, \omega\right) \right| \leq \frac{C}{n} \tag{1.10}$$

for all n*;* $A(u,\omega)$ *and* $\mu(u)$ *are assumed to be continuous in* u*.*

The smoothness of A in u guarantees that the process has (asymptotically) locally a stationary behavior. The function $f(u,\omega) = |A(u,\omega)|^2$ is called the time-varying spectral density of the process. Theorem 1 below (see Theorem 2.2 in Dahlhaus, 1996) proves that $f(u,\omega)$ is uniquely determined by the triangular array.

The above definition does not mean that a fixed continuous time process is discretized on a finer grid as n tends to infinity. If μ and A^0 do not depend on t and n then X does not depend on n as well and we obtain the spectral representation of a stationary process. Thus, the classical asymptotic theory for stationary processes is a special case of Dahlhaus' approach. There are similarities of Dahlhaus' definition to Priestley's definition of an oscillatory process (see Priestley, 1981, Chapter 11). However, there is a major difference and it is that Dahlhaus considers double indexed processes and he also makes use of asymptotic considerations. While Priestley's concern was a stochastic representation of the process itself, Dahlhaus' concern is mainly a representation which allows for a rigorous asymptotic treatment of statistical inference problems. One important consequence of this asymptotic approach is a uniqueness property of our spectral representation (proved by Theorem 1 below).

1.4.2 The Evolutionary Spectrum

The Wigner-Ville spectrum for fixed n (Martin and Flandrin, 1985) is

$$f_n(u,\omega) := \frac{1}{2\pi} \sum_{s=-\infty}^{\infty} \mathbb{C}\mathrm{ov}\left\{X_n\left([un - s/2]\right), X_n\left([un + s/2]\right)\right\} \exp(-i\omega s), \qquad (1.11)$$

where $X_n(s)$ is defined by (1.9) (with $A_n(t,\omega) = A(0,\omega)$ for $t < 1$ and $A_n^0(t,\omega) = A(1,\omega)$ for $t > n$). With the Theorem 1 below Dahlhaus proved that $f_n(u,\omega)$ tends in squared mean to $f(u,\omega) := |A(u,\omega)|^2$, the spectrum which corresponds to the spectral representation. Therefore we call $f(u,\omega)$ the time-varying spectral density of the process.

Theorem 1 *If $X_n(t)$ is locally stationary and $A(u,\omega)$ is uniformly Lipschitz continuous*

in both components with index $\alpha > \frac{1}{2}$*, then we have for all* $u \in (0,1)$

$$\int_{-\pi}^{\pi} |f_n(u,\omega) - f(u,\omega)|^2 \, d\omega = o(1). \tag{1.12}$$

Proof. We only give a brief sketch. We have

$$\int_{-\pi}^{\pi} |f_n(u,\omega) - f(u,\omega)|^2 \, d\omega = o(1) + \frac{1}{2\pi} \sum_{s=-\infty}^{\infty} |c_s|^2$$

with

$$c_s = \int_{-\pi}^{\pi} \exp(i\mu s) g\left(\frac{s}{2n}, \mu\right) d\mu$$

and

$$g\left(\frac{s}{2n}, \mu\right) = A\left(u + \frac{s}{2n}, \mu\right) A\left(u - \frac{s}{2n}, -\mu\right) - A\left(u, \mu\right) A\left(u, -\mu\right)$$

where $A(u,\mu) = A(0,\mu)$ for $u < 0$ and $A(u,\mu) = A(1,\mu)$ for $u > 1$. If we consider the decomposition

$$\sum_{s=-\infty}^{\infty} |c_s|^2 = \sum_{s=\ell}^{\infty} |c_{-s}|^2 + \sum_{s=1}^{\ell-1} |c_{-s}|^2 + \sum_{s=0}^{\ell-1} |c_s|^2 + \sum_{s=\ell}^{\infty} |c_s|^2$$

it can be show that

$$\sum_{s=-\infty}^{\infty} |c_s|^2 \propto \sum_{s=0}^{\ell-1} |c_s|^2 + \sum_{s=\ell}^{\infty} |c_s|^2 = O\left(\frac{\ell \ln \ell}{n^\alpha}\right) + O\left(\ell^{1-2\alpha}\right).$$

Choosing a suitable ℓ gives the result. □

Theorem 1 has an important consequence for the uniqueness of the spectral representation (1.9). It is well known (Priestley, 1981, Chapter 11.1) that the spectral representation (1.9) is not unique. However, Theorem 1 says that if there exists a spectral representation of the form (1.12) with a smooth $A(u,\omega)$, then $|A(u,\omega)|^2$ is uniquely determined from the whole triangular array (there may exist other nonsmooth representations). Furthermore, it is the limit of the Wigner-Ville spectrum (with the asymptotics in Dahlhaus, 1996). Since $\mu(u)$ is the mean of the process it is also uniquely determined.

Inspection of the above proof shows that only the values of $X_n(t)$ in the time interval

$$\frac{t}{n} \in \left[u - \frac{\ell}{n}, u + \frac{\ell}{n}\right]$$

contribute to $f(u, \omega)$. Since the length of this interval tends to zero, and $A(u, \omega)$ is smooth, the observations become "asymptotically stationary" on this interval which leads to the above uniqueness. The requirement $\ell \ln \ell / n^\alpha \to 0$ defines in some sense the interval on which the observations can be considered as stationary.

This approach describes mathematically very well what people mean when they speak of the spectrum at a timepoint t_0 of a nonstationary process $X(1), \ldots, X(n)$. Since the process is nonstationary only a few points around t_0 may have the same spectral structure. It is clear that the probability structure of these few points does not specify a spectral density uniquely. This is only guaranteed by an infinite number of observations. Dahlhaus approach says that $f(u, \omega) = |A(u, \omega)|^2$ is the spectral density if one had infinitely many observations of the same kind at a fixed time point.

1.4.3 Examples of Locally Stationary Processes

Time-modulated Stationary Processes

Let $Y(t)$ is a stationary process with spectral representation

$$Y(t) = \int_{-\pi}^{\pi} \exp(i\omega t) A(\omega) dZ(\omega)$$

and $\mu, :[0, 1] \to \mathbb{R}$ are continuous. Then

$$X_n(t) = \mu\left(\frac{t}{n}\right) + \sigma\left(\frac{t}{n}\right) Y(t)$$

is locally stationary with $A_n^0(t, \omega) = A\left(\frac{t}{n}, \omega\right) = \sigma\left(\frac{t}{n}\right) A(\omega)$. If $Y(t)$ is an $AR(2)-$process with complex roots close to the unit circle, then $Y(t)$ shows a periodic behaviour and σ may be regarded as a time-varying amplitude function of the process $X_n(t)$. If n tends to infinity more and more cycles of the process with $u = \frac{t}{n} \in [u_0 - \delta_n, u_0 + \delta_n]$, that is, with amplitude close to $\sigma(u_0)$ are observed.

Evolutionary Linear Processes

Let $\varepsilon(t) \overset{\text{iid}}{\sim} (0, \varepsilon)$ and

$$X_n(t) = \sum_{k=0}^{\infty} \psi_k \left(\frac{t}{n}\right) \varepsilon(t-k). \tag{1.13}$$

Then $X_n(t)$ is locally stationary with

$$A_n^0(t,\omega) = A\left(\frac{t}{n}, \omega\right) = \frac{\varepsilon}{\sqrt{2\pi}} \sum_{k=0}^{\infty} \psi_k \left(\frac{t}{n}\right) \exp(-i\omega k).$$

Kim (1999, 2012) provided examples of evolutionary time series models in Econometrics, e.g.: Evolutionary $VAR(p)$ Models, Unit-Root Process with Evolutionary Errors, Cointegration Models of $I(1)$ Processes with Evolutionary Errors.

Evolutionary $AR(p)$ Process

Consider the following system of difference equations:

$$\sum_{j=0}^{p} a_j \left(\frac{t}{n}\right) \left(X_n(t-j) - \mu\left(\frac{t-j}{n}\right)\right) = \sigma\left(\frac{t}{n}\right) \varepsilon(t) \tag{1.14}$$

where $\varepsilon(t) \overset{\text{iid}}{\sim} (0,1)$ and $a_0(u) \equiv 1$. We assume that $\sigma(u)$ and the $a_j(u)$'s are continuous on $\mathbb{R}$ with

$$\begin{aligned} \sigma(u) = \sigma(0), \quad a_j(u) = a_j(0) \quad \text{for} \quad u < 0 \\ \sigma(u) = \sigma(1), \quad a_j(u) = a_j(1) \quad \text{for} \quad u > 1 \end{aligned}$$

and differentiable for $u \in (0,1)$ with bounded derivatives. Then $X_n(t)$ is locally stationary with

$$\begin{aligned} A\left(\frac{t}{n}, \omega\right) &= \frac{\sigma\left(\frac{t}{n}\right)}{\sqrt{2\pi}} \left(1 + \sum_{j=1}^{p} a_j \left(\frac{t}{n}\right) \exp(-i\omega j)\right)^{-1} \\ A_n^0(t,\omega) &= \frac{1}{\sqrt{2\pi}} \sum_{k=0}^{\infty} \psi_{k,n}(t) \exp(-i\omega k). \end{aligned}$$

For example, in the case $p = 1$ we have $\psi_{j,n}(t) = (-1)^k \left\{\Pi_{j=0}^{k-1} a_1\left(\frac{t-j}{n}\right)\right\} \sigma\left(\frac{t-k}{n}\right) \exp(-i\omega k)$. The following Theorem of Dahlhaus (1996) gives the local stationarity of $X_{t,n}$ (see Theorem 2.3 in Dahlhaus, 1996).

Theorem 2 *Assume that* $\sum_{j=0}^{p} a_j(u) z^j \neq 0$ *for all* $|z| \leq 1 + c$ *with* $c > 0$ *uniformly in* u *and the coefficient functions* $a_j(u)$ *are continuous on* $\mathbb{R}$. *Then the difference equations (1.14) have a solution of the form (1.9) (that is (1.10) holds) with*

$$\begin{aligned} A(u,\omega) &= \frac{\sigma(u)}{\sqrt{2\pi}} \left(1 + \sum_{j=1}^{p} a_j(u) \exp(-i\omega j)\right)^{-1} \\ f(u,\omega) &= \frac{\sigma^2(u)}{2\pi} \left|\sum_{j=0}^{p} a_j(u) \exp(i\omega j)\right|^{-2}. \end{aligned}$$

An $AR(p)$ provides a good example of why we need the condition (1.10) instead of $A_n^0(t,\omega) = A\left(\frac{t}{n},\omega\right)$. In the $AR(1)$ case with $\sigma(u) \equiv 1$, the coefficients of the $MA(\infty)$ representation are $\psi_{k,n}(t) = \Pi_{j=0}^{k-1} a_1\left(\frac{t-j}{n}\right)$ instead of the coefficients $\psi_k\left(\frac{t}{n}\right)$ of the evolutionary linear process in (1.13). In the stationary case they would be the same: $\psi_{k,n}(t) = \psi_k\left(\frac{t}{n}\right) = a_1^k\left(\frac{t}{n}\right) = a_1^k \; \forall t$. In the locally stationary case the equality $A_n^0(t,\omega) = A\left(\frac{t}{n},\omega\right)$ does not hold in a finite sample, but only asymptotically as required in (1.10).

Evolutionary $MA(q)$ Process

Let $X_n(t)$ be a moving average process with time-varying coefficients

$$X_n(t) = \mu\left(\frac{t}{n}\right) + \sum_{j=0}^{q} b_j\left(\frac{t}{n}\right) \sigma\left(\frac{t-j}{n}\right) \varepsilon(t-j)$$

where $\varepsilon(t)$ is a mean zero iid sequence with unit variance, and $b_0 \equiv 1$. Then $X_n(t)$ is locally stationary

$$X_n(t) = \mu\left(\frac{t}{n}\right) + \int_{-\pi}^{\pi} \exp(i\omega t) A_n^0(t,\omega) dZ(\omega)$$

with

$$A_n^0(t,\omega) = A\left(\frac{t}{n},\omega\right) := \frac{\sigma\left(\frac{t}{n}\right)}{\sqrt{2\pi}}\sum_{j=0}^{q} b_j\left(\frac{t}{n}\right)\exp(-i\omega j).$$

In the mixed $ARMA$ case we can combine the above arguments which leads to the following representation.

Evolutionary $ARMA(p,q)$ Processes

Let $X_n(t)$ be an autoregressive moving average process with time-varying coefficients, i.e. the solution of

$$\sum_{j=0}^{p} a_j\left(\frac{t}{n}\right)\left(X_n(t-j) - \mu\left(\frac{t-j}{n}\right)\right) = \sum_{j=0}^{q} b_j\left(\frac{t}{n}\right)\sigma\left(\frac{t-j}{n}\right)\varepsilon(t-j)$$

where $\varepsilon(t)$ is a mean zero iid sequence with unit variance, and $a_0(u) \equiv b_0 \equiv 1$. Then $X_n(t)$ is locally stationary

$$X_n(t) = \mu\left(\frac{t}{n}\right) + \int_{-\pi}^{\pi}\exp(i\omega t)A_n^0(t,\omega)dZ(\omega)$$

with time-varying spectral density

$$f(u,\omega) = \frac{\sigma^2(u)}{2\pi}\frac{\left|\sum_{j=0}^{q} b_j(u)\exp(i\omega j)\right|^2}{\left|\sum_{j=0}^{p} a_j(u)\exp(i\omega j)\right|^2}.$$

In this case we only have (1.10) instead of $A_n^0(t,\omega) = A\left(\frac{t}{n},\omega\right)$:

$$\sup_{t,\omega}\left|A_n^0(t,\omega) - A\left(\frac{t}{n},\omega\right)\right| = O\left(n^{-1}\right).$$

The above results are interesting since without this framework $f(u,\omega)$ as in Theorem 2 cannot be interpreted as the spectral density of a time varying AR process (see Mélard and Herteleer-de Schutter, 1989). The function $f(u,\omega)$ from Theorem 2 is usually called instantaneous spectrum of a time varying AR process. The definition was motivated by the relationship between a stationary AR process and the theoretical spectrum of the process (see Kitagawa and Gersch, 1985). Theorem 2 gives a theoretical justification for this definition.

