Multiscale, Multi-Dimensional Space and Space-Time Function Estimation for Irregular Network Data

Naventhan Mahadevan

School of Mathematics

September 2009

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Doctor of Philosophy in the Faculty of Science

Word count: 51,730.

Abstract

We explore non-parametric space and space-time function estimation for multi-dimensional irregular network data. Recently a lifting based technique called *lifting one coefficient at a time* (LOCAAT) was developed by Jansen, Nason and Silverman. LOCAAT enables wavelet-like signal processing for irregular network data in multi-dimensions.

A main focus of this thesis is to provide a "good" function estimation strategy for network data collected through time. In order to carry out this task, we introduce the concept of spatial network and spatio-temporal network estimation. We propose several noise variance estimation techniques and thresholding strategies for the LOCAAT transform which work well even with small number of nodes. In addition to thresholding, we have explored other methods to further improve the estimation efficiency such as avoiding sensitive coefficients from being thresholded and smoothing temporal LOCAAT coefficients in wavelet domain.

The final part of the thesis provides a simple network forecasting strategy by predicting LOCAAT coefficient in wavelet domain which proves to work better than time domain forecasting.

Dedication

I dedicate this thesis to my mom and dad.

Acknowledgements

I would like to thank my supervisor Professor Guy Nason for his full support and the care he has shown throughout my PhD. Without his help and motivation this PhD would not have been possible. Thanks to Professor Guy Nason and Professor Alistair Munro for initiating this PhD and providing access to the facilities in both Departments of Mathematics and of Electrical and Electronic Engineering. I would like to thank the Data and Information Fusion Defence Technology Centre (DIF-DTC) for funding my PhD. I would also like to thank Professor Nishan Canagarajah for his support and useful conversations we had at times.

Thanks to Douglas Harding from the Health Protection Agency for supplying the example Mumps data set. Thanks to Matthew Nunes for providing code to produce a network of counties in England and the useful conversations we had in the past.

I would also like to thank my brother-in-laws Kugathasan and Rajakumar for all their support and encouragement when I needed it. I would like to thank my friends Nithin, Siva, Sun, Arindam, Anjulan, Georgios, Bernice, Meng and Judy for all the encouragement they gave me at difficult times and all the good times we had during my PhD.

Without all their support and encouragement this PhD would not have been possible, so I thank all of them once again!.

Author's Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED:..... DATE:.....

Contents

List of Acronyms

ACF ..... autocorrelation function

AODV ..... Ad-hoc On-demand Distance Vector

AR ..... Autoregressive

ARIMA ..... Autoregressive Integrated Moving Average

ARMA ..... Autoregressive Moving Average

BBC ..... Box Block Choice

BLUE ..... Best Linear Unbiased Estimator

BT ..... Block Threshold

DIF-DTC .... Data and Information Fusion Defence Technology Centre

DWT ..... Discrete Wavelet Transform

FFT ..... Fast Fourier Transform

GCV ..... Generalised Cross Validation

GUI ..... Graphical User Interface

HPA ..... Health Protection Agency

HT ..... Hard Threshold

IID ..... Independent and Identically Distributed

IQR ..... InterQuartile Range

LOCAAT ..... Lifting One Coefficient At A Time

LOWESS ..... LOcally WEighted Scatterplot Smoother

MA ..... Moving Average

MAD ..... Median Absolute Deviation

MISE ..... Mean Integrated Squared Error

MML ..... Marginal Maximum Likelihood

MMSE ..... Minimum Mean Squared Error

MOCV ..... Modified Ordinary Cross Validation

MRA ..... Multiresolution Analysis

MSE ..... Mean Squared Error

MSPE ..... Mean Squared Prediction Error

MST ..... Minimum Spanning Tree

NC ..... NeighCoeff

OCV ..... Ordinary Cross Validation

PACF ..... partial autocorrelation function

SAR ..... Spatial Autoregressive

SES ..... Simple Exponential Smoothing

ST ..... Soft Threshold

SURE..... Stein's Unbiased Risk Estimator

WSN..... Wireless Sensor Networks

List of Tables

List of Figures

Chapter 1

Introduction

Wavelet theory is relatively new and has attracted many researchers to develop the area. Wavelets are attractive for their simplicity and ability to represent functions with sparse representations. Much wavelet work is based on the fast Discrete Wavelet Transform (DWT) introduced in [57]. Traditionally, the fast wavelet transform relies on regular data assumptions and number of data to be a power of two. In the real world, data arise in an irregular fashion and in arbitrary number. Therefore, these restrictions have to be relaxed. The following lists attempts to relax the restrictions of regular wavelets.

The work in [55, 56] developed a theory for mapping irregularly spaced data to a regular grid by linear transformation and then applying standard wavelet techniques. A fast cross-validation method for work in [56] was proposed in [64]. Work in [82, 84] introduced a method called the *lifting scheme* which can handle multidimensional irregularly spaced data. An adaptation of lifting to curve estimation problems can be found in [29]. Lifting for two dimensions can be found in [28, 30]. Work in [46] introduced a new paradigm, which we refer to as LOCAAT throughout this thesis, for a multidimensional irregular data on graph. An adaptive lifting approach was presented in [68].

1.1 Motivation

Many physical problems require the estimation of a function that varies in both time and space. We consider estimating a function defined on a network subject to noise and given information about the network. Our motivation for this problem comes from networks such as Wireless Sensor Networks (WSN) and transport networks which have many potential irregularities. For example, in WSNs the irregularities arise in irregular node placement and irregular data sampling [3, 4]. There is a lot of recent interest in wavelet-like signal processing for WSNs (see references [18, 36, 37, 38, 85, 86]).

Generally, networks evolve over time and it would be useful to have some means of signal processing to denoise and hence analyse the behaviour of these networks. Although wavelet-based statistical signal processing holds much promise for network data, there were few wavelet techniques available until recently to handle these network irregularities. The development of LOCAAT opens up a new avenue of interest into research on wavelet-like signal processing for network data.

1.2 Main contributions

Motivated by possible applications of LOCAAT to network data (such as WSN data) we have carried out exploration of spatial and spatio-temporal function estimation and network forecasting using the LOCAAT algorithm. We outline our main contributions next.

- We have proposed and explored several reliable noise variance estimation procedures for networks, even with small number of nodes.
  - We propose two local Median Absolute Deviation (MAD) estimators for LOCAAT.

  - We propose a noise variance estimation for network data (when time series is available for each node) based on a variogram technique. This method is named TM1 in chapter 4.
  - We propose several noise variance estimation methods purely based on network data. The methods are named TM2, TM3, TM4 and TM5 in chapter 4.
  - We propose noise variance estimation based on LOCAAT coefficients (based on differences of LOCAAT coefficients in current time and future time). This method is named as WM1 in chapter 4.
  - We propose several noise variance estimation based on LOCAAT coefficients purely based on one snapshot of time. The methods are named WM2, WM3 and WM4 in chapter 4.
  - The variogram methods (such as TM2, TM3, WM2, WM3 and WM4) require bias correcting constants. We propose a method to find these.

- We proposed and explored several function estimation (both spatial and spatio-temporal) methods listed below.
  - We explore several existing threshold methods (such as hard, soft, block thresholding) for LOCAAT. We found most of them over smooth and introduce estimation bias. Therefore, we explore using a lower threshold.
  - We propose a block thresholding-like approach called Box Block Choice (BBC) for spatially distributed lifting coefficients.
  - We propose a method for improving estimation efficiency by avoiding sensitive coefficients being thresholded.
  - We propose a cross-validation and SURE based threshold methods for LOCAAT.

  - We propose a method for improving estimation efficiency of spatio-temporal network function by smoothing LOCAAT coefficients.
  - We recommend a stopping time for the number of LOCAAT transform steps.

- We have propose a simple network forecasting strategy by using LOCAAT coefficient prediction.

1.3 Thesis Organisation

Chapter 2 reviews some background material used throughout this thesis. We start by reviewing wavelets, the Discrete Wavelet Transform (DWT), and the lifting transform from [83]. We then introduce the main candidate, which we use to analyse the network data throughout the thesis, Lifting One Coefficient At A Time (LOCAAT) introduced in [46]. We also review some wavelet-based smoothing methods and noise variance estimation methods. For completeness, we have also outlined briefly some other popular smoothing methods in the non-parametric literature.

Chapter 3 introduces some test networks and test functions used throughout the thesis. We also introduce the concept of spatial network methods and spatio-temporal methods.

Chapter 4 explores noise variance estimation methods for networks. We introduce the concept of local Median Absolute Deviation (MAD) estimation for our network lifting coefficients. We also propose several variogram-based techniques in the time and wavelet domains to estimate noise variance. We also introduce a solution to the bias problem that arises in such variogram based methods.

Chapter 5 introduces function estimation using existing thresholding methods and then proposes some new thresholding methods such as mean correction methods and Box Block Choice (BBC). In addition to these thresholding methods, we

propose an improvement strategy to the estimation by avoiding sensitive coefficients. We also explore cross-validation and SURE thresholding methods. Once the thresholding methods are defined for spatial and spatio-temporal function estimation, we propose an improvement to the spatio-temporal function estimation by smoothing the temporal wavelet coefficients. We then explore sparsity and propose an optimal stopping time for the number of steps for LOCAAT transform.

Chapter 6 proposes simple network forecasting techniques using Autoregressive Integrated Moving Average (ARIMA) and Simple Exponential Smoothing (SES) modelling on both raw data and lifting coefficients. We demonstrate this with simulation experiments and a real data study.

Chapter 7 concludes the thesis and give some future directions.

Chapter 2

Literature Review

2.1 Introduction

In this chapter we review some fundamental theory and recent developments in the area of nonparametric regression. We mainly focus on the wavelet and the lifting transforms. The initial part of this chapter introduces a general overview of wavelets and the lifting scheme. Then we introduce the Lifting One Coefficient At A Time (LOCAAT) method, which we use as our main tool in this thesis. We then review some existing thresholding and noise variance estimation methods in the wavelet context. Finally, we introduce two standalone sections, optimisation and sparsity, as some of our later work requires them.

2.2 Introduction to wavelets

There is much literature on wavelets. Good introductions to wavelets can be found in [9, 17, 26, 27, 58, 61]. The description in this section closely follows [9].

A wave is an oscillating function in time or space, such as a sinusoid. A *wavelet* is a *"small wave"*. Unlike Fourier analysis, wavelet theory is relatively new. Wavelet theory has developed some promising ways for signal processing with various different wavelets. Key applications of wavelets are signal denoising, nonparametric

function estimation and smoothing [33, 49, 78, 80]. Density estimation of stationary long memory processes has been studied in [2, 60, 87].

The goal of wavelet transform theory is to use the mother wavelet $\psi(t)$, its dilations (scales) and their translations (shifts), as building blocks for a function $f(t) \in \mathcal{V}_0$ where $\mathcal{V}_0 \subset L^2(\mathbb{R})$.

Definition 2.1. *Let $\psi_{s,\tau}$, $s \in \mathbb{R} \setminus \{0\}, \tau \in \mathbb{R}$ be a family of functions being translations and dilations of a single function $\psi(t) \in L^2(\mathbb{R})$, called the mother wavelet (see [26, 78]), and is defined as follows,*

$$\psi_{s,\tau}(t) = |s|^{-1/2}\psi\left(\frac{t-\tau}{s}\right), \tag{2.1}$$

where s denotes dilation (scale) and τ denotes translation (shifts). The factor $|s|^{-1/2}$ is a normalising factor which is essential to preserve L^2 norms, i.e. $\|\psi_{s,\tau}\|_2 = \|\psi\|_2$. Typically $\|\psi\|_2 = 1$.

The mother wavelet is assumed to satisfy the admissibility condition,

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty, \tag{2.2}$$

where $\Psi(\omega)$ is the Fourier transform of $\psi(t)$. If $\psi(t) \in L^2(\mathbb{R})$, then $\Psi(\omega)$ is continuous and the condition (2.2) can only be satisfied if $\Psi(0) = 0$ or

$$\int_{\mathbb{R}} \psi(t)dt = 0. \tag{2.3}$$

See page 24 in [26] for the explanation of these choices. Condition (2.2) means that $\psi(t)$ should be localized in frequency domain. On the other hand, condition (2.3) means that $\psi(t)$ is localized in time and also oscillatory. Hence the name *wavelet*.

2.2.1 Multiresolution Analysis (MRA) and the Discrete Wavelet Transform (DWT)

The idea of the multiresolution analysis was first presented in [57]. The following review closely follows [9].

Scaling functions

The idea of scaling functions comes from the basis or expansion set. If one starts with a vector space of signals, $\mathcal{V}_0$, then if any $f(t) \in \mathcal{V}_0$ can be expressed as $f(t) = \sum_k c_k \phi_k(t)$ then the set of functions $\{\phi_k(t)\}$ is called an expansion set for the space $\mathcal{V}_0$. If the representation is unique, the set is a basis. Alternatively, one could start with the expansion set and define the space $\mathcal{V}_0$ where all the functions that belong to this space can be represented as a linear combination of the expansion set. This is called the span of the expansion set [9]. The translation (shifting) is defined as follows,

$$\phi_k(t) = \phi(t-k), \qquad k \in \mathbb{Z}, \qquad \phi \in L^2(\mathbb{R}). \tag{2.4}$$

The subspace of $L^2(\mathbb{R})$ spanned by these functions is denoted as

$$\mathcal{V}_0 = \overline{\text{Span}\{\phi_k(t)\}}, \qquad \forall f(t) \in \mathcal{V}_0. \tag{2.5}$$

Therefore, if the family of functions $\phi_k(t)$ span the subspace $\mathcal{V}_0$ then any function $f(t) \in \mathcal{V}_0$ can be written as a linear combination of scaling functions $\phi_k(t)$ and scaling coefficients,

$$f(t) = \sum_k c_k \phi_k(t) \qquad \text{for any} f(t) \in \mathcal{V}_0. \tag{2.6}$$

It is also possible to extend the size of the subspace spanned by changing the time scale of the scaling function. A family of scaling functions is given as follows,

$$\phi_{j,k}(t) = 2^{j/2}\phi(2^j t - k), \tag{2.7}$$

whose span over k is,

$$\mathcal{V}_j = \overline{\text{Span}\{\phi_k(2^j t)\}} = \overline{\text{Span}\{\phi_{j,k}(t)\}} \qquad \forall k \in \mathbb{Z}, \tag{2.8}$$

where $j \in \mathbb{Z}$.

Definition 2.2. *Multiresolution Analysis (MRA) of $L^2(\mathbb{R})$ is given by a system of closed subspaces $(V_j)_{j\in\mathbb{Z}}$ of $L^2(\mathbb{R})$ provided the following conditions are satisfied:*

1. $\cdots \subset \mathcal{V}_{-2} \subset \mathcal{V}_{-1} \subset \mathcal{V}_0 \subset \mathcal{V}_1 \subset \mathcal{V}_2 \subset \cdots \subset L^2(\mathbb{R})$

   or

   $\mathcal{V}_j \subset \mathcal{V}_{j+1}$

2. $\overline{\bigcup_{j\in\mathbb{Z}} V_j} = L^2(\mathbb{R})$, *i.e.* $\bigcup_{j\in\mathbb{Z}} V_j$ *is dense in* $L^2(\mathbb{R})$

3. $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$

4. $f(t) \in V_0 \iff f(2^j t) \in V_j$

5. $f(t) \in V_0 \iff f(t-k) \in V_0, \qquad \forall k \in \mathbb{Z}$

6. $\exists \phi \in V_0$, *called* scaling function *or* father wavelet *such that* $\{\phi_{0,k}; k \in \mathbb{Z}\}$ *is an orthonormal basis in* V_0

Figure 2.1 shows the relationship of the spanned spaces.

Wavelet functions

Important features of a signal are revealed not when the the signal is represented by the linear combination of scaling functions $\phi_{j,k}(t)$ but when they are represented

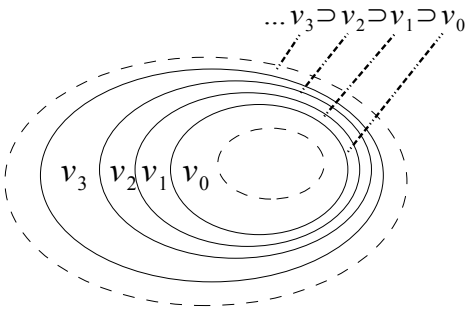


Figure 2.1: Nested vector space spanned by the scaling functions (see figure 2.1 in [9]).

by the slightly different functions $\psi_{j,k}(t)$, called wavelets, that span the differences between the spaces spanned by the various scales of scaling functions [9]. Let the wavelet spanned subspace $\mathcal{W}_0$ be defined by,

$$\begin{aligned} \mathcal{V}_1 &= \mathcal{V}_0 \oplus \mathcal{W}_0, \\ \mathcal{V}_2 &= \mathcal{V}_0 \oplus \mathcal{W}_0 \oplus \mathcal{W}_1, \\ L^2(\mathbb{R}) &= \mathcal{V}_0 \oplus \mathcal{W}_0 \oplus \mathcal{W}_1 \oplus \cdots, \end{aligned} \tag{2.9}$$

where $\mathcal{W}_j$ are the orthogonal complement of $\mathcal{V}_j$ in $\mathcal{V}_{j+1}$ (see figure 2.2). This means that all members of the $\mathcal{V}_j$ are orthogonal to all members in $\mathcal{W}_j$, i.e.

$$\langle \phi_{j,k}(t), \psi_{j,l}(t) \rangle = \int \phi_{j,k}(t)\psi_{j,l}(t)dt = 0, \tag{2.10}$$

for all appropriate $j, k, l \in \mathbb{Z}$.

Definition 2.3. *Let $\phi_{j_0,k}(t)$ be a family of scaling functions at scale j_0 and let $\psi_{j,k}(t)$ be the wavelet functions that span all of $L^2(\mathbb{R})$ space. Then the Discrete*

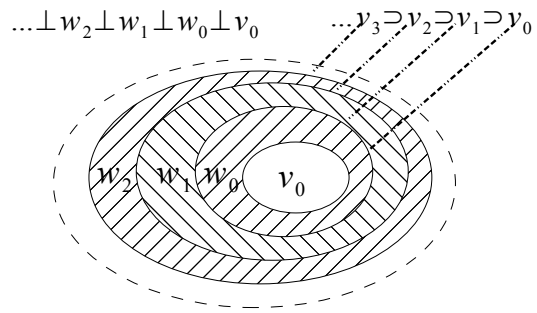


Figure 2.2: Scaling and wavelet vector spaces (see figure 2.3 in [9]). .

Wavelet Transform (DWT) of any function $f \in L^2(\mathbb{R})$ is given by,

$$f(t) = \sum_k c_{j_0,k}\phi_{j_0,k}(t) + \sum_{j=j_0}^{\infty}\sum_k d_{j,k}\psi_{j,k}(t), \tag{2.11}$$

where

$$\begin{aligned} c_{j_0,k} &= \langle f(t), \phi_{j_0,k}(t)\rangle = \int f(t)\phi_{j_0,k}(t)dt, \\ d_{j,k} &= \langle f(t), \psi_{j,k}(t)\rangle = \int f(t)\psi_{j,k}(t)dt. \end{aligned} \tag{2.12}$$

The choice of j_0 determines the coarsest scale whose space is spanned by $\psi_{j_0,k}(t)$. The coefficients $d_{j,k}$ in the expression (2.11) are called the Discrete Wavelet Transform (DWT) or the detail coefficients of the signal $f(t)$.
The DWT can be computed by the pyramidal approach found in [57] which is computationally fast and efficient compared to the Fast Fourier Transform (FFT). The DWT requires computational complexity $O(n)$, where $n = 2^J$ for some $J \in \mathbb{N}$ is the number of data points. The complexity of FFT is $O(n \log n)$.

2.2.2 Discrete Wavelet Transform (DWT): Pyramid algorithm

In here, we present a method proposed in [57]. If $\phi(t)$ is in $\mathcal{V}_0$, then it is also in $\mathcal{V}_1$. The space $\mathcal{V}_1$ is spanned by $\phi(2t)$, this means that $\phi(t)$ can be expressed as a weighted sum of translates of $\phi(2t)$, i.e.

$$\phi(t) = \sum_n h(n)\sqrt{2}\phi(2t-n), \qquad n \in \mathbb{Z}, \tag{2.13}$$

where the coefficients $h(n)$ are a sequence of real or complex numbers called the scaling filter coefficients [9] (page 13).
A general scaling function is therefore can be expressed as,

$$\phi(2^j t-k) = \sum_n h(n)\sqrt{2}\phi(2(2^j t-k)-n) = \sum_n h(n)\sqrt{2}\phi(2^{j+1}t-2k-n), \tag{2.14}$$

replacing $m = 2k+n$ we get,

$$\phi(2^j t-k) = \sum_m h(m-2k)\sqrt{2}\phi(2^{j+1}t-m). \tag{2.15}$$

The scaling coefficients, $c_{j,k}$, are calculated by the following inner product [9] (page 32),

$$\begin{aligned} c_{j,k} &= \langle f, \phi_{j,k}\rangle \\ &= \int f(t)2^{j/2}\phi(2^j t-k)dt, \end{aligned} \tag{2.16}$$

which, by using (2.15) and interchanging the sum and integral, can be written as,

$$c_{j,k} = \sum_m h(m-2k)\int f(t)2^{(j+1)/2}\phi(2^{j+1}-m)dt, \tag{2.17}$$

which leads to the following expression,

$$c_{j,k} = \sum_m h(m - 2k)c_{j+1,m}. \tag{2.18}$$

Similar derivation can be drawn for the detail coefficients (see [9, 57]),

$$d_{j,k} = \sum_m h_1(m - 2k)c_{j+1,m}, \tag{2.19}$$

where $h_1(n)$ are the detail filter coefficients.

The expressions (2.18) and (2.19) can be obtained by convolving the expansion coefficients $\mathbf{c}_{j+1}$ by the time reversed/mirrored recursion coefficients of $h(-n)$ and $h_1(-n)$ and then down-sampling or decimating the coefficients [9, 57] (page 33). Down-sampling or decimation means taking every other term (e.g. even terms). See figure 2.3. The pyramidal method finds $n/2$ detail coefficients at each scale

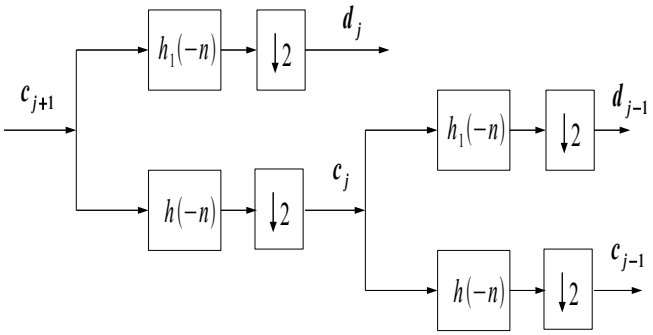
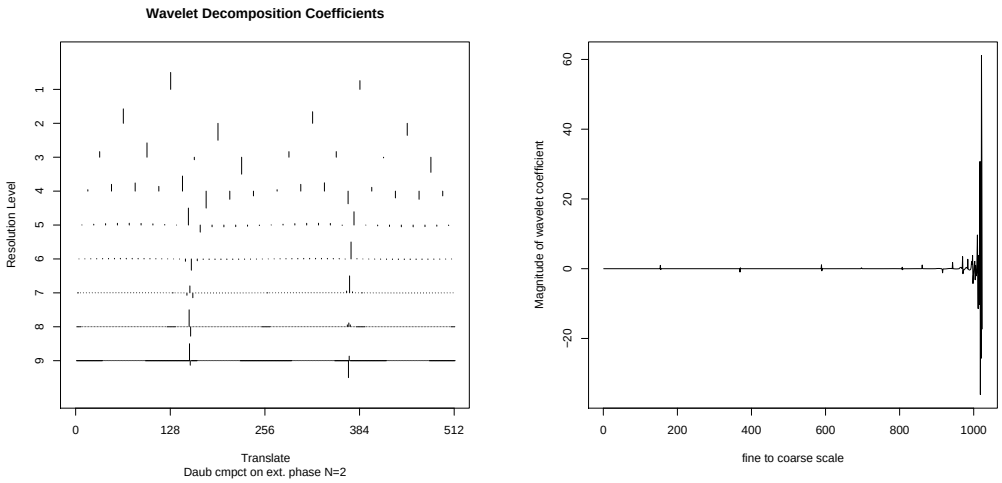


Figure 2.3: Two stages of the pyramid algorithm of [57] (see figure 3.3 in [9].

and it can run until there is no more decomposition possible, i.e the number of scaling coefficient available is one.

In order to demonstrate the sparsity (see section 5.14 for detail) of the DWT, we use a test function *heavisine*, which was used in the work [33]. We take the

number of samples $n = 1024$ and number of scales possible for this case is 10. Figure 2.4 shows the plot of wavelet coefficients (detail coefficients) using DWT implemented in *wavethresh* package. We can see majority of the coefficients are zero (965 out of 1023).



(a) Wavelet coefficients in each scale

(b) Wavelet coefficients plotted as a single vector (here we notice most of coefficients are zero and significant coefficients are at coarser scales)

Figure 2.4: Sparsity of DWT for *heavisine*. Figure on the left shows the detail coefficient at each scale. Figure on the right shows the plot of detail coefficients as a single vector (fine scale to coarse scale)

2.3 The lifting scheme

Lifting is a second generation wavelet technique introduced in [82]. Formal analysis of lifting scheme is presented in [83, 84]. Unlike traditional wavelets, so-called first generation wavelets, the lifting scheme does not rely on the Fourier transform. Therefore the lifting scheme can be used in situations where the Fourier transform is not defined. One such situation is that of irregularly spaced data which often occur in statistics.

The lifting transform consists of three stages, *split*, *predict*, and *update*. Lifting is an iterative process which is illustrated in figure 2.5.

Split: An abstract data set $\mathbf{c_0}$ can be split into two streams of data. One could say odd-indexed and even-indexed data, or by just splitting into two sets of data by separating them in the middle. This latter split is a bad way of splitting because the two sets of data will have much less correlation as the points in these subsets are far apart from each other and hence this will result in bad prediction. The aim of lifting is to have good compression of signals therefore one would like to choose the two subsets of data to be maximally correlated. For example, we could write,

$$\mathbf{c_J} = \mathbf{d_{J-1}} \cup \mathbf{c_{J-1}}, \tag{2.20}$$

where $\mathbf{c_{J-1}}$ is a vector which contains even-indexed elements in $\mathbf{c_J}$ and $\mathbf{d_{J-1}}$ is a vector which contains the odd-indexed elements in $\mathbf{c_J}$. The reason for splitting data in such a way is to increase the correlation between the subsets of data $\mathbf{c_{J-1}}$ and $\mathbf{d_{J-1}}$, because two adjacent elements in $\mathbf{c_J}$ will be more correlated than the ones that are far apart.

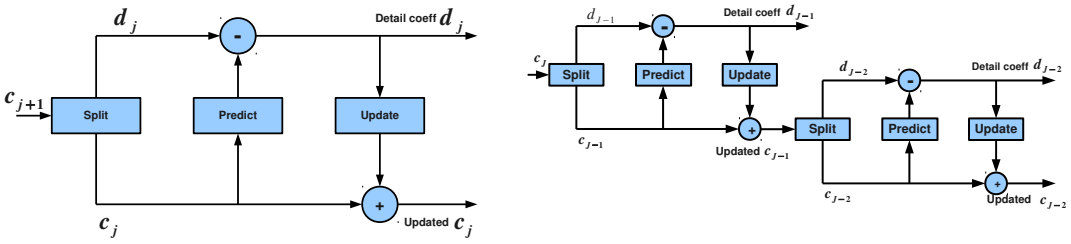
Predict: Find a prediction operator $\mathcal{P}$ independent of the data so that $\mathbf{c_j}$ could be used to predict $\mathbf{d_j}$. Then replace $\mathbf{d_j}$ with the residuals from this prediction. The residuals can be thought of as wavelet coefficients. In other words,

$$\begin{aligned} \tilde{\mathbf{d}}_\mathbf{j} &= \mathcal{P}(\mathbf{c_j}), \\ \mathbf{d_j} &:= \mathbf{d_j} - \tilde{\mathbf{d}}_\mathbf{j}. \end{aligned} \tag{2.21}$$

Update: After the prediction step, $\mathbf{c_j}$ needs to be carried forward to the next level $j-1$ and therefore for stability one has to find a suitable update operator $\mathcal{U}$ to preserve a certain scalar quantity, for example, expectation, $\mathbf{E}[\mathbf{c_j}] = \mathbf{E}[\mathbf{c_J}]$, which relates the original data to the scaling coefficients. The predict step can be written,

$$\mathbf{c_j} := \mathbf{c_j} + \mathcal{U}(\mathbf{d_j}). \tag{2.22}$$

After obtaining the forward lifting transform, it is easy to construct the inverse



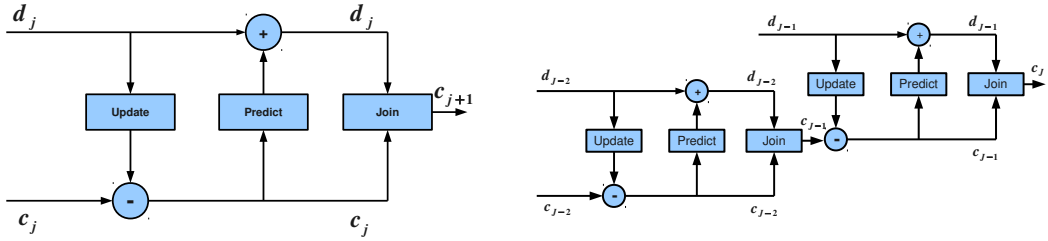
(a) Single Step (b) First Two Steps

Figure 2.5: Forward lifting transform.

lifting transform. Basically, the order is reversed and the signs are toggled. Easy inversion is one of the advantages of the lifting scheme over the traditional wavelet transforms:

$$\begin{aligned} \mathbf{c_j} &:= \mathbf{c_j} - \mathcal{U}(\mathbf{d_j}), \\ \mathbf{d_j} &:= \mathbf{d_j} + \mathcal{P}(\mathbf{c_j}), \\ \mathbf{c_{j+1}} &= \mathbf{d_j} \cup \mathbf{c_j}. \end{aligned} \tag{2.23}$$

The forward transform in illustrated in figure 2.5 and the inverse lifting transform is illustrated in figure 2.6.



(a) Single Step (b) Last Two Steps

Figure 2.6: Inverse lifting transform.

2.3.1 A simple example: Haar transform

We illustrate the lifting transform through the Haar example below (this example is taken from [46]). The abstract data set $c_J = \{x_m\}$ for $m = 1, 2, ...$ is split into even-indexed and odd-indexed data sets. At stage one the abstract data is split into two sets of data, $c_{J-1} = \{x_{2m}\}$, $d_{J-1} = \{x_{2m-1}\}$. We can now use elements in c_{J-1} to predict the elements in d_{J-1},i.e.

$$\tilde{x}_{2m-1} = x_{2m}. \tag{2.24}$$

The 'wavelet' coefficients are given by the residual from this prediction,

$$\begin{aligned} x^*_{2m-1} &= x_{2m-1} - \tilde{x}_{2m-1}, \\ &= x_{2m-1} - x_{2m}, \end{aligned} \tag{2.25}$$

where x^*_{2m-1} corresponds to the detail coefficients resulting from the lifting transform. The next step is to update the elements in c_{J-1}. This is done as follows,

$$\begin{aligned} x^*_{2m} &= x_{2m} + \frac{1}{2}x^*_{2m-1}, \\ &= \frac{1}{2}(x_{2m-1} + x_{2m}). \end{aligned} \tag{2.26}$$

By following the update step above maintains the mean of the data throughout the transform. Carry out the steps described above iteratively on x^*_{2m} until one element is left in the set c_0 and at this point one would have performed a complete lifting transform. Note that the lifting coefficients here are identical to the Haar wavelet coefficients.

To illustrate the above Haar transform we suppose that we have the set $c_3 = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\} = \{1, 1, 1, 1, -1, -1, -1, -1\}$. We split the data into an odd indexed set $d_2 = \{x_{2m-1}\} = \{1, 1, -1, -1\}$ and an even indexed set $c_2 = \{x_{2m}\} = \{1, 1, -1, -1\}$. Now we use the set c_2 to predict d_2 using (2.24)

and find the lifting coefficients at scale 2 according to (2.25). Therefore d_2 now becomes as, $d_2 = \{x^*_{2m-1}\} = \{0, 0, 0, 0\}$. Now perform updating on the set c_2 according to (2.26) which leads to, $c_2 = \{x^*_{2m}\} = \{1, 1, -1, -1\}$. We see no change in c_2 because of the detail coefficients in d_2 being zero.

Now we progress to the next scale where we start with the set $c_2 = \{x_1, x_2, x_3, x_4\} = \{1, 1, -1, -1\}$. After splitting we have the sets $d_1 = \{x_{2m-1}\} = \{1, -1\}$ and $c_1 = \{x_{2m-1}\} = \{1, -1\}$. Again we use the set c_1 to predict the elements in d_1 using (2.24) and find the lifting coefficients at scale 1 according to (2.25). Therefore the lifting coefficients are given by the set $d_1 = \{x^*_{2m-1}\} = \{0, 0\}$. After updating the set c_1 using (2.26) which again leads to no change in c_1.

Now we progress to the final scale of lifting where we start with the set $c_1 = \{x_1, x_2\} = \{1, -1\}$. Again split the data into odd indexed set $d_0 = \{1\}$ and even indexed set $c_0 = \{-1\}$. Using (2.24) and (2.25) we have the detail coefficient set $d_0 = \{2\}$. We get $c_0 = \{0\}$ by performing update on c_0 using (2.26).

2.4 Lifting One Coefficient At A Time (LOCAAT)

Based on the idea of lifting technique, the work in [45] introduces a new paradigm called "lifting one coefficient at a time", which we refer to as 'LOCAAT', to handle multidimensional irregular data.

The standard lifting transform, like the one considered in the example in the previous section, splits the data into two sets, odd- and even-indexed elements. However, the lifting one coefficient at a time scheme, as the name implies, calculates only one detail coefficient at each step. The following explanation is for spatially irregular data and taken entirely from [46].

We observe values $f_i = f(\mathbf{t}_i)$ of a function at n points or sites. Initially the

function can be approximated as follows

$$f(\mathbf{t}) = \sum_{k=1}^{n} c_{nk}\phi_{nk}(\mathbf{t}), \tag{2.27}$$

where ϕ_{nk} are scaling functions such that,

$$\phi_{nk}(\mathbf{t}_i) = \delta_{ik}, \tag{2.28}$$

where δ_{ik} is the Kronecker delta. The stages of the procedures are numbered downwards from n. So the order is, $n, n-1, n-2, ...$ At stage r, let $\mathcal{S}_r$ be the indices of the scaling coefficients and $\mathcal{D}_r$ be the indices of the detail coefficients. Initially the process starts with $\mathcal{S}_n = \{1, 2, ..., n\}$ and $\mathcal{D}_n = \emptyset$. At stage r, let $\mathcal{D}_r = \{i_{r+1}, i_{r+2}, ..., i_n\}$ be the indices of detailed coefficients already found. Note that the first removed point is the last element in the set. At stage r the function f can now be written as,

$$f(\mathbf{t}) = \sum_{l\in\mathcal{D}_r} d_l\psi_l(\mathbf{t}) + \sum_{k\in\mathcal{S}_r} c_{rk}\phi_{rk}(\mathbf{t}). \tag{2.29}$$

We will explain how the coefficients d_l and c_{rk} are calculated in the following subsection. The next stage is stage $r-1$ and if the next point to be removed is i_r then $\mathcal{S}_{r-1} = \mathcal{S}_r \backslash i_r$, $\mathcal{D}_{r-1} = \mathcal{D}_r \cup i_r$. Therefore the order in which the wavelet coefficients will be obtained is $i_n, i_{n-1}, ..., i_{r+1}$. Each removed point i_r is determined by the size of the integral of the scaling function, i.e. the scaling functions with smallest integrals will be removed first (see [46] for more detail).

2.4.1 Forward transform

For each i_r there are a number of neighbours n_r whose indices are stored in a set J_r. The values c_j for $j \in J_r$ will be used to construct an approximation for c_{i_r}. Each removed point i_r requires the definition of two vectors $\mathbf{a}^{i_r}$, $\mathbf{b}^{i_r}$ each of length

$|J_r|$. From here onwards, to avoid confusion in notation, i_r is replaced by i and J_r is replaced by J. Now the subscript r means the stage r in the multiresolution context.

Predict:

The detail coefficient d_i is obtained by the difference between the value at the removed point i and the sum of weighted values of its neighbours $j \in J$,

$$d_i = c_{ri} - \sum_{j \in J} a_j c_{rj}. \tag{2.30}$$

The weight vector $\mathbf{a}^i$ will depend on the particular lifting strategy. The lifting scheme we used to produce results in the next section employs inverse distance weights as the weight vector $\mathbf{a}^i$. This means that the larger the distance between the removed point i and its neighbours $j \in J$ the smaller the weight it carries. Therefore, when predicting the value at the removed point, if the distance is large, the impact of the neighbour is small.

$$\mathbf{a}^i = \left\{ \frac{\frac{1}{\text{dist}_j}}{\sum_{j \in J} \frac{1}{\text{dist}_j}} \right\} \qquad \forall j \in J. \tag{2.31}$$

From equation (2.31), the form of the weight vector $\mathbf{a}^i$ will ensure that it satisfies,

$$\sum_{j \in J} a_j = 1. \tag{2.32}$$

Update:

After obtaining the detail coefficient for the removed point i, the value of the neighbours are updated as follows,

$$c_{rj} := c_{rj} + d_i b_j, \qquad \forall j \in J. \tag{2.33}$$

Refer to [45] for the definition of the weight vector $\mathbf{b}^i$. Since this is lifting one coefficient at a time, the next level scaling function coefficients can simply be written as,

$$c_{r-1,j} = c_{rj}, \qquad \forall j \in \mathcal{S}_{r-1}. \tag{2.34}$$

The operations of predict and update steps are carried on recursively until the stage required or there are no more points to remove.

In order to demonstrate the sparsity of the LOCAAT transform, we use a test function *maartenfunc* (see section 3.2.3) with $n = 500$ and plot the detail coefficients in the order they are calculated in figure 2.7. Although the sparsity is less compared to the other wavelet transforms, we can still see that the fine scales have less significant coefficients than the coarser scales.

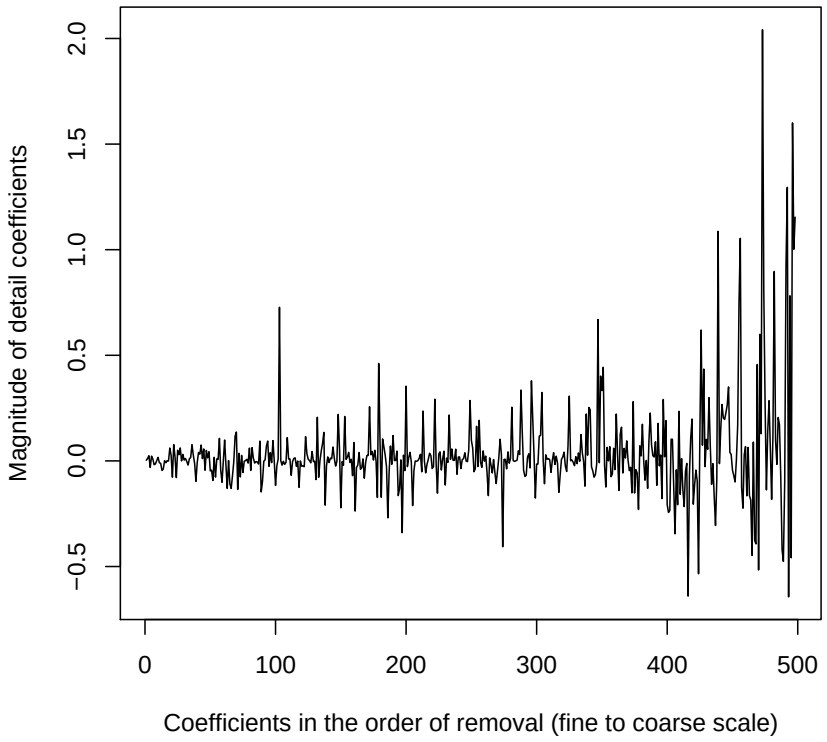


Figure 2.7: Plot of detail coefficients found using LOCAAT method for *maartenfunc*. The plot is in the order of removal and $n = 500$.

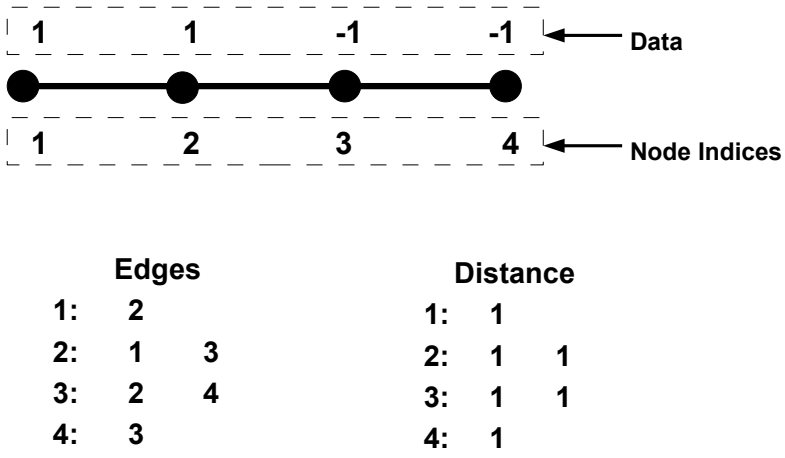


Figure 2.8: Toy network on which we assume the data $[1, 1, -1, -1]$ is observed. We also assume the nodes are separated by distance of 1 unit.

2.4.2 Inverse transform

Again the inverse is easy to construct by just reversing the order and changing the sign.

$$\begin{aligned} c_{rj} &:= c_{rj} - d_i b_j, \\ c_{ri} &:= d_i + \sum_{j \in J} a_j c_{rj}, \qquad j \in J. \end{aligned} \tag{2.35}$$

These steps are carried out recursively for all $i \in \mathcal{D}_{r-1}$ for complete reconstruction.

Numerical example of LOCAAT transform

We demonstrate a simple LOCAAT transform via the following example. Suppose we have a set of data $c_3 = \{1, 1, -1, -1\}$. In order to apply LOCAAT transform to this data, we have to first define a network on which the data is observed. Since this is a simple 1-D signal, we define the network as in figure 2.8. Once we have available requirements such as a defined network and the data, we can perform

the LOCAAT transform. First identify the first node to be removed which in this case is node 1. See [46] to find the argument for selecting the removed points. Node 1's neighbour is node 2 and the inverse distance weight $a_2 = 1$. Now by directly using (2.30) we get the LOCAAT coefficient $d_2 = 0$. Next step is to update the scaling coefficients. In this case we only have to update the node 2's value. Since the LOCAAT coefficient is zero, by using (2.33), we get unchanged value for node 2's observation. Therefore we end up with the next scale coefficients $c_2 = \{1, -1, -1\}$ and the detail coefficient set $d = \{0\}$ after first scale.

At the second scale, we start with the set $c_2 = \{1, -1, -1\}$. Now identify the second removed point which is node 4 in our example. Node 4's neighbour is node 3 and the inverse distance weight $a_3 = 1$. Again, by using (2.30), we get the LOCAAT coefficient $d_4 = 0$. We update the scaling coefficient (node 3's value) using (2.33). After the second scale, we end up with the next scale coefficients $c_1 = \{1, -1\}$ and the LOCAAT coefficient set $d = \{0, 0\}$.

