UNIVERSITAT DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Master's Thesis in Statistics and Operations Research

Financial Stress Indices

On the Connectedness of Systemic Risk between economic powers and forecasting with Recurrent Neural Networks

Oriol Gairín Armengol

Supervised by Dr. Helena Chuliá Soler
and Dr. Salvador Torra Porras

January 2023

Statement of Originality

I, Oriol Gairín Armengol, with a National Identity Document (DNI) 47237354T, and a student of the Master's degree in Statistics and Operations Research at the UPC-UB, in relation to the report of the final project presented for its defense and evaluation during the January call of the 2022-2023 academic year, declare that:

- The document presented is original and has been made by myself.
- The work has been carried out mainly with the aim of evaluating the Master's thesis course in Statistics and Operations Research at the UPC-UB, and has not been previously presented to be qualified in the evaluation of any other subject neither this nor at no other university.
- In the case of content of works published by third parties, the authorship is clearly attributed, citing the sources properly.
- In cases where my work has been carried out in collaboration with other researchers and/or students, it is stated exactly which contributions are derived from the work of third parties and which are derived from my contribution.
- With the exception of the points mentioned above, the work presented is my own.

Table of Contents

Abstract

Financial stress indices are a type of indicators that seek to provide an overall vision about the risk situation of the financial system, by comprising into a single statistic a set of individual indicators that measure, in some way, the existing risk within a certain segment of the financial system. In this project we explore the interdependencies of systemic risk between the different European powers and between the three main global financial powers: China, Europe and United States. Furthermore, we implement a Long Short-Term Memory (LSTM) network to predict such financial stress indices at three different time horizons, and compare our findings with classical econometric approaches such as ARMA-GARCH models.

Chapter 1

Introduction

The role of the financial system at a macro scale is critical, it has huge impact on the real economy and its actions can have direct consequences on our daily lives. The constant development of an increasingly globalized and interconnected world represents progress in many aspects, but it also brings with it many complexities, dysfunctions and risks that can trigger very severe systematic crises, which can easily spread between countries and cause periods of economic setback with strong consequences on our quality of life. Identifying and monitoring factors that can explain the risks in which the financial system is immersed is a key aspect to be considered by regulators and central banks to prevent potential crisis and avoid collapsing in the economy.

Currently, there are many individual indicators that undertake this function, such as volatility series of stock returns, interest rate spreads or ratios between financial quantities, which focus their attention on the risk situation in which a certain market finds itself within the financial system. However, sometimes one can't see the forest for the trees; the poor performance of one of these indicators can lead to erroneous generalizations about the overall state of the systemic risk, when it may well be that a particular event is simply the result of the idiosyncrasy present in that segment and does not correspond to a deeper crisis.

To this extent, one may want to define what is the **systemic risk**. As stated in 2009 by the International Monetary Fund (IMF), the Bank for International Settlements (BIS) and the Secretariat of the Financial Stability Board (FSB), the systemic risk may be understood as:

> *"[...] risk of disruption to financial services that is (i) caused by an impairment of all or parts of the financial system and (ii) has the potential to have serious negative consequences for the real economy."* [1]

To assess this phenomenon, financial stress indices arise as a type of indicators that intend to give an overall vision about the situation of the financial system. They are composite time series which are formed by the aggregation of individual risk indicators from the major financial segments. They aim to cover as much relevant information as it can be obtained from the markets and compress it into one unique indicator that summarizes the current systemic risk level. However, these indicators are not exempt from their own complexities,

opening up an interesting field of study within the branch of financial risk, which ranges from their own construction methodology and statistical design to subsequent evaluations of their capacity to identify financial stress events, their integration and interconnection with other macroeconomic series and other practical uses with potential regulatory impact.

This project aims to contribute to the existing literature on financial stress indices, focusing on the indicator designed in [2], named Composite Indicator of Systemic Stress (CISS). This indicator is recognized by the largest European financial institutions and is currently being used by the European Central Bank, which reports it daily for the greatest economic powers on the European continent, United States and China.

Our work is divided into two different research areas. In the first place, we carry out an analysis of the levels of connectivity that these indicators present between different regions. Following the methodological framework proposed in [3, 4], we exploit the so-called Connectedness measures and apply them over our time series of study. The goal of this part is to determine what kind of relationships exist between the systemic risk of the different European powers and between the three main global financial powers: China, Europe and United States. Our approach allows us to address the following questions: what is the total risk connectedness level of our financial network?, are there regions that have higher levels of interconnection than others?, what is the directionality of the risk in each case?, are there regions that are net producers of financial stress and others that are net receivers? In addition, we must consider whether these linkages are static, or if, on the contrary, the levels of connectedness change dynamically over time. At what times does risk transmission become more acute? How do specific events in one region affect the levels of systemic risk in others?

Second, we build a neural network to predict CISS series at different time horizons. Specifically, we use a network with Long Short-Term Memory (LSTM) units, which belongs to the Recurrent Neural Networks (RNN) family and contains memory modules designed to model sequential data. Through a Bayesian optimization process on the hyperparametric space, we determine which configurations are best suited to our experimental problem. In addition, we fit classic autoregressive models of the ARMA-GARCH type on the same sample. The objective is to determine whether the CISS series have some type of temporal structure, linear or nonlinear, that allows us to predict future values of systemic risk in the different regions considered. In this sense, this part consists of two steps, firstly, data engineering and design of the LSTM network and, secondly, comparison with respect to the classical econometric approaches commonly used in the financial industry. Our approach allows us to address the following questions: what limitations do neural network models have in the field of financial series?, in what situations do we find them to be advantageous?, what is the relative gain of such models with respect to the ARMA-GARCH approach?, what are the most predictable regions?, and how is the error term distributed as we increase the time horizon?

Hence, the interest of this work can be seen from three different perspectives. From the financial point of view, since we obtain results on the connectedness levels of the financial risk of different regions, and we also predict future values of the series. From the scientific point of view, since we contribute to the existing literature on financial stress indicators, applying methodologies that are proven to be useful in other fields of research but have not yet been extensively investigated in this one. And last, from the educational point of view since this

project gives us the possibility of enriching our knowledge around time series and the state-of-the-art of deep learning techniques for prediction.

Next, we briefly detail the structure of the work. Section 2 introduces the methodology and the theoretical framework used on this project. Subsection 2.1 pivots around the Connectedness measures and the approach used to enable for time-varying dependence. Subsection 2.2 explains how the LSTM network works. We start by introducing neural networks for giving some context and we proceed to describe the architecture of an LSTM layer. Subsequently, we comment on the weight optimization procedure and on the main characteristics of the learning stage. Section 3 focuses on the CISS data extracted from the European Central Bank (ECB). The statistical design of the indicator is explained in Subsection 3.1 and, later, we present a descriptive analysis of the series in Subsection 3.2. We closure the section by identifying historical stress events and discussing whether these are reflected in the CISS indexes. In Section 4, we present the results obtained. Subsection 4.1 shows the results regarding the systemic risk connectedness. In order, we analyse the connectedness within the European Area and later between China, United States and Europe as a whole. Subsection 4.2 shows the forecasts results. First, we present the Benchmark models for comparison. We proceed by describing how we engineered our LSTM network. Last, we present our results on the test sample, for the different time horizons considered. Section 5, Conclusions, summarizes the work done and ends up with a discussion on further work and future lines of research.

Chapter 2

Methodology

2.1 Connectedness

Consider a set of time series to which we want to assess its interdependencies and features of its linkages, from pairwise connections to system-wide levels of interconnection. Correlation-based measures can give an idea of the relationship between two variables, but they are limited to capture only linear and pairwise dependencies, plus they fail to give information about the directionality of the linkage. Other measures can arise from approximating models to the system of time series, which account for these possible interconnections. For instance, assuming that the system can be well modelled by an N-variable *VAR(p)* process, then the model contains all the features regarding interdependences between the time series. However, since many time series may compose the system and this can be dependent up to its p lagged past values, the number of parameters rapidly increases and the direct interpretation of the model by simply examining its coefficients becomes complex and generally a very useless practice.

To overcome this situation, in [3,4] the authors develop a generalized connectedness framework, related to the idea of computing the shares of variances that can be associated to the forecasting error of one variable due to shocks on the other variables. In that sense, the concept of connectedness [5] can be defined and measured consistently, providing with useful insights of directional linkages between individual variables, relations on a more global scale, and even enabling for time varying levels of connectedness.

In what follows, we describe the methodology presented above, that will be applied on Section 4 (Results) on the systemic risk indicators, to evaluate the possible spillovers of risk between the financial systems of the different regions studied.

2.1.1 Generalized variance decomposition

Consider we model our N-variable time series system x_t with a *VAR(p)* process:

$$x_t = \sum_{i=1}^{p} \Phi_i x_{t-i} + \varepsilon_t, \quad \text{with} \quad \varepsilon_t \sim (0, \Sigma) \tag{2.1}$$

Here ε_t is a N-dimensional independently and identically distributed vector of disturbances with 0 expected value and covariance matrix Σ, and Φ_i is the $N \times N$ coefficient matrix associated to the i-th lagged value of x_t. One can express the *VAR(p)* process in its infinite moving average representation:

$$x_t = \sum_{i=0}^{\infty} A_i \varepsilon_{t-i}, \tag{2.2}$$

where A_i represents the $N \times N$ coefficient matrix associated to the i-th lagged disturbances of x_t: ε_{t-i}. The A_i matrices can be directly obtained from the $VAR(p)$ model, as they follow the recursion:

$$A_i = \sum_{j=1}^{p} \Phi_j A_{i-j}, \quad \text{with} \quad A_0 = \mathbb{I}_{N \times N} \quad \text{and} \quad A_i = 0 \ \text{ for } \ i < 0 \tag{2.3}$$

From the vector moving average representation, one can compute the H-step-ahead variance decomposition matrix $D^H = [d_{ij}^H]$ with $i, j = 1, 2, \ldots, N$. This procedure allows us to decompose the variance of the forecasting error (at horizon H) into parts depending upon coming from the different variables of the system. That is, the ij-th element of the matrix (d_{ij}^H) represents the fraction of variance in variable i of the H-step-ahead forecasting error that comes due to shocks in variable j. However, since the estimated *VAR(p)* does not generally holds orthogonality, meaning that the disturbances tend to be contemporaneously correlated, we can't simply apply orthogonal variance decomposition, but rather we make use of the generalized variance decomposition approach [6, 7]. Therefore, following the literature, we compute the ij-th element of the generalized variance decomposition matrix of the H-step-ahead forecast error as:

$$d_{ij}^{gH} = \frac{\sigma_{jj}^{-1} \sum_{h=0}^{H-1} \left(e_i' A_h \Sigma e_j\right)^2}{\sum_{h=0}^{H-1} \left(e_i' A_h \Sigma A_h' e_i\right)} \tag{2.4}$$

where e_i is a unit vector of dimension N, with value 1 on position i and 0's elsewhere, Σ is the covariance matrix of the error vector ε and σ_{jj} stands for the standard deviation associated to the error term of the j-th variable.

As the *VAR(p)* shocks may not be orthogonal, the set of the H-step-ahead forecasting error variances affecting variable i do not necessarily add one. Therefore, to ease interpretations, we normalize the d_{ij}^H elements rowwise:

$$\tilde{d_{ij}^{gH}} = \frac{d_{ij}^{gH}}{\sum_{j=1}^{N} d_{ij}^{gH}} \tag{2.5}$$

It immediately follows the next properties:

$$\sum_{j=1}^{N} \tilde{d}_{ij}^{gH} = 1 \quad ; \quad \sum_{i,j=1}^{N} \tilde{d}_{ij}^{gH} = N \tag{2.6}$$

2.1.2 The Connectedness Table

From the generalized variance decomposition framework described above, we can define the connectedness measures, summarized in the Connectedness Table (equation 2.7). Note that we have dropped, for simplicity, the generalized (g) and normalized ($\tilde{}$) marks from the notation but will be referring to them implicitly from now on.

	x_1	x_2	$\cdots$	x_N	From Others ($C^H_{i\leftarrow\bullet}$)
x_1	d^H_{11}	d^H_{12}	$\cdots$	d^H_{1N}	$\sum_{j=1}^{N} d^H_{1j}, j \neq 1$
x_2	d^H_{21}	d^H_{22}	$\cdots$	d^H_{2N}	$\sum_{j=1}^{N} d^H_{2j}, j \neq 2$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_N	d^H_{N1}	d^H_{N2}	$\cdots$	d^H_{NN}	$\sum_{j=1}^{N} d^H_{Nj}, j \neq N$
To Others	$\sum_{i=1}^{N} d^H_{i1}$	$\sum_{i=1}^{N} d^H_{i2}$	$\cdots$	$\sum_{i=1}^{N} d^H_{iN}$	$\frac{1}{N}\sum_{i,j=1}^{N} d^H_{ij}$
($C^H_{\bullet\leftarrow j}$)	$i \neq 1$	$i \neq 2$		$i \neq N$	$i \neq j$

(2.7)

The Connectedness Table contains the H-step-ahead variance decomposition matrix $D^H = [d^H_{ij}]$ in its upper-left block. D^H is asymmetric, since its variance decomposition components are generally directional $d^H_{ij} \neq d^H_{ji}$.

The *pairwise directional connectedness* from j to i can be defined as:

$$C^H_{i\leftarrow j} = d^H_{ij} \tag{2.8}$$

This is a measure of connectedness between two individual variables of the time series system. It measures which proportion of the variance in i is attributable to the source of error coming from j.

Similarly, we can define the *net pairwise directional connectedness* between two variables i and j as:

$$C^H_{ij} = C^H_{j\leftarrow i} - C^H_{i\leftarrow j} \tag{2.9}$$

Here, if $C^H_{ij} > 0$ implies that the variable i transmits greater shocks on the error variance to j than it receives from it, whereas in the $C^H_{ij} < 0$ case, variable j is the one having biggest influence among the two.

Now, consider the sum of variance decompositions elements that i receives, excluding itself. That is, the *total directional connectedness from others* to i:

$$C^H_{i\leftarrow\bullet} = \sum_{\substack{j=1 \\ j\neq i}}^{N} d^H_{ij} \tag{2.10}$$

Note that it satisfies $C^H_{i\leftarrow\bullet} = 1 - d^H_{ii}$. When this value is low, then the major part of the variance in the forecasting error comes from its own variance decomposition element, meaning that the variable is highly exogenous with respect to the others. On the other hand, higher values of $C^H_{i\leftarrow\bullet}$ occurs when i is easily influenced by the realized innovations from other variables.

Analogously, we define the *total directional connectedness to others* from j:

$$C^H_{\bullet\leftarrow j} = \sum_{\substack{i=1 \\ i\neq j}}^{N} d^H_{ij} \ , \tag{2.11}$$

which in this case indicates the total amount of variance shocks transmitted by j, and is not bounded to 1.

From here, the *net total directional connectedness* can be computed for each variable, to evaluate whether i is a net receiver ($C^H_i < 0$) or a net sender ($C^H_i > 0$) of shocks:

$$C^H_i = C^H_{\bullet\leftarrow i} - C^H_{i\leftarrow\bullet} \tag{2.12}$$

Last, one can define a macro-scale measure of connectedness that somehow summarizes the degree of interconnection the system has as a whole. By accounting for all the cross-variance shares, that is, the non-diagonal terms of the H-step-ahead variance decomposition matrix, we obtain the *total connectedness* measure:

$$C^H = \frac{\sum_{\substack{i,j=1 \\ i\neq j}}^{N} d^H_{ij}}{\sum_{i,j=1}^{N} d^H_{ij}} = \frac{1}{N} \sum_{\substack{i,j=1 \\ i\neq j}}^{N} d^H_{ij} \tag{2.13}$$

In some sense, the measures described here can be seen as an analogy of a network of exchange of goods between regions or villages, where individual regions are be both receivers and providers of those goods. Each region sells and buys a different number of products to each other individual region (*pairwise connectedness*), counting at the end the total amount of products they have sold or bought (*total directional connectedness*). The *total connectedness* quantifies in a single value, the gross level of exchanges achieved among the whole network, and it is unique, as for a closed system, the total amount of shipped goods must equal the total amount of received goods.

2.1.3 Time varying Connectedness and Predictive Horizon H

Since now, we defined the connectedness measures we are going to use. Obtaining those measures, by estimating a *VAR(p)* model to our set of time series and computing the variance decomposition matrix, will give us an idea of the unconditional or mean levels of interconnection that our system exhibit. However, one can argue that the interdependences between the time series may change during the period of study, and therefore, we should allow for time-varying connectedness measures to better capture the dynamic nature of the interrelated time series. To do so, we must enable the variance decomposition matrix to be time dependent, and since its

calculation is given by the estimated coefficients of the approximating model *VAR(p)*, we have no choice but to let the estimated coefficients of the underlying time-series model to change over time. We address this issue by using a rolling-window approach, consisting of estimating the VAR parameters by taking only the most recent w observations in time and calculating the connectedness measures for each sub-period, then rolling the window of width w one step forward and repeating the process until the end of our sample.

This method requires the choice of the width of the rolling window w, that is subject to influence our results. Generally, one will expect smoother variations of the connectedness measures as window width increases, whereas sharper variations for small values of w. In a similar way, the choice of the predictive horizon H may also have an effect on our outcome. As pointed in [5], the connectedness measures do not necessarily need to be robust to the choice of H, but rather different values of H will lead us to explore the different relationships between the time series more on the "long-" or "short-run". In any case, the appropriate choice of these parameters will depend on the actual experimental scenario and on being consistent with the objectives to be met.

2.2 LSTM Neural Network

Artificial neural networks (ANN) is a topic that has raised great interest in the world of statistics and data science in last decades [8]. This has been achieved thanks to recent advances in computational science and to the ability of computers of processing larger amounts of data and performing mathematical operations more and more efficiently.

Deep learning (DL) models are a type of ANN's with an internal structure with several processing layers, which results in a great capacity for abstraction and generalization over the initial input data [9]. Most of these DL models focus on predictive tasks (supervised learning). Its fundamental structure (Deep Multi-Layer Perceptron models or DMLP) consists of a network of connected neurons (or nodes), which are divided into three types of layers. The input layer is the layer where the data is ingested into the model. The number of explanatory variables (feature space) defines the number of neurons in the input layer. The hidden layers are the intermediate layers, which are stacked one on top of the other. A hidden layer receives the information or input vector (x) from the neurons of the previous layer and performs a (typically) linear transformation of the inputs according to the parameters (or weights) of each neuron, which can be expressed in vector form as $z = Wx$, where (W) represents the weight matrix. The outcome is transformed by a non-linear function called activation function $\sigma(z)$, which produces an output that is transmitted to the neurons of the next layer. The result of the subsequent transformations is sent to the output layer, which returns the final output of the model.

Information in DMLP models flows forward in the network, from the input layer to the output layer. To train the model (that is, adjust the internal parameters of each neuron), a loss function f is defined that depends on the output obtained in the output layer and on the real expected value for the given input. As the tuple "input-real expected value" are known in the training phase and the output depends on the parameter space Ω,

it can be seen that the loss function can be expressed essentially as $f(\Omega)$. The error (precisely, how the error on the loss function changes when we infinitesimally modify the set of parameters) propagates in the opposite direction to the flow of information (backpropagation of the error, *i.e.*, the error flows from the output layer to the input layer) and is calculated using classical partial derivative calculus theory.

The gradient of the loss function $\nabla f(\Omega)$ (with respect to the parameter space) indicates the direction of maximum change of the loss function with respect to the infinitesimal change of the components of Ω. The idea behind the training of the model consists of iteratively modifying the parameters of Ω by a term proportional to the gradient of the loss function $(-\alpha\nabla f(\Omega))$, that is, in the direction of the maximum error decrease. The positive constant (α) that multiplies the gradient is the so-called learning rate and determines how much the parameters are modified at each step. Under certain conditions, the process converges to a local minimum at Ω for which the error is (locally) minimized.

Summarizing, a deep learning model is fundamentally a network of computational units that concatenates activation functions over linear transformations of the inputs of the previous layer. Interestingly, structures with several hidden layers and multiple neurons in each layer generate a large parameter space, capable of extracting complex structures from the initial data and generalizing the salient aspects of the input to optimize the predictive capacity of the model.

Apart from its multiple applications in speech recognition, visual object recognition, object detection or in many other diverse fields such as biostatistics or medicine [10], deep learning has also found its place in financial applications. In particular, the prediction of financial time series is of great interest both in the academic world and in the finance industry due to its potential economic impact.

For prediction problems with sequential data, such as language translation, handwriting recognition or prediction of financial time series, there is a type of DL models called Recurrent Neural Networks (RNN) that takes into account the dependency structure of the data with respect to their previous lagged values [11]. However, classical gradient-based algorithms applied to plain RNN models are often inefficient, as error backpropagation over time tends to either explode or vanish [12]. This leads to the non-convergence of the parameter space towards a local minimum, or in other situations, to the limitation of these models to learn relationships between data that are far apart in the temporal sequence (long-term dependencies) [13]. To solve this problem, the Long Short-Term Memory (LSTM) model proposes a specific architecture for the network nodes that allows for constant error flow backpropagation over time (*constant error carrousel* CEC) by means of truncated gradient-based algorithms for training its parameters [13, 14].

In this project, we focus on the prediction of financial time series with LSTM neural networks, since they allow capturing the temporal structure of the data both in the long and short term and have proven to be the reference RNN model within the financial sector [9, 15]. In the following sections, we describe the architecture of an LSTM network and present the most relevant aspects regarding the optimization methods and the hyperparameter space.

2.2.1 LSTM architecture

The fundamental component of a LSTM network is the LSTM unit. Analogous to the standard DMLP networks, a set of LSTM units form a LSTM layer. By stacking LSTM layers in between the input layer and the output layer, we form the LSTM network.

However, in distinction to the neuron of a standard DMLP set-up, the LSTM unit has a more complex structure (see Figure 2.1) which allows for a time dependent cell state with an internal memory capable of storing past information and updating itself with new one. To do so, the input is passed through the LSTM unit sequentially, in form of timesteps t.