1.4.4 Examples of Multivariate Locally Stationary Processes

Time-varying $MA(\infty)-$representations

There exists a close connection between the spectral representation in (1.9) and time-varying $MA-$representations. Let

$$\begin{aligned}\psi_{n,k}(t) &:= \int_{-\pi}^{\pi} A_n^0(t,\omega)\exp(i\omega k)d\omega,\\ \psi_k(u) &:= \int_{-\pi}^{\pi} A(u,\omega)\exp(i\omega k)d\omega\end{aligned}$$

and

$$\varepsilon(t) := \int_{-\pi}^{\pi}\exp(i\omega t)dZ(\omega)$$

(note that at k and are matrices; the Fourier transform is calculated componentwise). Then $\mathbb{E}\varepsilon(t) = 0$ and $\mathbb{E}\varepsilon(s)\varepsilon'(t) = 2\pi I_r$, that is the $\varepsilon(t)$ are uncorrelated (independent in the Gaussian case, see for example Definition 2.1 in Dahlhaus, 2000). Since

$$A_n^0(t,\omega) = \frac{1}{2\pi}\sum_{k=-\infty}^{\infty}\psi_{n,k}(t)\exp(-i\omega k)$$

and

$$A(u,\omega) = \frac{1}{2\pi}\sum_{k=-\infty}^{\infty}\psi_k(u)\exp(-i\omega k)$$

we obtain

$$X_n(t) = \mu\left(\frac{t}{n}\right) + \frac{1}{2\pi}\sum_{k=-\infty}^{\infty}\psi_{n,k}(t)\varepsilon(t-k). \tag{1.15}$$

Condition (1.10) implies

$$\sup_{t,k}\left|\left\{\psi_{n,k}(t) - \psi_k\left(\frac{t}{n}\right)\right\}_{ij}\right| = O\left(n^{-1}\right)$$

for all $i,j = 1,\ldots,d$. If we start conversely with an infinite $MA-$representation (1.15) where the coefficients fulfill

$$\sup_{t}\sum_{k=-\infty}^{\infty}\left|\left\{\psi_{n,k}(t) - \psi_k\left(\frac{t}{n}\right)\right\}_{ij}\right| = O\left(n^{-1}\right) \tag{1.16}$$

for all $i, j = 1, \ldots, d$, then then it can be shown in the same way that a representation (1.9) exists and (1.10) is fulfilled. Note that heteroscedastic $\varepsilon(t)$, and $\varepsilon(t)$ with dependent components can be included by choosing other $\psi_{n,k}(t)$ in (1.15). The complicated construction with different functions $A_n^0(t, \omega)$ and $A\left(\frac{t}{n}, \omega\right)$ ($\psi_{n,k}(t)$ and $\psi_k\left(\frac{t}{n}, \omega\right)$, respectively) is necessary since we need on the one hand a certain smoothness in time direction (guaranteed by the functions $A(u, \omega)$ and $\psi_k(u)$) and on the other hand a class which is rich enough to cover interesting applications. For example, the time varying $AR(1)-$process

$$X_n(t) = a\left(\frac{t}{n}\right) X_n(t-1) + \varepsilon(t)$$

does not have a solution of the form

$$X_n(t) = \sum_{k=0}^{\infty} \psi_k\left(\frac{t}{n}\right) \varepsilon(t-k)$$

but only of the form

$$X_n(t) = \sum_{k=0}^{\infty} \psi_{n,k}\left(t\right) \varepsilon(t-k)$$

with (1.16) where $\psi_k(u) = a^k(u)$.

1.5 Outline of the thesis

The purpose of this thesis is the build up to the application of LS-VAR(1) to analyze financial time series. We do this by first introducing some theory behind the standard VAR(1) as well as ways to estimate its coefficients with OLS. We verify the theory with simulations in subsection 2.3. Then, in chapter 3, we explain some methods for estimating coefficients of different types of Locally Stationary VAR(1) processes, which were introduced by Motta (2021). Then, we verify these methods with simulation. Finally, we apply these methods to study some financial time series.

Chapter 2

VAR modeling for Financial Time Series

2.1 Model definition

A general vector auto-regression of order p can be expressed as was done in (1.6). If instead of modelling the multivariate time series $\{\boldsymbol{X}_t | t \in \mathbb{Z}\}$, we model the centered time series $\{\boldsymbol{X}_t - \boldsymbol{\mu} | t \in \mathbb{Z}\}$, we obtain:

$$\underset{r\times 1}{(\boldsymbol{X}_t} - \underset{r\times 1}{\boldsymbol{\mu}}) = \underset{r\times 1}{\boldsymbol{\varepsilon}_t} + \sum_{i=1}^{p} \underset{r\times r}{\boldsymbol{\Phi}_i} (\underset{r\times 1}{\boldsymbol{X}_{t-i}} - \underset{r\times 1}{\boldsymbol{\mu}}). \tag{2.1}$$

Every VAR(p) process can be rewritten as a VAR(1) process. To represent (2.1) as a VAR(1), we define:

$$\underset{rp\times 1}{\boldsymbol{Y}_t} = \begin{bmatrix} \boldsymbol{X}_t - \boldsymbol{\mu} \\ \boldsymbol{X}_{t-1} - \boldsymbol{\mu} \\ \boldsymbol{X}_{t-2} - \boldsymbol{\mu} \\ \vdots \\ \boldsymbol{X}_{t-p+1} - \boldsymbol{\mu} \end{bmatrix}, \underset{rp\times rp}{\boldsymbol{\Phi}} = \begin{bmatrix} \boldsymbol{\Phi}_1 & \boldsymbol{\Phi}_2 & \dots & \boldsymbol{\Phi}_{p-1} & \boldsymbol{\Phi}_p \\ I_r & 0 & \dots & 0 & 0 \\ 0 & I_r & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & I_r & 0 \end{bmatrix}, \underset{rp\times 1}{\boldsymbol{e}_t} = \begin{bmatrix} \boldsymbol{\varepsilon}_t \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{2.2}$$

Then, equation (2.1) can be rewritten as $\boldsymbol{Y}_t = \boldsymbol{\Phi}\boldsymbol{Y}_{t-1} + \boldsymbol{e}_t$. We can recursively define $\boldsymbol{Y}_i$ as $\boldsymbol{\Phi}\boldsymbol{Y}_{i-1} + \boldsymbol{e}_i$ to obtain the following equations: $\boldsymbol{Y}_t = \boldsymbol{\Phi}\boldsymbol{Y}_{t-1} + \boldsymbol{e}_t = \boldsymbol{\Phi}(\boldsymbol{\Phi}\boldsymbol{Y}_{t-2} + \boldsymbol{e}_{t-1}) + \boldsymbol{e}_t =$

$\boldsymbol{\Phi}^2\boldsymbol{Y}_{t-2}+\boldsymbol{\Phi}\boldsymbol{e}_{t-1}+\boldsymbol{e}_t = \boldsymbol{\Phi}^3\boldsymbol{Y}_{t-3}+\boldsymbol{\Phi}^2\boldsymbol{e}_{t-2}+\boldsymbol{\Phi}\boldsymbol{e}_{t-1}+\boldsymbol{e}_t = \cdots = \boldsymbol{\Phi}^m\boldsymbol{Y}_{t-m}+\sum_{i=0}^{m-1}\boldsymbol{\Phi}^i\boldsymbol{e}_{t-i}$. In short, we have $\boldsymbol{Y}_t = \boldsymbol{\Phi}^m\boldsymbol{Y}_{t-m} + \sum_{i=0}^{m-1}\boldsymbol{\Phi}^i\boldsymbol{e}_{t-i}$. The stationary condition is the same as the condition that $\det(\mathbf{I}_{rp} - \boldsymbol{\Phi}z) \neq 0$ for all $|z| \leq 1$. If the process is stationary then as $m \to \infty$, then $\boldsymbol{\Phi}^m \to 0$ and we obtain $\boldsymbol{Y}_t = \sum_{i=0}^{\infty}\boldsymbol{\Phi}^i\boldsymbol{e}_{t-i}$. This is the infinite moving average representation of a stationary VAR(p) process.

2.2 OLS Estimation

Suppose we have data $\{\boldsymbol{X}_t | t = 1, 2, \ldots, n\}$ from the VAR(1) process $\boldsymbol{X}_t = \boldsymbol{c}+\boldsymbol{\Phi}_1\boldsymbol{X}_{t-1}+\boldsymbol{\varepsilon}_t$ and we have to estimate the coefficients $\boldsymbol{c}$ and $\boldsymbol{\Phi}_1$ from this data. We can achieve this with ordinary least squares. We first make the following definitions:

$$\underset{(n-1)\times r}{\mathbf{Y}} = \begin{bmatrix} \boldsymbol{X}_2' \\ \boldsymbol{X}_3' \\ \vdots \\ \boldsymbol{X}_n' \end{bmatrix}, \underset{(n-1)\times(r+1)}{\mathbf{X}} = \begin{bmatrix} 1 & \boldsymbol{X}_1' \\ 1 & \boldsymbol{X}_2' \\ \vdots & \vdots \\ 1 & \boldsymbol{X}_{n-1}' \end{bmatrix}, \underset{(r+1)\times r}{\mathbf{B}} = \begin{bmatrix} \boldsymbol{c}' \\ \boldsymbol{\Phi}_1' \end{bmatrix}, \underset{(n-1)\times r}{\mathbf{e}} = \begin{bmatrix} \boldsymbol{\varepsilon}_2' \\ \boldsymbol{\varepsilon}_3' \\ \vdots \\ \boldsymbol{\varepsilon}_n' \end{bmatrix}. \quad (2.3)$$

Then, we can represent $\boldsymbol{X}_t = \boldsymbol{c} + \boldsymbol{\Phi}_1\boldsymbol{X}_{t-1} + \boldsymbol{\varepsilon}_t$ for all t as the linear model $\mathbf{Y} = \mathbf{X}\mathbf{B} + \mathbf{e}$. If we center $\mathbf{Y}$ and $\mathbf{X}$, then we no longer need an intercept term and our linear model simplifies to $\tilde{\mathbf{Y}} = \tilde{\mathbf{X}}\boldsymbol{\Phi}_1' + \boldsymbol{\varepsilon}_t$, where $\tilde{\mathbf{Y}} = [(\boldsymbol{X}_2 - \bar{\boldsymbol{X}}_{(0)}), (\boldsymbol{X}_3 - \bar{\boldsymbol{X}}_{(0)}), \ldots, (\boldsymbol{X}_n - \bar{\boldsymbol{X}}_{(0)})]'$, $\tilde{\mathbf{X}} = [(\boldsymbol{X}_1 - \bar{\boldsymbol{X}}_{(1)}), (\boldsymbol{X}_2 - \bar{\boldsymbol{X}}_{(1)}), \ldots, (\boldsymbol{X}_{n-1} - \bar{\boldsymbol{X}}_{(1)})]'$ and $\bar{\boldsymbol{X}}_{(0)} = 1/(n-1)\sum_{t=2}^{n}\boldsymbol{X}_t$ and $\bar{\boldsymbol{X}}_{(1)} = 1/(n-1)\sum_{t=1}^{n-1}\boldsymbol{X}_t$. We can estimate $\boldsymbol{\Phi}_1'$ by minimizing the sum of squared errors and our estimate is $\hat{\boldsymbol{\Phi}}_1' = (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}'\tilde{\mathbf{Y}}$. We can use the estimated $\hat{\boldsymbol{\Phi}}_1$ to estimate the intercept using the formula $\hat{\boldsymbol{c}} = (\mathbf{I} - \hat{\boldsymbol{\Phi}}_1)\hat{\boldsymbol{\mu}}$.

The residual vectors are $\hat{\boldsymbol{\varepsilon}}_t = (\boldsymbol{X}_t - \bar{\boldsymbol{X}}_{(0)}) - \hat{\boldsymbol{\Phi}}_1(\boldsymbol{X}_{t-1} - \bar{\boldsymbol{X}}_{(1)})$ for $t = 2, 3, \ldots, n$. The covariance matrix of errors $\boldsymbol{\Sigma} = \mathbb{C}\mathrm{ov}(\boldsymbol{\varepsilon}_t, \boldsymbol{\varepsilon}_t)$ can be estimated by the residual sum of squares matrix. The residual sum of squares matrix can be calculated as $\hat{\boldsymbol{\Sigma}} = (1/(n-r-2))\sum_{t=2}^{n}\hat{\boldsymbol{\varepsilon}}_t\hat{\boldsymbol{\varepsilon}}_t'$. We later use this to describe the distribution of the estimated coefficients. The estimated coefficients $\mathrm{vec}(\hat{\boldsymbol{\Phi}}_1')$ are unbiased estimators of $\mathrm{vec}(\boldsymbol{\Phi}_1')$. The estimates

$\text{vec}(\hat{\mathbf{\Phi}}_1')$ follow a multivariate normal distribution with covariance matrix $1/(n-1)(\mathbf{\Sigma} \otimes \mathbf{\Gamma}(0)^{-1})$, which can be consistently estimated as $\hat{\mathbf{\Sigma}} \otimes (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}$. $\mathbf{\Gamma}(0)$ is as defined in (1.7).