Now we progress to the final lifting step. The last removed point in our example is node 2 whose neighbour is only node 3 because node 1 has already been removed in the earlier step (see [46] for detail on how the network is modified in each scale). We find the detail coefficient using (2.30) which is $d_2 = 2$. After performing update on the scaling coefficient (node 3's value) using (2.33) we get the scaling coefficient $c_0 = \{0\}$. After the complete transform we end up with the scaling coefficients $c_0 = \{0\}$ and the LOCAAT/detail coefficients $d = \{d_2, d_4, d_1\} = \{2, 0, 0\}$.

2.4.3 Variance approximation of the sample coefficients

Since the LOCAAT method operates linearly it is possible to calculate the covariance matrix for each lifting step. However with large data this is not computationally efficient. Suppose the original data c_k are independent random variables

with variances V_k. Consider a single prediction step of the lifting transform,

$$c_i^* = c_i - \sum_{j \in J} a_j c_j. \tag{2.36}$$

Therefore the modified variance after a single step, is given by,

$$\operatorname{var} c_i^* = V_i + \sum_{j \in J} a_j^2 V_j, \tag{2.37}$$

and the covariance,

$$\operatorname{cov}(c_i^*, c_j) = -a_j V_j, \qquad j \in J. \tag{2.38}$$

The update step for the lifting one coefficient at a time scheme is as follows,

$$c_j^* = c_j + c_i^* b_j, \tag{2.39}$$

and it follows that,

$$\begin{aligned} V_j^* &= V_j + b_j^2 \ \operatorname{var} c_i^* + 2 b_j \operatorname{cov}(c_i^*, c_j) \\ &= (1 - 2 a_j b_j) V_j + b_j^2 \operatorname{var} c_i^*. \end{aligned} \tag{2.40}$$

Therefore single step of the lifting transform approximates the variances V_k with V_k^*,

$$\begin{aligned} V_i^* &= V_i + \sum_{j \in J} a_j^2 V_j, \\ V_j^* &= (1 - 2 a_j b_j) V_j + b_j^2 V_i^*, \qquad j \in J. \end{aligned} \tag{2.41}$$

The approximation used is to neglect any correlation between the coefficients that are obtained in the next level [46]. This variance approximation will be used in estimating the noise variance and thus remove the noise by some means of thresholding (e.g hard thresholding using universal threshold). Refer to section 4.1 for how this variance approximation is used to estimate the noise variance in a denoising (noise removal) problem.

2.5 Denoising with regular wavelets

Most wavelet based function estimation is based on the model assumption expressed by the following equation,

$$f_i = g_i + \epsilon_i, \tag{2.42}$$

where ϵ_i is iid Gaussian with mean zero and variance σ_i^2. Generally the variance σ_i^2 is assumed to be constant and $i = 1, ..., n$. Wavelet based analysis usually require a regular grid, i.e. $t_i = i/n$ and often $n = 2^J$ for some $J \in \mathbb{N}$.
As mentioned earlier, data representation as wavelet coefficients d_i is often more sparse than its original representation. However, iid Gaussian noise is not compressed by the wavelet transform because the transform is orthogonal. So an effective denoising strategy is to,

- take the wavelet transform of noisy data
- the true function g_i usually gets compressed into a sparse wavelet representation. The noise ϵ_i remains 'spread evenly' across all coefficients.
- Since the wavelet transform is orthogonal it preserves energy, so the wavelet coefficients of the signal tend to 'stick out' above the noise more than with the original representation.

Hence a good denoising strategy is to threshold the wavelet coefficients [9, 32]. In the wavelet domain, the expression (2.42) can be written as,

$$d_i = \theta_i + \tilde{\epsilon}_i, \tag{2.43}$$

where d_i is the wavelet transform of f_i, θ_i is the wavelet transform of g_i and $\tilde{\epsilon}_i$ is the wavelet transform of the noise component ϵ_i.
We can apply the following thresholding methods to the wavelet coefficients d_i to

get an estimate for θ_i.

2.5.1 Hard, soft and universal thresholding

Now, to suppress noise, empirical wavelet coefficients are shrunk toward 0 by some shrinkage rule. Work in [32, 34] proposed two shrinkage methods, hard-thresholding and soft-thresholding, for suppressing the noise present in a signal.

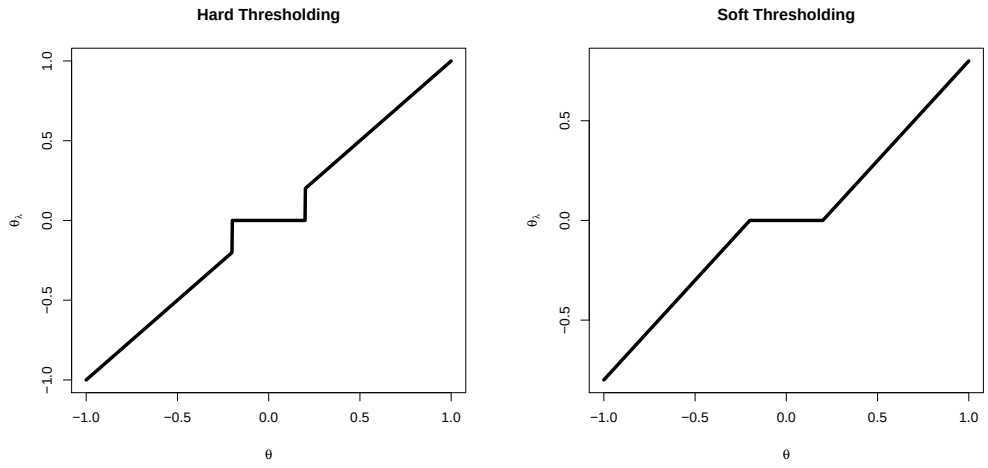
Hard thresholding:

$$\hat{\theta}_\ell^H = \begin{cases} d_\ell, & \text{if } |d_\ell| \geq \lambda, \\ 0, & \text{if } |d_\ell| < \lambda. \end{cases}$$

Soft thresholding:

$$\hat{\theta}_\ell^S = \begin{cases} \text{sgn}(d_\ell)(|d_\ell| - \lambda), & \text{if } |d_\ell| \geq \lambda, \\ 0, & \text{if } |d_\ell| < \lambda. \end{cases}$$

Figure 2.9 shows an example of both thresholding rules. One popular choice for



(a) Hard Thresholding (b) Soft Thresholding

Figure 2.9: Example of hard and soft thresholding. x-axis shows the coefficients θ and y-axis shows the thresholded coefficients θ_λ where $\lambda = 0.2$ is the threshold.

λ is the *universal threshold* [34],

$$\lambda = \sigma\sqrt{2\log n}, \tag{2.44}$$

where n is the number of wavelet coefficients and σ is the noise standard deviation.

2.5.2 Block thresholding

There is much literature for block thresholding methods in the wavelet context. See [10, 11, 12, 13, 15, 39, 40] for some popular block threshold methods for wavelets.

The discussion below assumes the model $f_i = g(t_i)+\epsilon_i$ where $i = 1, \ldots, n$, $t_i = i/n$, $n = 2^J$ for some $J \in \mathbb{N}$ and $\epsilon_i \sim N(0, \sigma^2)$. Let ϕ and ψ denote the scaling functions (also sometimes referred as father wavelets) and mother wavelets respectively. The discrete wavelet transform of function g is given below,

$$g(t) = \sum_{k=1}^{2^{j_0}} \epsilon_{j_0k}\phi_{j_0k}(t) + \sum_{j=j_0}^{\infty}\sum_{k=1}^{2^j} \theta_{jk}\psi_{jk}(t), \tag{2.45}$$

where $\epsilon_{jk} = \langle g, \phi_{jk}\rangle$ and $\theta_{jk} = \langle g, \psi_{jk}\rangle$. An empirical expansion of g is given by,

$$\tilde{g}(t) = \sum_{k=1}^{2^{j_0}} \tilde{\epsilon}_{j_0k}\phi_{j_0k}(t) + \sum_{j=j_0}^{J-1}\sum_{k=1}^{2^j} \tilde{\theta}_{jk}\psi_{jk}(t), \tag{2.46}$$

where J indicates the scale, $\tilde{\epsilon}_{jk} = \langle f, \phi_{jk}\rangle = \sum_i f_i\phi_{jk}(t_i)$ and $\tilde{\theta}_{jk} = \langle f, \psi_{jk}\rangle = \sum_i f_i\psi_{jk}(t_i)$. Since the DWT is an orthogonal transform it follows that $\tilde{\theta}_{jk} = \theta_{jk} + \epsilon_{jk}$ where $\epsilon_{jk} \sim N(0, \sigma^2)$. Now the term-by-term threshold estimator of $\tilde{g}$ is given by,

$$\hat{g}(t) = \sum_{k=1}^{2^{j_0}} \tilde{\epsilon}_{j_0k}\phi_{j_0k}(t) + \sum_{j=j_0}^{\infty}\sum_{k=1}^{2^j} \tilde{\theta}_{jk}\mathbb{I}(|\tilde{\theta}_{jk}| > \lambda)\psi_{jk}(t), \tag{2.47}$$

where $\mathbb{I}$ denotes the indicator function and λ is an appropriate threshold such as the universal threshold method proposed in [34], $\lambda = \sigma\sqrt{2\log n}$.

Work in [39] introduces a block thresholding wavelet estimator which thresholds coefficients in groups instead of term by term. This approach utilizes the fact that for a smooth g, θ_{j,k_1} and θ_{j,k_2} are close to each other in value if k_1 is close to k_2 and thus allowing decisions to be made on these coefficients by grouping these coefficients together for certain neighbouring values of k. This results in reduction of stochastic error in the estimator of θ_{jk} [40].

At each resolution level j, the empirical coefficients $\tilde{\theta}_{jk}$ are grouped into non-overlapping blocks of length $L = [(\log n)^2]$. Let $\mathscr{B}_{jb}$ denote the b^{th} block in resolution level j.

$$\mathscr{B}_{jb} = \{(j,k) : (b-1)L \leq k \leq bL\}. \tag{2.48}$$

Let $\tilde{B}^2_{jb} = \sum_{k \in \mathscr{B}_{jb}} \tilde{\theta}^2_{jk}$ denote the sum of squares of the empirical coefficients in block $\mathscr{B}_{jb}$. The block $\mathscr{B}_{jb}$ is considered important if $\tilde{B}^2_{jb} > T$ and all the coefficients within the block are retained, otherwise discarded. The work in [39] proposes the threshold value $T = \sigma^2 L\lambda$ where $\lambda \geq 48$. The block thresholded wavelet estimator of g is given by,

$$\hat{g}(t) = \sum_{k=1}^{2^{j_0}} \tilde{\epsilon}_{j_0 k}\phi_{j_0 k}(t) + \sum_{j=j_0}^{\infty}\sum_{b}\left(\sum_{k \in \mathscr{B}_{jb}} \tilde{\theta}_{jk}\psi_{jk}(t)\right)\mathbb{I}(\tilde{B}^2_{jb} > T). \tag{2.49}$$

A variant of this block thresholding procedure, BlockShrink, can be found in [11, 12]. The procedure is the same and the block length used is $L = [\log n]$ and $\lambda = 4.5052$ which attains the optimal rate of convergence. BlockShrink uses $T = 5L\sigma^2$.

2.5.3 NeighBlock and NeighCoeff

Work in [10] introduces two methods developed based on the block thresholding approach that was previously found to work well by incorporating information from neighbouring coefficients into decision making.

NeighBlock

At each resolution level j, group the empirical wavelet coefficients into disjoint blocks b_i^j of length $L_0 = [(\log n)/2]$. Each block is extended in both directions by further $L_1 = \max(1, L_0/2)$ coefficients to form an overlapping larger block B_i^j of length $L = L_0 + 2L_1$. The coefficients in block b_i^j are simultaneously thresholded depending on the coefficients in the larger block B_i^j.

$$\hat{\theta}_{j,k} = \beta_i^j \tilde{\theta}_{j,k} \qquad \forall (i,j) \in b_i^j, \tag{2.50}$$

where the shrinkage factor β_i^j is chosen in reference to the coefficients in the larger block B_i^j.

$$\beta_i^j = (1 - \lambda_* L\sigma^2 / S_{j,i}^2)_+, \tag{2.51}$$

where

$$S_{j,i}^2 = \sum_{(j,k)\in B_i^j} \tilde{\theta}_{j,k}^2, \tag{2.52}$$

where $\tilde{\theta}_{j,k}$ is the k^{th}empirical wavelet coefficient at scale j. The larger block B_i^j moves L_0 positions each time and half of the coefficients at the centre of the block is estimated.

NeighCoeff

The NeighCoeff procedure follows the same steps as the NeighBlock approach, but with $L_0 = L_1 = 1$, $L = 3$, and $\lambda = \frac{2}{3}\log n$. This again is a term by term thresholding but the threshold estimator is dependent on the coefficient itself and on its immediate neighbours. In NeighCoeff, a coefficient is estimated by zero when the sum of squares of the empirical coefficient and its immediate neighbours is less than $2\sigma^2 \log n$.

2.5.4 Stein's Unbiased Risk Estimator (SURE) threshold

Let $\boldsymbol{\mu} = (\mu_i : i = 1, \ldots, d)$ be a d-dimensional vector, and $x_i \sim N(\mu_i, 1)$ be a multivariate normal with that mean vector. Let $\hat{\boldsymbol{\mu}} = \hat{\boldsymbol{\mu}}(\mathbf{x})$ be a particular fixed estimator of $\boldsymbol{\mu}$. Work in [81] introduced a method to estimate the loss $||\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}||^2$ in an unbiased fashion.

Write $\hat{\boldsymbol{\mu}} = \mathbf{x} + \mathbf{g}(\mathbf{x})$, where $\mathbf{g} = (g_i)_{i=1}^d$ is a function form R^d into R^d. Stein [81] showed that when $\mathbf{g}(\mathbf{x})$ is weakly differentiable then,

$$E_{\boldsymbol{\mu}}||\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}||^2 = d + \mathbb{E}_{\boldsymbol{\mu}}\{||\mathbf{g}(\mathbf{x})||^2 + 2\boldsymbol{\nabla} \cdot \mathbf{g}(\mathbf{x})\}, \tag{2.53}$$

where,

$$\boldsymbol{\nabla} \cdot \mathbf{g}(\mathbf{x}) = \sum_i \frac{\partial}{\partial x_i} g_i. \tag{2.54}$$

The quantity $\mathrm{SURE}(\lambda; \mathbf{x}) = \frac{1}{d}[d - 2 \cdot \#\{i : |x_i| \leq \lambda\} + \sum_{i=1}^d (x_i \wedge \lambda)^2]$ is an unbiased estimate of the risk $E_{\boldsymbol{\mu}}||\hat{\boldsymbol{\mu}}^\lambda(x) - \boldsymbol{\mu}||^2 = \mathbb{E}_{\boldsymbol{\mu}}\,\mathrm{SURE}(\lambda; \mathbf{x})$ where $a \wedge b = \min(a, b)$. Work in [43] proposes a *SureShrink* estimator which selects the combination of threshold parameter, $\lambda^S = \arg\min_{0 \leq t \leq \sqrt{2\log d}} \mathrm{SURE}(t; \mathbf{x})$ and $\lambda = \sqrt{2 \log d}$ (see [43] for more details).

2.5.5 Cross validation based thresholding

The cross validation method is often used in statistical methods to optimize parameters by minimising error in approximation. In the wavelet context, the cross validation method is often used to find the optimal threshold parameter [43, 44, 63]. Many thresholding methods, such as the ones presented above, require a reliable estimation of noise variance σ^2. However cross validatory methods do not require it to be estimated. The cross validation method generally tries to estimate the MSE function. There are many cross validation methods however we present a method that is based on a popular leave-one-out cross validation.

Ordinary Cross Validation (OCV)

The following discussion is taken from [44]. The idea of OCV arises from the leave-one-out cross validation method. Suppose we have the model in (2.42), we simply leave a data point f_i out and interpolate (by linear combination of its neighbours) this point from the remaining data and call this $\tilde{f}_i$. Now DWT is applied to the modified vector $\mathbf{f}_i$,

$$\mathbf{f}_i = (f_1, \ldots, \tilde{f}_i, \ldots, f_n)^T, \tag{2.55}$$

where $n = 2^J$ for some $J \in \mathbb{N}$. The wavelet coefficients are then subjected to some threshold value λ and then inverted, we call this vector $\mathbf{f}_{\lambda i} = (f_{\lambda 1}, \ldots, \tilde{f}_{\lambda i}, \ldots, f_{\lambda n})$. The OCV score for point i for a particular λ is calculated

$$O_i(\lambda) = (\tilde{f}_{\lambda i} - f_i)^2. \tag{2.56}$$

This procedure is repeated for all the points and an overall cross validation score for a particular threshold λ is obtained,

$$O(\lambda) = \frac{1}{n} \sum_{i=1}^{n} (\tilde{f}_{\lambda i} - f_i)^2. \tag{2.57}$$

Generalised Cross Validation (GCV)

The above method is computationally intensive. Work in [44] shows that by taking $\tilde{f}_{\lambda i} = \tilde{f}_i$ will lead to the formula of GCV (see [44] for derivation). The GCV is given by,

$$G(\lambda) = \frac{\frac{1}{L}||\mathbf{f} - \mathbf{f}_\lambda||^2}{\left[\frac{L_0}{L}\right]^2}, \tag{2.58}$$

where L is the number of wavelet coefficients and L_0 is the number of these coefficients replaced by zero. Another way to calculate GCV is directly from

wavelet domain [43] as follows,

$$G(\lambda) = \frac{\frac{1}{L}||\mathbf{d} - \mathbf{d}_\lambda||^2}{\left[\frac{L_0}{L}\right]^2}, \tag{2.59}$$

where $\mathbf{d}$ is the vector of wavelet coefficients and $\mathbf{d}_\lambda$ is the thresholded coefficients.

2.5.6 Empirical Bayes thresholding

There are lot of references available related to Bayesian thresholding methods in wavelets (see [1, 23, 50, 51, 52]). The discussion below assumes the model $Z_i = g(\mathbf{t}_i) + \epsilon_i$, where the noise ϵ_i is independent $N(0, \sigma^2)$. Suppose we have a parameter θ and an observation $f_i \sim N(\theta, 1)$, where θ is the wavelet coefficient of g such that the empirical wavelet coefficient is rescaled to have unit variance. Work in [51, 52] suggested a prior distribution for θ_i each given by a mixture,

$$f_{\text{prior}}(\theta) = (1 - \pi)\delta_0(\theta) + \pi\gamma(\theta), \tag{2.60}$$

where γ is a symmetric density (work in [51, 52] suggested a heavy tailed density), π is the (prior) probability of a wavelet coefficient being non zero. Initially all θ_i have independent prior distributions (2.60) all with same value of π. Let $h = \gamma \star \phi$ where $\star$ denotes convolution and ϕ is the standard normal density. The marginal density of Z_i is,

$$(1 - \pi)\phi(z) + \pi h(z). \tag{2.61}$$

Work in [51, 52] explored Marginal Maximum Likelihood (MML) to select π, which is chosen to maximise the marginal log-likelihood,

$$\ell(\pi) = \sum_i \log\{(1 - \pi)\phi(z_i) + \pi h(z_i)\}. \tag{2.62}$$

For each observation $Z_i = z_i$ the posterior density, $f_{\text{post}(\theta_i|Z_i=z_i)}$, can be calculated. The estimator, $\hat{\theta}$, for the true wavelet coefficients θ can be derived through pos-

terior mean, posterior median or hard and soft thresholding methods (see [51, 52] for more detail).

In standard wavelet methods, a different prior π_j for each scale j is used . Typically π_j decrease as the resolution level increases. That is, at fine scale π_j is small and the observed coefficient must pass a high threshold in order to yield 0. At coarser scale, a smaller threshold is appropriate. See [46] for adaptation of empirical Bayes thresholding for the LOCAAT method.

2.6 Other smoothing methods

In this section we briefly describe some popular nonparametric smoothing methods other than wavelet methods.

2.6.1 Locally weighted polynomial regression

The work in [19] introduced a robust nonparametric method called LOcally WEighted Scatterplot Smoother (LOWESS) based on local polynomial fits. This method is robust in the sense that this method is resistant to outliers . This method produces a smooth curve through a scatter diagram, like linear regression in which the polynomial is a straight line, with certain properties. The curve is smooth and it minimises the variance of the residuals locally. The work in [20] extended the work in [19] to higher dimensions and from now we refer this method as LOESS.

The univariate case

Suppose f_i, are observations (dependent variable) and x_i are independent variables where $i = 1, \ldots, n$. The model for the regression is,

$$f_i = g(x_i) + \epsilon_i, \tag{2.63}$$

where $g(x)$ is a smooth function of x and ϵ_i are iid Gaussian with mean zero and variance σ^2. Let $W(u)$ for $u > 0$ be a weight function such that,

- $W(u) \geq 0$
- $W(u) = 0$ for $u > 1$, and
- $W(u)$ is non-increasing in $0 \leq u \leq 1$.

Let x be the point at which the estimate, $\hat{g}(x)$, of $g(x)$ is made. Let h be a number between 0 and 1 and let q be hn truncated to an integer. Let $d(x)$ be the distance of x to the qth nearest x_i. Let $w_i(x)$ be the weights defined by,

$$w_i(x) = W\left(\frac{|x_i - x|}{d(x)}\right), \qquad i = 1, \ldots, n. \tag{2.64}$$

To get $\hat{g}(x)$, a linear or a quadratic function (of the independent variable x) is fitted to the dependent variable f_i by weighted least squares with weights $w_i(x)$.

The multivariate case

The model for the multivariate case is similar to the univariate case case except the independent variable is now a vector with p elements for a p-dimensional case, i.e. $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$. Let ρ be a distance function in this space and let $d(x)$ be the distance of x to the qth nearest x_i. The weight $w_i(x)$ is defined as,

$$w_i(x) = W\left(\frac{\rho(x_i - x)}{d(x)}\right), \qquad i = 1, \ldots, n. \tag{2.65}$$

Again a linear or quadratic function of the independent variables is fit to f_i by weighted least squares with weights $w_i(x)$.

Choices

To carry out LOESS procedure one must decide between locally linear and locally quadratic fitting and must choose W, ρ, m. Work in [20] suggested that a good

W could be a tricube weight function,

$$W(u) = (1 - u^3)^3 \qquad \text{for } 0 \leq u \leq 1. \tag{2.66}$$

For ρ, the independent variables x are first scaled by dividing by its InterQuartile Range (IQR) and then use Euclidean distance for the scaled variables. Choosing m and deciding linear or quadratic function fitting are complex. As m increases, the neighbourhood size increases, the bias of $\hat{g}(x)$ increases, and the variance decreases. Therefore right choice of these parameters have to be made for a balanced bias/variance trade off.

This method is supposed to be computationally intensive since calculations have to be done for every x. However work in [21, 22] showed methods for doing the computations efficiently.

To summarise, the LOESS estimate is linear in f_i, i.e.,

$$\hat{g}(x) = \sum_{i=1}^{n} l_i(x) f_i, \tag{2.67}$$

where $l_i(x)$ is an element of the weight matrix L which is formed by $w_i(x)$. In R we can do LOESS or LOWESS by simply issuing commands `loess` or `lowess` with required parameters.

2.6.2 Smoothing splines

Polynomial smoothing splines were first introduced in [77, 79] and they have provided an attractive way of smoothing noisy data. A spline function is a curve constructed from polynomial segments that are subject to continuity at their joints (knots). A spline function, $S(x)$, is constructed to approximate $g(x)$ in the model given in (2.42). The spline function $S(x)$ must minimise the following criterion,

$$L = n^{-1} \sum \{f_i - S(x_i)\}^2 + \lambda \int_0^1 \{S''(x)\}^2 dx. \tag{2.68}$$

The parameter λ is known as the smoothing parameter.

2.6.3 Kernel smoothing

The kernel smoothing method approximates the function $g(x)$ in the model (2.42) by weighted average of the given data in a small window which is centred around x. The classical kernel estimator, also known as Nadaraya-Watson estimator, is in the form given by (see [62, 89]),

$$\hat{g}(x) = \frac{\sum_{i=1}^{n} f_i K\left(\frac{x-x_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right)}, \tag{2.69}$$

with kernel K and bandwidth h. Another popular method of kernel estimator is proposed in [75] and is given by,

$$\hat{g}(x) = \frac{1}{nh} \sum_{i=1}^{n} f_i K\left(\frac{x-x_i}{h}\right). \tag{2.70}$$

The bandwidth parameter h controls the smoothness or roughness of the estimator. It is well known that the choice of bandwidth is generally much more important than the choice of kernel (see [88]). Refer to the literature for more details on kernel smoothing.

2.6.4 Kriging

In the real world, it is practically impossible to get values at every physical points. Thus interpolation is desirable. The word "kriging" is synonymous with optimal prediction. The aim of kriging is to estimate the value of a random variable Z at one or more unsampled points or locations, from from given a support data, $\{z(\mathbf{x}_1), \ldots, z(\mathbf{x}_n)\}$, at locations $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. In kriging, the spatial data is modelled stochastically and then estimated using Best Linear Unbiased Estimator (BLUE) [16]. There are several types of kriging methods depending on the

variety of problems. We outline some popular kriging methods below.

Ordinary kriging

Suppose we have data $\{z(\mathbf{x}_1), \ldots, z(\mathbf{x}_n)\}$, at locations $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and we want to predict $Z(\mathbf{x}_0)$ at location $\mathbf{x}_0$. First, the ordinary kriging assumes $Z(\mathbf{x}) = \mu + \epsilon(\mathbf{x})$ where μ is the mean of $Z(\mathbf{x})$ and is an unknown constant. The quantity at location $\mathbf{x}_0$ is estimated as,

$$\hat{Z}(\mathbf{x}_0) = \sum_{i=1}^{n} \lambda_i Z(\mathbf{x}_i) + \epsilon(\mathbf{x}_0), \tag{2.71}$$

where λ_i is the weight for $Z(\mathbf{x}_i)$ and $\sum_{i=1}^{n} \lambda_i = 1$ and $E[\epsilon(\mathbf{x}_0)] = 0$. For each prediction at an unsampled location, appropriate weights λ_i have to be estimated. I.e.,

$$\{\hat{\lambda}_i\}_{i=1}^{n} = \arg\min E[\epsilon(\mathbf{x}_0)^2], \qquad \sum_{i=1}^{n} \hat{\lambda}_i = 1. \tag{2.72}$$

Simple kriging

Simple kriging and ordinary kriging are similar methods except the mean of $Z(\mathbf{x})$, μ, is a known constant.

Universal kriging

Universal kriging is a generalisation of ordinary kriging. In universal kriging $Z(\mathbf{x}) = \mu(\mathbf{x})$ and $\mu(\mathbf{x})$ is a linear combination of known functions $\{\phi_1(\mathbf{x}), \ldots, \phi_p(\mathbf{x})\}$. Therefore the model for this type of kriging is,

$$Z(\mathbf{x}) = \sum_{j=1}^{p} \phi_j(\mathbf{x})\beta_j + \epsilon(\mathbf{x}), \tag{2.73}$$

and the coefficients β_j have to be estimated from the data. Once this is performed, the estimation for unsampled location is similar to that in ordinary kriging.
These are few of the kriging methods available. Refer to the available literature for further detail on kriging (e.g. [41, 70, 71]). The work in [54] showed a method

for space-time function estimation using kriging update model.

2.7 Wavelet-based noise variance estimation methods

Most of the thresholding methods described earlier require estimation of the noise variance σ. In this section, we review some available methods to estimate noise variance from the wavelet coefficients.

2.7.1 Classical Median Absolute Deviation (MAD) estimate

Work in [34] proposes a robust estimator for σ by taking the Median Absolute Deviation (MAD) of the wavelet coefficients at the finest scale. This works well assuming the fine scale coefficients contain very little signal information which is a good assumption if the wavelet (lifting) transform of the true signal under consideration is sparse. Assume the following standard model used wavelet regression,

$$f_i = g(t_i) + \epsilon_i, \tag{2.74}$$

where $t_i = i/n; i = 1, \ldots, n,\ n = 2^J$ for some $J \in \mathbb{N}$ and ϵ_i are Gaussian iid with mean zero and variance σ^2. Let $\mathbf{f}$ denote the vector of observations f_i and $\mathbf{g}$ denote the vector of true signal values $g(t_i)$. The discrete wavelet transform of $\mathbf{f}$ results in the following wavelet domain decomposition.

$$\mathbf{w} = \boldsymbol{\theta} + \boldsymbol{\epsilon}, \tag{2.75}$$

where $\mathbf{w} \equiv W_n\mathbf{f},\ \boldsymbol{\theta} \equiv W_n\mathbf{g}$ and $\boldsymbol{\epsilon} \sim N(0, \sigma^2 I)$ because W_n is an orthogonal matrix. The vector $\mathbf{w}$ is of length 2^J and it contains scaling function coeffi-

cients and wavelet coefficients. The vector $\mathbf{w} = [(\mathbf{w}^*_{J_0})^T, \mathbf{w}^T_{J_0}, \ldots, \mathbf{w}^T_{J-1}]^T$, where $\mathbf{w}^*_{J_0} \equiv [w^*_{J_0,0}, \ldots, w^*_{J_0,2^{J_0}-1}]^T$ is the vector of scaling function coefficients and $\mathbf{w}_j \equiv [w_{j,0}, \ldots, w_{j,2^j-1}]^T$ is the vector of wavelet coefficients at the jth scale, $j = J_0, \ldots, J-1$. The MAD estimator of wavelet coefficients variance is given by,

$$\begin{aligned} \hat{\sigma}^{\text{MAD}} &= \text{MAD}\{w_{J-1,k}\} \\ &= \text{median}\{|w_{J-1,k} - \text{median}\{w_{J-1,k}\}|\}/0.6745, \end{aligned} \tag{2.76}$$

where the constant term $\frac{1}{0.6745}$ ensures consistency, i.e. $E[\text{MAD}(\mathbf{w})] = \sigma$.

2.7.2 Classical variogram method

The discussion below is taken from [25]. Let $\mathbf{s} \in \mathbb{R}^d$ be a generic data location in d-dimensional Euclidean space and suppose that the datum $Z(\mathbf{s})$ at spatial location $\mathbf{s}$ is a random quantity. Now let $\mathbf{s}$ vary over index set $D \subset \mathbb{R}^d$ so as to generate the multivariate random field (or random process).

$$\{Z(\mathbf{s}) : \mathbf{s} \in D\}. \tag{2.77}$$

There are potentially an infinite number of measurements that could be taken at location $\mathbf{s}$. A realisation of the random field $\{Z(\mathbf{s}) : \mathbf{s} \in D\}$ is denoted by $\{z(\mathbf{s}) : \mathbf{s} \in D\}$.

Suppose $\mu(\mathbf{s}) \equiv E(Z(\mathbf{s}))$ exists for all $\mathbf{s} \in D$. The existence of $\text{var}(Z(\mathbf{s}))$ for all $\mathbf{s} \in D$ allows the definition of second-order stationary and intrinsically stationary processes.

Second order stationarity

A process is second order stationary if its mean is constant and the autocovariance depends only on the lag.

$$E(Z(\mathbf{s})) = \mu, \qquad \forall \mathbf{s} \in D, \tag{2.78}$$

$$\begin{aligned} \operatorname{cov}(Z(\mathbf{s}_1), Z(\mathbf{s}_2)) &= C(\mathbf{s}_1 - \mathbf{s}_2) \qquad \forall \mathbf{s}_1, \mathbf{s}_2 \in D \\ &= C(\mathbf{h}), \text{say.} \end{aligned} \tag{2.79}$$

Calculation of the variogram

Suppose the spatial data $Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_n)$ is observed at spatial locations $D = \{\mathbf{s}_k : i = 1, \ldots, n\}$. Then *intrinsic stationarity* is defined through first differences:

$$\begin{aligned} E\left(Z(\mathbf{s}+\mathbf{h}) - Z(\mathbf{s})\right) &= 0, \\ \operatorname{var}\left(Z(\mathbf{s}+\mathbf{h}) - Z(\mathbf{s})\right) &= 2\gamma(\mathbf{h}). \end{aligned} \tag{2.80}$$

The quantity $2\gamma(\mathbf{h})$ is known as the variogram and $\gamma(\mathbf{h})$ is called the semi-variogram [25]. The classical estimator of the variogram was proposed in [59] and is given by,

$$2\hat{\gamma}(\mathbf{h}) = \frac{1}{|M(\mathbf{h})|} \sum_{M(\mathbf{h})} \left(z(\mathbf{s}_i) - z(\mathbf{s}_j)\right)^2, \tag{2.81}$$

where the sum is over $M(\mathbf{h}) \equiv \{(\mathbf{s}_i, \mathbf{s}_j) : \mathbf{s}_i - \mathbf{s}_j = \mathbf{h}; i, j = 1, \ldots, n\}$ and $|M(\mathbf{h})|$ is the number of distinct pairs in $M(\mathbf{h})$. The lag vector $\mathbf{h}$ that separates the two spatial locations $\mathbf{s}_i$ and $\mathbf{s_j}$ in both distance and direction is given by,

$$\mathbf{h} = h\mathbf{e}, \tag{2.82}$$

where h is the lag magnitude and $\mathbf{e}$ is the unit vector in the direction required.

Robust estimation of the variogram

In some situations the data contain outliers and, in others, the data can be highly skewed. For a Gaussian process $Z(.)$, $(Z(\mathbf{s}_i) - Z(\mathbf{s}_j))^2$ is a chi-squared random variable on 1 degree of freedom. The work in [24] finds the power transformation that makes the squared difference most Gaussian-like is the fourth root. The work in [24, 41] proposed the following robust estimator for the variogram based on mean and median operations to the transformed differences $\{|Z(\mathbf{s}_i)-Z(\mathbf{s}_j)|^{1/2} : (\mathbf{s}_i, \mathbf{s}_j) \in M(\mathbf{h})\}$,

$$2\hat{\gamma}^{\text{mean}}(\mathbf{h}) = \left\{ \frac{1}{M(\mathbf{h})} \sum_{M(\mathbf{h})} |Z(\mathbf{s}_i) - Z(\mathbf{s}_j)|^{1/2} \right\}^4 \Big/ B(\mathbf{h}), \tag{2.83}$$

$$2\hat{\gamma}^{\text{med}}(\mathbf{h}) = \left[\text{median}\{|Z(\mathbf{s}_i) - Z(\mathbf{s}_j)|^{1/2} : (\mathbf{s}_i, \mathbf{s}_j) \in M(\mathbf{h})\}\right]^4 \Big/ B(\mathbf{h}), \tag{2.84}$$

where $B(\mathbf{h}) = 0.457 + 0.494/|M(\mathbf{h})|$ and asymptotically $B(\mathbf{h}) = 0.457$.

2.7.3 A new variogram technique

The work in [42] proposes a variogram method for estimating σ arising from the model (2.74). They showed how to estimate σ in the time domain from the one dimensional case of $\mathbf{f} = \mathbf{g} + \boldsymbol{\epsilon}$, where $\mathbf{f}$ is the vector of observations f_i, $\mathbf{g}$ is the vector of true signal $g(t_i)$, $\boldsymbol{\epsilon}$ is the vector of ϵ_i, $t_i = i/n; i = 1, \ldots, n$, $n = 2^J$ for some $J \in \mathbb{N}$ and ϵ_i are Gaussian iid with mean zero and variance σ^2.

In the literature, the time lag is often denoted by the letter k. Since we use k to denote an arbitrary node, we use h to denote time lag.

Assuming $g(.)$ has a stationary covariance structure with autocovariance at lag h given by $C(h); h \in \mathbb{R}$ as follows,

$$\text{cov}(f_i, f_{i+h}) = \begin{cases} C(0) + \sigma^2, & \text{if } h = 0 \\ C(h), & \text{if } h \neq 0. \end{cases} \tag{2.85}$$

It follows that the variogram of $\mathbf{f}$ is given by,

$$\begin{aligned} 2\gamma(h) &\equiv \text{var}(f_{t+h} - f_t) \\ &= \begin{cases} 0, & \text{if } h = 0 \\ 2(\sigma^2 + C(0) - C(h)), & \text{if } h \neq 0. \end{cases} \end{aligned} \tag{2.86}$$

The semivariogram $\gamma(h) \to \sigma^2$ as $h \to 0$. They estimate σ^2 based on a linear extrapolation to the zero ordinate of the semivariogram at two small lags $0 < h_1 < h_2$, provided such an extrapolation is nondegenerate. Consequently, their estimator of σ^2 is $\hat{\sigma}^2$, where,

$$\hat{\sigma} \equiv \begin{cases} \left(\frac{h_2\hat{\gamma}(h_1) - h_1\hat{\gamma}(h_2)}{h_2 - h_1}\right)^{1/2}, & \text{if } h_2\hat{\gamma}(h_1) \geq h_1\hat{\gamma}(h_2) \geq h_1\hat{\gamma}(h_1) \\ \left(\frac{\hat{\gamma}(h_1) + \hat{\gamma}(h_2)}{2}\right)^{1/2}, & \text{if } \hat{\gamma}(h_2) < \hat{\gamma}(h_1) \\ 0, & \text{otherwise,} \end{cases} \tag{2.87}$$

and $2\hat{\gamma}(h) \equiv (\text{MAD}\{f_{i+h} - f_i\})^2$ is a robust estimator of the variogram at lag h. One may choose $h_1 = 1$ and $h_2 = 2$ for one-dimensional signals. The same idea could be extended to higher dimensions. For example, one may choose $h_1 = 1$ and $h_2 = \sqrt{2}$ for two-dimensional images [42].

2.8 Optimisation

In some of our work we need to optimise (minimising or maximising) a function, f (e.g. MSE function, efficiency function), with respect to one or more parameters. In R we can use built in functions `optimize` (for one parameter optimisation, in other words 1-D optimisation) or `optim` (more than one parameter optimisation, in other words higher dimensional optimisation). For more detail about using these functions issue `help(optimize)` or `help(optim)` in the R command window. Optimisation is a large field and it always challenging. There are many references available to read on optimisation techniques (see [7, 67, 74]). The func-

tion `optimize` use a combination of *golden section search* and *successive parabolic interpolation* methods in the optimisation. The function `optim` use a simplex method proposed in [66].

2.9 Sparsity

The sparsity in the wavelet literature refers to "very few coefficients contribute to signal" [34]. The time-frequency localisation of a signal can be quantified by the percentage of nonzero signal coefficients that occupy the time-frequency plane. If the signal has only a few nonzero wavelet coefficients then the signal must exist on a small percentage of wavelet coefficients. Such a signal is highly localised or has a high time-frequency localisation. In other words, the signal has a sparse wavelet transform. Conversely, if the signal exists on high percentage of wavelet coefficients, then the signal is not at all sparse.
The work in [73] defines the sparsity as follows,

Definition 2.4. *Let a signal have a discrete wavelet transform that is nonzero on P percent of the discrete wavelet transform coefficients, then the sparsity,*

$$sparsity = \frac{100 - P}{100}. \tag{2.88}$$

Sparsity defined above is normalised so that $0 \leq \text{sparsity} \leq 1$. A sparse signal will have a high sparsity. Gaussian random noise will have sparsity $= 0$.

Chapter 3

Methodology

3.1 Introduction

The main focus of this chapter is to develop a framework to be used in the later chapters on denoising. The methods presented in this chapter are analysed and compared computationally as it is difficult to develop a mathematical theory because of the complex nature of the lifting transform presented in [46]. It is intuitive that there are spatial and temporal patterns present in the observations of network data over time and we would like to exploit them in our function estimation problem. We denote the noisy observation by $f_{t,k}$, where $t = 1, \ldots, T$ are time points and $k = 1, \ldots, n$ are nodes. Initially, spatial patterns have been considered (denoising each time step separately) and later this spatial model is extended to exploit temporal patterns along with spatial patterns.

As an example, if we consider a Wireless Sensor Networks (WSN), the spatio-temporal irregularity of the nodes arises from spatially irregular deployments of sensor nodes. The fundamental reasons behind spatially irregular configuration is that the event of interest is not always spatially uniform and it is not feasible to deploy the sensor nodes uniformly because of the nature of sensor node deployment strategies, which vary according to the application [3, 4].

For irregularly spatially distributed nodes, the nodes that are close to each other are often more closely related when compared to the ones that are far apart. To take this into account, inverse distance weights are used in the lifting algorithm following [46].

Only a static network setting has been considered throughout this thesis, but the methods presented here could be altered for a dynamic network which is usually the case for wireless communication networks. The results presented in this thesis are based on the following two paradigms,

- Spatial paradigm: Considers spatial patterns (depending on neighbours) only and repeats, in isolation for each time instance separately.
- Spatio-temporal paradigm: Considers both spatial and temporal patterns simultaneously.

3.2 The spatial model

The Lifting One Coefficient At A Time (LOCAAT) algorithm developed in [46] uses two pieces of information. Firstly, it requires a network structure as a list containing edges and distances. Secondly, the data/observations to be transformed (the values at each node). We will start by giving a brief description of a network.

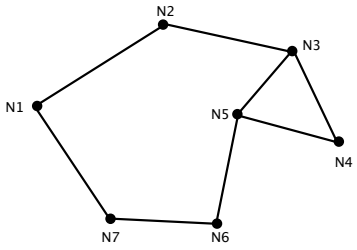
3.2.1 Network information for spatial network

A network is defined by a set of nodes and a set of edges that link nodes. Let (k, j) denote the edge between node k and j and let δ_{kj} be the distance between node k and node j. The algorithm developed in [46] uses a tree based network structure. Each time a node is removed due to the lifting step, the network tree is then updated into a complete connected tree using the Minimum Spanning Tree (MST) technique.

Figure 3.1 shows a simple seven node network, and its connectivity, to illustrate

the spatial case. For each node k there are n_k neighbours. Each node is connected to other nodes via an edge. The set of edges eliminated from a node k is denoted by J_k which is the set of destination nodes. The collection of sets J_k for all the nodes is also known as adjacency list.

In this thesis we assume the nodes have coordinates (x, y) as it is easier to visualise the results. However the LOCAAT method does not require coordinates. It simply requires a distance measure between the edges. We consider the edge distance δ_{kj} as Euclidean distance, i.e. $\delta_{kj} = \sqrt{(x_k - x_j)^2 + (y_k - y_j)^2}$, where (x_k, y_k) are the coordinates of node k. Neighbourhood information for communicating nodes



(a) Simple Network

Adjacency List (Edges)

1:	2	7	
2:	1	3	
3:	2	4	5
4:	3	5	
5:	3	4	6
6:	5	7	
7:	1	6	

(b) Edges

Figure 3.1: Simple network and its edge entries

can be obtained from their routing table (e.g routing table of Ad-hoc On-demand Distance Vector (AODV)) in the case of a wireless network or the linkage information if it is a wired network.

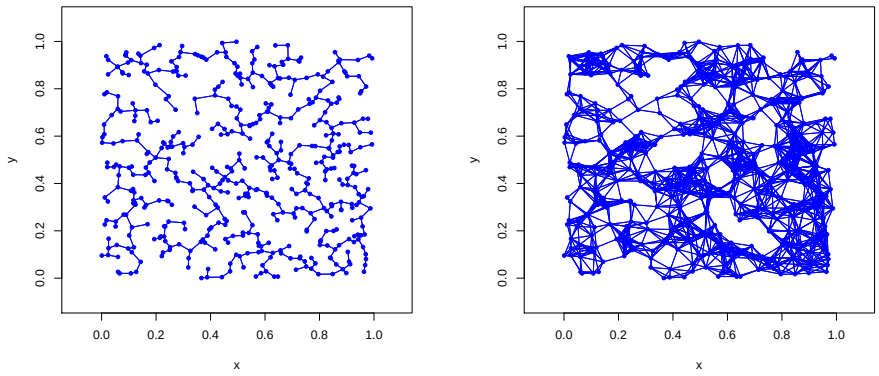
We assume the following model for the observations, $f_k = g_k + \epsilon_k$ where $k = 1, \ldots, n$ indicates the node index and ϵ_k is iid Gaussian with mean zero and variance σ_k^2. Time snapshots are treated separately, and therefore temporal correlations are not considered in this model. If we have to denoise a spatio-temporal data using this paradigm then we consider the time snapshot of the network data at each time instance separately, and perform denoising sequentially.