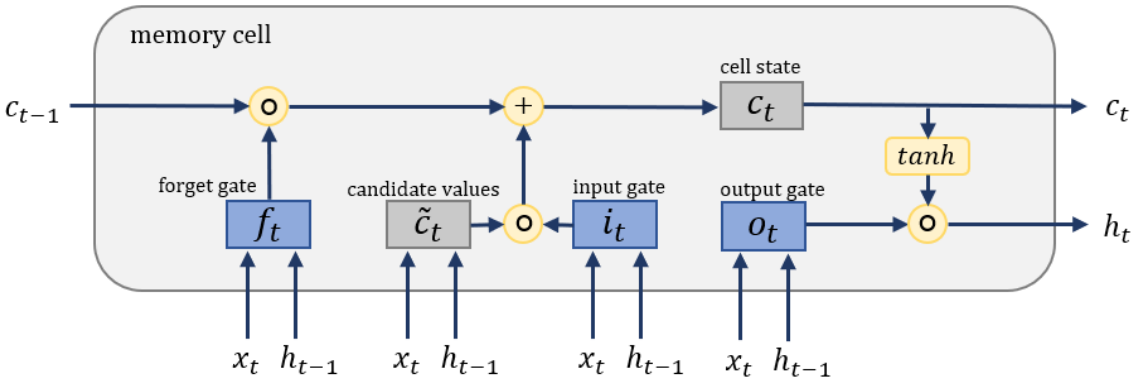

Figure 2.1: Schematic design of an LSTM unit

The structure of an LSTM layer can be described mathematically in vectorized form, for each timestep t. Four components build each LSTM unit and return the vector cell state c_t and the output vector h_t at time t. These are the forget gate f_t, the input gate i_t, the output gate o_t and the candidate values $\tilde{c}_t$ to enter the cell state. They are fed at each timestep with the input vector x_t and the past output vector of the LSTM layer h_{t-1}, as follows:

$$\begin{aligned} f_t &= \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \\ i_t &= \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) \\ o_t &= \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) \\ \tilde{c}_t &= \tanh\left(W_{\tilde{c}} x_t + U_{\tilde{c}} h_{t-1} + b_{\tilde{c}}\right) \end{aligned} \tag{2.14}$$

where W_j and U_j are weight matrices and b_j are bias vectors, with $j = \{f, i, o, \tilde{c}\}$. They are time-independent and conform the parameter space of the LSTM layer. Note that, f_t, i_t, o_t and $\tilde{c}_t$, are nothing else than linear combinations between x_t and h_{t-1} ($z_{t,j} = W_j x_t + U_j h_{t-1} + b_j$) squeezed into the differentiable, monotonically increasing functions $\sigma(z)$ and $\tanh(z)$:

$$\sigma(z) = \frac{1}{1+e^{-z}} \quad ; \quad \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.15}$$

These functions are called activation functions, which basically bound the input z that are given into the (0,1)

range for the sigmoid function $\sigma(z)$ and into the (-1,1) range for the hyperbolic tangent function $\tanh(z)$. They apply element-wise to the vector $z = (z_1, \ldots, z_n)$, that is $f(z) = (f(z_1), \ldots, f(z_n))$. Hence, the three gates compute a value, for each LSTM unit of the layer at timestep t, between 0 (the gate is fully closed and the information going through that gate is blocked) and 1 (the gate is fully open and the information going through that gate will flow). Note that the gates can also be partially open (intermediate values), as the activation functions are continuous. In contrast, the candidate values for the cell state $\tilde{c}_t$ contain values from -1 to 1 for each LSTM unit of the layer.

The cell state is then updated based on the new candidate values $\tilde{c}_t$ that arrived at timestep t and on the lagged cell state c_{t-1}, as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2.16}$$

where $\odot$ represents the Hadamard-product (i.e., element-wise multiplication between the components of the two vectors). The interpretation of the gates becomes clearer here. The forget gate actuates over the lagged cell state. Information of past cell state will be removed (forgotten) from the actual state if the forget gate is closed (values near to 0), whereas c_t will store information of the past as long as the forget gate is open. Likewise, the input gate actuates over the candidate values $\tilde{c}_t$ and decides whether to add this new information on the current cell state.

The current cell state c_t is then used to compute the output vector of the LSTM layer:

$$h_t = o_t \odot \tanh(c_t) \tag{2.17}$$

In this case, the output gate o_t determines whether the (squeezed) information of the cell state at time t will be propagated forward on the neural network.

2.2.2 Weight optimization and hyperparameter tunning

The number of parameters of an LSTM network depends on the number of layers and the number of nodes per layer of the network. Consider a hidden layer with N LSTM units that receives input from the previous layer from M different units. Therefore, the matrices W_j associated with the input vector x_t have dimension $N \times M$ (for $j = \{f, i, o, \tilde{c}\}$), while the matrices U_j, associated to the past output vector h_{t-1}, are square $N \times N$ matrices, since they refer to the elements of the same layer. Finally, the bias vectors b_j have dimension N. Since there are four internal components within each unit $(f_t, i_t, o_t, \tilde{c}_t)$ the number of parameters for the hidden layer i is expressed as:

$$\text{layer}_i = 4N_iM_i + 4N_i^2 + 4N_i \tag{2.18}$$

For a network with k hidden layers, the total number of network parameters is simply the sum of the parameters of the hidden layers added to the number of parameters of the output layer:

$$\#\text{ param} = \#\text{ output} + \sum_i^k \text{layer}_i \tag{2.19}$$

The output layer is typically formed by a dense layer of neurons as in plain DMLP networks. Hence, the number of parameters of the output layer is: $\#\,\text{output} = N_{out}M_{out} + N_{out}$. Note that the number of input features M_i of the layer i corresponds to the number of units of the previous layer N_{i-1}, that is, $M_i = N_{i-1}$.

As it can be seen, the number of parameters in an LSTM network scales quadratically with the number of neurons per layer and linearly with the number of layers. Together with the number of variables (features) of the input layer and the number of outputs that we want the model to return, these factors determine the specific architecture of the LSTM network and define the parameter space. Once the weights are fixed, the model is uniquely defined: it computes an output every time it is fed with an input. The weights are the so-called internal or structural parameters of the network. Initially, they are fixed in a random way to be later recursively updated in the learning phase.

For the adjustment of the internal parameters, it is necessary to firstly define a loss function. For predictive regression problems (as opposed to classification problems), where one wants to predict the quantitative continuous value of a target variable (or set of variables) Y, it is common to use the mean square error (MSE) or the mean absolute error (MAE) loss functions. For a sample of n observations, with real (target) value Y_i and predicted value $\hat{Y}_i$, with $i \in (1,2,\ldots,n)$, these are calculated as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \quad ; \quad \text{MAE} = \frac{\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right|}{n} \tag{2.20}$$

Generally, in order to achieve high levels of abstraction and generalization over the input data, deep learning models require several internal layers and a considerable number of neurons. Since the number of internal parameters in the model scales very quickly, this often becomes a high number and therefore large amounts of data are required to be able to estimate the parameters robustly and obtain good predictive performance. This implies a very intensive computational cost when calculating the loss function for the entire sample and recursively updating the parameters. To overcome this situation, the *stochastic gradient descent* SGD (or *mini-batch gradient descent*) algorithms are implemented, where the training sample is divided into batches of size n ($n = 1$ for pure stochastic gradient descent), and the weights are updated based on the respective gradients of each different subsample that is taken in each iteration. With this technique, the learning process is computationally more efficient and with an adequate batch size it is possible to control the variance of the estimated parameters and a stable convergence towards a minimum is usually obtained [16].

Hence, we define the batch size as the number of samples processed before updating the internal model parameters. Additionally, epochs are defined as the number of complete passes that the learning algorithm does through the whole training dataset. These two factors are set in the training phase of the model, and largely determine the training time. The larger the batch size, the longer it takes to update the parameters and therefore the learning becomes slower. Meanwhile, for small batch sizes, the convergence to a minimum occurs earlier, although the stability in the convergence is not ensured and the variability in the estimated parameters is typically higher. Regarding the number of epochs, the higher they are, the more times the algorithm will go through the training sample, which implies a higher time cost, but at the same time a greater ability to refine the minimum of the loss function.

At present, there are multiple gradient-based algorithms for optimizing the model weights and minimizing the cost function. For LSTM networks, as well as for other recurrent type networks, the error flow must also be propagated through previous timesteps. In such cases, Backpropagation Through Time (BPTT) algorithms are used, which are computed in a similar way to standard DMLP models but extending the partial derivative chain rule back in time [17]. Specifically, for this project, we have opted to use the Adam algorithm [18]. This is an optimization method that estimates the first and second moments of the stochastic gradient (with a subsample of batch size n) of the loss function and adds a bias correction to both estimators. This has proven to be a computationally efficient optimization method, suitable for a large majority of problems and frequently outperforming vanilla SGD and other reference algorithms such as RMSprop or Adagrad [16, 18].

In contrast to the internal parameters, the factors related to the training of the model (such as the optimizer choice, the learning rate) and the factors that describe the structure of the network (its depth, the number of units per layer, the choice of the activation function) constitute the hyperparameter space. The hyperparameter space is rather a (potentially infinite) list of possible models and architectures that can be assembled for a given input-output tuple, and each one of these models contains a series of weights which are fitted in training through the specified learning process. Given a loss function, the objective is not only to fit the internal parameters of a particular specific model, but also to estimate which of all the possible models that can be specified is the one that minimizes the loss function (*i.e.*, to find the best possible set of hyperparameters).

In conclusion, the predictive performance of neural networks depends on many aspects. Mainly, the elements that most influence the results are the quality of the information provided by the explanatory variables, the network architecture, the optimization methods used and the choice of the hyperparameters. However, these factors are usually limited by the experimental context of the problem and frequently the results are a trade-off between the desired precision threshold and the computational and economic cost of training such complex models.

Chapter 3

Data

Our study is based on the CISS series of financial stress for the countries or regions that we detail below. On one side, we have chosen the CISS series for France, Germany, Portugal, Spain, England and Italy, since they represent the financial stress of the main countries of the European region. On the other side, we have also selected the CISS series for China, the United States and the CISS for the Euro area as a whole, as they are three of the regions with the greatest global economic and financial activity. In the next section we will give a brief introduction to the CISS indices and explain how they are built. After that, we present the data and its main descriptive characteristics.

3.1 CISS construction

The Composite Indicator of Systemic Stress (CISS) is an index build in [2] which summarizes the levels of stress of the financial system for a specific region. It belongs to the family of the financial stress indices that aim to capture the conditions of instability of the financial system as a whole, by comprising into a single statistic a set of individual indicators that each measure, in some way, the existing risk within a certain segment of the financial system. These indicators have spread after the global financial crisis of 2007-2009, as authorities and major financial institutions became aware of the need of such tools and instruments to measure and monitor the health of the financial system. Apart from the CISS indicator, examples of such indices are the St. Louis Fed Financial Stress Index (STLFSI3), the EMFSI proposed in [19] for emerging market economies, the National Financial Conditions Index (NFCI) of the Federal Reserve Bank of Chicago, or the European Financial Market Stress Index (EFMSI) [20].

The CISS indicator is currently being used and reported daily by the European Central Bank, and it is arguably one of the preferred choices in the European area by financial authorities, regulators and major banks [21]. In what follows, we briefly describe its statistical design and methodology, extracted from the original papers [2, 22].

The CISS is composed by a set of individual financial risk indicators (15 time series for most regions and 16 for the China CISS region). These indicators are categorized into one of the next five segments of the financial

system, which are thought to be a good representation of it: the money market, the bond market, the equity market, financial intermediaries, and the foreign exchange market. Each of these 5 segments typically contains 3 of these individual risk indicators.

The choice of the individual risk indicators (variables of the CISS index) that compound the CISS indicators is made from a fundamental and economic point of view. It may vary according to the characteristics of each area of study and depending on what individual indicators are considered to best manifest the risk of the segment to which they belong. In addition, it is important that the information provided by the variables of a segment is complemented rather than being all explaining the same phenomena. What is more, the choices of variables should emphasize on two conditions. First, they should be available, or at least comparable, for different regions so that consistent CISS indicators can be constructed for various areas. And second, they should have a long enough record, so that they capture past financial stress events. Examples of individual indicators that meet these conditions and appear in most of the CISS segments are: asset return volatilities of series that are representative of its market, risk spreads (such as interest rate spreads and yield spreads) and liquidity indicators.

Once the indicators have been obtained in the form of raw time series, a transformation of each individual indicator is made according to its cumulative empirical distribution function. Being the time series x_t with the timestamp t ranging from 1 to the total number of observations n, we order the sample according to its value $(x_{[1]}, x_{[2]}, \ldots, x_{[n]})$ so that it fulfils $x_{[1]} \leq x_{[2]} \leq \ldots \leq x_{[n]}$. Here, $[r]$ represents the rank order assigned to a realization of x_t. The transformation is calculated as follows:

$$z_t = F_n(x_t) = \begin{cases} \frac{r}{n} & \text{for} \quad x_{[r]} \leq x_t \leq x_{[r+1]},\ r = 1,2,\ldots,n-1 \\ 1 & \text{for} \quad x_t \geq x_{[n]} \end{cases} \tag{3.1}$$

However, to allow efficient exploitation and real-time updating of the index, the empirical cumulative distribution is computed for each variable only for an initial sample period (of length n), and the subsequent observations are computed using the recurrence:

$$z_{n+T} = F_{n+T}(x_{n+T}) = \begin{cases} \frac{r}{n+T} & \text{for} \quad x_{[r]} \leq x_{n+T} \leq x_{[r+1]},\ r = 1,2,\ldots,n-1,\ldots,n+T-1 \\ 1 & \text{for} \quad x_{n+T} \geq x_{[n+T]} \end{cases} \tag{3.2}$$

where $T = 1,2,\ldots,N$ and N is the end of the full data sample.

The transformed variables are bounded on the interval (0,1]. This is a great advantage in terms of the homogenization of the variables in their scale and range, and it is much more robust than the classic approach of standardizing the variables by demeaning and dividing them by the standard deviation, which implicitly assumes that the variables are normally distributed. The transformed series are grouped into one of the main 5 above-mentioned segments of the financial system. For each segment, a subindex is created, giving an idea of the individual stress conditions in which each subpart of the financial system finds itself. These subindices are simply constructed as the equal-weighted aggregation of their elements, that is, by averaging the transformed

series:

$$s_{i,t} = \frac{1}{3}\sum_{j=1}^{3} z_{i,j,t} \tag{3.3}$$

where $i = 1,2,3,4,5$ stands for the different segments and $j = 1,2,3$ represents the individual risk indicators within each segment, at time t.

The last step consists of combining the sub-indices to form a single index that reflects the general stress level of the financial system. The underlying idea of this procedure focuses on considering time varying cross-correlations between the market segments. The CISS indicator should emphasize on those situations in which the stress is wide spreading across segments (high correlation), and it is then where it should display greater risk levels, indicating a potential collapse or crisis. However, in a situation of low financial stress, it may happen that one of the market segments presents periods of stress due to the own conditions of the sector, without this needing to rebound in a generalized increase in the system wide stress (in a low-correlated environment, an increase of the stress in one segment can just be as a response of its own idiosyncrasy). In addition, the subindices should be aggregated by weighting the importance that each one has on the real economy. These weights must be determined for each region of study, depending on the relation of their financial system with the economy. Normally, impulse response functions are estimated using VAR models, obtaining the average impact of each segment relative to the industrial production growth or other national economic series [2, 22].

That being said, the aggregation of the subindices into the composite indicator is computed as:

$$CISS_t = (w \odot s_t)\, C_t\, (w \odot s_t)' \tag{3.4}$$

where $w = (w_1, w_2, w_3, w_4, w_5)$ account for the subindices weights, $s_t = (s_1, s_2, s_3, s_4, s_5)$ are the subindices, C_t is the 5×5 time varying correlation matrix with elements $\rho_{ij,t}$ and $w \odot s_t = (w_1 s_{1,t}\ ,\ w_2 s_{2,t}\ ,\ \ldots\ ,\ w_5 s_{5,t})$ is the Hadamard-product (i.e. element-wise multiplication between the components of the two vectors at time t).

The correlation between market segment i and j at time t ($\rho_{ij,t}$) is approximated by means of an exponentially-weighted moving average (EWMA) model, where the covariances $\sigma_{ij,t}$ and volatilities $\sigma_{i,t}$ are computed as:

$$\begin{aligned} \sigma_{ij,t} &= \lambda\, \sigma_{ij,t-1} + (1-\lambda)\tilde{s}_{i,t}\tilde{s}_{j,t} \\ \sigma^2_{i,t} &= \lambda\, \sigma^2_{i,t-1} + (1-\lambda)\tilde{s}^2_{i,t} \\ \rho_{ij,t} &= \sigma_{ij,t} / \sigma_{i,t}\sigma_{j,t} \end{aligned} \tag{3.5}$$

for $i, j = 1,2,3,4,5$ and being $\tilde{s}_{i,t} = (s_{i,t} - 0.5)$ the demeaned subindex i, with theoretical median of 0.5. The lambda factor is set constant around a value of $\lambda = 0.93$, because it represents an intermediate choice that is well balanced between the persistence in volatility and the reaction to market events [22].

Hence, $CISS_t$ of equation 3.4 denotes the value of the CISS index at time t. Note that this index is unit-free, continuous and comprised on the interval (0,1].

3.2 CISS Overview

The CISS indices have been extracted from the metadata of the *Statistical Data Warehouse* of the European Central Bank (ECB), under its official naming *The New Composite Indicator of Systemic Stress (CISS), Index*, and with construction scheme as explained in section 3.1. We considered the following 9 regions: Germany, France, UK, Spain, Italy, Portugal, Euro Area, United States and China. During the project, we will refer loosely to the countries within the "European Union" or "Euro area" (CISS series of Germany, France, UK, Spain, Italy and Portugal) to differentiate from the three major global economies (CISS series of US, China and Euro Area), which we will call the "world area". Note that, technically, UK is neither part of the European Union since the Brexit nor it uses the Euro as his primary currency, but we include him in the analysis as it is geographically part of Europe and forms one of its largest economies.

The series have daily frequency, with observations being on trading days (approximately 1 trading year is composed of 252 observations). Our sample period consists of 5755 observations, ranging from 1st of September 2000 to 16th of September 2022, except from the China CISS, which starts reporting on the 10th of October of 2006 (to 16th of September 2022) and contains 4160 observations. Moreover, the China CISS contains 3 non-reported days from its sample period, due to desyncs with the Occidental trading calendar. To circumvent this problem, we imputed the 3 missing values with the value of the China CISS on the previous day.

The series are plotted below (see Figure 3.1 and Figure 3.2). One can readily observe that the dynamics of the series are quite similar, although each region slightly differs on the levels of stress reached in times of crisis. In general, it is observed a high level of market integration. Perhaps, the most dissimilar one is the China CISS (Fig. 3.2), which has presented generally lower levels of stress. By looking at the different regions of the European zone (Fig. 3.1), we can see that Germany has lead the major peaks from among his neighbours.

Examining its descriptive statistics, we find out that all time series studied follow the same pattern: they are positive skewed (to the tail of higher systemic stress levels) and they are leptokurtic, *i.e.*, presenting heavier tails than the normal distribution. Besides, the Jarque-Bera test rejects the null hypothesis of the individual indices being normally distributed.

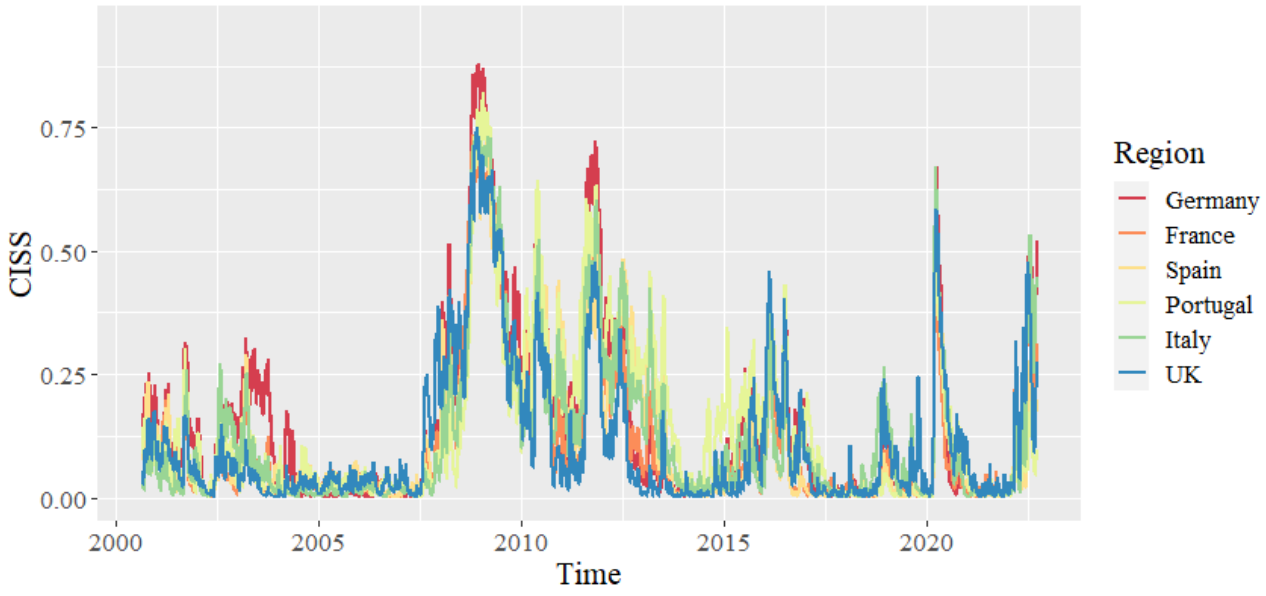

Figure 3.1: CISS series of the European countries.

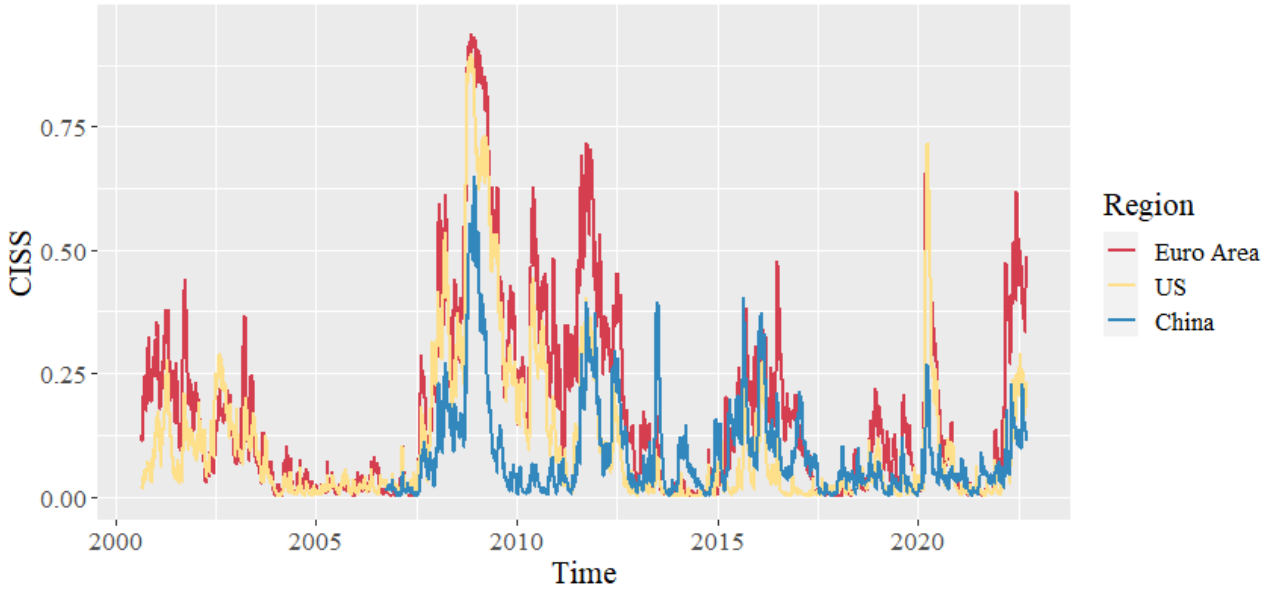

Figure 3.2: CISS series of the World area.

Since the index is constructed with the scope of capturing the levels of stress of the financial system, there is no better way to assess its utility but to look for the major high-stress events occurred historically within time frame considered and evaluate the ability of the index to identify such periods. To do so, we focus on the Euro Area index, shown in Figure 3.3, with the most significant historical stress episodes depicted with a vertical line.

First spike of stress appears on the 11th of September of 2001, after the terrorist attacks occurred in the US. In 2002, we visualize another upturn of the stress, coinciding with the crash of the dot-com bubble on the

United States, which ended in October 2002 (second dashed line), after the NASDAQ had capitulated to its local minimum, with a loss of about 75% with respect to its peak level. Another peak is observed no so long after that, on 19th of March of 2003, when a coalition led by the United States and Britain launched an air attack on Iraq, leading to the beginning of the invasion of Iraq.