For a VAR(p) process as defined in (2.1), we can obtain a multivariate linear model $\tilde{\mathbf{Y}} = \mathbf{\Phi}_{(p)}'\tilde{\mathbf{X}} + \mathbf{e}$, where $\tilde{\mathbf{Y}} = [(\boldsymbol{X}_{p+1} - \bar{\boldsymbol{X}}_{(0)})), \ldots, (\boldsymbol{X}_n - \bar{\boldsymbol{X}}_{(0)}))]'$, $\mathbf{\Phi}_{(p)}' = [\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_p]$, and $\tilde{\mathbf{X}}$ is an $(n-p) \times pr$ matrix in which the t-th row is $[(\boldsymbol{X}_{t-1} - \bar{\boldsymbol{X}}_{(1)}))', \ldots, (\boldsymbol{X}_{t-p} - \bar{\boldsymbol{X}}_{(p)}))']$ for $t = p+1, \ldots, n$ with $\bar{\boldsymbol{X}}_{(i)} = 1/(n-p)\sum_{t=p+1}^{n} \boldsymbol{X}_{t-i}$. The estimate for $\mathbf{\Phi}_{(p)}$ is $\hat{\mathbf{\Phi}}_{(p)} = (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}'\tilde{\mathbf{Y}}$. The estimate for $\mathbf{\Sigma}$ is $\hat{\mathbf{\Sigma}} = (1/(n - p(1+r) - 1))\sum_{t=p+1}^{n} \hat{\boldsymbol{\varepsilon}}_{\boldsymbol{t}}\hat{\boldsymbol{\varepsilon}}_{\boldsymbol{t}}'$. The distribution of $\text{vec}(\hat{\mathbf{\Phi}}_{(\mathbf{p})})$ can be approximated as $\mathcal{N}(\text{vec}(\mathbf{\Phi}_{(\mathbf{p})}), \hat{\mathbf{\Sigma}} \otimes (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1})$.

2.3 Simulation results

In this section, we simulate data from a known VAR(1) model and test the methods and properties discussed in section 2.2. A known VAR(1) means we know all of the coefficients and all of the covariances of errors. The simulations we perform in this section will be from the following VAR(1) model:

$$\underbrace{\begin{bmatrix} X_{t,0} \\ X_{t,1} \end{bmatrix}}_{\boldsymbol{X}_t} = \underbrace{\begin{bmatrix} \phi_{00} & \phi_{01} \\ \phi_{10} & \phi_{11} \end{bmatrix}}_{\mathbf{\Phi}} \underbrace{\begin{bmatrix} X_{t-1,0} \\ X_{t-1,1} \end{bmatrix}}_{\boldsymbol{X}_{t-1}} + \underbrace{\begin{bmatrix} \varepsilon_{t,0} \\ \varepsilon_{t,1} \end{bmatrix}}_{\boldsymbol{\varepsilon}_t}, \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \underbrace{\begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}}_{\mathbf{\Sigma}}), \tag{2.4}$$

$$\phi_{00} = -0.3,\ \phi_{01} = 0.7,\ \phi_{10} = 0.4,\ \phi_{11} = 0.5$$

for $t = 0, 1, 2, \ldots, n-1$. In this subsection, everything is 0-indexed in order to be consistent with the code in the appendix. The eigenvalues of $\mathbf{\Phi}$ are approximately -0.56 and 0.76. Since all eigenvalues of $\mathbf{\Phi}$ have modulus less than 1, the VAR(1) is stationary. The simulations in shown in figure 2.1 look like stationary processes since their mean and variance seem to be the same throughout time.

Next, we simulate the process (2.4) with $n = 500$ and use the simulations to test the OLS method described in section 2.2. First, we estimate the coefficients from the simulated data and refer to the estimate as $\hat{\mathbf{\Phi}}$. For this, we used the formula $\hat{\mathbf{\Phi}}' = (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}'\tilde{\mathbf{Y}}$, where $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ are the demeaned versions of $\mathbf{X}$ and $\mathbf{Y}$ respectively. The estimated covariance

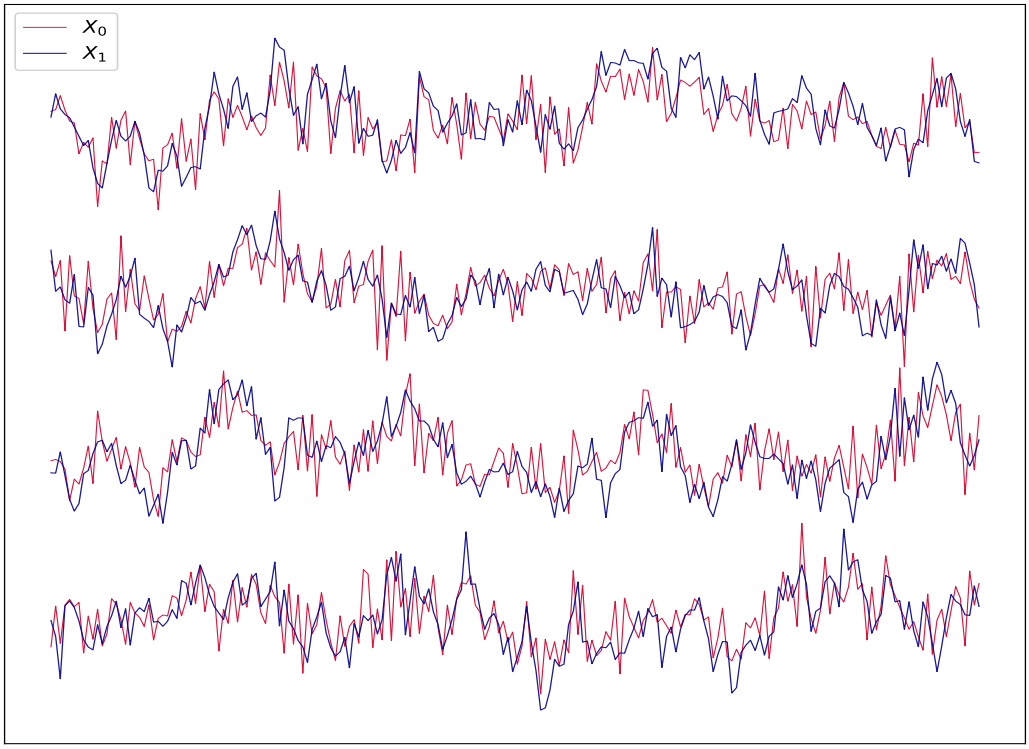


Figure 2.1 The VAR(1) process (2.4) is simulated 4 times, with $n = 200$.

matrix of errors $\hat{\mathbf{\Sigma}}$ can be computed as $(1/(500-2-2))(\sum_{t=1}^{499} \hat{\boldsymbol{\varepsilon}}_t \hat{\boldsymbol{\varepsilon}}_t')$, where the estimated errors are $\hat{\boldsymbol{\varepsilon}}_t = \mathbf{X}_t - \hat{\mathbf{\Phi}}\mathbf{X}_{t-1}$ for $t = 1, 2, \ldots, 499$. We use $\hat{\mathbf{\Sigma}}$ to estimate the covariance matrix of $\text{vec}(\hat{\mathbf{\Phi}}')$ as $\hat{\mathbf{\Sigma}} \otimes (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}$. The theoretical covariance matrix of $\text{vec}(\hat{\mathbf{\Phi}}')$ is $1/(500-1)(\mathbf{\Sigma} \otimes \mathbf{\Gamma}(0)^{-1})$. To compute the covariance matrix, we require $\mathbf{\Gamma}(0)$. To obtain this, we use the formula $\text{vec}(\mathbf{\Gamma}(0)) = [\mathbf{I} - (\mathbf{\Phi} \otimes \mathbf{\Phi})]^{-1}\text{vec}(\mathbf{\Sigma})$. This formula can be derived by substituting $\mathbf{\Gamma}(1) = \mathbf{\Gamma}(0)\mathbf{\Phi}'$ into $\mathbf{\Sigma} = \mathbf{\Gamma}(0) - \mathbf{\Gamma}(1)'\mathbf{\Phi}'$ and then using some properties of the $\text{vec}(\cdot)$ operation. We simulate 100 VARs each with $n = 500$ and use the average of the estimates $\hat{\mathbf{\Phi}}$ and $\hat{\mathbf{\Sigma}}$ from the 100 different samples to create figure 2.2.

We also investigate the evolution of the distribution of estimates as n increases. For this part, we first focus on the estimates for ϕ_{00}. First, we generate VAR processes with $n = 4, 6, 8, \ldots, 1000$ and estimate ϕ_{00} for each of those processes. To describe theoretically how the distribution of $\hat{\phi}_{00}$ over n, we need an expression for the variance with respect to n.

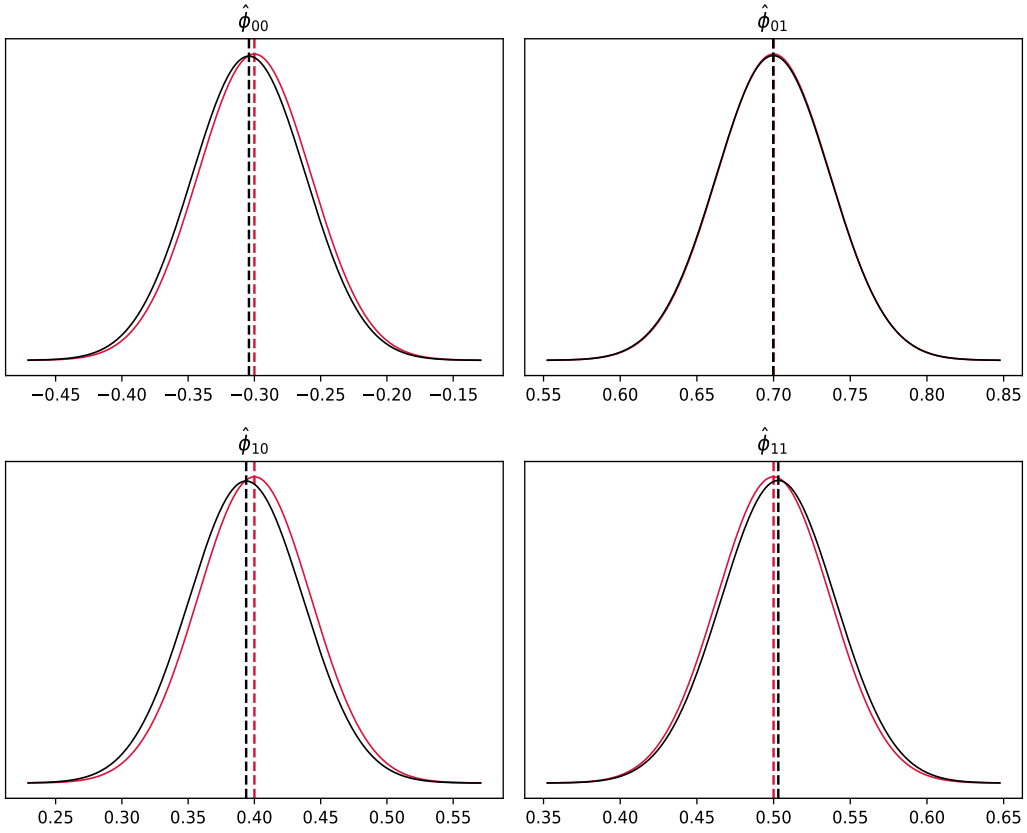


Figure 2.2 The distribution of the estimated coefficients $\hat{\mathbf{\Phi}}$. Red is according to $\mathcal{N}(\text{vec}(\mathbf{\Phi}), 1/(500-1)(\mathbf{\Sigma} \otimes \mathbf{\Gamma}(0)^{-1}))$. Black is according to $\mathcal{N}(\text{vec}(\hat{\mathbf{\Phi}}), \hat{\mathbf{\Sigma}} \otimes (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1})$.

That is $\sigma^2_{\hat{\phi}_{00}}(n)$; this value is equal to the top left element of the matrix $\mathbf{\Sigma} \otimes \mathbf{\Gamma}(0)^{-1}$ divided by $(n-1)$. In figure 2.3, we plot the lines $y = -0.3 \pm 3\sigma^2_{\hat{\phi}_{00}}(n)$. These lines are used to represent the region within 3 standard deviations of the theoretical mean. Therefore, we expect 99.7% of the estimates to fall within this region.

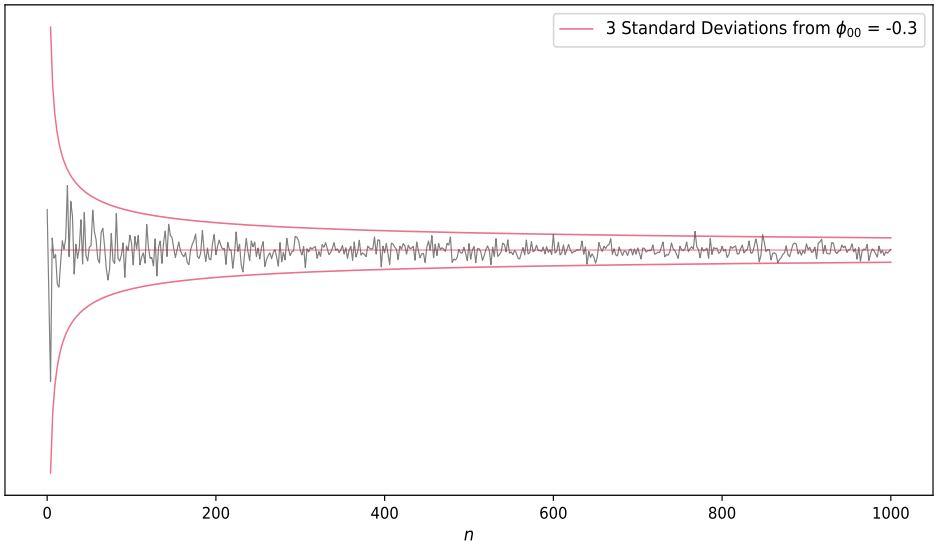


Figure 2.3 Distribution of $\hat{\phi}_{00}$ for $n = 3, 5, 7, \ldots, 999$.

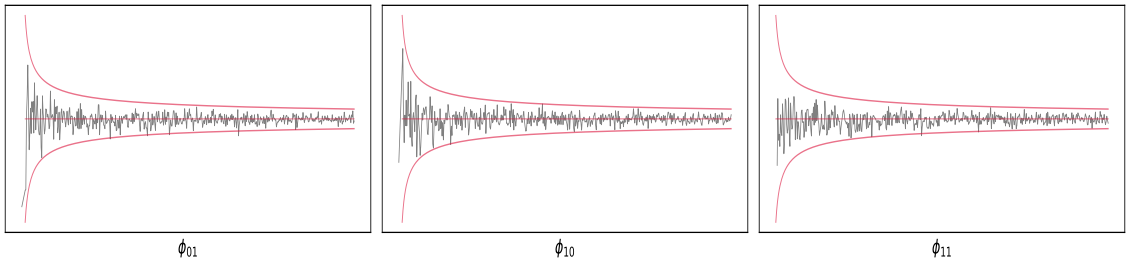


Figure 2.4 Distribution of $\hat{\phi}_{01}$, $\hat{\phi}_{10}$, and $\hat{\phi}_{11}$ as n increases.