3.2.2 Network scenarios used for simulation studies

We use two types of network scenarios, which closely model certain communication networks, in our simulation studies.

Minimum Spanning Tree (MST) type network

The nodes are placed uniformly at random in a unit square. The neighbours of nodes are determined by the MST. Figure 3.2(a) shows an example of this type of network. We refer to these networks as type-1 or T-1 for short. This type of network can be found in wireless or wired networks. These type of networks normally work on some kind of distance vector algorithm (see [6, 53] for some tree based routing).



(a) Type-1 Network - MST

(b) Type-2 Network - Circular disk approach, radius = 0.09

Figure 3.2: Two types of networks used in our simulation studies. $n = 500$

Circular disc approach

For any node k we place an imaginary circle with radius r whose center is node k, and nodes that lie within the circle are considered to be the neighbours of node k. Figure 3.2(b) shows an example of this type of network. We refer to these networks as type-2 or T-2 for short. These type of networks can occur in wireless mesh networking (see [6, 53] information on mesh networking).

3.2.3 Test functions

We use several test functions in our simulation studies. We use the superscript to differentiate between the test functions as the subscript is already in use for time and node identification. Figure 3.3 shows the plots of some test functions with discontinuity.

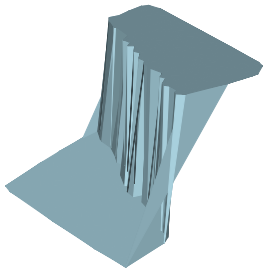
$$\begin{aligned} g^1 &\equiv g(x,y) = (2x+y)\mathbb{I}((3x-y)<1) + (10-x)\mathbb{I}((3x-y)\geq 1) \\ g^2 &\equiv g(x,y) = -0.5 - y + 3(x-0.5)^2 + (1+0.2\sin(2\pi x))\mathbb{I}(y > -(0.5-x)^2 + 0.5) \\ g^3 &\equiv g(x,y) = (2x+y)\mathbb{I}(3x-y<1) + (5x-y) * \mathbb{I}(3x-y\geq 1) \\ g^4 &\equiv g(x,y) = -2(x-0.5)^2 - 2(y-0.5)^2 + \mathbb{I}((x-0.5)^2 + (y-0.5)^2 \leq (0.25)^2) \end{aligned} \tag{3.1}$$

We also use some spatially smooth test functions (without discontinuity) defined below for some time domain estimation of the variogram methods proposed in section 4.3. Figure 3.4 shows the contour plot of these smooth functions.

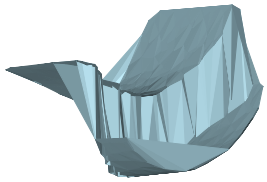
$$\begin{aligned} g^5 &\equiv g(x,y) = 6 - 12x + 12x^2 + 3y - 5y^2 \\ g^6 &\equiv g(x,y) = -0.5 - y + 3(x-0.5)^2 + (1+0.2\sin(2\pi x)) \\ g^7 &\equiv g(x,y) = x^2 + y^2 \\ g^8 &\equiv g(x,y) = 4x + y \end{aligned} \tag{3.2}$$

In addition to the test functions defined above, we also use some of the 2D analogues of Donoho and Johnstone test functions Doppler (figure 3.5(a)), Heavisine (figure 3.5(b)), Bumps (figure 3.5(d)) and Blocks (figure 3.5(c)) [34, 47]. We also use a test function called `maartenfunc` used in [46] (figure 3.4(c)). See appendix D for the R codes of 2-D analogues of Donoho and Johnstone test functions and `maartenfunc` which can be found in [47]. Each subfigure in figure 3.5 is plotted using the following R command,

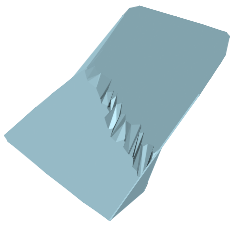
`image(interp(x,y,g))`.



(a) g^1



(b) g^2



(c) g^3(maartenfunc)

(d) g^4

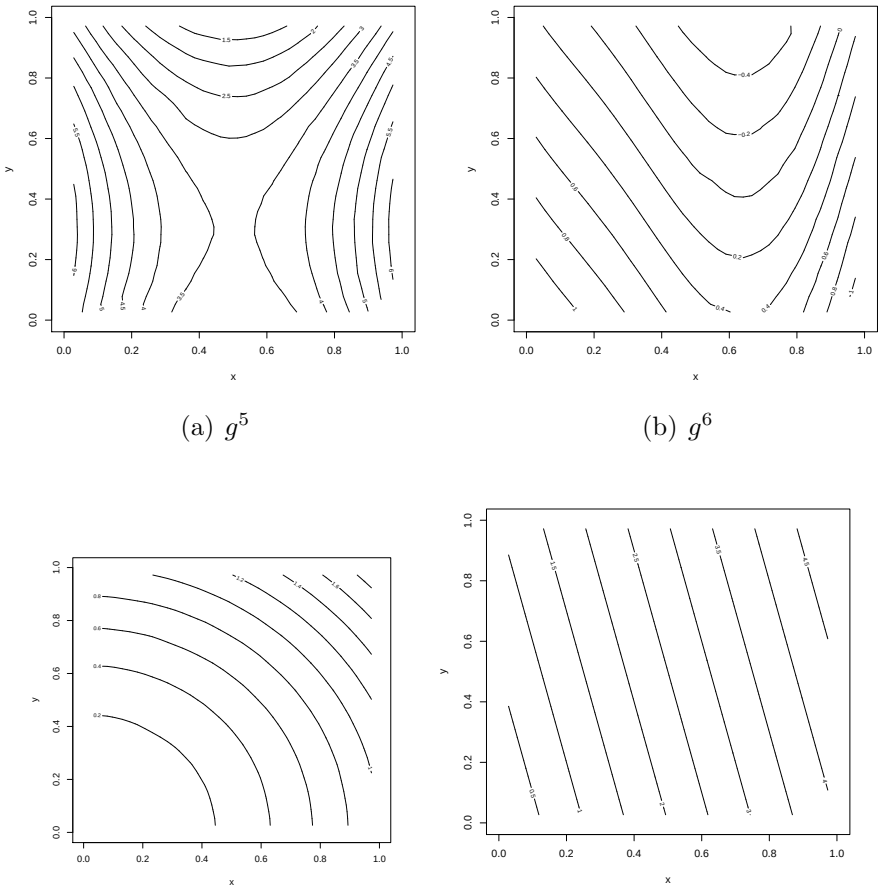
Figure 3.3: Some test functions with spatial discontinuity.

3.3 The spatio-temporal model

The spatio-temporal analysis models the temporal correlations and therefore we would expect a performance improvement by using this method for denoising and forecasting. The model is:

$$f_{t,k} = g_{t,k} + \epsilon_{t,k}, \qquad t = 1, \ldots, T, k = 1, \ldots, n, \tag{3.3}$$

where $f_{t,k}$ is the observation at node k at time t, $g_{t,k}$ is the 'unknown' true function at node k at time t, $\epsilon_{t,k}$ is iid Gaussian error with mean zero and variance σ^2. There are T time observations available. Our aim is to estimate $g_{t,k}$. We produce



(a) g^5 (b) g^6

(c) g^7(maartenfunc) (d) g^8

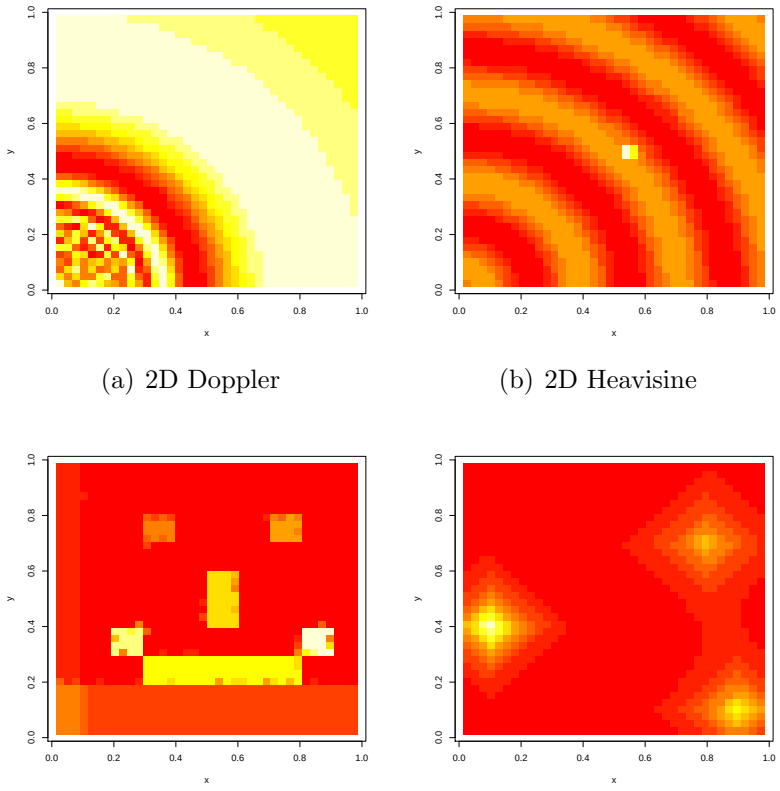
Figure 3.4: Smooth test functions without spatial discontinuity.

$g_{t,k}$ as a time varying function of one of our test functions, g^1 through g^4, defined in the previous section.

We are interested in denoising $f_{t,k}$ through thresholding its lifting coefficients therefore we should not consider a separable case such as the following,

$$g_{t,k} = g(x_k, y_k) + CH(t), \tag{3.4}$$

where $g(x_k, y_k)$ is one of the test functions from previous section, C is a constant and $H(t)$ is the time varying function which could be one of *one-dimensional* Donoho and Johnstone test functions Doppler (figure 3.6(a)), Heavisine (figure 3.6(b)), Bumps (figure 3.6(d)) or Blocks(3.6(c)). This separable case can



(a) 2D Doppler (b) 2D Heavisine

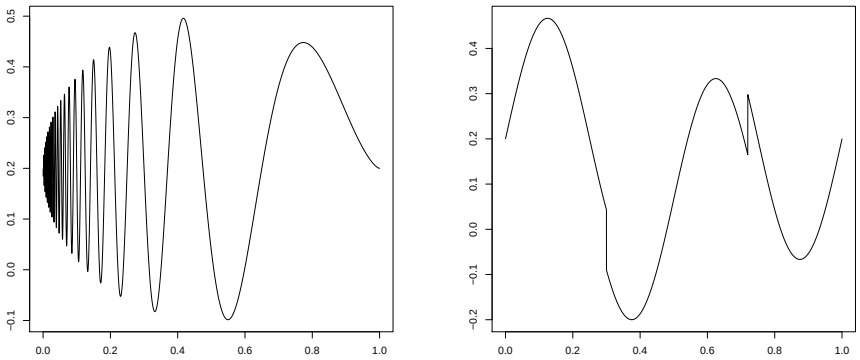
(c) 2D Blocks (d) 2D Bumps

Figure 3.5: Some test functions from [47], n = 1000

be denoised without having to transform or threshold. See appendix A for derivation.

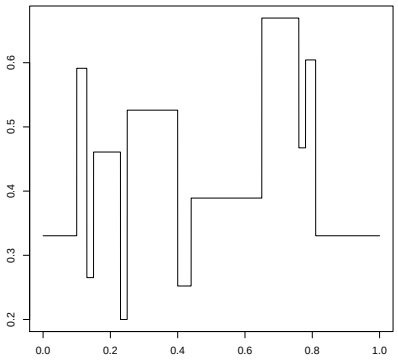
3.3.1 Spatio-temporal networks: notations and basics

This section describes building a spatio-temporal network which enables us to consider temporal correlations along with spatial correlations. We will only consider a wired/fixed network scenario whose neighbourhood structure does not change over time. In a wireless setting, the topology changes over time, and new nodes come into existence and some nodes disappear because of failures and/or they are out of reach. This dynamic nature is typical in a wireless network. In order to take this dynamic behaviour into account one may have to modify the following

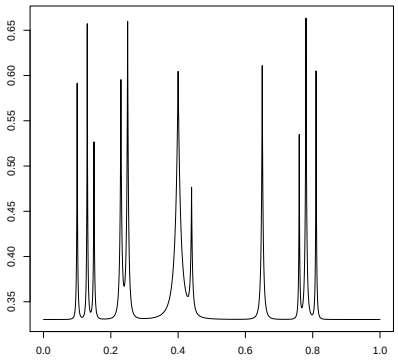


(a) Doppler

(b) Heavisine



(c) Blocks



(d) Bumps

Figure 3.6: Donoho Johnstone test functions

network descriptions (this is a future direction).

In spatio-temporal case, node k after time t will have a new node index in our way of indexing and is denoted by, $((t-1)n+k)$. Therefore the initial nodes will have a new index set, $\mathcal{N}_t = \{((t-1)n+k)|\forall k \in \mathcal{N}_1\}$. This way, over all time points, the total number of nodes will be Tn and the new node indices will be $1, 2, \ldots, Tn$. We consider distance along spatial dimension as Euclidean distance and the distance in the time dimension as the minimum distance in the spatial dimension multiplied by some temporal scale factor l_t. This scale factor is chosen to maintain any irregularity in the time domain. Let $\delta^{\min}$ denote the minimum spatial edge distance for all the nodes, i.e. $\delta^{\min} = \min\{(\delta_{kj})_{j\in J_k, k\in\mathcal{N}_1}\}$. We will

consider two cases for building a spatio-temporal network.

Case-1: New neighbours (copies from immediate past and future)

Now copy the network as many times as the number of time entries and allocate the indices (node ids) according to $\mathcal{N}_t$. In our spatio-temporal methods, there are additional neighbours, generally two, for each node. These additional neighbours are the nodes themselves at a previous time step and the next time step. Only the nodes at the starting and ending layer of the network will have only one additional neighbour. See figure 3.7. The seven node example shown in figure 3.8(a) has been extended into a spatio-temporal network following the steps described above (figure 3.8). Future developments could extend this method to include copies of itself at more distant time steps.

The edge distances between the nodes $(t-1)n+k$ and $tn+k$ (edge distance

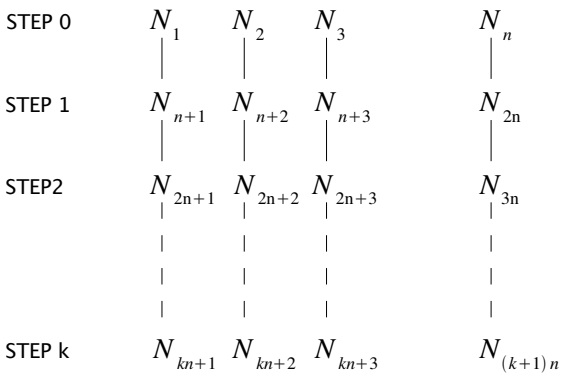
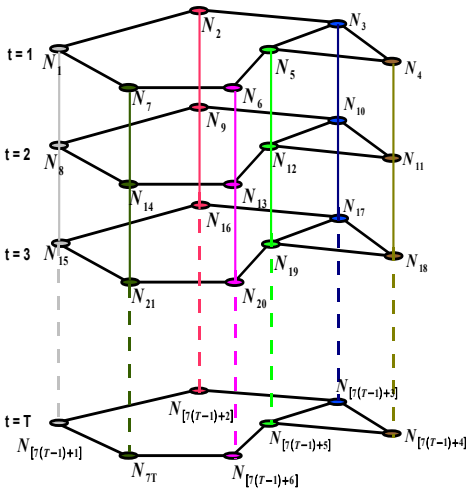
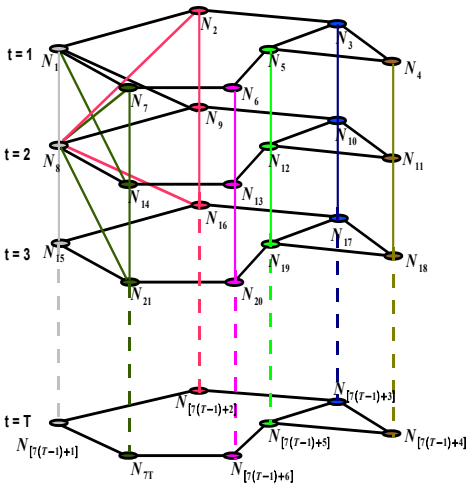


Figure 3.7: Extending the spatial network to the spatio-temporal network



(a) Network connection along time: case-1



(b) Network connection along time: case-2

Adjacency List

1:	2	7	8			
2:	1	3	9			
3:	2	4	5	10		
4:	3	5	11			
5:	3	4	6	12		
6:	5	7	13			
7:	1	6	14			
8:	1	9	14	15		
9:	2	8	10	16		
10:	3	9	11	12	17	
11:	4	10	12	18		
12:	5	10	11	13	19	
13:	6	12	14	20		
14:	7	8	13	21		

(c) Edges for the spatio-temporal network (truncated list): case-1

Adjacency List

1:	2	7	8	9	14						
2:	1	3	8	9	10						
3:	2	4	5	9	10	11	12				
4:	3	5	10	11	12						
5:	3	4	6	10	11	12	13				
6:	5	7	12	13	14						
7:	1	6	8	13	14						
8:	1	2	7	9	14	15	16	21			
9:	1	2	3	8	10	15	16	17			
10:	2	3	4	5	9	11	12	16	17	18	19
11:	3	4	5	10	12	17	18	19			
12:	3	4	5	6	10	11	13	17	18	19	20
13:	5	6	7	12	14	19	20	21			
14:	1	6	7	8	13	15	20	21			

(d) Edges for the spatio-temporal network (truncated list): case-2

Figure 3.8: Spatial network extended to spatio-temporal network

between the nodes at current state and future state) is fixed to be,

$$\delta_{(t-1)n+k,tn+k} = l_t \delta^{\min}. \tag{3.5}$$

The edge entries in figure 3.8, show only the edges for two time steps but the full edge list is much longer for the amalgamated network. Let $J_{(t-1)n+k}$ denote the

set of neighbours for a node $(t-1)n+k$ in the new amalgamated network and is given by,

$$J_{(t-1)n+k} = \begin{cases} \{\{(j)_{j\in J_k}\} \cup \{n+k\}\} & \text{if } t = 1 \\ \{\{((t-1)n+j)_{j\in J_k}\} \cup \{((t-2)n+k)\} & \\ \cup \{(tn+k)\}\} & \text{if } 1 < t < T \\ \{\{((t-1)n+j)_{j\in J_k}\} \cup \{(t-2)n+k\}\} & \text{if } t = T. \end{cases} \tag{3.6}$$

Case-2: New neighbours (copies and their neighbours)

We copy the network T times with the updated node indices given by $\mathcal{N}_t = \{((t-1)n+k)|\forall k \in \mathcal{N}_1\}$. Each node $(t-1)n+k$ will not only have itself in the past and the future as its new neighbours (case-1), but also have the spatial neighbours' past and future states as the neighbours,

$$J_{(t-1)n+k} = \begin{cases} \{\{(j)_{j\in J_k}\} \cup \{n+k\} \cup \{(n+j)_{j\in J_k}\}\} & \text{if } t = 1 \\ \{\{((t-1)n+j)_{j\in J_k}\} \cup \{((t-2)n+k), (tn+k)\} & \\ \cup \{((t-2)n+j)_{j\in J_k}\} \cup \{(tn+j)_{j\in J_k}\}\} & \text{if } 1 < t < T \\ \{\{((t-1)n+j)_{j\in J_k}\} \cup \{(t-2)n+k)\} & \\ \cup \{((t-2)n+j)_{j\in J_k}\}\} & \text{if } t = T \end{cases} \tag{3.7}$$

Figure 3.8(b) shows this type of spatio-temporal network. In this figure we only show the neighbours of node 1 for first two time steps as it may affect the readability if we connect more nodes for more time instances. Distance between the nodes $(t-1)n+k$ and $tn+k$ (distance between the node at current state and future state), $\delta_{(t-1)n+k,tn+k}$ is calculated as in (3.5). Since the time layers are parallel to each other, if we connect a node k in two separate time instances, this edge will be normal to the time layers. Therefore the distance between the node at current state, $(t-1)n+k$, and the neighbours' future state, $(tn+j)_{j\in J_k}$ is calculated by Pythagoras' theorem,

$$\delta_{(t-1)n+k,(tn+j)_{j\in J_k}} = \sqrt{(l_t\delta^{\min})^2 + \delta_{kj}^2}. \tag{3.8}$$

3.3.2 Non-overlapping moving window

Initial experiments with the spatio-temporal method by considering all the time points together, shows bad results compared to spatial-only method although we expected the spatio-temporal method to give better results. Therefore we adopt a moving window based spatio-temporal method. Experiments have shown that window size of $M_t = 3$ gives good results.

In more detail, we create a combined network as before but this time with the depth of M_t time slices. Therefore the new node indices will be $1, 2, \ldots, M_t n$. Edges and distances are determined as explained in the previous section (either

case-1 or case-2). Figure 3.9(a) shows an example network with $M_t = 3$.
We rearrange the double indexed observations $f_{t,k}$ into a single indexed long vector, $\mathbf{f}^{\text{obs}}$, with elements $f_{(t-1)n+k}$. We then slice the vector $\mathbf{f}^{\text{obs}}$ into non-overlapping blocks of $M_t n$. Let $\mathcal{B}_j$ denote the j^{th} block,

$$\mathcal{B}_j = \{(j-1)M_t n + 1, \ldots, jM_t n\} \qquad j = 1, \ldots, \lfloor T/M_t \rfloor \tag{3.9}$$

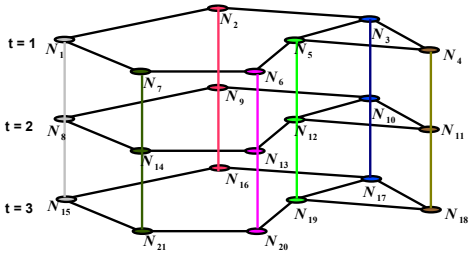
Now we perform LOCAAT transform on $(f_i^{\text{obs}})_{i \in \mathcal{B}_j}$.

3.3.3 Overlapping moving window

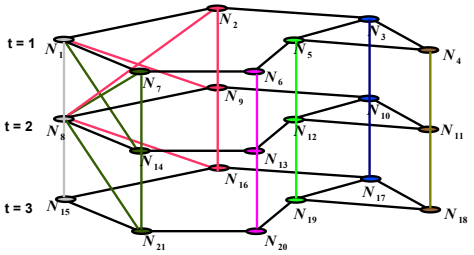
Another way to find a moving window is to move a window of M_t time slices 1-step at a time as opposed to M_t-steps in the previous moving window approach. Here the network remains the same as the non-overlapping case but the way we calculate LOCAAT coefficients differs. The set of indices of the block of observations, on which the LOCAAT transform is performed, is given by,

$$\mathcal{B}_j = \{(j-1)n + 1, \ldots, (j + M_t - 1)n, \}, \qquad j = 1, \ldots, T - M_t + 1. \tag{3.10}$$

We perform the LOCAAT transform on the overlapping blocks $(f_i^{\text{obs}})_{i \in \mathcal{B}_j}$. When we process, invert the coefficients and rearrange, we may have up to M_t estimates for $g_{t,k}$. Therefore we can take the average to have the improved estimate $\hat{g}_{t,k}$.



(a) Spatio-temporal network: Case-1



(b) Spatio-temporal network: Case-2

Adjacency List

1:	2	7	8			
2:	1	3	9			
3:	2	4	5	10		
4:	3	5	11			
5:	3	4	6	12		
6:	5	7	13			
7:	1	6	14			
8:	1	9	14	15		
9:	2	8	10	16		
10:	3	9	11	12	17	
11:	4	10	12	18		
12:	5	10	11	13	19	
13:	6	12	14	20		
14:	7	8	13	21		
15:	8	16	21			
16:	9	15	17			
17:	10	16	18	19		
18:	11	17	19			
19:	12	17	18	20		
20:	13	19	21			
21:	14	15	20			

(c) Edges-Case1

Adjacency List

1:	2	7	8	9	14						
2:	1	3	8	9	10						
3:	2	4	5	9	10	11	12				
4:	3	5	10	11	12						
5:	3	4	6	10	11	12	13				
6:	5	7	12	13	14						
7:	1	6	8	13	14						
8:	1	2	7	9	14	15	16	21			
9:	1	2	3	8	10	15	16	17			
10:	2	3	4	5	9	11	12	16	17	18	19
11:	3	4	5	10	12	17	18	19			
12:	3	4	5	6	10	11	13	17	18	19	20
13:	5	6	7	12	14	19	20	21			
14:	1	6	7	8	13	15	20	21			
15:	8	9	14	16	21						
16:	8	9	10	15	17						
17:	9	10	11	12	16	18	19				
18:	10	11	12	17	19						
19:	10	11	12	13	17	18	20				
20:	12	13	14	19	21						
21:	8	13	14	15	20						

(d) Edges-Case2

Figure 3.9: Window based approach, $M_t = 3$

Chapter 4

Noise Variance Estimation for Networks

4.1 Introduction

One main focus of this thesis is nonparametric function estimation for networks via wavelet thresholding methods. In order to estimate a function from noisy observations $\{f_k\}_{k=1}^n$ or $\{f_{t,k}\}_{t=1,k=1}^{T,n}$, we need to reliably estimate the noise variance σ^2. In nearly all wavelet shrinkage problems estimation of σ is crucial for success [65]. If σ is over-estimated then wavelet shrinkage will not only remove noise, but also remove some important features of the signal. We consider the following scenarios under which we have to estimate σ,

- spatial observations $\{f_k\}_{k=1}^n$ and no time series data available.
- the case with $\{f_{t,k}\}_{t=1,k=1}^{T,n}$ but T is not large, for example $T = 3, T = 5$, etc.
- finally we investigate $\{f_{t,k}\}_{t=1,k=1}^{T,n}$ with large T

We investigate several methods to estimate σ according to the scenarios above. Before we begin our investigation, we need to familiarize ourselves with the LOCAAT transform method introduced in [46].

Recall from section 2.4 that the LOCAAT algorithm produces detail coefficients in reverse order. After the complete LOCAAT transform the detail coefficient index set is $\mathcal{D}_1 = \{i_2, i_3, \ldots, i_n\}$, and the scaling coefficient index set is $\mathcal{S}_1 = \{i_1\}$. Let $\mathbf{d}_r^{\text{detail}} = (d_\ell)_{\ell \in \mathcal{D}_r}$ be the vector of detail coefficients and L be the number of detail coefficients. At intermediate stage r the number of coefficients will be $n-r$. After the complete transform, i.e $r = 1$, the number of coefficients is $L = n - 1$.
After obtaining lifting (wavelet) coefficients, these coefficients can be thresholded so that pure noise is killed with high probability. Various thresholding methods are described in chapter 5. If the noise is normally distributed with variance equal to one, we could use the threshold $p\sqrt{2\log n}$ where p is a proportion which is $p = 1$ for a universal threshold, and $p < 1$ for some fixed percentage of the universal threshold, which has been found to work well in practice [48] for standard wavelet models.
Since the LOCAAT transform is non-orthogonal, the noise variance of the detail coefficients is no longer σ^2 as it is modified by the transform. Refer to section 2.4.3 for details of the change in variance due to lifting steps. For a node k, the variance change due to the LOCAAT transform can be calculated and we denote it by $\overline{\sigma}_k^2$. The original data in the network domain has the following representation,

$$f_k = g_k + \epsilon_k, \tag{4.1}$$

where f_k is the observation at node k, g_k the function to be estimated and ϵ_k is iid Gaussian noise with mean zero and variance σ^2.
After applying the lifting transform, the network domain representation of the signal, as in (4.1), is transformed into the 'wavelet' domain and can be written as follows,

$$d_k = \theta_k + \tilde{\epsilon}_k, \tag{4.2}$$

where d_k is the lifted version of f_k, θ_k is the lifted version of the true function g_k and $\tilde{\epsilon}_k$ is the lifted version of the noise term ϵ_k. The noise term is now distributed as,

$$\tilde{\epsilon}_k \sim N(0, \overline{\sigma}_k^2 \sigma^2), \tag{4.3}$$

but not independently. Here $\overline{\sigma}_k$ is the node dependent variance change factor due to the LOCAAT transform which can be approximated through the method explained in section 2.4.3. Dividing (4.2) by the known variance factor $\overline{\sigma}_k$ we can ensure that the noise term will be approximately normally distributed with mean zero and variance σ^2 but still correlated, i.e.

$$\begin{aligned} \frac{d_k}{\overline{\sigma}_k} &= \frac{\theta_k}{\overline{\sigma}_k} + \frac{\hat{\epsilon}_k}{\overline{\sigma}_k}, \\ d_k^* &= \theta_k^* + \epsilon_k^*, \\ \epsilon_k^* &\overset{\text{approx}}{\sim} N(0, \sigma^2), \end{aligned} \tag{4.4}$$

where $d_k^* = d_k/\overline{\sigma}_k$, $\theta_k^* = \theta_k/\overline{\sigma}_k$ and $\epsilon_k^* = \epsilon_k/\overline{\sigma}_k$. Our main focus for the remainder of this chapter is to estimate the quantity σ.

We repeat the experiments carried out in this chapter with a fixed (**SNR!**). We define SNR as,

$$\text{SNR} = \frac{sd(g)}{\sigma}, \tag{4.5}$$

where,

$$\text{sd}(g)^2 = \text{var}(g) = \frac{1}{n-1} \sum_{k=1}^{n} \{g(x_k, y_k) - \bar{g}\}^2, \tag{4.6}$$

where $\bar{g} = \frac{1}{n} \sum_{k=1}^{n} g_k$. Although the signal g itself is not stochastic, $\text{var}(g)$ should be thought of as the signal variance.

4.2 MAD for LOCAAT

Work by Donoho and Johnstone [34] showed that a robust estimator for the unknown σ could be achieved by taking Median Absolute Deviation (MAD) for the fine scale wavelet coefficients. See section 2.7.1 to find out how this MAD operator works in wavelet domain. In this section we will be looking at ways of applying MAD techniques for the LOCAAT technique.

4.2.1 Global MAD

In our problem, we take the LOCAAT transform of network data and the wavelet domain decomposition is given in (4.2). Then we divide (4.2) by the known variance factor $\overline{\sigma}_k$ to get the set of coefficients $\{d_k^*\}_{k=1}^n$. The LOCAAT algorithm finds one lifting coefficient per scale and each coefficient has a unique scale. As there is no uniform 'finest scale' containing half of the wavelet coefficients as in regular wavelets, taking MAD of the finest scale coefficients to estimate σ does not make sense. Therefore, we take MAD of lifting coefficients of the first half of removed points in order. We denote the vector of fine-scale coefficients as $\mathbf{d}^{\text{fine}}$ given by,

$$\mathbf{d}^{\text{fine}} = \left(d_{i_r}^*\right)_{r=\lfloor \frac{L}{2} \rfloor + 1}^{L}, \qquad i_r \in \mathcal{D}_1 \tag{4.7}$$

Now the estimate for σ is given by,

$$\begin{aligned} \hat{\sigma}^{\text{MAD}} &= \text{MAD}\{\mathbf{d}^{\text{fine}}\} \\ &= \frac{\text{median}\{|\mathbf{d}^{\text{fine}} - \text{median}\{\mathbf{d}^{\text{fine}}\}|\}}{0.6745} \end{aligned} \tag{4.8}$$

We can apply this MAD approach to the both spatial and spatio-temporal LOCAAT coefficients. By spatial LOCAAT coefficients we mean the LOCAAT transformation of spatial data purely based on the spatial network connections. We transform spatio-temporal data using the spatio-temporal method described in section 3.3.1 and the resultant coefficients are called spatio-temporal LOCAAT

coefficients.

Table 4.1 shows some results of estimating σ using the classical MAD approach for some different test functions with SNR = 2. In order to compare various methods, we present the results as a percentage deviation from the true σ. Each entry in the table is calculated as follows,

$$\frac{\hat{\sigma} - \sigma}{\sigma} \times 100, \tag{4.9}$$

where $\hat{\sigma}$ is the estimator of the true σ, which is an average over 100 independent experiments. It is clear from the table that the estimation of σ gets better as the number of nodes increases. It is also clear that estimation on a type-2 network does not work as well compared to type-1 network.

Table 4.2 shows the results for estimating σ on the spatio-temporal LOCAAT

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{MAD}}$		True σ	$\hat{\sigma}^{\text{MAD}}$		True σ	$\hat{\sigma}^{\text{MAD}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	10.5	16.1	2.02	8.0	11.3	2.06	2.47	7.0
g^2	0.23	10.5	44.1	0.20	2.9	13.0	0.22	3.1	8.2
g^3	0.71	4.0	11.9	0.58	3.1	3.6	0.62	2.0	1.9
g^4	0.27	4.7	14.1	0.25	9.3	15.3	0.28	3.1	6.0

Table 4.1: Estimation of σ(% difference) using MAD for different test functions based on spatial LOCAAT coefficients, SNR=2. Each entry in the table shows an average of 100 repetitions.

coefficients. In this simulation we used the time depth of $M = 3$ to construct the network. Comparing tables 4.1 and 4.2 it is clear that both T-1 and T-2 spatio-temporal networks are giving better results compared to the estimation on the spatial LOCAAT coefficients (refer table 4.1) and the estimation gets better as the number of nodes increases as expected. The estimation based on T-2 network gives marginally better estimation compared to T-1.

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{MAD}}$		True σ	$\hat{\sigma}^{\text{MAD}}$		True σ	$\hat{\sigma}^{\text{MAD}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	7.4	7.4	2.02	3.8	1.3	2.06	1.5	0.8
g^2	0.23	7.3	10.1	0.20	3.1	0.0	0.22	0.1	2.1
g^3	0.71	6.5	1.1	0.58	-0.7	-1.1	0.62	0.6	0.8
g^4	0.27	4.5	4.0	0.25	0.3	0.2	0.28	1.1	0.6

Table 4.2: Estimation of σ(% difference) using MAD for different test functions based on spatio-temporal (network constructed with time depth $M = 3$) LOCAAT coefficients, SNR=2. Each entry in the table shows an average of 100 repetitions.

4.2.2 Local MAD for LOCAAT

In the last section we saw how the global MAD estimator is used with LOCAAT coefficients for the noise variance estimation. MAD estimator works better when there are larger numbers of nodes. We have seen that the global MAD estimator is not so good at lower numbers of nodes. In this section we extend the idea of MAD to node level. We can estimate the node dependent noise variance σ_k provided the network has sufficient number of *neighbours* for each node such as T-2 network. Our main aim in this chapter is to find a MAD estimator to reliably estimate σ for networks with a small number of nodes. We demonstrate this situation with the T-2 network where the neighbours are found using a circular range from a node. Later, we modify this MAD estimator slightly to use it with the T-1 network.

Let $\mathbf{f}$ be the vector of observations $\{f_k\}_{k=1}^n$ and let $\mathbf{d}$ be the LOCAAT transform of $\mathbf{f}$. Note that the vector $\mathbf{d}$ has scaling function coefficients $\mathbf{c} = (d_\ell)_{\ell \in \mathcal{S}_1}$ and detail coefficients given by $\mathbf{d}^{\text{detail}} = (d)_{\ell \in \mathcal{D}_1}$.

For any node k, there are a number of neighbours n_k, whose indices are denoted by a set J_k. The node dependent σ_k is estimated by taking MAD of the detail coefficients of the node itself and the detail coefficients of its neighbours. Let $\mathcal{A}_k$ be the set that stores the indices of the lifting coefficients of node k and its neighbours, in the order they were calculated, and L_k be the cardinality of $\mathcal{A}_k$.

Since the vector d has both scaling function coefficients and the detail coefficients, the node index k or some, or all, elements of the set J_k are not necessarily contained in the detail coefficient index set $\mathcal{D}_1$. Therefore we have to make sure the set $\mathcal{A}_k$ contains only the indices of the detail coefficients and not the index/indices of the scaling coefficients. So,

$$\mathcal{A}_k = \{J_k \cup k\} \cap \mathcal{D}_1 = \{i_1^k, \ldots, i_{L_k}^k\}. \tag{4.10}$$

Now we define a new index set, $\mathcal{A}_k^{\text{fine}}$, that stores the indices of the fine scale coefficients for this node,

$$\mathcal{A}_k^{\text{fine}} = \{i_1^k, \ldots, i_{\lfloor \frac{L_k}{2} \rfloor}^k\}, \qquad i_1^k \in \mathcal{A}_k. \tag{4.11}$$

We define the node dependent fine-scale coefficients as

$$\mathbf{d}_k^{\text{fine}} = \{d_i^* : i \in \mathcal{A}_k^{\text{fine}}\}. \tag{4.12}$$

We estimate the node dependent estimate of σ_k as

$$\hat{\sigma}_k^{m1} = \text{MAD}\{\mathbf{d}_k^{\text{fine}}\}. \tag{4.13}$$

In the case of the constant σ model, we can estimate the node dependent σ_k (which should be roughly equal to σ). Then we take the arithmetic mean of $\hat{\sigma}_k^{\text{m1}}$ over all the nodes to find $\hat{\sigma}^{\text{m1}}$.

$$\hat{\sigma}^{\text{m1}} = \frac{1}{n} \sum_{k=1}^{n} \hat{\sigma}_k^{\text{m1}}. \tag{4.14}$$

We propose this method because the global MAD approach fails to give a good estimate for σ with small numbers of nodes. By following the local MAD approach described above we artificially increase the information about the signal, although

the coefficients are correlated, and thus hope that it would give a better estimate for σ. We call the method described above as M-1. Now we present another way, M-2, of artificially increasing the knowledge about the signal.

We obtain the finest scale coefficient per node and then estimate σ by taking MAD of all these coefficients. Let $\mathcal{A}$ denote the set that stores the indices of one fine scale coefficient per node,

$$\mathcal{A} = \{i_1^k : k = 1, \ldots, n\}, \qquad i_1^k \in \mathcal{A}_k^{\text{fine}}. \tag{4.15}$$

The estimator using this approach, $\hat{\sigma}^{\text{m2}}$, is given by,

$$\hat{\sigma}^{\text{m2}} = \text{MAD}\{(d_i^*)_{i\in\mathcal{A}}\}. \tag{4.16}$$

Table 4.3 shows estimates of σ for the type-1 network, while table 4.4 shows the results for type-2 network. It is expected that the M-1 method will fail for T-1 network since there are not enough neighbours per node to perform the MAD operation. By comparing tables 4.1 and 4.3, the estimator following method M-2 gives a better estimation for T-1 network as the number of nodes increases. For a T-2 network, by comparing tables 4.1 and 4.4, the local MAD approach M-1 gives better results with few number of nodes and M-2 gives marginally better results as the number of nodes increases, compared to the global MAD estimator.

Table 4.5 shows the results of estimating σ using the spatio-temporal method with the time depth $M = 3$ for T-1 network. It is clear from the table that M-2 performs well and the estimation gets better with increasing number of nodes. Table 4.6 shows the results for T-2 network with similar settings. It is clear from this table that M-2 outperforms all the methods above for a T-2 network case.

Signals	n = 50			n = 150			n = 500		
	True σ	T-1		True σ	T-1		True σ	T-1	
		$\hat{\sigma}^{\mathrm{m1}}$	$\hat{\sigma}^{\mathrm{m2}}$		$\hat{\sigma}^{\mathrm{m1}}$	$\hat{\sigma}^{\mathrm{m2}}$		$\hat{\sigma}^{\mathrm{m1}}$	$\hat{\sigma}^{\mathrm{m2}}$
g^1	2.08	-84.5	9.8	2.02	-81.8	8.0	2.06	-81.2	1.6
g^2	0.23	-87.7	11.2	0.20	-83.7	1.8	0.22	-82.3	1.6
g^3	0.71	-87.2	3.9	0.58	-84.6	1.9	0.62	-83.2	2.3
g^4	0.27	-85.9	7.7	0.25	-82.3	8.1	0.28	-81.8	3.2

Table 4.3: Estimation of σ(% difference) using M-1 and M-2 for a T-1 network in a spatial setting (lifting coefficients produced using spatial network), SNR=2. Each entry in the table shows an average of 100 repetitions.

Signals	n = 50			n = 150			n = 500		
	True σ	T-2		True σ	T-2		True σ	T-2	
		$\hat{\sigma}^{\mathrm{m1}}$	$\hat{\sigma}^{\mathrm{m2}}$		$\hat{\sigma}^{\mathrm{m1}}$	$\hat{\sigma}^{\mathrm{m2}}$		$\hat{\sigma}^{\mathrm{m1}}$	$\hat{\sigma}^{\mathrm{m2}}$
g^1	2.08	5.27	12.1	2.02	-5.4	19.9	2.06	-9.2	8.8
g^2	0.23	8.8	16.7	0.20	-8.8	3.5	0.22	-9.9	6.4
g^3	0.71	-12.8	-4.4	0.58	-13.5	0.7	0.62	-15.2	0.4
g^4	0.27	-5.4	-4.0	0.25	-0.9	15.7	0.28	-8.7	3.6

Table 4.4: Estimation of σ(% difference) using M-1 and M-2 for a T-2 network in a spatial setting (lifting coefficients produced using spatial network), SNR=2. Each entry in the table shows an average of 100 repetitions.

Signals	n = 50			n = 150			n = 500		
	True σ	T-1		True σ	T-1		True σ	T-1	
		$\hat{\sigma}^{\mathrm{m1}}$	$\hat{\sigma}^{\mathrm{m2}}$		$\hat{\sigma}^{\mathrm{m1}}$	$\hat{\sigma}^{\mathrm{m2}}$		$\hat{\sigma}^{\mathrm{m1}}$	$\hat{\sigma}^{\mathrm{m2}}$
g^1	2.08	-27.0	5.8	2.02	-31.5	0.6	2.06	-30.3	0.1
g^2	0.23	-27.7	3.2	0.20	-30.4	0.2	0.22	-30.1	-0.6
g^3	0.71	-30.1	3.2	0.58	-32.4	-2.4	0.62	-31.7	0.1
g^4	0.27	-27.9	1.2	0.25	-28.8	-2.3	0.28	-30.6	-0.3

Table 4.5: Estimation of σ(% difference) using M-1 and M-2 for a T-1 network in a spatio-temporal setting (lifting coefficients produced using spatio-temporal network with time depth M=3), SNR=2. Each entry in the table shows an average of 100 repetitions.

