Institute of Telecommunications
TU Wien

Master Thesis

Learning directed Graph Shifts from High-Dimensional Time Series

Lukas Nagel
0927167

Supervisor
Univ.Prof. Dipl.-Ing. Dr.-Ing. Norbert Görtz

June 2017

Abstract

Graph Signal Processing is an emerging field of signal processing that combines classical signal processing with graph theory. There are two approaches which either use undirected weighted graphs that allow the usage of Laplacian matrix, or the more general approach, which is based on algebraic features, including all weighted directed graphs.

We investigate the concept of causal graph signal processing that was proposed by J. Mei and J.M.F. Moura. In a causal graph process, the current signal depends on the past signals through graph filters that consist of a polynomial of the graph shift matrix. With their algorithm, the graph shift matrix and filter coefficients can be learned from a sequence of observed data vectors. We evaluate the performance for estimating the shift matrix from an artificially generated causal graph process.

Furthermore, we apply the estimation algorithm on two real-world data sets. The first data set contains daily temperature data from different countries. In the second example, we tried to model Austrian stock prices with causal graph processes.

Kurzfassung

Graph-basierte Signalverarbeitung (graph signal processing) ist ein neu entstehendes Gebiet in der Signalverarbeitung, das klassische Signalverarbeitung mit der Graphentheorie vereint. Es gibt zwei Ansätze: einen mit ungerichteten gewichteten Graphen, die es erlauben, die Laplacematrix zu verwenden, und den allgemeineren Ansatz, welcher auf algebraischen Eigenschaften basiert und alle gewichteten gerichteten Graphen abdeckt.
Wir untersuchen das Konzept der kausalen Graph-basierten Signalverarbeitung, das von J. Mei und J.M.F. Moura vorgestellt wurde. In einem kausalen Graphprozess hängt das aktuelle Signal von vergangenen Signalen ab, auf die Graphfilter angewendet wurden. Die Graphfilter bestehen aus einem Polynom – der „Graph Shift“-Matrix. Mit ihrem Algorithmus können die „Graph Shift“-Matrix und die Filterkoeffizenten aus einer Folge von beobachteten Datenvektoren gelernt werden. Wir evaluieren die Leistungsfähigkeit der Methode für die Schätzung der „Graph Shift“-Matrix aus einem künstlich generierten kausalen Graphprozess.
Weiters wenden wir den Schätzalgorithmus auf zwei reale Datensätze an. Der erste Datensatz enthält tägliche Temperaturdaten aus verschiedenen Ländern. Im zweiten Beispiel versuchten wir österreichische Aktienmarktpreise mit kausalen Graphprozessen zu modellieren.

Acknowledgements

I would like to use this opportunity to express my thankfulness to everyone who supported me throughout my time at the university.

Thanks to my supervisor, Univ.Prof.Dr. Norbert Görtz for all the helpful meetings, countless ideas and useful advice.

I would furthermore like to thank my friends and colleagues at the university for all the good times, happy moments and countless hours of solving resilient problems and talking about interesting and not so interesting topics.

Thanks to Valerija for making me happy every day. Additionally, I would like to thank her for regularly listening to my concerns, telling me that I shouldn't worry about them and reminding me to write something even tough I was protesting. Without her, I would not be sitting here writing these lines.

Finally I want to thank my family, my parents Elfi and Reinhart, who were consistently supporting me, being there for me, cheering me up, listening to me when I was complaining or explaining what I just learned. Thanks to my not so little sister Anna.

Contents

List of Tables

List of Figures

1 Introduction

With the ongoing digitalization of our surroundings comes the need for new approaches on how to handle the flood of data. Therefore, it is natural to search for the connections and structure inside seemingly unorganized sets of numbers.

Over the last century, the manipulation of time series resulted in the field of signal processing. Different techniques were developed to filter, analyze and synthesize time signals that appear in various fields of science and engineering. Starting from simple one-dimensional time series the analysis evolved to higher-dimensional signals. With more and more sources of signals, the increasing complexity limits the investigation of all possible pairwise relations between data series. The search for sparser models led signal processing researchers to discover graphs, which were previously investigated in the scope of graph theory.

Combining graphs with signal processing resulted in the field of graph signal processing, which aims to incorporate connections characterized by graphs into the processing of signals. By associating each node of the graph with a signal value, we speak of graph signals. Since the field of graph signal processing is still new, there is still no canonical approach to the area.

Therefore, in this work, after giving a short definition of graphs, we try to present two popular approaches to graph signal processing, as well as mentioning their benefits and drawbacks. The first approach makes use of Laplacian matrices, which have their origins in graph theory: this was popularized by Shuman [1]. It is characterized by undirected graphs and offers an intuitive entry point to graph signal processing. As a second approach, we discuss the more formal path that was introduced by Sandryhaila and Moura [2]. It emphasizes the position of graph signal processing as a strict generalization of classical signal processing, by axiomatically requiring properties of signal processing and showing that these can be fulfilled by their graph counterpart.

To apply the framework of graph signal processing, one always needs the combination of signals and an associated graph. While there are some applications where the choice of the graph appears natural as in images or social networks, there is the intention to extend the scope of graph processing to other areas. One option is to derive the graph from the given nodes using a metric between them. This approach was chosen in [1] to create a graph between weather stations based on their geographical distance. By the undirected nature of distances, this always yields undirected graphs. It can capture similarities but not “causal” dependencies.

Another way is to learn the graph from known data that is interpreted as graph signals associated with an unknown, to be learned graph as it was shown in [3]. Again, there was a lack of a method which leads to a directed graph. In 2016, Mei and Moura published a paper introducing a new technique on how to learn a *directed* graph adjacency matrix

assuming the training data conforms to what they call a causal graph process, in which the current graph signal depends on previous signals over graph filters.

We discuss the algorithm, which they proposed, as well as the model of causal graph processes. Further on, we then briefly describe the estimation performance regarding the graph adjacency matrix and the graph filter coefficients by showing two different examples. To complete the review, we obtained two openly accessible data sets. One data set consists of daily temperature data from weather stations that are distributed in selected countries. The second data set, we used to test the algorithm with, are daily prices from the Austrian stock market.

In the end, we summarize our findings and conclusions from our experiments.

2 Graph Signal Processing

Signal Processing is an important area of telecommunications but also of many other fields such as control, finance or seismology.

Due to the continuous nature of our world, the first concepts of signals representing physical quantities were mostly described by continuous calculus. Numerical and discrete techniques were employed when finding analytical solutions was infeasible.

According to [4], starting in the 1950s, the field of discrete signal processing began gaining more and more attention. Sampling of analog signals resulted in discrete signals that could be analyzed, processed and stored in computers. Since then, computers have become faster and more affordable, which made digital signal processing very attractive. Nowadays, the digital paradigm prevails and analog systems have been pushed to niche applications.

Since computers are a part of our everyday life, a new type of signal is becoming of importance. Signals are no longer only simple sampled representations of their analog counterparts, but enormous amounts of data generated by information systems all around the world. Sensor networks, social networks and business systems, generate signals where a clear notion of time, as in a sampled physical quantity, is less important. We are interested to find, analyze and make use of connections and relations hidden in those signals.

This motivated the emerging field of *Graph Signal Processing* [1], which tries to bring together graph theory and signal processing. Each node of the graph is assigned a signal value and is related to the other nodes over the weighted graph.

In the following, we first introduce the different concepts of graphs and how those can be used to generalize (digital) signal processing to graph signal processing.

2.1 Types of Graphs

We start by introducing the variations of graphs that will be discussed in this thesis following the notation of [5]. In literature, we can find a variety of graph-like structures, but we will restrict ourselves to those which are used later.

2.1.1 Graphs and Directed Pseudographs

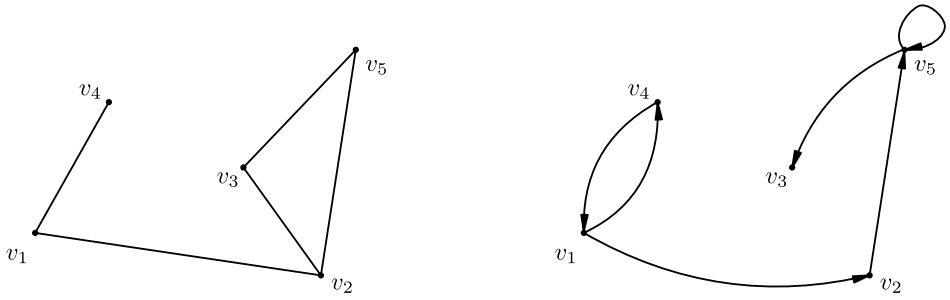
In graph theory a *graph* G, can be defined as an ordered couple $(V(G),\ E(G))$, where $V(G)$ is an nonempty set of vertices and $E(G)$ is the set of edges. Each edge e is a set $e = \{v_1, v_2\}$ with $v_1, v_2 \in V(G)$. An example is shown in Figure 2.1a.

While a graph connects different nodes with each other, a *directed pseudograph* can be seen as a more general version of a graph. A directed pseudograph G is a an ordered

couple $(V(G),\ A(G))$ of a nonempty set of vertices $V(G)$ and a set of arcs $A(G)$. Each arc is an ordered couple of the form $a = (v_1, v_2) \in A(G) \subseteq V(G) \times V(G)$. Note that, although the definitions of graph and directed pseudograph look similar, there are important differences.

First of all, the arcs of directed pseudographs are directed – they have a start and an end node. Looking at two nodes, they can have an arc in $a_1 = (v_1, v_2)$, from node v_1 to v_2, and another one $a_2 = (v_2, v_1)$ in the opposite direction. Secondly, self-loops are allowed, i.e. arcs of the form $a = (v, v)$, that start and end in the same vertex.

Figure 2.1 shows a graphical representation of two examples.



(a) graph (b) directed pseudograph

Figure 2.1: Examples of a graph and a directed pseudograph

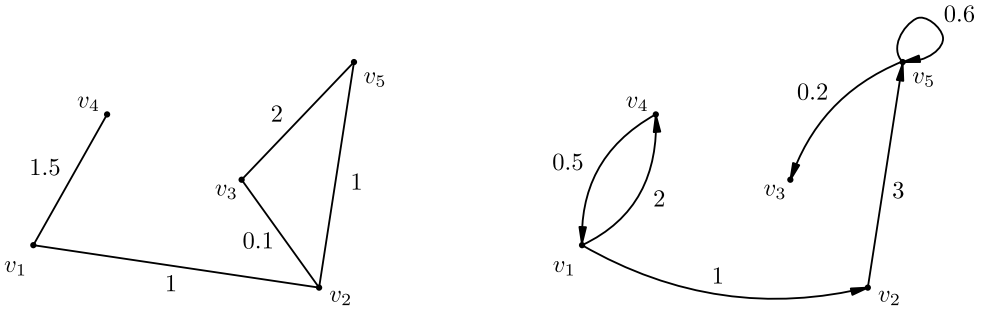
2.1.2 Weighted Graphs and Weighted Directed Pseudographs

In the previous section we defined graphs and directed pseudographs. Both structures include a binary decision for each pair of nodes – connected or not connected. For some applications, it might be beneficial to introduce a weighting for each edge or arc.

As an extension to a graph, we introduce a *weighted graph* G as the ordered triple $(V(G),\ E(G), w)$. For this, we extend the graph with a function $w : E(G) \to \mathbb{R}$ that assigns a real weight to each edge in $E(G)$.

In a similar fashion, we define the *weighted directed pseudograph* as the corresponding ordered triple $(V(G),\ A(G), w)$. Again, we need a weighting function $w : A(G) \to \mathbb{R}$.

Figure 2.2 shows two examples based on the graphs on Figure 2.1 but now the edges/arcs are weighted.



(a) weighted graph

(b) weighted directed pseudograph

Figure 2.2: Examples of a weighted graph and a weighted directed pseudograph

Until now, we introduced four different graph like structures. For clarity, a summary of all of them is shown in Table 2.1.

	undirected	directed
unweighted	graph	directed pseudograph
weighted	weighted graph	weighted directed pseudograph

Table 2.1: Introduced graphs

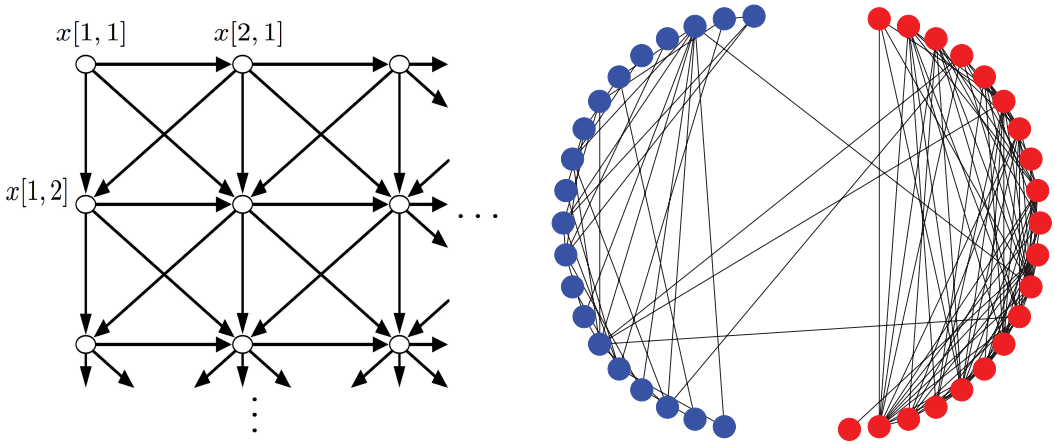
In the following, we will mostly use the *weighted directed pseudograph* and simply call it *directed graph*. The defined graphs enable us to proceed with the discussion of graph signals.

2.2 Graph Signals

Analyzing nodes and their connections are parts of graph theory. We can interpret each node as an entity that is generating signal values. For example, we can see a node as a temperature sensor that outputs temperature values. Nowadays, sensors are common and the temperature sensor might be part of a *network* of temperature sensors. For a given time, each sensor produces a reading, and we can establish a mapping from the nodes to the signal written as

$$x : V(G) \to \mathbb{C}, \quad v_i \mapsto x_i. \tag{2.1}$$

We call x graph signal. This is not restricted to sensor values. The signal values can for example be representations of opinions in a network of blogs, like it is shown in Figure 2.3b. They could also be color values of an image pixel as in 2.3a. For example the temperature values from weather stations in France are interpreted as a graph signal and shown in the Figure 2.4. The length of the red bars visualizes the relative temperature at each sensor node.



(a) pixels in a picture (from [6]) (b) connected political blogs (from [7])

Figure 2.3: Examples of graphs

Given N nodes, we number them in an arbitrary way and see the graph signal as a vector

$$\mathbf{x} = [x_1, x_2, \ldots, x_N] \in \mathbb{C}^N. \tag{2.2}$$

This way, we could treat the graph signal as a classical vector signal, but we would ignore the additional information we obtain when we make use of the underlying graph G the signal is living on. Until now, we only specified a mapping from the nodes to the signal values and deliberately left open how the connection information is incorporated within the model.

The literature on graph signal processing can be divided into two big groups that introduce the additional graph information in a different way. In the following, we will give a short introduction of their main ideas.

2.2.1 Graph Signals - the Laplacian Approach

The first approach to graph signals became well known by the paper of Shuman [1] which we will, in the scope of this thesis, briefly sum up. In this framework, graphs are usually assumed weighted and undirected. After labeling the nodes in our graph, the graph signal can be written as a vector $\mathbf{x}$. Each edge of the graph is represented by an entry $W_{i,j}$ in the symmetric weight matrix $\mathbf{W}$. Non-zero entries imply the presence of an edge, while zeros in the weight matrix tell that the corresponding nodes are not connected. Since no self-loops are allowed in a normal weighted graph, the matrix $\mathbf{W}$ has a zero diagonal.

A major question of graph signal processing is the question how the graph should be constructed. In the Laplacian approach, the graph weights often represent similarity between the nodes. Two nodes are connected with a high weight if they are alike. In contrast, a small or zero weight indicates low or no similarity. To express the similarity between two nodes, a notion of distance is needed. If a distance between the nodes

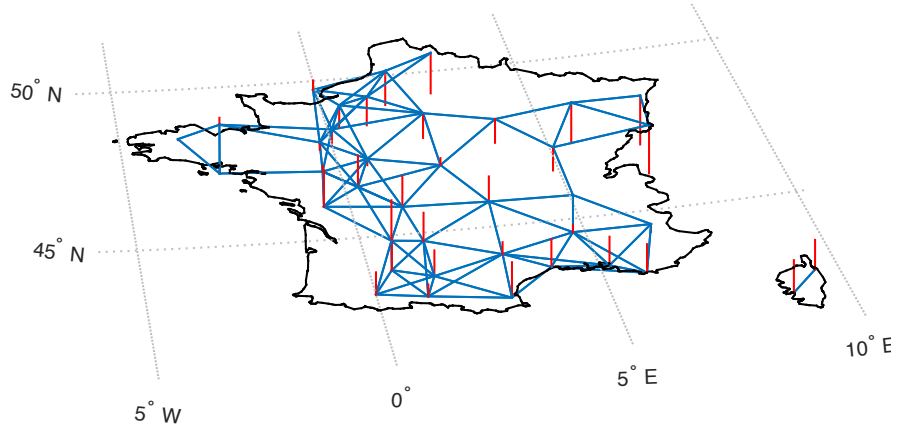


Figure 2.4: Distance based graph of temperature sensor network in France and temperature values shown as red bars for each station

can be calculated, [1] proposes to build the graph weights using a thresholded Gaussian kernel weighting function such as

$$\mathbf{W}_{i,j} = \begin{cases} \exp -\frac{d(i,j)^2}{2\theta^2} & , \quad d(i,j) \leq \varepsilon \\ 0, & \text{otherwise} \end{cases}. \tag{2.3}$$

Here $d(i,j)$ is a distance between the nodes i and j. The nodes are connected if their distance is smaller than some threshold ε and the weights are based upon the closeness with respect to the chosen distance $d(\cdot,\cdot)$. Figure 2.4 shows a graph which is connected according to equation 2.3, where we used the geographical distance to determine the graph between the temperature sensors. A maximum distance of $\varepsilon = 200\,\text{km}$ between the nodes was used. Even tough this way of constructing the graph appears to be straightforward a potential disadvantage is visible in our example. Because of the maximum distance, the two weather stations on Corsica are disconnected from the rest of the graph, which could be of disadvantage if the disconnected signal part should be recovered from the other signal values. An algorithm that does not share this property is described in [8].

The way of constructing the graph should be inferred from the problem type. Using the geographical distance to connect the stations follows from the underlying assumption that geometrically close nodes measure similar data. Note that the unweighted nature of the graph suggests similarity of connected nodes, but it cannot make statements regarding cause and effect, i.e., two nodes might have similar values because they are close and are influenced by the same weather phenomenon but one cannot say whether or how they influence each other. We will now try to formulate the notation of similarity in a mathematical way. For this, we will need the Graph Laplacian.

Given a weight matrix $\mathbf{W}$, we can define a diagonal degree matrix $\mathbf{D}$ which contains the sum of all incident weights $\mathbf{D}_{i,i} = \sum_{j=1}^{N} \mathbf{W}_{i,j}$. Using $\mathbf{D}$ and $\mathbf{W}$ we can define the graph Laplacian $\mathbf{L}$ as

$$\mathbf{L} := \mathbf{D} - \mathbf{W}.$$

The Graph Laplacian $\mathbf{L}$ can be seen as a difference operator that can be applied to graph signals $\mathbf{x} \in \mathbb{C}^N$. The component-wise result is

$$(\mathbf{L}\mathbf{x})(i) = \sum_{j=1}^{N} \mathbf{W}_{i,j} \left(\mathbf{x}(i) - \mathbf{x}(j)\right) = \sum_{j \in \mathcal{N}} \mathbf{W}_{i,j} \left(\mathbf{x}(i) - \mathbf{x}(j)\right). \tag{2.4}$$

This shows that the i-th component of the Laplacian output is the weighted difference of the signal value $\mathbf{x}(i)$ and the value at the adjacent nodes $\mathbf{x}(j)$. In equation 2.4 the second equation clarifies that the sum only consists of signals from the set of neighbor nodes $\mathcal{N}$, with a $\mathbf{W}_{i,j} \neq 0$, that are connected to node i. Before we discuss more on smoothness, we want to introduce the concept of a Graph Fourier transform in this setting.

In [1], the Graph Fourier transform is introduced by a comparison of eigenfunctions and eigenvectors. It is known that the classical Fourier transform

$$F(j\omega) := \langle f(t), e^{j\omega t} \rangle = \int_{-\infty}^{\infty} f(t) e^{j\omega t} \mathrm{d}t \tag{2.5}$$

expands the function $f(t)$ in terms of $e^{j\omega t}$. These complex exponentials are the eigenfunctions of the one-dimensional Laplace operator which can be seen by calculating

$$-\Delta(e^{j\omega t}) = \frac{\partial^2}{\partial t^2} e^{j\omega t} = \omega^2 e^{j\omega t}. \tag{2.6}$$

The eigenvalues are given by ω^2 and therefore, closely related to frequency. If we want to apply this concept to graph signal processing, we have to find the eigenvectors with respect to the in equation 2.2.1 defined Laplace operator $\mathbf{L}$.

Since the graph Laplacian $\mathbf{L}$ is, by definition, a real symmetric matrix, it has N orthonormal eigenvectors $\mathbf{u}_k$ which fulfill the equation

$$\mathbf{L}\mathbf{u}_k = \lambda_k \mathbf{u}_k \tag{2.7}$$

with the corresponding non negative eigenvalues λ_k. Analog to Equation 2.6 we can obtain the k-th component of the graph Fourier $\hat{\mathbf{x}}$ of the graph signal $\mathbf{x}$ transform by computing the scalar product with the matching eigenvector $\mathbf{u}_k$ as

$$\hat{\mathbf{x}}(k) = \langle \mathbf{x}, \mathbf{u}_k \rangle = \mathbf{u}_k^H \mathbf{x}. \tag{2.8}$$

By grouping all eigenvectors $\mathbf{u}_k$ into the eigenvector matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N]$ we can express the transform as

$$\hat{\mathbf{x}} = \mathbf{U}^H \mathbf{x} \tag{2.9}$$

and since $\mathbf{U}$ is a unitary matrix, we can find the inverse graph Fourier transform to be

$$\mathbf{x} = \left(\mathbf{U}^H\right)^{-1} \hat{\mathbf{x}} = \mathbf{U}\hat{\mathbf{x}}. \tag{2.10}$$

Another analogy to the classical Fourier transform we want to address, is the smoothness of the different Fourier components $\mathbf{u}_k$. The original Fourier transform has the eigenfunctions $e^{j\omega t}$ which correspond to the eigenvalues ω^2, where ω is usually termed to be

a frequency. Low frequencies - low ω correspond to a slowly changing harmonic signal – the corresponding eigenfunction, and higher frequencies capture stronger fluctuations. The ordering of the frequencies is naturally given by their magnitude.