Chapter 3

Locally-Stationary Vector Modelling

3.1 Introduction

A locally stationary VAR (LS-VAR) is an extension to the VAR discussed above. The difference is that in a LS-VAR, we assume that the coefficients are changing smoothly over time. In this chapter, we discuss methods of estimating the coefficients of an LS-VAR(1) process which were introduced by Motta (2021). An LS-VAR(1) process $\{\boldsymbol{X}_t | t = 1, 2, \ldots, n\}$ of r time series can be stated as:

$$\underset{r\times 1}{\boldsymbol{X}_t} = \underset{r\times r}{\boldsymbol{\Phi}}\left(\tfrac{t}{n}\right)\underset{r\times 1}{\boldsymbol{X}_{t-1}} + \underset{r\times 1}{\boldsymbol{\varepsilon}_t}\,, \tag{3.1}$$

where $\boldsymbol{X}_0 = \mathbf{0}$, $\mathbb{E}[\boldsymbol{\varepsilon}_t \boldsymbol{X}_t'] = \boldsymbol{\Gamma}_{\boldsymbol{\varepsilon}}$, $\boldsymbol{\varepsilon}_t$ is independent of all $\boldsymbol{X}_s$ for $t \neq s$, and $\boldsymbol{\Phi}(\cdot)$ is a smooth function of re-scaled time. This statement of the process does not allow for time-varying mean. Notice that all of the periods of training data, i.e. $t = 1, 2, \ldots, n$, correspond to input values of $\boldsymbol{\Phi}(u)$ in the interval $u \in [0, 1]$. The process $\boldsymbol{X}_t$ is locally stationary and causal if the largest eigenvalue of $\boldsymbol{\Phi}$ lies inside the unit circle:

$$\sup_{u\in(0,1)} |\mathrm{v}_1[\boldsymbol{\Phi}(u)]| < 1.$$

We also assume that $\boldsymbol{\Phi}(u)$ is differentiable and the derivative $\frac{d}{du}\boldsymbol{\Phi}(u)$ is bounded in the interval $u \in (0, 1)$. Motta (2021) showed that under these assumptions, $\boldsymbol{\Phi}(u)$ can be

estimated as:

$$\hat{\mathbf{\Phi}}(u) = \left[\sum_{t=1}^{n} \boldsymbol{X}_t \boldsymbol{X}'_{t-1} K_h(\tfrac{t}{n} - u)\right] \left[\sum_{t=1}^{n} \boldsymbol{X}_{t-1} \boldsymbol{X}'_{t-1} K_h(\tfrac{t}{n} - u)\right]^{-1}, \tag{3.2}$$

where $K_h(x) = \frac{1}{h} K(\frac{x}{h})$ and $K(\cdot)$ is a kernel function such that $K(\frac{x}{h}) = 0$ if $|x| > h$.

3.2 Estimation of LS-VAR with Time Varying Mean

In this section, we discuss an LS-VAR(1) process in which the mean also varies with time. Such a process can be represented as:

$$\underset{r\times 1}{\boldsymbol{X}_t} - \underset{r\times 1}{\boldsymbol{\mu}}(\tfrac{t}{n}) = \underset{r\times r}{\mathbf{\Phi}}(\tfrac{t}{n})[\underset{r\times 1}{\boldsymbol{X}_{t-1}} - \underset{r\times 1}{\boldsymbol{\mu}}(\tfrac{t}{n})] + \underset{r\times 1}{\boldsymbol{\varepsilon}_t}, \tag{3.3}$$

where $t = 1, 2, \ldots, n$, $\boldsymbol{X}_0 = \boldsymbol{\mu}(0) = \mathbf{0}$, $\mathbb{E}[\boldsymbol{\varepsilon}_t \boldsymbol{X}'_t] = \mathbf{\Gamma}_{\boldsymbol{\varepsilon}}$, $\boldsymbol{\varepsilon}_t$ is independent of all $\boldsymbol{X}_s$ for $t \neq s$, and $\mathbf{\Phi}(\cdot)$ is smooth. The process (3.3) is locally stationary and causal if the largest eigenvalue modulus of $\mathbf{\Phi}(u)$ in the interval $u \in (0, 1)$, is less than 1. After making the following definitions:

$$\underset{r\times(r+1)}{\mathbf{B}}(\tfrac{t}{n}) = [\boldsymbol{\mu}(\tfrac{t}{n}) - \mathbf{\Phi}(\tfrac{t}{n})\boldsymbol{\mu}(\tfrac{t-1}{n}), \mathbf{\Phi}(\tfrac{t}{n})], \qquad \underset{(r+1)\times 1}{\boldsymbol{Z}_t} = \begin{bmatrix} 1 \\ \boldsymbol{X}_t \end{bmatrix},$$

we can rewrite equation (3.3) as the following model:

$$\boldsymbol{X}_t = \mathbf{B}(\tfrac{t}{n})\boldsymbol{Z}_t + \boldsymbol{\varepsilon}_t. \tag{3.4}$$

We can estimate $\mathbf{B}(u)$ by minimizing a weighted least squares function:

$$\hat{\mathbf{B}}(u) = \underset{\mathbf{B}(u)}{\operatorname{argmin}} \sum_{t=1}^{n} \|\boldsymbol{X}_t - \mathbf{B}(\tfrac{t}{n})\boldsymbol{Z}_t\|^2 K_h(\tfrac{t}{n} - u), \tag{3.5}$$

where $\mathbf{B}(\frac{t_0}{n}) \approx \mathbf{B}(u_0)$ when t_0 is the largest integer that does not exceed $u_0 n$. This becomes a better approximation as $n \to \infty$, because $\sup(|\frac{t_0}{n} - u_0|) < \frac{1}{n}$. We also assume that $\mathbf{\Phi}(u)$ and $\boldsymbol{\mu}(u)$ are differentiable and the derivatives $\frac{d}{du}\mathbf{\Phi}(u)$ and $\frac{d}{du}\boldsymbol{\mu}(u)$ are bounded in

the interval $u \in (0,1)$. Motta (2021) showed that under these assumptions, local constant minimizer of (3.5) is:

$$\hat{\mathbf{B}}(u) = \mathbf{X}_1' \mathbf{K}_n(u) \boldsymbol{Z}_0 \left(\boldsymbol{Z}_0' \mathbf{K}_n(u) \boldsymbol{Z}_0\right)^{-1}, \tag{3.6}$$

if we define:

$$\underset{n\times r}{\mathbf{X}_1} = \begin{bmatrix} \underset{r\times 1}{\boldsymbol{X}_1} & \underset{r\times 1}{\boldsymbol{X}_2} & \dots & \underset{r\times 1}{\boldsymbol{X}_n} \end{bmatrix}',$$

$$\underset{n\times n}{\mathbf{K}_n(u)} = \operatorname{diag}\{K_h(\tfrac{1}{n} - u), K_h(\tfrac{2}{n} - u), \dots, K_h(1-u)\}.$$

We can obtain the local linear minimizer of the loss function (3.5) if in addition to the assumptions above, we also assume that the second derivatives $\frac{d^2}{d^2u}\boldsymbol{\Phi}(u)$ and $\frac{d^2}{d^2u}\boldsymbol{\mu}(u)$ are also bounded in the interval $u \in (0,1)$. Then, the local linear minimizer is:

$$\tilde{\mathbf{B}}(u) = \mathbf{X}_1' \mathbf{W}_n(u;\mathbf{X}) \mathbf{Z}_0 \left[\mathbf{Z}_0' \mathbf{W}_n(u;\mathbf{X}) \mathbf{Z}_0\right]^{-1}, \tag{3.7}$$

where

$$\underset{n\times(r+1)}{\mathbf{Z}_0} = [\mathbf{1}_n | \mathbf{X}_0], \text{ with } \mathbf{1}_n \text{ being an } n \times 1 \text{ vector of ones,}$$

$$\underset{n\times r}{\mathbf{X}_0} = \begin{bmatrix} \underset{r\times 1}{\boldsymbol{X}_0} & \underset{r\times 1}{\boldsymbol{X}_1} & \dots & \underset{r\times 1}{\boldsymbol{X}_{n-1}} \end{bmatrix}',$$

$$\mathbf{W}_n(u;\mathbf{X}) = \mathbf{K}_n(u) - \mathbf{K}_n(u)\boldsymbol{\Delta}_1(u)\mathbf{Z}_0 \left[\mathbf{Z}_0'\boldsymbol{\Delta}_1(u)\mathbf{K}_n(u)\boldsymbol{\Delta}_1(u)\mathbf{Z}_0\right]^{-1}\mathbf{Z}_0'\boldsymbol{\Delta}_1(u)\mathbf{K}_n(u),$$

$$\underset{n\times n}{\boldsymbol{\Delta}_1(u)} = \operatorname{diag}\{\tfrac{t}{n} - u, 1 \le t \le n\}.$$

From the estimate (3.7), and the formulas $\tilde{\mathbf{B}}(u) = [\tilde{\boldsymbol{\mu}}_1(u) - \tilde{\boldsymbol{\Phi}}(u)\tilde{\boldsymbol{\mu}}_0(u), \tilde{\boldsymbol{\Phi}}(u)]$, and $\tilde{\boldsymbol{\Phi}}(u) = \tilde{\mathbf{G}}(u,1)[\tilde{\mathbf{G}}(u,0)^{-1}]$, we obtain the following formula for the estimates:

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_0(u) &= \mathbf{X}_0'\mathbf{W}_n(u;\mathbf{X})\mathbf{1}_n / \mathbf{1}_n'\mathbf{W}_n(u;\mathbf{X})\mathbf{1}_n, \\
\tilde{\boldsymbol{\mu}}_1(u) &= \mathbf{X}_1'\mathbf{W}_n(u;\mathbf{X})\mathbf{1}_n / \mathbf{1}_n'\mathbf{W}_n(u;\mathbf{X})\mathbf{1}_n, \\
\tilde{\mathbf{G}}(u,0) &= \mathbf{X}_0'\mathbf{W}_n(u;\mathbf{X})\mathbf{X}_0 - \frac{\mathbf{X}_0'\mathbf{W}_n(u;\mathbf{X})\mathbf{1}_n\mathbf{1}_n'\mathbf{W}_n(u;\mathbf{X})\mathbf{X}_0}{\mathbf{1}_n'\mathbf{W}_n(u;\mathbf{X})\mathbf{1}_n}, \\
\tilde{\mathbf{G}}(u,1) &= \mathbf{X}_1'\mathbf{W}_n(u;\mathbf{X})\mathbf{X}_0 - \frac{\mathbf{X}_1'\mathbf{W}_n(u;\mathbf{X})\mathbf{1}_n\mathbf{1}_n'\mathbf{W}_n(u;\mathbf{X})\mathbf{X}_0}{\mathbf{1}_n'\mathbf{W}_n(u;\mathbf{X})\mathbf{1}_n}.
\end{aligned}$$

3.3 Simulation of LS-VAR(1)

In this section, we simulate LS-VAR(1) processes and test the estimation methods discussed in the last two subsections. Everything in this section will be 0-indexed in order to be consistent with the Python code in the appendix. First, we simulate LS-VAR(1) with time-invariant mean, $r = 3$ and time-varying coefficient $\mathbf{\Phi}(\frac{t}{n})$ with elements:

$$\begin{aligned}\phi_{i,j}(\tfrac{t}{n}) &= a_0 \frac{\sqrt{1+0.2i}}{\log(j+5)} \cos\left(\frac{6\pi}{\sqrt{j+5}}\frac{t}{n} - i\right), \text{when } i = 0,1,2 \text{ and } j = 0,1, \\ \phi_{i,j}(\tfrac{t}{n}) &= a_1 \sqrt{i+2} \sin\left(\frac{3\pi}{\log(i+2)}\frac{t}{n} - i\right), \text{when } i = 0,1,2 \text{ and } j = 2,\end{aligned} \tag{3.8}$$

where $a_0 = 0.4$ and $a_1 = 0.3$. This specification leads to the supremum of modulus of eigenvalues of $\mathbf{\Phi}(u)$ to be around 0.835 when $u \in (0,1)$. This ensures that the process (3.8) is locally stationary and causal. This LS-VAR(1) process is simulated $M = 100$ times with $n = 400$. The coefficients are estimated using the method described in section 3.1. The estimation is done with the Epanechnikov Kernel with $h = 0.03$.

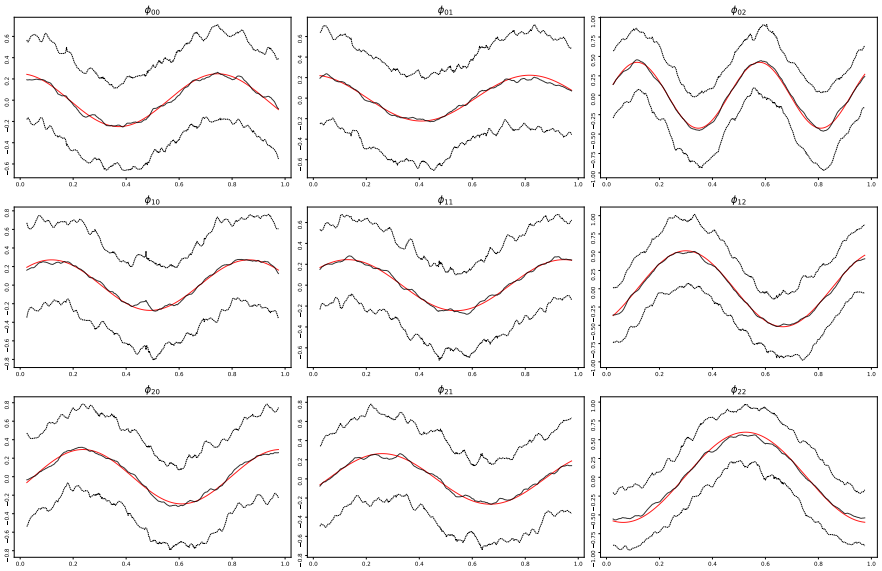


Figure 3.1 Estimation of 0-Mean LS-VAR(1) Parameters. Red lines: the coefficients according to (3.8). Black lines: the average estimates over the 100 replications and 2 standard deviations from the average estimates.