4.3 Variogram method for LOCAAT

The variogram method was first introduced by [59]. Further work by [24, 41] introduced the robust estimator of the variogram based on the mean and the

Signals	n = 50			n = 150			n = 500		
	True σ	T-2		True σ	T-2		True σ	T-2	
		$\hat{\sigma}_{m1}$	$\hat{\sigma}_{m2}$		$\hat{\sigma}_{m1}$	$\hat{\sigma}_{m2}$		$\hat{\sigma}_{m1}$	$\hat{\sigma}_{m2}$
g^1	2.08	-13.4	5.3	2.02	-12.8	1.0	2.06	-14.7	1.6
g^2	0.23	-12.9	0.1	0.20	-14.2	-3.6	0.22	-15.1	0.4
g^3	0.71	-19.5	-7.4	0.58	-16.8	-1.9	0.62	-16.2	0.2
g^4	0.27	-20.0	1.4	0.25	-12.0	-4.6	0.28	-14.4	0.2

Table 4.6: Estimation of σ(% difference) using M-1 and M-2 for a T-2 network in a spatio-temporal setting (lifting coefficients produced using spatio-temporal network with time depth M=3), SNR=2. Each entry in the table shows an average of 100 repetitions.

median operators. Recent literature in [42] demonstrated a robust estimator based on the MAD operator. In this section we will look at several estimators for σ using the variogram estimator proposed in [42]. For a given series $\{f_t\}$, the robust estimator of the variogram proposed in [42] is given by,

$$2\hat{\gamma}(h) = \text{MAD}(\{f_{t+h} - f_t\})^2, \tag{4.17}$$

where $2\hat{\gamma}(h)$ is the estimator of the variogram $2\gamma(h)$ and h is the lag.
Since we sometimes deal with networks that evolve over time, we can estimate the node dependent variograms in similar fashion. For a node k we denote the variogram as $2\gamma_k(h)$ and its estimator $2\hat{\gamma}_k(h)$ is given by,

$$2\hat{\gamma}_k(h) = \text{MAD}(\{f_{(t+h),k} - f_{t,k}\})^2, \tag{4.18}$$

where $f_{t,k}$ is the observation at node k at time t and h is the lag. The choices for h are (lag) 1 and (lag) 2. When we have the situation of dealing with just network data (no time series available), lag 1 means first order neighbour and lag 2 means the second order neighbours (these are the neighbours of the first order neighbours).
Since we are proposing several methods, we distinguish the different variogram

estimators by defining the following naming scheme,

- $2\hat{\gamma}^{\text{method}}(h)$ is the estimator of the variogram $2\gamma(h)$ for a given "method",
- $2\hat{\gamma}_k^{\text{method}}(h)$ is the node dependent estimator of the node dependent variogram $2\gamma_k(h)$ for a given "method".

We distinguish the different estimators for σ using similar naming scheme, i.e.

- $\hat{\sigma}^{\text{method}}$ is the estimator of σ for a given "method",
- $\hat{\sigma}_k^{\text{method}}$ is the node dependent estimator of the node dependent σ_k for a given "method".

Once we have the estimators for the variograms at lag 1 and lag 2, we find an estimator for σ (similar to the one in [42]) as,

$$\hat{\sigma}^{\text{method}} = \sqrt{\frac{\hat{\gamma}^{\text{method}}(1) + \hat{\gamma}^{\text{method}}(2)}{2}}. \tag{4.19}$$

The equivalent node dependent estimator for σ_k is given by,

$$\hat{\sigma}_k^{\text{method}} = \sqrt{\frac{\hat{\gamma}_k^{\text{method}}(1) + \hat{\gamma}_k^{\text{method}}(2)}{2}}. \tag{4.20}$$

Since we deal with constant σ model, we take the average of the node dependent estimators σ_k^{method} to find an estimator for σ, i.e.

$$\hat{\sigma}^{\text{method}} = \frac{1}{n}\sum_{k=1}^{n} \hat{\sigma}_k^{\text{method}} \tag{4.21}$$

In this section we will be looking at several variogram techniques based on the robust MAD operator both in time and wavelet domains.
Now we define the sets J_k, J_k^s and J_k^{s*} which we need later. Let J_k be the set that contains the indices of the neighbours of node k. Each neighbour of node k, $j \in J_k$, will have another set of neighbours whose indices are denoted by the set

$J_{k,j}$. This set will contain the index k since the node k will be a neighbour for a node $j \in J_k$ and may contain some elements of J_k if the node k and the node $j \in J_k$ have common neighbours, i.e, some of the first order neighbours may be mistakenly thought of second order neighbours. Let J_k^s be the set that stores node indices of the second order neighbours of node k,

$$J_k^s = \left\{ \bigcup_{j \in J_k} J_{k,j} \right\} \cap \{k \cup J_k\}' \tag{4.22}$$

where the set operation $\{.\}'$ denotes the complement of that set.

Let J_k^{s*} be the set that stores the indices of the closest second order neighbours to node k. The closest second order neightbours are the neighbours of the first order neighbours which are close to the node k itself. Clearly,

$$\begin{aligned} J_k^{s*} &\subseteq J_k^s, \\ |J_k^{s*}| &\leq |J_k|. \end{aligned} \tag{4.23}$$

4.3.1 Estimation in time domain: Method-1 (TM1)

We treat each node independently and monitor the time series for each node k. Let $\mathbf{f}_k = (f_{t,k})_{t=1}^T$ be the vector that stores the time series for a node k. The estimator for the node dependent variogram $\hat{\gamma}_k^{\text{tm1}}$ is found directly from (4.18). We then find the node dependent estimator $\hat{\sigma}_k^{\text{tm1}}$ from (4.20). Finally we get the estimator $\hat{\sigma}^{\text{tm1}}$ from (4.21).

Table 4.7 shows the estimated σ for some test functions with large available time observations (T=100). We can see from the table that this method estimates σ well.

The above method works well provided there are enough observations (time series $T > 50$) are available per node. However we can still have a reliable estimate for σ provided at least three observations are available per node. We define the

Signals	n=50		n = 150		n = 500	
	True σ	$\hat{\sigma}^{\text{tm1}}$	True σ	$\hat{\sigma}^{\text{tm1}}$	True σ	$\hat{\sigma}^{\text{tm1}}$
g^1	2.08	1.3	2.02	-0.4	2.06	0.0
g^2	0.23	-0.2	0.20	0.0	0.22	0.6
g^3	0.71	1.3	0.58	-0.1	0.62	0.5
g^4	0.27	3.0	0.25	-0.7	0.28	0.9

Table 4.7: Estimation of σ(% difference) using variogram in time domain: TM1, T=100, SNR=2

estimator of the variogram as,

$$2\hat{\gamma}^{\text{tm1}^*}(h) = \text{MAD}(\{f_{(t+h,k)-f_{t,k}}\}_{\forall k \in \mathcal{N}_1})^2, \tag{4.24}$$

where $\mathcal{N}_1 = \{1, \ldots, n\}$. We then find the estimator $\hat{\sigma}^{\text{tm1}^*}$ from (4.19).
Table 4.8 shows the results of estimating σ with only three time samples per node. Both ways of estimating σ give good results. Node level estimation may become preferable if σ is node dependent. Method TM1 works well for both

Signals	n = 50		n = 150		n = 500	
	True σ	$\hat{\sigma}^{\text{tm1}^*}$	True σ	$\hat{\sigma}^{\text{tm1}^*}$	True σ	$\hat{\sigma}^{\text{tm1}^*}$
g^1	2.08	3.4	2.02	-1.4	2.06	0.0
g^2	0.23	-2.1	0.20	-0.6	0.22	0.8
g^3	0.71	1.7	0.58	-1.0	0.62	0.1
g^4	0.27	-0.1	0.25	-0.4	0.28	1.1

Table 4.8: Estimation of σ(% difference) using variogram in time domain: TM1, T=3, SNR=2

spatially smooth functions (no discontinuity) and functions with spatial discontinuity, since this method takes the differences across time observations and no spatial differencing is considered. If the time series is also discontinuous (or has lack of correlation) then this method will fail.

4.3.2 Estimation in Time domain: Method-2 (TM2)

The following methods (TM2, TM3, TM4 and TM5) for variogram technique in time domain present a way to estimate σ with just one time snapshot of network data (i.e. T=1). This will be useful in situations where there are no time series data available or the number of available time series is not enough to perform calculations in the last subsection.

Lag 1 is assumed to contain the immediate neighbours. Therefore, the variograms at lag 1 and lag 2 are given by,

$$\begin{aligned} 2\hat{\gamma}_k^{\mathrm{tm2}}(1) &= \left(\mathrm{MAD}\{(f_j)_{j\in J_k} - f_k\}\right)^2, \\ 2\hat{\gamma}_k^{\mathrm{tm2}}(2) &= \left(\mathrm{MAD}\{(f_j)_{j\in J_k^s} - f_k\}\right)^2, \end{aligned} \tag{4.25}$$

where f_k is the spatial observation at node k, J_k is the set that contains the indices of neighbours of node k and J_k^s is the set that contains the indices of the second order neighbours (defined in (4.22)).

We find the estimator $\hat{\sigma}_k^{\mathrm{tm2}}$ from (4.20). We then find the estimator $\hat{\sigma}^{\mathrm{tm2}}$ from (4.21). This method works well provided there are sufficient number of neighbours for each node k, since it involves applying MAD for the differenced data $(f_j)_{j\in J_k} - f_k$ at each node k. Initial results implied that there might be bias involved in the estimation of the variogram using this method. We propose the following bias correction in a similar way that was presented in [24, 41],

$$\begin{aligned} 2\hat{\gamma}_k^{\mathrm{tm2}}(1) &= (\mathrm{MAD}\{(f_j)_{j\in J_k} - f_k\})^2/B^*, \\ 2\hat{\gamma}_k^{\mathrm{tm2}}(2) &= (\mathrm{MAD}\{(f_j)_{j\in J_k^s} - f_k\})^2/B^*. \end{aligned} \tag{4.26}$$

We find the bias correction B^* that minimises the error function given in (4.43). Section 4.4 shows how to estimate B. Table 4.9 shows simulation results of estimating σ with B^* estimated for $n = 500$. Table 4.9 shows the results of the method TM2 on a spatial network (no temporal data) and table 4.10 shows the results of

the method TM2 on a spatio-temporal network (with time depth $M = 3$). The results show that this method fails to give a good estimate for σ. The reason for failure is that T-1 network will not have enough neighbours to perform MAD operation and T-2 network may have enough neighbours but since the neighbours are far apart from the node k, the correlation is less. Therefore when we take differencing of the neighbours data, it results in large error and thus results in over estimation for σ. Therefore we recommend considering closest neighbours which we will consider in our next methods.

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{tm2}}$		True σ	$\hat{\sigma}^{\text{tm2}}$		True σ	$\hat{\sigma}^{\text{tm2}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	33.4	38.7	2.02	12.6	38.8	2.06	12.5	25.6
g^2	0.23	70.4	82.9	0.20	28.4	65.1	0.22	17.2	35.9
g^5	0.61	72.8	88.4	0.53	25.8	60.8	0.58	7.9	25.4
g^6	0.22	56.3	56.0	0.19	19.0	42.4	0.20	5.5	17.7
g^7	0.22	31.4	49.8	0.20	7.2	33.1	0.21	3.2	12.1
g^8	0.70	21.9	34.1	0.56	7.7	31.6	0.59	1.8	8.3

Table 4.9: Estimation of σ(% difference) using variogram in time domain (estimation on spatial network)): TM2, $B^* = 0.48$ for T-2 network and $B^* = 0.33$ for T-1 network, SNR=2

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{tm2}}$		True σ	$\hat{\sigma}^{\text{tm2}}$		True σ	$\hat{\sigma}^{\text{tm2}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	29.3	32.3	2.02	13.8	29.6	2.06	11.8	21.5
g^2	0.23	46.1	97.0	0.20	18.5	53.4	0.22	11.9	29.2
g^5	0.61	46.3	92.5	0.53	19.5	52.6	0.58	7.7	20.7
g^6	0.22	34.9	61.0	0.19	13.6	36.2	0.20	6.9	14.3
g^7	0.22	22.2	48.8	0.20	8.5	27.3	0.21	5.5	10.0
g^8	0.70	11.7	29.7	0.56	7.5	25.0	0.59	4.6	6.7

Table 4.10: Estimation of σ(% difference) using variogram in time domain (estimation on spatio-temporal network): TM2, $B^* = 0.48$ for T-2 network and $B^* = 0.33$ for T-1 network, SNR=2

4.3.3 Estimation in Time domain: Method-3 (TM3)

The estimation of variogram relies on differencing the data to the lagged versions of data. In the spatial data setting, the lagged version of data means the neighbouring data. We know that a smooth spatial function at a point can be approximated by taking the average of its neighbours' values. Therefore we define the estimators for the variograms at lag 1 and lag 2 as,

$$\begin{aligned} 2\hat{\gamma}^{\text{tm3}}(1) &= \text{MAD}(\{f_k - \frac{1}{|J_k|}\sum_{j\in J_k} f_j\}_{\forall k\in\mathcal{N}_1})^2/B^*, \\ 2\hat{\gamma}^{\text{tm3}}(2) &= \text{MAD}(\{f_k - \frac{1}{|J_k^{s*}|}\sum_{j\in J_k^{s*}} f_j\}_{\forall k\in\mathcal{N}_1})^2/B^*, \end{aligned} \tag{4.27}$$

where J_k, J_k^{s*} as defined earlier, $\mathcal{N}_1 = \{1, \ldots, n\}$ and B^* is the bias correcting constant for this method. Since initial experiments did not yield a good estimator, we have used B^* to correct the bias (see section 4.4 for more detail on how to find this constant). We then find the estimator $\hat{\sigma}^{\text{tm3}}$ from (4.19).

Table 4.11 shows the results for estimating σ based on the spatial network and table 4.12 shows the results of estimation on a spatio-temporal network. The results improve for the estimation on T-1 and T-2 networks with increasing number of nodes. It results in bad estimation with few number of nodes, because of the lack of similarity between points and neighbours. The bias correction, B^*, required for this method is different for T-1 and T-2 networks. We have estimated $B^* = 0.53$ based on a T-2 network with 500 nodes (repeated 100 times) and $B^* = 0.78$ estimated for a T-1 network with 500 nodes. Because of the lack of similarity in the function in the far away nodes, considering only closest first order neighbours as well as closest second order neighbours may improve the results in T-2 network case and this could be a future task for this method.

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{tm3}}$		True σ	$\hat{\sigma}^{\text{tm3}}$		True σ	$\hat{\sigma}^{\text{tm3}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	16.7	34.8	2.02	9.8	26.7	2.06	4.1	13.2
g^2	0.23	28.3	123.9	0.20	9.2	40.1	0.22	5.4	15.6
g^5	0.61	27.7	88.9	0.53	8.0	29.7	0.58	1.0	5.5
g^6	0.22	19.0	67.6	0.19	4.9	21.2	0.20	0.8	3.2
g^7	0.22	14.9	44.6	0.20	2.65	9.3	0.21	1.7	2.7
g^8	0.70	3.9	21.1	0.56	1.4	7.1	0.59	-0.4	0.6

Table 4.11: Estimation of σ(% difference) using variogram in time domain (estimation on spatial network): TM3, $B^* = 0.53$ for T-2 network and $B^* = 0.78$ for T-1 network, SNR=2

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{tm3}}$		True σ	$\hat{\sigma}^{\text{tm3}}$		True σ	$\hat{\sigma}^{\text{tm3}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	-0.2	19.2	2.02	-3.4	12.5	2.06	-5.8	7.6
g^2	0.23	-2.3	43.2	0.20	-4.6	11.2	0.22	-5.8	7.2
g^5	0.61	-3.2	41.8	0.53	-5.2	6.2	0.58	-8.8	0.1
g^6	0.22	-3.1	16.6	0.19	-5.8	4.5	0.20	-8.2	-0.3
g^7	0.22	-9.0	8.2	0.20	-6.5	1.2	0.21	-6.9	0.4
g^8	0.70	-8.3	5.4	0.56	-8.8	1.5	0.59	-8.1	-1.6

Table 4.12: Estimation of σ(% difference) using variogram in time domain (estimation on spatio-temporal network): TM3, $B^* = 0.53$ for T-2 network and $B^* = 0.78$ for T-1 network, SNR=2

4.3.4 Estimation in time domain: Method-4 (TM4)

Method TM2 only works well if there are enough number of close neighbours for each node. In practical situations, we may consider the method described below. Instead of taking the node dependent estimate of σ_k, we are going to estimate σ by taking the first order difference for all the nodes. Therefore the estimators for the variograms at lag 1 and lag 2 are given as,

$$\begin{aligned} 2\hat{\gamma}^{\text{tm4}}(1) &= \text{MAD}(\{(f_j)_{j\in J_k} - f_k\}_{\forall k\in\mathcal{N}_1})^2/B^*, \\ 2\hat{\gamma}^{\text{tm4}}(2) &= \text{MAD}(\{(f_j)_{j\in J_k^{s*}} - f_k\}_{\forall k\in\mathcal{N}_1})^2/B^*, \end{aligned} \tag{4.28}$$

where B^* is the bias correcting constant however for this method we estimate $B^* = 1$. We then find the estimator $\hat{\sigma}^{\text{tm4}}$ from (4.19).

Table 4.13 shows results based on spatial network using the variogram method TM4 discussed above. These are better than those in table 4.11. The estimation results improve with the increasing number of nodes. Table 4.14 shows the results of estimation on a spatio-temporal network. These are better than those in table 4.12 for large n. The bias correction for this method is approximately 1.

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{tm4}}$		True σ	$\hat{\sigma}^{\text{tm4}}$		True σ	$\hat{\sigma}^{\text{tm4}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	15.6	30.5	2.02	8.9	18.3	2.06	3.9	10.6
g^2	0.23	37.3	121.5	0.20	13.7	44.7	0.22	6.3	16.5
g^5	0.61	38.4	91.3	0.53	13.8	46.7	0.58	2.9	13.7
g^6	0.22	25.4	68.6	0.19	9.6	30.6	0.20	2.2	9.5
g^7	0.22	20.5	53.9	0.20	4.9	21.4	0.21	2.2	7.4
g^8	0.70	7.1	31.4	0.56	4.1	18.3	0.59	0.8	4.7

Table 4.13: Estimation of σ(% difference) using variogram in time domain (estimation on spatial network): TM4, $B^* = 1$, SNR=2

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{tm4}}$		True σ	$\hat{\sigma}^{\text{tm4}}$		True σ	$\hat{\sigma}^{\text{tm4}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	10.3	17.5	2.02	4.2	10.6	2.06	1.4	7.1
g^2	0.23	13.0	51.0	0.20	4.4	16.8	0.22	2.5	8.2
g^5	0.61	13.2	53.5	0.53	5.2	21.5	0.58	0.3	5.7
g^6	0.22	11.3	29.8	0.19	2.3	11.5	0.20	-0.5	3.4
g^7	0.22	4.8	22.3	0.20	2.2	8.7	0.21	1.7	3.6
g^8	0.70	0.6	12.6	0.56	0.8	8.5	0.59	-0.4	1.3

Table 4.14: Estimation of σ(% difference) using variogram in time domain (estimation on spatio-temporal network): TM4, $B^* = 1$, SNR=2

4.3.5 Estimation in time domain: Method-5 (TM5)

We know that the lag 1 difference can also be defined as the difference between the observation at node k and the observation at the closest neighbour (based on the Euclidean distance) whose node index is $j_c \in J_k$. We can also define the lag 2 difference as the difference between the observation at node k and the closest second order neighbour whose node index is $j_c^s \in J_k^s$.

We define the estimators for the variograms at lag 1 and lag 2 as,

$$\begin{aligned} 2\hat{\gamma}^{\text{tm5}}(1) &= \text{MAD}(\{f_k - f_{j_c}\}_{\forall k \in \mathcal{N}1})^2/B^*, \\ 2\hat{\gamma}^{\text{tm5}}(2) &= \text{MAD}(\{f_k - f_{j_c^s}\}_{\forall k \in \mathcal{N}1})^2/B^*, \end{aligned} \tag{4.29}$$

where all terms are as defined earlier. We then find the estimator $\hat{\sigma}^{\text{tm5}}$ from (4.19). Note that, in spatio-temporal network based estimation, the closest first order neighbour and closest second order neighbour will be copies of the node itself in the past and future unless the temporal scale factor is greater than 1. In our simulations we use the temporal scale factor of 1.

Table 4.15 shows the results of estimating σ using TM5 on a spatial network. Estimation results improve with increasing number of nodes. Table 4.16 shows the result of estimation on a spatio-temporal network. Estimation on a spatio-temporal network using TM5 shows extremely good results even with few number of nodes.

4.3.6 Estimation in wavelet domain: Method-1 (WM1)

The approach in this section is similar to the time domain one presented in section 4.3.1. We assume that we have sufficient time observations available. We perform the LOCAAT transform independently over the available time observations and and arrange the LOCAAT coefficients as a series for each node. Let $\mathbf{d}_k = (d_{k,t})_{t=1}^T$ be the vector that stores the LOCAAT coefficients for a node k.

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{tm5}}$		True σ	$\hat{\sigma}^{\text{tm5}}$		True σ	$\hat{\sigma}^{\text{tm5}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	16.9	24.6	2.02	10.3	11.5	2.06	3.0	3.9
g^2	0.23	22.0	89.2	0.20	9.5	29.4	0.22	5.2	11.3
g^5	0.61	26.9	71.4	0.53	10.3	33.0	0.58	2.5	8.1
g^6	0.22	13.7	49.6	0.19	7.5	23.0	0.20	1.5	6.5
g^7	0.22	11.2	30.9	0.20	4.0	13.4	0.21	2.4	5.0
g^8	0.70	3.4	19.3	0.56	2.1	10.0	0.59	-0.3	2.0

Table 4.15: Estimation of σ(% difference) using variogram in time domain (estimation on spatial network): TM5, $B^* = 1$, SNR=2

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{tm5}}$		True σ	$\hat{\sigma}^{\text{tm5}}$		True σ	$\hat{\sigma}^{\text{tm5}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	3.4	2.7	2.02	-0.7	0.3	2.06	-0.3	-0.4
g^2	0.23	-0.7	2.0	0.20	1.1	1.3	0.22	-0.4	-0.1
g^5	0.61	-0.6	8.3	0.53	1.8	2.7	0.58	-0.9	-0.8
g^6	0.22	3.5	6.6	0.19	-1.0	0.7	0.20	-1.5	-1.2
g^7	0.22	-2.4	2.1	0.20	-0.4	0.0	0.21	0.1	0.1
g^8	0.70	-3.0	-0.9	0.56	-0.36	0.5	0.59	-2.1	-1.6

Table 4.16: Estimation of σ(% difference) using variogram in time domain (estimation on spatio-temporal network): TM5, $B^* = 1$, SNR=2

We estimate the variogram at lag h for a node k as,

$$2\hat{\gamma}_k^{\text{wm1}}(h) = (\text{MAD}\{d_{(t+h),k} - d_{t,k}\})^2. \tag{4.30}$$

We then find the node dependent estimator $\hat{\sigma}_k^{\text{wm1}}$ using (4.20) and then by using (4.21) we find the estimator $\hat{\sigma}^{\text{wm1}}$. Table 4.17 shows the results of estimating σ with varying number of nodes.

The above method works well provided there are enough observations are available per node however, we can still find a reliable estimator for the variogram provided at least there are three observations are available per node (similar to

Signals	n = 50		n = 150		n = 500	
	True σ	$\hat{\sigma}^{\text{wm1}}$	True σ	$\hat{\sigma}^{\text{wm1}}$	True σ	$\hat{\sigma}^{\text{wm1}}$
g^1	2.08	1.9	2.02	0.1	2.06	0.1
g^2	0.23	2.7	0.20	0.4	0.22	0.8
g^3	0.71	0.3	0.58	-0.6	0.62	0.9
g^4	0.27	0.6	0.25	0.4	0.28	0.6

Table 4.17: Estimation of σ(% difference) using variogram in wavelet domain:WM1, T=100, SNR=2.

TM1). We now define the following estimator for the variogram at lag h,

$$2\hat{\gamma}^{\text{wm1}^*}(h) = (\text{MAD}\{(d_{(t+h),k} - d_{t,k})_{\forall k \in \mathcal{N}_1}\})^2. \tag{4.31}$$

The estimator $\hat{\sigma}^{\text{wm1}^*}$ is derived by using (4.19). Table 4.18 shows the results using this method. We find both methods estimate σ extremely well and do not require bias correction.

Signals	n = 50		n = 150		n = 500	
	True σ	$\hat{\sigma}^{\text{wm1}^*}$	True σ	$\hat{\sigma}^{\text{wm1}^*}$	True σ	$\hat{\sigma}^{\text{wm1}^*}$
g^1	2.08	5.0	2.02	0.6	2.06	-0.3
g^2	0.23	2.2	0.20	0.2	0.22	-0.2
g^3	0.71	-3.8	0.58	-1.6	0.62	0.5
g^4	0.27	-6.0	0.25	-0.4	0.28	-0.7

Table 4.18: σ Estimation using variogram in wavelet domain: WM1, T=3, SNR=2

4.3.7 Estimation in wavelet domain: Method-2 (WM2)

The following methods (WM2, WM3 and WM4) will attempt to estimate σ when time series is not available. We take the node dependent fine scale coefficients $\mathbf{d}_k^{\text{fine}}$ (given in (4.12)) to define the estimator for the variogram at lag 1 as,

$$2\hat{\gamma}_k^{\text{wm2}}(1) = \left(\text{MAD}\{\mathbf{d}_k^{\text{fine}}\}\right)^2 / B^*. \tag{4.32}$$

The following discussion defines a new set of fine scale coefficients, $\mathbf{d}_k^{\text{fine}_2}$ based on the second order neighbours which is used in the estimator for the variogram at lag 2.

Let $\mathcal{B}_k \subseteq J_k^s$ be the set that stores the indices of the detail coefficients of the second order neighbours of node k in the order of they were calculated and $L_k^{'}$ be its cardinality.

$$\mathcal{B}_k = J_k^s \cap \mathcal{D}_1 = \{i_1^k, \ldots, i_{L_k^{'}}^k\} \tag{4.33}$$

Now we define another set $\mathcal{B}_k^{\text{fine}} = \{i_1^k, \ldots, i^k_{\lfloor \frac{L_k^{'}}{2} \rfloor}\}$ that stores the fine scale coefficients of second order neighbours. We define the fine scale coefficients of the second order neighbours as follows,

$$\mathbf{d}_k^{\text{fine}_2} = \{d_i^* : i \in \mathcal{B}_k^{\text{fine}}\}. \tag{4.34}$$

Now the estimator for the variogram at lag 2 is defined as,

$$2\hat{\gamma}_k^{\text{wm2}}(2) = \left(\text{MAD}\{\mathbf{d}_k^{\text{fine}_2}\}\right)^2 / B^*. \tag{4.35}$$

The node dependent estimator $\hat{\sigma}^{\text{wm2}}$ is obtained by using (4.20) and (4.21).

This method only works with type-2 networks since there are not enough neighbours for a MAD operation at node level for type-1 network. Table 4.19 shows the estimation results using WM2 on spatial LOCAAT coefficients. Table 4.20 shows the results of estimating σ with the spatio-temporal LOCAAT coefficients and we can clearly see that this method works better for the T-2 network case.

4.3.8 Variogram method in wavelet domain: Method 3 (WM3)

We have seen that node level approach is not usually very successful as there are not usually enough neighbours and this results in poor estimation. Hence we

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{wm2}}$		True σ	$\hat{\sigma}^{\text{wm2}}$		True σ	$\hat{\sigma}^{\text{wm2}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	-32.4	36.0	2.02	19.1	19.2	2.06	-19.8	14.4
g^2	0.23	-12.9	46.2	0.20	8.5	24.3	0.22	-28.1	14.1
g^3	0.71	-25.0	20.9	0.58	24.3	11.4	0.62	-28.8	6.0
g^4	0.27	-20.5	45.6	0.25	3.8	27.8	0.28	-24.5	13.8

Table 4.19: Estimation of σ(% difference) using variogram in wavelet domain (spatial LOCAAT coefficients): WM2, $B^* = 0.44$, SNR=2

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{wm2}}$		True σ	$\hat{\sigma}^{\text{wm2}}$		True σ	$\hat{\sigma}^{\text{wm2}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	-17.5	14.7	2.02	-21.8	7.8	2.06	-24.1	6.3
g^2	0.23	-4.9	20.1	0.20	-21.2	8.6	0.22	-24.5	6.1
g^3	0.71	-14.7	9.0	0.58	-24.9	4.3	0.62	-26.5	3.2
g^4	0.27	-14.6	13.0	0.25	-19.7	8.6	0.28	-23.1	6.1

Table 4.20: Estimation of σ(% difference) using variogram in wavelet domain (spatio-temporal LOCAAT coefficients): WM2, $B^* = 0.44$, SNR=2

adopt the following approach in wavelet domain. We define the set of fine scale coefficients resulting from LOCAAT transform as,

$$\mathcal{D}_1^{fine} = \{i_r\}_{r=\lceil 0.5(n-1)\rceil}^{n} \qquad i_r \in \mathcal{D}_1 \tag{4.36}$$

hence,

$$|\mathcal{D}_1^{fine}| \leq 0.5|\mathcal{D}_1| \tag{4.37}$$

For each node k we define a new index set, $\mathcal{A}_k^*$, that stores the selected indices from $\{J_k \cup k\}$ that match the fine scale coefficients. We make sure we do not get the coarse scale coefficients into account by the following operation.

$$\mathcal{A}_k^* = \{J_k \cup k\} \cap \mathcal{D}_1^{\text{fine}} = \{i_1, \ldots, i_{L_k^{\mathcal{A}_k^*}}\}, \tag{4.38}$$

where $L_k^{\mathcal{A}_k^*}$ is the cardinality of the set $\mathcal{A}_k^*$. We define another new index set, $\mathcal{B}_k^*$, that stores only the selected indices from $\{J_k^{s^*} \cup k\}$ that are fine scale coefficients for the second order neighbours.

$$\mathcal{B}_k^* = \{J_k^{s^*} \cup k\} \cap \mathcal{D}_1^{\text{fine}} = \{i_1, \ldots, i_{L_k^{\mathcal{B}_k^*}}\}. \tag{4.39}$$

The estimators for the variograms at lag 1 and lag 2 for this method is defined as,

$$\begin{aligned} 2\hat{\gamma}^{\text{wm3}}(1) &= \text{MAD}\{\{\{d_j\}_{j\in\mathcal{A}_k^*}\}_{\forall k\in\mathcal{N}_1}\}^2/B^*, \\ 2\hat{\gamma}^{\text{wm3}}(2) &= \text{MAD}\{\{\{d_j\}_{j\in\mathcal{B}_k^*}\}_{\forall k\in\mathcal{N}_1}\}^2/B^*, \end{aligned} \tag{4.40}$$

where all the terms are as defined earlier and B^* is the bias correction. Having defined the estimators for the variogram, the estimator $\hat{\sigma}^{\text{wm3}}$ is obtained by using (4.19).

Table 4.21 and table 4.22 shows the results from this method. It is clear from these tables that the method gives better estimates compared to some of the methods described above.

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{wm3}}$		True σ	$\hat{\sigma}^{\text{wm3}}$		True σ	$\hat{\sigma}^{\text{wm3}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	4.5	15.9	2.02	9.4	11.5	2.06	2.6	6.9
g^2	0.23	5.5	38.6	0.20	1.1	11.8	0.22	3.7	7.3
g^3	0.71	3.2	13.4	0.58	1.5	2.7	0.62	0.7	1.0
g^4	0.27	-2.4	15.2	0.25	8.6	17.6	0.28	0.5	5.0

Table 4.21: Estimation of σ(% difference) using variogram in wavelet domain (spatial LOCAAT coefficients): WM3, $B^* = 0.51$, SNR=2

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{wm3}}$		True σ	$\hat{\sigma}^{\text{wm3}}$		True σ	$\hat{\sigma}^{\text{wm3}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	6.7	6.1	2.02	3.4	0.8	2.06	0.9	-0.5
g^2	0.23	7.2	8.8	0.20	5.0	-0.5	0.22	0.9	0.8
g^3	0.71	6.5	-0.7	0.58	-1.7	-2.9	0.62	0.2	-0.2
g^4	0.27	3.2	1.4	0.25	2.0	-0.6	0.28	0.7	-0.8

Table 4.22: Estimation of σ(% difference) using variogram in wavelet domain (spatio-temporal LOCAAT coefficients): WM3, $B^* = 0.51$, SNR=2

4.3.9 Variogram method in wavelet domain: Method 4 (WM4)

In this section we find the following is a robust estimator. We find only one first order fine scale coefficient. The estimators for the variograms at lag 1 and lag 2 are given as,

$$\begin{aligned} 2\hat{\gamma}^{\text{wm4}}(1) &= \text{MAD}\{(d_i^*)_{i\in\mathcal{A}^*}\}^2/B^*, \\ 2\hat{\gamma}^{\text{wm4}}(2) &= \text{MAD}\{(d_i^*)_{i\in\mathcal{B}^*}\}^2/B^*, \end{aligned} \tag{4.41}$$

where the sets $\mathcal{A}^*$ and $\mathcal{B}^*$ are defined as,

$$\begin{aligned} \mathcal{A}^* &= \{i_1^k \in \mathcal{A}_k^*\}, \qquad k = 1, \ldots, n, \\ \mathcal{B}^* &= \{i_1^k \in \mathcal{B}_k^*\}, \qquad k = 1, \ldots, n. \end{aligned} \tag{4.42}$$

Having found the estimators for the variogram, we find the estimator $\hat{\sigma}^{\text{wm4}}$ from (4.19). Table 4.23 and table 4.24 shows the results from this method. It is clear from these tables that this method gives a much improved estimator compared to other methods described above.

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{wm4}}$		True σ	$\hat{\sigma}^{\text{wm4}}$		True σ	$\hat{\sigma}^{\text{wm4}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	7.9	0.9	2.02	8.8	8.9	2.06	0.0	0.4
g^2	0.23	5.1	19.3	0.20	1.0	2.3	0.22	2.2	4.9
g^3	0.71	6.7	-4.5	0.58	0.1	0.0	0.62	0.1	-3.1
g^4	0.27	-3.4	0.1	0.25	7.5	3.7	0.28	-0.1	0.6

Table 4.23: Estimation of σ(% difference) using variogram in wavelet domain (spatial LOCAAT coefficients): WM4, $B^* = 0.52$, SNR=2

Signals	n = 50			n = 150			n = 500		
	True σ	$\hat{\sigma}^{\text{wm4}}$		True σ	$\hat{\sigma}^{\text{wm4}}$		True σ	$\hat{\sigma}^{\text{wm4}}$	
		T-1	T-2		T-1	T-2		T-1	T-2
g^1	2.08	8.9	1.1	2.02	2.8	0.0	2.06	-0.4	-2.2
g^2	0.23	1.0	-2.9	0.20	1.7	-5.9	0.22	-0.4	-1.8
g^3	0.71	6.5	-4.5	0.58	-2.2	-6.0	0.62	-0.7	-1.7
g^4	0.27	1.9	-0.8	0.25	1.1	-5.3	0.28	-1.6	-1.5

Table 4.24: Estimation of σ(% difference) using variogram in wavelet domain (spatio-temporal LOCAAT coefficients): WM4, $B^* = 0.52$, SNR=2

4.4 Estimation of B

The robust estimation of the variogram found in [24, 41] proposed some bias correction. We explain more about the bias correction here.

For a given fixed network, we generate artificial noise of known variance level σ. For time domain methods we use this noise as the observations and for the wavelet domain estimation, we perform the LOCAAT transform on the noise. We define the mean squared error function for estimating σ as,

$$e(B) = \frac{1}{N}\sum_{i=1}^{N}(\hat{\sigma}_i(B) - \sigma_i)^2, \tag{4.43}$$

where σ_i is known noise standard deviation, $\hat{\sigma}_i(B)$ is the estimator using the proposed method for a given B and N is the number of independent experiments.

We repeat the above calculation until we find $B = B^*$ that minimises the error function above.

$$B^* = \arg \min_{0<B\leq 1} e(B). \tag{4.44}$$

Figure 4.1 shows MSE vs B for various methods in time domain. Figure 4.2 shows MSE vs B for various methods in wavelet domain.

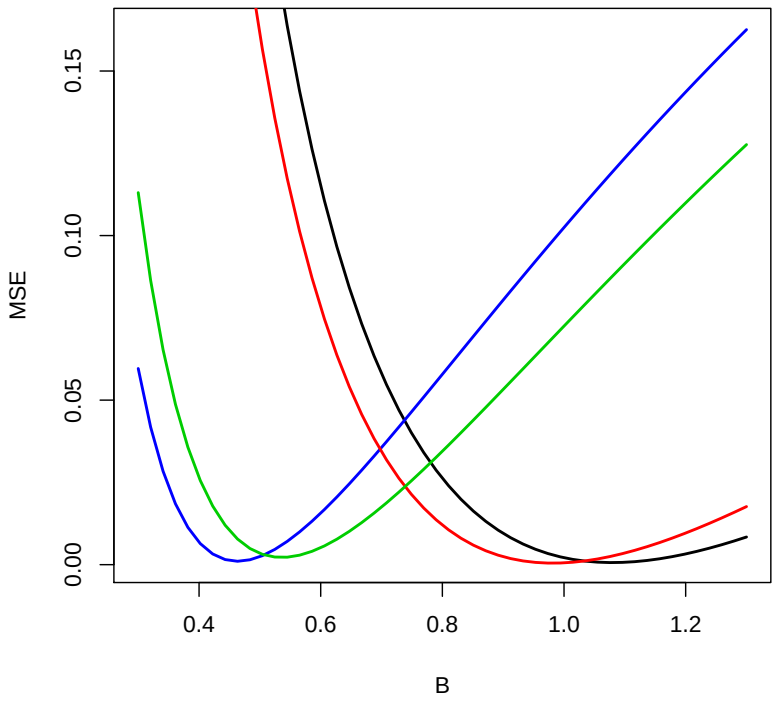


Figure 4.1: MSE error function for σ estimation against the bias correction B using time domain variogram methods. Blue line for TM2, green for TM3, red solid line for TM4 and black for TM5, $n = 500$, each point is an average over 100 independent experiments.

Table 4.25 shows B^* for time domain variogram methods and table 4.26 shows the values for B^* for wavelet domain variogram methods. Each value in the tables are averages over 100 independent experiments. Only for methods TM2 and TM3, the B^* vary with the type of network used.

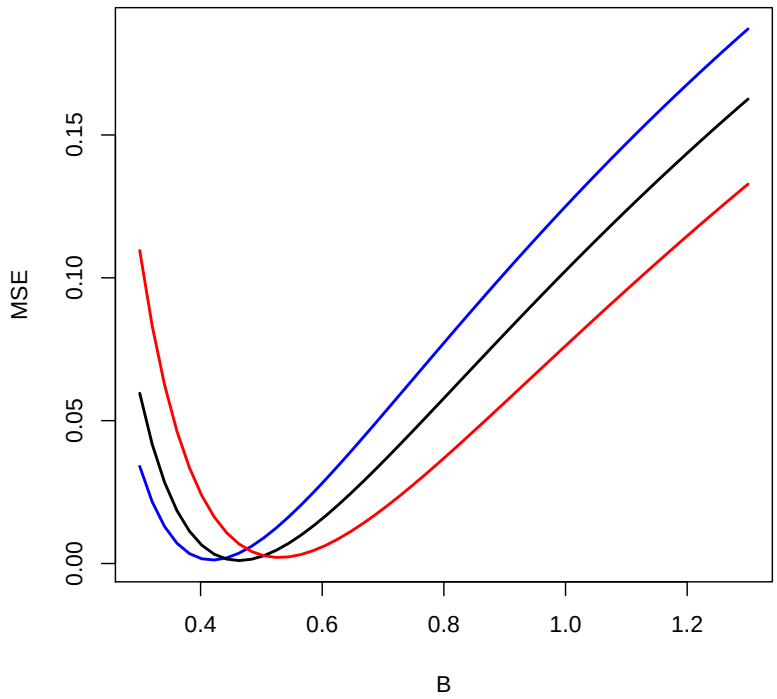


Figure 4.2: MSE error function for σ estimation against the bias correction B using wavelet domain variogram methods. Blue line for WM2, red solid line for WM3 and black for WM4, $n = 500$, each point is an average over 100 independent experiments.

Methods	B^*
TM2	0.48
TM3	0.53
TM4	1.0
TM5	1.0

Table 4.25: Estimated B^* for time domain variogram methods.

Methods	B^*
WM2	0.44
WM3	0.51
WM4	0.52

Table 4.26: Estimated B^* for wavelet domain variogram methods.

4.5 Conclusions

The reliable estimation of the noise variance σ^2 is crucial in wavelet shrinkage problems. In this chapter we have considered estimation of σ based on network data. We have considered some estimation methods based on the MAD technique, and then we moved on to some variogram-like techniques. In the variogram method, we looked at several ways to estimate σ both in time domain and wavelet domain. We initially considered the classical MAD approach and found that it overestimates when the number of nodes are few, but it reliably estimates with a large number of nodes. We adapted this classical MAD method to local MAD estimation. We proposed two methods under this local MAD technique, M-1 and M-2. We found M-2 works well for both T-1(MST type) and T-2(circular disc approach) network cases. M-1 fails for T-1 networks since there are typically not enough neighbours to perform MAD operation.

We then proposed an estimator based on the variogram method proposed in [42]. We called this method TM1 and it generally gives reliable estimation of σ. For this method to work, we need at least three time observations per node. We proposed another method, TM2, in order to estimate σ based on spatial observations but this method failed to yield a reliable estimation because there are not enough neighbours available for T-1 networks and the neighbours may be far apart for T-2 network (therefore less correlation, more error and results in overestimation). Therefore we proposed the method TM3 which gives better estimation for both T-1 and T-2 network when the number of nodes are large. We proposed another method,TM4, based on the closest second order neighbours. Under this method, T-1 network yields better results with large number of nodes. Since we did not consider the closest first order neighbours, T-2 networks result in poor estimation compared to T-1 networks. Considering closest first order neighbours will be a future task. We proposed our final time domain variogram method, TM5, based on the closest first order neighbour and closest second order neighbours. This

method yields reliable estimation with large numbers of nodes.

Having defined variogram methods in time domain, we then turned to variogram methods in wavelet domain as the wavelet coefficients are good at separating the noise from the signal. We proposed wavelet domain variogram method WM1, similar to TM1, that reliably estimate σ. Again we need at least three time observations per node for this method to work. We proposed another method WM2 that is aimed to estimate σ purely based on the LOCAAT transform of a network at a given time. T-1 networks typically will fail with this method as there are not enough neighbours to perform node level MAD operation based on the node level fine scale coefficients. Estimation on a T-2 network's LOCAAT coefficients using method WM2 improves with increasing number of nodes, but it is not robust enough. We then proposed method WM3 that takes all the node level fine scale coefficients based on immediate neighbours and the second order neighbours to estimate the variogram. From the variogram we find the estimator $\hat{\sigma}$. This method yields reliable estimation for both T-1 and T-2 type networks. We propose our final method WM4 based on finding one finest scale coefficient among immediate neighbours, and finding one finest scale coefficient among the second order neighbours. From these coefficients, the variogram is estimated and thus the estimator $\hat{\sigma}$ is obtained. This method yields a good estimation for σ even with a smaller number of nodes.

Some of the variogram methods presented in this chapter needed some bias correction. We followed a similar bias correction approach to that found in [24, 41]. For a given network, we estimate B^* that minimises the error function in (4.43). We have seen that for methods TM2 and TM3, B^* vary with the type of network used. In the real world problem, we would not know what generates the network or the data. Therefore, once we know the network (this information as a list of edges and distance), our method of estimating B^* ensures that we get the right B^* for a proposed method and thus result in reliable estimate for σ.

We give a quick summary of the variogram methods discussed in this chapter in table 4.27. The variogram method WM4 is recommended when time series data

Methods	Bias correction	Many neighbours	works for small n	Time series
TM1	No	No	Yes	Yes
TM2	Yes	Yes	No	No
TM3	Yes	No	No	No
TM4	No	No	No	No
TM5	No	No	No	No
WM1	No	No	Yes	Yes
WM2	Yes	Yes	No	No
WM3	Yes	No	Yes/No	No
WM4	Yes	No	Yes	No

Table 4.27: Summary of variogram methods (requirement check list).

is not available. When time series is available methods TM1 or WM1 can give a very good estimation for noise variance.

Chapter 5

Thresholding Methods

5.1 Introduction

In the wavelet domain, a signal is typically represented by few coefficients, i.e the energy of the signal is concentrated in few coefficients whereas noise tends to spread across all coefficients. This is also called the sparsity property of the wavelet representation. The sparsity property enables thresholding in wavelet domain to reduce the noise present in the noisy signal. In the last chapter, we saw some promising methods to estimate the noise level σ. In this chapter, we will investigate some suitable thresholding methods for LOCAAT. In most of this chapter we will assume that σ is known. We apply some standard thresholding methods first and then we propose a number of new thresholding strategies for LOCAAT. We evaluate these thresholding methods with the aid of simulation studies.