None of these events had as much impact as the Global Financial Crisis of 2007-2008. First signals appeared in August 2007, after French bank BNP Paribas suspended three of its investment funds linked to the U.S. subprime mortgage sector. Later in March 2008 we observe another local maximum, coinciding with the collapse of Bear Stearns. Finally, in September 2008, Lehman Brothers is filed for bankruptcy and AIG is rescued. Just two months later, after the financial crisis had extended worldwide, we reach to the maximum value of the series (0.94). The second highest peak is observed on late 2011, embedded in the context of the European sovereign debt crisis. We highlighted two episodes which caused a high increase on the CISS levels on that time: in May 2010, when several events related to the Greek crisis occurred and fears of default increased, and in August 2011, after fears of contagion of the sovereign debt crisis in Portugal, Italy and Spain and a consequent stock market fall worldwide.

Three more clusters of high stress are identified from 2012. First one corresponds to the 2015–2016 stock market selloff, added to the uncertainty created around the referendum on the UK's membership of the European Union, with its peak in June 2016, when the Brexit was finally voted. Later, a sharp peak is observed in March 2020, due to the outbreak of the Covid-19, which caused serious levels of stress in the Euro Area, although they vanished equally fast, returning to the low band of the CISS. Last highlighted episode started in 24th of February 2022, with Russia's announcement of a "special military operation" in Ukraine, which triggers the invasion of the country, with persistent effects until the end of the sample period.

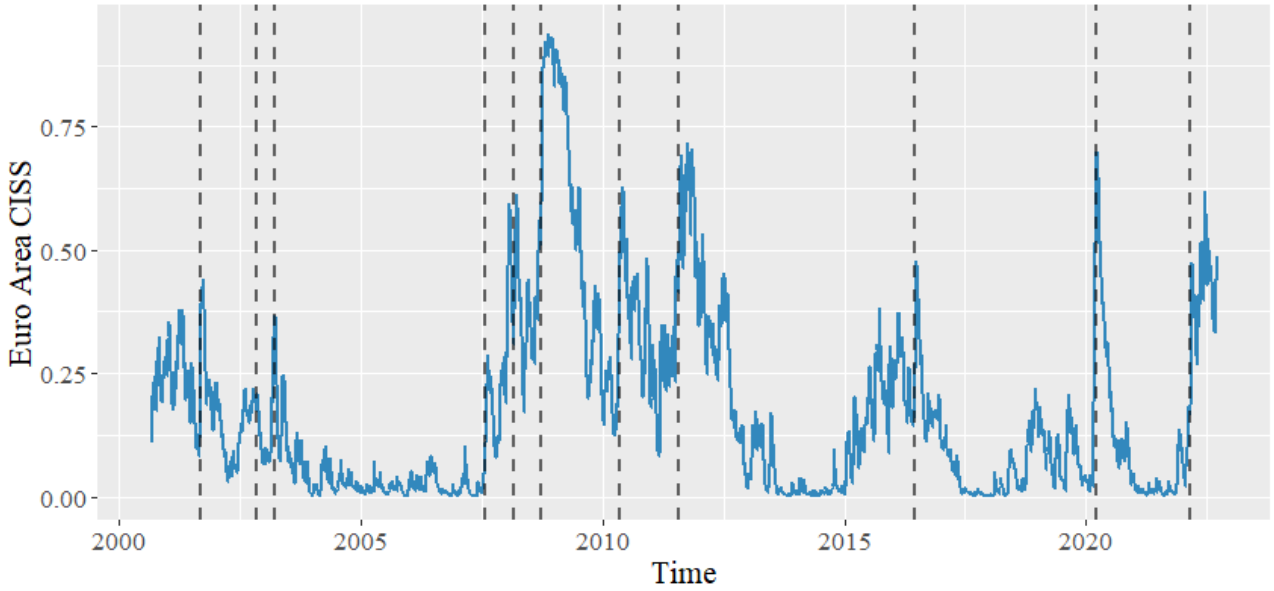

Figure 3.3: CISS series of the Euro Area superimposed on relevant historical market events.

As a conclusion, it seems clear that these indices reveal important historical events that have clearly had an impact on the financial system, and hence, we hope that the explanation just presented serves to appreciate the usefulness of the CISS indicator and to contextualize the results that we will discuss in the next section.

Chapter 4

Results

4.1 Systemic risk connectedness

Here, we present and analyse our results on the connectedness of the systemic risk within the countries of the European Union. Later, we carry on the same analysis between US, Euro Area and China, *i.e.*, the world area, and compare our findings between the two sets studied.

Following the standard methodology in the framework of vector autoregressive models, it is preferred to use stationary variables, so that we can estimate the VAR in level. To do so, we work with the differenced time series of the CISS indices to ensure stationarity and better properties on the estimation of the model[1]. Even so, the interpretation of the connectedness measures remains the same, as the forecast error variance decomposition is robust to such changes.

The order of the VAR model, the parameter p, is chosen according to the Hannan-Quinn Information Criterion (HQ). It is a measure of the goodness of fit often used in autoregressive models, which includes a penalty term for the number of parameters in the model [23]. Hence, according to HQ, we use $p_{euro} = 2$ for the VAR model on the European region and $p_{world} = 4$ for the VAR model on the world area.

With regards to the choice of the predictive horizon H for the obtention of the H step-ahead forecast error variance, we have taken $H = 10$, which represents a two-weeks ahead horizon (10 trading days) for the shocks of one of the variables to influence the forecast error variance of the others. In the Appendix A (Fig. 5.1 and 5.3) we show an analysis of the sensitivity of the total connectedness index for a range of values of H that are typically used in the literature [3, 24, 25]. We can see that for our series of study, the choice of H have little impact on the results.

With respect to the choice of the window width w for the dynamic case, we opted to use $w = 250$ for both models, corresponding to a full trading year. Sensitivity of the index for different width choices is shown in the Appendix A (Fig. 5.2 and 5.4). It is seen that the dynamics and the level of the total connectedness index

[1] We assess stationarity by using the Augmented Dickey-Fuller Test (ADF), the Philips-Perron Test (PP) and the KPSS Test. Our results conclude that the CISS series are I(1).

are preserved, so our results can be considered robust to the choice of w.

4.1.1 Systemic risk connectedness between European countries

The connectedness table for the European region is depicted in Table 4.1. Focusing first on the variance decomposition matrix (the pairwise elements), we observe that the highest contributions for each country come from its own variance share terms (diagonal elements). However, these do not represent in any case more than half of the total variance, being the contribution from others $C_{i\leftarrow\bullet}$ greater than the self-connectedness.

The highest levels of pairwise directional connectedness $C_{i\leftarrow j}$ are found between France and Germany, with $C_{France\leftarrow Germany} = 17.37$ and $C_{Germany\leftarrow France} = 17.16$. These coincide in being the most important countries of the European Union in terms of their economy. After these, the highest levels of pairwise connectedness occur for $C_{Spain\leftarrow France}$ and for $C_{Portugal\leftarrow Spain}$, which makes sense as they are neighbouring countries. The least interrelated country, speaking in terms of systemic risk, is the UK, both in terms of variance transmitted $C_{\bullet\leftarrow UK}$ and received $C_{UK\leftarrow\bullet}$. These results are in line with his recent history efforts of disassociating from the European Union, making his exit official in 2016 with Brexit. Even so, the interdependencies in its financial system with the rest of the European countries continue to be relevant and it still shows significant levels of connectedness.

As for the total directional connectedness to others $C_{\bullet\leftarrow j}$, Germany and France are the largest transmitters of systemic risk in the European region, with levels above 73%. The net measure of total directional connectedness indicates that, in mean, Germany, France, Spain and Italy are net transmitters of systemic risk, being Germany and France leading again. Meanwhile, Portugal and UK, notably with the biggest net value in absolute terms $C_{UK} = -10.59$, are net receivers of risk from the rest of the regions.

Overall, it seems clear that European regions exhibit high levels of systemic risk connectedness. This is what summarizes the total connectedness measure, which ascends to $C = 62.89\%$.

	Germany	France	Italy	Spain	Portugal	UK	C. from others
Germany	34.28	17.16	13.71	13.53	11.75	9.56	65.72
France	17.37	32.37	13.54	15.07	12.33	9.33	67.63
Italy	14.56	14.41	35.47	14.23	13.20	8.12	64.53
Spain	14.29	15.66	14.41	34.31	14.47	6.86	65.69
Portugal	13.88	14.22	14.26	15.66	36.33	5.64	63.67
UK	12.92	12.11	10.29	8.27	6.51	49.90	50.10
C. to others	73.02	73.57	66.21	66.76	58.27	39.51	**62.89**
Net C.	7.31	5.93	1.68	1.07	-5.4	-10.59	

Table 4.1: Connectedness table of the European region

Once seeing the connectedness from an "unconditional" point of view, we present the connectedness measures

in a time-varying environment. We start from the macro perspective and later we will break down into the more localized measures. The time-varying total connectedness is depicted in Figure 4.1, superimposed on the Euro CISS index for contextualization.

Its value mostly ranges from 40% to 70%, with a historical minimum in November 2014 (32.1%) and a maximum in March 2020 (83.0%), when the first Covid restrictions were established. A greater variability of the total connectedness can be observed as time progresses, with higher highs but also more spikey lows being observed. From late 2004, a gradual increase in the total connectedness can be observed until the end of the sovereign debt crisis, approximately in 2012. After a sharp drop in total connectedness, another period of increase is observed around the events of 2015-2016. The largest periods of connectedness in the sample are observed in the stage marked by Covid, stabilizing around 75% for more than a year after the shock caused by the lockdown, even when the systemic risk had already dropped to much lower levels. In the latest observations, a significant rise in total connectedness can also be seen, surely caused by the practically unanimous cooperation of European countries to deal with the Russian crisis. It is noteworthy that, although there seems to be a relationship between the increase in connectedness and systemic risk (measured by the Euro CISS index), this does not necessarily have to be the case. As observed in 2008, the risk was extreme while the levels of connectedness were only slightly higher from the observed mean, thus having the countries endogenous variance shares an important contribution in the variance of the forecast error.

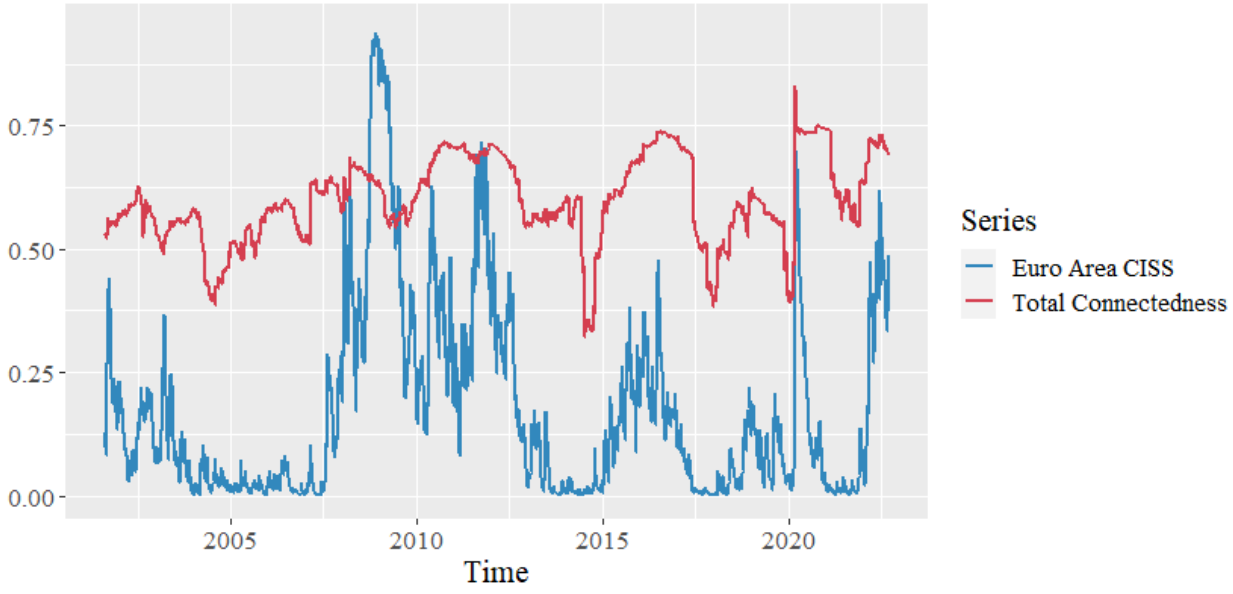

Figure 4.1: Total Connectedness Index. European region countries.

The results obtained of the net total directional connectedness for the 6 countries of study are shown in Figure 4.2. Keep in mind that, positive values indicate that the country is a net emitter of systemic risk, whereas negative values indicate that it is a net receiver. We can get several insights from these. For example, at the onset of the Covid restrictions, we can observe two different cases. The first occurs for Germany, Spain, Portugal and the UK, where the effect of Covid can be seen as an extreme value or outlier, but it is not

observed to cause a permanent effect on the structure of the connectedness series. However, for France and Italy there is a clear level shift in the series, which lasts significantly over time. Precisely, Italy was the first country in the Eurozone to apply the lockdown due to its high levels of contagion among its inhabitants in the earliest phases of the Covid pandemic, which supposed (in net terms) a strong transmission of the systemic financial risk to the rest of the countries. Another observation to highlight is found in the crisis period of 2007-2008, where both Germany, France and the UK, the three European powers, registered an increase in the net levels of connectedness with respect to their average levels, thus becoming transmitters of systemic risk and expanding it to the rest of the countries.

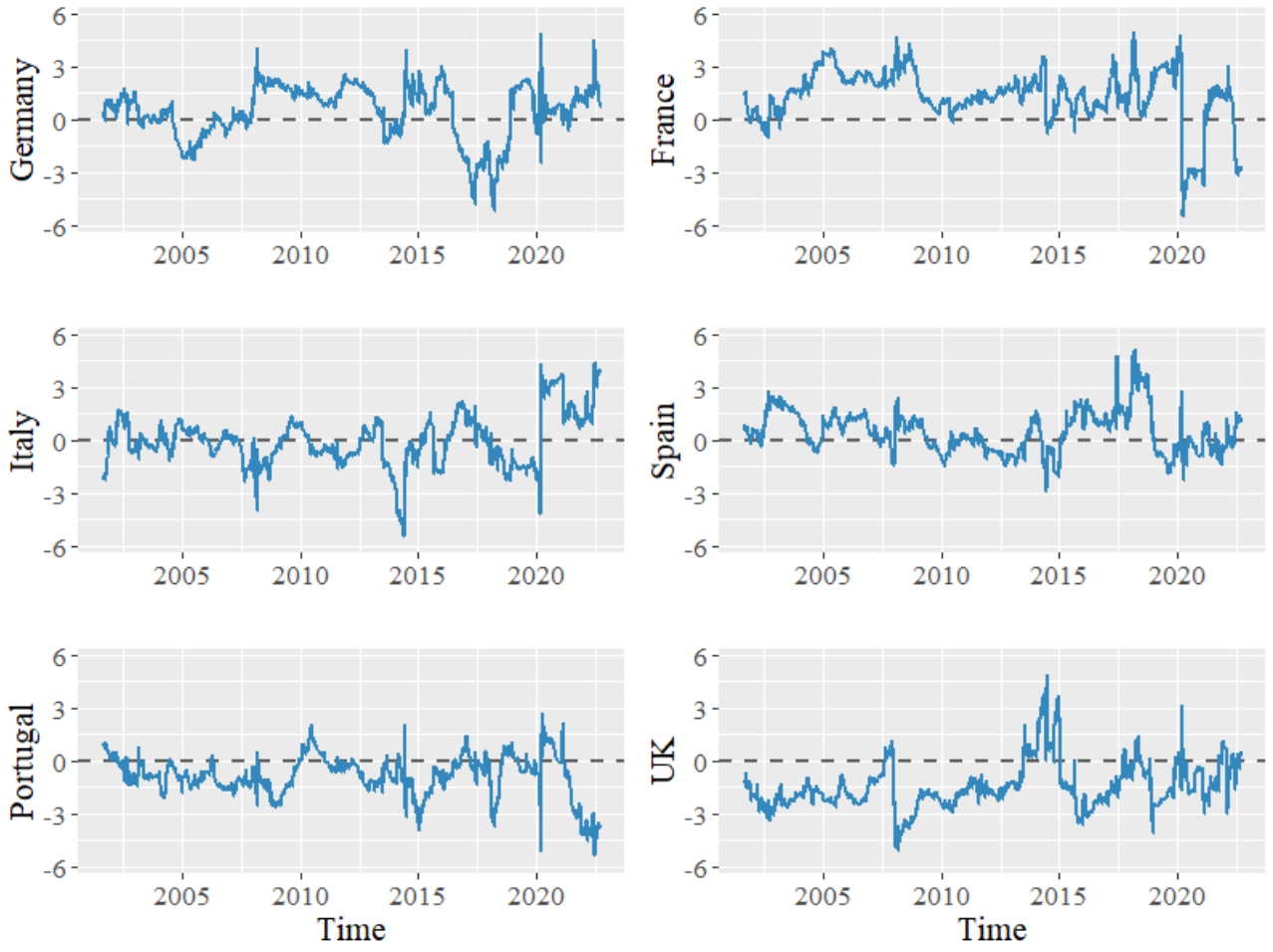

Figure 4.2: Net Total Directional Connectedness Index. European region countries.

Regarding the pairwise measures, being $N = 6$ the number of regions considered, note that there are $N^2 - N$ separate pairwise directional connectedness measures and $\frac{1}{2}(N^2 - N)$ net pairwise directional connectedness measures. Since our goal does not consist on doing an exhaustive analysis of all the possible combinations and for the sake of the extension of the research project, we limit ourselves on presenting only pairwise measures between the countries Germany and France and between Spain and France, since they were the ones that presented higher levels of pairwise directional connectedness in the static case (Table 4.1). The results are shown in Figure 4.3.

In the first place, we can observe that between complementary series of pairwise directional connectedness ($C_{i\leftarrow j}$ and $C_{j\leftarrow i}$), they tend to have similar structure and dynamics along the sample period, being the net terms C_{ij} originated when differences in level appear after certain events. A clear example of that is found after the shock originated by the Covid pandemic, where a positive net difference of around 5% persists in both graphs for approximately one year.

For Germany-France, we observe that the fractions of variance in the forecast error that are attributable to the opposite country range between 15% and 20% for most of the sample period. However, around 2005 and as of September 2017, we observe two episodes of strong decoupling between the risks of their financial systems, falling to levels of 5% and below. Interestingly, it is in the same period of 2017, when a substantial peak is observed in the pairwise directional connectedness levels between Spain and France, with a historical maximum of $C_{Spain\leftarrow France} = 32.6\%$. Hence, it can be argued that there was a transfer on the systemic risk of the French country, from Germany to Spain. These types of spillovers do not occur that commonly, but rather the decrease in the pairwise connectedness between two countries is usually absorbed in the self-variance share term of these countries or it is diluted in a more homogeneous way between the rest of the interrelated regions.

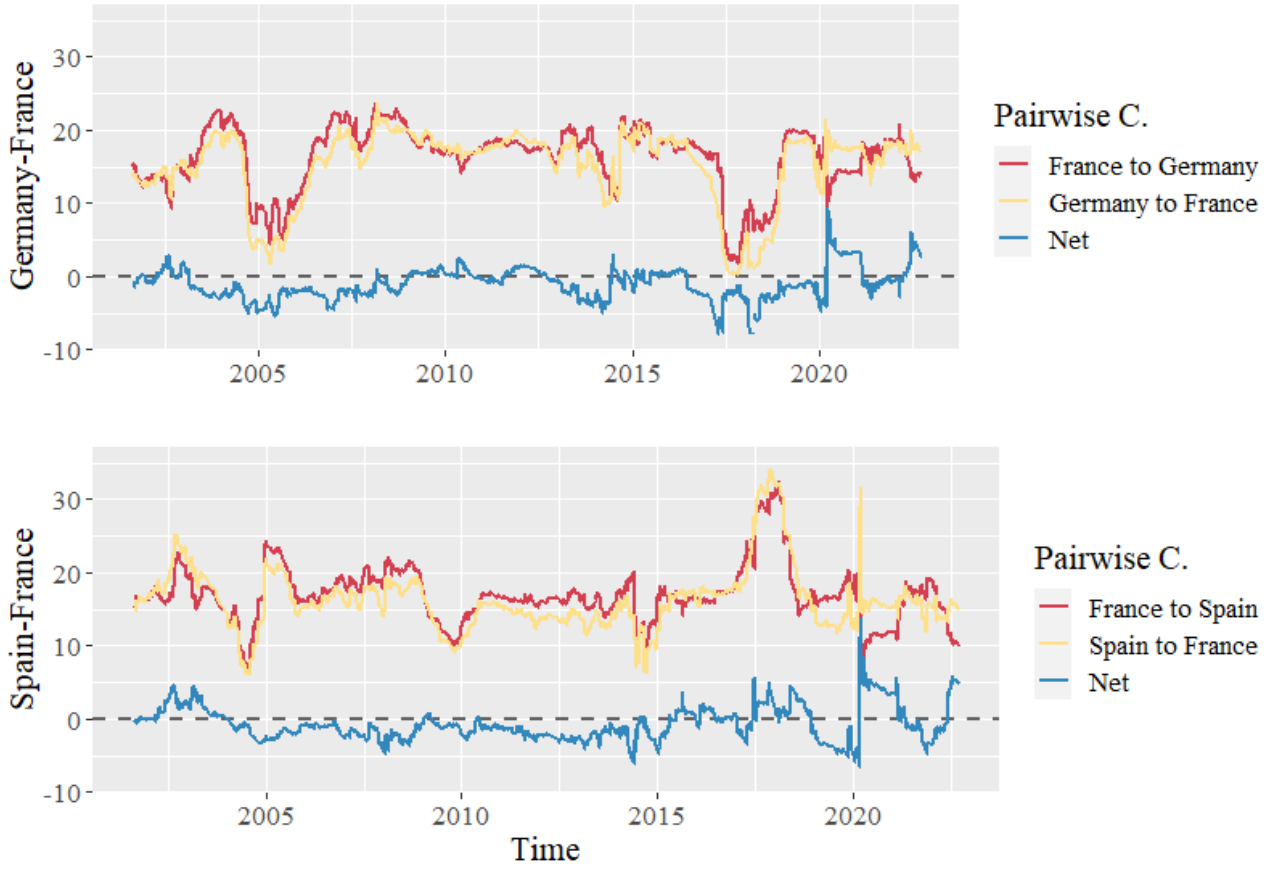

Figure 4.3: Pairwise and Net Pairwise Directional Connectedness indices for the pairs of countries Germany-France and Spain-France.

4.1.2 Systemic risk connectedness between US, Europe and China

The connectedness table of the "world" area is shown in Table 4.2. At first sight, we can see very similar values in magnitude of the pairwise directional connectedness to those seen in Table 4.1. However, this time they are overshadowed by the larger values of the diagonal terms (self-connectedness), which account for more than 80% of the total variance of the forecast error. Consequently, the system-wide connectedness measures (C. from others and C. to others) represent less than a quarter of the total variance. The same happens with the total connectedness measure, with a value of $C = 17.19\%$.

On the one hand, this phenomenon suggests that the three regions form a less interconnected financial network (in terms of risk) and therefore they are more independent and robust financial systems, which does not contradict the fact that they are regions highly connected in other sectors such as the commercial or the industrial. On the other hand, we must also keep in mind that any analysis of connectedness in an interrelated network (of regions) is clearly dependent on the variables (regions) included in the analysis. Thus, if we considered other major global financial regions[2], the self-connectedness term would potentially be lower, since there would be part of the variance that would be explained by these new variables, which in turn would result in an increase in the system wide connectedness measures.