Then, we simulate LS-VAR(1) with time varying mean:

$$\mu_i(\tfrac{t}{n}) = \frac{\sin(\log(4-i)\pi\frac{t}{n})}{i+2}, \text{for } i = 0, 1, 2 \tag{3.9}$$

and the same time varying $\mathbf{\Phi}(\frac{t}{n})$ as in (3.8). In estimating the parameters with local linear approximation, we tried different combinations of Kernels and h's. The best estimates came from the Epanechinikov Kernel with $h = 0.06$. This large value of h resulted in high bias when the second derivative of the curve is large. This is noticeable in the plot for ϕ_{22} in figure 3.2. In general, the attempts with Gaussian Kernels resulted in more jagged confidence interval lines and the estimates with lower values of h resulted in estimates with spikes. As before, we simulated $M = 100$ replications of the LS-VAR(1) with time varying mean and $n = 400$.

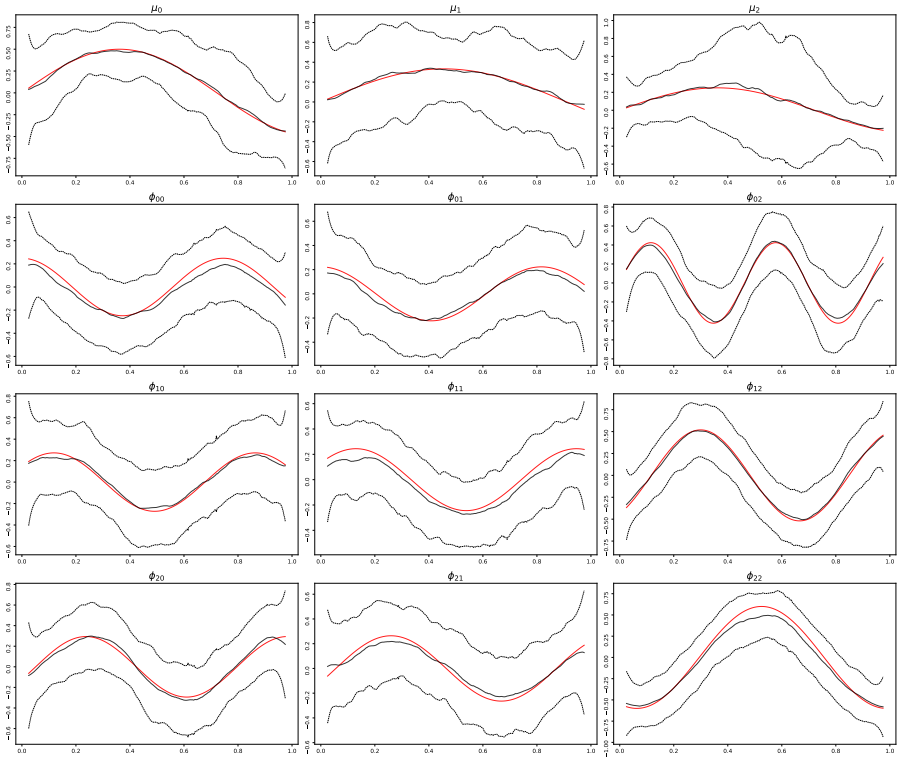


Figure 3.2 Estimation of LS-VAR(1) Parameters with Time Varying Mean using Local Linear Approximation. Red lines: the coefficients according to (3.8) and (3.9). Black lines: the average estimates over the 100 replications and 2 standard deviations from the average estimates.

3.4 Modelling Financial Time Series with LS-VAR(1)

Now, we return to the data we modelled in section 1.2.2 with VAR(1). We first model this data with structural breaks. We select breaks to separate times of crisis from the other time periods. One of the time periods we want to seperate is from December 2007 to December 2009, which represents the time period most affected by the 2007 financial crisis. We select a time period from February 2020 to December 2021, which is the time period for Covid-19. The time period after that is there to represents the period in which the FED tightened their monetary policies in order to fight inflation. The rest of the time periods will be referred to as the "regular" time period. Some of these time periods will have less than 30 data points if we use monthly data as we have done in the previous section. In order to have more samples, use daily data. The model for each time period is of the following form:

$$\begin{bmatrix} F_t \\ X_t \end{bmatrix} = \begin{bmatrix} \phi_{00} & \phi_{01} \\ \phi_{10} & \phi_{11} \end{bmatrix} \begin{bmatrix} F_{t-1} \\ X_{t-1} \end{bmatrix} + \begin{bmatrix} \varepsilon_{F,t} \\ \varepsilon_{X,t} \end{bmatrix}, \tag{3.10}$$

where F_t is the standardized log difference of the Federal funds exchange rate at time t and X_t is the standardized log difference of the price of XLF fund as explained in section 1.2.2.

In this paragraph, we interpret the structural break model results. Notice in top of figure 3.3 that ϕ_{00} is negative most of the time. This suggests that the Fed tends to adjust the interest rate by changing to the opposite direction of previous day's change. However, in times of crises, this relationship is much less significant. This is likely because in times of crises, the FED is adjusting interest rates based on other macroeconomic indicators. The coefficient ϕ_{01} on the other hand seems to be more significantly different from 0 during times of crises. This coefficient measures the causal dependence of the interest rate from the price of XTF. The pattern of ϕ_{01} suggests that the Fed are more active in responding to the XTF price changes with changes in the interest rate during times of crises. The coefficient ϕ_{10} is practically 0 at all times. This corresponds to the fact that it is difficult to forecast the price of an asset. The coefficient ϕ_{11} is also very close to 0 at most times. However, most of the time, it is negative. This suggests that the price of XTF tends moves

the opposite of the direction it moved in the previous day.

Next, we model this same data, but assume that the data follows an LS-VAR(1) process with time varying mean. For this fit, we settled on the Epanechinikov Kernel with $h = 0.1$. The largest eigenvalue modulus of estimated coefficient is around 0.458 in the input interval $(0, 1)$. The data is therefore locally stationary and causal. The LS-VAR fit at the bottom of figure 3.3 allows us to see more of the intricacies in the relationship between the two time series compared to the fit with structural breaks. For example, in the time period between 2012 and 2017, there are significant dips in the values of ϕ_{00} and ϕ_{01}. In the model with structural breaks, we did not notice this because we grouped this period of time with a larger time period and assumed that the coefficients are constant over that larger time period. The LS-VAR model helps us see that in the model with structural breaks, we made a good decision when we separated the times of crises from the other times. This is because almost all of the coefficients in bottom of figure 3.3 show that there is a significant change in the coefficients in the beginnings of these times of crises. The fitted time varying mean of the interest rate μ_0 shows that the interest rate tends to increase soon after times of crises as μ_0 is positive around middle of 2008 and in 2022. The fitted time varying mean on the log difference of the XTF price μ_1 is close to 0 at all times. This corresponds to the usual assumption on modelling stock prices, which is that the mean of the log difference can be assumed to be 0. This also supports the efficient-market hypothesis.

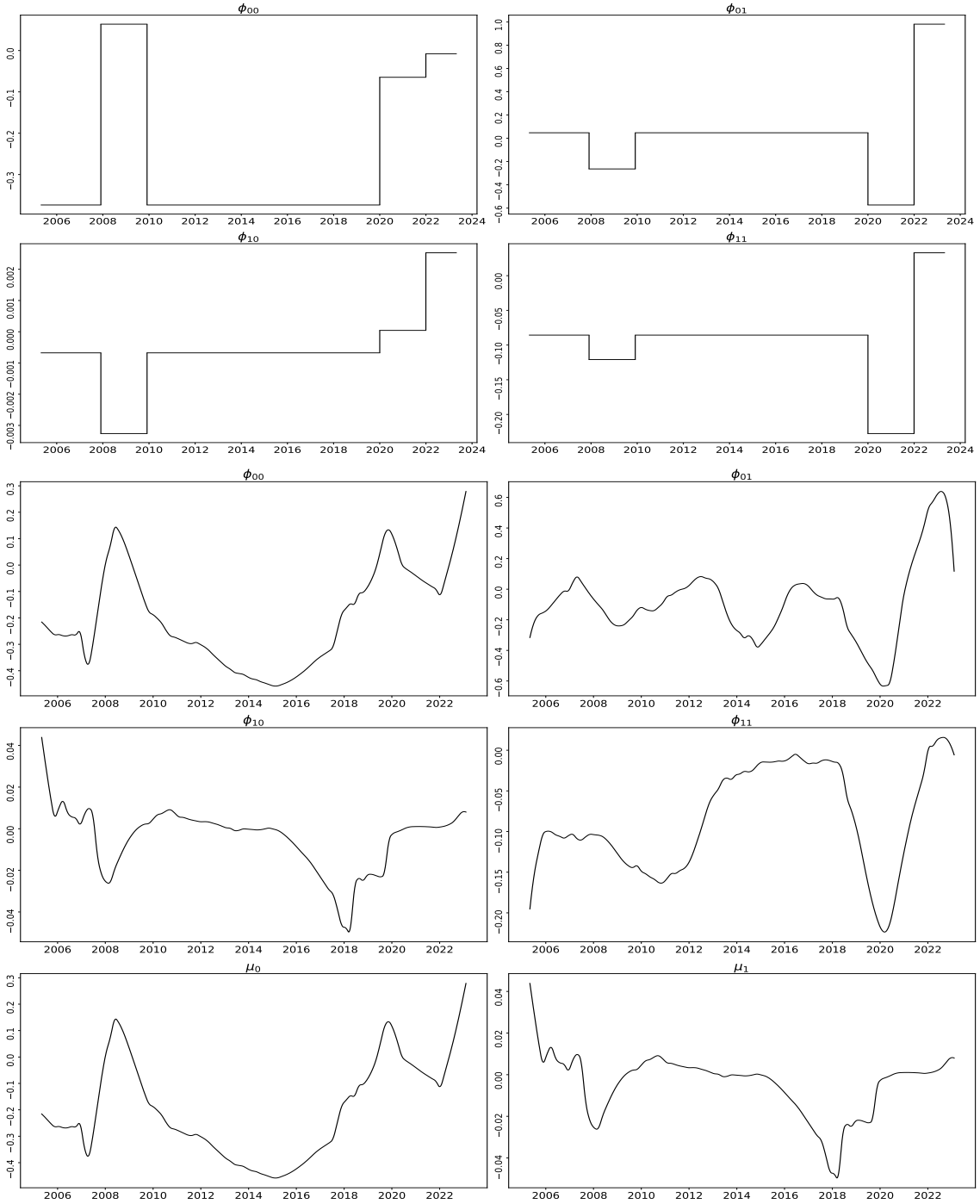


Figure 3.3 Top 2 rows: Structural breaks model of Federal Interest Rates & XTF Price.
Bottom 3 rows: LS-VAR(1) model of the same data.

Chapter 4

Conclusions and Future Research

In this thesis, we modelled various financial time series data with various auto-regression methods. The time series we modelled are the interest rate, the price of XTF fund, and the price of Vanguard index fund. We started by introducing the AR(p) process and used it to model the price of the Vanguard S&P 500 index fund. Then, we introduced the Multivariate AR process in subsection 1.2.2 and explained core concepts such as autocorrelation, stationarity, model selection, etc. We used VAR to model real world data and gave justification for why the data should instead be modelled as an LS-VAR process.

In chapter 2, we gave more details about the VAR method as we explained how to estimate the parameters with OLS and stated its asymptotic properties. Then, we verified the properties of OLS estimates through simulations in Python. Then in chapter 3, we introduced the LS-VAR(1) and explained how to estimate its parameters for when it has 0-mean and for when it has time-varying mean. We verified these estimation methods using simulations. Finally in subsection 3.4, we modelled the financial time series as LS-VAR(1) processes. In the future, it may be interesting to use the ideas relating to time varying parameters, to analyze how the coefficients in Capital Asset Pricing Models change over time.

Bibliography

Aue, A. and L. Horváth (2013). Structural breaks in time series. *Journal of time series Analysis 34*(1), 1–16.

Dahlhaus, R. (1996). On the kullback-leibler information divergence of locally stationary processes. *Stochastic Processes and their Applications 62*, 139–168.

Dahlhaus, R. (2000). A likelihood approximation for locally stationary processes. *The Annals of Statistics 28*(6), 1762–1794.

Kim, W. (1999). Econometric analyses of evolutionary time series. Humboldt Universität zu Berlin.

Kim, W. (2012). Nonparametric kernel estimation of evolutionary autoregressive processes. *Seoul Journal Economics 25*(4), 463–488.

Kitagawa, G. and W. Gersch (1985). A smoothness priors time-varying ar coefficient modeling of nonstationary covariance time series. *IEEE Trans. Automat. Control 30*, 48–56.

Mélard, G. and A. Herteleer-de Schutter (1989). Contributions to evolutionary spectral theory. *Journal of Time Series Analysis 10*(1), 41 – 63.

Mikosch, T. and C. Starica (2004). Nonstationarities in financial time series, the long-range dependence, and the igarch effects. *The Review of Economics and Statistics 86*(1), 378–389.

Motta, G. (2021). Joint mean-vector and var-matrix estimation for locally stationary var(1) processes. *arXiv:2104.11358v1*.

Priestley, M. B. (1981). *Spectral analysis and time series*. Academic Press.

Starica, C. and C. Granger (2005). Nonstationarities in stock returns. *The Review of Economics and Statistics 87*(3), 503–521.

Appendices

A Source Code

All the code required to produce the results of this paper is provided below. In order to reproduce the results, the codes for each subsection must be run starting from the beginning of that subsection, because the correct packages need to be imported, the correct data must be formatted in the correct way, and the correct functions need to be defined. The code is ordered such that everything will run correctly if the code is run in the order provided.

A.1 Code for Section 1.2.1

Import necessary libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
```

This code reads the data of the VOO stock into a pandas dataframe and modifies it to be monthly data.