In the second part of this chapter, we will look at some threshold selection methods that directly minimise the mean squared error. These methods are useful as they circumvent the issue of having to separately estimate the noise variance σ^2. After that, we will look at improving estimation efficiency by smoothing wavelet coefficients. In the final part, we will look at the sparsity of LOCAAT with

two types of networks, type-1 (a network formed using a Minimum Spanning Tree (MST) technique) and type-2 (a node's neighbours are those nodes that lie within the circle with radius r and the centre being the node), which we defined earlier in section 3.2.2.

5.2 Experimental set up and Bias, Variance and MSE calculations

In our simulation study we choose sample sizes of $n = 50, 150, 500$. We test our methods with two test functions with discontinuities. Function g^1 consists of two planes with discontinuity between them and function g^2 is a quadratic surface with a discontinuity (see section 3.2.3 for definition of these functions).
We estimate bias for our estimation procedures by repeating this experiment $N = 100$ times independently. Estimation bias for node k is calculated as,

$$\begin{aligned} \text{Bias}(\hat{g}_k) &= E(\hat{g}_k) - g_k \\ &\approx \frac{1}{N}\sum_{i=1}^{N} \hat{g}_{ik} - g_k \\ &= \bar{g}_k - g_k, \end{aligned} \tag{5.1}$$

where $\hat{g}_{ik}$ is the estimated signal at node k in the i^{th} experiment. The variance of node k is calculated as,

$$\begin{aligned} \text{var}(\hat{g}_k) &= E[(\hat{g}_k - E[\hat{g}_k])^2] \\ &= E[\hat{g}_k^2] - E[\hat{g}_k]^2 \\ &\approx \frac{1}{N}\sum_{i=1}^{N} \hat{g}_{ik}^2 - \bar{g}_k^2. \end{aligned} \tag{5.2}$$

Empirically the MSE is calculated as follows,

$$\mathrm{MSE}(\hat{g}_k) = \frac{1}{N}\sum_{i=1}^{N}(\hat{g}_{ik} - g_k)^2. \tag{5.3}$$

Each point in subsequent bias, variance and MSE plots in the following sections is calculated by taking the average of all nodes i.e,

$$\begin{aligned} \mathrm{Bias}^2(\hat{g}) &= \frac{1}{n}\sum_{k=1}^{n}\mathrm{Bias}^2(\hat{g}_k), \\ \mathrm{var}(\hat{g}) &= \frac{1}{n}\sum_{k=1}^{n}\mathrm{var}(\hat{g}_k), \\ \mathrm{MSE}(\hat{g}) &= \mathrm{Bias}^2(\hat{g}) + \mathrm{var}(\hat{g}). \end{aligned} \tag{5.4}$$

These quantities give an overall evaluation across the network. Some of the results in the following sections are assessed by means of efficiency. Efficiency is defined as follows [28, 30],

$$\mathrm{eff}(\hat{g}) = \log_{10}\left(\frac{\mathrm{var}(g)}{\mathrm{var}(\hat{g}-g)}\right) / \log_{10}\left(\frac{\mathrm{var}(g)}{\mathrm{var}(f-g)}\right), \tag{5.5}$$

where g is the original function, f is the observation and $\hat{g}$ is the estimator of g. A good estimator will have a large efficiency eff > 1. If eff ≤ 1, this means the estimator is not better than the noisy observation itself.

5.3 Hard and soft thresholding

We apply hard and soft shrinkage rules, proposed in [32, 34], to the empirical wavelet coefficients $\mathbf{d}^*$ (see (4.4)). For these methods, a good choice of threshold parameter λ is essential. Work in [34] proposed the universal threshold $\lambda = \sigma\sqrt{2\log n}$. It is well known that the estimates resulting from the universal threshold are biased [48]. Therefore we choose a threshold parameter $\lambda = p\sigma\sqrt{2\log n}$ that is a portion of the universal threshold, where $p \in [0, 1]$

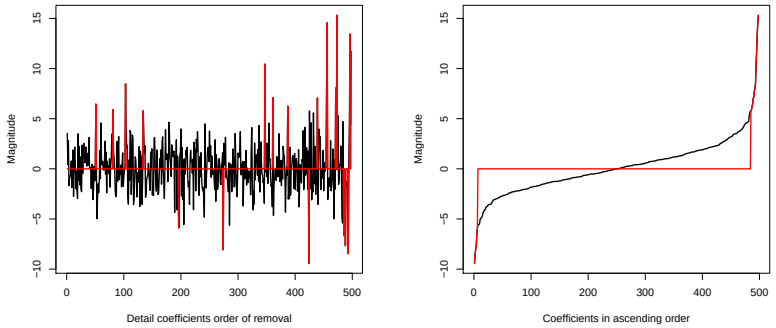
and $p = 1$ yields universal threshold. examples $p < 1$ has been found to work well in practice [48].

We calculate bias, variance and MSE according to (5.4) with p varying from 0% to 100%. Figure 5.6(a) shows the results of hard thresholding for the g^1 function with 500 nodes in T-1 network. It is clear that the universal threshold is not the optimal threshold for hard thresholding of LOCAAT coefficients. For LOCAAT hard-thresholding we have found that good thresholds seems to lie around 80% of the universal threshold. The variance is monotonically decreasing as expected, however the bias seems to be appearing after 60% and the bias starts to dominate after around 80%. Figure 5.6(c) shows the results of soft thresholding for g^1 function with 500 nodes in type-1 network. Although the variance seems to decrease monotonically as p increases, the bias starts to dominate at lower level than that of the hard thresholding. The p value for the optimal soft thresholding seems to be around 35%. Figure 5.3 shows efficiency plot with varying p and this agrees with the conclusions given above.

Figure 5.1 shows a plot of the coefficients in black and the thresholded coefficients in red. It can be seen from the figure that the hard thresholded coefficients have few non-zero coefficients compared to the soft thresholded coefficients since the soft thresholding uses less severe threshold (about 35% of universal threshold as opposed to the hard threshold's 80% of universal threshold).

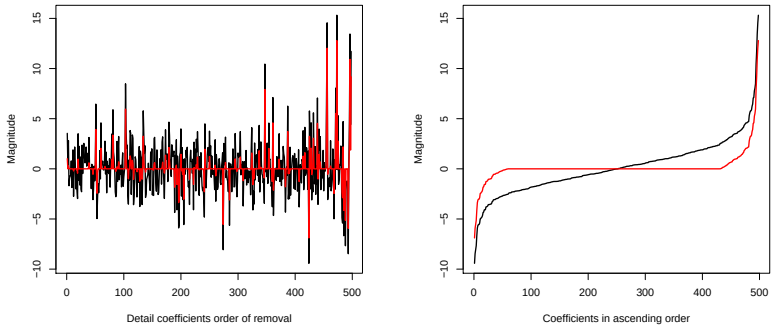
5.4 Block Thresholding - Across scale coefficient

The block thresholding methods proposed [11, 12, 13, 39, 40] grouping wavelet coefficients within a scale across translates. Since LOCAAT finds one coefficient at a time, the coefficients can be treated as across-scale coefficients. We group these LOCAAT coefficients, in the order they were calculated, into non-overlapping blocks of $M(= 3$ in our simulations) coefficients. If L denote the number of LOCAAT coefficients then we have the number of non-overlapping



(a) Hard threshold, coefficients in the order of removal

(b) Hard threshold, coefficients in ascending order



(c) Soft threshold, coefficients in the order of removal

(d) Soft threshold, coefficients in ascending order

Figure 5.1: Hard and soft thresholded LOCAAT coefficients for g^1 function on T-1 network. The number of detail coefficients $L = 498$, number of nodes $n = 500$, SNR =2.

blocks $n_b = \lfloor L/M \rfloor$. We then threshold each of these blocks if sum of squares of the coefficients within these blocks are larger than a threshold λ (see section 2.5.2 for a block thresholding on regular wavelet methods). Let $\mathscr{B}_j$ denote the indices of the j^{th} block.

$$\mathscr{B}_j = \{i_r : L - (j+1)M \leq r \leq (L - jM)\} \qquad i_r \in \mathcal{D}_1 \tag{5.6}$$

where $\mathcal{D}_1$ is the set containing the indices of the coefficients arising from complete LOCAAT transform. Let $B_j^2 = \sum_{i \in \mathscr{B}_j} d_i^{*2}$ denote the sum of squares of the detail coefficients in block j. We keep the block j if $B_j^2 > T$ where T is some

threshold choice. We choose $T = 2p\sigma^2 \log L$ where $0 < p \leq 1$. We estimate $\boldsymbol{\theta^*}$ by thresholding $\mathbf{d}^*$ where $\boldsymbol{\theta^*} = (\frac{\theta_k}{\overline{\sigma}_k})_{k=1}^n$, θ_k is the LOCAAT transform of the true function g_k and $\overline{\sigma}_k$ is the known variance change factor due to the LOCAAT transform.

$$\hat{\boldsymbol{\theta}}^* = \begin{cases} d_i^* \mathbb{I}(B_j^2 > T), & \forall i \in \mathscr{B}_j, j = 1, \ldots, n_b \\ d_i^*, & \text{otherwise.} \end{cases} \tag{5.7}$$

We then perform inverse transform with the new coefficients $\hat{\boldsymbol{\theta}}$ to estimate the true function $g(t)$. Note that we use $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta^*}.\, \overline{\boldsymbol{\sigma}}$ to estimate the true function $\mathbf{g}(t)$, where $\overline{\boldsymbol{\sigma}}$ is the vector of known variance change factor due to the LOCAAT transform and $\mathbf{g}(t) = (g_k(t))_{k=1}^n$.

$$\hat{\mathbf{g}}(t) = \sum_{l \in \mathcal{D}_r} \hat{\theta}_l \psi_l + \sum_{k \in \mathcal{S}_r} c_{rk} \phi_{rk}, \qquad r = 1, \ldots, n. \tag{5.8}$$

Figure 5.6(e) shows the results for block thresholding of LOCAAT coefficients of g_1 function with a type-1 network. Minimum Mean Squared Error (MMSE), in other words maximum efficiency, is achieved when $p = 1.05$. Bias appears after 60% but the decreasing variance dominate the Mean Squared Error (MSE) until $p = 1.05$. These results can be fine tuned by varying the block size M.

5.5 Block thresholding - Box Block Choice (BBC)

In the previous approach we just divided the LOCAAT coefficients into blocks based on the order in which the coefficients are produced and thresholded. The coefficients that are grouped together not necessarily actual spatial neighbours of each other. Figure 5.2 shows the LOCAAT coefficients in spatial domain. In our new approach, we divide coefficients into blocks based on their spatial position. We divide the unit square into a grid with grid spacing of δ_x (= 0.2) and δ_y (= 0.2). We group the coefficients in each grid square to form a block. The total number of grid boxes along x-axis is $n_X = 1/\delta_x$. Similarly, for the y-axis it is $n_Y = 1/\delta_y$. Let

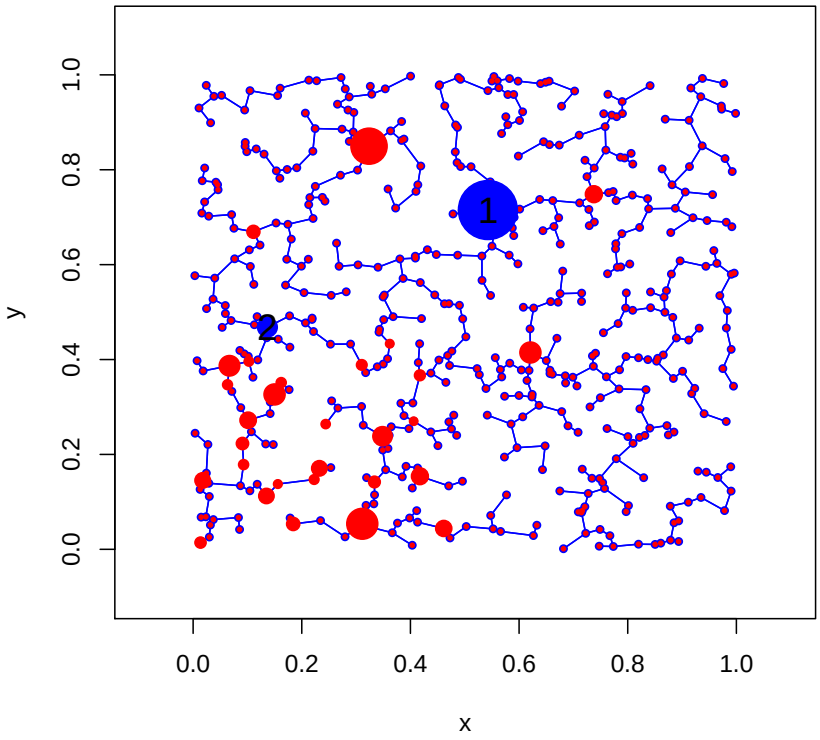


Figure 5.2: Spatial plot LOCAAT coefficients of $2-D$ Doppler function on a T-1 network. The number of detail coefficients $L = 498$, number of nodes $n = 500$, SNR =2. The detail coefficients are shown in red. The scaling coefficients in blue and are numbered.

$\mathscr{B}_{a,b}$ denote the set which contains the indices of the nodes related to the set $\mathcal{D}_1$ which are in grid square (a, b), where $1 \leq a \leq n_X$ and $1 \leq b \leq n_Y$. We keep the coefficients in the block contained in grid square (a, b) if $B^2_{a,b} = \sum_{k \in \mathscr{B}_{a,b}} d^{*2}_k > T$ with $T = 2p\sigma^2 \log L$ where $0 < p \leq 1$. We estimate $\boldsymbol{\theta^*}$ by thresholding the detail coefficients $\mathbf{d}^*$.

$$\hat{\boldsymbol{\theta}}^* = (d^*_k \mathbb{I}(B^2_{a,b} > T))_{k \in \mathscr{B}_{a,b}, a=1,\ldots,n_X, b=1,\ldots,n_Y} \tag{5.9}$$

We then perform the inverse transform, similar to (5.8), using modified coefficients $\mathbf{d}^*$ to get an estimate for the true function $\mathbf{g}(t)$.

Figure 5.7(c) shows the results for BBC thresholding approach of LOCAAT coefficients of g^1 function with type-1 network settings. MMSE is achieved when

$p = 1.28$. The results could be fine tuned by automatically selecting the box size. In the spatio-temporal case we extend this box in the spatial domain to include the past and future state nodes of those who are within the grid square (a, b).

5.6 Block Thresholding - Neighcoeff

Neighcoeff method is introduced in [10], which is a term by term thresholding method that depends on the coefficient itself and the coefficients of immediate neighbours. The neighbouring coefficients also contribute to thresholding and the threshold is less severe than the one in the MAD technique (see section 2.5.3). We adapt this technique to LOCAAT by just taking the coefficients of a node and its neighbours to construct a block of coefficients, let number of blocks be k. The number of closest neighbours we choose will depend on the design. In the T-1 network, most of the nodes will have few neighbours. For a T-2 network there may be many neighbours for each node, however the correlation in the far away nodes will be less. Therefore we choose 2 closest neighbours for each node k (in both network types) and denote them by J_k^{near}. Let $\mathscr{B}_k = \{J_k^{\text{near}} \cup k\}$ denote the block for node k. We decide to keep a coefficient if $B_k^2 = \sum_{i \in \mathscr{B}_k} d_i^{*2} > T$ with $T = 2p\sigma^2 \log L$ where $0 < p \leq 1$. By repeating this for all the coefficients we estimate $\boldsymbol{\theta}^*$,

$$\hat{\boldsymbol{\theta}}^* = d_k^* \mathbb{I}(B_k^2 > T). \tag{5.10}$$

We perform inverse transform similar to (5.8) to estimate the true function $\mathbf{g}(t)$. Figure 5.7(a) shows the results for neighcoeff thresholding of LOCAAT coefficients of g^1 function with type-1 network settings. MMSE is achieved when $p = 0.76$. When time series data is available, this block thresholding in spatial method can take future and past state coefficients, in addition to the detail coefficients of spatial neighbours as explained above, into the decision making. This approach improves the results significantly. The thresholding efficiency using the neighcoeff

approach, for a g^1 function on a type-1 network scenario with snr=2, is 1.58 (from table 5.1). When we apply the second neighcoeff approach of considering past and future state coefficients, the efficiency is 2.01 for the same simulation conditions used for the first approach. This efficiency is close to the results of spatio-temporal denoising presented in table 5.2.

Results

Figure 5.3 shows the results of efficiencies for various thresholding methods discussed so far. Soft and empirical Bayes (`ebayesthresh`) thresholding methods give the best efficiency of approximately 1.77 (from table 5.1). Block thresholding and hard thresholding methods also give an acceptable performance. Figure 5.4 shows the efficiency results for the spatio-temporal method with both overlapping and non-overlapping cases. Figure 5.8 and figure 5.9 shows the bias variance and mse analysis for various threshold methods in the spatio-temporal setting. We see that block threshold generally performs better than the others and the overall efficiency for all the methods increases in the spatio-temporal setting and especially the overlapping moving window type performs well. Table 5.2 shows the maximum efficiencies achieved for various methods and their corresponding $p = p^*$ that maximises the efficiency for that particular method. All the results presented are average of 100 independent experiments. Further results spatial network case with fewer number of nodes (n=50 and n=150) are shown in appendix B.

5.7 Mean correction method

We found that the previous thresholding methods introduce bias into our estimation. Some methods have severe bias issues compared to the other methods. We can correct the mean after thresholding to compensate for the bias and this mean correction can be performed with all the above methods to improve the bias at

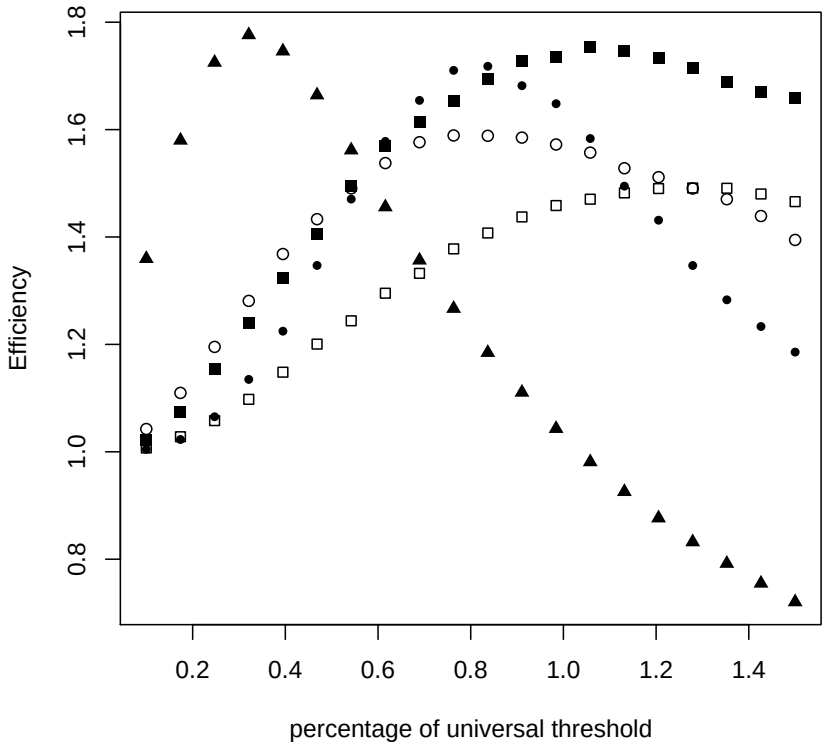


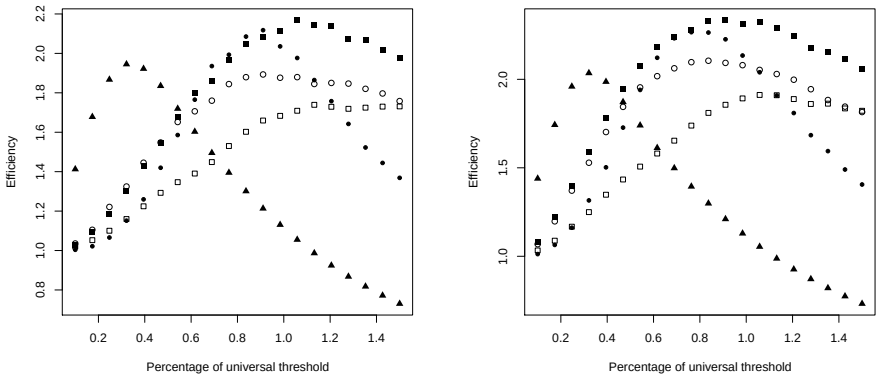
Figure 5.3: Efficiency results for g^1 function on a type-1 network with 500 nodes with SNR = 2, Hard threshold(=•), soft threshold(=▲), block threshold(=■), BBC(=□) and neighcoeff(=○) against p=varying percentage of universal threshold. Each point is an average of 100 experiments.

the expense of increased variance. We calculate a mean value for each node k based on the observations of its neighbours $j \in J_k$ and the node k itself denoted by μ_k^{obs}.

$$\mu_k^{\text{obs}} = \frac{\sum_{j \in J_k} y_j + y_k}{n_k + 1}, \tag{5.11}$$

where $n_k = |J_k|$ and y_k is the noisy observation at node k. We then find the estimated mean, μ_k^{est}, for each node in a similar fashion after denoising.

$$\mu_k^{\text{est}} = \frac{\sum_{j \in J_k} \hat{g}_j + \hat{g}_k}{n_k + 1}, \tag{5.12}$$



(a) Non-overlapping moving window (b) Overlapping moving window

Figure 5.4: Efficiency results for g^1 function on a type-1 network with 500 nodes with SNR = 2 in spatio-temporal setting, Hard threshold(=●), soft threshold(=▲), block threshold(=■), BBC(=□) and neighcoeff(=○) against p=varying percentage of universal threshold. Each point is an average of 100 experiments.

Method	Maximum Efficiency	p^*
Hard threshold(HT)	1.718	0.837
Soft threshold(ST)	1.776	0.321
Block threshold(BT)	1.754	1.058
Neighcoeff(NC)	1.589	0.763
BBC	1.491	1.279
Mean Corrected HT	1.426	2.373
Mean Corrected ST	1.427	0.858
Mean Corrected BT	1.421	2.500
Mean Corrected NC	1.379	2.500
Mean Corrected BBC	1.354	2.500
Ebayesthresh	1.777	NA

Table 5.1: Maximum efficiencies and the corresponding $p* = \arg\max_p \text{eff}$ for various thresholding methods for g^1 function on a type-1 network with SNR = 2 and n = 500. Each entry is an average of 100 repetition.

where $\hat{g}_k$ is the estimate of the true function g_k at node k. We then find the difference in the mean levels and add to the denoised value at node k.

$$\mu_k^{\text{diff}} = \mu_k^{\text{obs}} - \mu_k^{\text{est}} \tag{5.13}$$

$$\hat{g}_k^{'} = \hat{g}_k + \mu_k^{\text{diff}}. \tag{5.14}$$

Methods	Maximum efficiency		p^*	
	Non-overlapping	Overlapping	Non-overlapping	Overlapping
Hard threshold	2.112	2.266	0.911	0.763
Soft threshold	1.945	2.035	0.321	0.321
Block threshold	2.168	2.334	1.058	0.911
Neighcoeff	1.893	2.105	0.911	0.837
BBC	1.739	1.912	1.132	1.058
Ebayesthresh	2.090	2.203	NA	NA

Table 5.2: Maximum efficiencies and the corresponding $p* = \arg\max_p \text{eff}$ for various thresholding methods for g^1 function on a type-1 network with SNR = 2 and n = 500 in spatio-temporal setting. Each entry is an average of 100 repetition.

This effectively ensures that the mean of the estimation stays close to the mean of the noisy data. This is a data based method similar in spirit to the update step in lifting which maintains the mean of the data through the transform. Since some nodes can be far away from the node (less correlation), we prefer to use only the nearest neighbours to find the mean instead of taking all neighbours into consideration. Assume we have n_k^* nearest neighbours whose indices are J_k^*. $J_k^* \subseteq J_k$.

We can apply this to any of our proposed methods. Figure 5.5 shows the efficiency plot for all the methods discussed earlier with the mean correction. The maximum efficiency approaches around 1.4 for g^1 function with the simulation settings above. It is clear by comparing the figure 5.3 and figure 5.5 that the mean correction method is much more consistent and less sensitive to the choice of p. In the real world problems, since the true function is usually unknown, it is difficult to determine the choice of p for various other methods earlier (unless we use an approximation of the true function by taking average of nearest neighbours' observations). Therefore we recommend the mean correction method in such scenario.

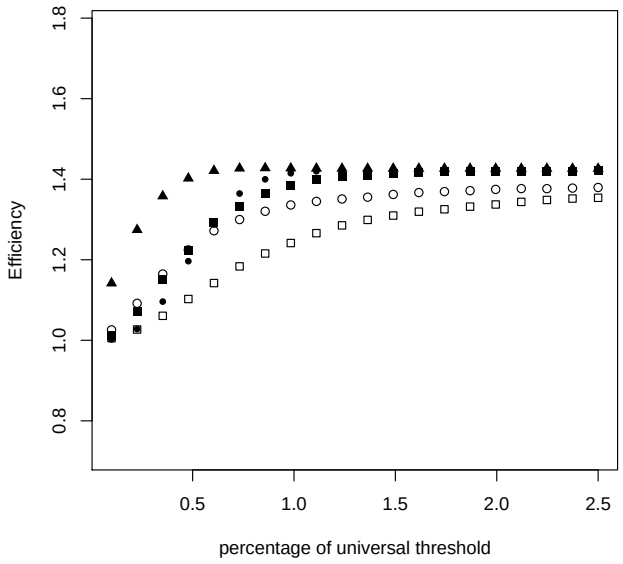
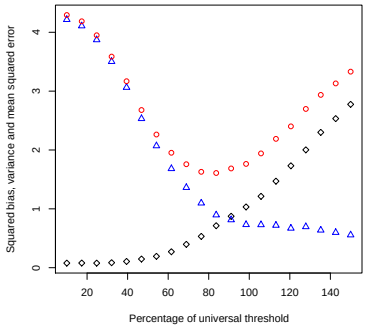
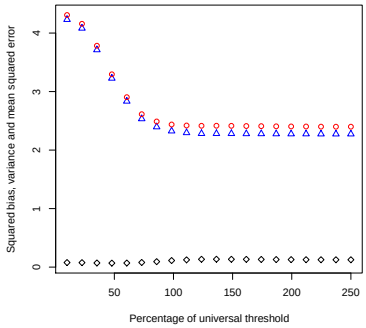


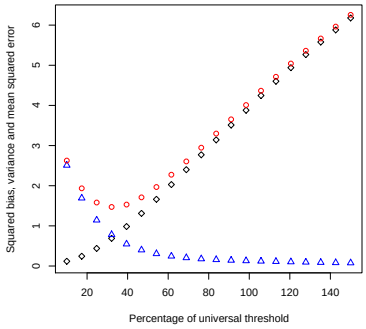
Figure 5.5: Efficiency results for g^1 function on a type-1 network with 500 nodes with SNR = 2 mean corrected, hard threshold(=•), soft threshold(=▲), block threshold(=■), BBC(=□) and neighcoeff(=○) against p=varying proportion of universal threshold. Each point is an average of 100 experiments.



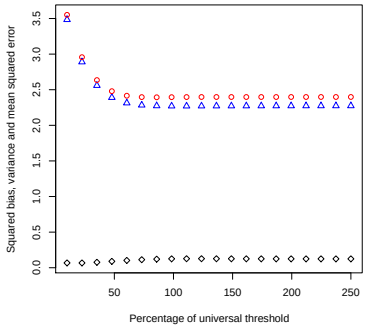
(a) Hard Thresholding



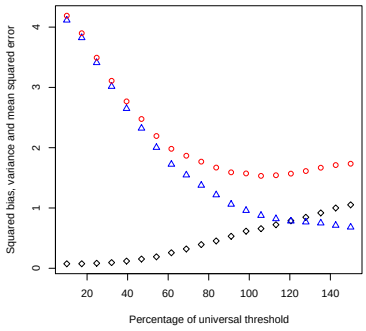
(b) Hard thresholding with mean correction



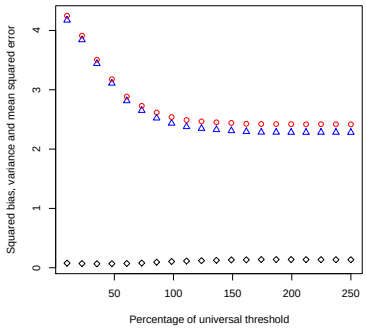
(c) Soft Thresholding



(d) Soft Thresholding with mean correction

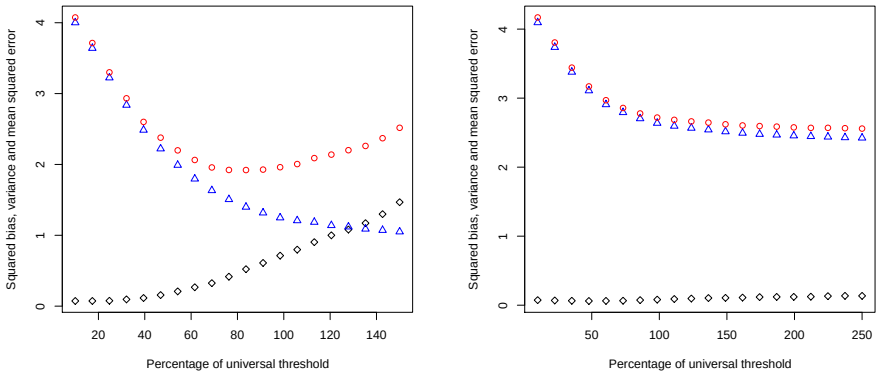


(e) Block Thresholding

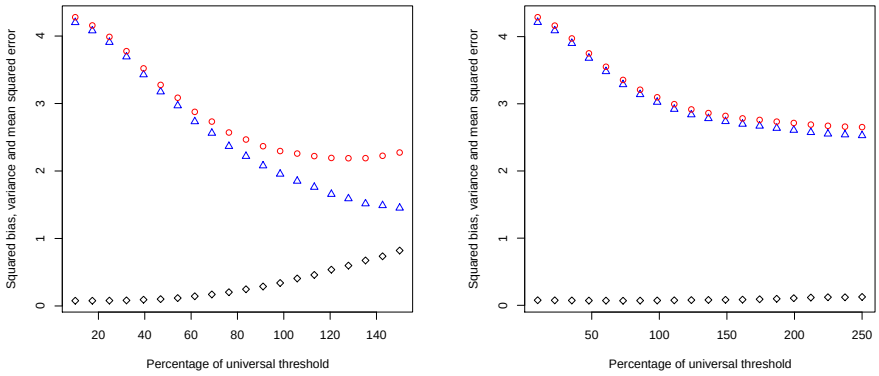


(f) Block Thresholding with mean correction

Figure 5.6: Various thresholding results for Type-1 network with 500 nodes with SNR = 2, Average squared bias(=$\diamond$), variance(=$\triangle$) and mean squared error(=$\circ$) against $p\hat{\sigma}\sqrt{2\log n}$, p=varying proportion of universal threshold. Each point is an average of 100 experiments.

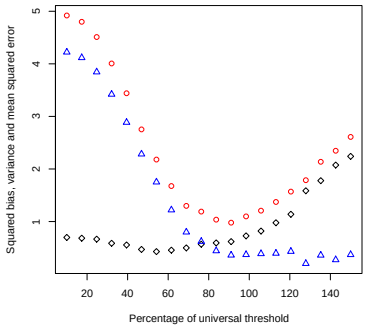


(a) Neigh Coeff method (b) Neigh Coeff with mean correction

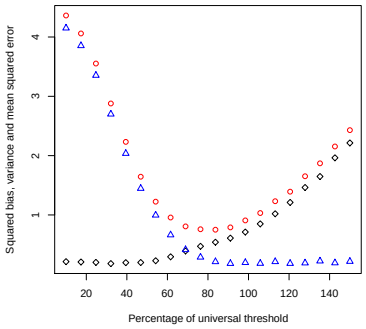


(c) BBC (d) BBC with mean correction

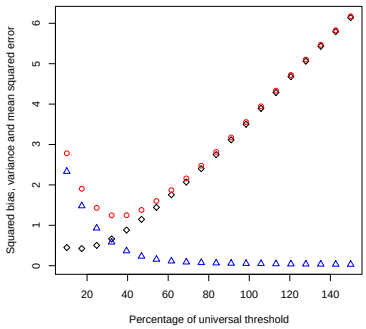
Figure 5.7: Various thresholding results for Type-1 network with 500 nodes, Average squared bias(=◇), variance(=△) and mean squared error(=○) against $p\hat{\sigma}\sqrt{2\log n}$, p=varying proportion of universal threshold. Each point is an average of 100 experiments.



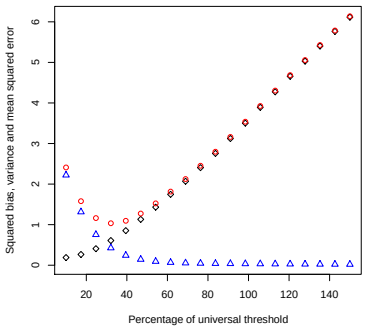
(a) Hard Thresholding



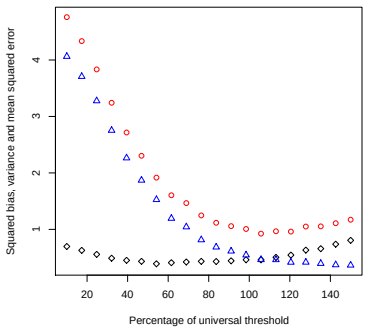
(b) Hard thresholding with mean correction



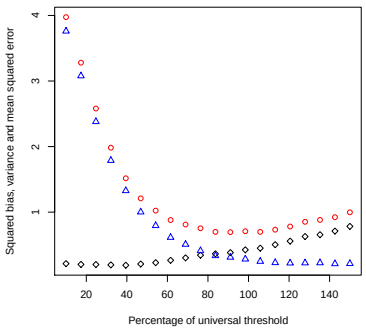
(c) Soft Thresholding



(d) Soft Thresholding with mean correction

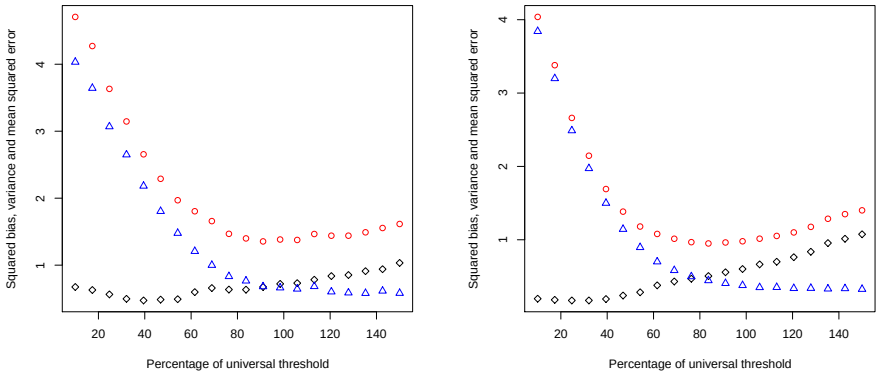


(e) Block Thresholding



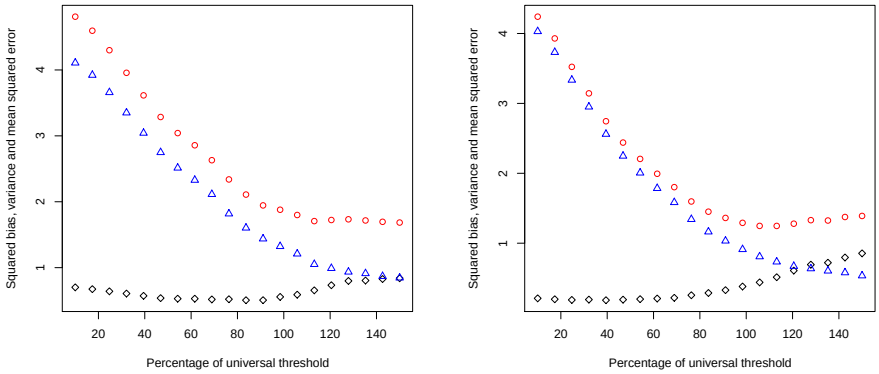
(f) Block Thresholding with mean correction

Figure 5.8: Various thresholding results for Type-1 network with 500 nodes with SNR = 2 in the spatio-temporal setting, figures on the left hand side are non-overlapping moving window case and the figures on the right are the overlapping moving window case. Average squared bias(=$\diamond$), variance(=$\triangle$) and mean squared error(=$\circ$) against $p\hat{\sigma}\sqrt{2\log n}$, p=varying proportion of universal threshold. Each point is an average of 100 experiments.



(a) Neigh Coeff method

(b) Neigh Coeff with mean correction



(c) BBC

(d) BBC with mean correction

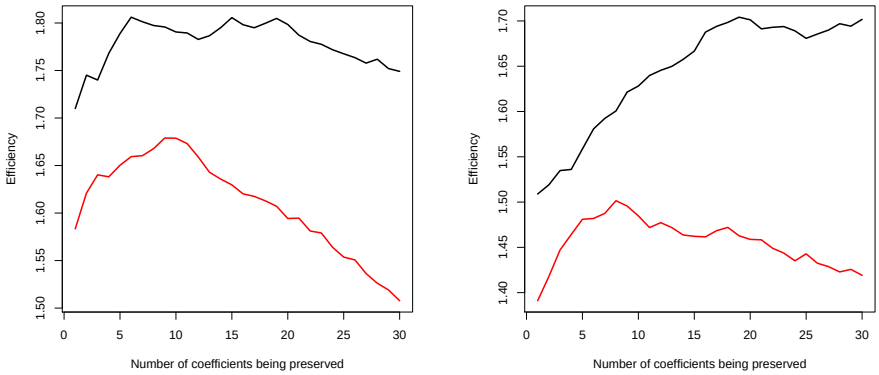
Figure 5.9: Various thresholding results for Type-1 network with 500 nodes with SNR = 2 in the spatio-temporal setting, figures on the left hand side are non-overlapping moving window case and the figures on the right are the overlapping moving window case. Average squared bias(=$\diamond$), variance(=$\triangle$) and mean squared error(=$\circ$) against $p\hat{\sigma}\sqrt{2\log n}$, p=varying proportion of universal threshold. Each point is an average of 100 experiments.

5.8 Improving estimation by identifying sensitive coefficients of estimation

Thresholding a small coefficient may have big impact especially if it is in a region of few nodes [43]. We investigate a way to find these sensitive coefficients (akin to points of high leverage in ordinary regression). We threshold the coefficients first and then identify the coefficients that were set to zero by the thresholding operation, we denote this set by J_{zero}. Now, we start with the full set of coefficients and setting one coefficient to zero at a time and perform the inverse transform. We calculate the squared error $\text{Err}(j) = \sum_{k=1}^{n}(f_k - \tilde{f}_k^j)^2$ where n is the number of nodes, f_k is the noisy observation at node k and $\tilde{f}_k^j$ is the inverse transform of LOCAAT coefficients with coefficient j set to zero and $j \in J_{\text{zero}}$. We repeat this step for all $j \in J_{\text{zero}}$ and calculate the error function $\text{Err}(j)$. Now we can identify the nodes whose coefficients, when set to zero, result in large error. Then, when performing estimation, we avoid some of these coefficients from being thresholded. This method improves the performance.

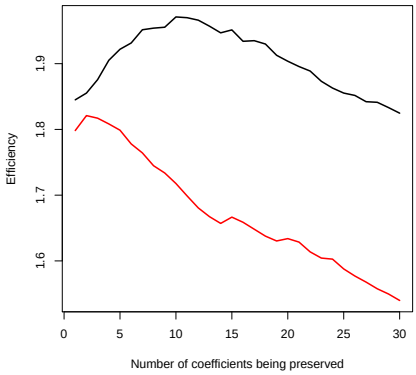
We determine the number of coefficients to avoid from being thresholded in the following way. We arrange the error $\text{Err}(j)$ in descending order. We perform $|J_{\text{zero}}|$ number of inverse transforms, each time by avoiding $1, \ldots, j$ coefficients being replaced by zero by the threshold operation and calculate the efficiency (as in (5.5)) of resulting estimation. We repeat this step until avoiding all the coefficients , i.e. $1, \ldots, |J_{\text{zero}}|$, being replaced by zero (this will result in efficiency = 1 since no thresholding is applied to the detail coefficients). Now we avoid the number of coefficients (that are avoided being replaced by zero) that maximise the efficiency.

Figure 5.10 shows the results of improving the efficiency by avoiding thresholding the sensitive coefficients for MST type network. It is clear from the figures that we can improve the results significantly by avoiding about 10 to 15 coefficients, whose error is large, in a 500 node simulation study on a T-1 network. Figure 5.11 shows

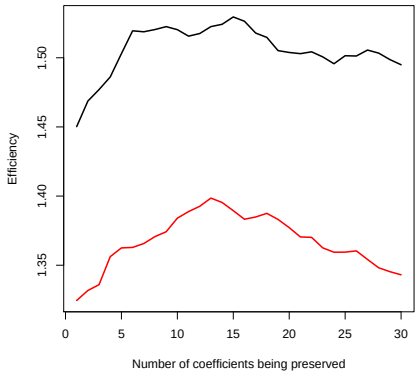


(a) Function g^1

(b) Function g^2



(c) maartenfunc

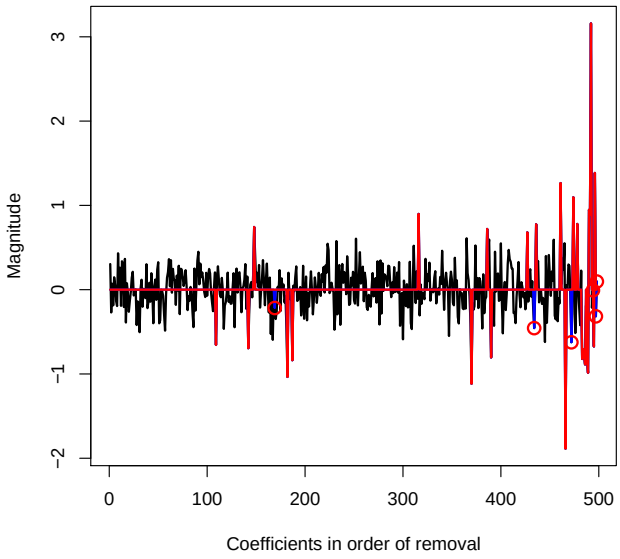


(d) 2D Doppler function

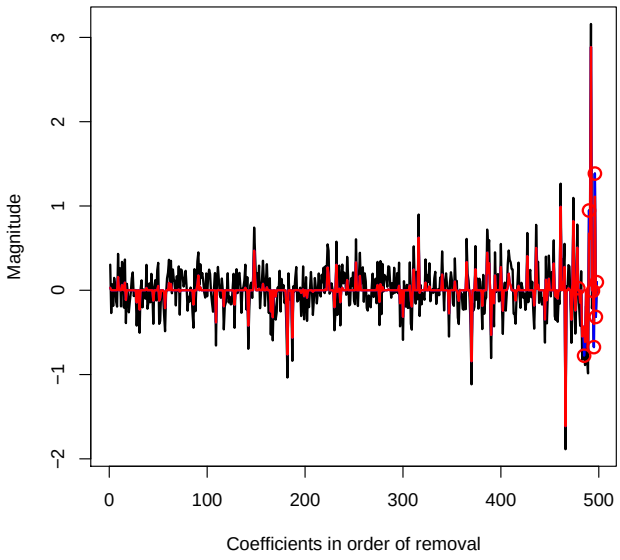
Figure 5.10: Efficiency Vs Number of coefficients avoided being thresholded whose error is large. Plot in red shows hard thresholding results and the black one for soft thresholding. Network T-1 used with various test functions with SNR=2, $n = 500$.

the coefficients in the order of removal in black, ordinary thresholded coefficients (either hard or soft threshold) with the percentage of universal threshold (hard = 80%, soft = 35%) shown in red and the preserved coefficients are shown in blue. It can be noted that the preserved coeffients come from the latest removed points which led to the conclusion in section 5.15.