The question arises whether such a notion of low and high frequencies can be ported to the graph Fourier transform. To answer this question, one first needs to define how to quantitatively define the concept of slow and fast variation, which seems clear intuitively for 1D signals. Although the graph signal $\mathbf{x}$ can be seen as a vector in $\mathbb{C}^N$, it is not enough to relate only the neighboring elements in the vector to each other. Instead, we want to see how close the signal values on connected nodes are and specifically weigh in differences of edges that are linked with a high weight. We first define a local variation for one node index by i as

$$\|\nabla_i \mathbf{x}\|_2 = \left(\sum_{j \in \mathcal{N}} W_{i,j} (\mathbf{x}(i) - \mathbf{x}(j))^2 \right)^{\frac{1}{2}}, \tag{2.11}$$

the term can be interpreted as smoothness of the signal around the node i. Consequently, we can build a global smoothness expression by summing over the local variation at all nodes which yields

$$S_p(\mathbf{x}) = \frac{1}{p} \sum_{i \in V} \|\nabla_i \mathbf{x}\|_2^p = \frac{1}{p} \sum_{i \in V} \left(\sum_{j \in \mathcal{N}_i} W_{i,j} (\mathbf{x}(i) - \mathbf{x}(j))^2 \right)^{\frac{p}{2}} \tag{2.12}$$

where $\mathcal{N}_i$ are again the neighboring nodes of i. The term $S_p(\mathbf{x})$ was named p-Dirichlet form of $\mathbf{x}$ and measures smoothness on the graph. It should be noted that this is not the only way one can define smoothness and different definitions might fit to different applications. If we choose $p = 1$, we get what [1] calls total variation of the graph signal while [6] also use $p = 1$ but squared weights $W_{i,j}^2$ in their definition of total variation. Another option is to take, instead of the squared differences $(\mathbf{x}(i) - \mathbf{x}(j))^2$, the absolute value $|\mathbf{x}(i) - \mathbf{x}(j)|$. The latter definition has a smaller penalty to big differences and might be beneficial when trying to recover images with edges.

Since this discussion is centered around the Laplacian approach of graph signal processing, we want to focus on the definition given in Equation 2.12, furthermore, set $p = 2$ which enables simpler calculations. To avoid further naming confusion, we will refer to $S_2(\mathbf{x})$ as *global smoothness*.

There is an interesting connection between global smoothness and the Laplacian $\mathbf{L}$ given by

$$S_2(\mathbf{x}) = \frac{1}{2} \sum_{i \in V} \sum_{j \in \mathcal{N}_i} W_{i,j} (\mathbf{x}(i) - \mathbf{x}(j))^2 = \mathbf{x}^H \mathbf{L} \mathbf{x}. \tag{2.13}$$

Using this form, one can easily compute the global smoothness of the eigenvectors $\mathbf{u}_k$ to be

$$S_2(\mathbf{u}_k) = \mathbf{u}_k^H \mathbf{L} \mathbf{u}_k = \mathbf{u}_k^H \lambda_k \mathbf{u}_k = \lambda_k, \tag{2.14}$$

where we used Equation 2.7 and the orthonormality of the eigenvectors $\mathbf{u}_k^H \mathbf{u}_k = 1$. Therefore, if $\lambda_j < \lambda_k \Rightarrow S_2(\mathbf{u}_j) < S_2(\mathbf{u}_k)$ and the value of λ_k, can be interpreted as the

frequency of a certain component – the eigenvector $\mathbf{u}_k$. Given a graph Fourier transform $\hat{\mathbf{x}}$, we can see the components of the vector $\hat{\mathbf{x}}$ that correspond to small λ as the low frequencies and those for big λ as high frequency components.

Having a frequency representation, allows to introduce filtering. In classical DSP, we are able to write the filter operation in the frequency domain as $Y(j\omega) = H(j\omega)X(j\omega)$. Each frequency component of the original signal $X(j\omega)$ is multiplied by a number $H(j\omega)$. Similarly, we can define the graph spectral filtering as

$$\hat{\mathbf{y}}(k) = \hat{\mathbf{h}}(k)\hat{\mathbf{x}}(k) \tag{2.15}$$

whereby, we take each spectral component of the graph signal and multiply it with a filter coefficient $\hat{\mathbf{h}}(k)$. As the Fourier transform is known to be $\hat{\mathbf{x}} = \mathbf{U}^H\mathbf{x}$ and the inverse transfrom is $\mathbf{y} = \mathbf{U}\hat{\mathbf{y}}$, we specify the filtering also in the vertex domain as

$$\mathbf{y} = \mathbf{U}\begin{pmatrix} \hat{\mathbf{h}}(1) & 0 & \\ 0 & \ddots & 0 \\ & 0 & \hat{\mathbf{h}}(N) \end{pmatrix}\mathbf{U}^H\mathbf{x} = \mathbf{H}\mathbf{x} \tag{2.16}$$

with the graph filter matrix $\mathbf{H}$.

One example where a graph filter can be applied is the problem of Tikhonov regularization. Given a noisy graph signal $\mathbf{x} = \mathbf{x}_0 + \mathbf{n}$, where x_0 is the original signal and $\mathbf{n}$ is iid Gaussian noise. To recover the original signal, we want to enforce global smoothness on a chosen graph.

This leads to the minimization problem

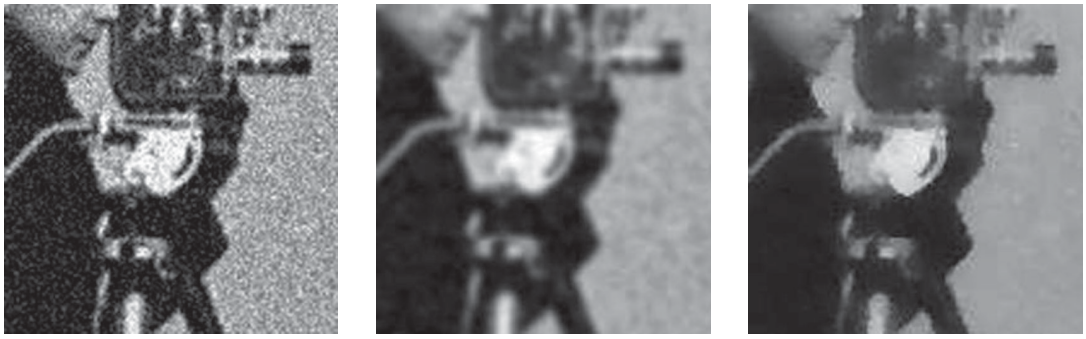
$$\underset{\mathbf{y}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{x}\|_2^2 + \gamma \mathbf{y}^T\mathbf{L}\mathbf{y} \tag{2.17}$$

with a constant $\gamma > 0$ that weights smoothness against close representation.

The optimal solution to this problem can be found in [9] and can be written using the graph filter $\hat{\mathbf{h}}(k) = 1/(1 + \gamma\lambda_k)$. We can, therefore, compute the optimum solution to equation 2.17 as

$$\mathbf{y} = \mathbf{U}\begin{pmatrix} 1/(1+\gamma\lambda_1) & 0 & \\ 0 & \ddots & 0 \\ & 0 & 1/(1+\gamma\lambda_N) \end{pmatrix}\mathbf{U}^H\mathbf{x}. \tag{2.18}$$

If a greyscale image is the graph signal, the graph can be constructed by connecting each pixel node to each neighboring pixel. In Figure 2.5, from [1] denoising performance of the Tikonov method is compared against Gaussian filtering.



(a) noisy image (b) gaussian filtering ($\sigma = 1.5$) (c) graph filtered

Figure 2.5: Detail view of the denoising results from [1]

This short introduction to Laplace-based graph signal processing should give the reader an idea about the approach. Furthermore, a translation operator can be introduced, as well as a concept similar to frequency modulation. Nevertheless, one has to trust the arguments used to introduce the spectrum, and a strong focus is put on the smoothness property. Moreover, the choice of the presented Laplacian is not unique. Instead of the regular $\mathbf{L} = \mathbf{D} - \mathbf{W}$, a normalized Laplacian $\hat{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ can be used.

Additionally, the presented approach limits us to undirected graphs. Although this ensures the existence of a Laplacian and therefore, a *real* well ordered spectrum, it excludes applications where the directivity in the graph is needed. This motivates the following algebraic approach to graph signal processing.

2.2.2 Graph Signals – an Algebraic Approach

Choosing the Laplace matrix as the defining relationship allows to define a real spectrum and also offers an intuitive relationship to a signal's smoothness. Despite these advantages it is relatively weak from an axiomatic view point. Why should the graph Fourier transform be represented by the eigenvectors of the Laplacian? How can we express non-symmetric relationships between nodes? What is the connection to classical signal processing?

To answer these questions, we have to follow the more stringent path to graph signal processing published in [2]. One of the authors - J. M. F. Moura tried to abstract the main principles of signal processing, which are signals, filters, shifts and Fourier transforms. This led to, what is called Algebraic Signal Processing (ASP) introduced in [10], [11]. In the following section, we want to introduce the basic ideas of this concept starting with the ideas of ASP as in [11].

2.2.2.1 Algebraic Signal Processing

The idea is that signal processing consists of an interplay between signals and filters. To extend the framework into a more extensive setting, those interactions need to be

identified. If the requirements to signals and filters are known, those can be applied to introduce the more general signal processing on graphs (DSPG). We present the requirements in the fashion of [11] rather than in a more mathematically precise way to emphasize the intuition behind them.

We consider two sets, the set of signals $\mathcal{M}$ and the set of filters $\mathcal{A}$. The set of signals is, per assumption, a vector space with a set of scalars α. Therefore, the following holds:

$$\text{signal} + \text{signal} = \text{signal} \tag{2.19}$$

$$\text{signal} \cdot \alpha = \text{signal}. \tag{2.20}$$

Two signals can be added to give another signal, and a signal can be scaled with a scalar. We will only consider linear signal processing. We can define the interaction between a signal and a filter, usually known as convolution, as a multiplication

$$\text{filter} \cdot \text{signal} = \text{signal} \tag{2.21}$$

which results in a signal from the signal space $\mathcal{M}$. The linear property expresses itself algebraically in a distributive law given by

$$\text{filter} \cdot (\text{signal} + \text{signal}) = \text{filter} \cdot \text{signal} + \text{filter} \cdot \text{signal}. \tag{2.22}$$

Onto the filter space $\mathcal{A}$ we have the requirements

$$\text{filter} + \text{filter} = \text{filter} \tag{2.23}$$

$$\text{filter} \cdot \alpha = \text{filter}. \tag{2.24}$$

that require $\mathcal{A}$ to be a vector space. The first equation can be seen as a parallel connection of filters and the second operation scales the filter. Additionally, filters have another requirement that makes them different from signals, which is

$$\text{filter} \cdot \text{filter} = \text{filter}. \tag{2.25}$$

This is the algebraic requirement that ensures serial concatenation of two filters. We conclude that the set of filters $\mathcal{A}$ has to be vector space and a ring that shares the same addition which is known to be an *algebra*. The set of signals $\mathcal{M}$ is an $\mathcal{A}$-*module* $\mathcal{M}$. Both constructs are defined more formally in the appendix of [11].

Axiomatically, we want to operate on signals and filters defined in equations 2.20-2.25. In applications, signals are usually defined over a vector space V rather than a module $\mathcal{M}$. To apply filtering operations, we need to map the vector signals from V to the module $\mathcal{M}$. For this, we introduce a bijective linear mapping

$$\Phi : V \to \mathcal{M}. \tag{2.26}$$

As an example we can map infinite time series $\ell(\mathbb{Z})$ to the Z-space:

$$\Phi : V = \ell(\mathbb{Z}) \to \mathcal{M} \tag{2.27}$$

$$\mathbf{s} \mapsto S(z) = \sum_n s_n z^{-n} \tag{2.28}$$

In this example, the mapping Φ is the well known Z-transform. According to [11], the triplet

$$(\mathcal{A}, \mathcal{M}, \Phi) \tag{2.29}$$

is known as *Linear Signal Model* and is sufficient to introduce further signal processing concepts as spectrum and Fourier transform. We will postpone the definition of the Fourier transform so we can introduce it together with the specialized graph signal processing.

There is one more important concept that should be introduced before we progress to graphs. This is the notion of *shift*. We want to assume that every filter in our filter algebra $\mathcal{A}$ can be expressed by only additions and multiplications of the shift filter. In algebraic terms, this means that the chosen shift is a generator for the algebra $\mathcal{A}$. Staying with our previous example, the shift for the 1-D model is the time shift operator $x = z^{-1}$.

A further specialization of the signal model is the requirement for shift invariance. We want the output of shifted and then filtered signal $h(x\, s)$ to be equal to the output from the signal shifted afterwards $x(h\, s)$. Using the filter algebra this can be written as

$$x \cdot h = h \cdot x \quad \forall h \in \mathcal{A}. \tag{2.30}$$

This requires the algebra $\mathcal{A}$ to be commutative. The reverse holds as well – a commutative algebra leads to a shift-invariant model. To obtain N-dimensional filter algebras, we use polynomials modulo a polynomial $p(x)$ of degree N. Without the modulo operation, the product of two filter polynomials could have a higher degree than N and therefore, not be part of the algebra – which would contradict equation 2.25. This theoretical background will be helpful in a more rigorous way of introducing graph signal processing in the next section. A more detailed, lengthy discussion of algebraic signal processing can be found in [10].

2.2.3 Algebraic Graph Signal Processing

In the previous section, we concluded that under certain circumstances a shift operator with a signal space is sufficient to define signal processing basics. We will apply those concepts and mainly follow the lines of [2] to develop graph signal processing.

As in the Laplacian approach we have in mind a graph G with vertices $V(G)$. Each of the vertices is assigned a signal value given by equation 2.1. To capture the connections of the graph, we introduce a weighted adjacency matrix $\mathbf{A}$ similar to the weight matrix $\mathbf{W}$ before. In contrast to the weight matrix the adjacency matrix $\mathbf{A}$ allows for directed connections and self loops and therefore, describes a weighted directed pseudograph as shown in figure 2.2b.

The strategy is to interpret this adjacency matrix as the shift operator in our filter algebra operating on the graph signals $\mathbf{x}$. This means we understand the operation

$$\tilde{\mathbf{x}} = \mathbf{A}\mathbf{x}$$

as a shift of the graph signal $\mathbf{x}$ with respect to the underlying graph G whose connectivity is captured entirely in the adjacency matrix $\mathbf{A}$. A compelling argument why

this is a reasonable thing to do is that we can represent the cyclic shift operation $\tilde{\mathbf{x}}(n) = \mathbf{x}(n-1 \bmod N)$ and the operator z^{-1} by a special matrix $\mathbf{A}$ as

$$\tilde{\mathbf{x}} = \begin{pmatrix} 0 & & & & 1 \\ 1 & 0 & & & \\ 0 & 1 & \ddots & & \\ & & \ddots & 0 & \\ & & 0 & 1 & 0 \end{pmatrix} \mathbf{x} = \mathbf{A}\mathbf{x}. \tag{2.31}$$

The corresponding directed graph is shown in Figure 2.6. This gives graph signal processing with directed graphs an edge over undirected graphs where it is impossible to represent a classical cyclic shift as a special case in the theory.

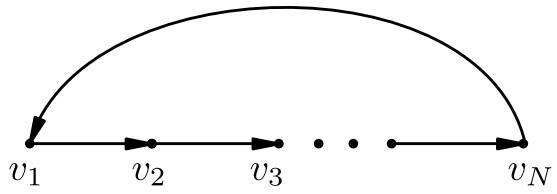


Figure 2.6: Graphical representation of the graph for the graph shift matrix $\mathbf{A}$ given in Equation 2.31

An advantage of graph signal processing is that it implicitly solves the boundary conditions, which had to be addressed with the modulo operation in the classical shift operation.

Unlike in the Section 2.2.2.1, where we defined filter algebra α as an abstract construct, we want to work with the practical objects, i.e. matrices and vectors and afterwards make the connection to the corresponding polynomial algebra. To define graph filters we first start with arbitrary matrices $\mathbf{H} \in \mathbb{C}^{N \times N}$. As matrices, they already fulfill the requirement of linearity when applied to signals as shown in

$$\mathbf{H}(\alpha\mathbf{x}_1 + \beta\mathbf{x}_2) = \alpha\mathbf{H}\mathbf{x}_1 + \beta\mathbf{H}\mathbf{x}_2 \tag{2.32}$$

which holds for all signals $\mathbf{x}$, as for any scalars α and β. As in classical signal processing we wish shift invariance for filters, which can be expressed as

$$\mathbf{A}(\mathbf{H}\mathbf{x}) = \mathbf{H}(\mathbf{A}\mathbf{x}) \tag{2.33}$$

in terms of filter matrices $\mathbf{H}$ and graph shifts $\mathbf{A}$.

To define the structure of graph filters, we need the characteristic polynomial $p_{\mathbf{A}}(x)$ and the minimal polynomial $m_{\mathbf{A}}(x)$, which are defined in Definition 6.1 and Definition 6.2

of the appendix. If for an adjacency matrix $\mathbf{A}$, $p_{\mathbf{A}}(x) = m_{\mathbf{A}}(x)$, then the graph filter $\mathbf{H}$ is shift invariant iff it can be written as matrix polynomial of the form

$$\mathbf{H} = h_0\mathbf{I} + h_1\mathbf{A} + h_2\mathbf{A}^2 + \cdots + h_L\mathbf{A}^L \tag{2.34}$$

using complex coefficients $h_i \in \mathbb{C}$. These coefficients can be seen as taps equivalent to the taps in classical filter responses.

Since we required matrices $\mathbf{A}$ that fulfill $p_{\mathbf{A}}(x) = m_{\mathbf{A}}(x)$, we need a way to deal with graphs G whose weighted adjacency matrix $\mathbf{A}$ does not fulfill this condition. According to [2], for each $\mathbf{A}$, with $p_{\mathbf{A}}(x) \neq m_{\mathbf{A}}(x)$, there exists a function r that connects $\mathbf{A}$ over $\mathbf{A} = r(\tilde{\mathbf{A}})$ with a matrix that complies with $p_{\tilde{\mathbf{A}}}(x) = m_{\tilde{\mathbf{A}}}(x)$. We can then use our graph filters $h(\mathbf{A}) = h(r(\tilde{\mathbf{A}}))$. Therefore, we can replace $\mathbf{A}$ with another graph using the matrix $\tilde{\mathbf{A}}$ that has the same nodes but possibly different edges.

By using a polynomial division onto the filter polynomial $h(x)$

$$h(x) = q(x)m_{\mathbf{A}} + r(x) \tag{2.35}$$

we can split the filter polynomial into a first part that contains the minimal polynomial and a remainder part that has a degree of $\deg r(x) < N$. Applying this to our filter we see that

$$h(\mathbf{A}) = q(\mathbf{A})m_{\mathbf{A}}(\mathbf{A}) + r(\mathbf{A}) = q(\mathbf{A})\mathbf{0} + r(\mathbf{A}) = r(\mathbf{A}). \tag{2.36}$$

Therefore, we can conclude that the maximal filter length of a graph filter is N. With this knowledge, we can define the algebra of graph filters as

$$\mathcal{F} = \left\{ \mathbf{H} \ : \ \mathbf{H} = \sum_{l=0}^{N-1} h_l \mathbf{A}^l \right\}. \tag{2.37}$$

To make the connection to the algebraic model introduced before, we need to define the graph z-transform which is a generalization of the discrete time z-transform. Given a filter algebra $\mathcal{F}$ and its elements $\mathbf{H}$, we can use the mapping $\mathbf{A} \mapsto x$ to map the graph filter which is defined by a polynomial $h(\mathbf{A})$ to a polynomial $p(x)$ of the polynomial algebra

$$\mathcal{A} = \frac{\mathcal{C}[x]}{m_{\mathbf{A}}(x)}. \tag{2.38}$$

This mapping $h(\mathbf{A}) \mapsto h(x)$ is an isomorphism from the filter algebra $\mathcal{F}$ to the polynomial algebra $\mathcal{A}$. The concatenation of filters is represented by the polynomial multiplication modulo $m_{\mathbf{A}}(x)$.

In contrast to classical signal processing, the graph z-transform is different for signals and filters. We can see that the two z-transforms have to be different, just by looking at the different structures of filters and signals. While a signal is represented by a vector $\mathbf{x} \in V \subset \mathbb{C}^n$, the filters are matrices $\mathbf{H} \in \mathcal{F} \subset \mathbb{C}^{N \times N}$.