```
tickerData = yf.Ticker('VOO')
voo = tickerData.history(period='1d', start='2013-5-1',
    end='2023-5-1')
voo = voo.Close
voo.index = voo.index.to_series().dt.date

df = voo.to_frame()
df.columns = ['voo']
df.index = pd.to_datetime(df.index)
df = df.resample('MS').first()
```

This code performs ADF test on different transformations of the VOO stock data.

```
from statsmodels.tsa.stattools import adfuller
def perform_adf_test(series):
    result = adfuller(series)
    print('ADF Statistic: %f' % result[0])
    print('p-value: %f' % result[1])

df['voo_log'] = np.log(df.voo/df.voo.shift(1))

perform_adf_test(df.voo.dropna())
perform_adf_test(df.voo.diff().dropna())
perform_adf_test(df.voo_log.dropna())
```

This produces Figure 1.3 (a).

```
plt.rcParams["figure.figsize"] = (10,5)
plt.plot(df.voo.diff(), color='green')
plt.ylabel("$D_t$", fontsize=19, rotation=0)

lower_bound = df.voo.diff().mean() - df.voo.diff().std()
upper_bound = df.voo.diff().mean() + df.voo.diff().std()
plt.axhspan(lower_bound, upper_bound, facecolor='silver',
    alpha=0.5, label='Shaded Region')

plt.grid()
plt.show()
```

This produces Figure 1.3 (b).

```
plt.rcParams["figure.figsize"] = (10,5)
plt.plot(df.voo_log, color='green')
plt.ylabel("$Y_t$", fontsize=19, rotation=0)
```

```
lower_bound = df.voo_log.mean() - df.voo_log.std()
upper_bound = df.voo_log.mean() + df.voo_log.std()
plt.axhspan(lower_bound, upper_bound, facecolor='silver',
    alpha=0.5, label='Shaded Region')

plt.grid()
plt.show()
```

Produces both plots in Figure 1.5

```
f, ax = plt.subplots(figsize=(10, 5))
plot_acf(df.voo_log.dropna(), lags=10, ax=ax)
# plt.title("ACF Plot of $Y_{t}$")
plt.title("")
# plt.grid()
plt.xlabel("Lag", fontsize=17)
plt.xlim(-0.5, 11)
plt.ylim(-.5, .5)
ax = plot_acf_colors(ax)

f, ax = plt.subplots(figsize=(10, 5))
plot_pacf(df.voo_log.dropna(), lags=10, ax=ax, alpha=0.05)
# plt.title("ACF Plot of $Y_{t}$")
# plt.grid()
plt.title("")
plt.xlabel("Lag", fontsize=17)
ax = plot_acf_colors(ax)
```

Some additional data analysis and plot.

```
plt.rcParams["figure.figsize"] = (9,7)
```

```
temp = df.copy()
temp['lag1'] = df.voo_log.shift(1)
temp = temp[['voo_log', 'lag1']]
temp.dropna(inplace=True)
plt.scatter(temp['lag1'], temp['voo_log'])
plt.title("Lag Plot", fontsize=17)
plt.xlabel(r"$V_{t-1}$", fontsize=15)
plt.ylabel(r"$V_{t}$   ", rotation=0, fontsize=15)
m, b = np.polyfit(temp['lag1'], temp['voo_log'], 1)
plt.plot(temp['lag1'], m*temp['lag1']+b, color='lightblue',
    label="Line of best fit")
# plt.plot(temp['lag1'], temp['lag1'], color='lightgrey',
    label='Line with slope = 1, intercept = 0')
plt.legend()
plt.show()
```

Fits the AR(1) model stated in equation (1.5).

```
import statsmodels.api as sm
mod = sm.tsa.AutoReg(df.voo_log.dropna(), lags=1)
result = mod.fit()
print(result.summary())
```

A.2 Code for Section 1.2.2 and plots for Section 1.1

Import necessary libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
```

```
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
from statsmodels.tsa.api import VAR
```

This code below reads in all of the necessary data and modifies them to be of the correct format. To get the interest rate data, you have to first download the data it from here: https://fred.stlouisfed.org/series/DFF

```
path = '/content/drive/MyDrive/Master_Thesis/'
interest = pd.read_excel(path+'longer_ffer.xls',
    skiprows=10, index_col = 'observation_date')

tickerData = yf.Ticker('XLF')
bank = tickerData.history(period='1d', start='2005-5-1',
    end='2023-4-15')
bank = bank.Close

interest.index = interest.index.to_series().dt.date
bank.index = bank.index.to_series().dt.date

df = pd.merge(bank, interest, left_index=True,
    right_index=True)
df.columns = ['bank', 'ffer']
df.index = pd.to_datetime(df.index)
df = df.resample('MS').first()



df['ffer_log'] = np.log(df.ffer/df.ffer.shift(1))
df['bank_log'] = np.log(df.bank/df.bank.shift(1))
```

This Produces Figure 1.2 and 1.3.

```
plt.rcParams["figure.figsize"] = (12,5)
plt.plot(df.bank, color='teal')
plt.ylabel("Price in Dollars", fontsize=11)
plt.grid()
plt.show()

plt.rcParams["figure.figsize"] = (12,5)
plt.plot(df.ffer, color='crimson')
plt.ylabel("Percent", fontsize=11)
plt.grid()
plt.show()
```

This Produces Figure 1.6.

```
ax = plt.axes()

plt.rcParams["figure.figsize"] = (14,5)
plt.plot(df.ffer_log, color='crimson', linewidth=.9,
    label='FFER', alpha=0.75)
plt.plot(df.bank_log, color='teal', linewidth=1.1,
    label='XLF Fund', alpha=.9)

plt.ylim(-.8, 1.2)
ax.set_yticks([-0.5, 0, 0.5, 1])
ax.set_yticklabels([-.5, 0, 0.5, 1])

plt.legend(fontsize=14)
plt.grid(alpha=0.5)
plt.show()
```

Some data analysis to look for combinations with significant correlation.

```
df.bank_log.autocorr(1), df.bank_log.autocorr(2),
    df.bank_log.autocorr(3)

for i in range(0, 8):
  print('cross_corr for lag', i, 'is: ',
    df.bank_log.corr(df.ffer_log.shift(i)))

for i in range(0, 8):
  print('cross_corr for lag', i, 'is: ',
    df.ffer_log.corr(df.bank_log.shift(i)))
```

Standardizes data.

```
standard = df[['ffer_log', 'bank_log']].copy().dropna()

standard.ffer_log = (standard.ffer_log -
    standard.ffer_log.mean())/standard.ffer_log.std()
standard.bank_log = (standard.bank_log -
    standard.bank_log.mean())/standard.bank_log.std()
```

Produces the plots in Figure 1.7.

```
from scipy.stats import t

def cross_corr(x, y, max_lag):
    # Ensure that the inputs are numpy arrays
    x = np.asarray(x)
    y = np.asarray(y)

    # Calculate the cross-correlation
    cross_corr_values = []
```

```
    for lag in range(0, max_lag + 1):
        if lag > 0:
            cross_corr_value = np.corrcoef(x[lag:],
                y[:-lag])[0, 1]
        else:
          cross_corr_value = np.corrcoef(x, y)[0, 1]
        cross_corr_values.append(cross_corr_value)

    return cross_corr_values

def plot_cross_corr(cross_corr_values, max_lag, alpha=0.05,
    color='violet'):

    plt.figure(figsize=(10, 5))
    lags = np.arange(0, max_lag + 1)
    markerline, stemlines, baseline = plt.stem(lags,
        cross_corr_values, use_line_collection=True,
        linefmt=color, basefmt=color)

    # You can change the line width here.
    plt.setp(stemlines, linewidth=1.4)

    # Customize the baseline
    plt.setp(baseline, linewidth=0)
    plt.axhline(y=0, color=color, label='Horizontal Line',
        linewidth=1)

    plt.xlabel('Lag', fontsize=17)
```

```
    # crit_value = 1.96 / np.sqrt(119)
    x_points = [i for i in range(1, 11)]
    x_points.append(10.5)
    x_points.insert(0, 0.5)

    y_points = [1.96 / np.sqrt(119-i) for i in range(1, 11)]
    y_points.append(1.96 / np.sqrt(119-10.5))
    y_points.insert(0, 1.96 / np.sqrt(119-0.5))

    # Add the significance region (alpha=0.05) to the plot
    plt.fill_between(x_points, [-n for n in y_points],
        y_points, facecolor=color, alpha=0.16)


    plt.xlim(-0.5, 11)
    plt.ylim(-.45, .45)

    plt.show()

cross_corr_values = cross_corr(standard.ffer_log,
    standard.bank_log, 10)
plot_cross_corr(cross_corr_values, 10, color='blueviolet')

cross_corr_values = cross_corr(standard.bank_log,
    standard.ffer_log, 10)
plot_cross_corr(cross_corr_values, 10, color='blueviolet')
```

Gets the different AICs that are in Table 1.2.

```
for i in range(11):
  print('AIC for order', i, 'is: ', model.fit(i).aic)
```

Fits the VAR model to get the model stated in equation (1.8) and checks its eigenvalues.

```
model = VAR(standard)
print(model.fit(1).summary())

print(np.linalg.eigvals(np.array([[-0.130832, 0.244065],
                                  [0.0102776, 0.002486]])))
```

A.3 Code for Section 2.3: Simulation of VAR

Import necessary files.

```
import numpy as np
from numpy import random
from numpy.random import multivariate_normal
import matplotlib.pyplot as plt
from scipy.stats import norm
```

Check that eigenvalues of the coefficient matrix have modulus less than 1.

```
phi = np.array([[-.3, .7],
                [.4, .5]])
np.linalg.eig(phi)
```

Define the function to generate samples from a VAR(1) process.

```
def generate_var(n, seed=0):
  if seed != 0:
    np.random.seed(seed)

  err_matrix = [[1, .5],
                [.5, 1]]
```

```
  data = multivariate_normal([0, 0], err_matrix, size=n)
  for i in range(1, n):
    data[i] = data[i] + [ -.3*data[i-1,0] + .7*data[i-1,1],
                          .4*data[i-1,0] + .5*data[i-1,1]]
  return data
```

Define the function to estimate VAR coefficients using OLS and to estimate the covariance matrix of errors.

```
def ols_var(data): #data.shape = (n, 2)
  x_0 = [data[:,0].mean(), data[:,1].mean()]
  x_1 = [data[1:,0].mean(), data[1:,1].mean()]

  tilde_y = (data - x_0)[1:,:]
  tilde_x = (data - x_1)[:-1,:]

  xtx_inv = np.linalg.inv(tilde_x.T @ tilde_x)
  xty = tilde_x.T @ tilde_y

  est_coef = (xtx_inv @ xty).T

  return (est_coef, xtx_inv)

def estimate_cov_errors(data, coef):
  n = len(data)
  sigma_hat = np.array([[0, 0],
                        [0, 0]], dtype=np.float64)

  for i in range(1, n):
    hat = (coef @ data[i-1].T).T
```

```
    error = data[i] - hat

    var_0 = error[0]*error[0]
    cov = error[0]*error[1]
    var_1 = error[1]*error[1]

    sigma_hat += np.array([[var_0, cov],
                           [cov, var_1]], dtype=np.float64)
  sigma_hat = sigma_hat/(n - 4)

  return sigma_hat
```

Generate Figure 2.1.

```
plt.rcParams["figure.figsize"] = (9,8)

data = generate_var(200)
plt.plot(data[:,0], color='darkgreen',
        linewidth=0.7, label='$X_{0}$')
plt.plot(data[:,1], color='hotpink',
        alpha=0.9, linewidth=0.8, label='$X_{1}$')

for i in range(1, 4):
  temp = generate_var(200)
  plt.plot(temp[:,0] - 8*i, color='darkgreen',
           linewidth=0.7)
  plt.plot(temp[:,1] - 8*i,color='hotpink',
           alpha=0.9, linewidth=0.8)

plt.legend(fontsize=13, framealpha=0.9)
```

```
plt.xticks([])
plt.yticks([]);
```

Generate data and estimates for Figure 2.2.

```
A = np.array([[-.3, .7],[.4, 0.5]])
#Below is the covariance matrix of the erros.
err_matrix = np.array([[1, .5],[.5, 1]])

Gamma_0 = ((np.linalg.inv(np.eye(4) - np.kron(A, A)) @
            (err_matrix.T.reshape(4,1))).reshape(2,2).T)

coef_cov = np.kron(err_matrix, np.linalg.inv(Gamma_0))/(500-1)



est_coef = np.zeros((2,2), dtype=np.float64)
xtx_inv = np.zeros((2,2), dtype=np.float64)
Sigma_hat = np.zeros((2,2), dtype=np.float64)

for i in range(100):
  curr_data = generate_var(500)

  curr_coef, curr_xtx_inv = ols_var(curr_data)
  curr_sigma_hat = estimate_cov_errors(curr_data, curr_coef)

  est_coef += curr_coef
  xtx_inv += curr_xtx_inv
  Sigma_hat += curr_sigma_hat

est_coef = est_coef/100
xtx_inv = xtx_inv/100
```

```
Sigma_hat = Sigma_hat/100
hat_coef_cov = np.kron(Sigma_hat, xtx_inv)
```