That being said, we can observe several important aspects. We can see high levels of pairwise directional connectedness between the Euro Area and the US, while China is a fairly isolated region in terms of its financial systemic risk. Even so, the variance fractions it receives from the US and Europe double what it transmits to them, being the only region with a negative value of net total directional connectedness $C_{China} = -7.64\%$.

	US	Euro	China	C. from others
US	81.26	16.10	2.65	18.74
Euro	16.93	80.26	2.81	19.74
China	6.97	6.12	86.91	13.09
C. to others	23.90	22.22	5.45	**17.19**
Net C.	5.16	2.48	-7.64	

Table 4.2: Connectedness table between US, Euro Area and China

We now extend our analysis to the dynamic case. In Figure 4.4 we visualize the time-varying total connectedness measure. The series oscillates around the 20% value. The sample period for this case is smaller, as China CISS index is only available from 2006.

Like the analysis between European regions, the levels of total connectedness are not particularly high for the period of the global financial crisis of 2007-2008, while the maximum of the series is found in the financial

[2]Currently, from the major global economies, only US, China, and Euro CISS indices (apart from the European countries) are being reported publicly on the ECB Statistical Data Warehouse, so we could not include other economies of interest such as Russia, India, Japan or Brazil.

shock caused by Covid in 2020, with a value of $C = 42.45\%$. We also find periods with high connectedness coinciding with the events of the war between Russia and Ukraine and in the 2015–2016 stock market selloff. In addition, we can see a period of high connectedness between the financial systems in 2013-2014, in times when the systemic risk was stabilized at low levels.

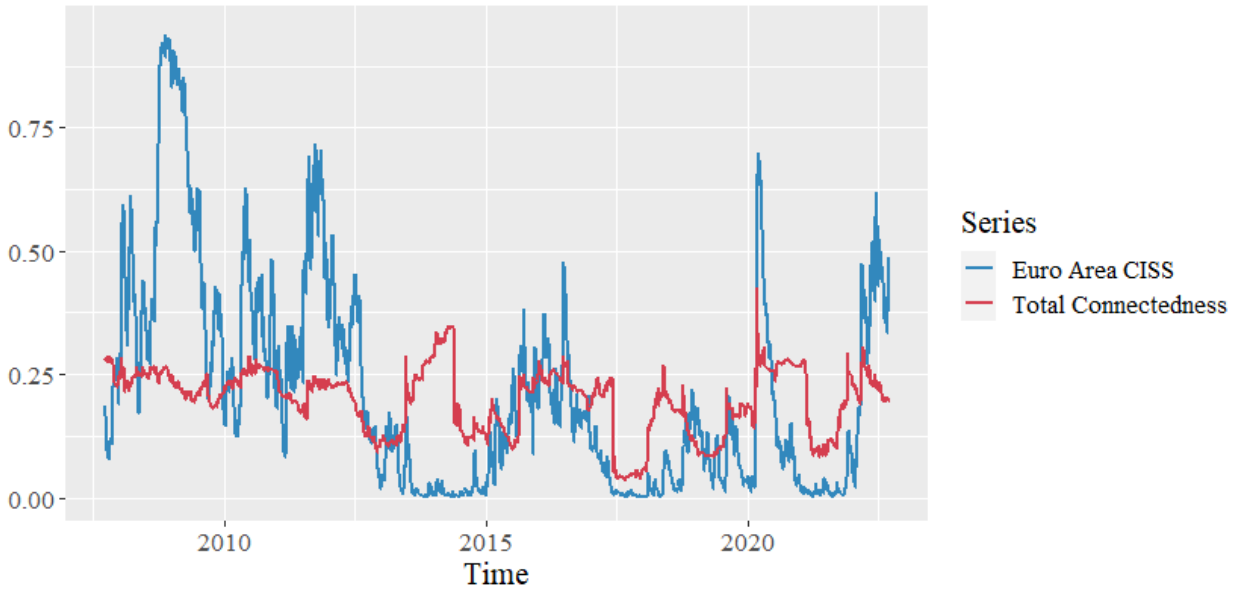

Figure 4.4: Total Connectedness Index. World area.

The net total directional connectedness measures obtained are shown in the Figure 4.5. In general, the series oscillate around their average levels of net total directional connectedness seen in the Table 4.2. For instance, we can observe during 2020 in the Covid period, that there is a significant increase in the transmission of systemic risk from the Euro zone to the rest of the regions. For China, it is generally observed that its value tends to stabilize close to zero in times when the systemic risk is relatively low, as occurs between 2015 and 2020. In contrast, for times with high financial stress it is usually the region with the greatest propensity of receiving contributions coming from shocks on the US and Europe.

Finally, we take a closer look to the United States and European block and analyse their measures of connectedness at pairwise level. These are shown in Figure 4.6. We can see how until Q3 2012, the two complementary series of pairwise directional connectedness ($C_{US \leftarrow Euro}$ and $C_{Euro \leftarrow US}$) are practically identical and consequently the net pairwise directional connectedness stays close to 0. As of 2014, the US begins to transfer a greater fraction of the forecast error variance to Europe and the series diverge slightly, creating a positive trade of net connectedness levels from the US to Europe. This is temporarily reversed during the Covid restrictions in Europe, which had a large impact on its financial stress and an increase in risk contagion levels, but returns to positive for the latest sample observations, relating to the conflict with Russia.

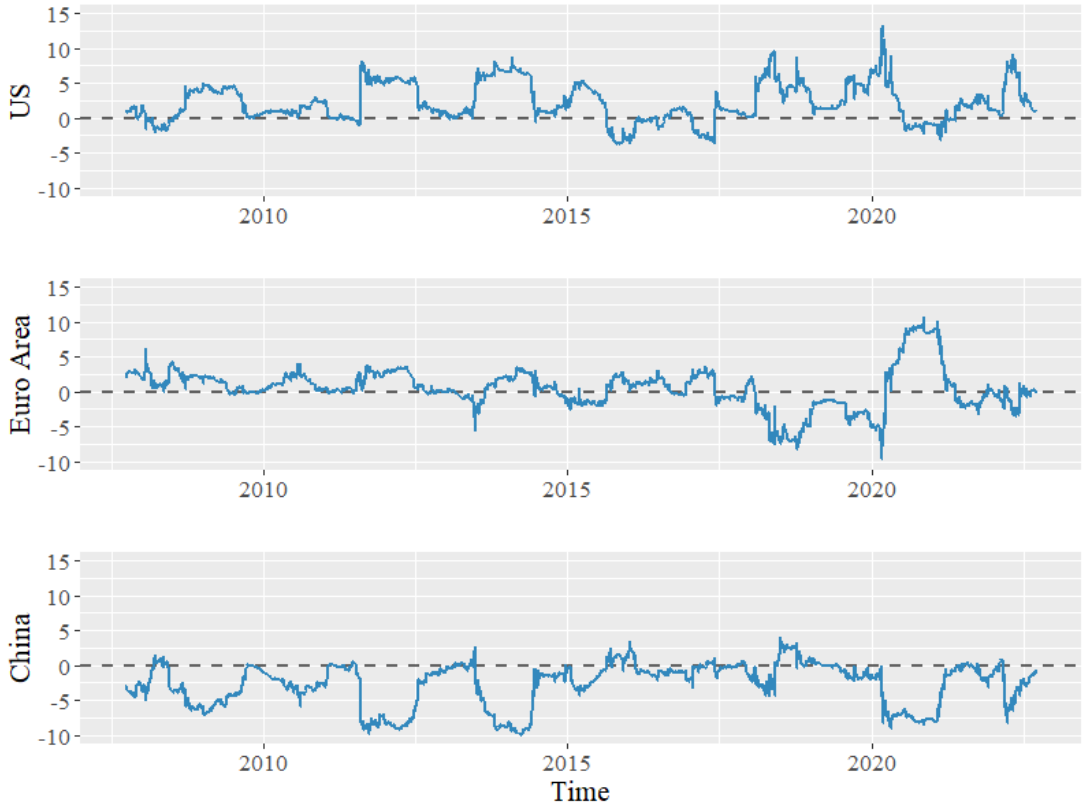

Figure 4.5: Net Total Directional Connectedness Index. World area.

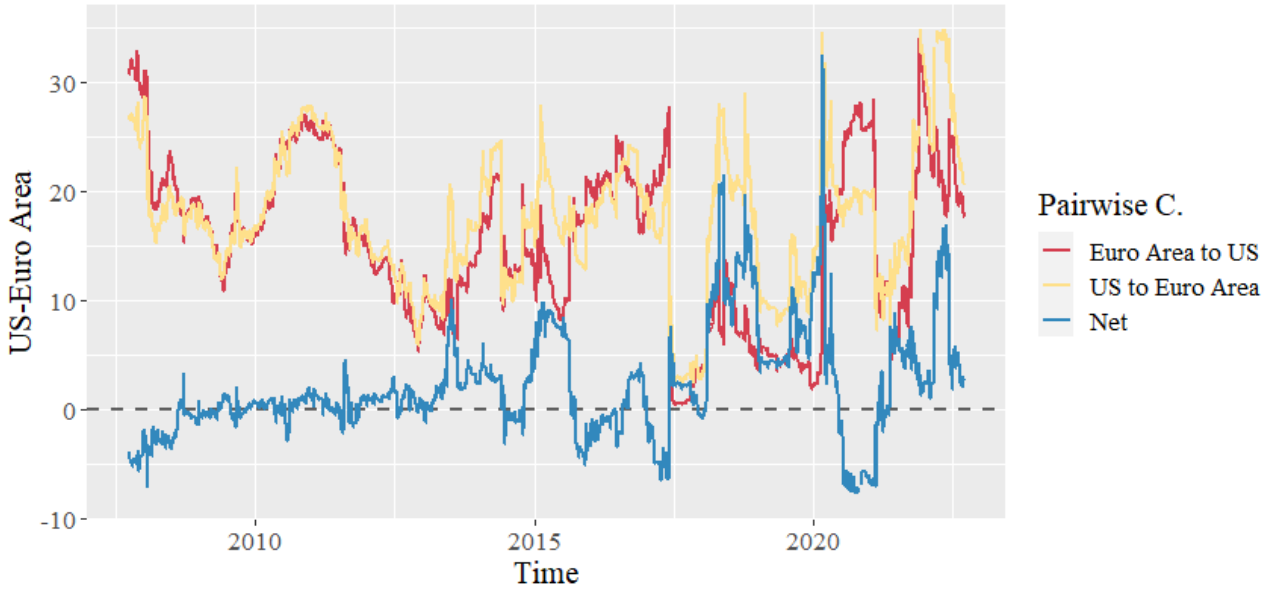

Figure 4.6: Pairwise and Net Pairwise Directional Connectedness indices for the pair US-Euro Area.

4.2 Prediction of systemic risk

In this section, we present and analyse the results obtained on the prediction of the CISS indicators, over the 9 previously defined regions. The goal of this section is to determine whether these financial stress indices present any kind of internal structure, based on the previous values of the series, that can be extracted and generalized by our predictive models. If so, these series could be used not only to assess the current observed systemic stress situation in each region, but also to use the predictions as a proxy for an early warning about the future development of the latent financial stress in each region.

Precisely, in this research paper, we will focus on the ability of LSTM neural networks to model this type of composite indices. We intend to provide evidence on whether these LSTM network models can provide added value over the already existing and widely used classical econometric models. Therefore, we will compare this type of recurrent neural networks, of a non-linear nature, against ARMA-GARCH models, of a linear nature in mean (and non-linear in variance). In addition, to have several views of the model's performance, we propose three different forecast horizons: $t+1|t$, for next day forecasts, and $t+5|t$ and $t+10|t$ for one week and two weeks ahead predictions, respectively.

For the evaluation of these models, we reserve a test sample, which is set to contain 10% of the total observations. This sample will not be used in any case to adjust the internal parameters of the model or to select a specific model among those considered, but will serve to determine its ability of generalization, on observations that have never been seen during the learning process. Due to the sequential nature of the time series, the test sample cannot be chosen randomly, since we would be introducing a bias in the evaluation of the model by using test observations contained among other past and future information that the model would have seen in beforehand during the training phase. This, of course, does not occur in a real experimental situation. Therefore, we choose the test set as the last observations of the sample period, which represent a total of 575 observations per region, from July 10, 2020 to September 16, 2022. This test set contains the latest remarkable stress episode observed, that is, the current war between Russia and Ukraine, with all the economic consequences that this has entailed worldwide.

In the following sections, we specify the benchmark models used in the comparison against LSTM networks. Subsequently, the specification and construction of the LSTM model is detailed. We later display the experimental results obtained and conclude with a final discussion on the topic.

4.2.1 Benchmark models

Here, we propose two different univariate approaches for forecasting the CISS indicators at different time horizons h. The two will serve as a benchmark for evaluating the performance of the LSTM network against the indices of systemic risk. First, we present a Naive approach, *i.e.*, the simplest model one could implement to forecast next values of a time series. It is a point of reference to distinguish between models that add some value in forecasting the specific time series of study (at horizon h) and models that are not capable

of extracting any pattern with the previous information available. Second, we propose classic econometric time series models, the so-called ARMA-GARCH models, which are autoregressive processes conditioned to the previous values of the time series and the previous error terms, which allow to compute time-varying estimates of its conditional mean and conditional variance (or volatility). We provide with basic context and mathematical formulation of the models and present related literature for further in-depth reading.

Naive approach

The naive approach consists of using the last observation available (at time t) and suppose the time series does not change in the following observation periods, so the best possible estimation of $x_{t+h|t}$ is the current value of $x_{t|t}$ itself:

$$\hat{x}_{t+h|t} = x_{t|t} \tag{4.1}$$

In practice, the series of forecasts is just the series itself lagged h times. This approach benefits, in principle, from time series that do not vary much in time, whereas it will compute worse forecasts for series with high variability.

ARMA-GARCH model

A sequence of observations X_t follows an ARMA (AutoRegressive Moving Average) process if its behaviour can be explained by its lagged values $X_{t-1}, X_{t-2}, \ldots$ and its past previous errors $Z_{t-1}, Z_{t-2}, \ldots$, plus a random component Z_t (also called innovation or disturbance) which follows a white noise process (independent and identically distributed, with expected value equal to zero and variance σ^2) [26,27]. However, often in financial statistics, the variance error is seen to be serially autocorrelated. We circumvent this problem by allowing the variance of the error term to follow an autoregressive moving average process, analogously to that followed by the model for the mean. In mathematical form, that reads:

$$\begin{gathered} X_t = \alpha + \phi_1 X_{t-1} + \ldots + \phi_p X_{t-p} + Z_t + \theta_1 Z_{t-1} + \ldots + \theta_q Z_{t-q} \\ Z_t = \sigma_t \varepsilon_t \quad \varepsilon_t \sim WN(0,1) \end{gathered} \tag{4.2}$$

where the parameters p and q indicate the order of the autoregressive (AR) and moving average (MA) parts of the model, usually specified as $ARMA(p,q)$.

Regarding the models for the variance, these are called GARCH models (Generalized AutoRegressive Conditional Heteroskedasticity). There exist many different variations within these models [28], but we have narrowed our study to only 3 of them, which are mainly the most used ones. These are presented below.

The GARCH(1,1) [29], models the conditional variance as a linear function of the past innovation term squared and the past variance term:

$$\sigma_t^2 = \alpha_0 + \alpha_1 Z_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \tag{4.3}$$

Sometimes, it is seen that the variance reacts asymmetrically to the market, *i.e.*, conditional variance generally increases more when the past innovation is negative compared to when the past innovation is positive, or vice versa. Such behaviour can't be explained in a plain GARCH(1,1) model, as the dependence between the past innovation and σ_t^2 is symmetrical. Extensions like the GJR-GARCH(1,1) [30], introduce a dummy variable S_t^- that takes into account this effect [31], by modelling σ_t^2 as follows:

$$\begin{aligned} \sigma_t^2 &= \alpha_0 + \alpha Z_{t-1}^2 + \beta \sigma_{t-1}^2 + \gamma S_{t-1}^- Z_{t-1}^2 \\ S_t^- &= \begin{cases} 1 & \text{if} \quad Z_t < 0 \\ 0 & \text{if} \quad Z_t \geq 0 \end{cases} \end{aligned} \tag{4.4}$$

Similarly, the EGARCH(1,1) model [32], is an alternative to the GJR-GARCH, which also takes into account the asymmetrical behaviour of the variance, but is formulated in terms of a logarithmic scale:

$$\ln\left(\sigma_t^2\right) = \alpha_0 + \beta \ln\left(\sigma_{t-1}^2\right) + \gamma \frac{Z_{t-1}}{\sqrt{\sigma_{t-1}^2}} + \theta \left[\frac{|Z_{t-1}|}{\sqrt{\sigma_{t-1}^2}} - \frac{2}{\pi} \right] \tag{4.5}$$

Estimation of the parameters for the presented ARMA-GARCH models is done via Maximum Likelihood Estimation (MLE). As stated previously (see section 4.1), the CISS indices do not present a clear stationary behaviour. Thus, we have opted to adjust the ARMA-GARCH models on the differenced time series to ensure stationarity. Specification of the orders p and q for the ARMA part is done via the sample ACF and PACF. Additionally, for selecting between the possible candidates, we relied on AIC and BIC criteria and ensured that all estimated coefficients were statistically significant. Then, we proceeded to choose a model for the variance. First, via an ARCH test and with the ACF and PACF plots of the squared residuals, we checked that there are indeed conditional heteroscedasticity effects on all the studied series. Besides, we found out the residuals to be heavy tailed and non-normal, so we allowed the error-term to be modelled as a t-Student. Among the 3 possible GARCH candidates, we based our decision on 3 different aspects. First on AIC, BIC and HQ information criteria. Second, on whether the estimated coefficients were statistically significant, emphasizing on the significance of the asymmetry coefficients for the GJR-GARCH and EGARCH models. We found that most of the regions considered do present such effect. Third, on whether the validation of the model is correct and if its assumptions are reasonably fulfilled, which are: no autocorrelation structure on the standardized residuals, no autocorrelation structure on the squared standardized residuals and whether the residuals follow the underlying fitted distribution.

The results obtained are shown in the next table:

Region	Model selected	Error term distribution
Germany	ARMA(4,0)-gjrGARCH(1,1)	t-Student
France	ARMA(4,0)-gjrGARCH(1,1)	t-Student
Italy	ARMA(1,1)-EGARCH(1,1)	t-Student
Portugal	ARMA(1,1)-gjrGARCH(1,1)	t-Student
UK	ARMA(1,1)-GARCH(1,1)	t-Student
Spain	ARMA(3,0)-gjrGARCH(1,1)	t-Student
EU	ARMA(4,0)-EGARCH(1,1)	t-Student
US	ARMA(1,1)-GARCH(1,1)	t-Student
China	ARMA(0,5)-gjrGARCH(1,1)	t-Student

Table 4.3: Selected ARMA-GARCH models for the differenced CISS time series.

We want to highlight the fact that even the model for the variance does not directly intervene in the calculus of point predictions, it does influence in the estimation of the parameters when the ARMA-GARCH models are fitted jointly. Besides, from the statistical point of view, we should always favour models that are robust and fulfil the initial assumptions, especially when the samples clearly show volatility effects.

Once the model parameters have been adjusted and validation is seen correct, we proceed to forecast the observations on the test sample. Note that, as the model is initially fitted to the differentiated series, we must inverse the differencing to recover the forecasts on the original CISS series. If $F_t^{(h)}$ is the h-step-ahead forecast of $\Delta x_{t+h|t}$, done at time t and therefore with observed values up to that point, then $\hat{x}_{t+h|t}$ fulfils:

$$\hat{\Delta x}_{t+h|t} = \hat{x}_{t+h|t} - \hat{x}_{t+h-1|t} = F_t^{(h)} \quad \Rightarrow \quad \hat{x}_{t+h|t} = \hat{x}_{t+h-1|t} + F_t^{(h)} \tag{4.6}$$

When $h = 1$, then $\hat{x}_{t|t} = x_{t|t}$ is known and we directly retrieve the forecast for the original series. However, when $h > 1$, we need to recursively compute the previous forecasts until arriving to our last observed value $x_{t|t}$, following the same logic as before:

$$\hat{x}_{t+h|t} = \left(\hat{x}_{t+h-2|t} + F_t^{(h-1)}\right) + F_t^{(h)} = \ldots = x_{t|t} + \sum_{i=0}^{i=h-1} F_t^{(h-i)} \tag{4.7}$$

Taking a closer look, we realize that the forecasts for the ARMA-GARCH models at horizon h contain the same term $x_{t|t}$ as in the Naive approach, plus a sum of terms linked to the forecasts of the differenced series up to the h-th horizon. As h increases, the $F_t^{(h)}$ term converges to zero, that is, towards the unconditional mean of the differenced series. Hence, since our time series do not present significant (lineal) autocorrelation structure beyond the 5th lag, we expect the performance of the ARMA-GARCH models to initially outperform the Naive approach, but to equalize as the horizon is extended.

4.2.2 Data engineering and LSTM tuning

This section explains the assembly process used for the LSTM network in the prediction of the CISS indices. We detail the aspects that we have considered in the data engineering process, the design and choice of the network architecture and in the learning stage. Also, we briefly comment on other modelling strategies that we have taken into account, but which have been finally discarded of the project as they did not add any significant improvement.

Note that, in the case of "black box" models, especially for neural networks, it is difficult to acknowledge at a fundamental level what is the internal logic or meaning behind each of the adjusted parameters and what interpretation can we give to them with respect to our experimental data. While classical statistical models are built under assumptions about their underlying distribution and contain interpretable relationships between their components, neural networks lack these characteristics and therefore the validation process shall be focused on its predictive capacity. Therefore, often the only way to choose a model among all the possible architectures and designs is to select the configuration that obtains the best performance metrics in a subsample reserved for this purpose, named validation sample. This can be advantageous in some cases, since no prior statistical assumptions need to be made for modelling our set of data. On the other side, we should be aware that it may introduce a bias in our chosen model, since the validation sample does not have to be robust with respect to the entire observed period, and the temporal structure of the series may vary in the test sample.

Generation of training, validation and test sequences

That being said, for our experimental problem, we have divided the total sample into the following 3 partitions. The training sample contains 75% of the observations (4317 observations for each region except for China, with 2722 observations), comprised from the beginning of the sample period to March 20, 2017. These observations are used to adjust the internal weights of each specific LSTM network. The validation sample contains 15% of the observations (864 observations per region), from March 21, 2017, to July 9, 2020. This sample is used to select the hyperparameters of the model, that is, the network architecture and training features. Finally, the test sample comprises the final 10% of the observations, that is, 575 observations per region, from July 10, 2020, to September 16, 2022, coinciding with the test sample used in the benchmark models, whose results are discussed in section 4.2.3.