Similar to the filter algebra $\mathcal{A}$, we find as the z-domain representation of the signal space V to be the $\mathcal{A}$-module $\mathcal{M}$:

$$\mathcal{M} = \mathbb{C}[x]/p_{\mathbf{A}}(x) = \left\{ v(x) = \sum_{i=0}^{N-1} v_n b_n(x) \right\}. \tag{2.39}$$

In Equation 2.39, we already represented the polynomials $v(x) \in \mathcal{M}$ as a sum of basis polynomials. The signal graph z-transform can be defined using those polynomials as the mapping

$$\mathbf{v} = (v_0, \dots, v_{N-1})^T \in V \mapsto v(x) = \sum_{n=0}^{N-1} v_n b_n(x). \tag{2.40}$$

We still have to determine how the basis polynomials can be calculated. According to [2], we need to follow the subsequent procedure. We first need to group all basis polynomials into a vector

$$\mathbf{b}(x) = (b_0(x), \dots, b_{N_1}(x))^T. \tag{2.41}$$

The basis polynomials have to satisfy the condition

$$\mathbf{b}^{(r)}(\lambda_m) = \left(b_0^{(r)}(\lambda_m), \dots, b_{N-1}^{(r)}(\lambda_m)\right)^T = r!\tilde{\mathbf{v}}_{m,0,r} \tag{2.42}$$

for $0 \leq r < R_{m,0}$ and $0 \leq m < M$, with λ_m and $\tilde{\mathbf{v}}_{m,0,r}$ being the generalized eigenvectors of $\mathbf{A}^T$. Here M are the number of distinct eigenvalues, $R_{m,0}$ is the length of the mth Jordan chain and $b_i^{(r)}(x)$ is the rth derivative of $b_i(x)$. Those N equations are enough to find all N^2 coefficients of the basis polynomials.

In the graph z-domain filtering is performed as the multiplication

$$\tilde{v}(x) = \sum_{n=0}^{N-1} \tilde{v}_n b_n(x) = h(x)v(x) \bmod p_{\mathbf{A}}(x). \tag{2.43}$$

In Figure 2.7 the filtering operation is depicted in both domains.

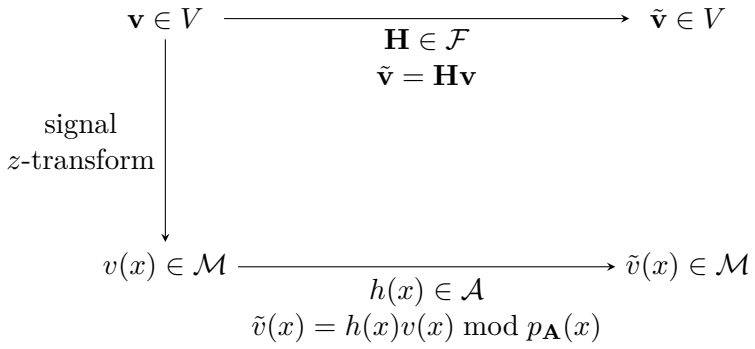


Figure 2.7: Filtering in vector and z-transform domain

By defining the algebra $\mathcal{A}$, the module $\mathcal{M}$ and the z-transform for the case of graph signal processing, we have shown that the latter can be seen as a generalization of classical signal processing. From now on, we will stick to the matrix and vector notation as it is simpler to work directly with the given vectors and shift matrices than with the corresponding polynomials.

2.2.4 Frequency analysis

We want to decompose the signal space V into subspaces V_k that are invariant to filtering, which means that

$$x_k \in V_k \quad \Rightarrow \quad h(\mathbf{A})x_k \in V_k \quad \forall\, h(\mathbf{A}) \in \mathcal{F}. \tag{2.44}$$

This is equivalent to the effect of a filter onto a single frequency which only multiplies it by a scalar $H(e^{j\omega t})$. Given a graph signal $\mathbf{x}$, we want to decompose it into $\mathbf{x}_k \in V_k$ such that

$$\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_{K-1}. \tag{2.45}$$

All those subspaces should decompose the space V in the form

$$V = V_1 \oplus V_2 \oplus \cdots \oplus V_{K-1}. \tag{2.46}$$

Following [2], we identify these subspaces V_k as

$$V_k = \operatorname{span}\left\{\mathbf{v}_{m,d,0}, \ldots, \mathbf{v}_{m,d,R_{m,d}-1}\right\} \tag{2.47}$$

with the generalized eigenvectors $\mathbf{v}_{m,d,r}$, as defined in Definition 6.3. Each subspace is related to the block matrix $\mathbf{V}_{m,d}$. If we use the Jordan decomposition from Definition 6.3 $\mathbf{A} = \mathbf{V}\mathbf{J}\mathbf{V}^{-1}$ we can reformulate graph filtering as

$$h(\mathbf{A}) = \sum_{n=0}^{N-1} h_n \left(\mathbf{V}\mathbf{J}\mathbf{V}^{-1}\right) = \mathbf{V}\left(\sum_{n=0}^{N-1} h_n \mathbf{J}^n\right)\mathbf{V}^{-1} = \mathbf{V}h(\mathbf{J})\mathbf{V}^{-1}. \tag{2.48}$$

This spectral decomposition leads us to the *graph Fourier transform* given by the matrix of generalized eigenvectors $\mathbf{V}$

$$\hat{\mathbf{x}} = \mathbf{V}^{-1}\mathbf{x} \tag{2.49}$$

where $\mathbf{V}^{-1}$ takes the role of a Fourier transform matrix $\mathbf{F} = \mathbf{V}^{-1}$. The inverse Fourier transform is given by

$$\mathbf{x} = \mathbf{V}\hat{\mathbf{x}}. \tag{2.50}$$

Filtering of graph signals can also be used in the spectral domain since

$$\tilde{\mathbf{x}} = h(\mathbf{A})\mathbf{x} = \mathbf{V}h(\mathbf{J})\mathbf{V}^{-1}\mathbf{x} = \mathbf{F}^{-1}h(\mathbf{J})\mathbf{F}\mathbf{x} = \mathbf{F}^{-1}h(\mathbf{J})\hat{\mathbf{x}} \tag{2.51}$$

and therefore, we can describe the filtering operation as

$$\tilde{\hat{\mathbf{x}}} = h(\mathbf{J})\hat{\mathbf{x}}. \tag{2.52}$$

To support the interpretation of the graph Fourier transform we reconsider the graph as depicted in Figure 2.6. This graph has the adjacency matrix as given in Equation 2.31 and its decomposition yields according to [7]

$$\mathbf{A} = \frac{1}{N}\mathbf{D}\mathbf{F}\mathbf{T}_N^{-1}\begin{pmatrix} e^{-j\frac{2\pi 0}{N}} & & \\ & \ddots & \\ & & e^{-j\frac{2\pi(N-1)}{N}} \end{pmatrix}\mathbf{D}\mathbf{F}\mathbf{T}_N \tag{2.53}$$

with $\mathbf{DFT}_N$ being the discrete Fourier transform matrix. Comparing Equation 2.53 with the Jordan decomposition $\mathbf{A} = \mathbf{V}\mathbf{J}\mathbf{V}^{-1}$, we see that in this example $\mathbf{F} = \mathbf{V}^{-1} = \mathbf{DFT}_N$. Therefore, the regular discrete Fourier transform can be seen as a special case of the graph Fourier transform with a cyclic graph as in Figure 2.6.

In the above example, the adjacency matrix $\mathbf{A}$ is diagonalizable, and there we can read the eigenvalues from diagonal matrix $\mathbf{J}$. When talking about graph signal processing it is common to call those eigenvalues λ_n of $\mathbf{A}$ *frequencies* contrary to the classical case where we would call the exponents of the eigenvalues i.e. $\frac{2\pi}{N}n$ frequencies.

As in the Laplacian case, we are again interested in ordering of frequencies – meaning we want to be able to categorize them in low and high frequencies. One advantage of the approach based on the Laplace matrix is that, due to its symmetry, N eigenvectors exist for a $N \times N$ matrix $\mathbf{L}$. By removing the restrictions on the adjacency matrix $\mathbf{A}$, we allow for non diagonalizable matrices. Although we can still define a Fourier transform matrix consisting of the generalized eigenvectors, we want to restrict the further discussion onto diagonalizable matrices.

To define a frequency ordering we will follow the approach of [7] that uses a concept of *total variation*, which is similar to what we called *global smoothness* previously. For classical time signals we can define total variation as

$$\mathrm{TV}(\mathbf{x}) = \sum_{n=0}^{N-1} \left| \mathbf{x}_n - \mathbf{x}_{n-1 \bmod N} \right|. \tag{2.54}$$

This definition compares the absolute difference of consecutive signal values in the signal vector $\mathbf{v}$. A rapidly varying signal is expected to have a high total variation while for a smooth signal, this metric should be small. If we take the adjacency matrix from Equation 2.31 and name it $\mathbf{A}_{\mathrm{cycl}}$, we can write the classical total variation as

$$\mathrm{TV}(\mathbf{x}) = \left\| \mathbf{x} - \mathbf{A}_{\mathrm{cycl}}\mathbf{x} \right\|_1. \tag{2.55}$$

This interprets the total variation as the difference of the vector signal and its (graph) shifted counterpart. Naturally, we can generalize this definition to a *graph total variation* that makes use of the graph shift $\mathbf{A}$. A difference between the classical cyclic shift and the graph shift is the effect on the signals norm $\|\mathbf{v}\|_2$. While the cyclic shift with $|\lambda_n| = 1$ preserves the norm $\|\mathbf{A}_{\mathrm{cycl}}\mathbf{v}\|_2 = \|\mathbf{v}\|_2$, a general graph shift $\mathbf{A}$ can decrease or increase the signal's norm. From the algebraic structure of the filter algebra $\mathcal{F}$, we know that we can scale the filters and also the shift freely. For the following statements about frequency ordering it is convenient to work with a *normalized* shift

$$\mathbf{A}^{\mathrm{norm}} = \frac{1}{|\lambda_{\max}|}\mathbf{A} \tag{2.56}$$

which is scaled by the magnitude of the largest eigenvalue $|\lambda_{\max}|$ of $\mathbf{A}$. This ensures that shifted signals will have a smaller norm because $\|\mathbf{A}^{\mathrm{norm}}\mathbf{v}\|_2 \leq \|\mathbf{v}\|_2$. Finally, we can define the *total variation on graphs* in analogy to Equation 2.55 as

$$\mathrm{TV}_G(\mathbf{x}) = \left\| \mathbf{x} - \mathbf{A}^{\mathrm{norm}}\mathbf{x} \right\|_1. \tag{2.57}$$

We will now investigate the graph total variation of proper eigenvectors $\mathbf{v}$. By applying the total variation definition on $\mathbf{v}$, we obtain

$$\mathrm{TV}_G(\mathbf{v}) = \|\mathbf{v} - \mathbf{A}^{\mathrm{norm}}\mathbf{v}\|_1 = \|\mathbf{v} - \frac{\lambda}{|\lambda_{\max}|}\mathbf{v}\|_1 = \left|1 - \frac{\lambda}{|\lambda_{\max}|}\right| \|\mathbf{v}\|_1. \tag{2.58}$$

If we normalize all eigenvectors to $\|\mathbf{v}\|_1$, the total variation of the eigenvectors only depends on the eigenvalue. For the special case of all real eigenvalues, we obtain the condition

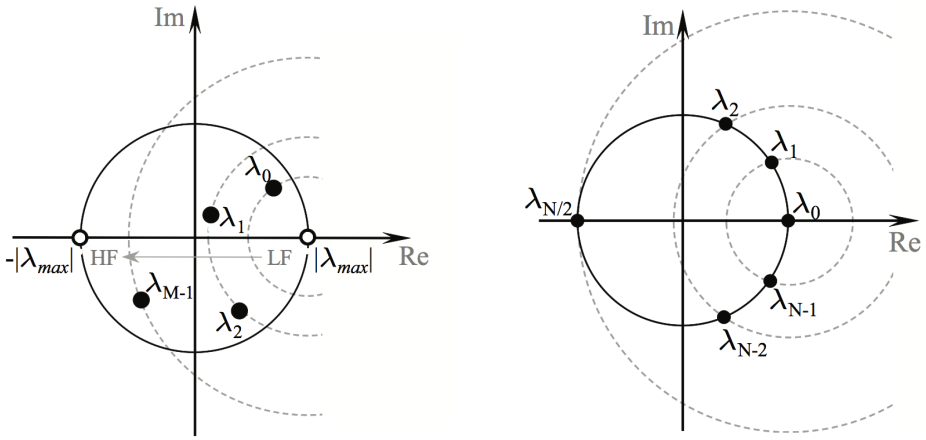
$$\lambda_m < \lambda_n \quad \Rightarrow \quad \mathrm{TV}_G(\mathbf{v}_m) > \mathrm{TV}_G(\mathbf{v}_n), \tag{2.59}$$

which is proven in [7]. For the real eigenvalues a big eigenvalue corresponds to an eigenvector with a low total variation. This is similar to the conclusion we drew from Equation 2.14 for the global smoothness. For complex eigenvalues no natural ordering exists, and therefore, we ask how Equation 2.59 can be extended to this more general case.

From [7] we find that then

$$||\lambda_{\max}| - \lambda_m| < ||\lambda_{\max}| - \lambda_n| \quad \Rightarrow \quad \mathrm{TV}_G(\mathbf{v}_m) < \mathrm{TV}_G(\mathbf{v}_n) \tag{2.60}$$

holds. This condition expresses that the total variation is lower the closer the eigenvalue for the corresponding eigenvector is to the point $|\lambda_{\max}| + 0j$ in the complex plane. This situation is visualized in Figure 2.8a.



(a) graph Fourier transform

(b) classical Fourier transform

Figure 2.8: Frequency ordering depending on the position of the eigenvalues λ in $\mathbb{C}$. Both graphics are from [7]

This can also be compared to the corresponding plot for the classical Fourier transform in Figure 2.8b. As the eigenvalues are as well complex, they have to be compared to eigenvalue λ_0 that represents the DC component of the signal. Conjugated complex eigenvalues correspond to the same frequency and therefore, also share the same total variation.

2.2.5 The importance of the graph shift

Up to now we argued that discrete time signal processing can be generalized to graph signal processing using a shift matrix $\mathbf{A}$. We saw that this generalization fulfills the requirements we derived from classical signal processing. Furthermore, there are equivalents to the z-transform and more importantly to the Fourier transform for graph signals. Under some circumstances the latter also allows for a frequency ordering, that is related to the signal variation on the graph. More advanced topics which build on the graph framework are the closely connected areas of signal sampling and recovery [12][13] or the extension to "big data" as suggested by [14].

Nevertheless, the usefulness of graph signal processing stands or falls with the actual graph information and its interpretation which, in case of the Moura-approach, has to be answered for each application separately, that is packed into the graph shift. One can easily withdraw to the theoretic standpoint and argue that due to similar properties, it is simply a generalization of the 1D-shift, and therefore, a graph shift is "the same" for graphs. Unfortunately, this position is, despite arguably being correct as shown before, not particularly constructive.

It is a common practice to use graph shifts that represent graphs that are directly given by the application, for example a network of connected political blogs, discussed in [7]. There every link represents a directed link in the graph and one entry in the graph shift matrix. Given a graph signal, which can be interpreted as a snapshot of the current political position of each blog, the application of the shift would then correspond to a kind of (political) believe propagation and might be used to predict future changes in the opinion. The same graph can also be used for semi supervised classification, like it was used in [15], where the authors try to induct the political orientation of all blogs from a low number of initially known starting points.

If data is available, but the inherent graph connection between the nodes is not initially visible, we can still want to apply graph signal processing. For this, we need a method to infer the graph first. In the remaining part of this thesis, we want to focus on one proposed method and evaluate its practicality and interpretation.

3 Causal Graph Signals

In the previous chapter, we introduced the concepts of graph signal processing. Until this day, the scientific community that is pushing the graph signal technology shares a limited number of examples ie. the network of US weather stations [14][7][2][12], the Minnesota road graph [1][16] or a blog network [7][15]. This suggests the question why these examples were used by so many authors. First of all, one could argue that many authors work together and share their results on common known graphs to make their results comparable. Another reason could be that these given examples are usually intuitive to understand and there is a kind of shared expectation for the desired outcome. However, there is one more important reason which we want to address in this chapter. That reason is the availability of appropriate graphs that can be associated with graph signals. Even if graphs and corresponding signals are available, there is a computational restriction on the size that can be treated in analysis. For example, the straight-forward application of the graph Fourier transform requires an eigenvalue decomposition of the adjacency/shift matrix as well as computing the inverse of the eigenvector matrix. Until this is addressed the choice of useable graphs is limited by their size, and this leads to the same examples being used repeatedly.

Another approach than applying graph signal processing to learn more about the technique itself, is rooted in the desire to understand and process available *unstructured* data that arises in different applications. This was the motivation for the paper [17] which we want to investigate in this and the following chapters. The main idea is that we obtained a realization of a graph signal process, which contains consecutive snapshots of a graph signal and assumes that the process follows a certain model. Then we can estimate the parameters that make up the model using an algorithm given in [17] which in this case includes an adjacency/shift matrix. Unlike existing methods as for example [3] this method has the advantage that we do not restrict ourselves to symmetric shift matrix and smooth signals. In the following, we will introduce the used model.

3.1 Causal Graph Processes

We start from an observed graph signal

$$\mathbf{x}[k] = (x_0[k], x_1[k], \dots x_{N-1}[k])^T \in \mathbb{C}^N \tag{3.1}$$

on a graph with N nodes that is indexed by a time index k. Furthermore, we want to call the ensemble of $\mathbf{x}[k]$ for all available k a *graph process*. In [17], those graph processes are further specialized to *causal graph processes*. The basic idea behind this class of processes is that the signal $\mathbf{x}[k]$ at a time k depends on the past graph signals $\mathbf{x}[k-1]$, $\mathbf{x}[k-2]$, and so on.

A *Causal Graph Process* is defined as a process of the form

$$\mathbf{x}[k] = \mathbf{z}[k] + \sum_{i=1}^{M} \mathbf{P}_i(\mathbf{A}, \mathbf{c})\mathbf{x}[k-i] \tag{3.2}$$

$$= \mathbf{z}[k] + \sum_{i=1}^{M} \left[\sum_{j=0}^{i} c_{ij}\mathbf{A}^j \right] \mathbf{x}[k-i]. + \tag{3.3}$$

We observe that the current graph signal $\mathbf{x}[k]$ depends on a noise term $\mathbf{z}[k]$ and M past graph signals $\mathbf{x}[k-i]$ which are filtered by the graph filters $\mathbf{P}_i(\mathbf{A}, \mathbf{c})$. In Equation 3.3, the graph filters are seen to be matrix polynomials in a shift matrix $\mathbf{A}$, and therefore, conform with the filters defined in [2]. We will call the parameter M of the graph process *model order* of the process.

To get a better understanding for this rather complex expression we want to show the first terms of the process are

$$\begin{aligned} \mathbf{x}[k] = \mathbf{z}[k] &+ \left[c_{10}\mathbf{I} + c_{11}\mathbf{A}\right] \mathbf{x}[k-1] \\ &+ \left[c_{20}\mathbf{I} + c_{21}\mathbf{A} + c_{22}\mathbf{A}^2\right] \mathbf{x}[k-2] \\ &+ \left[c_{30}\mathbf{I} + c_{31}\mathbf{A} + c_{32}\mathbf{A}^2 + c_{33}\mathbf{A}^3\right] \mathbf{x}[k-3] + \dots \\ &+ \left[c_{M0}\mathbf{I} + \dots + c_{MM}\mathbf{A}^M\right] \mathbf{x}[k-M] \end{aligned} \tag{3.4}$$

where $\mathbf{I}$ is a $N \times N$ identity matrix. To describe a graph process next to the graph shift $\mathbf{A}$ also the filter coefficients c_{ij} have to be known.

According to [17], the full parameterization in Equation 3.4 with all c_{ij} has problems with identifiability. If we assume that $\mathbf{P}_1(\mathbf{A}, \mathbf{c}) \neq \alpha\mathbf{I}$, which is a reasonable assumption since if $\mathbf{P}_1(\mathbf{A}, \mathbf{c}) = \alpha\mathbf{I}$ and also $\mathbf{A} = \alpha'\mathbf{I}$ and therefore, $\mathbf{P}_i(\mathbf{A}, \mathbf{c}) = \alpha_i''\mathbf{I}$ with some α_i''. Then, without loss of generality, we can set $c_{10} = 0$ and $c_{11} = 1$ and therefore, $\mathbf{P}_1(\mathbf{A}, \mathbf{c}) = \mathbf{A}$. Using a full parameterization with the coefficients c'_{ij} and the shift matrix $\mathbf{A}'$ we want to describe the same process with only a shift matrix $\mathbf{A}$ - this yields

$$\mathbf{P}_1(\mathbf{A}', \mathbf{c}') = c'_{10}\mathbf{I} + c'_{11}\mathbf{A}' = \mathbf{A} = \mathbf{P}_1(\mathbf{A}, \mathbf{c}). \tag{3.5}$$

Because we assumed $c'_{11} \neq 0$ we can solve for $\mathbf{A}'$ and obtain

$$\mathbf{A}' = (c'_{11})^{-1}(\mathbf{A} - c'_{10}\mathbf{I}). \tag{3.6}$$

After a lengthy calculation that can be found in [17], we can write the graph filters $\mathbf{P}_i(\mathbf{A}', \mathbf{c}') = \sum_{k=0}^{i} c_{ik}\mathbf{A}^k = \mathbf{P}_i(\mathbf{A}, \mathbf{c})$ with the new coefficients

$$c_{ik} = \sum_{j=k}^{i} c'_{ij}(c'_{11})^{-j} \binom{j}{k} (-c'_{10})^{j-k} \tag{3.7}$$

and can fix $\mathbf{P}_1(\mathbf{A}, \mathbf{c}) = \mathbf{A}$ for further usage. To summarize this technical section, we can represent the causal graph process as

$$\mathbf{x}[k] = \mathbf{z}[k] + \mathbf{A}\mathbf{x}[k-1] + \left[c_{20}\mathbf{I} + c_{21}\mathbf{A} + c_{22}\mathbf{A}^2\right] \mathbf{x}[k-2] + \dots \tag{3.8}$$

with implicitly setting $c_{10} = 0$ and $c_{11} = 1$.

With the process formal being defined in Equation 3.4, or the reduced form in Equation 3.8, we want to develop a better intuition for the model. Unfortunately, the original paper [17] is rather parsimonious with explanations of their model. The authors claim that they are, in some sense, generalizing the idea of a light cone used in the paper [18], to the graph domain. Underlying is the idea that the information that is captured in the nodes' signal values spreads with the constant speed of one *graph shift* per sampling time. This explains why the order of the graph filter grows for samples that are located further in the past. While the previous signal $\mathbf{x}[k-1]$ enters the current signal in a once shifted version $\mathbf{A}\mathbf{x}[k-1]$ the signal $\mathbf{x}[k-i]$ has terms for all shifts $\mathbf{A}^1, \mathbf{A}^2, \ldots$ up to $\mathbf{A}^i$ which results from a i-times shifted signal. Additionally, there is also one term that includes the identity matrix $\mathbf{I}$, which can capture self-loops in case the graph process correlates with itself.