Generate Figure 2.2.

```
plt.figure(figsize=(9, 7))

# Subplot 1: a00
plt.subplot(2, 2, 1)

# Theoretical value
a, sigma = A[0,0], coef_cov[0,0]**(.5)
x = np.linspace(a - 4*sigma, a + 4*sigma, 1000)
y = norm.pdf(x, a, sigma)
plt.plot(x, y, color = 'crimson', linewidth = 1)
plt.axvline(x=a, color='crimson', linestyle='--')

# Estimated value
hat_a, hat_sigma = est_coef[0,0], hat_coef_cov[0,0]**0.5
y = norm.pdf(x, hat_a, hat_sigma)
plt.plot(x, y, color = 'black', linewidth = 1,
        label ='Estimated Distribution')
plt.axvline(x=hat_a, color='black', linestyle='--')

plt.title('$\hat \phi_{00}$', fontsize=12)
plt.yticks([])



# # Subplot 2: a01
plt.subplot(2, 2, 2)
```

```
# Theoretical value
a, sigma = A[0,1], coef_cov[1,1]**(.5)
x = np.linspace(a - 4*sigma, a + 4*sigma, 1000)
y = norm.pdf(x, a, sigma)
plt.plot(x, y, color = 'crimson', linewidth = 1)
plt.axvline(x=a, color='crimson', linestyle='--')

# Estimated value
hat_a, hat_sigma = est_coef[0,1], hat_coef_cov[1,1]**0.5
y = norm.pdf(x, hat_a, hat_sigma)
plt.plot(x, y, color = 'black', linewidth = 1)
plt.axvline(x=hat_a, color='black', linestyle='--')

plt.title('$\hat \phi_{01}$', fontsize=12)
plt.yticks([])



# # Subplot 3: a_10
plt.subplot(2, 2, 3)

# Theoretical value
a, sigma = A[1,0], coef_cov[2,2]**(.5)
x = np.linspace(a - 4*sigma, a + 4*sigma, 1000)
y = norm.pdf(x, a, sigma)
plt.plot(x, y, color = 'crimson', linewidth = 1)
plt.axvline(x=a, color='crimson', linestyle='--')
```

```
# Estimated value
hat_a, hat_sigma = est_coef[1,0], hat_coef_cov[2,2]**(0.5)
y = norm.pdf(x, hat_a, hat_sigma)
plt.plot(x, y, color = 'black', linewidth = 1)
plt.axvline(x=hat_a, color='black', linestyle='--')

plt.title('$\hat \phi_{10}$', fontsize=12)
plt.yticks([])



# # Subplot 4: a_11
plt.subplot(2, 2, 4)

# Theoretical value
a, sigma = A[1,1], coef_cov[3,3]**(.5)
x = np.linspace(a - 4*sigma, a + 4*sigma, 1000)
y = norm.pdf(x, a, sigma)
plt.plot(x, y, color = 'crimson', linewidth = 1,
        label ='Theoretical Distribution')
plt.axvline(x=a, color='crimson', linestyle='--')

# Estimated value
hat_a, hat_sigma = est_coef[1,1], hat_coef_cov[3,3]**0.5
y = norm.pdf(x, hat_a, hat_sigma)
plt.plot(x, y, color = 'black', linewidth = 1,
        label ='Estimated Distribution')
plt.axvline(x=hat_a, color='black', linestyle='--')
```

```
plt.title('$\hat \phi_{11}$', fontsize=12)
plt.yticks([])


plt.tight_layout()
plt.legend(loc='upper center', bbox_to_anchor=(0, 2.51),
           fontsize=11)
plt.show()
```

Generate data and plot for Figure 2.3.

```
memory00 = np.array([0, 0])

for i in range(4, 1001, 2):
  temp_data = generate_var(i)
  memory00 = np.vstack((memory00,
           [i, ols_var(temp_data)[0][0,0]]))


plt.figure(figsize=(9, 5))

plt.plot(memory00[:,0], memory00[:,1],
        color='gray', linewidth = 0.8)

var = (coef_cov[0,0]*499)
x = np.linspace(4, 1000, 400)

plt.plot(x, np.full((400,), -0.3), color='crimson',
        alpha=0.5, linewidth=0.8)

y = 3*((var/(x-1))**.5) #this is basically 3 times the sigma
```

```
plt.plot(x, y - 0.3, color='crimson', alpha=0.6, linewidth=1,
        label ='3 Standard Deviations from $\phi_{00}$ = -0.3')
plt.plot(x, -y - 0.3, color='crimson', alpha=0.6, linewidth=1)

plt.xlabel('$n$', fontsize=11)
plt.yticks([])
plt.legend(fontsize=11)
```

Generate data and plot for figure 2.4.

```
memory01 = np.array([0, 0])
memory10 = np.array([0, 0])
memory11 = np.array([0, 0])

for i in range(10, 1001, 2):
  temp_data = generate_var(i)
  memory01 = np.vstack((memory01,
                        [i, ols_var(temp_data)[0][0,1]]))
  memory10 = np.vstack((memory10,
                        [i, ols_var(temp_data)[0][1,0]]))
  memory11 = np.vstack((memory11,
                        [i, ols_var(temp_data)[0][1,1]]))

plt.figure(figsize=(15, 2.5))

x = np.linspace(10, 1000, 400)

#subplot for a_01
plt.subplot(1, 3, 1)
```

```
plt.plot(memory01[:,0], memory01[:,1], color='gray',
        linewidth = 0.6)

var = (coef_cov[1,1]*99)

plt.plot(x, np.full((400,), 0.7), color='crimson', alpha=0.5,
        linewidth=0.7)

y = 3*((var/(x-1))**.5) #this is basically 3 times the sigma
plt.plot(x, y + 0.7, color='crimson', alpha=0.6, linewidth=.9,
        label='3 Standard Deviations from $\phi_{00}$ = -0.3')
plt.plot(x, -y + 0.7, color='crimson', alpha=0.6, linewidth=.9)

plt.yticks([])
plt.xticks([])
plt.xlabel('$\phi_{01}$', fontsize=12)


#subplot for a_10
plt.subplot(1, 3, 2)

plt.plot(memory10[:,0], memory10[:,1], color='gray',
        linewidth = 0.7)

var = (coef_cov[2,2]*99)

plt.plot(x, np.full((400,), 0.4), color='crimson', alpha=0.5,
        linewidth=0.8)
```

```
y = 3*((var/(x-1))**.5) #this is basically 3 times the sigma
plt.plot(x, y + 0.4, color='crimson', alpha=0.6, linewidth=.9,
        label='3 Standard Deviations from $\phi_{00}$ = -0.3')
plt.plot(x, -y + 0.4, color='crimson', alpha=0.6, linewidth=.9)

plt.yticks([])
plt.xticks([])
plt.xlabel('$\phi_{10}$', fontsize=12)



#subplot for a_11
plt.subplot(1, 3, 3)

plt.plot(memory11[:,0], memory11[:,1], color='gray',
        linewidth = 0.7)

var = (coef_cov[3,3]*99)

plt.plot(x, np.full((400,), 0.5), color='crimson',
        alpha=0.5, linewidth=0.8)

y = 3*((var/(x-1))**.5) #this is basically 3 times the sigma
plt.plot(x, y + 0.5, color='crimson', alpha=0.6, linewidth=.9,
        label='3 Standard Deviations from $\phi_{00}$ = -0.3')
plt.plot(x, -y + 0.5, color='crimson', alpha=0.6, linewidth=.9)

plt.yticks([])
plt.xticks([])
plt.xlabel('$\phi_{11}$', fontsize=12)
```

```
#Adjusting final plot
plt.subplots_adjust(wspace=0.025)
plt.show()
```

A.4 Code for Section 3.3: Simulation of LS-VAR(1)

Import necessary packages.

```
import numpy as np
from numpy import random
from numpy.random import multivariate_normal
import matplotlib.pyplot as plt
from scipy.stats import norm
import math
```

Pre-compute in order to avoid repeated computations in order to make code faster.

```
root_1_2 = math.sqrt(1.2)
root_1_4 = math.sqrt(1.4)

root_2 = math.sqrt(2)
root_3 = math.sqrt(3)
root_4 = math.sqrt(4)
root_5 = math.sqrt(5)
root_6 = math.sqrt(6)

log_2 = math.log(2)
log_3 = math.log(3)
```

```
log_4 = math.log(4)
log_5 = math.log(5)
log_6 = math.log(6)

pi = math.pi

def sin(x):
  return math.sin(x)
def cos(x):
  return math.cos(x)
```

Define the coefficient function for the LS-VAR(1) process

```
def Phi(u):
  a_0, a_1 = .4, .3

  #j = 0
  phi_00 = a_0*(1 / log_5)*cos(6*pi*u / root_5)
  phi_10 = a_0*(root_1_2 / log_5)*cos(6*pi*u / root_5 - 1)
  phi_20 = a_0*(root_1_4 / log_5)*cos(6*pi*u / root_5 - 2)

  #j = 1
  phi_01 = a_0*(1 / log_6)*cos(6*pi*u / root_6)
  phi_11 = a_0*(root_1_2 / log_6)*cos(6*pi*u / root_6 - 1)
  phi_21 = a_0*(root_1_4 / log_6)*cos(6*pi*u / root_6 - 2)

  #j = 2
  phi_02 = a_1*root_2*sin(3*pi * u / log_2)
  phi_12 = a_1*root_3*sin(3*pi * u / log_3 - 1)
  phi_22 = a_1*root_4*sin(3*pi * u / log_4 - 2)
```

```
  return np.array([[phi_00, phi_01, phi_02],
                   [phi_10, phi_11, phi_12],
                   [phi_20, phi_21, phi_22]])
```

Checking if the process defined above is locally stationary and causal.

```
discrete_time = np.linspace(0, 1, 800)

max_mod_eig = 0

for u in discrete_time:
  eig0, eig1, eig2 = np.linalg.eigvals(Phi(u))
  eig0, eig1, eig2 = np.abs(eig0), np.abs(eig1), np.abs(eig2)

  max_mod_eig = max(max_mod_eig, eig0, eig1, eig2)

print(max_mod_eig)
```

Define function in order to simulate the LS-VAR(1) process. Also define a function to estimate the coefficient function of LS-VAR(1) processes with mean 0.

```
err_matrix = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]

def generate_mean0_ls_var(n, seed=0):
  if seed != 0:
    np.random.seed(seed)

  data = multivariate_normal([0, 0, 0], err_matrix, size=n)

  for i in range(1, n):
```

```
    data[i] = data[i] + data[i-1] @ Phi(i/n).T

  return data


def gauss_kern(x, h = 0.03):
  denominator = math.sqrt(2*pi)*h
  return math.exp(-0.5 * (x/h) ** 2)/denominator

def epanech_kern(x, h):
  return 0.75 * (1 - ((x/h) ** 2))

def estimate_coef(u, data, h = 0.03):
  n = len(data)

  cov_0 = np.zeros((3, 3))
  cov_1 = np.zeros((3, 3))

  t_lower = math.floor((u-h)*n)
  t_upper = math.floor((u+h)*n)

  for i in range(max(1, t_lower), min(n, t_upper+1)):
    k = epanech_kern(i/n - u, h)

    cov_0 += k*(data[i].reshape(3, 1) @ data[i-1].reshape(1, 3))
    cov_1 += k*(data[i-1].reshape(3, 1) @ data[i-1].reshape(1, 3))

  return cov_0 @ np.linalg.inv(cov_1)
```

Simulate LS-VAR(1) processes, estimate the coefficients and store them in order to plot it

later.

```
memory_est_coef = np.array([np.empty((100, 3, 3),
    dtype=np.float64) for t in range(400)])

for i in range(100):
  curr_data = generate_mean0_ls_var(400)

  for t in range(400):
    curr_u = t/400
    memory_est_coef[t, i] = estimate_coef(curr_u, curr_data)
```

Produce figure 3.1.

```
fig = plt.figure(figsize=(20, 13))

x = np.array([t/400 for t in range(400)])
x = x[10:391]
y = np.array([Phi(u) for u in x]) #shape = (400, 3, 3)

for j in range(3):
  for i in range(3):
    plt.subplot(3, 3, 3*i + j+1)


    mean, lower_bound, upper_bound = [], [], []
    # for t in range(400):
    for t in range(10, 391):
      curr = t/400
```

```
      mean.append(np.mean(memory_est_coef[t, :, i, j]))

      std = math.sqrt(np.var(memory_est_coef[t, :, i, j]))
      lower_bound.append(mean[-1] - 2*std)
      upper_bound.append(mean[-1] + 2*std)

    plt.plot(x, y[:, i, j], color='red', linewidth = 0.9,
            alpha = .9)
    plt.plot(x, mean, color= 'black', linewidth = 0.9,
            alpha = 0.8)
    plt.plot(x, lower_bound, color= 'black', linestyle='--',
            linewidth = 0.6)
    plt.plot(x, upper_bound, color= 'black', linestyle='--',
            linewidth = 0.6)

    plt.yticks(fontsize=9, rotation=90)
    plt.xticks(fontsize=9)
    # plt.xticks([0, 0.2, 0.4, 0.6, 0.8, 1])

    plt.title("$\phi_{" + str(i) + str(j) + "}$", fontsize=16)

plt.tight_layout()
```

Define the mean function for LS-VAR(1) with time varying mean.

```
def Mu(u):
  mu_0 = sin(u*pi*log_4)/2
  mu_1 = sin(u*pi*log_3)/3
  # mu_2 = sin(u*pi*log_2)/4
```

```
  mu_2 = sin(2*u*pi*log_2)/4

  return np.array([[mu_0],[mu_1],[mu_2]])

def B(u):
  phi = Phi(u)
  mu = Mu(u)
  return np.hstack((mu - (phi @ mu), phi))
```

Define function to simulate LS-VAR(1) processes and to estimate the coefficients with local linear approximations.

```
err_matrix = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]

def generate_ls_var(n):

  data = multivariate_normal([0, 0, 0], err_matrix, size=n)

  for i in range(1, n):
    data[i] = data[i] + (np.hstack(([1], data[i-1])) @ B(i/n).T)

  return data


h = 0.06

denominator = math.sqrt(2*pi)*h
def gauss_kern(x):
  return math.exp(-0.5 * (x/h) ** 2)/denominator
```

```
def epanech_kern(x):
  return 0.75 * (1 - ((x/h) ** 2))

def local_linear(u, data):
  n = len(data)

  t_lower = math.floor((u-h)*n)
  t_upper = math.floor((u+h)*n)

  Delta = np.zeros((n, n))
  K = np.zeros((n, n))
  for i in range(max(1, t_lower), min(n, t_upper+1)):
    Delta[i, i] = i/n - u
    K[i,i] = epanech_kern(Delta[i, i])

  n_1s = np.ones((n, 1))
  X_0 = np.vstack(([0, 0, 0], data[:-1,:]))
  X_1 = data
  Z_0 = np.hstack((n_1s, X_0))


  W = K - (K @ Delta @ Z_0 @
      np.linalg.inv(Z_0.T @ Delta @ K @ Delta @ Z_0) @
      Z_0.T @ Delta @ K)

  oneT_W_one =  n_1s.T @ W @ n_1s
  W_Xo = W @ X_0

  G_0 = (X_0.T @ W_Xo - (X_0.T @ W @ n_1s @
```

```
        n_1s.T @ W_Xo / oneT_W_one))
  G_1 = (X_1.T @ W_Xo - (X_1.T @ W @ n_1s @
        n_1s.T @ W_Xo / oneT_W_one))


  mu = X_1.T @ W @ n_1s / oneT_W_one
  phi = G_1 @ np.linalg.inv(G_0)

  return mu, phi
```