Next, we generate, for each partition and each region, the input-output pairs of sequences in such a way that they can be ingested by the neural network. We define the input sequence $x^{(t,l)}$ as the ordered list of length l containing the last l observations since time t, including x_t. That is, $x^{(t,l)} = \{x_t, x_{t-1}, \ldots, x_{t-l+1}\}$. The parameter l, called sequence length, determines the number of past observations that we consider to predict future values of the time series. In our experimental problem, we decided to choose a sequence length of $l = 64$, which spans up to 3 (financial) months of observations prior to the forecast. We consider that this is a sufficiently long period of information with respect to the specified forecast horizons $h = \{1, 5, 10\}$. Even so, we must be aware that this choice may condition the results. The output $y^{(t,h)}$ associated with the sequence

$x^{(t,l)}$ is the real target value of the model at horizon h, that is, the value observed at $t+h$: $y^{(t,h)} = x_{t+h}$. Therefore, an input-output tuple can be written as a pair of arrays $(x := x^{(t,l)},\ y := y^{(t,h)})$, of shapes $(1,l)$ and $(1,1)$ respectively. Being n the number of observations for one specific region and partition, the total number of tuple sequences that can be generated from there equals $N = n-h-l+1$, as we have to subtract the first $l-1$ observations and the last h observations from the subsample to obtain complete (x,y) pairs. Hence, if we stack each tuple row-wise, the input-output set constitutes two paired arrays of shapes (N,l) and $(N,1)$, respectively for x and y.

Network design

Considering the mentioned partitions, we can arrange no more than 4253 sequences (of 64 timesteps) per region in the training phase. This supposes a major constraint in our problem, since the number of sequences that we could input to our network would be relatively low with respect to its potentially high number of parameters. For instance, an LSTM network for one region, with a single hidden layer of 18 units, would suppose a total of 1459 trainable parameters, which would imply a ratio of approximately only 3 training sequences for each estimated parameter. This may be undesirable since the network learns to memorize the training examples rather than abstract representations that later generalize with unseen data. Although we could restrict from the beginning the model to really small architectures, we have been able to verify in our trials that this approach does not lead to the best results in the forecast horizons considered. In the same way, a multivariate approach, with simultaneous forecasts of all regions, does not solve our limitation as the total number of different sequences that we can input to the network is the same, but being in this case the input at timestep t (x_t) a vector with 9 features, one for each country of study. Namely, the set of tuple sequences for the multivariate approach can be generalized to one tensor of shape (N,l,r) for the input and another of shape (N,r) for the output, where r is the number of features we fit at a time ($r=1$ for the univariate case and $r=9$ for the multivariate) and N, l are, as stated previously, the total number of sequences and the number of timesteps in each sequence, respectively.

To alleviate the issue of having a small dataset, we propose a univariant approach that combines the sequences from all the regions into one unique network. We do so by concatenating (stacking row-wise) the tuple sequences of all the countries within the same partition. In this way, we effectively multiply by almost 9 times (China has fewer observations than the others) the number of sequences that the network receives in its training phase. As we have seen both in the CISS construction methodology and in the study of their connectedness levels, the series present a very similar structure among them, with common characteristics. Therefore, what we intend to do with this approach is to generalize the behaviour of this type of financial stress indices into a single neural network, capable of capturing common structural patterns, while eliminating the individual noise of each of them. In this way, we can better fit architectures with few more layers and neurons without falling so quickly into overfitting problems. In addition, we implemented 3 measures in our design to reduce the risk of overfitting, described below.

The first measure consists of applying dropout and recurrent dropout rates between the neural connections

of the LSTM units with respect to the inputs and recurrent inputs they receive. That is, during the training process, a random fraction of the received inputs are dropped (are set to 0) in each unit with probability p, along with their connections. For the test and validation samples, the predictions are calculated with the network fully connected and the network weights are adjusted by averaging the weights obtained from the various partially connected networks in training [33]. This technique is potentially prevents the network from being too unbalanced and placing too much importance on only some of the units and weights.

For the second and third measures, we implement two call-backs on the validation set during the training phase. That is, at the end of each epoch, we evaluate the validation sample with the current weights of the model, using the same loss function as in training, and we save the result. Then, we define an action to implement if we observe that during a certain number of contiguous epochs, called patience period, the loss function in the validation set does not improve by at least an increment greater than Δ. The first action we implement is to reduce the current learning rate by a factor of 10, in order to fine-tune the direction of the gradient-descend as we approach the minimum. This allows us to dynamically adjust the learning rate of the model, making it learn faster at the start of training and more granularly as the learning stagnates, improving its computational efficiency. On the second action, we implement an early stopping of the model training if we observe that it no longer improves in the validation set, thus preventing the model from starting to overfit the training sample and losing generalization capacity over unseen samples. In this way, we can set the epochs to a larger number and let the results themselves decide when it is no longer worthwhile to continue training the network. Note that, when these two call-backs are implemented together, one should set the patience period for the early stopping to a number higher than the patience period needed to reduce the learning rate, thus leaving a reasonable margin for improvement potentially caused by a more accurate learning before definitively stopping the optimization process. For our experimental case, we have opted for a minimum Δ of improvement of $\Delta = 10^{-7}$, a patience period equal to 3 before reducing the learning rate and a patience of 7 epochs before early stopping the training.

Bayesian Hyperparameter Optimizer

Once the design of the neural network has been defined jointly with the characterization of the content and form of the input-output data and the overfitting prevention modules, the hyperparametric space is specified and the next step consists on finding which set of hyperparameters hp optimizes the loss function over the validation sample. Namely, this is $hp = (Nl, u_1, \ldots, u_{Nl}, d_1, \ldots, d_{Nl}, rd_1, \ldots, rd_{Nl}, lr, act)$, which stands for the number of LSTM layers (Nl), the number of units (u_i), the dropout rate (d_i) and the recurrent dropout rate (rd_i) in each LSTM layer $i = 1, 2, \ldots, Nl$, the initial learning rate (lr) and the activation function (act) of the output dense layer.

There are many techniques to adjust the hyperparameters of the model, the simplest being manual search or random grid search (creating a discretized grid and randomly trying possible combinations). In this project, we use a much more computationally efficient and statistically interesting technique, the so called Bayesian Optimization [34]. This technique uses the "Bayes' theorem" and other tools of Bayesian statistics to direct

the search within the hyperparametric space, through a recursive algorithm that updates our beliefs over the posterior distribution of an objective function based on the combination of prior knowledge that we have been accumulating from the first sampled trial and on the likelihood of the new results obtained given the current state of knowledge. We do so by assuming an approximating distribution of the objective function f, such as a Gaussian process (GP). In our case, the objective function is the loss function on the validation set. In a brief, being hp_i the i-th sampled set of hyperparameters (among all the possible configurations), we refer to $f(hp_i)$ as the value of the loss function evaluated at hp_i. The evidence set of observations accumulated up to trial t reads $\mathscr{D}_{1:t} = \{hp_{1:t}, f(hp_{1:t})\}$. Accordingly, the likelihood of the evidence set given the loss function follows the distribution $P(\mathscr{D}_{1:t} \mid f)$, while the prior distribution itself is $P(f)$. The multiplication of the last two terms is proportional to the posterior distribution:

$$P(f \mid \mathscr{D}_{1:t}) \propto P(\mathscr{D}_{1:t} \mid f) P(f) \tag{4.8}$$

At this point, an acquisition function $a(hp|\mathscr{D}_{1:t})$ is usually used to draw the next trial hp_{t+1} to evaluate on the loss function. The acquisition function is defined in such a way that its value is high where the GP predicts a high value of the posterior distribution (exploitation) or in the regions where the GP presents greater uncertainty (exploration). The whole process is then repeated several times, each step sampling a new trial from the hyperparametric space, up to the maximum number of trials that one specifies.

Note that, in our scenario, the number of layers and the number of units per layer can be theoretically as high as we want. Furthermore, the learning rate and the dropout rates are continuous variables, so infinitely many trials can be done. Clearly, it makes sense to bound and discretize our hyperparametric space, mainly because of computational cost but also due to the context of small sample data. We restricted our initial learning rate values for the Adam optimizer to $lr = \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-7}\}$, and the dropout rates to the interval $d_i, rd_i = [0,\ 0.25]$ in form of steps separated 0.05 units each other apart. We opted for two different possibilities for the activation function of the output dense layer: the *sigmoid* function and the *rectified linear unit* ReLU function ($f(x) = x^+ = \max(0,x)$). Finally, we limited the maximum number of LSTM layers to 4. We set a range of LSTM units per layer between 2 and 18 units (in steps of 2) for the configuration of only 1 LSTM layer, and between 2 and 10 units (in steps of 2) per layer for the configurations with 2 or more LSTM layers. In that way, we limited the minimum number of training sequences given per parameter to estimate to an approximate ratio of 13 in the worst-case scenario. Due to convenience in compilation and constraints on computational time[3], we have set the number of trials for the Bayesian hyperparametric optimizer to 15 for the 1 LSTM layer case, with a batch size of 16 and 35 epochs per sampled configuration, and a number of trials to 10 for the 2 to 4 LSTM layers case, with a batch size of 16 and number of epochs equal to 20.

4.2.3 Experimental results

The loss function defined in the optimization process has been the Mean Squared Error (MSE), both for the adjustment of the internal parameters on the training set and for the selection of the hyperparameters on the

[3] We have used Keras framework on top of TensorFlow 2, in python programming via Google Colab.

validation set. The selected final configurations of the hyperparametric space are shown in Table 4.4. In total, 3 different LSTM networks are obtained, each one optimized to predict the systemic risk indices in one of the time horizons h fixed. These are the configurations that have obtained the best predictive performance on the validation set within the trials carried out in the Bayesian optimization process; MSE $= 6.31 \cdot 10^{-5}$ for the $h = 1$ case, MSE $= 8.27 \cdot 10^{-4}$ for the $h = 5$ case, and MSE $= 2.80 \cdot 10^{-3}$ for the $h = 10$ case.

Hyperparameter	**Description**	**h=1**	**h=5**	**h=10**
Nl	LSTM hidden layers	1	4	1
u_1	N. units layer 1	2	10	18
d_1	Dropout rate layer 1	0	0	0
rd_1	Recurrent droput rate layer 1	0	0	0
u_2	N. units layer 2	-	10	-
d_2	Dropout rate layer 2	-	0	-
rd_2	Recurrent droput rate layer 2	-	0	-
u_3	N. units layer 3	-	2	-
d_3	Dropout rate layer 3	-	0	-
rd_3	Recurrent droput rate layer 3	-	0	-
u_4	N. units layer 4	-	10	-
d_4	Dropout rate layer 4	-	0	-
rd_4	Recurrent droput rate layer 4	-	0	-
lr	Initial learning rate	0.01	0.01	0.01
act	Activation function (output layer)	ReLU	ReLU	Sigmoid

Table 4.4: Hyperparametric configurations (hp) selected from the Bayesian Optimization process for the LSTM network, for time horizons $h = \{1, 5, 10\}$.

We obtain for $h = 1$ and $h = 10$ an LSTM network with only 1 hidden layer, while the network to predict 5-days-ahead contains 4 LSTM layers. Regarding the number of units per layer or the preferred activation function, the results do not seem to show any significant common pattern. The number of internal parameters of the network varies from 35 for the case $h = 1$, 1459 adjustable parameters for $h = 10$ and up to 1955 for the case $h = 5$, being the model among the 3 with the greatest complexity.

We can glimpse two observations from these results. First, the optimal dropout rate and recurrent dropout rate are equal to 0 in all LSTM layers of the best selected configurations. Our experimental problem does not benefit from this regularization measure, which indicates that preventing overfitting does not compensate the downside of losing a fraction of the information (the dropped inputs). Second, we see that the preferred initial learning rate is 0.01, presumably due to the algorithm's ability to dynamically reduce the learning rate during training if learning stalls.

In the following tables, we show the prediction results of the CISS indices on the test sample, broken down by region and compared with the ARMA-GARCH model and the Benchmark model (the Naive model). Two

error metrics are displayed, the MSE and the MAE (see equation 2.20).

Table 4.5 shows the forecast metrics for $h = 1$ ($\hat{x}_{t+1|t}$). In total terms, the metrics for the LSTM model and for the ARMA-GARCH models are lower than the metrics for the Naive model (the default Benchmark model), which means that the prediction error is lower. Both models add value in forecasting the CISS series. Regarding the MSE metric, we find that the predictions of the ARMA-GARCH models are better (in total terms) than the LSTM model, while this relationship is reversed for the MAE. Note that for the MSE metric, the error is given in squared terms (also the metric units are squared). It penalizes more harshly large errors (and outliers) than small errors, whereas the MAE gives all errors equal weight (more interpretable, metric units match CISS units). Thus, due to the difference in the units, one must be careful when comparing both metrics in terms of magnitudes as it could lead to misleading conclusions. As for the regions, Portugal is the one with the smallest error above all, obtaining a MSE $= 3.04 \cdot 10^{-5}$ with the LSTM model, the lowest in the table, while scoring MSE $= 3.34 \cdot 10^{-5}$ with the ARMA-GARCH model. The worst forecast for the LSTM model, in absolute terms, is for the Euro Area CISS, with MSE $= 1.30 \cdot 10^{-4}$ and MAE $= 6.58 \cdot 10^{-3}$.

	ARMA-GARCH Model		**Benchmark Model**		**LSTM Model**	
Country	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
Germany	8.70E-05	4.81E-03	1.05E-04	5.31E-03	9.28E-05	4.95E-03
France	8.68E-05	5.17E-03	9.87E-05	5.50E-03	9.12E-05	5.28E-03
Italy	8.87E-05	4.90E-03	1.09E-04	5.50E-03	8.89E-05	4.74E-03
Portugal	3.34E-05	3.66E-03	3.67E-05	3.93E-03	3.04E-05	3.42E-03
UK	1.24E-04	6.49E-03	1.49E-04	7.20E-03	1.23E-04	6.37E-03
Spain	4.60E-05	3.65E-03	5.44E-05	3.99E-03	4.65E-05	3.52E-03
Euro Area	1.24E-04	6.59E-03	1.43E-04	7.17E-03	1.30E-04	6.58E-03
US	7.08E-05	5.01E-03	7.20E-05	5.21E-03	7.55E-05	5.04E-03
China	5.17E-05	4.70E-03	6.07E-05	4.99E-03	5.40E-05	4.73E-03
Total	**7.91E-05**	**5.00E-03**	**9.21E-05**	**5.42E-03**	**8.15E-05**	**4.96E-03**

Table 4.5: MSE and MAE metrics for the forecasts of the CISS series on the test sample. Case $h = 1$.

To have a better comparative perspective of the performances of the models, we define the % gain of one model over another model as the relative change $\%c$ in its metric (% Gain m_1 to m_2 is $\%c = 100 \cdot (m_2 - m_1)/m_2$). The computed relative gains are depicted in Table 4.6 for $h = 1$:

	% Gain ARMA to Bench.		% Gain LSTM to Bench.		% Gain LSTM to ARMA	
Country	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
Germany	17.31	9.48	11.83	6.89	-6.63	-2.86
France	12.04	5.89	7.58	3.89	-5.07	-2.12
Italy	18.37	10.92	18.14	13.84	-0.28	3.28
Portugal	8.92	6.71	16.98	12.97	8.85	6.71
UK	17.02	9.96	17.45	11.60	0.52	1.82
Spain	15.34	8.64	14.41	11.69	-1.10	3.34
Euro Area	13.55	8.09	8.74	8.28	-5.57	0.21
US	1.73	3.75	-4.85	3.21	-6.69	-0.56
China	14.80	5.87	11.04	5.36	-4.42	-0.54
Total	**14.08**	**7.83**	**11.52**	**8.57**	**-2.98**	**0.80**

Table 4.6: Relative gains of the MSE and MAE metrics between the different fitted models for the CISS series. Results for the test sample. Case $h = 1$.

As we have seen previously, the LSTM model is the best in total terms according to the MAE metric, while according to the MSE metric the relative gain of the LSTM model against the ARMA model is negative, with a loss of 2.98%. In the comparison LSTM model over Naive model (Benchmark), the most important relative gains according to the MAE are given for Italy and Portugal; 13.84% and 12.97%. According to the MSE, the two largest gains are in Italy, first with the ARMA model (18.37%), followed by the LSTM model (18.14%). For the LSTM model, mostly all relative gains with respect to Benchmark are positive, with an exception to US, which presents a -4.85% loss in the MSE. Similarly, in the ARMA-GARCH models, US is also the country that annotates the worst performance, although it still presents positive gains in both metrics.

For the horizon $h = 1$ we cannot conclude that there is a clear winning model between the LSTM and the ARMA-GARCH. While it is true that the ARMA outperforms the LSTM on some occasions, this also occurs in the opposite direction just as many times. Besides, we can see that the regions with the lowest prediction error (in absolute terms) are not necessarily the ones that present a higher relative gain when compared to the Naive model.

Moving now to the horizon $h = 5$, Table 4.7 shows the results of the predictions for the test sample and Table 4.8 shows their relative gains with respect to the other fitted models.

	ARMA-GARCH Model		**Benchmark Model**		**LSTM Model**	
Country	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
Germany	1.08E-03	1.68E-02	1.21E-03	1.80E-02	1.16E-03	1.78E-02
France	9.48E-04	1.68E-02	1.04E-03	1.74E-02	9.44E-04	1.68E-02
Italy	1.23E-03	1.92E-02	1.38E-03	2.04E-02	1.24E-03	1.89E-02
Portugal	3.09E-04	1.18E-02	3.31E-04	1.26E-02	2.75E-04	1.16E-02
UK	1.55E-03	2.48E-02	1.73E-03	2.65E-02	1.54E-03	2.49E-02
Spain	5.57E-04	1.30E-02	5.94E-04	1.34E-02	5.83E-04	1.33E-02
Euro Area	1.34E-03	2.21E-02	1.47E-03	2.31E-02	1.47E-03	2.27E-02
US	4.68E-04	1.40E-02	4.75E-04	1.45E-02	4.85E-04	1.46E-02
China	5.80E-04	1.50E-02	6.21E-04	1.53E-02	6.13E-04	1.53E-02
Total	**8.96E-04**	**1.71E-02**	**9.82E-04**	**1.79E-02**	**9.23E-04**	**1.73E-02**

Table 4.7: MSE and MAE metrics for the forecasts of the CISS series on the test sample. Case $h = 5$.

	% Gain ARMA to Bench.		**% Gain LSTM to Bench.**		**% Gain LSTM to ARMA**	
Country	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
Germany	10.08	6.52	3.85	1.07	-6.92	-5.83
France	9.20	3.60	9.61	3.79	0.46	0.20
Italy	10.71	5.89	9.82	7.41	-1.00	1.62
Portugal	6.57	5.71	16.94	7.89	11.10	2.31
UK	10.46	6.49	10.84	6.14	0.42	-0.38
Spain	6.10	3.51	1.82	0.79	-4.56	-2.82
Euro Area	8.45	4.41	-0.04	1.91	-9.27	-2.62
US	1.38	3.39	-2.19	-0.74	-3.61	-4.27
China	6.54	1.90	1.24	-0.39	-5.68	-2.34
Total	**8.76**	**4.78**	**6.03**	**3.33**	**-3.00**	**-1.53**

Table 4.8: Relative gains of the MSE and MAE metrics between the different fitted models for the CISS series. Results for the test sample. Case $h = 5$.

First, we observe a decline in the predictive capacity of the three approaches used. This is to be expected, since the forecast is further forward in time and therefore the range of variation of the stress index is potentially greater. This time, in total terms, the ARMA-GARCH models present better metrics than the rest. Although the LSTM network is still more predictive than the Naive model, it is outperformed by classical autoregressive models. In relative terms, the LSTM network loses 3.00% in the MSE metric and 1.53% in the MAE compared to the ARMA. Even so, for some regions, such as Portugal or France, favourable results are still obtained for the neural network. While the ARMA-GARCH models keep their relative gain (with respect to the Benchmark) positive for all regions, we behold negative values for the Euro Area, US and China on the LSTM model (with respect to the Benchmark).

Finally, we analyse the forecasts obtained for two weeks ahead ($h = 10$). The metrics are shown in Table 4.9, and in Table 4.10 appear the corresponding relative gains between each of the fitted models.

	ARMA-GARCH Model		**Benchmark Model**		**LSTM Model**	
Country	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
Germany	2.80E-03	2.78E-02	2.84E-03	2.85E-02	3.02E-03	3.17E-02
France	2.52E-03	2.73E-02	2.55E-03	2.75E-02	2.51E-03	2.90E-02
Italy	3.37E-03	3.23E-02	3.44E-03	3.29E-02	3.79E-03	3.56E-02
Portugal	7.61E-04	1.94E-02	7.61E-04	1.95E-02	7.24E-04	2.10E-02
UK	4.21E-03	4.16E-02	4.34E-03	4.26E-02	4.09E-03	4.07E-02
Spain	1.27E-03	2.06E-02	1.26E-03	2.05E-02	1.31E-03	2.39E-02
Euro Area	3.45E-03	3.61E-02	3.37E-03	3.61E-02	3.86E-03	3.81E-02
US	1.07E-03	2.20E-02	1.05E-03	2.21E-02	1.08E-03	2.36E-02
China	1.31E-03	2.29E-02	1.29E-03	2.28E-02	1.37E-03	2.46E-02
Total	**2.31E-03**	**2.78E-02**	**2.32E-03**	**2.80E-02**	**2.42E-03**	**2.98E-02**

Table 4.9: MSE and MAE metrics for the forecasts of the CISS series on the test sample. Case $h = 10$.

	% Gain ARMA to Bench.		**% Gain LSTM to Bench.**		**% Gain LSTM to ARMA**	
Country	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
Germany	1.20	2.48	-6.53	-11.23	-7.82	-14.05
France	1.23	0.52	1.47	-5.69	0.24	-6.24
Italy	2.19	1.59	-10.02	-8.40	-12.48	-10.15
Portugal	0.04	0.65	4.92	-7.70	4.87	-8.40
UK	3.18	2.36	5.84	4.56	2.74	2.26
Spain	-1.20	-0.37	-3.82	-16.73	-2.59	-16.29
Euro Area	-2.43	0.06	-14.54	-5.41	-11.82	-5.47
US	-1.88	0.35	-2.66	-6.72	-0.76	-7.10
China	-1.79	-0.47	-5.74	-8.16	-3.87	-7.66
Total	**0.67**	**0.96**	**-4.03**	**-6.26**	**-4.73**	**-7.29**

Table 4.10: Relative gains of the MSE and MAE metrics between the different fitted models for the CISS series. Results for the test sample. Case $h = 10$.

For the case $h = 10$, the LSTM network does not show any improvement with respect to the Naive approach (Benchmark), but rather the results worsen in relative terms 4.03% according to the MSE and 6.26% according to the MAE. The neural network is unable to extract generalizable information from the training and, when evaluating on an unseen sample, it is not capable of adding value to the forecast. Only for one country, UK, it manages to obtain both metrics with positive relative gains with respect to the Naive and ARMA-GARCH models. For the ARMA-GARCH models, the forecasts slightly improve, achieving in total terms better metrics when compared to the Benchmark, although the gains are close to zero; 0.67% according to the MSE and 0.96% according to the MAE. As we have seen in section 4.2.1, the ARMA-GARCH models fitted have the advantage that the forecast (on the differenced series) tends to zero as we extend the horizon, so the forecast in the original series resembles the predictions of the Naive approach when h increases, with the characteristic that the forecast cannot get much worse compared to the Benchmark no matter how much we continue to extend the horizon. On the other hand, for the LSTM model, this kind of boundary does not

exist, which can cause results with more variability that further deviate from the Naive approach. In our initial trials, we even attempted to set larger horizon frames, up to $h = 60$, but we quickly realize that we were not able to outperform the Naive approach.

From the fundamental point of view of the CISS series, it can be argued that the time structure of the series is very non-uniform and hardly obeys patterns or empirical regularities for sample periods that are far from each other. Therefore, its structure from the training sample must have mutated several times compared to the test sample. For example, in the LSTM model, if we look at the MSE at $h = 1$ for the validation sample ($\text{MSE} = 6.31 \cdot 10^{-5}$) we can see that its value is significantly lower than the $\text{MSE} = 8.15 \cdot 10^{-5}$ later obtained in the test sample.