Figure 3.1 shows how the past samples influence the current sample of the causal graph process. Multiple applications of the shift matrix $\mathbf{A}$ ie. $\mathbf{A}^2$, $\mathbf{A}^3$ are represented by more than one hop.

We are using the reduced formulation given by Equation 3.8, which does not include explicit self-loops in the form of an identity matrix for the first term. If a signal strongly depends on the same nodes' values of the previous signal, this dependency has to be modeled into the shift matrix $\mathbf{A}$ as diagonal entries. Therefore, for higher order graph shifts the self loops can either be included through the shift matrix $\mathbf{A}$ or through the identity term $\mathbf{I}$.

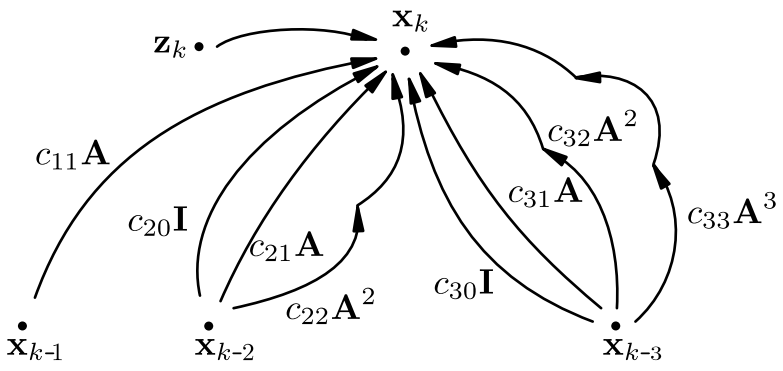


Figure 3.1: Visual representation of a causal graph signal process with model order 3

To show the spreading through the graph shifts Figure 3.2 gives a visualization how the signal value of one node propagates through the graph through multiple graph shifts. Notice that Figure 3.2 shows the matrix polynomial $\mathbf{P}_3(\mathbf{A}, \mathbf{c})$ which is only one term that has to be added and one node is depicted because otherwise the Figure would not be intelligible. Also a simple graph without own loops was chosen to focus upon the

shift effect.

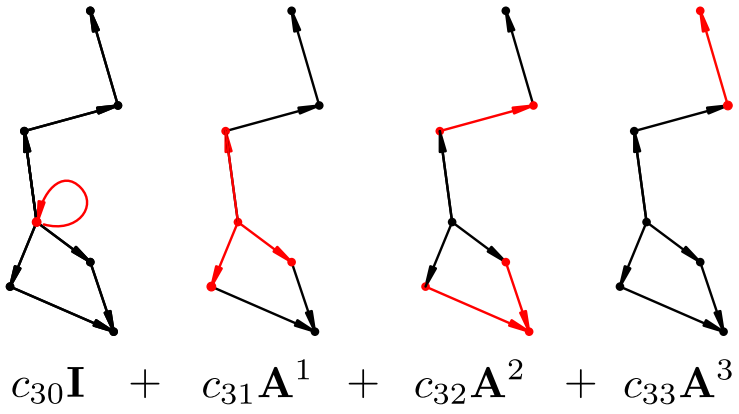


Figure 3.2: Visualization of the information spreading through graph shifts for $\mathbf{P}_3(\mathbf{A}, \mathbf{c})$

The matrix polynomials $\mathbf{P}_i(\mathbf{A}, \mathbf{c})$ contain up to i shift operations. The order of the matrix polynomial and, therefore, also the model order of the process is limited by the graph's number of nodes N. Since $\mathbf{P}_i(\mathbf{A}, \mathbf{c})$ are as well graph filters, they can be represented by a filter with N taps [2] – meaning the highest possible shift is $\mathbf{A}^{N-1}$. If we follow the shifts in Figure 3.2 for the selected node, we see that another shift, i.e. $\mathbf{A}^4$ would not have any effect. This behavior strongly depends upon the structure of the underlying shift/graph and we therefore choose the graph from Figure 2.6 with N cyclically connected nodes. In this case, the signal shifted from any node reaches the same node after N shifts and can therefore be replaced by the shift $\mathbf{A}^0 = \mathbf{I}$. We can clearly see that a graph filter of an order bigger than $N - 1$ cannot be well defined for this example, but also on any other graph. Later, we will see that, when treating the problem of estimation, the model order is usually much smaller than the number of nodes N and this case is not practically relevant.

In this section, we defined the class of causal graph signals. The next section will describe the algorithm presented in [17] which can be used to estimate the shift matrix $\mathbf{A}$ and the graph filter coefficients c_{ij} from a given set of realizations $\mathbf{x}[k]$.

3.2 The algorithm

This section should describe the algorithm used in work that originated from the paper [17]. Finding the shift $\mathbf{A}$ and the coefficients c_{ij} for a process of the form given in Equation 3.8 appears to be non trivial. We assume that K consecutive realizations of the graph signal process $\mathbf{x}_k$ are given, and that the samples $\mathbf{x}[k]$ can be described by the causal graph signal model. In Chapter 4, we will apply the model on real-world data.

The first approach for the estimation task can be formulated in the minimization problem

$$(\mathbf{A}, \mathbf{c}) = \arg\min_{\mathbf{A},\mathbf{c}} \frac{1}{2} \sum_{k=M}^{K-1} \left\| \mathbf{x}[k] - \sum_{i=1}^{M} \mathbf{P}_i(\mathbf{A}, \mathbf{c}) \mathbf{x}[k-i] \right\|_2^2 + \lambda_1 \|\mathrm{vec}(\mathbf{A})\|_1 + \lambda_2 \|\mathbf{c}\|_1. \quad (3.9)$$

In Equation 3.9 we collected all coefficients c_{ij} into the coefficient vector

$$\mathbf{c} = (c_{10}\ c_{11}\ c_{20}\ c_{21}\ c_{22}\ \ldots\ c_{ij}\ \ldots c_{MM}.)^T \quad (3.10)$$

and the vec operator to vectorize the shift matrix $\mathbf{A}$. The minimization varies three terms to find unknown shift $\mathbf{A}$ and all the filter coefficients $\mathbf{c}$. While the first term has the obvious function of minimizing the squared error between the accessible samples of the process and the estimation using the past samples, the two regularization terms need to be discussed.

The term $\lambda_1 \|\mathrm{vec}(\mathbf{A})\|_1$ influences the resulting shift in a way that solutions with many entries are penalized by the l_1-norm. This is a way of forcing the model described by the shift $\mathbf{A}$ to be sparse. A sparse shift relates to a graph with few connections that should ideally help to discover the hidden dependencies in the unstructured data. The constant λ_1 is a positive weighting factor that determines the level of sparsity we require from our solution.

Secondly, we investigate the term $\lambda_2 \|\mathbf{c}\|_1$ which also promotes sparsity but this time in the coefficient vector $\mathbf{c}$. The authors of [17] argue that this works in the way of a model order selection when we choose M to be larger as the causal graph process we try to estimate. We can imagine a process where $\mathbf{P}_i(\mathbf{A}, \mathbf{c}) = \mathbf{0}$ for all i bigger than M'. The regularization on the coefficient vector would then try to null all c_{mj} with $m > M'$. If the model order can be guessed correctly or is known through some genie knowledge, this term is of questionable value. As the first regularization a positive constant λ_2 regulates its influence.

3.2.1 The base algorithm

Unfortunately, Equation 3.9 is very hard to solve directly because it is non convex. This is due to the first term which includes polynomials of the target $\mathbf{A}$. To make this problem solvable by convex optimization, the paper suggests to instead solve a substitute problem. The idea is (instead of directly solving for $\mathbf{A}$) to solve for the resulting matrix polynomials $R_i = \mathbf{P}_i(\mathbf{A}, \mathbf{c})$. This approach leads to the following procedure:

1. Solve the substitute problem to find $\mathbf{R}_i$
2. Using $\mathbf{R}_i$ recover the shift $\mathbf{A}$
3. Estimate the filter coefficient vector c_{ij}

We name the procedure described in the following the *base algorithm* to distinguish it from a simplified version that will be discussed afterwards.

3.2.1.1 Solving for $\mathbf{R}_i$

In the first step we need to solve the minimization problem

$$\hat{\mathbf{R}}_{i\forall i} = \underset{\mathbf{R}_i \forall i}{\arg\min} \frac{1}{2} \sum_{k=M}^{K-1} \left\| \mathbf{x}[k] - \sum_{i=1}^{M} \mathbf{R}_i \mathbf{x}[k-i] \right\|_2^2 + \lambda_1 \|\mathbf{R}_1\|_1 + \lambda_3 \sum_{i \neq j} \|[\mathbf{R}_i, \mathbf{R}_j]\|_F^2. \quad (3.11)$$

The first term in the optimization problem formulated in Equation 3.11, closely resembles the first term of Equation 3.9, assuming $\mathbf{R}_1 = \mathbf{P}_1(\mathbf{A}, \mathbf{c})$. Also the second summand can be related to the original problem under the assumption that we used a reduced representation of the causal process and therefore, $\mathbf{P}_1(\mathbf{A}, \mathbf{c}) = \mathbf{A}$. Lastly, we are left with the last term $\lambda_3 \sum_{i \neq j} \|[\mathbf{R}_i, \mathbf{R}_j]\|_F^2$ which was not present in the original problem. The used model represents the influence of past signals over the graph filters $\mathbf{P}_i(\mathbf{A}, \mathbf{c})$. In our substitute problem, we allow for arbitrary matrices $\mathbf{R}_i$ instead. To restrict those matrices to be more graph-filter-like, we want to require them to commute, which is a basic property of graph filters. This requirement can be enforced by minimizing the term

$$\|[\mathbf{R}_i, \mathbf{R}_j]\|_F^2 = \|\mathbf{R}_i \mathbf{R}_j - \mathbf{R}_j \mathbf{R}_i\|_F^2 \quad (3.12)$$

for all i and j with $i \neq j$. For commutative matrices, the resulting matrix inside the norm would be the $\mathbf{0}$ matrix. In the optimization problem, we desire to achieve a small difference which is measured with the squared Frobenius norm.

Despite the reduced complexity, the problem from Equation 3.11 is still not convex. It is, however, multi-convex [17] which means that if we minimize for one $\mathbf{R}_i$ while holding all other $\mathbf{R}_j$ with $j \neq i$ constant the problem is convex. In one iteration, we have to solve M problems of the form

$$\hat{\mathbf{R}}_i = \underset{\mathbf{R}_i}{\arg\min} \frac{1}{2} \sum_{k=M}^{K-1} \|\mathbf{x}[k] - \sum_{i=1}^{M} \mathbf{R}_i \mathbf{x}[k-i]\|_2^2 + \lambda_1 \|\mathbf{R}_1\|_1 + \lambda_3 \sum_{i \neq j} \|[\mathbf{R}_i, \mathbf{R}_j]\|_F^2. \quad (3.13)$$

which looks almost like Equation 3.11, but the important difference is that we minimize for only one matrix $\mathbf{R}_i$ each while keeping the other $\mathbf{R}_j$ constant. We start with all $\mathbf{R}_i$ set to $\mathbf{I}$ and then estimate for one $\mathbf{R}_i$ at a time, until we have an estimation for all $\mathbf{R}_i$. This can be iterated multiple times until the difference in the estimate matrices is below a preset threshold, and the iteration yields the estimates for the matrix polynomials $\mathbf{P}_i(\mathbf{A}, \mathbf{c})$.

3.2.1.2 Recovering A

The iteration of convex estimation problems yields matrices $\hat{\mathbf{R}}_i$ from which we desire to extract our estimation of the actual shift matrix $\mathbf{A}$. The simplest and most straightforward approach is to the estimate $\hat{\mathbf{A}} = \hat{\mathbf{R}}_1$ that is motivated by the reduced form in Equation 3.8.

This method just uses the $\mathbf{R}_1$ to recover $\mathbf{A}$ while the other $\mathbf{R}_i$ only enter implicitly in the optimization for all matrices. As a second proposal [17], offers another minimization

problem

$$\hat{\mathbf{A}} = \arg\min_{\mathbf{A}} = \|\hat{\mathbf{R}}_1 - \mathbf{A}\|_2^2 + \lambda_1 \|\mathrm{vec}(\mathbf{A})\|_1 + \lambda_3 \sum_{i=2}^{M} \|\left[\mathbf{A}, \hat{\mathbf{R}}_i\right]\|_F^2 \tag{3.14}$$

that explicitly uses all estimated matrices $\hat{\mathbf{R}}_i$. It can be compared to another iteration step of the minimization that finds the $\hat{\mathbf{R}}_i$. The idea is, that since all $\hat{\mathbf{R}}_i$ should be graph filters, they should commute with the graph shift matrix $\mathbf{A}$.

3.2.1.3 Estimate the coefficient vector c

In the last step the base estimation algorithm needs a way to find the coefficients c_{ij}, that are needed for the graph filters. While [17] presented two different methods to achieve this, we will only present one since it produced satisfactory results and didn't appear to be the weak point of the whole procedure.

Again, we formulate an convex optimization problem to find the coefficients for each matrix polynomial by

$$\hat{\mathbf{c}}_i = \arg\min_{\mathbf{c}_i} \frac{1}{2} \|\mathrm{vec}\left(\hat{\mathbf{R}}_i\right) - \mathbf{Q}_i \mathbf{c}_i\|_2^2 + \lambda_2 \|\mathbf{c}_i\|_1 \tag{3.15}$$

with the matrix

$$\mathbf{Q}_i = \left(\mathrm{vec}(\mathbf{I})\ \mathrm{vec}(\mathbf{A})\ \mathrm{vec}(\mathbf{A}^2)\ \ldots\ \mathrm{vec}(\mathbf{A}^i)\right) \tag{3.16}$$

and the estimation for the filter coefficients of $\mathbf{P}_i(\mathbf{A}, \mathbf{c})$

$$\mathbf{c}_i = (c_{i0}\ c_{i1}\ \ldots\ c_{ii})^T . \tag{3.17}$$

Besides requiring the filter coefficients to be chosen in such a way that the resulting graph filters closely reassemble the estimated matrices $\hat{\mathbf{R}}_i$, there is again a regularization term $\lambda_2 \|\mathbf{c}_i\|_1$. This term is the equivalent to $\lambda_2 \|\mathbf{c}\|_1$ in the original formulation of Equation 3.9 and also ensures model order selection, in case we chose a too high M for the process that we are estimating.

This three-step algorithm can produce estimates for the shift $\mathbf{A}$ and the vector $\mathbf{c}$. Due to possible multiple required iterations over the multi-step optimization process of Equation 3.13, the algorithm turns out to be computationally rather complex for large graphs that contain many nodes. Especially the commutativity enforcing term $[\mathbf{R}_i, \mathbf{R}_j]$ requires a considerable amount of computation time. For these reasons, we will also adopt the simplified algorithm of [17].

3.2.2 The simplified algorithm

Especially for problems with big graphs the base algorithm turns out to be slow. By modifying it, we can save computational power while still capturing important features of the base algorithm.

The simplified algorithm shares the same three-step structure with its predecessor. It turns out that the bottle neck of the base algorithm is the first step of the procedure

and there especially the commutativity term causes problems. By dropping this term, we obtain for our first step the minimization problem

$$\hat{\mathbf{R}} = \arg\min_{\mathbf{R}} \sum_{k=M}^{K-1} \|\mathbf{x} - \sum_{i=1}^{M} \mathbf{R}_i \mathbf{x}[k-i]\|_2^2 + \lambda_1 \|\mathrm{vec}(\mathbf{R})\|_1. \tag{3.18}$$

Note that by dropping the last term a global optimization over all $\mathbf{R}_i$ became feasible. We, therefore, not only save the expensive matrix multiplication operation of potentially large matrices, but also skip the iteration of convex optimization problems. Instead we can solve the first step of the now simplified algorithm by one optimization problem. This, of course, comes at the price of not enforcing commutativity and can yield matrices that do not approximately commute and the estimated matrices $\hat{\mathbf{R}}$ are strictly not commutative graph filters.

After this step, $\mathbf{A}$ can either be estimated by $\hat{\mathbf{A}} = \hat{\mathbf{R}}_1$ or the optimization from Equation 3.14. As most of the complexity lies in the first step, both options are viable. Finally, the coefficient vector should be estimated with the problem described in Equation 3.15.

This concludes the description of the simplified algorithm. In Section 3.2.3 we will summarize both algorithms in a compact form.

3.2.3 Summary of the presented Algorithms

The previous section described the two used algorithms to estimate causal graph processes in some detail. For convenience, both algorithms are shown in concise pseudo code form. The first algorithm that is shown, Algorithm 1, is the base estimation algorithm.

Algorithm 1 Base estimation algorithm

Input: $\mathbf{X}$, M, λ_1, λ_2, λ_3, ε
1: Initialize step $t = 0$, $\hat{\mathbf{R}}^{(0)} = \mathbf{0}$, $\hat{\mathbf{R}}^{(-1)} = \infty$
2: **while** $d(\hat{\mathbf{R}}^{(t)}, \hat{\mathbf{R}}^{(t-1)}) > \varepsilon$ **do**
3: **for** $i = 1 : M$ **do**
4: minimize to find $\hat{\mathbf{R}}_i^{(t)}$ with fixed $\hat{\mathbf{R}}_{<i}^{(t)}$, $\hat{\mathbf{R}}_{>i}^{(t-1)}$ by Equation 3.13
5: **end for**
6: $t \leftarrow t + 1$
7: **end while**
8: Set $\mathbf{A} = \hat{\mathbf{R}}_1^{(t)}$ or use Equation 3.14 to obtain $\mathbf{A}$
9: Use Equation 3.15 from $\hat{\mathbf{R}}$ and $\mathbf{A}$
Output: $\mathbf{A}$, $\mathbf{c}$

As input the base algorithm needs:

- $\mathbf{X}$ – the K given samples grouped as $\mathbf{X} = (\mathbf{x}[0], \ \dots \ \mathbf{x}[K-1])$
- M – the model order of the process we try to fit
- λ_1 – larger λ_1 prefer more sparse $\mathbf{A}$

- λ_2 – larger λ_2 prefer more sparse $\mathbf{c}$
- λ_3 – larger λ_3 are more strict on commutativity of $\mathbf{P}_i(\mathbf{A}, \mathbf{c})$
- ε – defines a stopping criterion, a smaller ε leads to slower convergence

The output of the algorithm are the shift matrix $\mathbf{A}$ and the coefficient vector $\mathbf{c}$.

Algorithm 2 Simplified estimation algorithm

Input: $\mathbf{X}$, M, λ_1, λ_2
1: Initialize $\hat{\mathbf{R}} = \mathbf{0}$
2: Estimate $\hat{\mathbf{R}}$ using Equation 3.18
3: Set $\mathbf{A} = \hat{\mathbf{R}}_1^{(t)}$ or use Equation 3.14 to obtain $\mathbf{A}$
4: Use Equation 3.15 from $\hat{\mathbf{R}}$ and $\mathbf{A}$
Output: $\mathbf{A}$, $\mathbf{c}$

The simplified version of the base estimation algorithm is the simplified algorithm that is shown in Algorithm 2. All inputs and outputs that coincide with their counterparts in the base algorithm, have the same interpretation. It should be mentioned that the simplified algorithm simply replaces the while loop with a single optimization.

In the following section, we will discuss the implementation of the algorithm in the scope of this work. Further on, numerical examples try to show the workings of the procedure and the effects of the input parameters on the output.

3.3 Numerical Experiments

With all parameters and algorithms defined, we can now proceed to an investigation of the model and algorithm introduced in [17]. Both Algorithm 1 and Algorithm 2 require solving multiple convex optimization problems. For this task, the MATLAB [19] environment and programming langue was used. To solve the convex optimization problems, we used a third party MATLAB toolbox, called CVX, in which one can specify and solve convex programs [20], [21]. CVX was chosen because it is widespread and specifying the problem was very intuitive. The actual convex solver which was used, was SDPT3 [22], [23] which is the default solver that is shipped with CVX and is freely obtainable under a non commercial GNU GPL v2 license.

In this section, we want to evaluate the performance of the estimation algorithm based on several examples. For this purpose, we will assume that the model for the generation of the graph signal samples $\mathbf{x}[k]$ is known and those signal samples are grouped into the matrix $\mathbf{X}$. This is a first test of the viability of the presented algorithm. The purpose is to better understand the used signal model and the parameter values that are needed to accurately recover the generating process' parameters the shift matrix $\mathbf{A}$ and the filter coefficients c_{ij}. Even though, this might seem redundant at a first glance, it is necessary to choose the algorithms parameters correctly when estimating the model's parameters for data, where the underlying procedures are not known. It is also beneficial for the

interpretation when applying the algorithm to real data, which is shown in Chapter 4. We want to start our discussion with a simple process.

3.3.1 A first simple example

In our first example, we want to investigate the recovery performance of the algorithm for a small dimensional problem of low complexity. For this purpose, we pick the $N = 5$ case of Figure 2.6, which is redrawn for clarity with the chosen number of nodes in Figure 3.3.

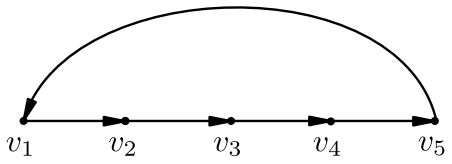


Figure 3.3: A simple Graph used to test the algorithm

Figure 3.3 can be represented by the unweighted *directed* graph shift matrix

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{3.19}$$

We can see that each connection has the same weight which is set to 1. This choice allows the signal energy to remain constant under the shifting process. Therefore, we do not need additional tinkering regarding the shift matrix. This is a special case, and we will see which problems arise if this, as usual, not the case. Of course, specifying the shift matrix is not sufficient to specify a causal graph process.