This method is extremely inefficient in terms of the amount of computing involved since we have to perform many ($|J_{\text{zero}}|$) inverse transforms by avoiding some coefficients being replaced by zero. In a real world scenario, usually the true func-



(a) Hard threshold



(b) Soft threshold

Figure 5.11: Detail coefficients (in the order of removal) for g^2 function on a T-1 network. The number of detail coefficients $L = 498$, number of nodes $n = 500$, SNR =2. Detail coefficients in black, ordinary thresholded coefficients in red (with $p^* = 0.8$ for hard threshold and $p* = 0.35$ for soft threshold), where p^* is the proportion of universal threshold that minimise the MSE for the chosen threshold method, and preserved coefficients in blue with red circle on top.

tion that we want to estimate is usually unknown (thus difficult to calculate the efficiency). Therefore, when we use this method in real world application we may calculate the efficiency using the approximation based on the nearest neighbours and this method still gives similar results.

5.9 Ordinary Cross Validation (OCV)

In ordinary cross validation the optimal threshold is chosen to minimise the mean squared error function $R(\lambda)$

$$R(\lambda) = \frac{1}{n}||\hat{\mathbf{f}}_\lambda - \mathbf{g}||^2, \tag{5.15}$$

where $||.||$ means L_2 norm, $\hat{\mathbf{f}}_\lambda$ is the estimator of the true function $\mathbf{g}$ and n is the number of nodes. We assume the true signal is regular so that g_k can be well-approximated by linear combination of its neighbours. To validate the performance of a thresholding parameter at a given data point f_k, we leave this point out and interpolate this intermediate point and call this $\tilde{f}_k$.

$$\tilde{f}_k = \frac{1}{|J_k^{\text{near}}|} \sum_{k \in J_k^{\text{near}}} f_k \tag{5.16}$$

where J_k^{near} are the indices of the nearest neighbours of node k. We assume this $\tilde{f}_k$ is relatively noise free.

$$\tilde{\mathbf{f}}^k = (f_1, \ldots, \tilde{f}_k, \ldots, f_n)^T. \tag{5.17}$$

We perform the LOCAAT transform on this modified data $\tilde{\mathbf{f}}^i$ and perform a chosen threshold operation (we choose soft thresholding for its continuous nature) with a choice of threshold parameter λ. We perform inverse transform and denote the resulting estimator as $\tilde{f}_{\lambda i}$. We repeat these steps for all the nodes and calculate

the ordinary cross validation function

$$O(\lambda) = \frac{1}{n}\sum_{k=1}^{n}(f_k - \tilde{f}_{\lambda k})^2 \tag{5.18}$$

For a small value of λ, the error term is dominated by the noise, while for a large λ error is large because the signal is deformed.

This method is extremely inefficient because it involves n forward LOCAAT transforms and n inverse transforms for a choice of the threshold parameter λ. Therefore we apply the following modification to the OCV so that the computing is limited to 1 forward transform for all the choices of λ and 1 inverse transform per choice of λ.

5.9.1 Modified Ordinary Cross Validation (MOCV)

Since the true function $\mathbf{g} \equiv (g_k)_{k=1}^n$ can be well-approximated by the linear combination of its nearest neighbours' observations, we obtain an approximation for the true function $\mathbf{g}$ as follows,

$$\tilde{f}_k = \frac{1}{|J_k^{\text{near}}| + 1}\left(\sum_{i \in J_k^{\text{near}}} f_i + f_k\right), \qquad k \in \mathcal{N}_1, \tag{5.19}$$

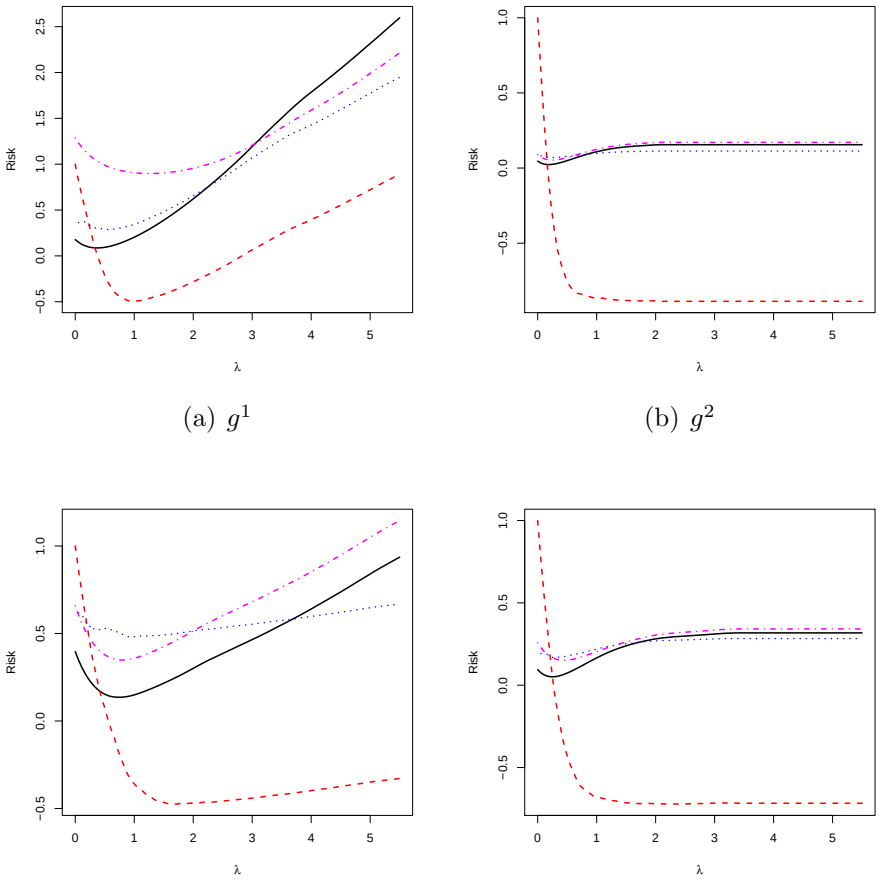
where J_k^{near} is the set that stores the indices of the closest neighbours of node k. We repeat this step for all the nodes and obtain $\tilde{\mathbf{f}}$ which we can assume to be a relatively noise free version of $\mathbf{g}$. Now we perform LOCAAT transform on the noisy signal $\mathbf{f}$.

$$\tilde{\mathbf{f}} = (\tilde{f}_1, \ldots, \tilde{f}_k, \ldots, \tilde{f}_n)^T. \tag{5.20}$$

We threshold the LOCAAT coefficients with threshold λ and calculate the difference $\tilde{f}_k - f_{\lambda k}$. We now compute the cross validation function $O^{\text{m}}(\lambda)$ as follows,

$$O^{\text{m}}(\lambda) = \frac{1}{n}\sum_{k=1}^{n}(\tilde{f}_k - f_{\lambda k})^2 \tag{5.21}$$

This way we quickly mimic the MSE function given in (5.15) like in other cross validation methods. See figures 5.12, 5.13, 5.14 to see how well our MOCV mimics the MSE function.



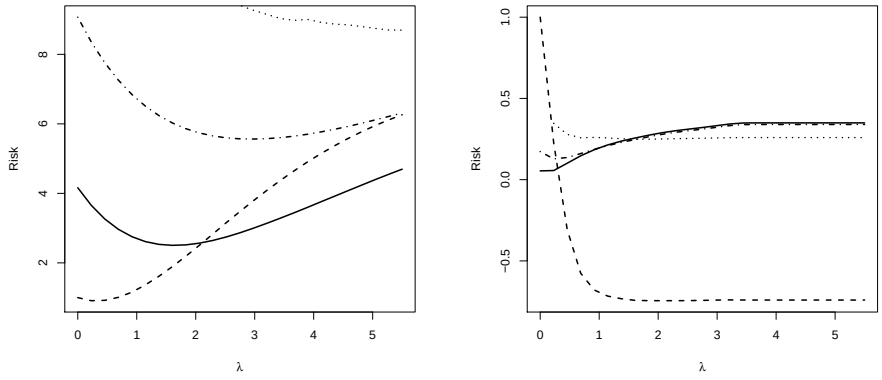
(a) g^1 (b) g^2

(c) maartenfunc (d) 2D Doppler

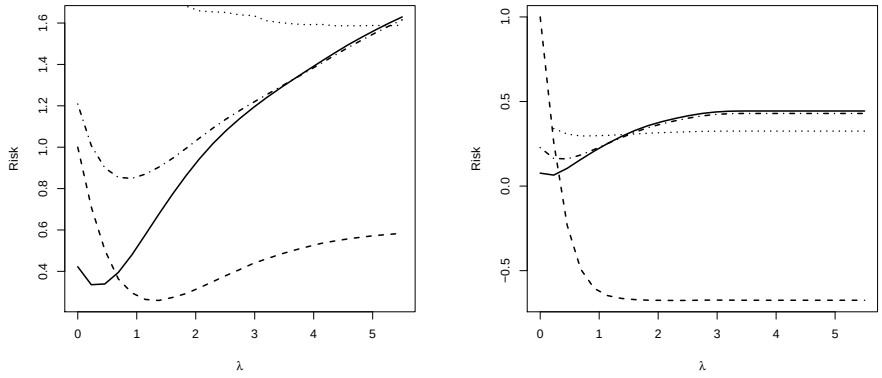
Figure 5.12: MSE(—-), SURE(- - - -), GCV(....), MOCV(-.-.-).

5.10 Stein's Unbiased Risk Estimator (SURE)

The Sureshrink method was introduced in [33] and it uses threshold selection by SURE which was introduced in [81] by Stein. We will also use this for threshold selection and compare with other threshold selection methods such as MOCV and Generalised Cross Validation (GCV). We calculate the SURE value as follows (see



(a) g^1 (b) g^2



(c) maartenfunc (d) 2D Doppler

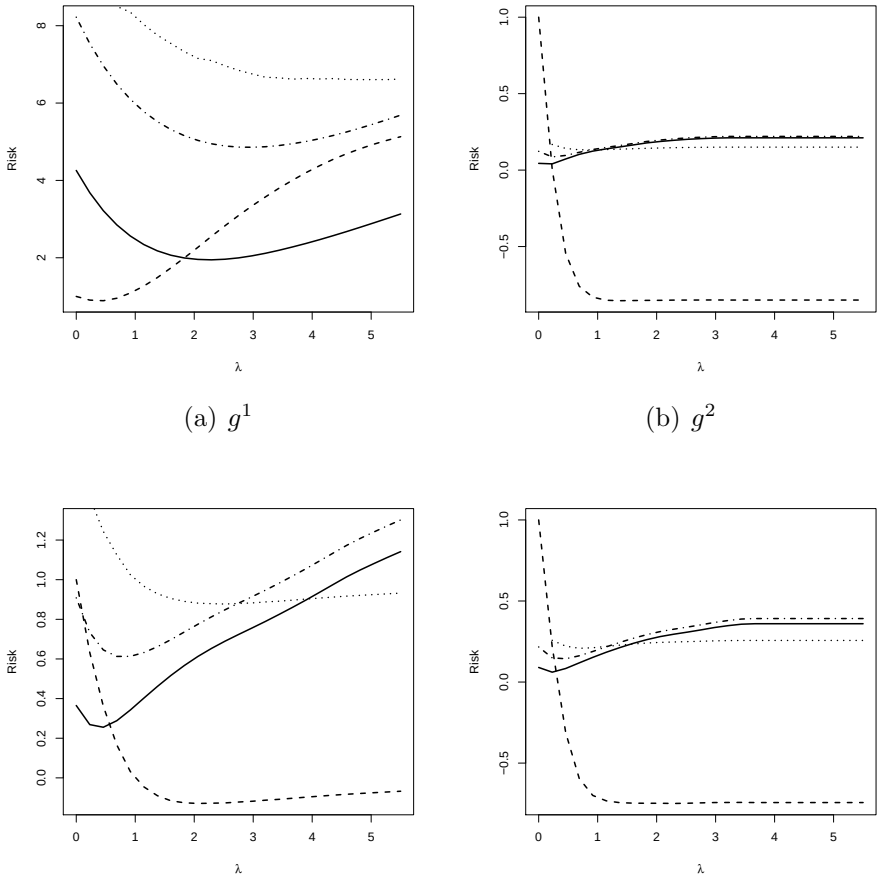
Figure 5.13: Spatio-temporal non-overlapping window case. MSE(—-), SURE(- - - -), GCV(....), MOCV(-.-.-).

section 2.5.4 for details),

$$S(\lambda) = \frac{1}{L}[L - 2N_0 + \sum_{i=1}^{L} \min(d_k^*, \lambda)^2], \tag{5.22}$$

where L is the number of detail coefficients, N_0 is the number of coefficients that are replaced by zero by the threshold parameter λ.

See figures 5.12, 5.13, 5.14 to see how well the SURE mimics the MSE function.



(a) g^1 (b) g^2

(c) maartenfunc (d) 2D Doppler

Figure 5.14: Spatio-temporal overlapping window case. MSE(—-), SURE(- - - -), GCV(....), MOCV(-.-.-).

5.11 Generalised Cross Validation (GCV)

A general description of GCV is given in section 2.5.5. The GCV score $G(\lambda)$ is calculated as [44],

$$G(\lambda) = \frac{\frac{1}{n}\sum_{k=1}^{n}(f_k - f_{\lambda k})^2}{(N_0/n)^2}, \tag{5.23}$$

where f_k is the observation at node k, $f_{\lambda k}$ is the estimator of the true function value g_k at node k using the threshold λ, n is the number of nodes and N_0 is the number of coefficients that are set to zero by the threshold operation. This method will perform slowly as it involves having to invert the transform. However the GCV

score can also be approximated from wavelet domain coefficients as follows [43],

$$G(\lambda) = \frac{\frac{1}{L}\sum_{k=1}^{n}(d_k^* - d_{\lambda k}^*)^2}{(N_0/L)^2}, \tag{5.24}$$

where d_k^* is the detail coefficient at node k, L is the number of detail coefficients, $d_{\lambda k}^* = d^*\mathbb{I}(d^* > \lambda)$ and N_0 is the number of coefficients replaced by zero. See figures 5.12, 5.13, 5.14 to see how well GCV mimics the MSE function.

5.12 Optimising the risk estimators

Having seen some risk estimators that mimic the MSE function in (5.15), now we will use this information to find $\lambda = \lambda^*$ that minimises the MSE function

$$\lambda^* = \arg\min_{\lambda} R(\lambda). \tag{5.25}$$

We find the following by optimising the MOCV score, the SURE score and the GCV scores respectively $\lambda_{O^m}^* = \arg\min_\lambda O^m(\lambda)$, $\lambda_S^* = \arg\min_\lambda S(\lambda)$ and $\lambda_G^* = \arg\min_\lambda G(\lambda)$. In theory these $\lambda_{O^m}^* \approx \lambda_S^* \approx \lambda_G^* \approx \lambda^*$. It can be seen from figures 5.12, 5.13, 5.14 that this is true for some cases.

We perform optimisation in 1-D using the combination of *golden section search* and *successive parabolic interpolation* methods. There is a built in function in 'R' called `optimize` which performs this. See section 2.8 for detail. Table 5.3

	λ^*			Efficiency		
Signals	SURE	GCV	MOCV	SURE	GCV	MOCV
g^1	0.397	2.576	2.839	1.195	1.644	1.700
g^2	3.524	0.411	0.358	0.049	1.350	1.450
maartenfunc	1.805	1.002	1.785	1.266	1.676	1.785
2D Doppler	2.343	0.401	0.496	0.370	1.350	1.252

Table 5.3: Optimising the risk and corresponding efficiencies for type 1 network (spatial network) with SNR = 2 and n = 500.

Signals	λ^*			Efficiency		
	SURE	GCV	MOCV	SURE	GCV	MOCV
g^1	0.316	4.106	3.015	1.126	1.195	1.316
g^2	1.947	1.149	0.270	0.218	0.537	1.012
maartenfunc	1.713	2.618	0.825	0.798	0.667	1.081
2D Doppler	1.938	1.027	0.349	0.210	0.685	1.031

Table 5.4: Optimising the risk and corresponding efficiencies for type 1 network in spatio-temporal setting with non-overlapping window. SNR = 2 and n = 500.

Signals	λ^*			Efficiency		
	SURE	GCV	MOCV	SURE	GCV	MOCV
g^1	0.384	3.808	2.949	1.186	1.966	2.039
g^2	1.573	0.654	0.252	0.384	1.027	1.461
maartenfunc	2.087	2.085	0.804	1.211	1.281	1.664
2D Doppler	1.992	0.687	0.370	0.380	1.266	1.552

Table 5.5: Optimising the risk and corresponding efficiencies for type 1 network in spatio-temporal setting with overlapping window. SNR = 2 and n = 500.

shows the optimised efficiency for various risk estimators and their corresponding λ^* that minimise the respective risk estimator for various test functions. It is clear that MOCV performs better however GCV performs to an acceptable level. Notice that λ^* for MOCV and GCV are closer to 40% of universal threshold which is what we found for soft thresholding (note that all the risk estimators use soft thresholding because of its continous nature) as the best threshold choice. Table 5.4 and table 5.5 shows the similar results in spatio-temporal networks. General outcome is that MOCV performs better however considering computing efficiency GCV is prefered. Table B.1 in appendix B shows the results for the T-2 network with different test functions.

5.13 Improving spatio-temporal denoising by smoothing coefficients

When time series data is available, we can do coefficient smoothing in both the time and wavelet domain to improve the denoising results. First, we present the spatial network method with smoothing coefficients in wavelet domain and then we present the spatio-temporal network with non-overlapping window method with smoothing coefficients in wavelet domain and finally, we present the spatio-temporal network overlapping window method with smoothing coefficients in wavelet domain.

We consider the following model,

$$f_{t,k} = g_{t,k} + \epsilon_{t,k}, \tag{5.26}$$

where $f_{t,k}$ is the observation at node k at time t, $g_{t,k}$ is the underlying true signal, varying spatially and temporally, we are trying to estimate and $\epsilon_{t,k}$ is iid Gaussian error with mean zero and variance σ^2. For our simulations, we generate $g_{t,k}$ as follows,

$$g_{t,k} = g(x_k, y_k, t), \tag{5.27}$$

where $g(x_k, y_k, t)$ is the time varying version of one of the test functions defined earlier in section 3.2.3.

5.13.1 Via sequential and separate transforms using spatial network

We perform the LOCAAT transform on the the spatial data sequentially and separately through time.

$$d_{t,k} = \theta_{t,k} + \acute{\epsilon}_{t,k} \tag{5.28}$$

where $d_{t,k}$, $\theta_{t,k}$, $\acute{\epsilon}_{t,k}$ are the LOCAAT transforms of $f_{t,k}$, $g_{t,k}$ and $\epsilon_{t,k}$ respectively. Note that $\acute{\epsilon}_{t,k}$ no longer has variance as σ^2 since the LOCAAT transform is non-orthogonal. However, the variance change due to the lifting transform can be calculated as given in section 2.4.3 [46]. Let us denote this by $\overline{\sigma}_k^2$.

$$\acute{\epsilon}_{t,k} \sim N(0, \overline{\sigma}_k^2 \sigma^2) \tag{5.29}$$

Since there is temporal correlation, we move a running mean window of size 3 (we can use higher window size) to replace the middle layer with the averaged coefficients. We move this window along until the last layer of coefficients. Let $\tilde{d}_{t,k}$ denote the new, smoothed coefficients for node k at time t,

$$\tilde{d}_{t,k} = \begin{cases} \frac{d_{(t-1),k}+d_{t,k}+d_{(t+1),k}}{3}, & \text{if } 1 < t < T \\ d_{t,k}, & \text{otherwise,} \end{cases} \tag{5.30}$$

and assume

$$\tilde{d}_{t,k} = \tilde{\theta}_{t,k} + \tilde{\epsilon}_{t,k}, \tag{5.31}$$

where $\tilde{\theta}_{t,k} = \frac{\theta_{t-1,k}+\theta_{t,k}+\theta_{t+1,k}}{3} \approx \theta_{t,k}$ and $\tilde{\epsilon}_{t,k} = \frac{\acute{\epsilon}_{t-1,k}+\acute{\epsilon}_{t,k}+\acute{\epsilon}_{t+1,k}}{3}$. Note that we assume $\tilde{\theta}_{t,k} \approx \theta_{t,k}$ because the signal is sparse and it is distinguished from the noise by the LOCAAT transform and also the signal has temporal correlation, i.e. $\theta_{t-1,k} \approx \theta_{t,k}$. Now, because of this wavelet domain coefficient smoothing, the noise variance is reduced such that $\tilde{\sigma} < \sigma$,

$$\tilde{\epsilon}_{t,k} \sim N(0, \overline{\sigma}_k^2 \tilde{\sigma}^2). \tag{5.32}$$

This $\tilde{\sigma}$ can be estimated by the variogram method or MAD technique (see section 4.3) and we denote it by $\hat{\tilde{\sigma}}$. Then we form:

$$\frac{\tilde{d}_{t,k}}{\overline{\sigma}_k \hat{\tilde{\sigma}}} = \frac{\tilde{\theta}_{t,k} + \tilde{\epsilon}_{t,k}}{\overline{\sigma}_k \hat{\tilde{\sigma}}} \tag{5.33}$$

$$d^*_{t,k} = \theta^*_{t,k} + \epsilon^*_{t,k} \tag{5.34}$$

where $d^*_{t,k} = \frac{\tilde{d}_{t,k}}{\overline{\sigma}_k \hat{\tilde{\sigma}}}$, $\theta^*_{t,k} = \frac{\tilde{\theta}_{t,k}}{\overline{\sigma}_k \hat{\tilde{\sigma}}}$ and $\epsilon^*_{t,k} = \frac{\tilde{\epsilon}_{t,k}}{\overline{\sigma}_k \hat{\tilde{\sigma}}}$. Now the error term $\epsilon^*_{t,k} \overset{\text{approx}}{\sim} N(0,1)$ and we can perform thresholding of these $d^*_{t,k}$ to get the estimates $\hat{\theta}^*_{t,k}$. Now we have to carry out the following operation before performing the inverse transform: $\hat{\theta}_{t,k} = \overline{\sigma}_k \hat{\theta}^*_{t,k}$. We can now perform inverse transform on the estimated coefficients $\hat{\theta}_{t,k}$ to estimate the underlying true signal $\hat{g}_{t,k}$.

Table 5.6 shows the performance improvement of spatial network based spatio-

Signals	No smoothing	With smoothing
g^1	1.726	2.464
g^2	1.387	2.083
maartenfunc	1.721	2.256
2D Doppler	1.212	2.037

Table 5.6: Efficiencies compared for spatial network based denoising. The first column shows estimation efficiency in an ordinary estimation (no coefficient smoothing involved) and the second column showing estimation efficiency by smoothing coefficients in the wavelet domain. Type-1 network, SNR=2 , $n = 500$, Ebayesthresh is used to perform thresholding.

temporal denoising by smoothing the wavelet coefficients. The first column in the table shows the ordinary estimation efficiency (the results for function g^1 is similar to table 5.1 under ebayesthresh category). The second column in the table shows the improved results by smoothing the coefficients in wavelet domain. The performance of this method can be improved by smoothing the wavelet coefficients by averaging more than 3 coefficients. This is a future task.

5.13.2 Via sequential transform using spatio-temporal network

A larger network can be formed by using the idea that node k becomes the neighbour of itself in the past and the future time instances. This idea is discussed in section 3.3.1. We form the observations over time into blocks with Mn number of data, where n is the number of nodes, which is the same size as the network

formed by coalescing M time slots. We form the block of observations as

$$f^j_{k'} = f_{t,k}, \qquad t = (j-1)M_t + 1, \ldots, jM_t, k = 1, \ldots, n, \tag{5.35}$$

where j indicate the block index. Thus $|f^j_{k'}| = Mn$ and $k' = 1, \ldots, Mn$. Therefore our model for this spatio-temporal method is, $f^j_{k'} = g^j_{k'} + \epsilon^j_{k'}$, and after the LOCAAT transform, the wavelet domain representation is $d^j_{k'} = \theta^j_{k'} + \acute{\epsilon}^j_{k'}$ where, $\acute{\epsilon}^j_{k'} \sim N(0, \overline{\sigma}^2_k \sigma^2)$. We perform smoothing on these blocks of coefficients similar to the spatial method smoothing, but this time smoothing by taking average of adjacent block of coefficents., i.e.,

$$\tilde{d}^j_{k'} = \begin{cases} \frac{d^{j-1}_{k'} + d^j_{k'} + d^{j+1}_{k'}}{3}, & \text{if } 1 < j < X \\ d^j_{k'}, & \text{otherwise,} \end{cases} \tag{5.36}$$

where $X = \lfloor T/M \rfloor$ if non-overlapping case is considered and $X = T - M + 1$ if overlapping case is considered. After smoothing, the wavelet domain representation of coefficients can be written as

$$\tilde{d}^j_{k'} = \tilde{\theta}^j_{k'} + \tilde{\epsilon}^j_{k'}, \tag{5.37}$$

where $\tilde{\theta}^j_{k'} = \frac{\theta^{j-1}_{k'} + \theta^j_{k'} + \theta^{j+1}_{k'}}{3} \approx \theta^j_{k'}$ and $\tilde{\epsilon}^j_{k'} = \frac{\acute{\epsilon}^{j-1}_{k'} + \acute{\epsilon}^j_{k'} + \acute{\epsilon}^{j+1}_{k'}}{3}$. Note that we assume $\tilde{\theta}^j_{k'} \approx \theta^j_{k'}$ because the signal is sparse in wavelet domain and therefore it is distinguished from the noise present in the signal. Also the signal has temporal correlation and therefore $\theta^{j-1}_{k'} \approx \theta^j_{k'}$. The noise term, $\tilde{\epsilon}^j_{k'} \sim N(0, \overline{\sigma}^2_k \tilde{\sigma}^2)$, gets smoothed by the operation above therefore it is now $\tilde{\sigma} < \sigma$ and this and can be estimated through variogram or MAD technique (see section 4.3 and section 4.2). We denote it by $\hat{\tilde{\sigma}}$.

$$\frac{\tilde{d}^j_{k'}}{\overline{\sigma}_{k'} \hat{\tilde{\sigma}}} = \frac{\tilde{\theta}^j_{k'} + \tilde{\epsilon}^j_{k'}}{\overline{\sigma}_{k'} \hat{\tilde{\sigma}}}, \tag{5.38}$$

which we write as,

$$d_{k'}^{j*} = \theta_{k'}^{j*} + \epsilon_{k'}^{j*}, \tag{5.39}$$

where $d_{k'}^{j*} = \frac{\tilde{d}_{k'}^{j}}{\overline{\sigma}_{k'}\hat{\tilde{\sigma}}}$, $\theta_{k'}^{j*} = \frac{\tilde{\theta}_{k'}^{j}}{\overline{\sigma}_{k'}\hat{\tilde{\sigma}}}$ and $\epsilon_{k'}^{j*} = \frac{\tilde{\epsilon}_{k'}^{j}}{\overline{\sigma}_{k'}\hat{\tilde{\sigma}}}$. Now the error term, $\epsilon_{k'}^{j*} \overset{\text{approx}}{\sim} N(0,1)$. We can perform the thresholding operation on $d_{k'}^{j*}$ to estimate $\hat{\theta}_{k'}^{j*}$. Correcting for the variance gives $\hat{\theta}_{k'}^{j} = \overline{\sigma}_{k'}\hat{\theta}_{k'}^{j*}$. Now we estimate the $\hat{g}_{k'}^{j}$ by performing inverse transform on $\hat{\theta}_{k'}^{j}$.

In the non-overlapping case,

$$\hat{g}_{t,k} = \hat{g}_{k'}^{j}, \tag{5.40}$$

with $t = (j-1)M_t + 1 + \lfloor k'/n \rfloor$ and $k = k' - [t - (M_t(j-1)+1)]n$.

Overlapping case,

$$\hat{g}_{t,k} := \hat{g}_{t,k} + \hat{g}_{k'}^{j}, \tag{5.41}$$

$t = j + \lfloor k'/n \rfloor$ and $k = k' - (t-j)n$.

$$\hat{g}_{t,k} = \begin{cases} \frac{\hat{g}_{t,k}}{t}, & \text{if } t < M_t \\ \frac{\hat{g}_{t,k}}{M_t}, & \text{if } M_t \le t \le T - M_t + 1 \\ \frac{\hat{g}_{t,k}}{T-t+1}, & \text{otherwise.} \end{cases} \tag{5.42}$$

Table 5.7 and table 5.8 shows the performance improvement by smoothing co-

Signals	No smoothing	With smoothing
g^1	2.146	2.629
g^2	1.637	2.067
maartenfunc	1.809	2.490
2D Doppler	1.488	2.151

Table 5.7: Efficiencies compared for spatio-temporal network (extended from type-1 network) with non-overlapping window case. The first column shows estimation efficiency in an ordinary estimation (no coefficient smoothing involved) and the second column shows estimation efficiency by smoothing coefficients in the wavelet domain. SNR=2 , $n = 500$, Ebayesthresh is used to perform thresholding.

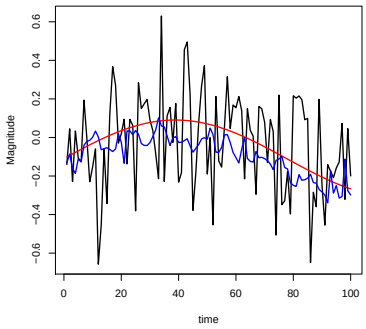
efficients for the spatio-temporal network with non-overlapping and overlapping

Signals	No smoothing	With smoothing
g^1	2.051	2.601
g^2	1.693	2.296
maartenfunc	2.004	2.502
2D Doppler	1.579	2.329

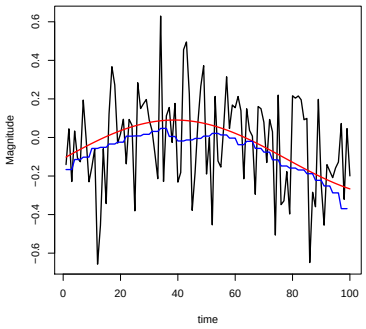
Table 5.8: Efficiencies compared for spatio-temporal network (extended from type-1 network) with overlapping window case. The first column shows estimation efficiency in an ordinary estimation (no coefficient smoothing involved) and the second column shows estimation efficiency by smoothing coefficients in the wavelet domain. SNR=2 , $n = 500$, Ebayesthresh is used to perform thresholding.

methods respectively. The second column in the table shows the estimation efficiency using ordinary spatio-temporal based estimation (the results for function g^1 are similar to table 5.2 under the ebayesthresh category), and the third column shows the improved results by smoothing the coefficients in wavelet domain. The performance of this method can be improved by smoothing the wavelet coefficients by averaging more than 3 coefficients. Selecting the correct number for good performance is a future task.

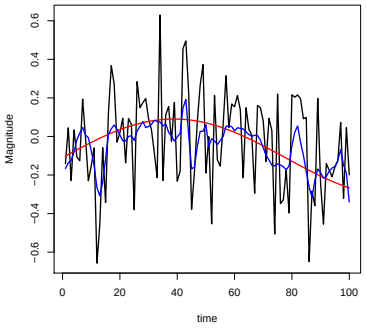
By comparing tables 5.6, 5.7 and 5.8, we can conclude that spatio-temporal network overlapping window based approach performs better than other methods however the others also perform to an acceptable level. Other methods may be preferred if computational efficiency is required. Figure 5.15 shows a plot of time variation of node 10 (random choice for illustration) with different methods to illustrate how each method performs when coefficient smoothing is applied in wavelet domain. Figure 5.16 shows a plot of estimated function at $t = 6$ (a random choice) using different methods to show how each methods estimate the function g.



(a) Spatial

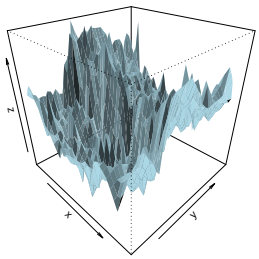


(b) Spatio-temporal non-overlapping window

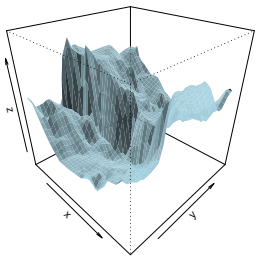


(c) Spatio-temporal overlapping window

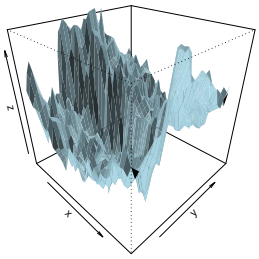
Figure 5.15: Denoising along time for node 10 (a random choice) with coefficient smoothing in wavelet domain. SNR=2, $n = 500$, ebayesthresh used to perform thresholding. Original series is shown in red, noisy series in black and the estimated series in blue.



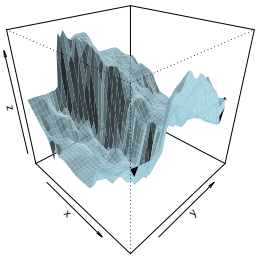
(a) Noisyg^2



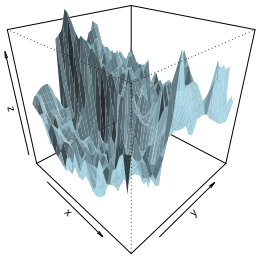
(b) Denoised g^2 - Spatial case with coefficient smoothing



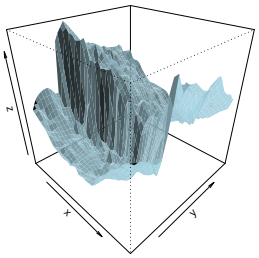
(c) Noisyg^2



(d) Denoised g^2 - Spatio-temporal non overlapping, with coefficient smoothing



(e) Noisyg^2



(f) Denoised g^2 - Spatio-temporal overlapping, with coefficient smoothing

Figure 5.16: Denoising spatial function via different methods using coefficients smoothing in wavelet domain. SNR=2, $n = 500$, ebayesthresh used to perform thresholding.

5.14 Sparsity

One main advantage of wavelet methods is the sparsity of signal in wavelet domain, i.e. the signal is represented by few coefficients. In this section we explore the sparsity of the LOCAAT transform with two types of networks, type-1 and type-2 networks for two test functions g^1 and g^2. We define the Mean Integrated Squared Error (MISE) between the original signal $\mathbf{g}$ and the estimate $\hat{\mathbf{g}}$ as follows,

$$\text{MISE}(\mathbf{g}, \hat{\mathbf{g}}) = E\frac{1}{n}\sum_{k=1}^{n}(g_k - \hat{g}_k)^2, \tag{5.43}$$

where n is the number of nodes.

We find the complete LOCAAT coefficients of the noisy data and arrange them in ascending order of magnitude in absolute values. We find the MISE by setting the smallest coefficient to zero. Then we proceed with setting two smallest coefficients to zero and finding the MISE, and so until all the coefficients are set to zero except the scaling function coefficients [69]. We find the MISE(i) as follows,

$$\text{MISE}(i) = E\frac{1}{n}\sum_{k=1}^{n}(g_k - \hat{g}_k^i)^2, \tag{5.44}$$

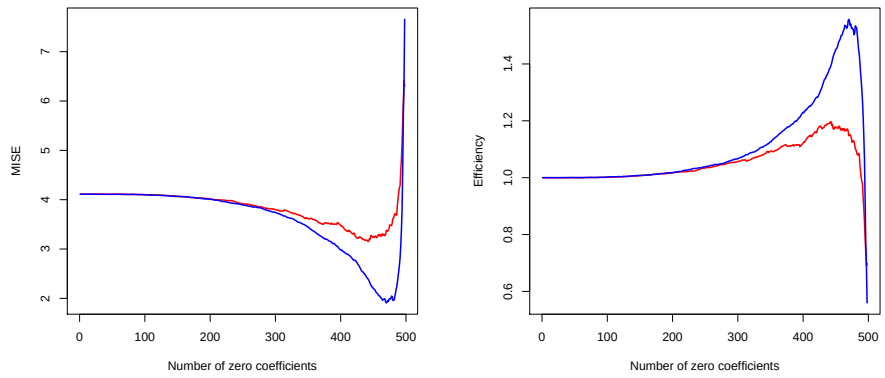
where $\hat{g}_k^i$ is the estimate for true function value g_k at node k by setting i smallest coefficients to zero. We also calculate the efficiency,

$$\text{eff}(i) = E\log_{10}\left(\frac{\text{var}(g)}{\text{var}(\hat{g}^i - g)}\right) / \log_{10}\left(\frac{\text{var}(g)}{\text{var}(f - g)}\right), \tag{5.45}$$

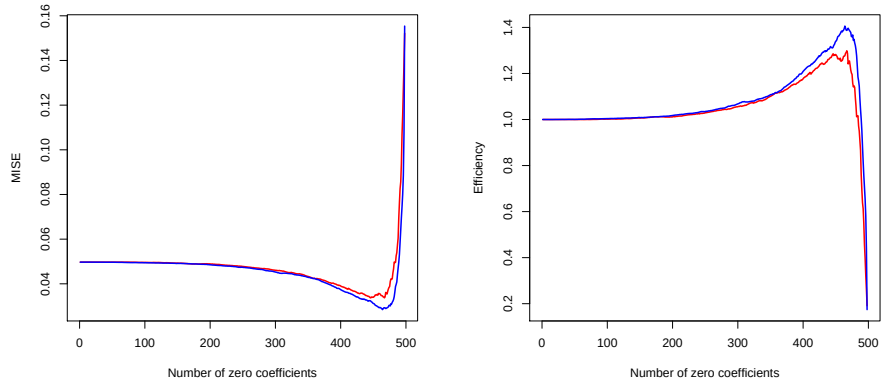
where g is the original function (spatial function), $\hat{g}_i$ is the inverted transform by setting i smallest coefficients to zero and f if the noisy observation. Figure 5.17 shows the results of sparsity for our two test networks with two test functions,g^1 and g^2. We can see from the figure that type-2 network has slightly weaker sparsity as the energy is spread across more neighbours causing the same threshold as type-1 network to perform less efficiently. This is visible in the efficiency plot (right

hand side plots in figure 5.17). Note that this way maximum efficiency achieved is 1.68 for g^1 function under type-1 network. This is also the maximum achievable by hard thresholding.

Work in [31, 73] shows another way of measuring the sparsity,



(a) g^1 (b) g^1



(c) g^2 (d) g^2

Figure 5.17: Sparsity and Efficiency plots against the number of zero coefficients for two test functions, g^1 and g^2, SNR=2, n=500. Type-1 network results in blue line and type-2 network results in red. The results are average of 50 independent experiments.

$$S(\%) = 100 - \frac{N - \#0}{N} \times 100, \tag{5.46}$$

where N is the number of coefficients and $\#0$ is the number of coefficients that were replaced by zero *by the threshold*. In a K-scale decomposition, total sparsity

is defined as,

$$\begin{aligned} S(\%) &= \frac{1}{K}\sum_{i=1}^{K} S_i \\ &= \frac{1}{K}\sum_{i=1}^{K}\left\{100 - \left[\left(\frac{N_i - \#0_i}{N_i}\right)\times 100\right]\right\}, \end{aligned} \tag{5.47}$$

where N_i is the number of coefficients in the i^{th} scale and $\#0_i$ is the number of coefficients replaced by zero by the threshold operation in the i^{th} scale. Since the LOCAAT algorithm finds one coefficient per scale, we calculate the sparsity as follows. First we calculate 1 LOCAAT coefficient, threshold (ebayesthresh) that coefficient, then calculate the sparsity and efficiency. Then we calculate 2 LOCAAT coefficients, threshold these coefficients, then calculate sparsity and efficiency and so on until $i = n - 2$, where n is the number of nodes. In i^{th} stage, we calculate the sparsity as,

$$S(i) = 100 - \left[\left(\frac{i - \#0_i}{i}\right)\times 100\right], \tag{5.48}$$

where i is the number of detail coefficients, $\#0_i$ is the number of coefficients replaced by the threshold operation. We also calculate the efficiency as,

$$\text{eff}(i) = E\log_{10}\left(\frac{\text{var}(g)}{\text{var}(\hat{g}^i - g)}\right) / \log_{10}\left(\frac{\text{var}(g)}{\text{var}(f - g)}\right), \tag{5.49}$$

where i is the number of detail coefficients, g is the true function (spatial), $\hat{g}^i$ is the estimator of g from i number of LOCAAT coefficients and f is the observation. Figure 5.18 shows the plot of sparsity on the left and efficiency on the right. The efficiency plot has an interesting outcome. It reveals that the maximum efficiency is attained when we transform 80-90% of data as detail coefficents and leave the remaining as scaling coefficients. This implies that we can perform the threshold operation on the 80-90% of the detail coefficients on a fully transformed

coefficients (in the order they were calculated and the order is important). This is visible from further experiments shown in figure 5.19. We fix the threshold (e.g. ebayes threshold) and perform LOCAAT transform such that we obtain 80-100% of data as the detail coefficients, threshold the detail coefficents and invert the transform. We repeat this for several independent experiments and find the MISE.

Figure 5.18 also tells us that the signal transformed using type-1 network has better sparsity than performing LOCAAT transform using type-2 network for the same signal. This is because the detail coefficient calculations depend on spatial closeness of neighbours. Type-1 network is based on MST technique. Therefore, the neighbours are much closer than in the the type-2 network.

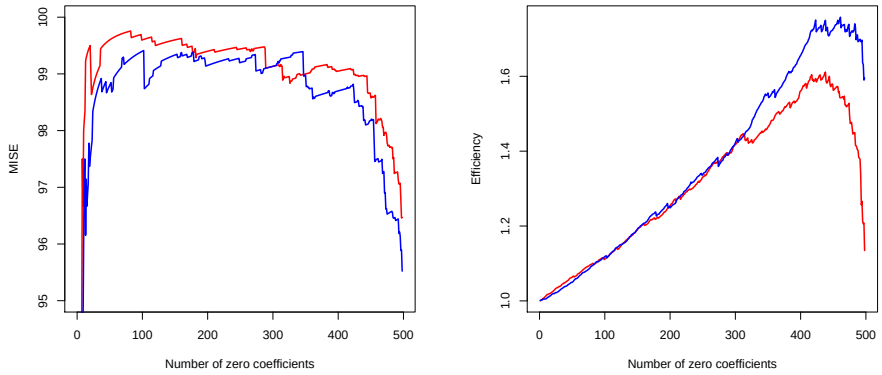
5.15 Stopping time for LOCAAT transform

Section 5.8 and the results from the second part of the last section 5.14 raise the question about the optimal stopping time for the LOCAAT algorithm. All the results presented so far in the thesis are produced by finding $L = n-2$ coefficients. In this section we investigate when to stop the LOCAAT transform.