Although it is true that at short horizons we observe some autocorrelation structure with its past values, the most distant lags (on the differenced series) could well be modelled by a white noise process. Even for non-linear models, as the LSTM, we do not manage to extract valuable information to predict long horizons. The CISS series are fundamentally driven by financial and macroeconomic changes and relevant market events, and so own lagged terms of the series do not seem to have good explanatory power in the long-range view.

In addition to the metrics presented, one can also come upon with relevant insights by plotting the predictions obtained and comparing them between the different time horizons considered. We do so for the UK CISS, in Figures 4.7, 4.8 and 4.9. On the left, we depict the forecasted values in the test sample, including the actual observed CISS. On the right, a scatter plot of the same forecast versus the actual values gives us an idea of the dispersion of our predictions, where the diagonal line $(y = x)$ represents the perfect case. From top to bottom, the time horizons $h = \{1, 5, 10\}$ are shown, respectively.

As the horizon increases, the dispersion becomes more noticeable. For $h = 1$, the predictions appear accurate to the naked eye and the dispersion plot looks quite uniform along the diagonal axis line, while being most of the observations located in values of low stress regime. For $h = \{5, 10\}$, dispersion magnifies, being the greater the actual observed value, the greater the error. This is expected, as when financial stress is higher, uncertainty also increases, resulting in more variability on the time series and thus more difficulty in giving accurate predictions. Besides, we observe that the dispersion plot is skewed towards the upper diagonal. That is, it is more common to see the forecasted value to be smaller than the actual value (more source of error in average) than it is to forecast a value that is higher than the observed. This undershooting is clearer to see in the $h = 10$ horizon; it happens mostly in cases where the financial risk suddenly increases and models are not able to level up until some periods ahead when high values of the index begin to enter as inputs. Last, comparing the models adjusted, and in concordance with the metrics presented before, we see that for the UK CISS the LSTM model is the one performing better. This is qualitatively reflected in the scatter plots, where the LSTM predictions are generally more centred around the diagonal line, while the Naive approach is the one that remains in the outermost parts of the graph. This last observation, is of course subject to change with the other studied regions, depending in each case which was the model we found out to have a better fit. But, for the rest of the analysis, we have seen that the dispersion phenomena are common in a qualitative way to all the other studied regions, so we avoid overextending our work by just presenting this case as an example.

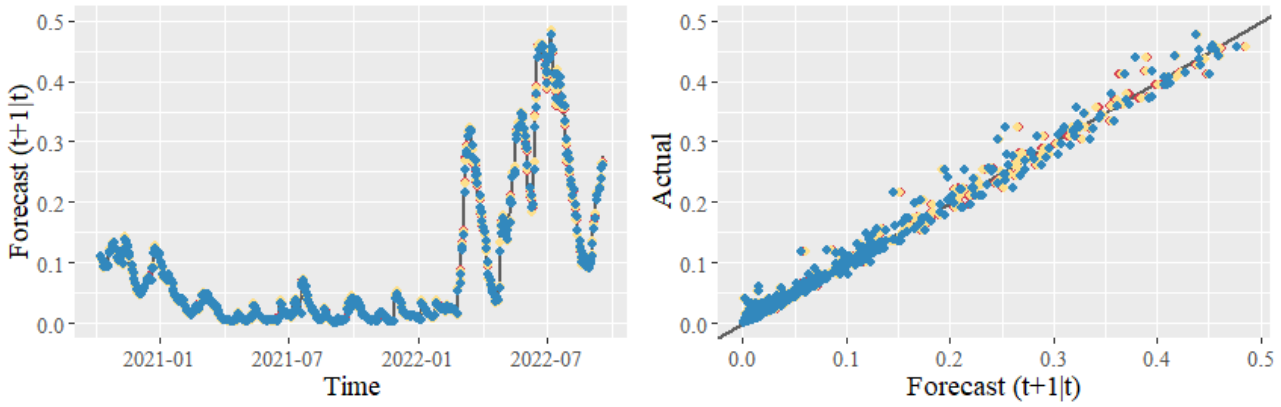

Figure 4.7: Predicted values by the 3 fitted models: LSTM network (**red**), ARMA-GARCH model (**yellow**), and Naive approach (**blue**). Predictions for $h = 1$.

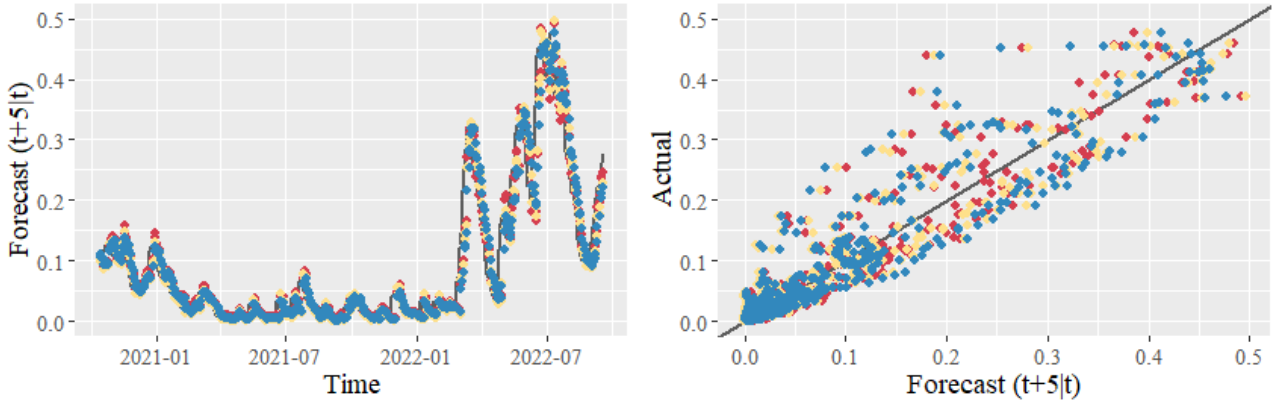

Figure 4.8: Predicted values by the 3 fitted models: LSTM network (**red**), ARMA-GARCH model (**yellow**), and Naive approach (**blue**). Predictions for $h = 5$.

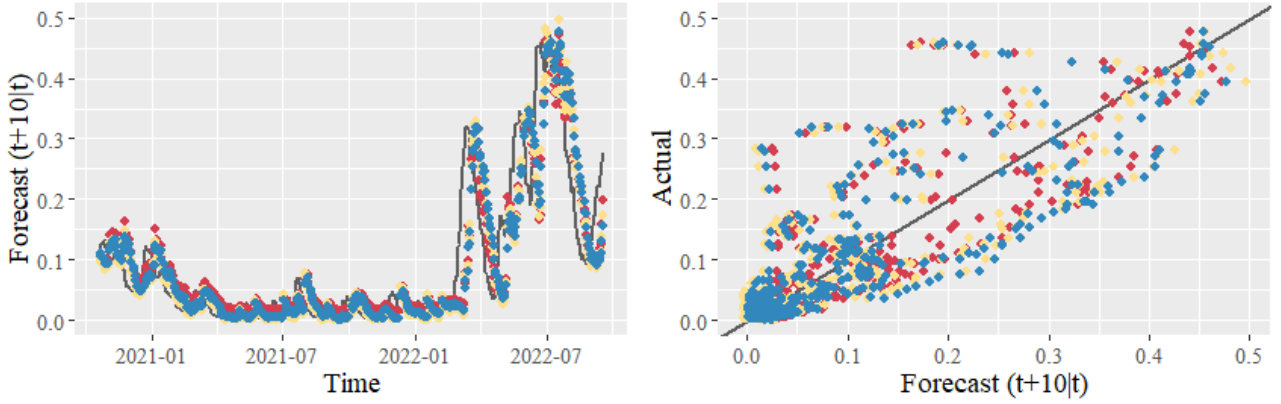

Figure 4.9: Predicted values by the 3 fitted models: LSTM network (**red**), ARMA-GARCH model (**yellow**), and Naive approach (**blue**). Predictions for $h = 10$.

In summary, under the presented experimental context, we believe that LSTM networks can provide with

satisfactory results on the short horizon. However, due to the circumstances (small sample data, composite financial time series, little serial dependency, structural changes, etc), we consider classical econometric approaches such as ARMA-GARCH models to be more robust and to return with better results in this scenario, plus they provide simplicity in terms of parameters to estimate, computational time and interpretability of the model.

Chapter 5

Conclusions

Throughout this project we have been working around financial stress indices, in particular, with the Composite Index of Systemic Stress. We have applied various methodologies to these indicators to extract valuable insights about the behaviour of systemic stress on different selected regions. On this section, we highlight the major conclusions reached and propose a discussion on future lines of work within this research area.

In relation to the analysis of connectedness on the systemic risk between the regions studied, we resolved the main goals that were initially proposed. Within the Euro Area, we encounter with a highly interconnected network, with a measure of total connectedness of $C = 62.89\%$. Despite being the own variance share terms the bigger individual elements among the variance decomposition matrix, the aggregated connectedness from others exceeds the 50% level in all countries. Interestingly, from the unconditional point of view, the two major economic powers of the EU, that is, France and Germany, exhibit the greatest levels of connectedness, and they are the countries with highest capacity of transmitting systemic risk to others. Also, neighbouring countries, such as France and Spain or Portugal and Spain, present strong linkages in terms of pairwise connectedness. Overall, Portugal and specially UK, are the two countries less interrelated in the network. UK efforts to disassociate from the European Union are also seen reflected in the financial sector, although this does not prevent him from being the greatest net receiver of risk among all, with $C_{UK} = -10.59\%$. Besides, we explored the dynamic behaviour of the connectedness measures, from the macro to the micro perspective. In general, it is seen that systemic risk interdependencies between countries usually peak or exhibit highest variability when financial stress increases due to an important market event. We localized the greatest levels of total connectedness to be coinciding with the beginning of the lockdown measures on the Covid pandemic, sustaining values around the $C = 75\%$ for an entire year. Other periods of high connectedness were found during the European sovereign debt crisis and during the years 2015-2016, linked to the stock market selloff and to the uncertainty around Brexit.

With regards to the connectedness study between China, United States, and the Euro Area, we have also found interesting results. The total connectedness measure stands at $C = 17.19\%$, being lower compared to the previous case studied. Even though, pairwise connectedness measures between Euro Area and United States show strong linkage, as high as the top countries in Europe. In contrast, China is the most isolated country in terms of financial stress, showing connectedness measures significantly lower in magnitude. In

net terms, the Euro Area and United States are transmitters of risk, impacting China with a net total term of $C_{China} = -7.64\%$. From the dynamic point of view, we observed similar qualitative results as in the within Europe countries case, being the greatest shocks of financial stress transmitted during the Covid period.

Our results obtained in this project are in line with other related papers. In [35], interdependences on the CISS indices are analysed for 11 countries from the European Area, by using two different connectedness frameworks: the generalized connectedness approach as presented here [5], and the frequency connectedness measures as proposed in [36]. In this paper, the author limits himself to showing only the overall (total) connectedness measure in the dynamic rolling window case. We extended the analysis by showing also pairwise, net pairwise, total directional and net total directional connectedness measures, both in the static and dynamic case. In [37], the authors present dynamic connectedness measures by realizing a TVP-VAR model between the stock markets of different European countries (including US and China), where they put special focus on the Covid period. Our results regarding this period are coincident, which show strong evidence that stock markets behaviour is quite indicative to the overall financial stress, in the same way that it is often used as a barometer for the state of the economy. However, one point in favour towards using financial stress indices rather than stock market data is that the first ones also target other relevant financial segments, such as the bond market or the money market, so a wider view can be obtained if we want to evaluate the whole financial system.

We believe further work on this topic would be meaningful if other global or emerging economic powers (such as Japan, India, or Brazil, among others) were included in the analysis, and we could explore how the directionality of risk is transmitted to those countries and how those interdependences would evolve over time. Besides, as done in related literature, applying other alternative connectedness frameworks that exist and comparing them with the one presented here, would also enrich our analysis and perhaps it would give us with a wider perspective within the research field.

In relation to the forecasts on the CISS indices at different time horizons, we have proposed a univariate LSTM network with a unique structure for all regions as a data augmentation method and with various modules to prevent overfitting the training sample. We have also approached the problem by adjusting ARMA-GARCH models into the differenced time series and we have complemented the work with a Naive approach as a Benchmark model for comparison. Our results outperform the Naive approach in the one-day-ahead and one-week-ahead horizons but fail to consistently deliver a better forecast on the two-weeks-ahead case. The results suggest that the temporal structure of the CISS series is rarely preserved during long periods and its lagged observations have explanatory power only for the short term. Comparing the LSTM model with the ARMA-GARCH framework, we see that the second may be more suitable to use in experimental contexts as the one presented here, as it has proven to provide with generally better results. However, for countries like Portugal and UK, the LSTM network is arguably the preferred choice in all time horizons considered, as it outperforms the other approaches in the majority of the metrics presented. Perhaps, in terms of forecasting, the best choice would be not to choose any model in particular, but to have ARMA-GARCH models and LSTM networks competing in parallel, and ultimately one can decide at each moment what model to use for each country based on the recent predictive performance of the two.

Also, we presented a comparative analysis of the dispersion of the error term for the different horizons considered. The results suggest that for one-week-ahead and two-weeks-ahead horizons, the three approaches tend to undershoot the forecast, as only having as inputs its own lagged observations, the models fail to guess an upcoming raise in risk levels.

To our knowledge, few literature exists related to forecasting financial stress indices with neural networks. Besides, it is not a common practice to use Benchmark models to have a real comparative outlook on whether the new models proposed suppose any improvement with respect to more classical and simpler approaches. By searching and comparing promising LSTM approaches within the financial sector, we find most of them to be designed to predict stock prices. What characterizes those is that: usually the prediction horizons are short (one-day-ahead [15]), or that there are more features available such as candlestick type data [38] (OHLCV: open price, high price, low price, adjusted close price and volume), or that there is a much bigger database available, obtained by extracting high-frequency intraday data [39]. Therefore, we believe that in our project it has been difficult to outperform classical approaches due to the fact that our experimental setup does not contain these type of characteristics.

Future lines of research regarding financial time series forecasting, specially related to stress indices, would follow on proposing hybrid models which combine the LSTM network with autoregressive ARMA-GARCH models. Since the error is seen stationary and centred around zero, one possibility to examine would be to model the CISS series first by using a neural network, and then trying to model its error by an ARMA-GARCH model (or vice-versa, first ARMA-GARCH, then LSTM network). This approach is similar in some sense to hybrid models for volatility estimates that combine the GARCH framework with deep learning methods; see, for instance, [40–42]. Another alternative would be to ideate a data augmentation method (similar to the one proposed in [43]) which could combine information from various CISS series and also from other macroeconomic variables of interest, perhaps in a structured way making use of the results obtained from an analysis of connectedness between those series. All in all, the predictive performance of neural networks in financial time series modelling will mostly depend on whether we are capable of having a large enough data set, with information that has good explanatory power over the series of study.

As end point, I would like to emphasize on the importance of financial stress indices. Even though individual risk indicators such as volatility series or interest rate spreads are still the most used indicators in the industry to monitor financial risk, I believe composite systemic stress indices, such as the CISS series presented here, provide a much more direct, robust, and interpretable vision of the overall levels of risk in the financial system. Thus, major financial corporations, policy makers and central banks, but also within the scientific research sector, should pay close attention to the field of financial stress indices if they want to modernize risk management, have a better understanding of financial risk, and prevent on future economic crisis.

Acknowledgements

This project was possible thanks to my supervisors Dr. Helena Chuliá Soler and Dr. Salvador Torra Porras. Acknowledgements to them for helping me throughout the project. Special thanks to my parents, for unconditional support and for encouraging me to do my best. Last, I would like to thank my colleagues from the UPC Faculty of Mathematics and Statistics, for making the master's degree an enjoyable and amazing experience. I believe this project reflects part of what I have learned during these years at the university, and I hope to make the people who know me best feel proud about the work done here.

References

[1] International Monetary Fund, Bank for International Settlements and Financial Stability Board. ”Guidance to Assess the Systemic Importance of Financial Institutions, Markets and Instruments: Initial Considerations”. Report to G20 Finance Ministers and Governors (2009).

[2] D. Holló, M. Kremer, and M. Lo Duca. ”CISS – A Composite Indicator of Systemic Stress in the Financial System”. ECB Working Paper No. 1426 (2012).

[3] F.X. Diebold, and K. Yilmaz. ”Better to give than to receive: Predictive directional measurement of volatility spillovers”. International Journal of Forecasting, vol. 28, No. 1, pages 57-66 (2012).

[4] F.X. Diebold, and K. Yilmaz. ”Measuring Financial Asset Return and Volatility Spillovers, with Application to Global Equity Markets”. The Economic Journal, vol. 119, No. 534, pages 158-171 (2009).

[5] F.X. Diebold, and K. Yilmaz. ”On the Network Topology of Variance Decompositions: Measuring the Connectedness of Financial Firms”. Journal of Econometrics, vol. 182, pages 119-134 (2014).

[6] M.H. Pesaran, and Y. Shin. ”Generalized impulse response analysis in linear multivariate models”. Economics Letters, vol. 58, pages 17-29 (1998).

[7] G. Koop, M.H. Pesaran, and S.M. Potter. ”Impulse response analysis in nonlinear multivariate models”. Journal of Econometrics, vol. 74, No. 1, pages 119-147 (1996).

[8] J. Schmidhuber. ”Deep learning in neural networks: An overview”. Neural Networks, vol. 61, pages 85-117 (2015).

[9] O.B. Sezer, M.U. Gudelek, and A.M. Ozbayoglu. ”Financial time series forecasting with deep learning: A systematic literature review: 2005–2019”. Applied Soft Computing, vol. 90, pages 106-181 (2020).

[10] Y. LeCun, Y. Bengio, and G. Hinton. ”Deep learning”. Nature, vol. 521, pages 436-444 (2015).

[11] J.T. Connor, R.D. Martin, and L.E. Atlas. ”Recurrent Neural Networks and Robust Time Series Prediction”. IEEE Transactions on Neural Networks, vol. 5, No. 2, (1994).

[12] Y. Bengio, P. Simard, and P. Frasconi. ”Learning long-term dependencies with gradient descent is difficult”. IEEE Transactions on Neural Networks, vol. 5, No. 2, pages 157-166 (1994).

[13] S. Hochreiter, and J. Schmidhuber. ”Long Short-Term Memory”. Neural Computation, vol. 9, No. 8, pages 1735–1780 (1997).

[14] F.A. Gers, J. Schmidhuber, and F. Cummins. "Learning to forget: continual prediction with LSTM". Neural Computation, vol. 12, No. 10, pages 2451–2471 (2000).

[15] T. Fischer, and C. Krauss. "Deep learning with long short-term memory networks for financial market predictions". European Journal of Operational Research, vol. 270, No. 2, pages 654-669 (2018).

[16] S. Ruder. "An overview of gradient descent optimization algorithms". arXiv:1609.04747 (2017).

[17] R.J. Williams, and J. Peng. "An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories". Neural Computation, vol. 2, No. 4, pages 490–501 (1990).

[18] D.P. Kingma, and J. Lei Ba. "ADAM: A Method for Stochastic Optimization". Conference paper at ICLR 2015, arXiv:1412.6980v9 (2017).

[19] CY. Park, and R.V. Mercado Jr. "Determinants of financial stress in emerging market economies". Journal of Banking and Finance, vol. 45, pages 199-224 (2014).

[20] S. Corbet. "The European Financial Market Stress Index". International Journal of Economics and Financial Issues vol. 4, No. 1, pages 217-230 (2014).

[21] M. Kremer. "Financial Stress Indices: An Introduction". The Spanish Review of Financial Economics, vol. 14, No. 1, pages 1–4 (2016).

[22] L. Estévez Cerqueira, and M.I. Cambón Murcia. "A Spanish Financial Market Stress Index (FMSI)". Comisión Nacional del Mercado de Valores, Working paper No. 60 (2015).

[23] E.J. Hannan, and B.G. Quinn. "The Determination of the Order of an Autoregression". Journal of the Royal Statistical Society: Series B (Methodological), vol. 41, No. 2, pages 190-196 (1979).

[24] F.X. Diebold, and K. Yilmaz. "Trans-Atlantic Equity Volatility Connectedness: U.S. and European Financial Institutions, 2004–2014". Journal of Financial Econometrics, vol. 14, No. 1, pages 81–127 (2015).

[25] E. Uluceviz, and K. Yilmaz. "Real-financial connectedness in the Swiss economy". Swiss Journal of Economics and Statistics, vol. 156, No. 1 (2020).

[26] R.S. Tsay. "Linear Time Series Analysis and Its Applications". In *Analysis of Financial Time Series*. John Wiley and Sons, 3rd Edition, pages 64-71 (2010).

[27] J.A. Cadzow. "ARMA Time Series Modeling: an Effective Method". IEEE Transactions on Aerospace and Electronic Systems, vol. AES-19, No. 1, pages 49-58 (1983).

[28] E. Ruiz Ortega. "Modelos para series temporales heterocedásticas". Cuadernos Económicos del ICE, vol. 56, pages 73-108 (1994).

[29] T. Bollerslev. "Generalized autoregressive conditional heteroskedasticity". Journal of Econometrics, vol. 31, No. 3, pages 307-327 (1986).

[30] L.R. Glosten, R. Jagannathan, and D.E. Runkle. ”On the Relation between the Expected Value and the Volatility of the Nominal Excess Return on Stocks”. The Journal of Finance, vol. 48, No. 5, pages 1779-1801 (1993).

[31] N. Hamzaoui, and B. Regaieg. ”The Glosten-Jagannathan-Runkle-Generalized Autoregressive Conditional Heteroscedastic approach to investigating the foreign exchange forward premium volatility”. International Journal of Economics and Financial Issues, vol. 6, No. 4, pages 1608-1615 (2016).

[32] D.B. Nelson. ”Conditional Heteroskedasticity in Asset Returns: A New Approach”. Econometrica, vol. 59, No. 2, pages 347-370 (1991).

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. ”Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. The Journal of Machine Learning Research, vol. 15, No. 1, pages 1929–1958 (2014).

[34] E. Brochu, V.M. Cora, and N. de Freitas. ”A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning”. arXiv:1012.2599 (2010).

[35] O. Polat. ”On systemic risk contagion in the euro area: Evidence from frequency connectedness and the DY approaches”. Borsa Istanbul Review, vol. 22, No. 3, pages 441-451 (2022).

[36] J. Baruník, and T. Křehlík. ”Measuring the Frequency Dynamics of Financial Connectedness and Systemic Risk”. The Journal of Financial Econometrics, vol. 16, No. 2, pages 271-296 (2018).

[37] M. Youssef, K. Mokni, and A.N. Ajmi. ”Dynamic connectedness between stock markets in the presence of the COVID-19 pandemic: does economic policy uncertainty matter?”. Financial Innovation, vol. 7, No. 13 (2021).

[38] S. Il Lee, and S. Joon Yoo. ”Threshold-based portfolio: the role of the threshold and its applications”. The Journal of Supercomputing, vol. 76, No. 3 (2020).

[39] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis. ”Using deep learning to detect price change indications in financial markets”. 25th European Signal Processing Conference EUSIPCO (2017).

[40] E. Hajizadeh, A. Seifi, M.H. Fazel Zarandi, and I.B. Turksen. ”A hybrid modeling approach for forecasting the volatility of S&P 500 index return”. Expert Systems with Applications, vol. 39, No. 1, pages 431-436 (2012).