Our graph process will be the simplest possible – which is a process of order $M = 1$. The causal graph process of equation 3.8 reduces to

$$\mathbf{x}[k] = \mathbf{z}[k] + \mathbf{A}\mathbf{x}[k-1]. \tag{3.20}$$

The current signal consists of the previous sample and a noise vector which we pick as iid $\mathcal{N}(0, \sigma^2)$ distributed. Besides interpreting Equation 3.20 as a causal graph signal process associated with the shift depicted in Figure 3.3, it can also be seen as a Vector Autoregressive Process (VAR) of order 1. Note that even this simple example of directed nature would not be accessible to a Laplacian analysis.

For the numerical example, we generated the first signal sample randomly from five values between 0 and 1. The remaining samples we produced along Equation 3.20 with a given standard deviation σ of the noise. This had to be iterated until we obtained

K samples of the causal graph process which were grouped into a matrix $\mathbf{X}$. The filter coefficients are given by $c_{10} = 0$ and $c_{11} = 1$, which is necessary, if we want to conform to the reduced model presented in Equation 3.8.

The natural test of the algorithm is to recover the underlying shift matrix $\mathbf{A}$. As our model order is $M = 1$ there is only one matrix $\mathbf{R}_1$ to be estimated, which by default, "commutes" with itself. Therefore, it is unnecessary to additionally enforce this property by the use of the base estimation algorithm which is Algorithm 1. We can safely choose the simplified Algorithm 2. Since we have no need to estimate coefficients, the only parameter needed is λ_1 which controls the sparsity of $\mathbf{A}$. We set $\lambda_1 = 0.1$ to obtain Figure 3.4 for a causal graph signal of length $K = 20$.

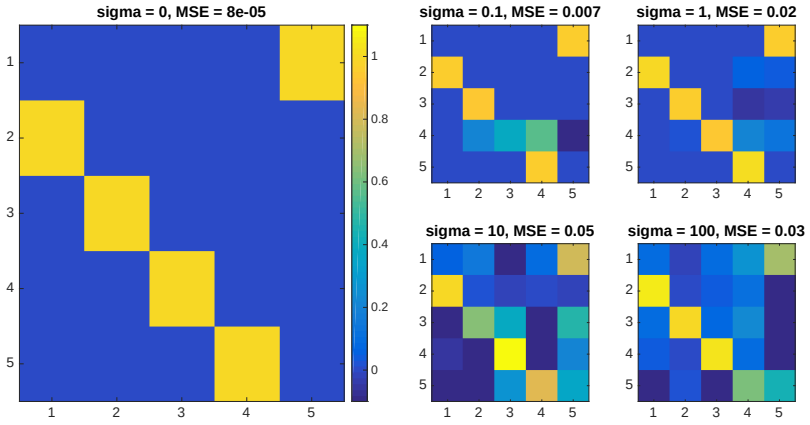


Figure 3.4: Recovery of $\mathbf{A}$ used in the model defined by Equation 3.20 for different noise standard deviations σ, process length $K = 20$ using $\lambda_1 = 0.1$

In Figure 3.4 the recovery results for different levels of noise are shown. The $\sigma = 0$ case without noise works so well, that we can use it as a reference result. To give the results a quantitive dimension, we introduced a MSE metric. This metric is defined as

$$\mathrm{MSE}(\mathbf{A}, \mathbf{A}_{\mathrm{est}}) = \frac{1}{N^2}\|\mathbf{A} - \mathbf{A}_{\mathrm{est}}\|_{\mathrm{frob}}^2 = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left|(\mathbf{A} - \mathbf{A}_{\mathrm{est}})_{i,j}\right|^2. \tag{3.21}$$

While the recovered results always show one exemplary trial of the recovery process, we ran the algorithm 20 times at each noise level to calculate an average MSE. While the MSE might be problematic when used a general-purpose metric, it underlines the qualitative results one can perceive visually.

It can be seen that the performance at higher levels of noise the process gradually decreases. Intuitively we would expect a steeper decline in the estimation performance, but we need to keep in mind that the additive noise in some sense, drives the process. Meaning a stronger noise also results in larger additional terms from past signal samples, which compete with the randomness in the current sample.

Furthermore, it should be said that for the noiseless case, we have to see the process with initial random values since otherwise, the signal would be $\mathbf{0}$ for all samples, and no structure could be estimated from that. This emphasizes the double-edged sword character of the noise in the causal graph process.

Another feature of causal graph processes that already shows up in this simple example is visible in Figure 3.5. In both plots the absolute values of the graph signal samples $\mathbf{x}[k]$ are shown over time for the process discussed in Equation 3.20. The upper part of the plot shows, the noiseless case with $\sigma = 0$ that is only seeded an initial random signal. It can be seen clearly that during the absence noise, those initial samples are just rotated through the graph. The same rotation is also present in the second plot, but it is superimposed with a noise of standard deviation of $\sigma = 100$. Since the chosen shift matrix $\mathbf{A}$ preserves the signal's energy and has no dampening effect, the energy added by the noise ramps up after several samples. We have to keep track of this effect, especially for more complicated examples with a higher model order M.

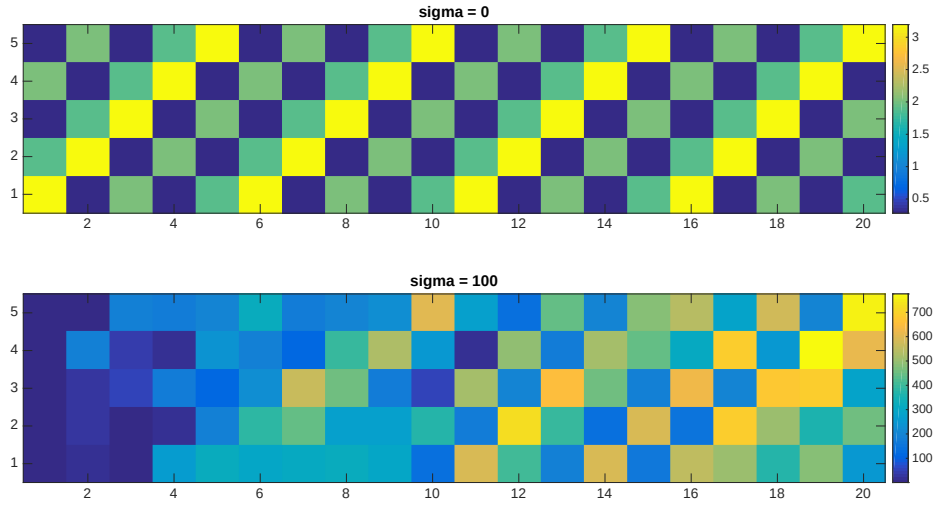


Figure 3.5: Absolute value of the graph Process $|\mathbf{X}|$ with and without noise, process length $K = 20$

To show the effect of a shift that does not keep the energy of the shifted signal constant we scaled the shift from Figure 3.3 down by a factor 0.8 and used this scaled shift

$$\mathbf{A}' = 0.8\mathbf{A} \tag{3.22}$$

while keeping the rest of the model unchanged. We first give the results for the recovered $\mathbf{A}'$ in Figure 3.6. The performance compared with the simulation results showed in Figure 3.4, decreased qualitatively as well as quantitatively in the MSE. This can be explained by the lesser amount of signal that is contained in the current sample as the shifted past value is now damped by 20%.

The working of the dampening shift is also visible in the plot of the signal in Figure 3.7. For the no-noise case, the initial seed signal quickly dies out over the samples.

Nevertheless, the shift structure is still clearly visible and also the result in terms of MSE is still good. In the case with $\sigma = 100$, the dampening process is not that obvious since the strong noise keeps driving the process. By comparing Figure 3.7 and Figure 3.5, we can see that the scaled shift matrix compensated the growth of the signals magnitudes.

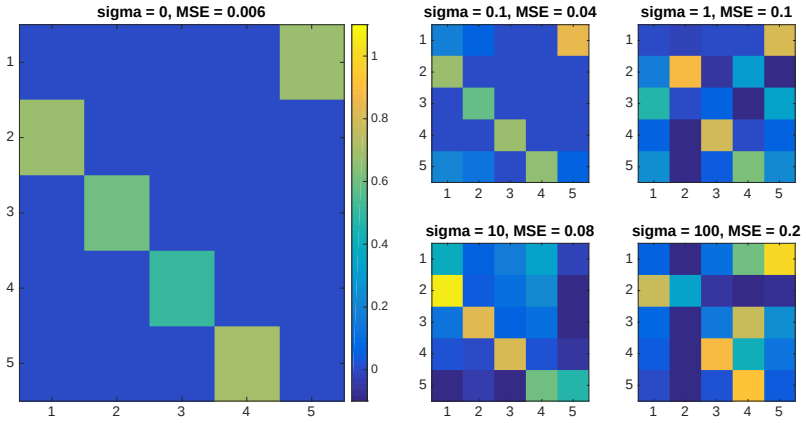


Figure 3.6: Recovery of the dampened $\mathbf{A}'$ for different noise standard deviations σ, process length $K = 20$ using $\lambda_1 = 0.1$

When we want to generate test signals for causal graph processes of higher order or simply graph shifts with more complicated adjacency matrices, we need a more precise way to deal with stability. We need a criterium on how to check, if the chosen shift $\mathbf{A}$ and the graph filters $\mathbf{P}_i(\mathbf{A}, \mathbf{c})$, lead to stable processes. We cannot use processes that "explode" after a few samples. For this, we need to realize that a causal graph processes of model order M can be interpreted as vector autoregressive process of order M – short a VAR(M) process. For this, we need to go back to Equation 3.3 and rewrite it using $\mathbf{P}_i(\mathbf{A}, \mathbf{c}) = \mathbf{R}_i$ as

$$\mathbf{x}[k] = \mathbf{R}_1\mathbf{x}[k-1] + \mathbf{R}_2\mathbf{x}[k-2] + \cdots + \mathbf{R}_M\mathbf{x}[k-M] + \mathbf{z}[k]. \tag{3.23}$$

From [24] we know that a VAR(M) process like the one presented in Equation 3.23 is stable if

$$\det\left(\mathbf{I} - \mathbf{R}_1 x - \cdots - \mathbf{R}_M x^M\right) \neq 0 \quad \forall\, |x| \leq 0. \tag{3.24}$$

Since the causal graph processes are special cases of the VAR model, this criterion, can be used to check if a potential graph process is stable. With the stability criterion in Equation 3.24, we can analyze our simple model process by setting $\mathbf{R}_1 = \mathbf{A}$. Using MATLAB's symbolic math toolbox, we can solve the equation $\det(\mathbf{I} - \mathbf{A}x) = 0$ and visualize the poles in a complex plot shown in Figure 3.8. The regular shift $\mathbf{A}$ aggregates the energy that is introduced to the process in form of the noise and is therefore, not stable with poles on the complex unit circle. By having all poles with magnitude 1, it is the limiting case and all scaling factors $\alpha < 1$ will lead to stable processes independent of the noise power introduced into the process.

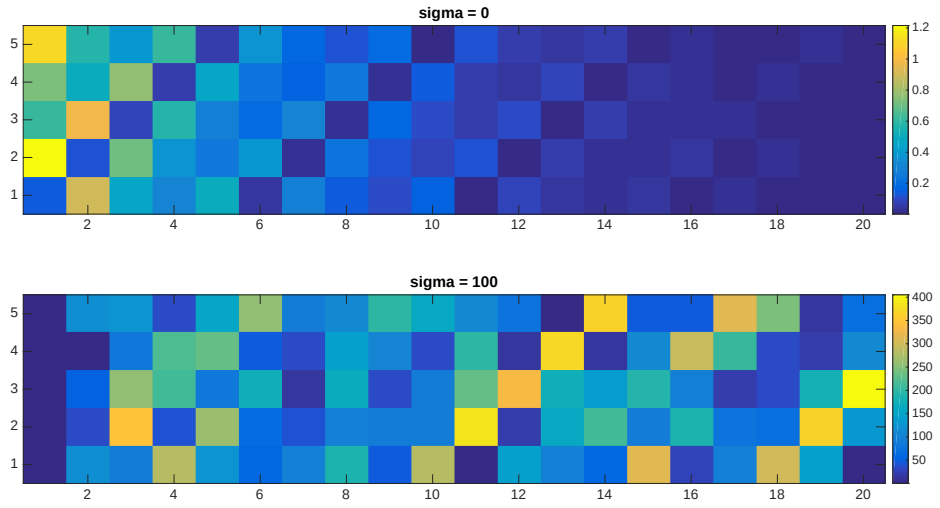


Figure 3.7: Absolute value of the graph Process $|\mathbf{X}|$ for the dampened matrix $\mathbf{A}'$ with and without noise, process length $K = 20$

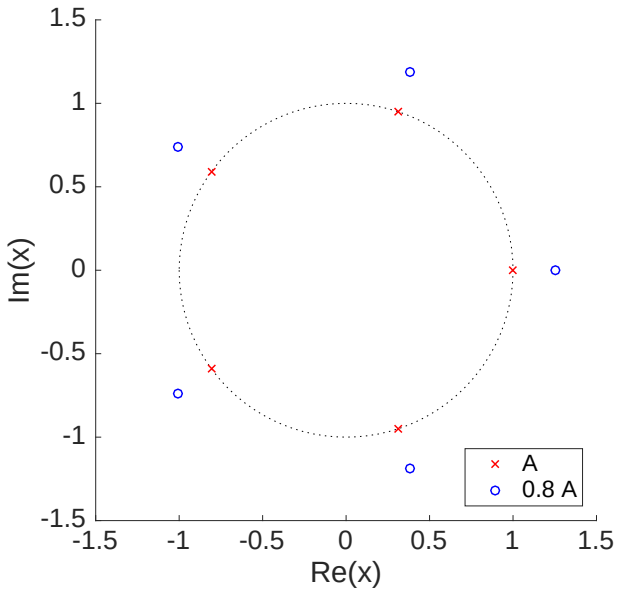


Figure 3.8: Poles of the stability determining polynomial for $\mathbf{A}$ and $0.8\mathbf{A}$, if all poles lie outside of the unit circle the process is stable

3.3.2 More general examples

For the further numerical analysis, we want to switch to less general shift matrix $\mathbf{A}_{m1}$, which is shown in Figure 3.9. We obtained this 8×8 matrix by randomly selecting 20 entries of the matrix and set them to numbers selected from a Gaussian distribution.

Afterwards, we clipped a few entries with big magnitudes and scaled the matrix down to make the causal graph process of order $M = 1$ stable by satisfying Equation 3.24.

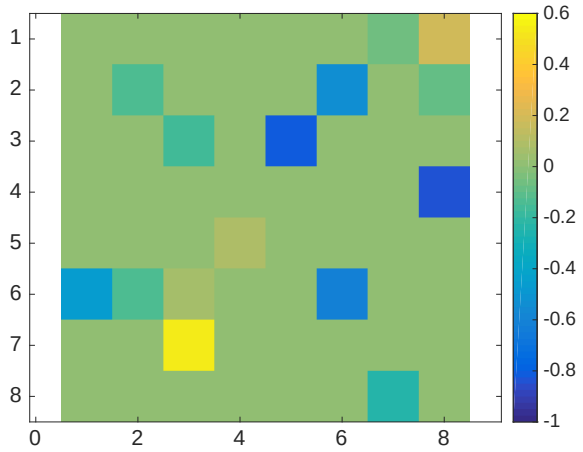


Figure 3.9: A slightly more complicated shift matrix $\mathbf{A}_{m1}$ that was generated randomly with 20 non zero entries, if used for $M = 1$ causal graph process, it will be stable

Until now, we always investigated a process with model order $M = 1$ and tried to estimate a process of the same order for a regular shift matrix. The authors of [17] claim that the Algorithms 1 and 2 implicitly perform model order selection. To test this claim, we used the matrix $\mathbf{A}_{m1}$ shown in Figure 3.9 in the Equation 3.20.

While the process is of order $M = 1$, we tried to estimate the intermediate graph shift matrices $\mathbf{R}_i$ under the assumption that a process of order $M = 1, 2, 3$ is present. We expect that the estimation yields the shift matrix $\mathbf{R}_1 = \mathbf{A}_{m1}$ and sets the other matrices $\mathbf{R}_{2,3}$ to $\mathbf{0}$. Figure 3.10 shows the result of the simulation. In the first column, we always present the ground truth $\mathbf{A}_{m1}$ as a reference. We observe that the matrices $\mathbf{R}_1$ for all three trials closely resemble the shift matrix both qualitatively but also quantitatively in terms of the MSE defined in Equation 3.21. For the matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ we calculated the MSE with a reference to the expected all zero matrix $\mathbf{0}$. From the results, we conclude that the algorithm is robust enough to find the right intermediate results for a simple graph signal process.

In a next step, we chose the simplest estimation to find $\mathbf{A}_{\text{est}}$ which is to choose $\mathbf{A}_{\text{est}} = \mathbf{R}_1$ because the MSE shows already good agreement. We then used the third step of the algorithm which is to estimate the coefficients c_{ij}. Under the assumption that our causal graph signal model is of the form given in Equation 3.8, we selected $c_{10} = 0$ and $c_{11} = 1$. This left only the coefficients of higher orders for estimation. We expected all of them to be close to 0 and indeed for both cases the magnitude of every coefficient was smaller than 10^{-7}. Therefore, in this example, the algorithm could recover both the shift matrix and the coefficients to a high degree of accuracy. For the estimation of

the shift matrices, we again used the simplified Algorithm 2, as both yield very similar results and the base estimation algorithm takes more time. To achieve this precision, a process with $K = 500$ graph signal samples had to be observed.

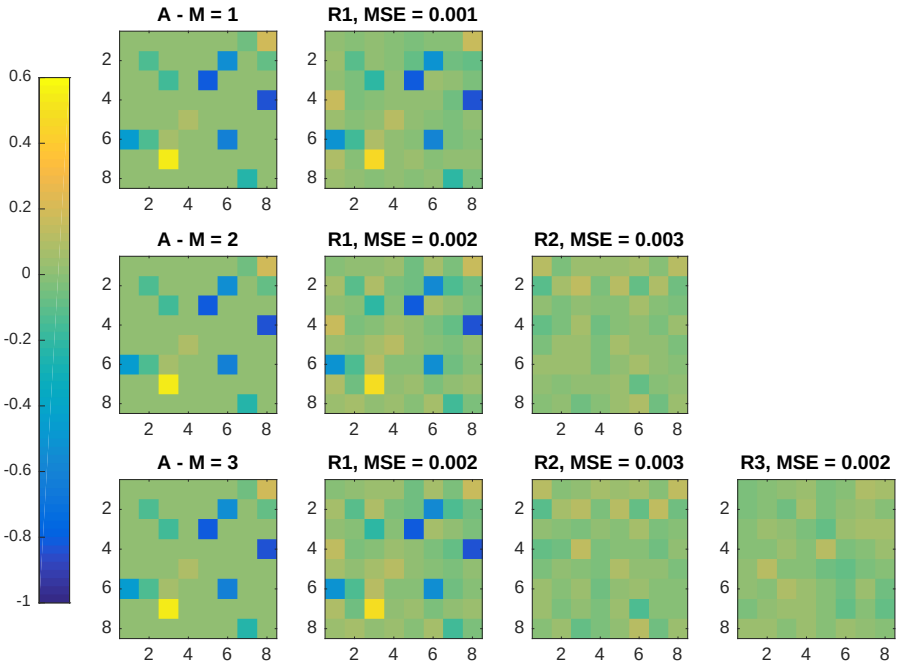


Figure 3.10: Results for the first step of the algorithm for generating a process of order $M = 1$ from a process of length $K = 500$, $\sigma = 1$ with $\lambda_1 = 0.1$

We want to demonstrate that the estimation algorithm also works for processes with order $M = 2$. For this, we reuse the shift matrix $\mathbf{A}_{m1}$ shown in Figure 3.9, but this time with two graph filters. As a generating causal graph process we use equation

$$\mathbf{x}[k] = \mathbf{z}[k] + \mathbf{A}_{m1}\mathbf{x}[k-1] + \left[c_{20}\mathbf{I} + c_{21}\mathbf{A}_{m1} + c_{22}\mathbf{A}_{m1}^2\right]\mathbf{x}[k-2] \tag{3.25}$$

with the coefficients $c_{20} = 0.3$, $c_{21} = 0.5$ and $c_{22} = 0.5$. To ensure stability of the process, we checked that the process is stable according to Equation 3.24. Since this process is more complicated, we used $K = 1000$ samples for the estimation results. Figure 3.11 shows the resulting filter matrices $\mathbf{R}_1$ and $\mathbf{R}_2$. Both matrices are very similar to the ground truths $\mathbf{R}_1 = \mathbf{A}_{m1}$ and $\mathbf{R}_2 = c_{20}\mathbf{I} + c_{21}\mathbf{A}_{m1} + c_{22}\mathbf{A}_{m1}^2$. Despite these good results the filter coefficients are estimated to $\hat{c}_{20} = 0.29$, $\hat{c}_{21} = 0.42$ and $\hat{c}_{22} = 0.2$ which shows that this estimation is troublesome even with accurately estimated graph filter matrices. The coefficient estimation captures if the model order is too high compared to the real process and sets the higher-order coefficients to 0, but it gives only a rough estimate for the real coefficients.