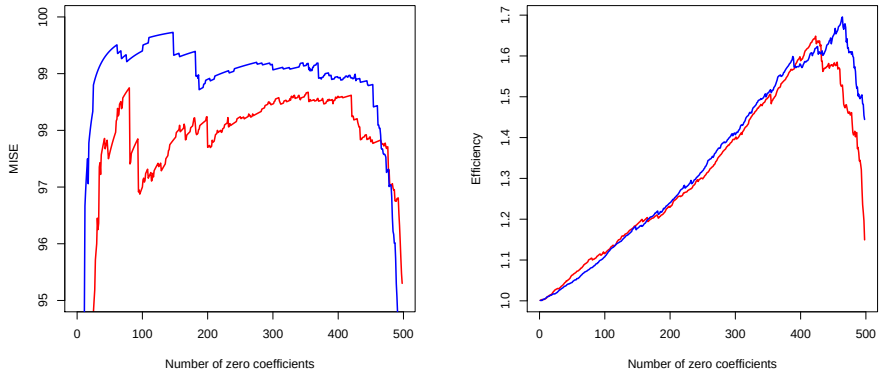
From figure 5.18 and figure 5.19 we see that for an ebayesthresh method, the optimal stopping time lies between 80-90% of transformation. Some of the other thresholding methods presented at the beginning of this chapter also rely on another parameter, p, the percentage of universal threshold. Let us denote the optimal stopping time as q. This then leads to a 2-D optimisation problem. The function in 'R' called `optim` will do this for us. See section 2.8 for detail on optimisation.

We define the following hard threshold operation for the detail coefficients $\mathbf{d}^*$ (see (4.4))

$$\hat{\boldsymbol{\theta}}^*_{p,q} = \mathbf{d}^*_q \mathbb{I}(\mathbf{d}^*_q > \lambda_p), \tag{5.50}$$



(a) g^1 (b) g^1

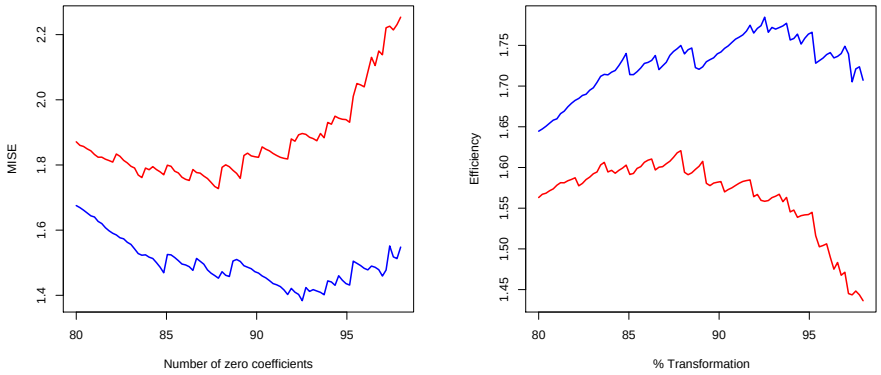


(c) g^2 (d) g^2

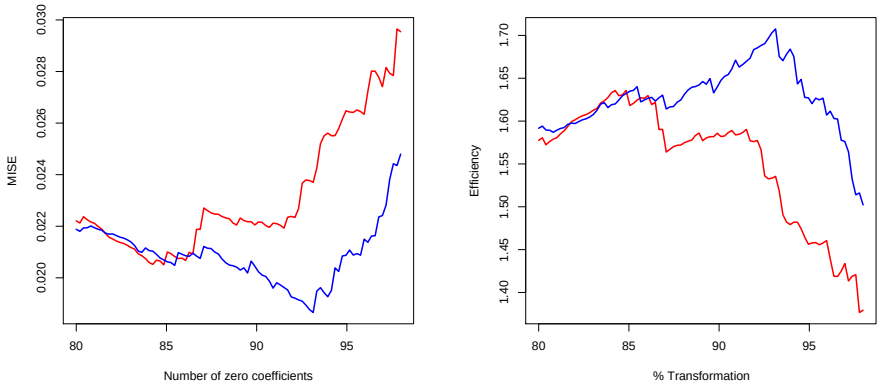
Figure 5.18: Sparsity and Efficiency plots against the number of zero coefficients for two test functions, g^1 and g^2, n=500. Type-1 network results in blue line and type-2 network results in red. The results are average of 50 independent experiments.

where $\lambda_p = p\sigma\sqrt{2\log L_q}$, $\mathbf{d}_q^*$ is the $q\%$ of earliest detail coefficients, $\hat{\boldsymbol{\theta}}_{p,q}^*$ is an estimator for the true detail coefficients $\boldsymbol{\theta}^*$. Note that $L_q = |\mathbf{d}_q^*|$ is the number of detail coefficients. Similarly we can define an estimator for $\boldsymbol{\theta}^*$ for each thresholding methods. Once we found the estimator for the true detail coefficients we can perform the inverse LOCAAT transform to find an estimator for the true function $\hat{\mathbf{g}}(p,q)$. We then define the following MISE function on which the optimisation is performed,

$$\text{MISE}(p,q) = E\frac{1}{n}\sum_{k=1}^{n}(\hat{g}_k(p,q) - g_k)^2, \tag{5.51}$$



(a) g^1 (b) g^1



(c) g^2 (d) g^2

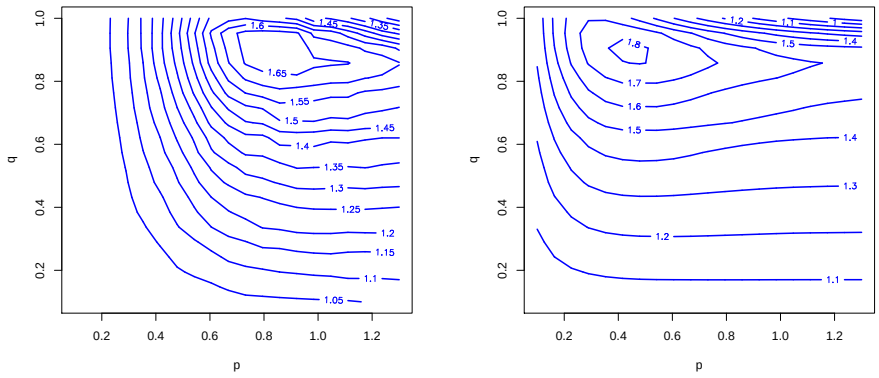
Figure 5.19: Sparsity and Efficiency plots against the % transform coefficients for two test functions, g^1 and g^2, n=500. Type-1 network results in blue line and type-2 network results in red. The results are average of 50 independent experiments.

where n is the number of nodes, p is the percentage of universal threshold, q is the percentage of detail coefficients that are thresholded and $\hat{g}_k(p,q)$ is the estimator of true function g_k. We denote $p = p^*$ and $q = q^*$ as the minimisers of MSE(p,q) Table 5.9 shows the results for various thresholding methods. It can be noted that p^* does not change (compare with table 5.1) however by finding q^*, the efficiency is improved (compare with table 5.1). Figure 5.20 shows contour plot of efficiency as a function of both p and q. From the table 5.9 and the figure 5.20, we can come to the conclusion that processing only 90% of earliest detail coefficient will

provide the best results regardless of threshold method.

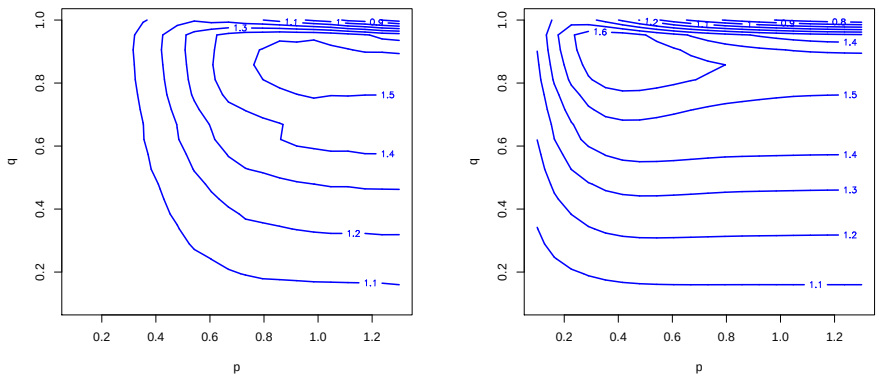
Threshold methods	p^*	q^*	maximum efficiency
Hard	0.87	0.85	1.71
Soft	0.43	0.91	1.86
Ebayesthresh	NA	0.92	1.82

Table 5.9: Optimised values for p and q for a type 1 network (spatial network) for various threshold methods. SNR=2, n=500, function g^1 used. Each entry is an average of 100 independent simulations.



(a) Hard threshold, function g^1

(b) Soft threshold, function g^1



(c) Hard threshold, function g^2

(d) Soft threshold, function g^2

Figure 5.20: Contour plot of efficiency as a function of p and q. Test functions g^1 and g^2 are used on T-1 network, SNR=2, $n = 500$. The results are averaged over 50 independent experiments.

5.16 Conclusions

We have introduced a number of thresholding methods for the LOCAAT method in this chapter. We have seen that universal-type thresholding introduces bias into the estimation and the level of bias introduced depends on the type of thresholding applied. We have also looked at compensating for bias by correcting the mean shift. We have also looked at some risk estimators and minimised these risk estimators to get a good estimate for the true function g. We then looked at further improving estimation by smoothing coefficients in the wavelet domain when the time series data is available. Finally we looked at the sparsity property of the LOCAAT transform using two test networks T-1 and T-2.

The HT, ST, BT, NC, BBC threshold methods combined with a suitable p^*, yield good estimators for true function with efficiency > 1. All of these methods are sensitive to the choice of p^*, however the mean correction method appeared less sensitive to the choice of p^* and it can be used with any threshold methods. This mean correction method reduces the estimation bias at the expense of increased variance (therefore estimation efficiency will be less than that of the other methods). When time series data are available, spatio-temporal overlapping window based estimation yields the best efficiency compared to the spatial and spatio-temporal non-overlapping window based estimations.

We obtained significant performance improvements by identifying the sensitive coefficients and preventing them from being thresholded by the threshold operation. The computing cost for this method is very high as it involves having to perform inverse transforms to calculate the error by not thresholding a particular coefficient (there will be several inverse transforms). The risk estimator based on OCV is also very slow as there will be several inverse transforms involved. Therefore we introduced a MOCV which reduces the computing cost however it requires some inverse transforms to calculate the cross validation scores (risk value). The SURE and GCV do not require inverse transforms to calculate the risk value because

the risk value is directly calculated from the wavelet coefficients. All these risk estimators give a good value for λ^*, which minimises the MSE function. We cross-checked this with the optimised soft threshold's λ which is $p^*\sigma\sqrt{2\log L} \approx \lambda^*$.

When time series data are available, we have already seen that the spatio-temporal overlapping window based estimation method yields the best efficiency. The spatial, and both the spatio-temporal methods improved the results significantly by smoothing the coefficients in wavelet domain. The performance of these methods rely on smoothing the wavelet coefficients (by averaging coefficients from the past and future). We considered one past present and one future coefficient (therefore 3 in total) to smooth the present coefficient. The results can be improved by considering more past and future coefficients (e.g. 2 past, 1 present, 2 future so 5 in total, and so on). Further investigation is left to the future work.

Finally, we looked at sparsity of the LOCAAT transform which led to the conclusion that performance can be improved by not processing all the detail coefficients. For ebayesthresh method, processing about 90% of earliest detail coefficients improved the results significantly compared to processing 100% of detail coefficients. Similarly we have found that hard thresholding require 85% and soft threshold require 90%.

Chapter 6

Network Forecasting

6.1 Introduction

A time series is a collection of observations made sequentially through time. Time series data arise in many areas, and it is interesting and challenging to model these data. In this chapter, we present some time series modelling of network data with the aid of a simulation study and a real data example. The initial part of the chapter develops the methodology to understand the modelling in the latter part. Our main goal in this chapter is to forecast network data both in the time domain and the wavelet domain, and to compare their prediction accuracy. There is some literature on wavelet domain forecasting of time series data [5, 35, 72, 90], however there seems to be no literature for network forecasting of time series. Network behaviour is complex to model and therefore it is challenging to forecast. Since there are potentially many nodes and each possesses a time series, it is cumbersome to model each process in the conventional way. Therefore we are more or less forced to use a some sort of automated modelling [14]. For our work, by way of exemplar we choose to use a simple ARIMA(p_k,1,0) or AR(p_k) model or a Simple Exponential Smoothing (SES) for a particular node k in our simulation study. However other complex models can also be explored with our proposed

network forecasting methods. Here we concentrate on the network aspects rather than the time series details.

It will be useful if we could predict the network behaviour in advance. We have demonstrated the importance of network forecasting in this chapter using a real data study (mumps data). There are many other applications where predicting a network behaviour is useful. As another example, predicting the future network state of a communication network would enable the optimisation of the network performance.

We now review some basic theory of stationary processes.

6.2 Basic theory of stationary processes

In this section we introduce some basic time series models for stationary processes. We also introduce some forecasting techniques for stationary processes. A stationary process is loosely defined as follows. A time series is said to be stationary if there is no systematic change in mean (no trend), if there is no systematic change in variance and if strictly periodic variations have been removed [14]. In other words, the properties of one section of data is much like those of any other sections. The following descriptions closely follow [14].

6.2.1 Stationary Process

A time series is said to be strictly stationary if the joint distribution of $X(t_1), \ldots, X(t_k)$ is the same as the joint distribution of $X(t_1+\tau), \ldots, X(t_k+\tau)$ for all $t_1, \ldots, t_k, \tau$. In other words, shifting the time origin by τ has no effect on the joint distribution. Therefore for a strictly stationary process, its mean and variance do not change with t. For $k = 1$,

$$\begin{aligned} \mu_t &= \mu \\ \sigma_t^2 &= \sigma^2, \end{aligned} \tag{6.1}$$

and for $k = 2$, the joint distribution of $X(t_1)$ and $X(t_2)$ depends only on the lag, $\tau = t_2 - t_1$. Hence the autocovariance function, $C(\tau)$, can be written as follows,

$$\begin{aligned} C(\tau) &= E\{[X(t) - \mu][X(t+\tau) - \mu]\}, \\ &= \text{cov}[X(t), X(t+\tau)]. \end{aligned} \tag{6.2}$$

In practice, it is often useful to describe stationarity in a less restrictive way than that described above. A process is called second order stationary (or weakly stationary) if its mean is constant and its autocovariance function depends only on the lag τ, i.e.

$$\begin{aligned} E[X(t)] &= \mu, \\ \text{cov}[X(t), X(t+\tau)] &= \gamma(\tau). \end{aligned} \tag{6.3}$$

6.2.2 Purely random process

A discrete-time process is called a purely random process if it consists of a sequence of random variables, Z_t, which are mutually independent and identically distributed. We usually assume that the random variables are normally distributed with mean zero and variance σ^2. From the definition, the autocovariance function of purely random process is,

$$C(\tau) = \text{cov}(Z_t, Z_{t+\tau}) = \begin{cases} \sigma^2, & \text{if } \tau = 0, \\ 0, & \text{if } \tau \neq 0. \end{cases} \tag{6.4}$$

6.2.3 Moving average process

Suppose that Z_t is a purely random process with mean zero and variance σ^2. Then a process X_t is said to be a moving average process of order q, MA(q), if,

$$\begin{aligned} X_t &= \beta_0 Z_t + \beta_1 Z_{t-1} + \ldots + \beta_q Z_{t-q} \\ &= \sum_{i=0}^{q} \beta_i Z_{t-i}, \end{aligned} \tag{6.5}$$

where $\{\beta_i\}$ are constants [14]. Using a backward shift operator, the MA(q) process can be written as,

$$X_t = (\beta_0 + \beta_1 B + \ldots + \beta_q B^q) Z_t. \tag{6.6}$$

Note that no restrictions are required on $\{\beta_i\}$ for a MA process to be stationary, however restrictions are placed on $\{\beta_i\}$ so that the process is invertible. For a MA(q) process to be invertible, the roots of the equation

$$\theta(B) = \beta_0 + \beta_1 B + \ldots + \beta_q B^q = 0 \tag{6.7}$$

must lie outside the unit circle (B is considered as a complex variable rather than an operator).

6.2.4 Autoregressive (AR) process

The autoregressive model of order p, AR(p), is defined by,

$$\begin{aligned} X_t &= \alpha_1 X_{t-1} + \alpha_2 X_{t-2} + \ldots + \alpha_p X_{t-p} + Z_t \\ &= \sum_{i=1}^{p} \alpha_i X_{t-i} + Z_t, \end{aligned} \tag{6.8}$$

where Z_t are purely random process with mean zero and constant variance σ^2 and $\alpha_1, ..., \alpha_p$ are the parameters of the model. As a simple model, the AR(1) process can be written as follows,

$$X_t = \alpha X_{t-1} + Z_t. \tag{6.9}$$

By successive substitutions, we may write the AR(1) process as the following infinite order Moving Average (MA) process,

$$X_t = Z_t + \alpha Z_{t-1} + \alpha^2 Z_{t-2} + \ldots, \tag{6.10}$$

provided $|\alpha| < 1$ for the stationarity. Because of this duality between AR and MA processes, it may be convenient to use the backward shift operator, B. Thus (6.9) can be written as follows using the backward shift operator,

$$(1 - \alpha B)X_t = Z_t. \tag{6.11}$$

For stationarity $|\alpha| < 1$. For the general AR(p) processes, conditions can be placed on $\{\alpha_1, \ldots, \alpha_p\}$ to make the process stationary. An AR(p) process can be expressed as follows using the backward shift operator,

$$(1 - \alpha_1 B - \ldots - \alpha_p B^p)X_t = Z_t. \tag{6.12}$$

For stationarity of AR(p) process, the roots of the equation

$$\phi(B) = 1 - \alpha_1 B - \ldots - \alpha_p B^p = 0 \tag{6.13}$$

must lie outside the unit circle (where B is considered as a complex variable rather than as an operator, so that the roots, which may be complex, are greater than one in modulus [14]). Following the above rule, for an AR(2) process, the following three conditions must hold,

$$\alpha_1 + \alpha_2 < 1; \alpha_1 - \alpha_2 > -1; \alpha_2 > -1. \tag{6.14}$$

The above three conditions define a stationarity triangle. The AR(2) process will be stationary provided that α_1 and α_2 lie within this triangle.

6.2.5 Mixed models: Autoregressive Moving Average (ARMA)

Another useful model in time series modelling is to combine an AR(p) process and MA(q) process. It is denoted as ARMA(p,q) and is given as follows,

$$X_t = \alpha_1 X_{t-1} + \ldots + \alpha_p X_{t-p} + Z_t + \beta_1 Z_{t-1} + \ldots + \beta_q Z_{t-q}. \tag{6.15}$$

Using the backward shift operator B, ARMA(p,q) can be written as,

$$\phi(B)X_t = \theta(B)Z_t, \tag{6.16}$$

where $\phi(B) = 1 - \alpha_1 B - \ldots - \alpha_p B^p$ is a polynomial of order p and $\theta(B) = 1 + \beta_1 B + \ldots + \beta_q B^q$ is a polynomial of order q. For an ARMA process to stationary, the roots of $\phi(B) = 0$ must lie outside the unit circle. No conditions required on $\{\beta_i\}$ for the ARMA(p,q) process to be stationary, however the roots of $\theta(B) = 0$ must lie outside the unit circle for the process to be invertible [14].

6.2.6 Integrated models: Autoregressive Integrated Moving Average (ARIMA)

Many real time series are non-stationary. One approach for fitting a stationary model is to remove the non-stationarity first. If the time series is non-stationary in the mean it is common practice to difference the time series [14]. The difference operator ∇ is defined as follows, suppose $\{x_t\}$ is a time series then first order differencing of lag 1 is given by,

$$w_t = x_{t+1} - x_t = \nabla x_{t+1}. \tag{6.17}$$

Occasionally, second order differencing is required. Second order differencing of lag 1 is achieved by ∇^2 operator,

$$\begin{aligned} \nabla^2 x_{t+2} &= \nabla(\nabla x_{t+2}) \\ &= x_{t+2} - 2x_{t+1} + x_t. \end{aligned} \tag{6.18}$$

Generally the d^{th} order differencing of process X_t can be written as,

$$W_t = \nabla^d X_t. \tag{6.19}$$

Certain kinds of non-stationary time series can be modelled as ARIMA(p, d, q). This basically means ARMA modelling of W_t.

$$W_t = \alpha_1 W_{t-1} + \ldots + \alpha_p W_{t-p} + Z_t + \beta_1 Z_{t-1} + \ldots + \beta_q Z_{t-q}. \tag{6.20}$$

6.3 Forecasting techniques

In this section we introduce two simple forecasting techniques. Suppose we have a time series $f_{1,k}, \ldots, f_{T,k}$ for a node k. Typically, the forecast for $f_{T+h,k}$ made at time T for h steps ahead, is denoted by $\hat{f}_{T,k}(h)$. Let $e_{T+h,k}$ denote the estimation error resulting from the forecast for $f_{T+h,k}$,

$$e_{T+h,k} = f_{T+h,k} - \hat{f}_{T,k}(h). \tag{6.21}$$

The Mean squared prediction error (MSPE) of a forecast $\hat{f}_{T,k}(h)$ is given by,

$$\text{MSPE}(\hat{f}_{T,k}(h), f_{T+h,k}) = E((\hat{f}_{T,k}(h) - f_{T+h,k})^2). \tag{6.22}$$

6.3.1 Box-Jenkins forecasting

The first step in the Box-Jenkins procedure (this method was introduced in earlier edition of [8]) is to identify the model by some diagnostic procedures and then determine the parameters p, d, q of the ARIMA model (see [14]). When a satisfactory model is found, forecasts may readily be computed using the model equation by setting future values of Z to zero (this can be proved by minimising mean squared forecasting error). As an example, we consider a simple 1-step ahead forecast for an AR(p_k) model,

$$\hat{f}_{T,k}(1) = \sum_{i=1}^{p_k} \alpha_i f_{T+1-i,k}, \tag{6.23}$$

assuming the future values of error term, which normally present in the AR(p) model, are zero.

6.3.2 Simple exponential smoothing (SES)

Given a non-seasonal time series with no trend, $f_1, \ldots, f_T$, a one step ahead forecast can be given by the weighted sum of the past observations.

$$\hat{f}_T(1) = c_0 f_T + c_1 f_{T-1}, \ldots \tag{6.24}$$

It is sensible to put more weight to the most recent observations. For example the geometric weight can be applied

$$c_i = \alpha(1-\alpha)^i, \qquad i = 0, 1, \ldots \tag{6.25}$$

where α is a constant in the range $0 < \alpha < 1$. With geometric weights, the forecast is

$$\hat{f}_T(1) = \alpha f_T + \alpha(1-\alpha) f_{T-1} + \alpha(1-\alpha)^2 f_{T-2} + \ldots \tag{6.26}$$

Since the real data are finite in number, the above equation can be written as,

$$\begin{aligned}\hat{f}_T(1) &= \alpha f_T + (1-\alpha)[\alpha f_{T-1} + \alpha(1-\alpha)f_{T-2} + \ldots] \\ &= \alpha f_T + (1-\alpha)\hat{f}_{T-1}(1).\end{aligned} \tag{6.27}$$

This is in a nice recurring form which suggests that the 1-step-ahead forecast at any given time depends on the current observation and the previous forecast. We can estimate α based on the available time series. For a given particular value of α, perform 1-step ahead forecast iteratively,

$$\begin{aligned}\hat{f}_1(1) &= f_1 \\ e_2 &= f_2 - \hat{f}_1(1) \\ \hat{f}_2(1) &= \alpha f_2 + (1-\alpha)\hat{f}_1(1) \\ e_3 &= f_3 - \hat{f}_2(1) \\ &\vdots \\ e_T &= f_T - \hat{f}_{T-1}(1).\end{aligned} \tag{6.28}$$

Now calculate the $\sum_{i=2}^{T} e_i^2$ and repeat these steps for various values of α. The smoothing parameter α is estimated such that the minimum $\sum_{i=2}^{T} e_i^2$ is obtained.

6.4 Simulation study

In this section, we wish discover whether it is advantageous to forecast in the wavelet domain compared to the time domain. we start by simulating some non-stationary in the mean time series (as this is the case for many real time series), fitting these series both in the time domain and wavelet domain (fitting the LOCAAT coefficient of each node as time series) and finally look at forecasting using both approaches.

6.4.1 Network simulation

Many real world time series data are non-stationary. We have decided to use the following ARIMA(1,1,0) model as it looks like the data that would have come from a communicating node (see figures 6.1(a) and 6.1(c)). We generate the time series data $f_k(t)$ for a node k as follows,

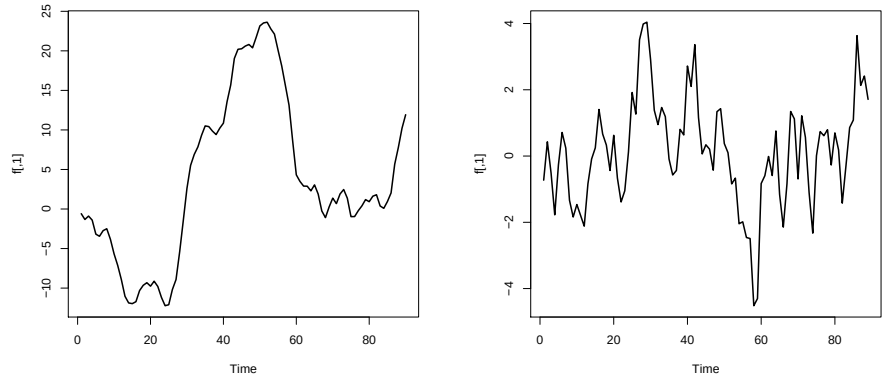
```
f[,k] <- arima.sim(list(order = c(1,1,0), ar = 0.7), n = 100).
```

This generates a series of $T = 100$ samples for each node. Hence, we obtain an AR(1) process for each node, when the time series is differenced (figure 6.1(b) and figure 6.1(d) shows an example of differenced series for node 1 and node 2). This is confirmed in figure 6.2 which shows the acf and pacf for node 1 and node 2. We generate a network of 200 nodes (placed uniformly at random on a unit square) whose interconnections are formed using MST method (as described earlier in section 3.2.2) and the time series for each node is obtained from the above code. We connect the nodes to form a network to perform network forecasting.

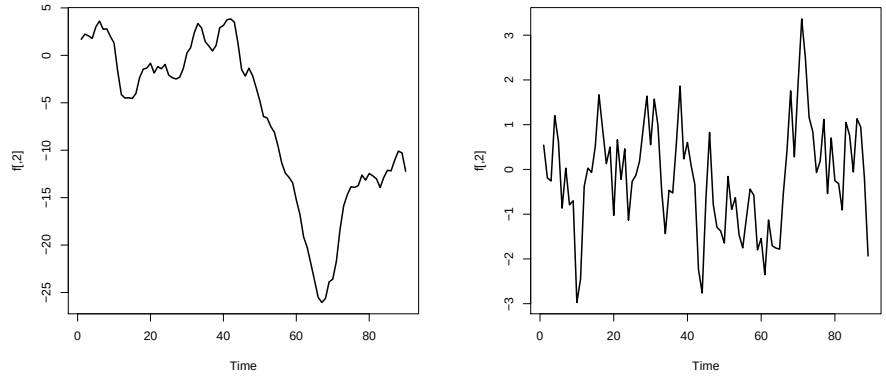
6.4.2 Time domain modelling

In this section, we model the time series for each node separately regardless of the interconnections between the nodes using some automatic approach. We have decided to choose an automatic approach since, for real world data, it would not be feasible to analyse each time series by hand and model each one of them in a conventional way. We choose ARIMA(p_k,1,0) as the model for each series. Then we can determine the p_k for each node k either through AIC or MLE methods. Note the MLE method estimation of p_k is slow for a large series. The simulated data $f_{t,k}$ is differenced and the AR(p_k) model is fitted to the differenced data. Let $u_{t,k}$ denote the differenced time series of $f_{t,k}$,

$$u_{t,k} = f_{t+1,k} - f_{t,k} = \nabla f_{t+1,k}, \qquad t = 1, \ldots, T-1. \tag{6.29}$$



(a) time series for node 1 (b) differenced time series for node 1



(c) time series for node 2 (d) differenced time series for node 2

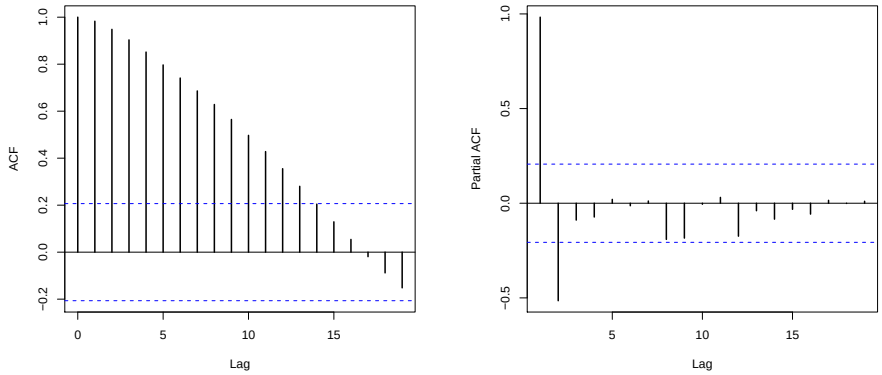
Figure 6.1: Time series data for node 1 and node 2 and the differenced time series for those nodes.

We fit an AR(p_k) process for $u_k(t)$.

$$u_{t,k} = \sum_{i=1}^{p_k} \hat{\alpha}_{ki} u_{t-i,k} + Z_{t,k}. \tag{6.30}$$

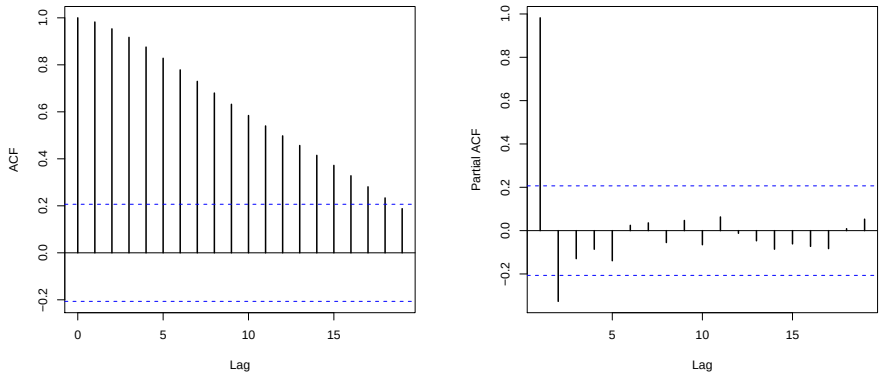
The parameters, α_i are estimated by least squares by minimising,

$$S_k = \sum_{t=p+1}^{T} [u_{t,k} - \alpha_1 u_{t-1,k} - \ldots - \alpha_p u_{t-p,k}]^2, \tag{6.31}$$



(a) autocorrelation function (acf) for node 1

(b) partial autocorrelation function (pacf) for node 1



(c) autocorrelation function (acf) for node 2

(d) partial autocorrelation function (pacf) for node 2

Figure 6.2: acf and pacf for node 1 and node 2.

with respect to $\{\alpha_i\}_{i=1}^{p}$. Once we have fitted the differenced time series, we reverse the differencing operation in order to get the actual fit $f_{t,k}^{\text{fit}}$,

$$f_{t+1,k}^{\text{fit}} = f_{t,k} + u_{t,k}^{\text{fit}}, \qquad t = p+1, \dots T-1, \tag{6.32}$$

where $u_{t,k}^{\text{fit}}$ is the fitted series for $u_{t,k}$. In R we get the fit for the data as follows,

`fit[,k] <- f[,k] - arima(f[,k],c(p_k,1,0))$residuals`.

6.4.3 Wavelet domain modelling

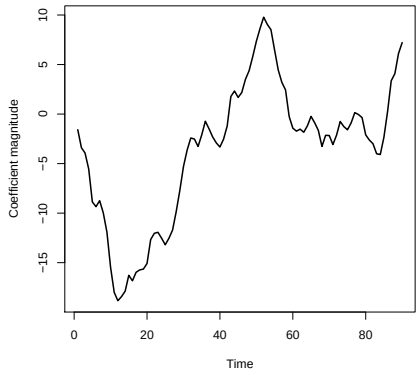
Now we will transform the time domain data into wavelet domain data using the LOCAAT transform in two different ways. First we use the LOCAAT transform by treating each time layer separately (spatial network method), and in the second method, we use the time depth $M = 3$ to develop a larger network (spatio-temporal network method, see section 3.3) and transform the data into wavelet domain. In both cases we will have a time series of coefficients for each node. Modelling these coefficients is somewhat more intricate than for the time domain modelling due to the nature of the transform. The series for each node can be qualitatively different. Some nodes we model the coefficients as ARIMA(p_k,0,0) and some nodes we model as ARIMA(p_k,1,0). Figure 6.3 shows the time series of LOCAAT coefficient for node 1 and their acf, pacf plots for both spatial and spatio-temporal network methods. We model the series of the LOCAAT coefficients for a particular node k using both models and calculate the sum of squared errors arising from both fits and choose the model that best fits the coefficients.

Let $d_{t,k}$ denote the time series of LOCAAT coefficients for a node k. First we fit an ARIMA(p_k,0,0) or simply AR(p_k) model to the coefficients $d_{t,k}$. The order p_k is determined through AIC or MLE methods and we calculate the average squared errors e_k,

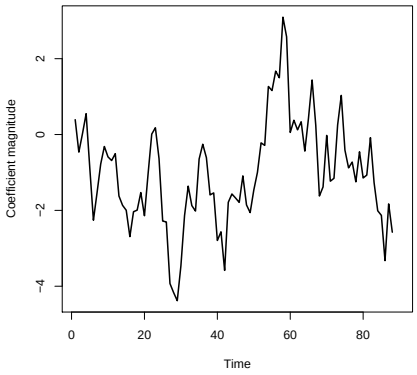
$$e_k = \frac{1}{T} \sum_{t=p_k+1}^{T} (d_{t,k} - \hat{d}_{t,k})^2, \tag{6.33}$$

where T is the number time samples and $\hat{d}_{t,k}$ is the fit for the series $d_{t,k}$. In the second model, we difference the coefficients $d_{t,k}$ to obtain a new set of coefficients $w_{t,k}$,

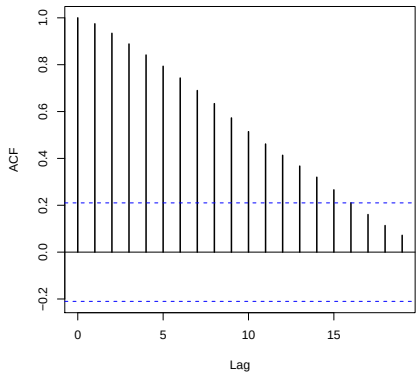
$$w_{t,k} = d_{t+1,k} - d_{t,k} = \nabla d_{t+1,k}, \qquad t = 1 \dots T-1. \tag{6.34}$$



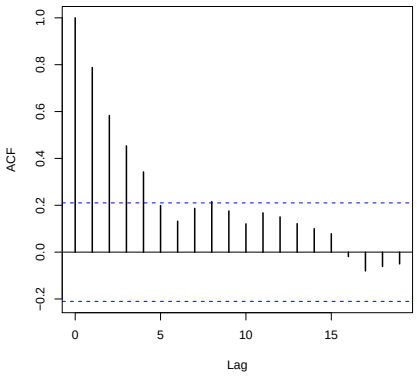
(a) Time series of coefficient(spatial network)



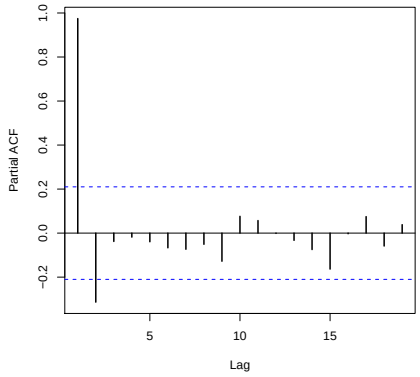
(b) Time series of coefficient(spatio-temporal network)



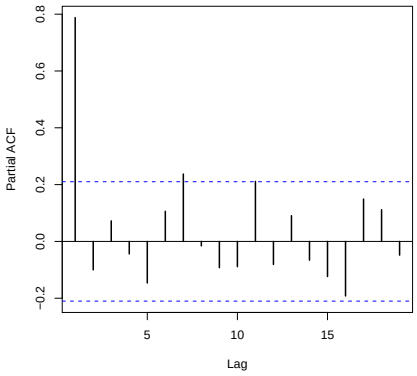
(c) acf (spatial network)



(d) acf (spatio-temporal network)



(e) pacf (spatial network)



(f) pacf (spatio-temporal network)

Figure 6.3: Time series, acf, pacf plot of LOCAAT coefficient for node 1 in both spatial network method and spatio-temporal network method.

We model these $w_{t,k}$ as AR(p_k) process and obtain the fit $\hat{w}_{t,k}$. We now reverse the difference operation to obtain the fit $\hat{d}_{t,k}$,

$$\hat{d}_{t+1,k} = \hat{w}_{t,k} + d_{t,k}, \qquad t = p+1 \ldots T-1. \tag{6.35}$$

Now we calculate the average squared error e_k^d arising from this fit,

$$e_k^d = \frac{1}{T-p-1} \sum_{t=p+2}^{T} (d_{t,k} - \hat{d}_{t,k})^2, \tag{6.36}$$

and by comparing the average of squared errors, we choose ARIMA(p,0,0) or ARIMA(p,1,0) based on whichever error term is smaller. We then take the inverse transform on these fitted LOCAAT coefficients.

Figure 6.4 shows the time series fitting for node 1 using three different approaches, i.e. time domain modelling, wavelet domain spatial method (treating each time separately) coefficient modelling and wavelet domain spatio-temporal method (building a larger network with time depth $M = 3$) coefficient modelling. It is difficult to see the fitted time series as it overlaps the data. Therefore, for clarity, we differ the fitted time series by a constant. We differ time domain fitting by $+1$, wavelet domain spatial method by -1 and the wavelet domain spatio-temporal method by $+2$.

We assess the fit by the following average squared error,

$$e_k = \frac{1}{T} \sum_{t=1}^{T} (\hat{f}_{t,k} - f_{t,k})^2, \tag{6.37}$$

where $\hat{f}_{t,k}$ is the fitted value for $f_{t,k}$. Figure 6.5 shows the average squared error of time series fitting for each node by using time domain, wavelet domain (spatial network) and wavelet domain spatio-temporal methods. The mean squared error, $\frac{1}{n}\sum_{k=1}^{n} e_k$, for time domain modelling is 0.966, wavelet domain (spatial case) is 0.979 and spatio-temporal case is 0.604. This shows that wavelet domain spatio-

temporal case fits the data well. Note that these experiments are not repeated many times as we average across many nodes which gives the repeatability and precision.

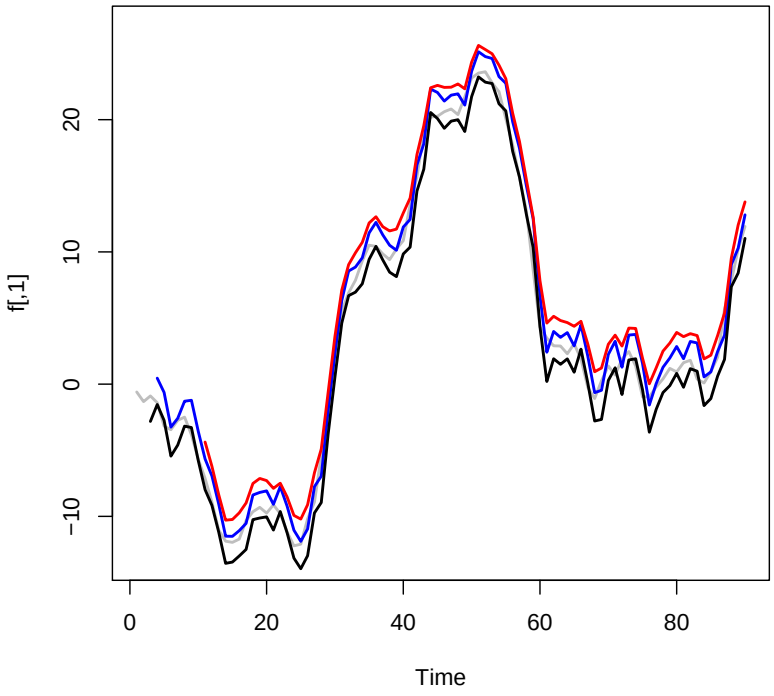


Figure 6.4: Time series fit for node 1 using time domain fitting and wavelet domain coefficient modelling methods. The data is shown in gray, time domain fit is shown in black, wavelet domain spatial method is shown in blue and wavelet domain spatio-temporal method is shown in red. The fitted time series (from all three methods) is differenced in the vertical axis for visual clarity.

6.4.4 Box-Jenkins forecasting

Forecasting the future values of an observed time series is an interesting problem in many areas such as economics, engineering and many others. In the last section we considered time series modelling of observed data $f_{t,k}$. Suppose we have an observed time series for a node k $f_{1,k} \dots f_{T,k}$. We are interested in estimating future value such as $f_{T+h,k}$, where the integer h is the lead time or forecasting

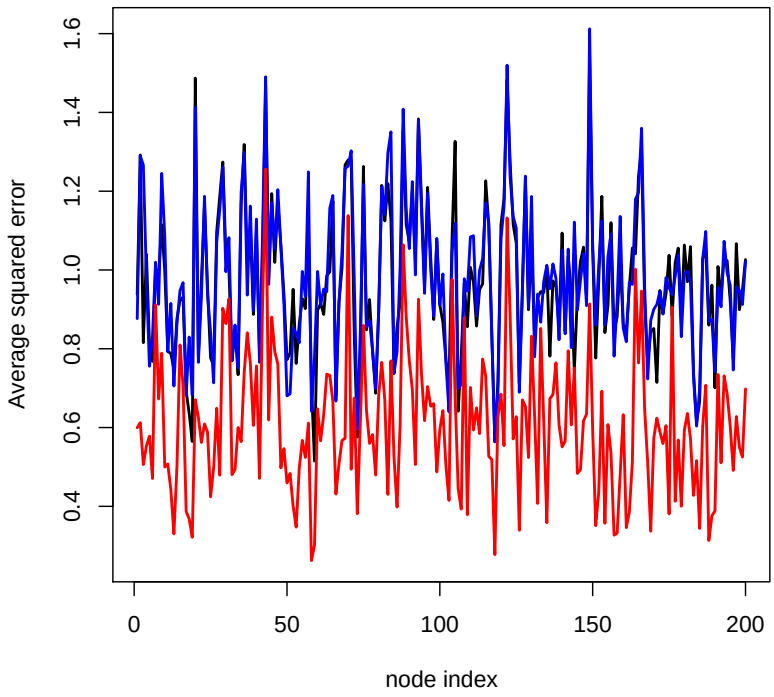


Figure 6.5: Average squared error plot for each node by using time domain, wavelet domain spatial method, and wavelet domain spatio-temporal method time series fitting. The average squared error from time domain fit is shown in black, wavelet domain spatial method is shown in blue and wavelet domain spatio-temporal method is shown in red.

horizon. For example, in R we can get the 10-step ahead predictions as follows,

`obj <- predict(arima(g[,k],c(1,1,0)), n.ahead=10)`,

`pred[,k] <- obj$pred`. Figure 6.6 shows the 10-step ahead forecast for node 1 using time domain, wavelet domain spatial method, wavelet domain spatio-temporal method forecasts.