[41] X. Lua, D. Quea, and G. Caoa. ”Volatility Forecast Based on the Hybrid Artificial Neural Network and GARCH-type Models”. Procedia Computer Science, vol. 91, pages 1044–1049 (2016).

[42] L. Di Persio, M. Garbelli, and K. Wallbaum. ”Forward-Looking Volatility Estimation for Risk-Managed Investment Strategies during the COVID-19 Crisis”. Risks, vol. 9, No. 2, pages 1-16 (2021).

[43] Y. Baek, and H.Y. Kim. ”ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module”. Expert Systems with Applications, vol. 113, pages 457-480 (2018).

Appendix A

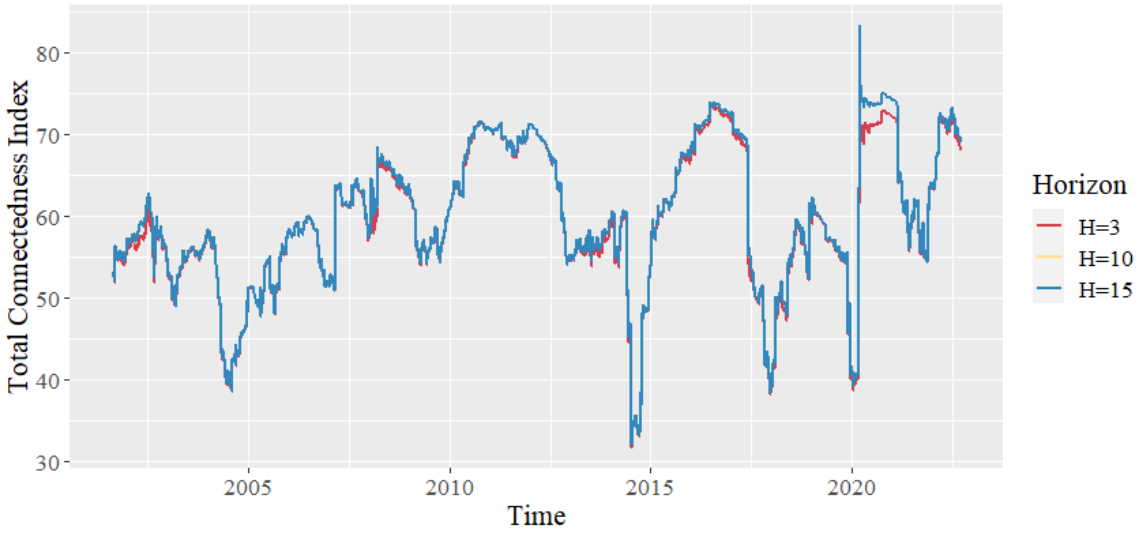

Figure 5.1: Sensitivity analysis of the Total Connectedness Index for different horizons H. European region countries.

Figure 5.2: Sensitivity analysis of the Total Connectedness Index for different window widths w. European region countries.

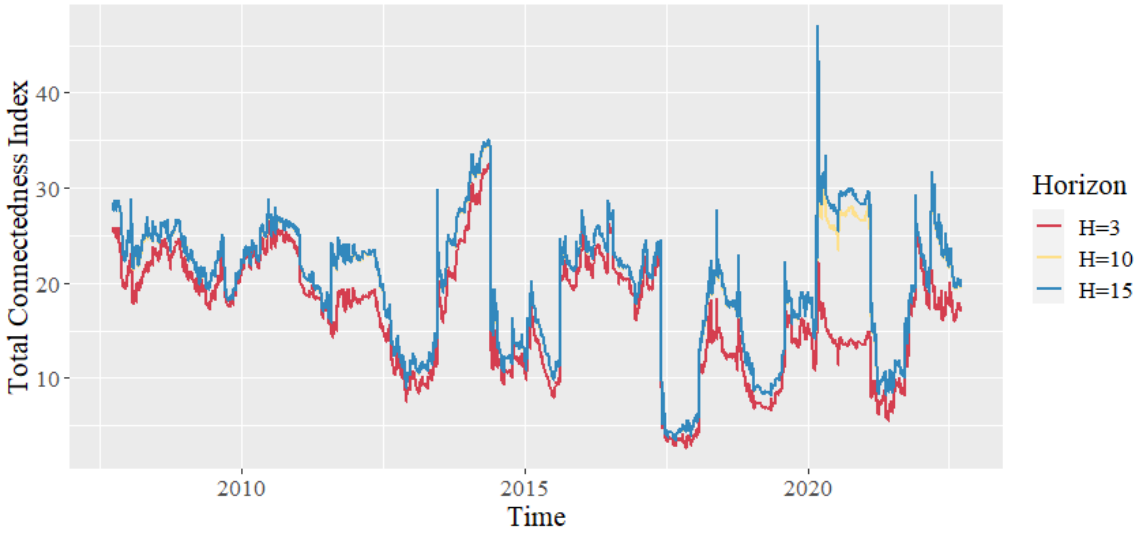

Figure 5.3: Sensitivity analysis of the Total Connectedness Index for different horizons H. World area.

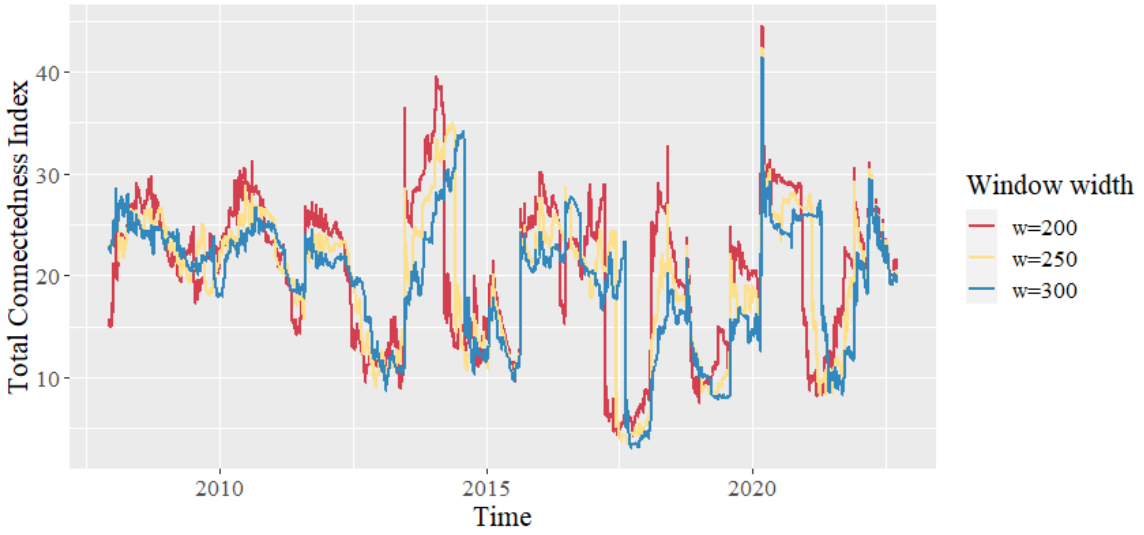

Figure 5.4: Sensitivity analysis of the Total Connectedness Index for different window widths w. World area.

Appendix B

Code details

On this project, we have used two different programming languages: **R** and **Python**.

- On **R**, via RStudio, we have carried out the exploratory and descriptive analysis of the CISS series. Also, the identification, fitting, validation and forecasting of the ARMA-GARCH models (and the Naive approach). Besides, we have carried out the Connectedness analysis. All the plots presented on this project have been made using the *ggplot2* package.
- On **Python**, via Google Colab, we have designed and deployed the part regarding the LSTM network and its forecasting. Precisely, for training the model and specifying the network architecture, we have used the Keras framework on top of TensorFlow 2.

The full extend of the code built to make this project can be found in the following GitHub repository:

`https://github.com/oriol33/TFM_OriolGairin.git`

In the following pages, we provide some snippets of the code that may be of interest. Note that they just represent a short brief, they are not complete and do not cover the extend of our work.

Germany CISS

Identification, fitting and validation of an ARMA-GARCH model.

```

#Validation function (model for the mean)
############################################################
validation=function(model,dades){
  s=frequency(get(model$series))
  resid=model$residuals
  par(mfrow=c(2,2),mar=c(3,3,3,3))
  #Residuals plot
  plot(resid,main="Residuals")
  abline(h=0)
  abline(h=c(-3*sd(resid),3*sd(resid)),lty=3,col=4)
  #Square Root of absolute values of residuals (Homocedasticity)
  scatter.smooth(sqrt(abs(resid)),main="Square Root of Absolute residuals",
                 lpars=list(col=2))

  #Normal plot of residuals
  qqnorm(resid)
  qqline(resid,col=2,lwd=2)

  ##Histogram of residuals with normal curve
  hist(resid,breaks=20,freq=FALSE)
  curve(dnorm(x,mean=mean(resid),sd=sd(resid)),col=2,add=T)


  #ACF & PACF of residuals
  par(mfrow=c(1,2))
  acf(resid,ylim=c(-1,1),lag.max=60,col=c(2,rep(1,s-1)),lwd=1)
  pacf(resid,ylim=c(-1,1),lag.max=60,col=c(rep(1,s-1),2),lwd=1)
  par(mfrow=c(1,1))

  #ACF & PACF of square residuals
  par(mfrow=c(1,2))
  acf(resid^2,ylim=c(-1,1),lag.max=60,col=c(2,rep(1,s-1)),lwd=1)
  pacf(resid^2,ylim=c(-1,1),lag.max=60,col=c(rep(1,s-1),2),lwd=1)
  par(mfrow=c(1,1))

  #Ljung-Box p-values
  par(mar=c(2,2,1,1))
  tsdiag(model,gof.lag=7*s)
  cat("\n----------------------------------------------------------------------\n")
  print(model)

  #Stationary and Invertible
  cat("\nModul of AR Characteristic polynomial Roots: ",
      Mod(polyroot(c(1,-model$model$phi))),"\n")
```

```
  cat("\nModul of MA Characteristic polynomial Roots: ",
      Mod(polyroot(c(1,model$model$theta))),"\n")

  #Model expressed as an MA infinity (psi-weights)
  psis=ARMAtoMA(ar=model$model$phi,ma=model$model$theta,lag.max=36)
  names(psis)=paste("psi",1:36)
  cat("\nPsi-weights (MA(inf))\n")
  cat("\n--------------------\n")
  print(psis[1:20])

  #Model expressed as an AR infinity (pi-weights)
  pis=-ARMAtoMA(ar=-model$model$theta,ma=-model$model$phi,lag.max=36)
  names(pis)=paste("pi",1:36)
  cat("\nPi-weights (AR(inf))\n")
  cat("\n--------------------\n")
  print(pis[1:20])

  #Sample ACF vs. Teoric ACF
  par(mfrow=c(2,2),mar=c(3,3,3,3))
  acf(dades, ylim=c(-1,1) ,lag.max=36,main="Sample ACF")

  plot(ARMAacf(model$model$phi,model$model$theta,lag.max=36),ylim=c(-1,1),
       type="h",xlab="Lag",  ylab="", main="ACF Teoric")
  abline(h=0)

  #Sample PACF vs. Teoric PACF
  pacf(dades, ylim=c(-1,1) ,lag.max=36,main="Sample PACF")

  plot(ARMAacf(model$model$phi,model$model$theta,lag.max=36, pacf=T),ylim=c(-1,1),
       type="h", xlab="Lag", ylab="", main="PACF Teoric")
  abline(h=0)
  par(mfrow=c(1,1))
}
##############################################################################

# Work with differencied series
plot(germany_ts)
d1serie <- diff(germany_ts)
d1serie <- d1serie[-1]
plot(d1serie)

# Check overdifferentation
var(germany_ts)
var(d1serie)

# Check stationarity  (#H0: serie is I(1))
d1serie.df<-ur.df(d1serie, type = c("none"), lags = 3)
summary(d1serie.df)
plot(d1serie.df)
```

```
95
96 #Identification (model for the mean)
97 win.graph(width=8,height=5)
98 par(mfrow=c(1,2),font=2,font.lab=4,font.axis=2,las=1)
99 acf(d1serie,ylim=c(-0.1,1),main="d1serie")
100 pacf(d1serie,ylim=c(-0.1,1),main="d1serie")
101
102 #Proposed model for the mean: ARMA(5,0)
103 (model = arima(d1serie, order = c(5,0,0),include.mean = TRUE))
104 pnorm(c(abs(model$coef)/sqrt(diag(model$var.coef))), mean=0, sd=1, lower.tail=FALSE)
105 BIC(model)
106
107 #Validation
108 validation(model,d1serie)
109 #Clear signs of volatility
110
111
112 ## Model for the variance ##
113
114 #ARMA-GARCH(1,1) t-Student
115 spec1=ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)), mean.model
    =list(armaOrder=c(4,0)), distribution.model = "std")
116 m1=ugarchfit(spec=spec1,data=d1serie)
117 m1
118 plot(m1)
119 #Observations: Good Validation !
120
121
122 #ARMA-GJR-GARCH t-Student
123 spec2=ugarchspec(variance.model=list(model="gjrGARCH", garchOrder = c(1,1)), mean.model=
    list(armaOrder=c(4,0)), distribution.model = "std")
124 m2=ugarchfit(spec=spec2,data=d1serie)
125 m2
126 #gamma<0 indica que cuando la diferencia es negativa (disminuye el systemic risk),
127 # la volatilidad de la serie ligeramente disminuye, respecto a diferencias positivas.
128 plot(m2)
129 #Observations: Good validation! Captures asymmetry
130
131
132 #ARMA-EGARCH t-Student
133 #alpha1 se corresponde con el gamma de las diapositivas (coef. asimetrico)
134 #Gamma de R se corresponde con el theta de las diapositivas.
135 spec3=ugarchspec(variance.model=list(model="eGARCH", garchOrder = c(1,1)), mean.model=
    list(armaOrder=c(4,0)), distribution.model = "std")
136 m3=ugarchfit(spec=spec3,data=d1serie)
137 m3
138 # alpha1 positivo (coef. asim.) significa que para diferencia negativa (disminuye
    systemic risk),
139 # la volatilidad del dia siguiente disminuye respecto si la diferencia fuera positiva.
```

```
plot(m3)
#Observations: Worst fit.


##############################
###  Final model proposed: ###
##############################

#ARMA(4,0)-gjrGARCH(1,1) t-Student
#Observations: Good fit. Asymmetric behavior. Good Validation !
m2
plot(m2)
```

Germany CISS

Forecasting on the test sample with the ARMA-GARCH and Naive models. Horizon $h = 1$.

```

#Partitions for Train + Test
#Validation is done from econometric p.o.v.
train_pct <- 0.9
test_pct <- 1-train_pct

##### Univariate Forecasts ###################

##### Germany ################################

# We fit the model on differenced data, then we inverse transform.
d1germany_ts <- diff(germany_ts)
d1germany_ts <- d1germany_ts[-1]

out_sample <- round(test_pct*length(d1germany_ts),0) #Observations that we leave outside
    the training set

#ARMA(4,0)-gjrGARCH(1,1) t-StudentFrance
# We fit the model, leaving out the test data
spec2=ugarchspec(variance.model=list(model="gjrGARCH", garchOrder = c(1,1)), mean.model=
    list(armaOrder=c(4,0)), distribution.model = "std")
m_germany=ugarchfit(spec=spec2,data=d1germany_ts, out.sample = out_sample )

#FORECAST y(t+1|t)
ugarchforecast(m_germany, n.ahead=10, n.roll= 0)
forc_germany_1 = ugarchforecast(m_germany, n.ahead=1, n.roll= out_sample)

fpm(forc_germany_1) #Forecast performance measures
sigma(forc_germany_1)  #Forecast standard deviation
fitted(forc_germany_1) #Forecast time series (point prediction)
#plot(forc_germany_1)   #Plot forecast, with confidence intervals

```

```
31
32 ####   FORECAST y(t+1 .... t+10 |t)     #########
33 forc_germany = ugarchforecast(m_germany, n.ahead=10, n.roll= out_sample)
34
35 forc_germany_sigma_TOT <- t(as.data.frame(sigma(forc_germany)))
36 forc_germany_mean_TOT <- t(as.data.frame(fitted(forc_germany)))
37
38 ######  T+1|T  #####
39
40 #Horizon
41 H <- 1
42
43 #Substract H last predictions, as we don't have data to compare.
44 forc_germany_mean <- forc_germany_mean_TOT[1:(dim(forc_germany_mean_TOT)[1]-H),]
45
46 #Select T+H forecast
47 forc_germany_mean <- as.data.frame(forc_germany_mean[,1])
48
49 #First day of forecast
50 F1 <- (round(train_pct*length(germany_ts),0)+H)
51
52 #Select forecast dates
53 dates_f <- dates[F1:length(dates)]
54
55 #Build dataset with dates, actual values and predicted values
56 forc_germany_mean <- cbind(forc_germany_mean,dates_f) #date
57
58 forc_germany_mean <- cbind(forc_germany_mean, d1germany_ts[(F1-1):length(d1germany_ts)])
       #actual_diff
59 forc_germany_mean <- cbind(forc_germany_mean,germany_ts[F1:length(germany_ts)]) #actual
60
61 lagged_actual <- germany_ts[(F1-H):(length(germany_ts)-H)] #lagged H times
62 forc_germany_mean <- cbind(forc_germany_mean,lagged_actual)
63
64 rownames(forc_germany_mean) <- NULL
65 colnames(forc_germany_mean) <- c('pred_diff', 'date', 'actual_diff', 'actual', 'lagged_
       actual')
66
67 #We undo de diff() transformation. We get the prediction for the time series
68 pred <- forc_germany_mean$pred_diff+forc_germany_mean$lagged_actual
69 forc_germany_mean <- cbind(forc_germany_mean, pred)
70
71 #Plot the results
72 plot(forc_germany_mean$actual)
73 lines(forc_germany_mean$pred)
74
75 #Compute forecast metrics
76
77 #MSE
```

```
mse(actual = forc_germany_mean$actual, predicted = forc_germany_mean$pred)   #original series
mse(actual = forc_germany_mean$actual_diff, predicted = forc_germany_mean$pred_diff) # differenced series

#RMSE
rmse(actual = forc_germany_mean$actual, predicted = forc_germany_mean$pred)
rmse(actual = forc_germany_mean$actual_diff, predicted = forc_germany_mean$pred_diff)

#MAE
mae(actual = forc_germany_mean$actual, predicted = forc_germany_mean$pred)
mae(actual = forc_germany_mean$actual_diff, predicted = forc_germany_mean$pred_diff)


####### NAIVE APROXIMATION x_{t-1} = x_t #######

# T+H|T

#Horizon
H <- 1
#Time series
ts_naive <- array(germany_ts)

#First day of forecast
F1 <- (round(train_pct*length(ts_naive),0)+H)

#Select forecast dates
dates_f <- dates[F1:length(dates)]

naive_aprox <- data.frame(matrix(ncol = 3, nrow = length(dates_f)))
colnames(naive_aprox) <- c('dates_f', 'actual', 'lagged_H')

naive_aprox$dates_f <- dates_f
naive_aprox$actual <- ts_naive[F1:length(ts_naive)]
naive_aprox$lagged_H <- ts_naive[(F1-H):(length(ts_naive)-H)]

#Plot the results
plot(naive_aprox$actual)
lines(naive_aprox$lagged_H)

#MSE
mse(actual = naive_aprox$actual, predicted = naive_aprox$lagged_H)

#RMSE
rmse(actual = naive_aprox$actual, predicted = naive_aprox$lagged_H)

#MAE
mae(actual = naive_aprox$actual, predicted = naive_aprox$lagged_H)

```

```

###### RELATIVE GAIN (ARMA-GARCH models vs. Naive approach) #######

#MSE gain
-100*(mse(actual = forc_germany_mean$actual, predicted = forc_germany_mean$pred) -
     mse(actual = naive_aprox$actual, predicted = naive_aprox$lagged_H) )/
    (mse(actual = naive_aprox$actual, predicted = naive_aprox$lagged_H))

#RMSE gain
-100*(rmse(actual = forc_germany_mean$actual, predicted = forc_germany_mean$pred) -
        rmse(actual = naive_aprox$actual, predicted = naive_aprox$lagged_H) )/
  (rmse(actual = naive_aprox$actual, predicted = naive_aprox$lagged_H))

#MAE
-100*(mae(actual = forc_germany_mean$actual, predicted = forc_germany_mean$pred) -
        mae(actual = naive_aprox$actual, predicted = naive_aprox$lagged_H) )/
  (mae(actual = naive_aprox$actual, predicted = naive_aprox$lagged_H))
```