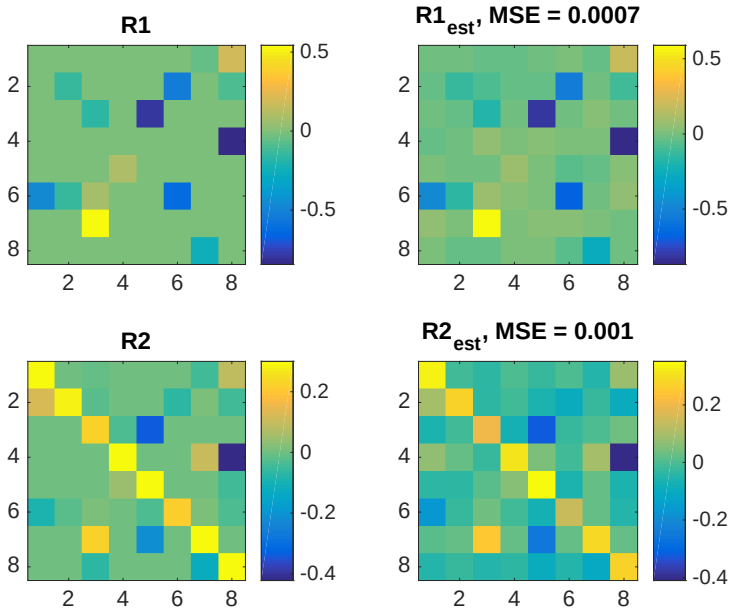


Figure 3.11: Results of the matrix estimation using the simplified algorithm to find $\mathbf{R}_i$ for the process described in Equation 3.3.2 $K = 1000$, $\sigma = 1$ with $\lambda_1 = 0.1$

Another point that needs to be discussed, is the choice of the simplified algorithm 2 over the more sophisticated base algorithm 1. We ran several estimations of the discussed process with different parameters using both algorithms. While the performance of both algorithms is relatively similar, the base estimation is about 3 times slower for this small example. This computation advantage becomes even bigger if we turn to problems with larger N and longer sequence lengths K, where the base estimation algorithm turns infeasible. The results of both simulations are shown in Table 3.1. We see that

	base Algorithm 1	simplified Algorithm 2
$\text{MSE}(\mathbf{R}_1, \mathbf{R}_{1,\text{est}})$	0.003	**0.0007**
$\text{MSE}(\mathbf{R}_2, \mathbf{R}_{2,\text{est}})$	0.003	**0.001**
$\|[\mathbf{R}_{1,\text{est}}, \mathbf{R}_{2,\text{est}}]\|^2_{\text{fro}}$	0.045	**0.042**
c_{2i}	$[0.21, 0.22, 0]^T$	$[\mathbf{0.29}, \mathbf{0.42}, \mathbf{0.2}]^\mathbf{T}$
simulation time	$\approx 90\,s$	$\approx \mathbf{30\,s}$

Table 3.1: Performance comparison between the simplified algorithm and the base algorithm for a process of order $M = 2$, $\mathbf{A}_{m1}$ and $c_{2i} = [0.3, 0.5, 0.5]^T$ with the parameters: $N = 8$, $K = 1000$, $\lambda_1 = 0.1$, $\lambda_2 = 0.1$, $\lambda_3 = 0.1$, maxiter $= 5$, $\varepsilon = 0.01$

the simplified algorithm is more accurate, faster, leads to superior estimates of the coefficients. It even yields matrices that resemble commutative filters better, with smaller

$\|[\mathbf{R}_{1,\text{est}}, \mathbf{R}_{2,\text{est}}]\|_{\text{fro}}^2$ than the base estimation, that incorporates this term explicitly in its minimization process. We conclude that for causal graph processes with small filter orders the simplified estimation algorithm is a viable replacement for the more complex base algorithm.

In the previous examples, we always chose a fixed set of parameters to do the performance comparison. Even when using the simplified algorithm, we can besides the model order M also vary the two sparsity parameters λ_1 and λ_2. While λ_1 controls the sparsity of the shift matrix $\mathbf{A}$, the second λ_2 influences how sparse the solution of the coefficient vector should be. As an example, we will keep $\lambda_2 = 0.1$ constant and change the first sparsity λ_1. To measure the performance, we employ the same MSE as in Equation 3.21 for the $\mathbf{R}_i$ and use the squared 2-norm of the differences for the coefficient vector containing c_{2i}.

The results for the filter matrices are shown in Figure 3.12. While the MSE seems to decrease for larger values of λ_1, the difference is not as dramatic and the MSE stays in around 10^{-3} for all simulated λ_1s.

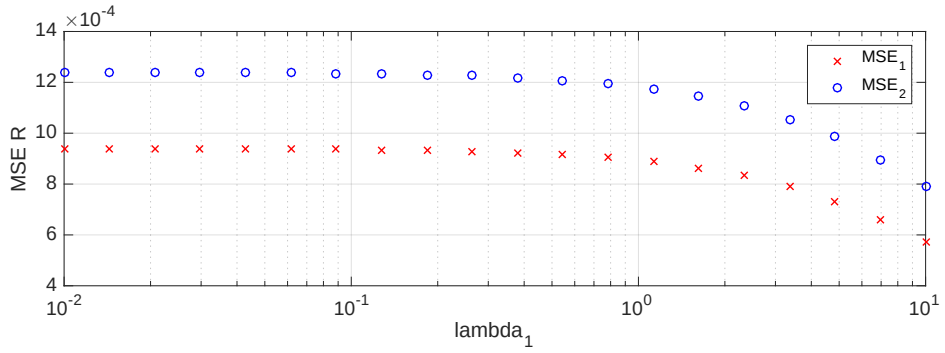


Figure 3.12: MSE_i between the estimate and actual $\mathbf{R}_i$s

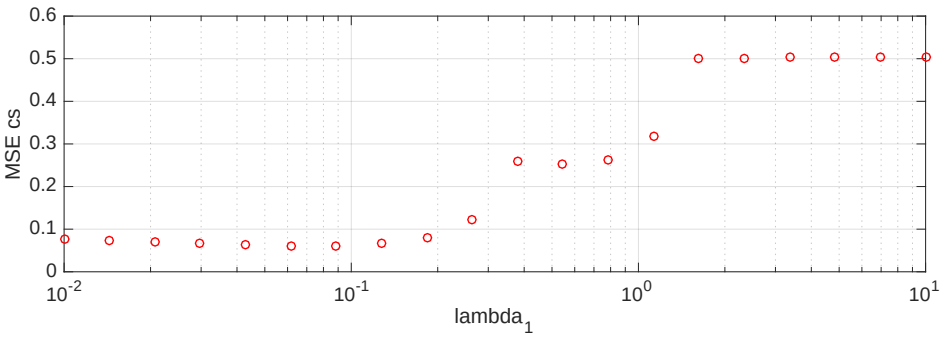


Figure 3.13: MSE_i ($\|\cdot\|_2^2$) between the estimate and actual coefficient vector c_{2i}

When we have a look at the MSE of the coefficient vector in Figure 3.13 the differences appear more drastic. In contrast to the error in the filter matrices a small λ_1 leads to a more accurate filter vector. Specifically, the figure shows a step like behavior.

By inspecting the values of c_{2i} before and after the jumps, it becomes clear that the optimization algorithm yields nonzero components for all c_{2i} as long as lambda is low. After the first step, it sets c_{22} to zero and for even larger λ_1 also $c_{21} = 0$.

From these plots, it appears that our choice for λ_1 achieves relatively good approximations for the coefficient vector with a reasonable MSE in the filter matrices. Furthermore, we encounter a major problem of the algorithm – the question of how to find the right λ_1 and λ_2? As long as we have known the ground-truth this can be answered by running many exhaustive simulations but in real-world problems, we cannot use this method, and we need to use another way to deal on this issue for those kinds of problems.

Before moving to the real data examples, we want to address a point that we touched several times. We need to discuss the effect of the process length K on the estimation quality. Generally, we assumed that for bigger K the estimation performance improves, which is the intuition one has when running multiple simulations. To confirm this expectation, we ran the estimation algorithm for the process discussed in Equation with the parameters $\lambda_1 = \lambda_2 = 0.1$ and $M = 2$.

While varying K we again looked at the same MSEs we already addressed in Figure 3.12,3.13. The results are depicted in Figure 3.14 for the matrices $\mathbf{R}_i$ and for the coefficient vector c_{2i} in Figure 3.15.

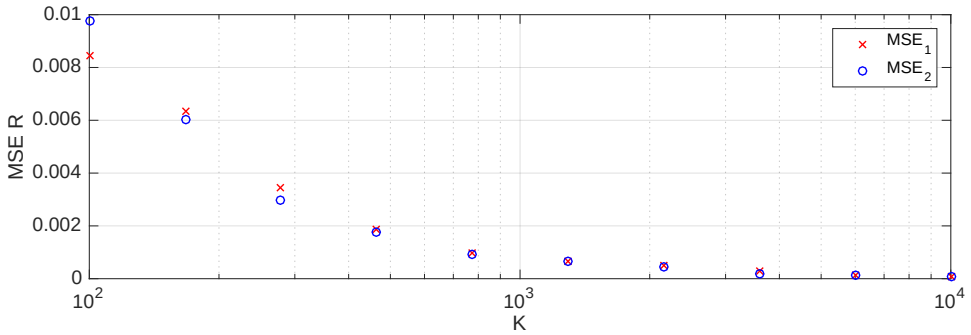


Figure 3.14: MSE_i between the estimate and actual $\mathbf{R}_i$s for different process lengths K

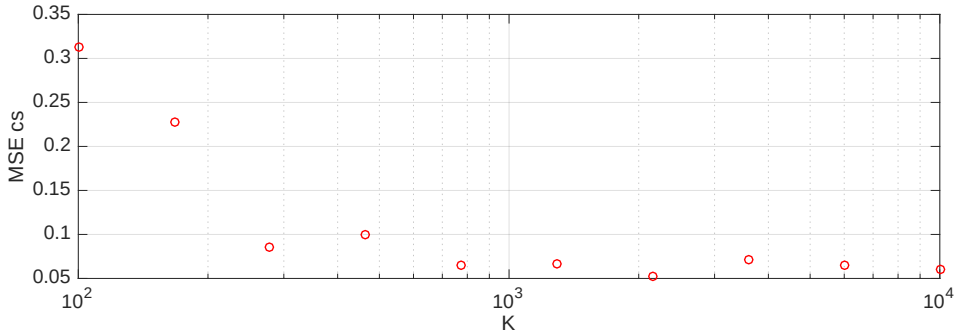


Figure 3.15: MSE_i between the estimate and actual $\mathbf{R}_i$s for different process lengths K

Both figures are more accurate the more samples are included, but this comes at the price of long computation times. To give an example, simulations with $K = 100$ take ≈ 5 seconds, while those with $K = 10000$ take around 15 minutes. Clearly, especially for simulations with bigger graphs, this becomes an ever more important issue and a tradeoff between better estimation and simulation effort is needed.

Throughout this chapter, we discussed the Algorithms 1 and 2. We concluded that the simplified algorithm works well enough for problems with a low model order M and is much more efficient. The overall estimation of artificially generated causal graph processes works well especially in terms of the shift matrix $\mathbf{A}$ but worse for finding the accurate coefficient vector $\mathbf{c}$. This means, we can capture the effects between nodes well, but we might estimate differing graph filters. We also showed that the choice of the parameters λ_1 and λ_2 has a non negligible influence on the simulation results. Unfortunately, there is no clear rule how to choose them in advance. If the sparsity of the solution is known, we can adapt the parameters to make use of this information. Overall, these parameters turn out to be the weak point of the model. In the next chapter, we want to apply the simplified algorithm to two real-world examples.

4 Experiments with Real World Data

In the previous chapter, we evaluated the algorithm proposed in [17] for artificially generated data. We analyzed its performance regarding the recovery of shift matrices $\mathbf{A}$ and filter coefficients $\mathbf{c}$ from generated causal graph processes. Despite difficulties that arise in the choice of parameters, the shift matrices could be well recovered in general. These results motivated us to apply the same procedure onto real-world data.

It turns out that finding suitable data, which can be fed into the estimation algorithm, is challenging. We want to find groups of time series that are, in some sense, dependent and or influencing each other. Additionally, the algorithm's complexity imposes a soft limit to the size of the data we can use to learn the shift matrix in the length K as well as in the number of nodes N. Besides the upper limit due to complexity, the number of nodes should be not too small to deduce some non trivial conclusions. Last but not least, the data should be openly available in a reasonable format to facilitate further processing.

In the scope of this thesis, we found and analyzed two data sets, which fulfill all stated requirements. As a first example, we chose data from temperature sensors which we obtained from the European Climate Assessment Dataset [25]. The ECAD homepage[1] allows downloading historic weather data for specific countries. For our analysis, we focused on the daily mean temperature data.

While the interconnection between weather stations that are distributed over a country allows some hand-waving interpretation of the domain, the second example is more abstract. Instead of weather data, we tried to estimate the dependencies of daily stock prices listed on the Austrian stock exchange.

Both examples intend to showcase the usefulness of the algorithm to real-world applications. Without a clear ground truth, validation is difficult. It should be seen as a first approach to the problem rather than a comprehensive study of all dependencies and interrelations that might be hidden behind the real data.

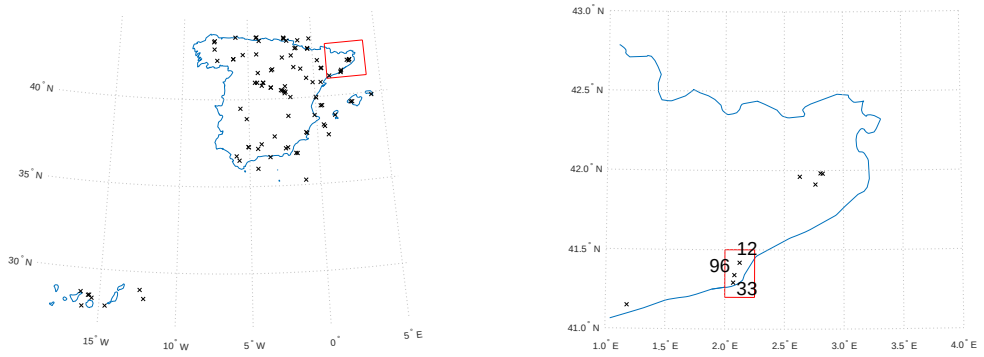
4.1 Experiment 1 – Temperature Sensor Data

In our first example, we will apply the model of causal graph processes to daily mean temperature data of multiple countries. As raw data, we obtained the daily mean temperature values for one country a time from the webpage of ECAD using custom queries. To test the performance, we took temperatures from the year 1979 and the following years. We first estimated the shift matrix $\mathbf{A}$ from one to three years of sensor data and used the learned model to forecast the next days temperature for the following year under

[1] Data available at `http://www.ecad.eu/dailydata/index.php`

the assumption that the days before are known perfectly. For the simulations, we always grouped all the temperature data of all stations into a $N \times 1$ vector $\mathbf{x}[k]$ and then stacked these vectors together into a data matrix $\mathbf{X} = (\mathbf{x}[1], \ldots, \mathbf{x}[K-1]) \in \mathbb{R}^{N \times K}$. From the so assembled data, we estimated the shift matrix $\mathbf{A}$ using the simplified Algorithm 2. We estimated causal process assuming a model order $M = 1$ or $M = 2$ due to largely increased complexity for higher-order processes. The choice of order $M = 1$ appeared to be especially appealing as it decreased the available parameter space to the choice of the sparsity parameter λ_1 as the only parameter influencing the solution. Furthermore, this also facilitates the interpretation of the shift matrix $\mathbf{A}$ as we will see later.

One of the first countries we want to investigate is Spain. It has a relatively large number, $N = 97$, of weather stations from which we know the mean temperature, and therefore, seemed to be a worthwhile country to be investigated. In Figure 4.1a, we depicted the geographic boundary of Spain, including the mainland as well as the Canary and the Balearic Islands. Each cross indicates a weather station from which data is available. The downside of this data set is that we have several groups of stations in close vicinity. As an example we zoomed in on the red boxed area of Figure 4.1a and show this part on the map magnified in Figure 4.1b.



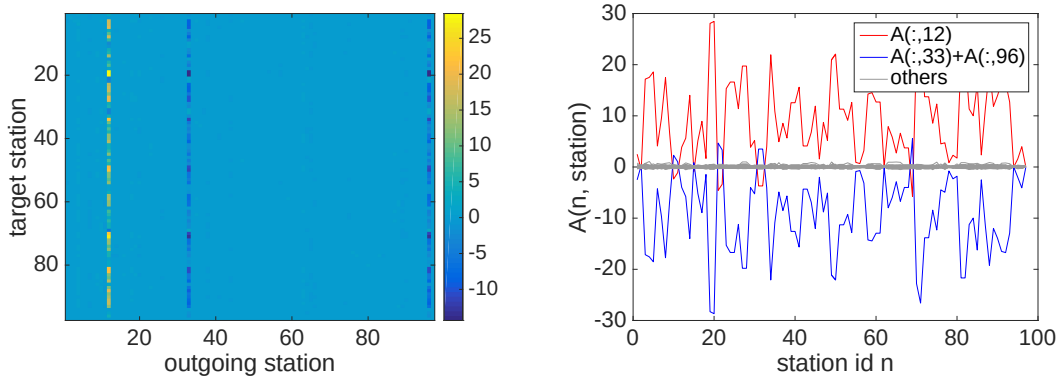
(a) all stations and the selected area

(b) selected area showing dominant stations

Figure 4.1: Locations of the weather stations in Spain

In the magnified map, we marked a group of three base stations that are particularly close. Due to their closeness, we expect the measured temperature data to be very similar. One might expect this would simply lead to large coefficients of the stations towards each other. Unfortunately, the proposed algorithm yields a different result.

Because of the number of base stations and simplicity of the model, we chose model order $M = 1$ with a sparsity coefficient of $\lambda_1 = 0.01$ that already yielded accurate results for artificial examples discussed in the last chapter. With these parameters, the algorithm yields the shift matrix $\mathbf{A}$ which is depicted in Figure 4.2a.



(a) Shift matrix $\mathbf{A}$ with the 3 dominant stations 12, 33, 96

(b) coefficients of $\mathbf{A}$ plotted column-wise

Figure 4.2: Estimated shift matrix $\mathbf{A}$ for the temperature data of Spain with $K = 365$, $M = 1$, $\lambda_1 = 0.01$

With that many stations distributed over the country, we would expect a sparse shift matrix that contains several entries representing effects of different, usually closely located base stations connected to each other. If we look upon the solution in Figure 4.2a, we are presented a model that can in a first approximation be described as

$$\mathbf{x}[k] \approx \mathbf{A}_{(:,12)}\mathbf{x}_{12}[k-1] + \mathbf{A}_{(:,33)}\mathbf{x}_{33}[k-1] + \mathbf{A}_{(:,96)}\mathbf{x}_{96}[k-1] \tag{4.1}$$

where $\mathbf{A}_{(:,i)}$ denotes the i-th column of the matrix $\mathbf{A}$. This means if we interpret $\mathbf{x}[k]$ as an estimate for the next vector of temperature values $\hat{\mathbf{x}}[k]$, the estimate only consists of a weighted combination of three stations past temperatures. This strange behavior can be explained by investigating the stations coefficients more closely. For this purpose, we plotted the three coefficients in Figure 4.2b. Since the stations 33 and 96 which are the closest to each other (which is visible in Figure 4.1b) have approximately the same coefficients ie. $\mathbf{A}_{(:,33)} \approx \mathbf{A}_{(:,96)}$, we displayed their sum instead of the individual values. It can be seen that this sum is a negative copy of the coefficients $\mathbf{A}_{(:,12)}$. This means, if we assume that the three close stations have about the same temperatures at day $k-1$, their weighted sum for Equation 4.1 collapses to

$$\mathbf{x}[k] \approx \left(\mathbf{A}_{(:,12)} + \mathbf{A}_{(:,33)} + \mathbf{A}_{(:,96)}\right)\mathbf{x}_{12,33,96}[k-1] \approx \mathbf{0}. \tag{4.2}$$

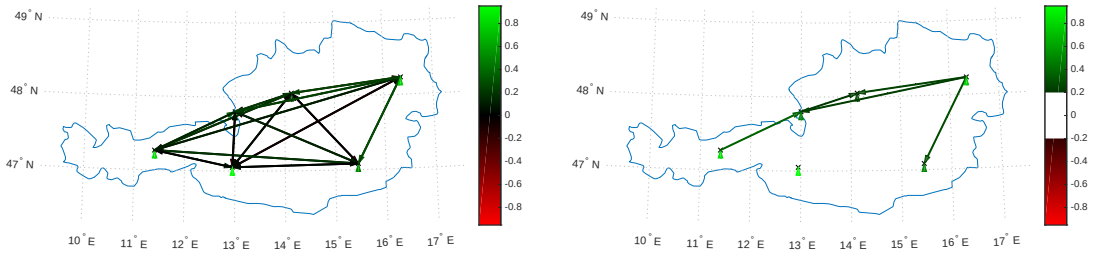
This says that all the most dominant coefficients in $\mathbf{A}$, that were included in the three mentioned columns, basically cancel each other out. Even if the remaining coefficients which are comparatively small lead to a good estimation, the resulting shift matrix is heavily misleading. We can conclude that groups of time series that have very similar values can introduce unwanted results and make the interpretation difficult. In the given example of a geographical map, one could apply a clustering based on geographical distance and replace the clusters with one virtual station. This cannot be seen as a general solution since it depends upon the assumption that a metric like in this example,

the geographical distance exists, which can be used to filter out problematic groups of time series. Alternatively, the purging of the training data could be done by directly searching for similar values. In any case, a preprocessing step appears to be necessary to avoid falling into this numerical trap.

Since this work was done in Vienna, it is the obvious candidate for investigation in Austria. The downside of using Austria stations for this analysis is the small number of participating weather stations that can be obtained from the ECAD dataset. There are only $N = 6$ stations with available temperature measurements. We ran the algorithm on the Austrian data using the same parameters as in the previous example.

Before we presented the resulting shift matrix $\mathbf{A}$ in visualization that closely resembled the matrix structure. To facilitate the interpretation, we chose a new way of displaying the result. Looking at the shift matrix, we interpret an entry $\mathbf{A}_{ij}$ as an outgoing connection from station j towards the target station i. We can say that the signal value $\mathbf{x}_j[k-1]$ influences the station j over the term $\mathbf{A}_{ij}\mathbf{x}_j[k-1]$. A positive $\mathbf{A}_{ij}$ indicates an influence with the same sign as the signal value while a negative entry shows that the next value of i is estimated to be smaller if j is bigger.