Generate LS-VAR(1) processes, estimate coefficients, and store the coefficients.

```
memory_est_phi = np.array([np.empty((100, 3, 3),
        dtype=np.float64) for t in range(400)])
memory_est_mu = np.array([np.empty((100, 3, 1),
        dtype=np.float64) for t in range(400)])

for i in range(100):
  curr_data = generate_ls_var(400)


  for t in range(400):
    curr_u = t/400
    memory_est_mu[t, i], memory_est_phi[t, i] = (
            local_linear(curr_u, curr_data))

  if i % 10 == 0:
    print(i)
```

Produce figure 3.2.

```
fig = plt.figure(figsize=(20, 17)) #w, h #20, 13

x = np.array([t/400 for t in range(400)])
x = x[10:391]
y = np.array([Mu(u) for u in x]) #shape = (400, 3, 1)


for i in range(3):
  plt.subplot(4, 3, i+1)


  mean, lower_bound, upper_bound = [], [], []
  for t in range(10, 391):
    curr = t/400


    mean.append(np.mean(memory_est_mu[t, :, i, 0]))
    std = math.sqrt(np.var(memory_est_mu[t, :, i, 0]))

    lower_bound.append(mean[-1] - 2*std)
    upper_bound.append(mean[-1] + 2*std)

  plt.plot(x, y[:, i, 0], color='red', linewidth = 0.9,
        alpha = .9)
  plt.plot(x, mean, color= 'black', linewidth = 0.9,
        alpha = 0.8)
  plt.plot(x, lower_bound, color= 'black', linestyle='--',
        linewidth = 0.6)
  plt.plot(x, upper_bound, color= 'black', linestyle='--',
```

```
        linewidth = 0.6)
  plt.yticks(fontsize=9, rotation=90)
  plt.xticks(fontsize=9)


  plt.title("$\mu_{" + str(i) + "}$", fontsize=16)


y = np.array([Phi(u) for u in x]) #shape = (400, 3, 3)

for j in range(3):
  for i in range(3):
    plt.subplot(4, 3, 3*i + j + 4)


    mean, lower_bound, upper_bound = [], [], []
    for t in range(10, 391):
      curr = t/400


      mean.append(np.mean(memory_est_phi[t, :, i, j]))

      std = math.sqrt(np.var(memory_est_phi[t, :, i, j]))
      lower_bound.append(mean[-1] - 2*std)
      upper_bound.append(mean[-1] + 2*std)

    plt.plot(x, y[:, i, j], color='red', linewidth = 0.9,
            alpha = .9)
    plt.plot(x, mean, color= 'black', linewidth = 0.9,
```

```
            alpha = 0.8)
    plt.plot(x, lower_bound, color= 'black', linestyle='--',
            linewidth = 0.6)
    plt.plot(x, upper_bound, color= 'black', linestyle='--',
            linewidth = 0.6)

    plt.yticks(fontsize=9, rotation=90)
    plt.xticks(fontsize=9)


    plt.title("$\phi_{" + str(i) + str(j) + "}$", fontsize=16)

plt.tight_layout()

plt.savefig('fig_3.2.pdf', format='pdf')
```

A.5 Code for Section 3.4: Modelling Financial Time Series with LS-VAR(1)

Import necessary packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
from statsmodels.tsa.api import VAR
import math
from datetime import timedelta
```

```
from datetime import datetime
from scipy.interpolate import interp1d
```

Load data.

```
path = '/content/drive/MyDrive/Master_Thesis/'
interest = pd.read_excel(path+'longer_ffer.xls',
    skiprows=10, index_col = 'observation_date')

tickerData = yf.Ticker('XLF')
bank = tickerData.history(period='1d', start='2005-5-1',
                          end='2023-4-15')
bank = bank.Close

# voo.index = voo.index.to_series().dt.date
interest.index = interest.index.to_series().dt.date
bank.index = bank.index.to_series().dt.date

Df = pd.merge(bank, interest, left_index=True, right_index=True)
# Df = pd.merge(Df, bank, left_index=True, right_index=True)
Df.columns = ['bank', 'ffer']
Df.index = pd.to_datetime(Df.index)
Df = Df.resample('MS').first()

# Df['voo_log'] = np.log(Df.voo/Df.voo.shift(1))
Df['ffer_log'] = np.log(Df.ffer/Df.ffer.shift(1))
Df['bank_log'] = np.log(Df.bank/Df.bank.shift(1))
```

Reformat and label data properly.

```
df = pd.merge(bank, interest, left_index=True, right_index=True)
```

```
# df = pd.merge(df, bank, left_index=True, right_index=True)
df.columns = ['bank', 'ffer']
df.index = pd.to_datetime(df.index)

df['ffer_log'] = np.log(df.ffer/df.ffer.shift(1))
df['bank_log'] = np.log(df.bank/df.bank.shift(1))

df = df.dropna()

crisis = df.loc['2007-12-01':'2009-12-01']
covid = df.loc['2020-02-01':'2021-12-31']

post_covid = df.loc['2022-01-01':]

regular = df.loc[
    ((df.index > '2005-05-01') & (df.index < '2007-12-01')) |
    ((df.index >= '2010-01-01') & (df.index <= '2020-01-01'))]
```

Fit VAR(1) with structural breaks.

```
reg_mod = VAR(regular[['ffer_log', 'bank_log']])
reg_mod.fit(1).params

cris_mod = VAR(crisis[['ffer_log', 'bank_log']])
cris_mod.fit(1).params

covid_mod = VAR(covid[['ffer_log', 'bank_log']])
covid_mod.fit(1).params

post_mod = VAR(post_covid[['ffer_log', 'bank_log']])
```

```
post_mod.fit(1).params
```

Define function to estimate LS-VAR coefficients with time varying mean.

```
h = 0.1

def epanech_kern(x):
  return 0.75 * (1 - ((x/h) ** 2))

def local_linear(u, data):
  n = len(data)

  t_lower = math.floor((u-h)*n)
  t_upper = math.floor((u+h)*n)

  Delta = np.zeros((n, n))
  K = np.zeros((n, n))
  for i in range(max(1, t_lower), min(n, t_upper+1)):
    Delta[i, i] = i/n - u
    K[i,i] = epanech_kern(Delta[i, i])

  n_1s = np.ones((n, 1))
  X_0 = np.vstack(([0, 0], data[:-1,:]))
  X_1 = data
  Z_0 = np.hstack((n_1s, X_0))


  W = K - (K @ Delta @ Z_0 @
      np.linalg.inv(Z_0.T @ Delta @ K @ Delta @ Z_0) @
      Z_0.T @ Delta @ K)
```

```
  oneT_W_one =  n_1s.T @ W @ n_1s
  W_Xo = W @ X_0

  G_0 = X_0.T @ W_Xo - (X_0.T @ W @ n_1s @ n_1s.T @ W_Xo / oneT_W_one)
  G_1 = X_1.T @ W_Xo - (X_1.T @ W @ n_1s @ n_1s.T @ W_Xo / oneT_W_one)


  mu = X_1.T @ W @ n_1s / oneT_W_one
  phi = G_1 @ np.linalg.inv(G_0)

  return mu, phi
```

Produce figure 3.3.

```
plt.figure(figsize=(20, 12))

dates = [(datetime(2005, 5, 1) +
    timedelta(days=i) for i in range(6570))]

#phi_00
plt.subplot(2, 2, 1)

#regular
plt.plot(dates[:944], [-0.373560]*len(dates[:944]), color='black')
plt.plot(dates[1675:5358], [-0.373560]*len(dates[1675:5358]),
    color='black')


#Financial Crisis
```

```
plt.plot(dates[944:1675], [0.063859]*len(dates[944:1675]),
    color='black')

#Covid
plt.plot(dates[5358:6088], [-0.064634]*len(dates[5358:6088]),
    color='black')

#Post covid
plt.plot(dates[6088:], [-0.007635]*len(dates[6088:]),
    color='black')


#horizontal lines
plt.vlines(datetime(2007, 12, 1), ymin=-0.373560,
    ymax=0.063859, color='black')
plt.vlines(datetime(2009, 12, 1), ymin=-0.373560,
    ymax=0.063859, color='black')
plt.vlines(datetime(2020, 1, 1), ymin=-0.373560,
    ymax=-0.064634, color='black')
plt.vlines(datetime(2021, 12, 31), ymin=-0.064634,
    ymax=-0.007635, color='black')


plt.yticks(fontsize=14, rotation=90)
plt.xticks(fontsize=16)
plt.title("$\phi_{00}$", fontsize=20)
```

```
#phi_01
plt.subplot(2, 2, 2)

#regular
plt.plot(dates[:944], [0.047256]*len(dates[:944]), color='black')
plt.plot(dates[1675:5358],
    [0.047256]*len(dates[1675:5358]), color='black')


#Financial Crisis
plt.plot(dates[944:1675], [-0.263530]*len(dates[944:1675]),
    color='black')

#Covid
plt.plot(dates[5358:6088], [-0.573015]*len(dates[5358:6088]),
    color='black')

#Post covid
plt.plot(dates[6088:], [0.980110]*len(dates[6088:]),
    color='black')


#horizontal lines
plt.vlines(datetime(2007, 12, 1), ymin=-0.263530,
    ymax=0.047256, color='black')
plt.vlines(datetime(2009, 12, 1), ymin=-0.263530,
    ymax=0.047256, color='black')
plt.vlines(datetime(2020, 1, 1), ymin=-0.573015,
    ymax=0.047256, color='black')
```

```
plt.vlines(datetime(2021, 12, 31), ymin=-0.573015,
    ymax=0.980110, color='black')


plt.yticks(fontsize=14, rotation=90)
plt.xticks(fontsize=16)
plt.title("$\phi_{01}$", fontsize=20)




#phi_10
plt.subplot(2, 2, 3)

#regular
plt.plot(dates[:944], [-0.000676]*len(dates[:944]), color='black')
plt.plot(dates[1675:5358], [-0.000676]*len(dates[1675:5358]),
    color='black')


#Financial Crisis
plt.plot(dates[944:1675], [-0.003260]*len(dates[944:1675]),
    color='black')

#Covid
plt.plot(dates[5358:6088], [0.000043]*len(dates[5358:6088]),
    color='black')

#Post covid
```

```
plt.plot(dates[6088:], [0.002529]*len(dates[6088:]), color='black')


#horizontal lines
plt.vlines(datetime(2007, 12, 1), ymin=-0.003260,
    ymax=-0.000676, color='black')
plt.vlines(datetime(2009, 12, 1), ymin=-0.003260,
    ymax=-0.000676, color='black')
plt.vlines(datetime(2020, 1, 1), ymin=-0.000676,
    ymax=0.000043, color='black')
plt.vlines(datetime(2021, 12, 31), ymin=0.000043,
    ymax=0.002529, color='black')


plt.yticks(fontsize=14, rotation=90)
plt.xticks(fontsize=16)
plt.title("$\phi_{10}$", fontsize=20)


#phi_11
plt.subplot(2, 2, 4)

#regular
plt.plot(dates[:944], [-0.085651]*len(dates[:944]), color='black')
plt.plot(dates[1675:5358], [-0.085651]*len(dates[1675:5358]),
    color='black')


#Financial Crisis
```

```
plt.plot(dates[944:1675], [-0.120927]*len(dates[944:1675]),
    color='black')

#Covid
plt.plot(dates[5358:6088], [-0.227552]*len(dates[5358:6088]),
    color='black')

#Post covid
plt.plot(dates[6088:], [0.033004]*len(dates[6088:]),
    color='black')


#horizontal lines
plt.vlines(datetime(2007, 12, 1), ymin=-0.120927,
    ymax=-0.085651, color='black')
plt.vlines(datetime(2009, 12, 1), ymin=-0.120927,
    ymax=-0.085651, color='black')
plt.vlines(datetime(2020, 1, 1), ymin=-0.227552,
    ymax=-0.085651, color='black')
plt.vlines(datetime(2021, 12, 31), ymin=-0.227552,
    ymax=0.033004, color='black')


plt.yticks(fontsize=14, rotation=90)
plt.xticks(fontsize=16)
plt.title("$\phi_{11}$", fontsize=20)
```

```
plt.tight_layout()



memory_est_phi = np.array([np.empty((2, 2), dtype=np.float64)
        for t in range(100)])
memory_est_mu = np.array([np.empty((2, 1), dtype=np.float64)
        for t in range(100)])


for t in range(100):
  curr_u = t/100
  memory_est_mu[t], memory_est_phi[t] = local_linear(curr_u,
                                        daily_data)

fig = plt.figure(figsize=(20, 12))

x = np.array([t/100 for t in range(100)])

start_date = datetime(2005, 5, 3)
end_date = datetime(2023, 4, 14)
date_values = ([start_date + (end_date - start_date) * val
                for val in x])

for j in range(2):
  for i in range(2):

    plt.subplot(2, 2, 2*i + j + 1)
```

```
    func = interp1d([date.timestamp() for date in date_values],
                memory_est_phi[:, i, j], kind='quadratic')

    x = np.linspace(min([date.timestamp()
                         for date in date_values]),
                    max([date.timestamp()
                         for date in date_values]),
                    1000)

    plt.plot([datetime.fromtimestamp(date) for date in x],
             func(x), color='black')

    plt.yticks(fontsize=14, rotation=90)
    plt.xticks(fontsize=16)

    plt.title("$\phi_{" + str(i) + str(j) + "}$", fontsize=20)

plt.tight_layout()

plt.savefig('fig_3.3.b.pdf', format='pdf')
```