Connectedness: European countries

Connectedness measures and robustness analysis.

```

#########################
#   VAR(p) fitting     # Euro Area. Differencied series.
#########################

length(germany_ts)
length(france_ts)
length(italy_ts)
length(spain_ts)
length(portugal_ts)
length(uk_ts)

# Work with differencied series
d1germany <- diff(germany_ts)
d1germany <- d1germany[-1]

d1france <- diff(france_ts)
d1france <- d1france[-1]

d1italy <- diff(italy_ts)
d1italy <- d1italy[-1]

d1spain <- diff(spain_ts)
d1spain <- d1spain[-1]

d1portugal <- diff(portugal_ts)
d1portugal <- d1portugal[-1]
```

```
28
29 d1uk <- diff(uk_ts)
30 d1uk <- d1uk[-1]
31
32 CISS_euro <- as.zoo(data.frame(d1germany,d1france,d1italy,d1spain,d1portugal,d1uk))
33
34 ### VAR(p) ###
35
36 #lag selection
37 VARselect(CISS_euro, lag.max =10)
38 p.lag <- 2
39
40 #estimation
41 var.mod_euro <- VAR(CISS_euro, p=p.lag)
42
43 ########################
44 #   Conectedness       #
45 ########################
46
47 #Hyperparameters
48 H <- 10 #horizon
49 p.lag <- 2 #VAR(p)
50 window <- 250 #width rolling sample
51
52 #Generalized spillover index (Conectedness Table)
53 ConTable_euro <- G.spillover(var.mod_euro, n.ahead = H, standardized = FALSE)
54 ConTable_euro[1:7,1:7]
55
56 #Net spillovers
57 net(ConTable_euro)
58
59 #Dynamic (Net) Spillover Index (Estimates the dynamic spillover index given a moving
       window)
60 DSI_net_euro <- roll.net(CISS_euro, width= window, n.ahead= H, index= "generalized", p=p.
       lag)
61 colnames(DSI_net_euro) <- c("Germany", "France", "Italy", "Spain", "Portugal", "UK")
62 plot(DSI_net_euro, xlab="Time", main="Net Total Directional Connectedness", ylim=c(-6,6))
63
64 #Graphs formating
65
66 #data <- as.data.frame(DSI_net_euro)
67 #dim(data)
68 #dates2 <- rev(dates)[(length(dates)-5505+1):length(rev(dates))]
69 #length(dates2)
70 #dim(data)
71 #data$day <- dates2
72 #
73 #p1 <- ggplot(data, aes(x=day)) +
74 #  geom_hline(yintercept=0, size=1, linetype="dashed", alpha=0.6)+
```

```
#  geom_line(aes(y = data[,1]), size = 1, color="#3288BD") +
#  labs(x = "", y = "Germany") +ylim(-5.8,5.8)+
#  theme(text = element_text(family = "serif", size=16))
#p1
#
#p2 <- ggplot(data, aes(x=day)) +
#  geom_hline(yintercept=0, size=1, linetype="dashed", alpha=0.6)+
#  geom_line(aes(y = data[,2]), size = 1, color="#3288BD") +
#  labs(x = "", y = "France") +ylim(-5.8,5.8)+
#  theme(text = element_text(family = "serif", size=16))
#p2
#
#p3 <- ggplot(data, aes(x=day)) +
#  geom_hline(yintercept=0, size=1, linetype="dashed", alpha=0.6)+
#  geom_line(aes(y = data[,3]), size = 1, color="#3288BD") +
#  labs(x = "", y = "Italy") +ylim(-5.8,5.8)+
#  theme(text = element_text(family = "serif", size=16))
#p3
#
#p4 <- ggplot(data, aes(x=day)) +
#  geom_hline(yintercept=0, size=1, linetype="dashed", alpha=0.6)+
#  geom_line(aes(y = data[,4]), size = 1, color="#3288BD") +
#  labs(x = "", y = "Spain") +ylim(-5.8,5.8)+
#  theme(text = element_text(family = "serif", size=16))
#p4
#
#p5 <- ggplot(data, aes(x=day)) +
#  geom_hline(yintercept=0, size=1, linetype="dashed", alpha=0.6)+
#  geom_line(aes(y = data[,5]), size = 1, color="#3288BD") +
#  labs(x = "Time", y = "Portugal") +ylim(-5.8,5.8)+
#  theme(text = element_text(family = "serif", size=16))
#p5
#
#p6 <- ggplot(data, aes(x=day)) +
#  geom_hline(yintercept=0, size=1, linetype="dashed", alpha=0.6)+
#  geom_line(aes(y = data[,6]), size = 1, color="#3288BD") +
#  labs(x = "Time", y = "UK") +ylim(-5.8,5.8)+
#  theme(text = element_text(family = "serif", size=16))
#p6
#
#grid.arrange(p1, p2, p3, p4, p5, p6, nrow = 3)

#Dynamic Spillover Index
DSI_euro <- roll.spillover(CISS_euro, width= window, n.ahead= H, index= "generalized", p=
    p.lag)
plot(DSI_euro, xlab="Time", ylab="Total connectedness")
max(DSI_euro)
min(DSI_euro)
which(DSI_euro == min(DSI_euro))
```

```
DSI_euro[4842]
DSI_euro[3366]

#DSI vs. "mean" CISS (or vs. Euro CISS)
time = data.frame(euro_ts[251:5755,], DSI_euro/100)
ts.plot(time, gpars= list(xlab="time", ylab=",", col = 1:ncol(time)))
legend("topleft", c("CISS","DSI"), lty=c(1,1), col=c("black","red"), cex=0.6)

#Graphs formating

#data <- time
#dim(data)
#dates2 <- rev(dates)[(length(dates)-5505+1):length(rev(dates))]
#length(dates2)
#dim(data)
#data$day <- dates2
#
#colors <- c("Euro Area CISS" = "#3288BD", "Total Connectedness" = "#D53E4F")
#
#p <- ggplot(data, aes(x=day)) +
#  geom_line(aes(y = data[,1], color="Euro Area CISS"), size = 1) +
#  geom_line(aes(y = data[,2], color="Total Connectedness"), size = 1)+
#  labs(x = "Time", y = "", color = "Series", ylim=c(0,1)) +ylim(0,0.95)+
#  scale_color_manual(values = colors)+theme(text = element_text(family = "serif", size
    =16))
#p

#Dynamic pairwise directional conectedness ij
ConTable_euro[1:7,1:7]
#Germany & france maximum exchange
#Spain & france 2nd maximum exchange
roll.pairwise = function(wind=250, ahead=10, Varp=2){

  dynTable <<- array(dim=c(8,7,length(1:(dim(CISS_euro)[1]-wind+1))))

  for (k in 1:(dim(CISS_euro)[1]-wind+1)) {
    var.mod <- VAR(CISS_euro[k:(k+wind-1)], p=Varp)
    ConTable <- G.spillover(var.mod, n.ahead = ahead, standardized = FALSE)
    dynTable[,,k] <<- ConTable
  }
}

roll.pairwise(wind=250, ahead=10, Varp=2)

ts.plot(dynTable[1,2,], ylim=c(-8,25), ylab="Germany-France") # Dynamic pairwise
    directional connectedness i <- j (germany <- france)
lines(dynTable[2,1,],col="red") # Dynamic pairwise directional connectedness i <- j (
    france <- germany)
lines(dynTable[2,1,]-dynTable[1,2,], col="blue") #Net pairwise directional connectedness
```

```
    ij (j<-i - i<-j) (germany-france)
abline(h = 0, lty = "dashed", col = "black")
dynTable[1,2,][4100]
DSI_euro[4100]

ts.plot(dynTable[4,2,], ylim=c(-6,35), ylab="Spain-France") # Dynamic pairwise
    directional connectedness i <- j (spain <- france)
lines(dynTable[2,4,], col="red") # Dynamic pairwise directional connectedness i <- j (
    france <- spain)
lines(dynTable[2,4,]-dynTable[4,2,], col="blue") #Net pairwise directional connectedness
    ij (j<-i - i<-j) (spain-france)
abline(h = 0, lty = "dashed", col = "black")
max(dynTable[4,2,])

ts.plot(dynTable[2,2,])#france-france


### Robustness analysis ###
## H

roll.spillover
G.spillover

#G.spillover_mod  (MODIFIED FUN)
G.spillover_mod <- function (x, n.ahead=H, standardized = TRUE)
{
  if (!(class(x) == "varest")) {
    stop("\nPlease provide an object of class 'varest', generated by 'VAR()'.\n")
  }
  var.est <- x
  n.ahead <- abs(as.integer(n.ahead))
  EVD <- g.fevd(var.est, n.ahead = n.ahead)
  K <- var.est$K
  table <- do.call(rbind, lapply(EVD, "[", n.ahead, )) *
    100
  CFO <- rowSums(table - diag(diag(table)))
  CTO <- colSums(table) - diag(table)
  CTOT <- colSums(table)
  table <- rbind(table, `C. to others (spillover)` = CTO,
                 `C. to others including own` = CTOT)
  index <- function(x) {
    (sum(x) - sum(diag(x)))/nrow(x)
  }
  if (standardized) {
    table <- cbind(table, `C. from others` = c(CFO,
                                               sum(CFO), sum(CTOT)))/K
  }
  else {
    table <- cbind(table, `C. from others` = c(CFO,
```

```
                                                index(table[1:K, ]), sum(CTOT)))
  }
  return(table)
}

#roll.spillover_mod (MODIFIED FUN)
roll.spillover_mod <- function (data, width, n.ahead = H, index = c("orthogonalized",
                                "generalized"), ortho.type = c("single", "partial",
                                "total"), ...)
{
  if (!(class(data) == "zoo")) {
    stop("\nPlease provide an object of class 'zoo', generated by 'as.zoo'.\n")
  }
  K <- ncol(data) + 1
  roll.index <- switch(match.arg(index), orthogonalized = {
    switch(match.arg(ortho.type), single = {
      rollapply(data, width = width, FUN = function(z) {
        O.spillover(VAR(z, ...), ortho.type = c("single"))[K,
                                                           K]
      }, by.column = FALSE, align = "right")
    }, partial = {
      rollapply(data, width = width, FUN = function(z) {
        O.spillover(VAR(z, ...), ortho.type = c("partial"))[K,
                                                            K]
      }, by.column = FALSE, align = "right")
    }, total = {
      rollapply(data, width = width, FUN = function(z) {
        O.spillover(VAR(z, ...), ortho.type = c("total"))[K,
                                                          K]
      }, by.column = FALSE, align = "right")
    })
  }, generalized = {
    rollapply(data, width = width, FUN = function(z) {
      G.spillover_mod(VAR(z, ...))[K, K]
    }, by.column = FALSE, align = "right")
  })
  return(roll.index)
}

########

H <- 10
DSI_H10_euro <- roll.spillover_mod(CISS_euro, width= 250, n.ahead=H, index="generalized",
     p=p.lag)
H <- 3
DSI_H3_euro <- roll.spillover_mod(CISS_euro, width= 250, n.ahead=H, index="generalized",
    p=p.lag)
H <- 15
DSI_H15_euro <- roll.spillover_mod(CISS_euro, width= 250, n.ahead=H, index="generalized",
```

```
    p=p.lag)

time2 = data.frame(DSI_H3_euro,DSI_H10_euro,DSI_H15_euro)
ts.plot(time2, gpars= list(xlab="time", ylab=",", col = 1:ncol(time2)))
legend("topleft", c("H=3","H=10", "H=15"), lty=c(1,1,1), col=1:ncol(time2), cex=0.6)

#Graphs formating

#data <- time2
#dates2 <- rev(dates)[(length(dates)-length(DSI_H3_euro)+1):length(rev(dates))]
#length(dates2)
#dim(data)
#data$day <- dates2

#colors <- c("H=3" = "#D53E4F", "H=10" = "#FEE08B", "H=15"="#3288BD")
#
#p <- ggplot(data, aes(x=day)) +
#  geom_line(aes(y = DSI_H3_euro, color="H=3"), size = 1) +
#  geom_line(aes(y = DSI_H10_euro, color="H=10"), size = 1)+
#  geom_line(aes(y = DSI_H15_euro, color="H=15"), size = 1)+
#  labs(x = "Time", y = "Total Connectedness Index", color = "Horizon") +
#  scale_color_manual(values = colors)+theme(text = element_text(family = "serif", size
    =16))
#p

########

## Window
DSI_150_euro <- roll.spillover(CISS_euro, width= 150, n.ahead= 10, index= "generalized",
    p=p.lag)
DSI_200_euro <- roll.spillover(CISS_euro, width= 200, n.ahead= 10, index= "generalized",
    p=p.lag)
DSI_250_euro <- roll.spillover(CISS_euro, width= 250, n.ahead= 10, index= "generalized",
    p=p.lag)
DSI_300_euro <- roll.spillover(CISS_euro, width= 300, n.ahead= 10, index= "generalized",
    p=p.lag)
length(DSI_150_euro[101:length(DSI_150_euro)])
length(DSI_200_euro)
length(DSI_250_euro)

time = data.frame(DSI_150_euro[101:length(DSI_150_euro)], DSI_200_euro[51:length(DSI_200_
    euro)], DSI_250_euro)
ts.plot(time, gpars= list(xlab="time", ylab=",", col = 1:ncol(time)))
legend("topleft", c("w=150","w=200","w=250"), lty=c(1,1,1), col=1:ncol(time), cex=0.6)

time = data.frame(DSI_200_euro[101:length(DSI_200_euro)], DSI_250_euro[51:length(DSI_250_
    euro)], DSI_300_euro)
ts.plot(time, gpars= list(xlab="time", ylab=",", col = 1:ncol(time)))
legend("topleft", c("w=200","w=250","w=300"), lty=c(1,1,1), col=1:ncol(time), cex=0.6)
```

```
302
303 #Graphs formating
304
305 #data <- time
306 #dim(time)
307 #dates2 <- rev(dates)[(length(dates)-5455+1):length(rev(dates))]
308 #length(dates2)
309 #dim(data)
310 #data$day <- dates2
311 #
312 #colors <- c("w=200" = "#D53E4F", "w=250" = "#FEE08B", "w=300"="#3288BD")
313 #
314 #p <- ggplot(data, aes(x=day)) +
315 #  geom_line(aes(y = data[,1], color="w=200"), size = 1) +
316 #  geom_line(aes(y = data[,2], color="w=250"), size = 1)+
317 #  geom_line(aes(y = data[,3], color="w=300"), size = 1)+
318 #  labs(x = "Time", y = "Total Connectedness Index", color = "Window width") +
319 #  scale_color_manual(values = colors)+theme(text = element_text(family = "serif", size
    =16))
320 #p
```

LSTM network for $h = 5$

Input-output sequences preparation, LSTM architecture and Bayesian Optimizer.

```
1
2 #Read csv
3 euro = pd.read_csv('euro.csv', skiprows=5)
4 uk = pd.read_csv('uk.csv', skiprows=5)
5 us = pd.read_csv('us.csv', skiprows=5)
6 china = pd.read_csv('china.csv', skiprows=5)
7 germany = pd.read_csv('germany.csv', skiprows=5)
8 italy = pd.read_csv('italy.csv', skiprows=5)
9 spain = pd.read_csv('spain.csv', skiprows=5)
10 portugal = pd.read_csv('portugal.csv', skiprows=5)
11 france = pd.read_csv('france.csv', skiprows=5)
12
13 # Change column names
14 euro.columns = ['date', 'value', 'status']
15 uk.columns = ['date', 'value', 'status']
16 us.columns = ['date', 'value', 'status']
17 china.columns = ['date', 'value', 'status', 'comment']
18 germany.columns = ['date', 'value', 'status', 'comment']
19 italy.columns = ['date', 'value', 'status']
20 spain.columns = ['date', 'value', 'status']
21 portugal.columns = ['date', 'value', 'status']
22 france.columns = ['date', 'value', 'status']
23
24 #Reverse data frame (so that first observation is the oldest)
```

```
euro = euro.loc[::-1].reset_index(drop=True)
uk = uk.loc[::-1].reset_index(drop=True)
us = us.loc[::-1].reset_index(drop=True)
china = china.loc[::-1].reset_index(drop=True)
germany = germany.loc[::-1].reset_index(drop=True)
italy = italy.loc[::-1].reset_index(drop=True)
spain = spain.loc[::-1].reset_index(drop=True)
portugal = portugal.loc[::-1].reset_index(drop=True)
france = france.loc[::-1].reset_index(drop=True)

#Plot
plt.figure(figsize=(10,4))
plt.title("CISS")
plt.plot(france['value'])
plt.xlabel('Date', fontsize=18)
plt.ylabel('', fontsize=18)
plt.show()

#Select value column
euro = euro.filter(['value'])
uk = uk.filter(['value'])
us = us.filter(['value'])
china = china.filter(['value'])
germany = germany.filter(['value'])
italy = italy.filter(['value'])
spain = spain.filter(['value'])
portugal = portugal.filter(['value'])
france = france.filter(['value'])

#Convert dataframe to numpy
euro = euro.values
uk = uk.values
us = us.values
china = china.values
germany = germany.values
italy = italy.values
spain = spain.values
portugal = portugal.values
france = france.values

france.shape , china.shape , len(euro)

#Partitions for Train + Validation + Test
train_pct = 0.75
valid_pct = 0.15
test_pct = 0.10

training_data_len = math.ceil(len(euro)*train_pct)
valid_data_len = math.ceil(len(euro)*valid_pct)
```

```
74
75 #training_data_len = 4320
76 #valid_data_len = 860
77
78 #Hyperparamether sequence_length (prediction at t+1 by using previous 'sequence_length'
       days)
79 sequence_length = 64
80
81 #Create the Train , Validation & Test datasets
82 train_euro = euro[0:training_data_len, :]
83 train_uk = uk[0:training_data_len, :]
84 train_us = us[0:training_data_len, :]
85 train_germany = germany[0:training_data_len, :]
86 train_italy = italy[0:training_data_len, :]
87 train_spain = spain[0:training_data_len, :]
88 train_portugal = portugal[0:training_data_len, :]
89 train_france = france[0:training_data_len, :]
90
91 valid_euro = euro[training_data_len:(training_data_len+valid_data_len), :]
92 valid_uk = uk[training_data_len:(training_data_len+valid_data_len), :]
93 valid_us = us[training_data_len:(training_data_len+valid_data_len), :]
94 valid_germany = germany[training_data_len:(training_data_len+valid_data_len), :]
95 valid_italy = italy[training_data_len:(training_data_len+valid_data_len), :]
96 valid_spain = spain[training_data_len:(training_data_len+valid_data_len), :]
97 valid_portugal = portugal[training_data_len:(training_data_len+valid_data_len), :]
98 valid_france = france[training_data_len:(training_data_len+valid_data_len), :]
99
100 test_euro = euro[(training_data_len+valid_data_len): , :]
101 test_uk = uk[(training_data_len+valid_data_len): , :]
102 test_us = us[(training_data_len+valid_data_len): , :]
103 test_germany = germany[(training_data_len+valid_data_len): , :]
104 test_italy = italy[(training_data_len+valid_data_len): , :]
105 test_spain = spain[(training_data_len+valid_data_len): , :]
106 test_portugal = portugal[(training_data_len+valid_data_len): , :]
107 test_france = france[(training_data_len+valid_data_len): , :]
108
109 print(train_euro.shape ,valid_euro.shape, test_euro.shape)
110
111 #Adjust china valid and test datasets to the other countries lengths
112 train_china = china[0:(len(china)-len(test_euro)-len(valid_euro)), :]
113 valid_china = china[(len(china)-len(test_euro)-len(valid_euro)):(len(china)-len(test_euro
       )), :]
114 test_china = china[(len(china)-len(test_euro)): , :]
115 print(train_china.shape, valid_china.shape, test_china.shape)
116
117 #Create function to compute x_train and y_train sequences (x: input & y:output).
118 def seq_fun(train_data, valid_data, test_data):
119   x_train = []
120   y_train = []
```

```
  x_valid = []
  y_valid = []
  x_test = []
  y_test = []

  for i in range(sequence_length, len(train_data)-4):
    x_train.append(train_data[(i-sequence_length):i, 0])
    y_train.append(train_data[i+4,0])

  for i in range(sequence_length, len(valid_data)-4):
    x_valid.append(valid_data[(i-sequence_length):i, 0])
    y_valid.append(valid_data[i+4,0])

  for i in range(sequence_length, len(test_data)-4):
    x_test.append(test_data[(i-sequence_length):i, 0])
    y_test.append(test_data[i+4,0])

  x_train, y_train, x_valid, y_valid, x_test, y_test = np.array(x_train), np.array(
    y_train), np.array(x_valid) , np.array(y_valid), np.array(x_test), np.array(y_test)
  return(x_train, y_train, x_valid, y_valid, x_test, y_test)

#Obtain sequences on our datasets
train_euro_x , train_euro_y, valid_euro_x, valid_euro_y, test_euro_x, test_euro_y =
    seq_fun(train_euro, valid_euro, test_euro)
train_uk_x , train_uk_y, valid_uk_x, valid_uk_y, test_uk_x, test_uk_y = seq_fun(train_uk,
     valid_uk, test_uk)
train_us_x , train_us_y, valid_us_x, valid_us_y, test_us_x, test_us_y = seq_fun(train_us,
     valid_us, test_us)
train_germany_x , train_germany_y, valid_germany_x, valid_germany_y, test_germany_x,
    test_germany_y = seq_fun(train_germany, valid_germany, test_germany)
train_italy_x , train_italy_y, valid_italy_x, valid_italy_y, test_italy_x, test_italy_y =
     seq_fun(train_italy, valid_italy, test_italy)
train_spain_x , train_spain_y, valid_spain_x, valid_spain_y, test_spain_x, test_spain_y =
     seq_fun(train_spain, valid_spain, test_spain)
train_portugal_x , train_portugal_y, valid_portugal_x, valid_portugal_y, test_portugal_x,
     test_portugal_y = seq_fun(train_portugal, valid_portugal, test_portugal)
train_france_x , train_france_y, valid_france_x, valid_france_y, test_france_x,
    test_france_y = seq_fun(train_france, valid_france, test_france)
train_china_x , train_china_y, valid_china_x, valid_china_y, test_china_x, test_china_y =
     seq_fun(train_china, valid_china, test_china)

#Concatenate train and validation sequences (input and output)
x_train_con = np.concatenate((train_euro_x, train_uk_x, train_us_x, train_germany_x,
                              train_italy_x, train_spain_x, train_portugal_x,
    train_france_x, train_china_x))

y_train_con = np.concatenate((train_euro_y, train_uk_y, train_us_y, train_germany_y,
                              train_italy_y, train_spain_y, train_portugal_y,
    train_france_y, train_china_y))
```

```
158
159 x_valid_con = np.concatenate((valid_euro_x, valid_uk_x, valid_us_x, valid_germany_x,
160                                valid_italy_x, valid_spain_x, valid_portugal_x,
    valid_france_x, valid_china_x))
161
162 y_valid_con = np.concatenate((valid_euro_y, valid_uk_y, valid_us_y, valid_germany_y,
163                                valid_italy_y, valid_spain_y, valid_portugal_y,
    valid_france_y, valid_china_y))
164
165
166 x_train_con.shape , y_train_con.shape, x_valid_con.shape, y_valid_con.shape
167
168 #Reshape the data to fit to LSTM. You have to input 3d array (number of samples, number
    of timesteps, number of features).
169 #In this scheme, we only feed the model with one sequence (feature) at a time.
170 x_train_con = np.reshape(x_train_con, (x_train_con.shape[0], x_train_con.shape[1], 1))
171 x_valid_con = np.reshape(x_valid_con, (x_valid_con.shape[0], x_valid_con.shape[1], 1))
172 x_train_con.shape, x_valid_con.shape
173
174 #Build the LSTM model
175 #Define a function that creates a hypermodel with a bounded (but not specified)
    hyperparameter space.
176
177 def hypermodel(hp):
178   model=Sequential()
179   model.add(LSTM(units=hp.Int('base_layer',min_value=2, max_value=10, step=2),
180                                dropout=hp.Float('droput_base',min_value=0.0, max_value
    =0.20, step=0.1),
181                                recurrent_dropout=hp.Float('redroput_base',min_value=0.0,
    max_value=0.20, step=0.1), return_sequences=True, input_shape = (x_train_con.shape
    [1],1)))
182
183   for i in range(hp.Int('n_hlayers', 1, 2)):
184       model.add(LSTM(units=hp.Int('dense_' + str(i), min_value=2, max_value=10, step=2),
    dropout=hp.Float('droput_' + str(i),min_value=0.0, max_value=0.20, step=0.1),
    recurrent_dropout=hp.Float('redroput_' + str(i),min_value=0.0, max_value=0.20, step
    =0.1), return_sequences=True))
185
186   model.add(LSTM(hp.Int('final_layer',min_value=2, max_value=10, step=2),
187                                dropout=hp.Float('droput_final',min_value=0.0, max_value
    =0.20, step=0.1),
188                                recurrent_dropout=hp.Float('redroput_final',min_value=0.0,
    max_value=0.20, step=0.1), return_sequences=False))
189
190   model.add(Dense(1, activation=hp.Choice('dense_activation',values=['sigmoid', 'relu'],
    default='sigmoid')))
191
192   model.compile(loss='mean_squared_error', optimizer=tf.keras.optimizers.Adam(hp.Choice('
    learning_rate', values=[1e-2, 1e-3])))
```

```
  return model

#Bayesian Hyperparameter Optimizer
bayesian_opt_tuner = keras_tuner.BayesianOptimization(
    hypermodel=hypermodel,
    objective='val_loss',
    max_trials=8,
    executions_per_trial=1,
    seed=2022,
    project_name='run4_t5_v1',
    overwrite=True,)

bayesian_opt_tuner.search(x=x_train_con ,y=y_train_con,
             epochs=10,
             validation_data=(x_valid_con, y_valid_con),
             batch_size=16,
             callbacks=[tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                patience=7,
                                min_delta=1e-7,
                                verbose=1,
                                restore_best_weights=True),
                        tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                    factor=0.1,
                                    patience= 3,
                                    verbose=1,
                                    min_delta=1e-7,
                                    mode='min')]
             )

model = bayesian_opt_tuner.get_best_models(num_models=1)[0]

#Check
check = model.evaluate(x_valid_con, y_valid_con, batch_size=16)
print("test loss", check)

hyp= bayesian_opt_tuner.get_best_hyperparameters(1)[0]
hyp.values
```