Therefore, we have chosen green arrows from i to j to represent positive $\mathbf{A}_{ij}$ and red arrows for the negative counterpart. The intensity of the color represents the strength of the connection. Bright green and red arrows show the strongest entries in the shift $\mathbf{A}$. In Figure 4.3a, we see the shift matrix visualized in this way. It also immediately shows the biggest problem of this approach – it results in too many connections. Even tough we require the shift $\mathbf{A}$ to be sparse, in real applications the resulting applications appear to be densely populated. There are however, fewer coefficients, that have larger magnitude then all others. To highlight the strongest coefficients we selected a threshold which is in this example 0.2 and consecutively only show coefficients whose magnitude exceeds this value. This results in Figure 4.3b with only several arrows representing the strongest links. As the temperature is strongly correlated with the previous day, the strongest arrows are drawn for the self-loops showing that the last day is estimated to be the strongest predictor.



(a) all connections

(b) only connections with a weight bigger than 0.2

Figure 4.3: Estimated shift matrix $\mathbf{A}$ for the temperature data of Austra with $K = 1095$, $M = 1$, $\lambda_1 = 0.01$

We can see that the Sonnblick station has no strong connections to other stations but only to itself. This could be explained by the location of this station, which is 3106 m above sea level while all others in the 500 m range. Other than this, no definite conclusions can be drawn from the resulting figure.

As the third country, we have a look at France. It is a bit larger than Spain, but there are fewer weather stations available. In total, we have data from $N = 39$ stations that are distributed quite uniformly over the whole country. This makes the country a more promising candidate than Austria with too few and Spain with too close weather stations.

For the two already mentioned countries, we chose the sparsity parameter without further discussion, since we could reveal the problems without calibrating the sparsity parameter λ_1. In this example, we elaborate more on the parameter choice. While we could calibrate λ in the previous chapters against the ground truth, this option is no longer available for real data. Since in the causal graph process the current sample $\mathbf{x}[k]$ is dependent on the previous graph signals $\mathbf{x}[k-m]$, it seems natural to compare the signal values predicted from past samples using the causal graph process using

$$\hat{\mathbf{x}}[k] = \mathbf{A}\mathbf{x}[k-1] + \left(c_{20}\mathbf{I} + c_{21}\mathbf{A} + c_{22}\mathbf{A}^2\right)\mathbf{x}[k-2] + \dots \tag{4.3}$$

where $\mathbf{x}[k-m]$ are the known values of past signals. This gives only a prediction of the next which is on itself not particularly useful. Luckily, we are less interested in the prediction performance itself but in comparing the predicted value with the real known value for the purpose of evaluating how well the model fits the given data.

First, we set the model order to $M = 1$ and used the temperature data of three years $K = 3 \cdot 365$ to estimate the shift matrix $\mathbf{A}$. With this matrix we evaluated Equation 4.3 for each day of the 4th year to obtain 365 estimated graph signals $\hat{\mathbf{x}}[k]$. As a reference, we used MATLAB's `yulewalker` function to estimate an auto regressive model (AR) for each time series. These model only considers the autocorrelation of the time series and cannot make use of other stations' data.

Figure 4.4 shows an example for a time series for one sensor containing all the one day ahead predictions. As a ground truth, we plotted the actual temperature values. Next we show the forecast achieved with the AR model of order 3, also always predicting the next day. Then, we show results obtained with Equation 4.3 using the shift matrices $\mathbf{A}$ we estimated with $\lambda_1 = 0.1, 0.01, 0.001$. Despite the figure only showing the time series of one station, the general trend is similar for the other weather stations. The estimation performance with $\lambda_1 = 0.01$ and $\lambda = 0.001$ is comparable to the autoregressive process. When we choose $\lambda_1 = 0.1$ the sparsity requirements are too strict to predict the next day's temperature.

To capture the estimation performance for all stations quantitatively, we grouped all predictions $\hat{\mathbf{x}}[k]$ into the matrix $\hat{\mathbf{X}}$ and all actual values into $\mathbf{X}_{\text{ref}}$. Then we calculated the MSE of the prediction for a method with the predicted values $\hat{\mathbf{X}}$ as

$$\text{MSE}(\hat{\mathbf{X}}) = \frac{1}{K' \cdot N} \|\hat{\mathbf{X}} - \mathbf{X}_{\text{ref}}\|_{\text{fro}}^2 \tag{4.4}$$

where we used set $K' = 365'$ to normalize for each day and $N = 39$.

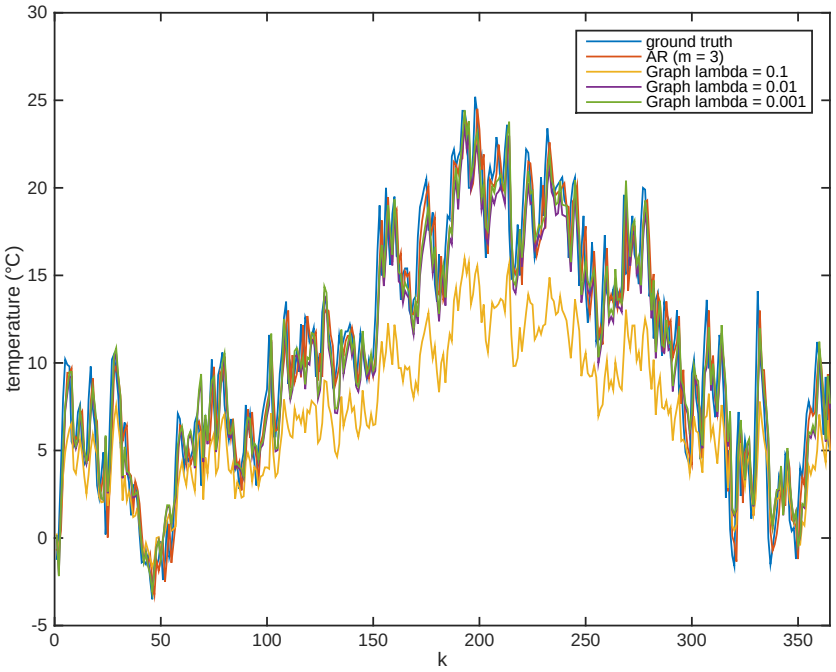


Figure 4.4: One day ahead prediction for one temperature sensor in France. Ground truth shows the real temperatures, AR is an auto regressive model of order 3 and the the others were estimated using causal graph processes with matrices that were obtained by varying λ_1

Table 4.1 shows the MSE results for $M = 1$ as well as $M = 2$. We used the autoregressive model order of $m = 3$ in Figure 4.4 as a reference. With higher orders of m the MSE steadily decreases, but only by a insignificant amount. To show this we also provide the error for the AR order 50.

	MSE $M = 1$	MSE $M = 2$
AR($m = 3$)	4.45	4.45
AR($m = 50$)	4.43	4.43
Causal graph process $\lambda_1 = 0.1$	17.90	36.87
Causal graph process $\lambda_1 = 0.01$	3.90	7.02
Causal graph process $\lambda_1 = 0.001$	3.73	5.90

Table 4.1: MSE evaluated by Equation 4.4 for an autoregressive model and different estimation parameters

The results in Table 4.1 confirm the visually observable trend of Figure 4.4. The causal graph process gives better results when the sparsity constraint is relaxed by choosing smaller values of λ_1. While the results for λ_1 are worse than the simple prediction based

on autocorrelation, we can outperform it with the causal graph process prediction with model order $M = 1$ and $\lambda \leq 0.01$ by using interrelations of different base stations.

Looking only at the MSE metric one would be tempted to select λ_1 as small as possible. Instead, we fix $\lambda_1 = 0.01$ because it outperforms the autoregressive process while still providing a reasonably sparse solution. When we visualize the results, we have to drop entries $\mathbf{A}_{ij}$ with a small magnitude for the solution to be interpretable. If we back off too much on the sparsity, the visualized part of the solution might be misleading as we drop too many entries.

Figure 4.5 shows the estimated shift matrix $\mathbf{A}$ for the France temperature data using a model order $M = 1$ and $\lambda_1 = 0.01$. For the sake of clarity, we only show 5% of the coefficients which have the biggest magnitude. The shift contains many positive self-loops which capture the autocorrelation with the previous temperature values. In contrast to Figure 4.2a, we do not observe stations that cancel their effects, which could lead to wrong interpretations. Notably, station 21 has negative weights to many different stations for which we lack an explanation.

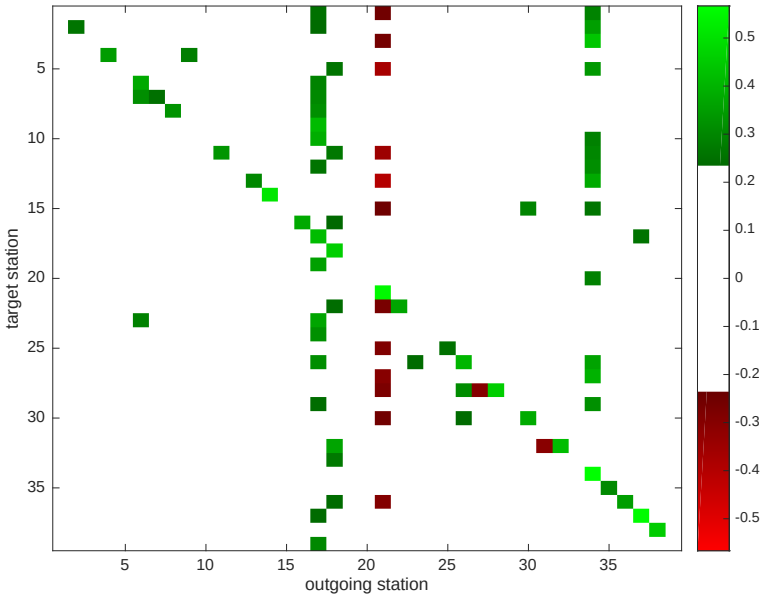


Figure 4.5: Estimated shift matrix $\mathbf{A}$ for the daily mean temperature data of France using $K = 3 \cdot 365$ days and $N = 39$ stations with a sparsity parameter $\lambda_1 = 0.01$ showing 5% of the matrix entries that have the largest magnitude

While Figure 4.5 gives us some first insights, plotting the same entries on a geographical map in Figure 4.6 reveals potential interpretations. Besides the strong auto correlations of the stations, there seems to be a trend of western stations on the coast influencing those in the central and eastern part of the country. The same trend, a west to east directivity of the weather, can be observed in Figure 4.7.

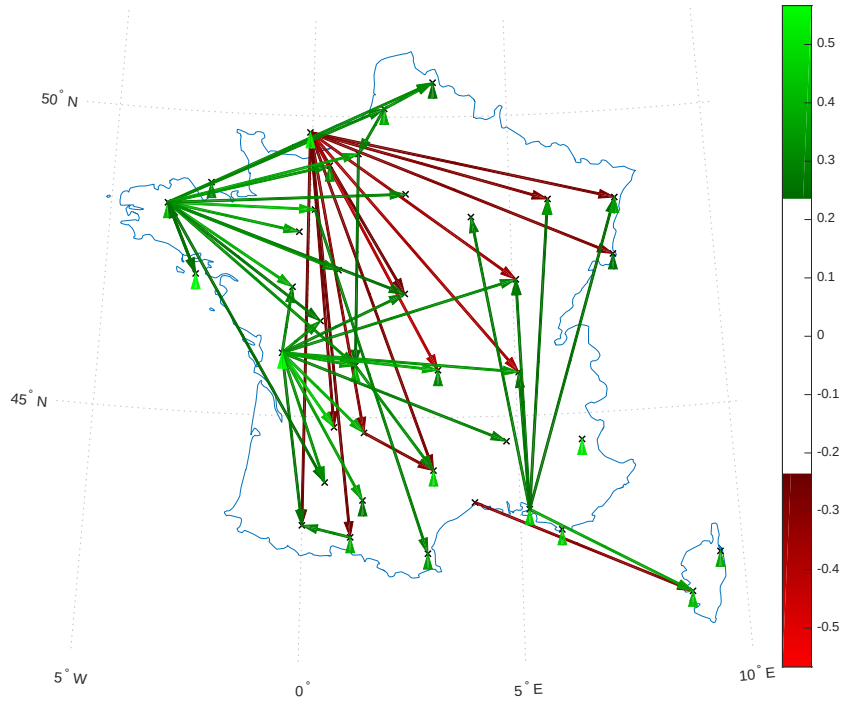


Figure 4.6: Visualization of the estimated graph shift of France's weather stations using model order $M = 1$ and $\lambda_1 = 0.01$

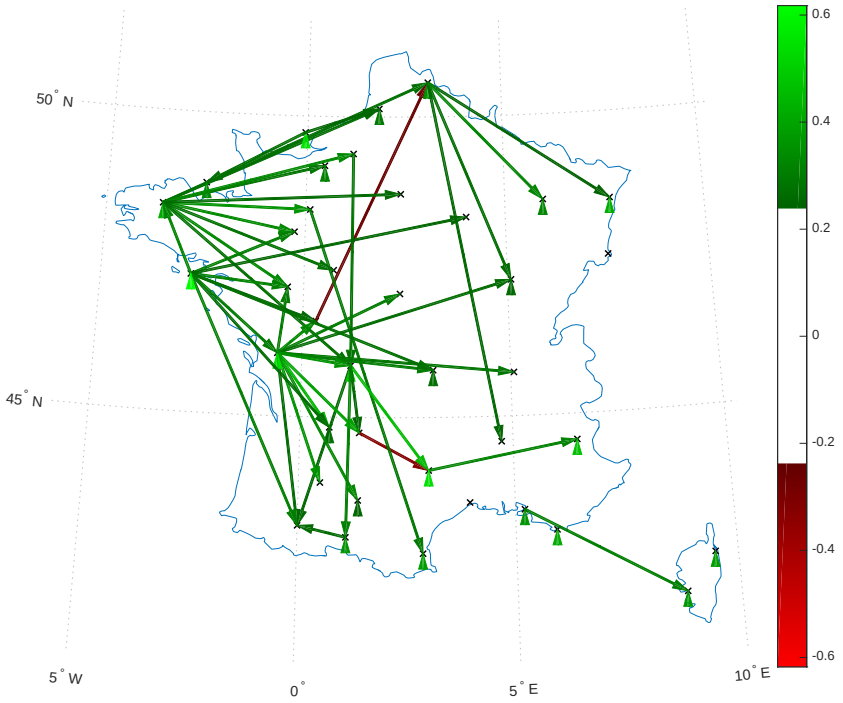


Figure 4.7: Visualization of the estimated graph shift of France's weather stations using model order $M = 2$ and $\lambda_1 = 0.01$

In the figure for model order $M = 2$, the west to east directivity seems to be more clearly. It might be tempting to call Figure 4.7 the "better" result since it looks more convincing. This, however, is a weak statement considering the MSE results shown in Table 4.1, showing the forecast performance for $M = 2$ to be worse than $M = 1$ and even not as good as the simple AR prediction.

This section showed that the algorithm could be used to investigate weather trends based on temperature data. For France, we could find a trend similar to that shown in [17] for the USA. The results, however, should be handled with care as we have shown that difficulties might arise with stations that are too close or there are too few data sources available. The next experiment will leave the physical domain towards the finance world were physical laws are not available and interpretations are even harder.

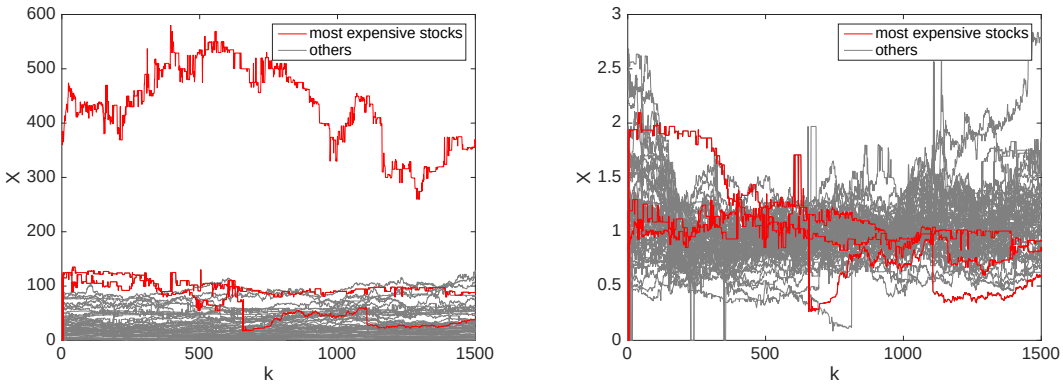
4.2 Experiment 2 – Austrian Stock Data

In this section, we want to apply the causal graph process estimation algorithm to stock prices and especially point out some additional points of failure we spotted. It is not intended as a rigorous financial or economic investigation.

The Viennese stock exchange[2]provides daily stock price data of all companies under the Austrian Traded Index (ATX). These market prices are available in the form of `.csv` files for each day. For this thesis, we manually downloaded the data for 1500 trading days[3] and parsed into MATLAB for further analysis.

From the 113 companies that were traded between January 2011 until March 2017, we selected those 61 which were continuously listed on the ATX. This reduces the problem size to be dealt within reasonable computation time. The time index k is relative to the first trading day in 2011 and simply excludes days where no trading took place. For each day, we know the market end price in Euros. Grouping the market closing prices for one day for each company, we obtain a vector sample $\mathbf{x}[k]$ for the trading day k. We again stack all these samples into the training matrix $\mathbf{X} = (\mathbf{x}[1] \dots \mathbf{x}[K])$ for all $K = 1500$ trading days.

Before applying the algorithm onto the stock data set, we plotted all time series into one graph. This is shown in Figure 4.8a where they y-axis shows the stock price in Euro. Besides the rapidly varying nature of some stocks, we can also observe a large difference in the average stock price. Especially the highest stock has an average that is magnitudes larger than the cheaper stocks. We expected this disparity to affect the numeric results for the estimation algorithm. Figure 4.9a shows the largest coefficients in the shift matrix $\mathbf{A}$ for the stocks depicted in Figure 4.8a.



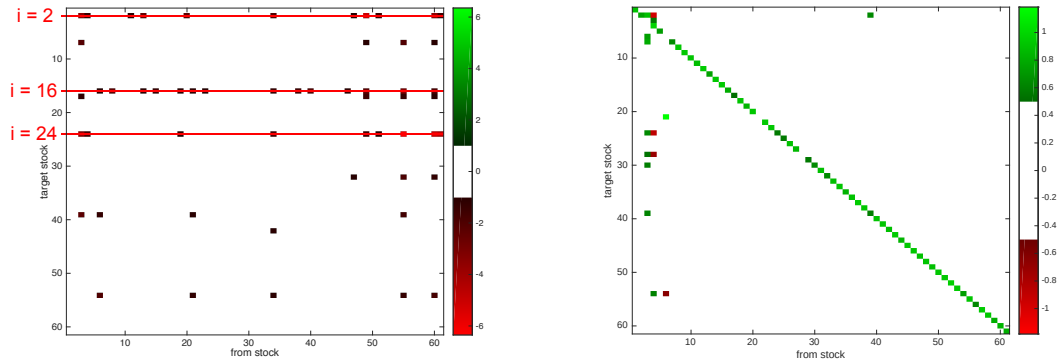
(a) without scaling

(b) with scaling

Figure 4.8: All stocks over time with and without scaling to mean 1

[2]`https://www.wienerborse.at/`

[3]Daily statistics are available at `https://www.wienerborse.at/marktdaten/statistiken/tagesstatistiken/tagesstatistik-download/`



(a) estimated **A** without scaling

(b) estimated **A** with scaling

Figure 4.9: Estimation result for **A** with $K = 1500$ days, $N = 61$ stocks, $M = 1$, $\lambda_1 = 0.01$

In the shift matrix shown in Figure 4.9a, we observe that the stocks number 2, 16 and 24 have many incoming connections. It appears unlikely that this is due to some economic relations and points towards a numerical problem. If we trace those three stocks in Figure 4.8a and mark them in red it is obvious that they correspond to some of the most expensive stocks. The algorithm tries to derive their next stock price from past other stocks but needs to use large coefficients because their value is so much larger than those of the others. This problem is not visible in the temperature data as most sensors have measurements of the same magnitude.

To overcome this potential pitfall, we decided to scale all stock price time series to have a mean of 1 over all 1500 trading days. This removes the direct interpretation as a price but since the relations are relative, we can afterwards rescale them to the actual prices if needed. The scaled time series are plotted in Figure 4.8b with the former pricey stocks drawn in red. When we run the algorithm using this averaged time series, we obtain the Figure 4.9b. We see that the horizontal lines in the shift matrix vanished and, as the biggest influence, we find diagonal entries in **A**. We conclude that given the scaled stocks the best prediction for the next day is most of the time using the previous day's stock price.

As we were trying to find potential interrelations between the different stocks and their companies, the almost diagonal shift matrix was not a desired result. Since the stock prices of some stocks change vary rapidly, while others remain constant, we suspected that those short time effects would overlay the long term relations we were interested in. Therefore, we decided to group always a number of N_g trading days together and replace their daily prices with one average price for each N_g days. In Figure 4.10 we show this grouping for the example of 100 days of Telekom Austria AG's stock prices. While the blue curve represents the daily changing market prices, the red curve shows the averaged prices for our virtual days. It can be seen, that, while the 5 day average still follows the price quite closely, the 15 day average is only weakly influenced by daily fluctuations in the market.

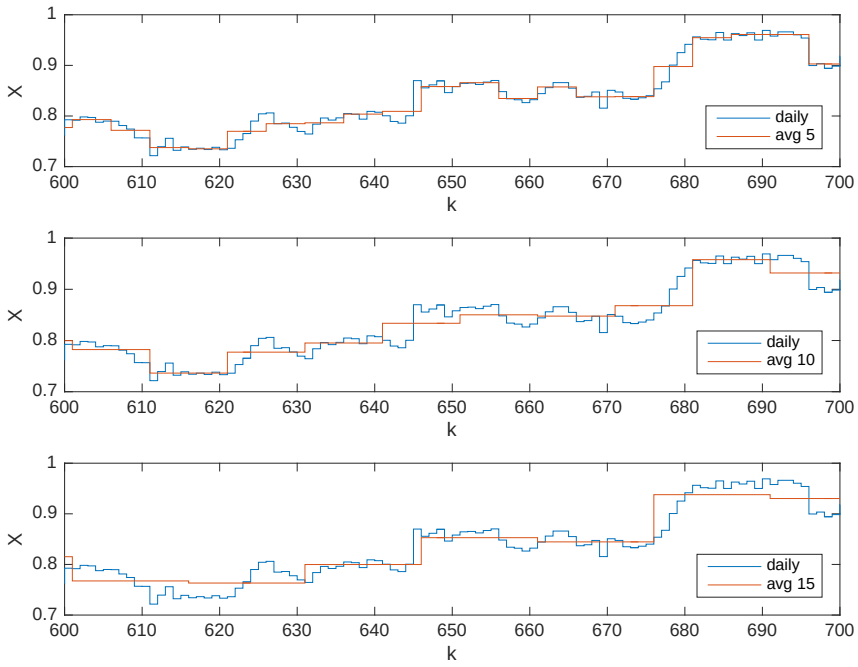


Figure 4.10: Temporal averaging of stock prices grouping 5, 10 and 15 days to into one data point

This casts our original problem of estimating the stock price of the next day to the question how the average of the last N_g affects the average of the next N_g trading days. We, then again, used the algorithm to estimate the shift matrices. To compare the results, we used a MSE as in the Equation 4.4 but with the stocks' data as a reference. Because there was only a limited amount of days available, especially in the case of grouped days, we calculated the prediction MSE using the same data as the one used to learn the matrix $\mathbf{A}$.

In Table 4.2, we show the MSE results we achieved for different parameters. All simulations were done with the sparsity parameter set to $\lambda_1 = 0.01$. We see that the graph based estimation method achieves results in the order of the autoregressive simulation. The resulting shift matrix for $M = 1$ and groups of $N_g = 5$ averaged days is shown in Figure 4.11. Besides the high autocorrelation we also observe a matrix which has no obvious algorithmic problems. This is in contrast to the other matrices, that were estimated after stronger averaging, which are shown in Figure 6.1, Figure 6.2 and Figure 6.3. These contain more horizontal and vertical "lines" of high coefficients, which can be seen the appendix. This could be explained by the shorter training sequences due to the averaging.

N_g	MSE AR	MSE $M = 1$	MSE $M = 2$
5	**0.0067**	0.0070	0.0290
10	**0.0138**	0.0139	0.0149
15	0.0202	**0.0189**	0.0200

Table 4.2: MSE for the stocks estimation using different parameters

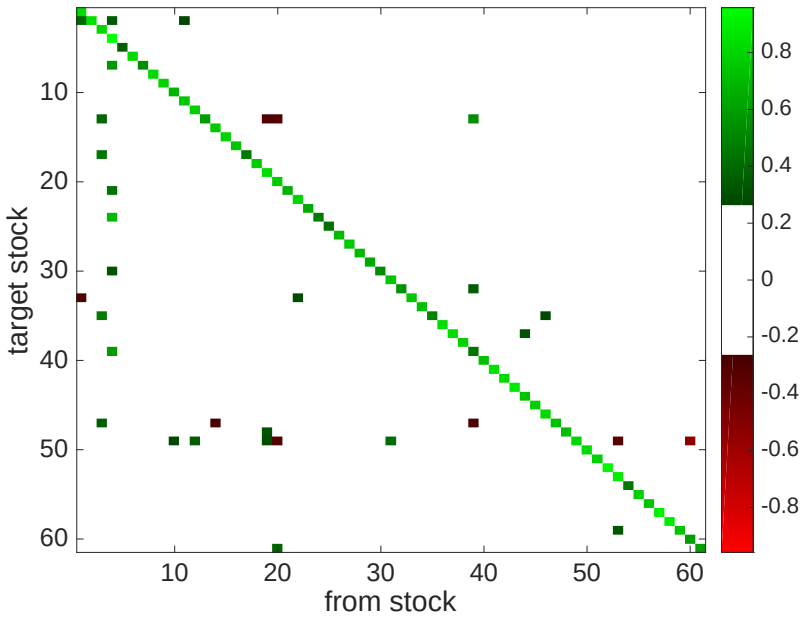


Figure 4.11: 2.5% largest entries in $\mathbf{A}$ estimated from Austrian stock data using $\lambda = 0.01$, $M = 1$, grouping $N_g = 10$ trading days

As we were interested in potential interpretations of the shift recovered from the stock data, we chose to visualize the largest possible directions of the shift shown in Figure 4.11 as a graph in Figure 4.12. The only observation we could draw from the graph is that there are multiple bank stocks, which affect multiple other stocks. Otherwise, the connected companies show no common ownership structure nor even similar or related products.

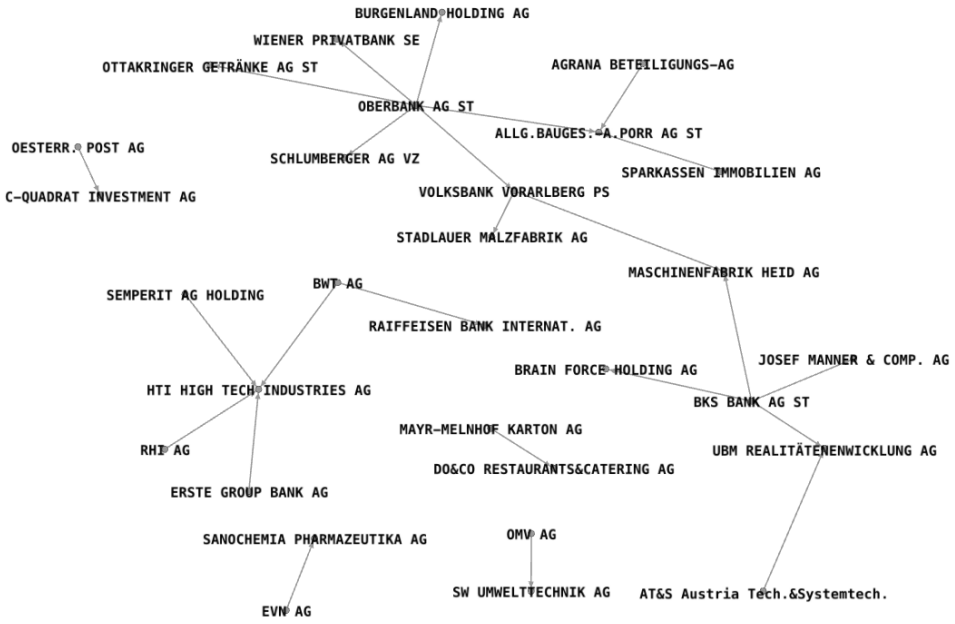


Figure 4.12: Strongest positive connections for $N_5 = 5$, $M = 1$ shown as a directed graph

Finally, we want to sum up this chapter of real-world examples. The weather data estimation yielded reasonable results when it was applied to countries with a uniform distribution of weather stations. Further preprocessing could potentially increase the significance in the results.

The stocks example with no clear expectation did not lead to promising results. Despite this, we described with scaling and averaging two processing steps that could be applied before starting the estimation algorithm. It is unclear if further tuning were needed or the domain of daily stock data cannot reasonably be modeled with causal graph processes, and we, therefore, leave this question open for future research.

5 Conclusion

In this thesis, we dealt with the topic of graph signal processing. It extends the field of signal processing by incorporating graphs to tackle data with irregular structure. Naturally, the graph itself plays an important role in this area of signal processing. Research in this area is focused on two different approaches.

The first approach we examined is based on unweighted graphs and the closely related Laplace matrix. It is often connected to a notion of smoothness, which requires graph signal values that are associated with connected nodes to be of similar value. Because of the symmetric matrices that describe the graph, the graph Fourier transform leads to real eigenvalues that can be interpreted as frequency equivalents. The downside of this approach is the inability to capture non-symmetric relations, which cannot be described by symmetric graphs.

Secondly, we reviewed the competing approach in which graph signal processing is seen as a generalization of classical signal processing in a more axiomatic way. By identifying key features of signal processing, the requirements for signals and filters lead to the basic building block of a graph shift. The graph shift itself can be seen as a more involved realization of the cyclic shift which is part of classical filter operations. Besides its interpretation as a shift, it can further be seen as a matrix representation of the graph. In contrast to the Laplacian direction, this branch of graph signal processing is capable of handling directed graphs.

While basic tools of the signal processing toolbox have been ported to graph based signal processing, the choice of the graph shift or simply the graph is still an interesting research question. In this work, we investigated a method of learning graph shifts, which was presented in [17]. In it, the authors propose the model of causal graph signals, which connect multiple time samples with graph filters. They assume that the current graph signal at one node is dependent on past samples at this node as well as past samples of connected nodes.

Assuming that an N dimensional time series can be modeled using such a causal graph process, [17] provides an algorithm that estimates the graph shift as well as the coefficients used in the graph filters that make up the causal graph model. This is achieved by casting the learning problem into a convex minimization task. There are two versions of the algorithm: the more complex base algorithm and its simplified counterpart. While the base algorithm requires multiple iterative convex minimization steps, the simplified algorithm recovers the shift matrix through only one minimization problem.

We investigated the performance of the algorithm by multiple artificial examples where we generated training data by using a causal graph process with a known shift matrix and known filter coefficients. When generating data we need to be careful to choose stable processes since otherwise the signals' magnitude tend to infinity and make us

hit the limits of floating point processing. We showed that, given filter coefficients and a shift matrix, the stability can be evaluated using a criterion used for vector auto regressive processes. For causal processes, we simulated both versions of the algorithm could recover the shift matrix with a high degree of accuracy for low model orders. The recovery of the filter coefficients appears less reliable. The simplified algorithm is less complex and even outperforms the more sophisticated base algorithm for causal graph processes with a short memory. The successful tests on toy examples motivated us to apply the same model to real-world data.

As the traditional example in the area of graph signal processing, we used weather data from weather stations distributed in selected countries. We had difficulties applying the model to countries with many close stations as they had very similar signal values. This lead to artifacts produced by the minimization procedure which render the interpretation not useful. For the network of France's weather stations, we could identify a directivity in the graph shift, similar to the one presented in the original paper.

After the weather stations, we additionally tried to apply the algorithm to snapshots of the Austrian stock market, treating the individual stocks as nodes on an unknown graph. This example showed the difficulties induced by high dynamics between the signal values as well as high temporal fluctuations. We employed two different approaches of preprocessing to the raw data. Still the interpretation of the graph shift as relation between companies was inconclusive. We decided to include this example as it shows the reality in which models do not fit every type of data.

The investigated algorithm appears capable of estimating data that follows the causal graph model, but has shortfalls when applied to real world data. Preprocessing steps are likely to be needed to make raw data accessible to this graph learning technique. Further research should focus on the applicability of the model to the investigated domain and how the data needs to be modified before the estimation algorithm can be applied.

6 Appendix

6.1 Definitions

Definition 6.1. The *Characteristic Polynomial* $p_{\mathbf{A}}(x)$ of a $N \times N$-matrix $\mathbf{A}$ is the polynomial $p_{\mathbf{A}}(x) = \det(x\mathbf{I} - \mathbf{A})$.

Definition 6.2. The *Minimal Polynomial* $m_{\mathbf{A}}(x)$ of a $N \times N$-matrix $\mathbf{A}$ is the monic polynomial of smallest degree such that $m_{\mathbf{A}}(\mathbf{A}) = \mathbf{0}$.

Definition 6.3. Given a $N \times N$-matrix $\mathbf{A}$ there exists a *Jordan decomposition* of the form

$$\mathbf{A} = \mathbf{V}\mathbf{J}\mathbf{V}^{-1} \tag{6.1}$$

where $\mathbf{V}$ is a a matrix containing the generalized eigenvectors of $\mathbf{A}$ and $\mathbf{J}$ is called the *Jordan normal form* of $\mathbf{A}$.

If $\mathbf{A}$ has N eigenvalues the Jordan normal form is a diagonal matrix containing the eigenvalues and $\mathbf{V}$ contains all eigenvectors of $\mathbf{A}$. Otherwise $M \leq N$ distinct eigenvalues exist and eigenvalue λ_m has D_m independent eigenvectors $\mathbf{v}_{m,0}, \ldots, \mathbf{v}_{m,D_m-1}$ where D_m is called the geometric multiplicity of λ_m. Each eigenvector $\mathbf{v}_{m,d}$ generates a Jordan chain of $R_{m,d} \geq 1$ linear independent generalized eigenvectors $\mathbf{v}_{m,d,r}$ with $0 \leq r < R_{m,d}$ and the first generalized eigenvector is the eigenvector i.e. $\mathbf{v}_{m,d} = \mathbf{v}_{m,d,0}$. The generalized eigenvectors satisfy the equation

$$(\mathbf{A} - \lambda_m\mathbf{I})\mathbf{v}_{m,d,r} = \mathbf{v}_{m,d,r-1}. \tag{6.2}$$

Each eigenvector $\mathbf{v}_{m,d}$ corresponds to a Jordan block of dimension $R_{m,d}$ of the form

$$J_{r_{m,d}}(\lambda_m) = \begin{pmatrix} \lambda_m & 1 & & \\ & \lambda_m & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_m \end{pmatrix}. \tag{6.3}$$

For each eigenvalue λ_m there are D_m Jordan blocks of the form of equation 6.3. Furthermore we can group all generalized eigenvectors that correspond to an eigenvector $\mathbf{v}_{m,d}$ to a $N \times R_{m,d}$ matrix

$$\mathbf{V}_{m,d} = (\mathbf{v}_{m,d,0} \ldots \mathbf{v}_{m,d,R_{m,d}-1}). \tag{6.4}$$

Grouping all those block matrixes $\mathbf{V}_{m,d}$ into one matrix we obtain

$$\mathbf{V} = (\mathbf{V}_{0,0} \ldots \mathbf{V}_{M-1,D_{M-1}}) \tag{6.5}$$

as the matrix of generalized eigenvectors. Those belong to the Jordan normal form

$$\mathbf{J} = \begin{pmatrix} \mathbf{J}_{R_{0,0}}(\lambda_0) & & \\ & \ddots & \\ & & \mathbf{J}_{R_{M-1,D_{M-1}}}(\lambda_{M-1}). \end{pmatrix} \tag{6.6}$$

This definition is adapted from the Appendix A of [2].

6.2 Additional Figures

This section contains additional figures that did not make it into the main text.

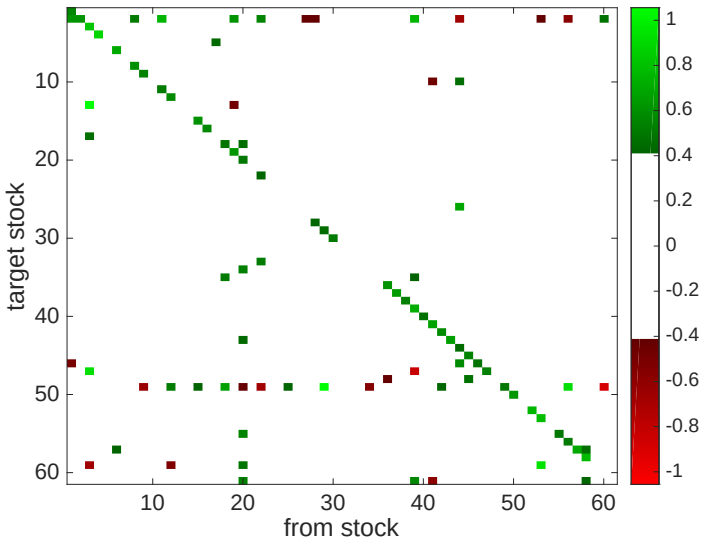


Figure 6.1: 2.5% largest entries in $\mathbf{A}$ estimated from Austrian stock data using $\lambda = 0.01$, $M = 1$, grouping $N_g = 10$ trading days

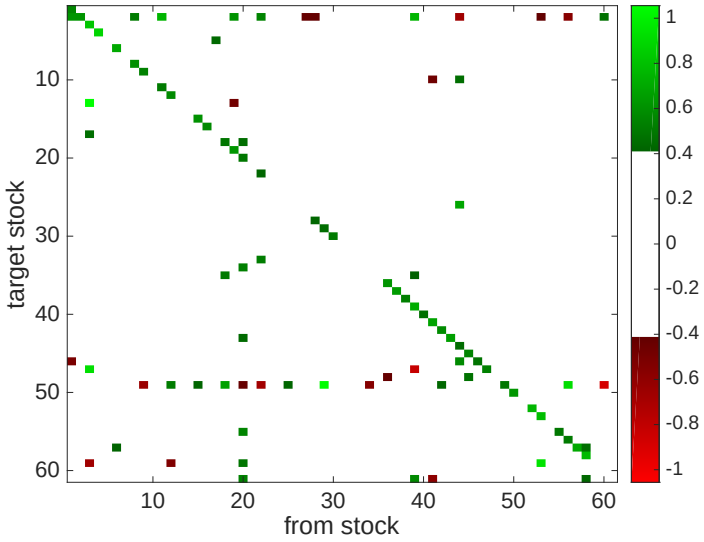


Figure 6.2: 2.5% largest entries in $\mathbf{A}$ estimated from Austrian stock data using $\lambda = 0.01$, $M = 1$, grouping $N_g = 15$ trading days

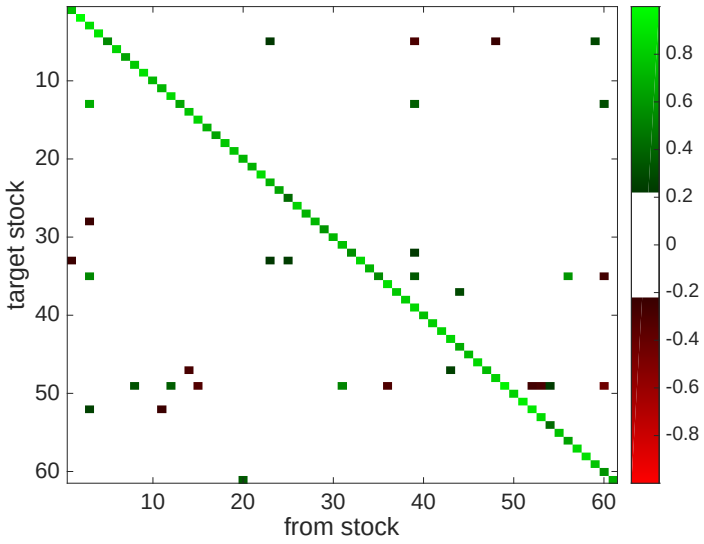


Figure 6.3: 2.5% largest entries in $\mathbf{A}$ estimated from Austrian stock data using $\lambda = 0.01$, $M = 2$, grouping $N_g = 5$ trading days

Bibliography

[1] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains", *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[2] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs", *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[3] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian Matrix in Smooth Graph Signal Representations", *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160–6173, 2016.

[4] A. V. Oppenheim, *Discrete-time signal processing*. Pearson Education India, 1999.

[5] J. Bang-Jensen and G. Z. Gutin, *Digraphs: Theory, algorithms and applications*. Springer Science & Business Media, 2008.

[6] A. Jung, P. Berger, G. Hannak, and G. Matz, "Scalable graph signal recovery for big data over networks", in *Signal Processing Advances in Wireless Communications (SPAWC), 2016 IEEE 17th International Workshop on*, IEEE, 2016, pp. 1–6.

[7] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs: Frequency analysis.", *IEEE Trans. Signal Processing*, vol. 62, no. 12, pp. 3042–3054, 2014.

[8] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation", *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[9] A. J. Smola and R. Kondor, "Kernels and regularization on graphs", in *Learning theory and kernel machines*, Springer, 2003, pp. 144–158.

[10] M. Püschel and J. M. Moura, "Algebraic signal processing theory", *ArXiv preprint cs/0612077*, 2006.

[11] M. Puschel and J. M. Moura, "Algebraic signal processing theory: Foundation and 1-d time", *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3572–3585, 2008.

[12] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory", *IEEE Transactions on Signal Processing*, vol. 63, no. 24, pp. 6510–6523, 2015.

[13] S. Chen, R. Varma, A. Singh, and J. Kovačević, "Signal recovery on graphs: Fundamental limits of sampling strategies", *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 539–554, 2016.

[14] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure", *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[15] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovacevic, "Adaptive graph filtering: Multiresolution classification on graphs", in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, IEEE, 2013, pp. 427–430.

[16] S. Chen, R. Varma, A. Singh, and J. Kovačević, "Representations of piecewise smooth signals on graphs", in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, IEEE, 2016, pp. 6370–6374.

[17] J. Mei and J. M. Moura, "Signal processing on graphs: Causal modeling of unstructured data", *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 2077–2092, 2016.

[18] G. M. Goerg and C. R. Shalizi, "Licors: Light cone reconstruction of states for non-parametric forecasting of spatio-temporal systems", *ArXiv preprint arXiv:1206.2398*, 2012.

[19] MATLAB, *Version 8.6.0 (r2015b)*. Natick, Massachusetts: The MathWorks Inc., 2015.

[20] M. Grant and S. Boyd, *CVX: Matlab software for disciplined convex programming, version 2.1*, `http://cvxr.com/cvx`, Mar. 2014.

[21] ——, "Graph implementations for nonsmooth convex programs", in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, V. Blondel, S. Boyd, and H. Kimura, Eds., `http://stanford.edu/~boyd/graph_dcp.html`, Springer-Verlag Limited, 2008, pp. 95–110.

[22] K.-C. Toh, M. J. Todd, and R. H. Tütüncü, "SDPT3—a matlab software package for semidefinite programming, version 1.3", *Optimization methods and software*, vol. 11, no. 1-4, pp. 545–581, 1999.

[23] R. H. Tütüncü, K.-C. Toh, and M. J. Todd, "Solving semidefinite-quadratic-linear programs using SDPT3", *Mathematical programming*, vol. 95, no. 2, pp. 189–217, 2003.

[24] H. Lütkepohl, *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

[25] A. Klein Tank, J. Wijngaard, G. Können, R. Böhm, G. Demarée, A. Gocheva, M. Mileta, S. Pashiardis, L. Hejkrlik, C. Kern-Hansen, *et al.*, "Daily dataset of 20th-century surface air temperature and precipitation series for the european climate assessment", *International Journal of Climatology*, vol. 22, no. 12, pp. 1441–1453, 2002.

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct – Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des jeweiligen Mitteilungsblattes der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In– noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, 6. Juni 2017

Lukas Nagel