We calculate 1-step ahead Mean Squared Prediction Error (MSPE) by repeating the experiment independently for 50 times. Figure 6.7 shows the squared error for 1-step ahead prediction from time domain, wavelet domain spatial method and wavelet domain spatio-temporal method forecasting. The 1-step ahead MSPE using time domain forecast is 2.1, wavelet domain spatial method is 2.08 and

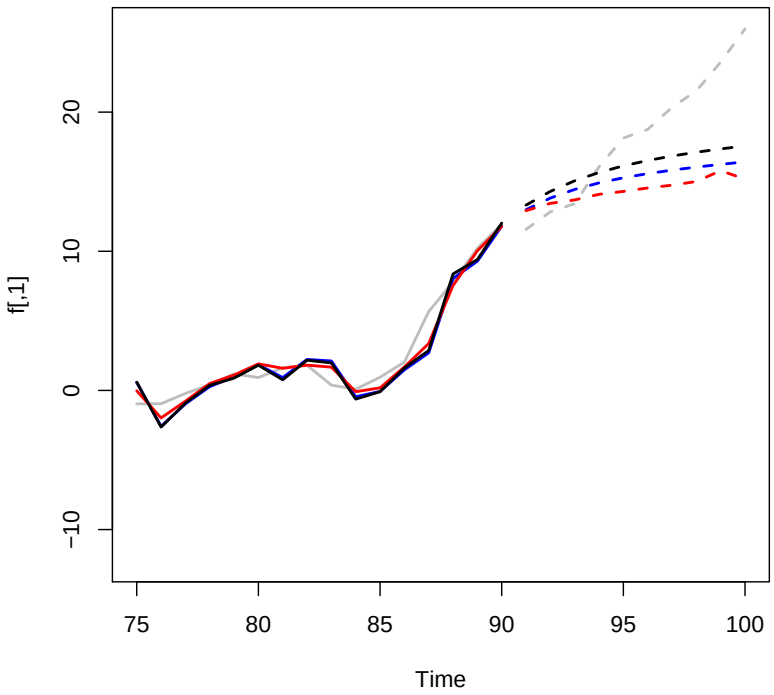


Figure 6.6: Prediction(10-step ahead) for node 1 using time domain, wavelet domain spatial method, spatio-temporal method forecast. The data is shown by the gray solid line, for the prediction interval it is shown in a gray dashed line. Time domain fit in black solid lines and prediction in black dashed lines. Wavelet domain spatial method fit in blue solid line and prediction in blue dashed line. Wavelet domain spatio-temporal method fit in red solid line and prediction in red dashed line.

wavelet domain spatio-temporal method is 3.9. Although the wavelet domain spatio-temporal method fits the data well but somehow fails to predict the future value accurately.

6.4.5 Simple Exponential Smoothing (SES)

We model our time series, $f_{t,k}$, as follows,

$$f_{T,k} = \alpha_k f_{T-1,k} + (1 - \alpha_k)\hat{f}_{T-1,k}(1), \tag{6.38}$$

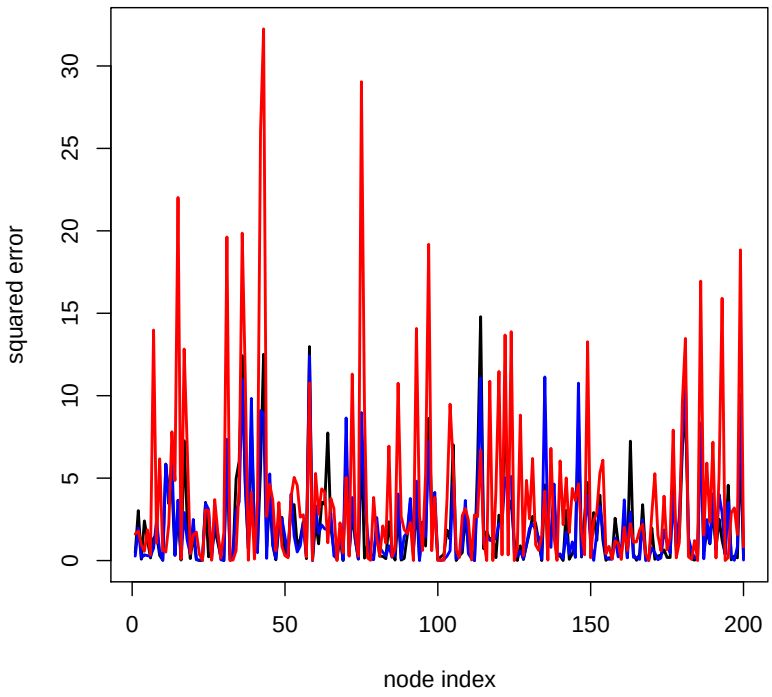


Figure 6.7: Squared error plot for 1-step ahead prediction using Box-Jenkins approach. Time domain error in black and wavelet domain spatial method error in blue and wavelet domain spatio-temporal method error red.

where $f_{T,k}$ is the observation for node k at time T, α_k is the parameter of the exponential smoothing which is estimated as given in section 6.3.2, $\hat{f}_{T-1,k}(1)$ is the 1-step ahead forecast for node k at time $T-1$.

Figure 6.8 shows the squared error for 1-step ahead prediction by repeating for 50 independent experiments for ARIMA(1,1,0). The average 1-step ahead MSPE for time domain forecast is 1.23, wavelet domain spatial method is 1.29 and wavelet domain spatio-temporal method is 1.65. We have also repeated this simulation with an ARIMA(0,1,1) model. Figure 6.8(b) shows the squared error for 1-step ahead prediction for ARIMA(0,1,1) model (again repeated for 50 experiments). The 1-step ahead MSPE for time domain forecast is 2.61, wavelet domain spatial method is 1.74 and wavelet domain spatio-temporal method is 2.3. Figure 6.9 shows the parameter α_k for both ARIMA(1,1,0) and ARIMA(0,1,1) models. The

α_k parameter in wavelet domain modelling of an ARIMA$(1,1,0)$ simulated data is generally smaller than that of the α_k for time domain modelling of the same data. On the other hand, α_k for both time domain and wavelet domain modelling of an ARIMA(0,1,1) simulated data is generally large.

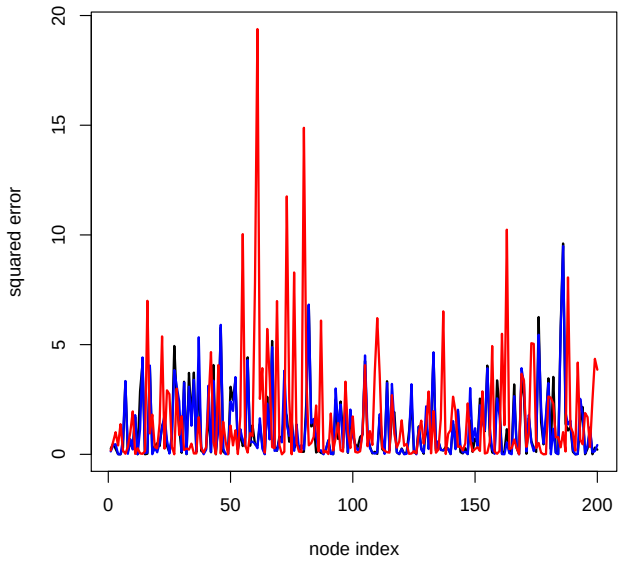
6.5 Real data study: Mumps data modelling

We have obtained mumps data in year 2005 from the Health Protection Agency (HPA). The mumps data is rearranged to get a time series data from week 1 to week 52 for the 47 counties in the UK. Let $\{x_{t,c}\}$ be a time series for a county c. By looking at the time series plots, e.g. 6.10(a) and 6.10(c), it has non-stationary behaviour. Figure 6.11 shows the acf, pacf plots for Avon, Bedfordshire and Berkshire counties. In some cases the data has two parts, weeks 1 to 30 has one phase where the disease was still spreading and week 31 to 52 has another phase where the spread of disease was controlled or stopped. We model the first 27 weeks as ARIMA(p_c,1,0) or ARIMA(p_c,0,0)(see plots of acf and pacf shown in figure 6.11). We decide which model to use by first fitting the data to both models and calculate the sum of squared errors from the fit and choose the model that has minimum squared error. We sometimes need to take first difference of the data to make the series stationary see figures 6.10(b) and 6.10(d). We denote the differenced series by $w_{t,c}$,

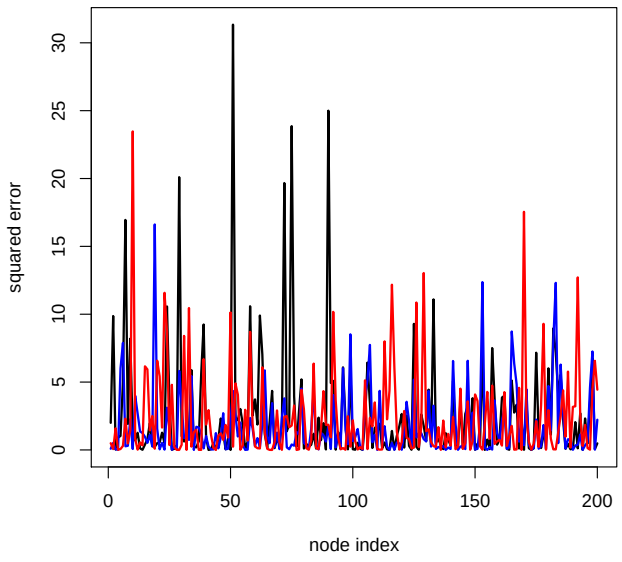
$$w_{t,c} = \nabla x_{t+1,c} = x_{t+1,c} - x_{t,c}. \tag{6.39}$$

The order p_c of the model is determined through AIC or MLE methods.

Once we have finished modelling the data in the time domain, we would like to model the data in the wavelet domain. In order to transform the data using the LOCAAT algorithm, we need to specify a network. We obtained R code from Matthew Nunes to link the nearest county towns (see figure 6.12). Then we trans-

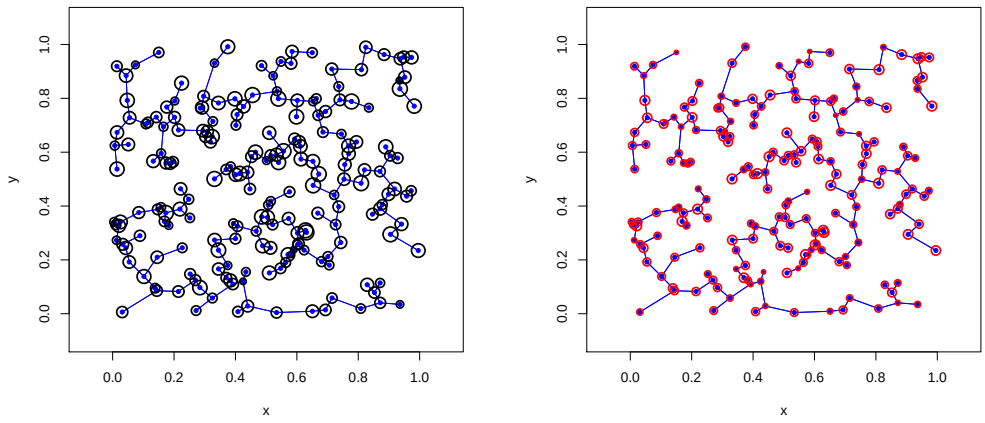


(a) ARIMA(1,1,0)



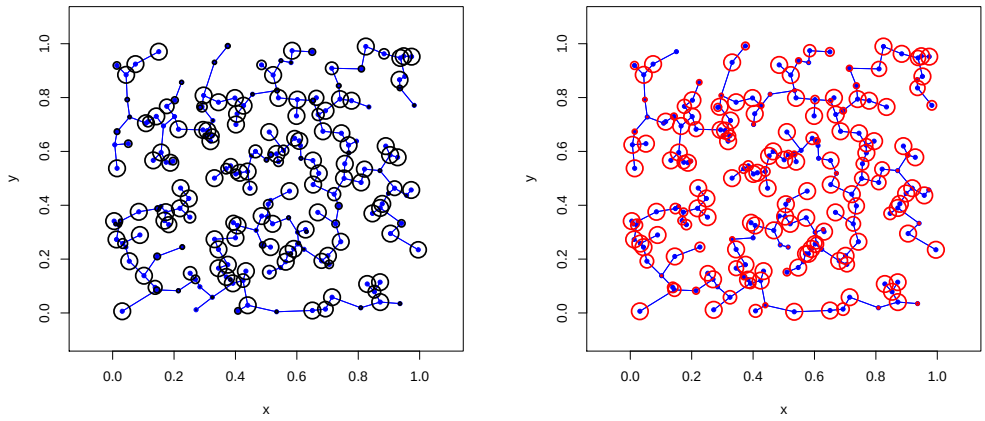
(b) ARIMA(0,1,1)

Figure 6.8: Squared error plot for 1-step ahead prediction for ARIMA(1,1,0) and ARIMA(0,1,1) simulation data using SES method. Time domain error in black and wavelet domain spatial method error in blue and wavelet domain spatio-temporal method error red.



(a) α_k for ARIMA(1,1,0) model in time domain

(b) α_k for ARIMA(1,1,0) model in wavelet domain



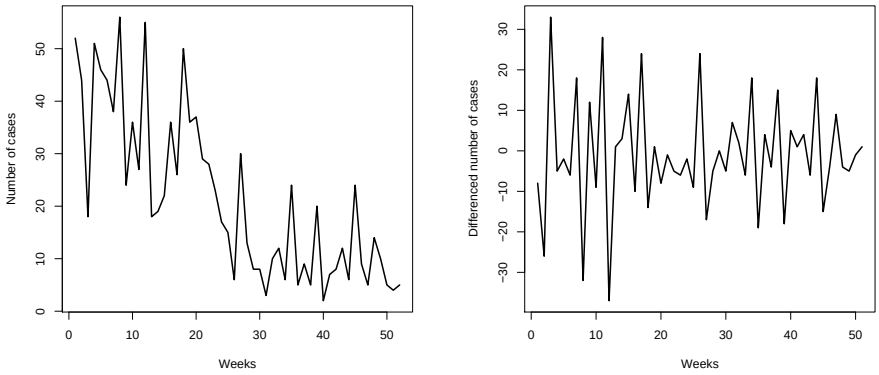
(c) α_k for ARIMA(0,1,1) model in time domain

(d) α_k for ARIMA(0,1,1) model in wavelet domain

Figure 6.9: The parameter α_k for exponential smoothing. The figure on the top shows α_k for ARIMA(1,1,0) and the bottom figure for ARIMA(0,1,1). The network (T-1) is shown in blue and the black circles are the α_k for time domain modelling and the red circles are α_k for wavelet domain spatial method coefficient modelling. The radius of the circles are proportional to α_k.

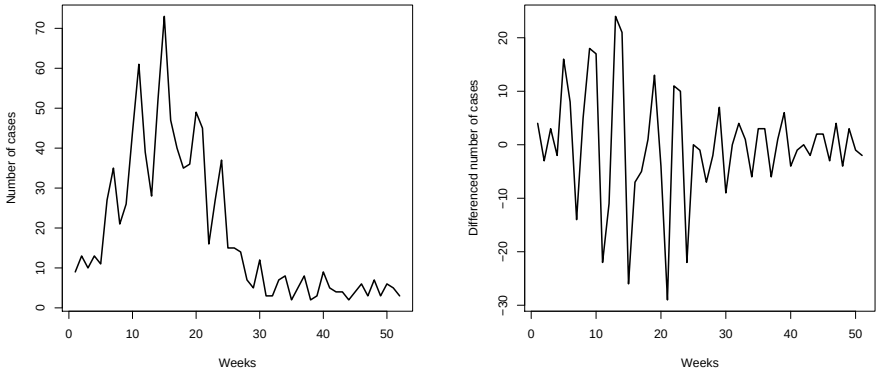
form the data into the wavelet domain using the LOCAAT transform and model the coefficients in similar way to the time domain modelling by choosing either ARIMA(p_c,1,0) or ARIMA(p_c,0,0) that best fits the coefficients.

Once we have modelled the series, we perform 3-steps ahead (an arbitrary choice) forecast on the series. Figure 6.13 shows the fitting and forecast for Bedfordshire



(a) Mumps cases registered for Avon County

(b) Differenced data for Avon County

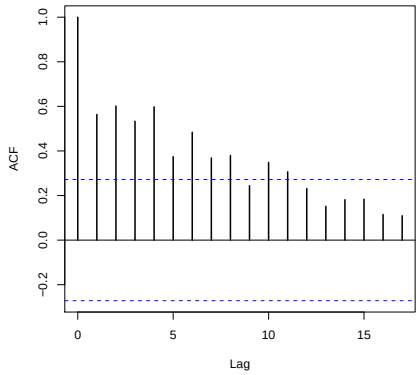


(c) Mumps cases registered for Bedfordshire County

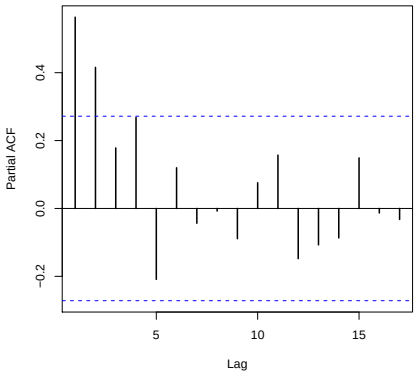
(d) Differenced data for Bedfordshire County

Figure 6.10: Number of Mumps cases recorded in year 2005 for Avon and Bedfordshire counties.

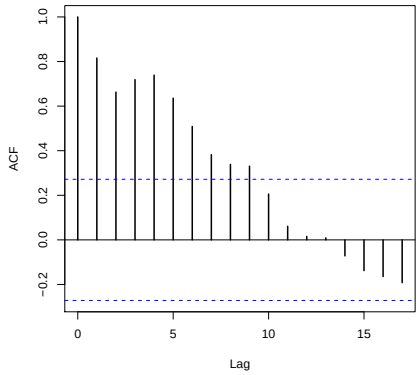
county as an example. We have also made a 1-step ahead forecast using SES. Figure 6.14 shows the average 1-step ahead prediction error from previous method (fitting ARIMA(p_c,1,0) or ARIMA(p_c,0,0)) and the SES method. The average fitting error using Box-Jenkins method for time domain modelling is 149.4, for wavelet domain spatial method is 148.42 and for wavelet domain spatio-temporal method is 115.4. The average 1-step ahead forecast error using Box-Jenkins method for time domain modelling is 285.7, for wavelet domain spatial method is 256.9 and for wavelet domain spatio-temporal model is 470.2. The average fitting error using SES method for time domain modelling is 65.2, for wavelet domain spatial



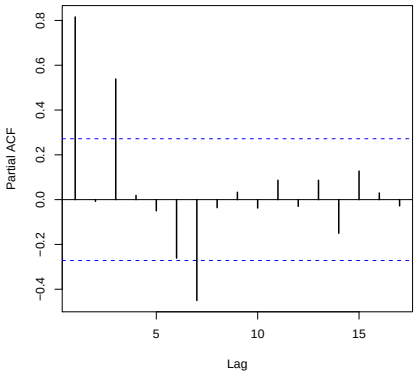
(a) acf plot for Avon County



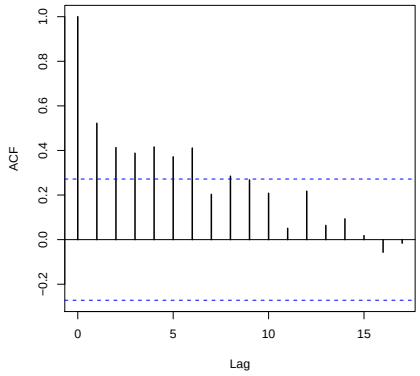
(b) pacf plot for Avon County



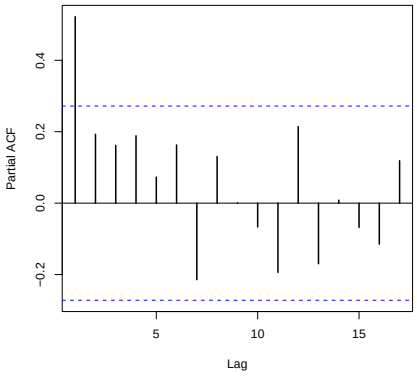
(c) acf plot for Bedfordshire County



(d) pacf plot for Bedfordshire County



(e) acf plot for Berkshire County



(f) pacf plot for Berkshire County

Figure 6.11: Autocorrelation function, partial autocorrelation function plots for Avon, Bedfordshire and Berkshire counties.

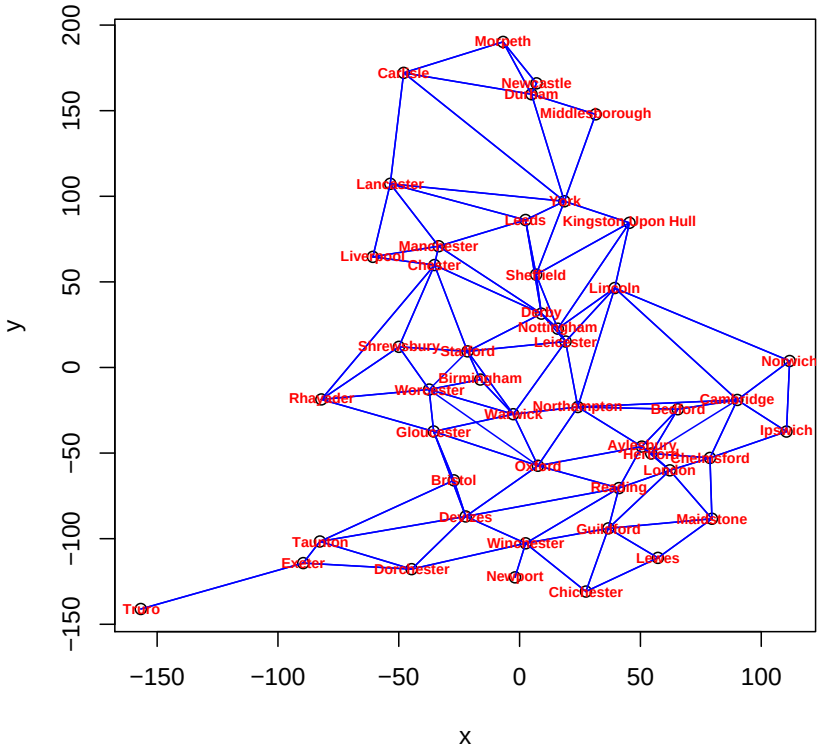


Figure 6.12: Nearest cities in each county is linked to form a network

method is 63.4 and for wavelet domain spatio-temporal method is 53.3. The average 1-step ahead forecast error using SES for time domain modelling is 75.1, for wavelet domain spatial method is 58.5 and for wavelet domain spatio-temporal method is 61.4.

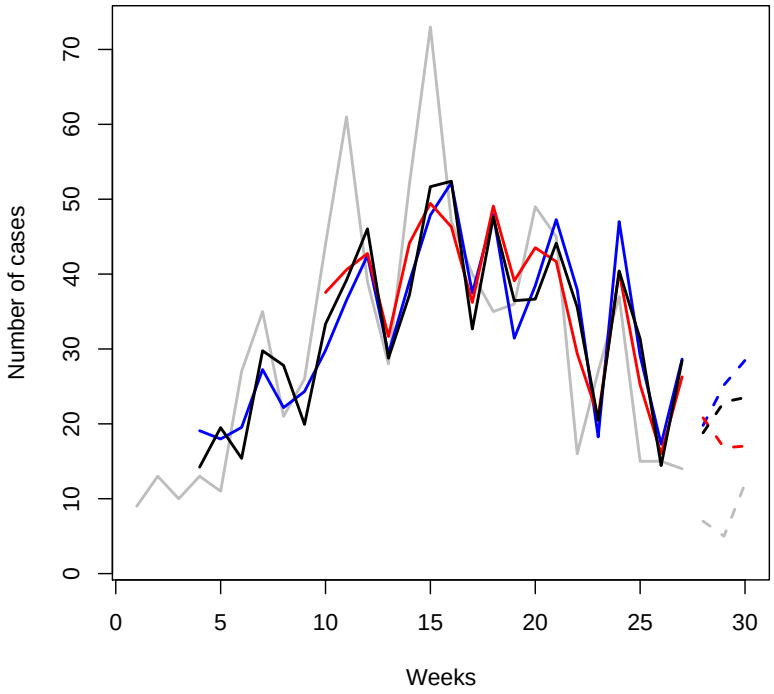
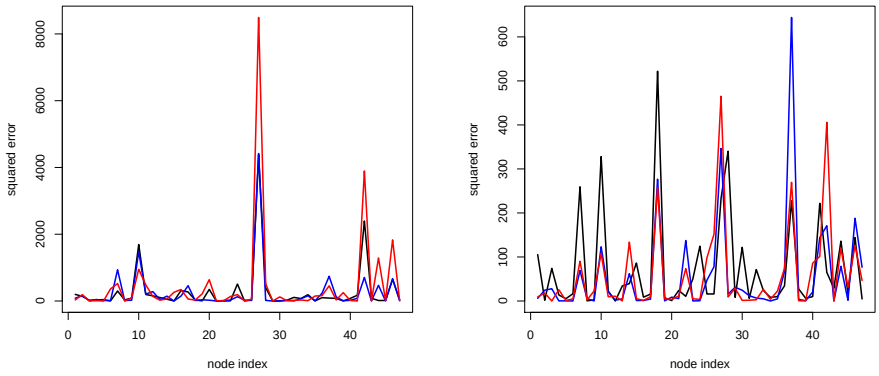


Figure 6.13: Prediction (3-step ahead) for Bedforshire county using time domain, wavelet domain spatial method, spatio-temporal method forecast. The data is shown in gray solid line, for the prediction interval it is shown in gray dashed line. Time domain fit in black solid lines and prediction in black dashed lines. Wavelet domain spatial method fit in blue solid line and prediction in blue dashed line. Wavelet domain spatio-temporal method fit in red solid line and prediction in red dashed line.



(a) 1-step ahead prediction error - Box-Jenkins method

(b) 1-step ahead prediction error - SES method

Figure 6.14: Squared error plot for 1-step ahead prediction using Box-Jenkins and SES methods. Time domain error in black and wavelet domain spatial method error in blue and wavelet domain spatio-temporal method error red.

6.6 Conclusions

We have studied time series modelling of networks in both the time and wavelet domains with the aid of simulation and real data. In the wavelet domain modelling, we considered treating each time layer separately (spatial network method) to produce LOCAAT coefficients and also connecting a few layers to build up a spatio-temporal based network. Generally, the fitting error is more or less similar for wavelet domain spatial method and time domain modelling whereas in the wavelet domain spatio-temporal method, the fitting error is typically smaller. We have also looked at forecasting techniques based on Box-Jenkins approach and a SES methods.

We initially looked at modelling and forecasting a time series which was generated by the process ARIMA(1,1,0). We have generated this process for a network of 200 nodes and looked at some automated modelling approaches by selecting ARIMA(p_k,1,0) or ARIMA(p_k,0,0) whichever fits the data well. We have adopted a similar approach for wavelet domain coefficient modelling. Although the wavelet domain spatio-temporal method fits the data well, it somehow fails to predict the data accurately. For this simulation, we found time domain, and wavelet domain spatial method forecasting yield similar performance. The fitting error is also similar for both of these cases.

Then we looked at SES approach of ARIMA(1,1,0) and ARIMA(0,1,1) simulated data. Again we found that wavelet domain spatio-temporal method fits the data well in both scenarios, and yet again fails to give a good forecast for 1-step ahead prediction. The performance of both time domain and wavelet domain spatial methods are similar for ARIMA(1,1,0), however for ARIMA(0,1,1), the wavelet domain spatial method gives a better forecast for 1-step ahead prediction. Although the 1-step ahead prediction of wavelet domain spatio-temporal method is better than time domain forecast for ARIMA(0,1,1) simulated data, yet it does not provide the best forecast (wavelet domain spatial method is the best forecast).

We finally looked at a real data study with the mumps data set. We looked at forecast using Box-Jenkins approach and SES methods. Again, wavelet domain spatio-temporal method fits the data well In both Box-Jenkins and SES, however it fails to yield a good forecast. The wavelet domain spatial method provides the good forecast for 1-step ahead prediction in both forecasting methods, however the average prediction error is better with SES.

The results presented for wavelet domain spatio-temporal method is based on overlapping window based approach, and it somehow seems to overfit the data. Further work could investigate suitable modelling for spatio-temporal LOCAAT coefficients. Further work could also investigate the non-overlapping window based approach. We could investigate to find a good forecasting method for the spatio-temporal approach since it yields the best the fit for all the scenarios we have studied. We also need to study other time series models with our wavelet domain time series fitting and forecasting methods (one example is to investigate Spatial Autoregressive (SAR) models).

Chapter 7

Conclusions and Future Work

In this chapter we summarise our key achievements on a chapter by chapter basis. We then give some future directions in the final section.

7.1 Noise variance estimation for networks

Estimating the noise variance σ^2 is crucial for the success of almost all the wavelet shrinkage methods. We first approached the problem by using the Median Absolute Deviation (MAD) technique (introduced in [34]) for the LOCAAT coefficients. We found that the MAD technique provides a reliable estimation when the number of nodes is large. In some applications we may be dealing with a network with few nodes. Hence, we introduced a local MAD technique which gives a reliable estimation even with a relatively small number of nodes (see section 4.2).
We then applied the variogram technique introduced in [42]. We propose several methods to estimate noise variance directly from network data. For these methods, we considered estimation for two cases,

- when time series data for each node is available
- only a snapshot of spatial network data (i.e. single time point)

We established that the noise variance can be estimated reliably when the time series data is available (see subsection 4.3.1). However, we have also proposed a method that can reliably estimate noise variance with a snapshot of network data with large number of nodes (see subsections 4.3.3, 4.3.4 and 4.3.5).
We moved on to estimating noise variance from lifting coefficients using variogram methods. We proposed several variogram-based methods in the lifting domain. Again we consider scenarios when time series is availble and situations where estimation is performed purely on the network data at a given snapshot of time. We established that the variogram method performs well when time series data is available (see subsection 4.3.6). However, we have proposed a method which estimates the noise variance reliably, purely from the snapshot of network data even with few nodes (see subsection 4.3.9). Some of these variogram-based techniques require bias correction and we introduced a method to find this.

Main contributions

- Proposed two local Median Absolute Deviation (MAD) estimators for coefficient variance for LOCAAT (section 4.2).

- Introduced two noise variance estimation, using variogram techniqe, when time series for each node is available. The methods are called TM1 and WM1 in chapter4.

- Introduced several noise variance estimation methods for networks, using variogram techniques, purely based on spatial data (subsections 4.3.1, 4.3.3, 4.3.4 and 4.3.5).

- Introduced several noise variance estimation methods for networks, using the variogram, purely based on LOCAAT coefficients (spatial only, no time series involved. see subsections 4.3.7, 4.3.8 and 4.3.9).

- Proposed a method to find bias correcting constants for variogram methods (section 4.4).

7.2 Thresholding methods

We explored thresholding LOCAAT coefficients using some existing thresholding techniques. We have found that these methods introduce a large bias into the estimation and therefore needed some investigation using a lower threshold level. We see that various threshold techniques require different threshold levels, $p\sigma\sqrt{2\log n}$ (where p is the percentage of universal threshold, n is the number of detail coefficients and σ is the noise standard deviation), to obtain a good minimum Mean Squared Error (MSE) performance. We explored a mean correction method which compensates for the bias at the expense of increased variance and this method is less sensitive to the choice of p.

Having explored some thresholding techniques for LOCAAT coefficients, we found that further improvement to the estimation efficiency is possible by avoiding some sensitive coefficients from being thresholded. However, this method is computationally inefficient. We then found that cross validation and Stein's Unbiased Risk Estimator (SURE) based threshold selection for LOCAAT coefficients also results in better estimation.

When time series data are available, the spatio-temporal function estimation efficiency can be further improved by smoothing the LOCAAT coefficients for a particular node by considering coefficients from adjacent time slots, e.g. take the average of current coefficient, previous coefficient for that particular node and the next coefficient for the same node.

Finally we explored sparsity and recommended an optimal stopping time for LOCAAT transform. The optimal stopping time lies around 90% of transformation, i.e, it is recommended to keep 10% as the scaling coefficients rather than transforming data until 2 scaling coefficients remaining (as is default in the software)

and perform thresholding on the detail coefficients.

Main contributions

- Examination of existing thresholding strategies for LOCAAT and the introduction of a *mean correction* approach to account for bias issue (sections 5.3, 5.4 and 5.6). We found unacceptable estimation bias and suggested using a lower threshold level.
- We proposed a block thresholding-like approach for spatial wavelet coefficients (see section 5.5).
- A proposal for improving estimation efficiency by avoiding sensitive coefficient thresholding (see section 5.8).
- Cross validation-based and SURE-based threshold selection approaches for LOCAAT method (sections 5.9, 2.5.4 and 2.5.5).
- Space-time function estimation using a concept called the spatio-temporal network method, and estimation improvement by smoothing temporal coefficients (section 5.13).
- Exploration of the optimal stopping time for LOCAAT method (section 5.15).

7.3 Network forecasting

We investigated the time series modelling in both the time domain and the lifting domain. At first, we treated each node time series separately and modelled it using a simple ARIMA(p, 0,0) or ARIMA(p,1,0). Then again, for the same time series data, we performed the LOCAAT transform sequentially using spatio-network and spatio-temporal network methods. We then model these LOCAAT coefficients for each node separately using ARIMA(p,0,0) or ARIMA(p). We establish that the spatio-temporal based LOCAAT coefficient modelling fits the data well.

We performed a simple 1-step ahead forecasting using the Box-Jenkins approach and Simple Exponential Smoothing (SES) methods for both time domain and lifting domain modelling. We establish that the lifting domain forecast results in a good prediction for an ARIMA(0,1,1) simulated data.
We performed modelling and forecasting using our network forecasting technique for mumps disease data in 2005 (obtained from HPA). We establish that lifting domain SES based forecast results in better prediction for a 1-step ahead forecast.

Main contributions

- We proposed a better network forecast technique using lifting coefficient modelling (section 6.4).

7.4 Future work

We have discussed some of the possibilities for future work, for the methods we proposed, in the conclusions in each chapters. We point out some key future work in this section.
The LOCAAT algorithm we used throughout our work utilises inverse distance weights to calculate the detail coefficients. An investigation into using different weights such as an exponentially decaying weights may be advisable. Investigation into considering second order neighbours in the calculation of detail coefficients may be advisable (without over-complicating and making the technique computationally inefficient).
We have constructed the spatio-temporal network by using time depth $M = 3$. Further investigation is required to find out what is a good choice in different situations. We have also seen that estimation efficiency increases by smoothing temporal coefficients. Again, further investigation is required to find what is an optimum smooth (i.e. how many previous coefficients and how many future coefficients). Further improvement to Box Block Choice (BBC) method is possible

by considering temporal lifting coefficients into the decision.

Investigation of more time series models and forecasting techniques, with our proposed network forecasting method, is required. Further investigation is required to find out the failure (or find the best forecasting technique) of spatio-temporal network based prediction.

The methods proposed in the thesis require a static network setting. An adaptaion of these techniques to a dynamic network (network that changes with time), which is a common scenario in communication wireless networks, will be interesting.

Appendix A

Separable form denoising of space- time functions

Let $f_{\mathbf{x}_k,t}$ be the observation of a space-time function, $S(\mathbf{x}_k, t)$, and assume the following model,

$$f_{\mathbf{x}_k,t} = g(\mathbf{x}_k) + C(t) + \epsilon_{\mathbf{x}_k,t}, \tag{A-1}$$

where $g(\mathbf{x}_k)$ is the true spatial function value at site k, $C(t)$ is the temporal variation and $\epsilon_{\mathbf{x}_k,t}$ is the Gaussian error with mean zero and variance σ^2. There are n sites in total. We are interested in estimating the space-time function given by,

$$S(\mathbf{x}_k, t) = g(\mathbf{x}_k) + C(t). \tag{A-2}$$

We define the spatial average as,

$$\begin{aligned} \overline{G}(t) &= \frac{1}{n}\sum_{k=1}^{n} f_{\mathbf{x}_k,t} \\ &= \frac{1}{n}\sum_{k=1}^{n} g(\mathbf{x}_k) + \frac{1}{n}\sum_{k=1}^{n} C(t) + \frac{1}{n}\sum_{k=1}^{n} \epsilon_{\mathbf{x}_k,t} \\ &\approx \bar{g} + C(t), \end{aligned} \tag{A-3}$$

where $\bar{g} = \frac{1}{n}\sum_{k=1}^{n} g(\mathbf{x}_k)$. Now we define the following difference between the observation and the spatial average,

$$\begin{aligned} D_{\mathbf{x}_k,t} &= f_{\mathbf{x}_k,t} - \overline{G}(t) \\ &= f_{\mathbf{x}_k,t} - \bar{g} - C(t) \\ &= g(\mathbf{x}_k) - \bar{g} + \epsilon_{\mathbf{x}_k,t}. \end{aligned} \tag{A-4}$$

Now we define the following average for each site $\mathbf{x}_k$,

$$\overline{D}(\mathbf{x}_k) = \frac{1}{T}\sum_{t=1}^{T} D_{\mathbf{x}_k,t} \approx g(\mathbf{x}_k) - \bar{g}. \tag{A-5}$$

Therefore the estimator of the space-time function is given by,

$$\hat{S}(\mathbf{x}_k, t) = \overline{G}(t) + \overline{D}(\mathbf{x}_k). \tag{A-6}$$

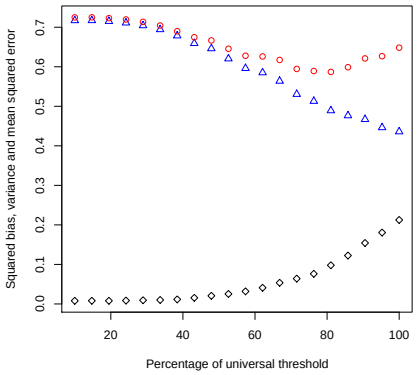
Appendix B

Further Results

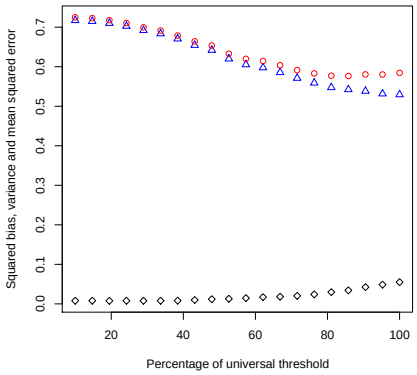
In this appendix we include some further detailed results.

B.1 Further results from various thresholding methods

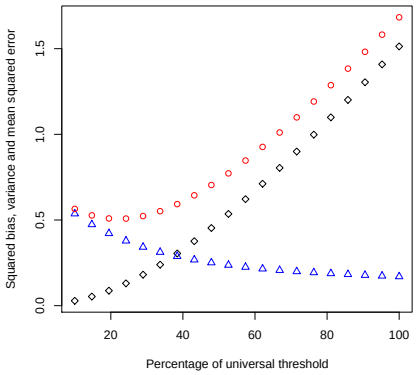
We present results of bias, variance and MSE plots of hard, soft, NeighCoeff, block, Box Block Choice (BBC) thresholding and mean correction method with a T-1 network with different number of nodes ($n = 50$ and $n = 150$). Figures B.1 and B.2 shows the results with $n = 50$ with a test function g^1. Figures B.3 and B.4 shows the results with $n = 150$ with a test function g^1.



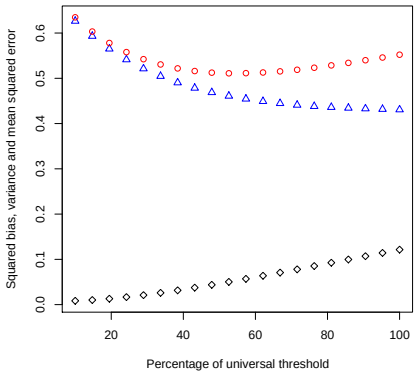
(a) Hard Thresholding



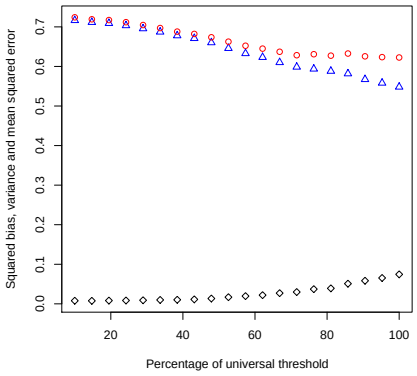
(b) Hard thresholding with mean correction



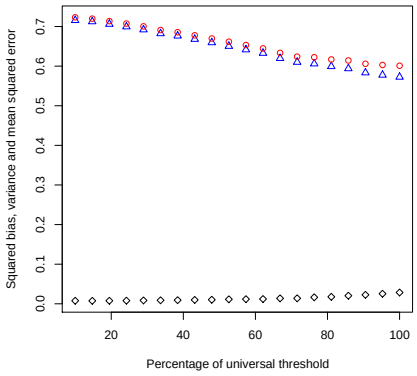
(c) Soft Thresholding



(d) Soft Thresholding with mean correction

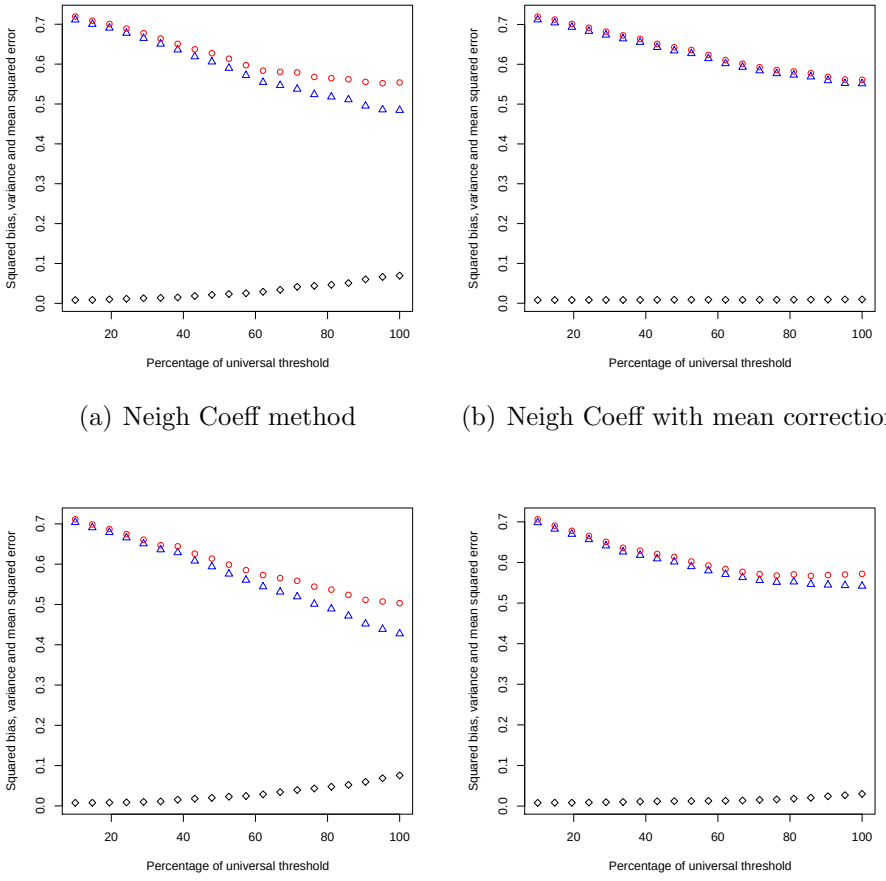


(e) Block Thresholding



(f) Block Thresholding with mean correction

Figure B.1: Various thresholding results for Type-1 network with 50 nodes, Average squared bias(=$\diamond$), variance(=$\triangle$) and mean squared error(=$\circ$) against $p\hat{\sigma}\sqrt{2\log n}$, p=varying percentage of universal threshold.



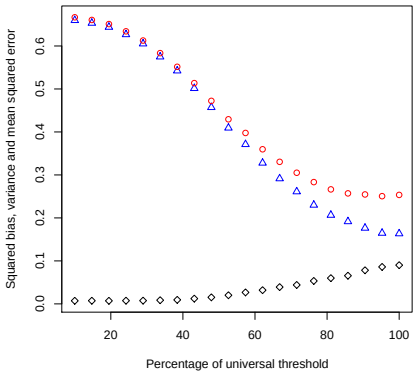
(a) Neigh Coeff method

(b) Neigh Coeff with mean correction

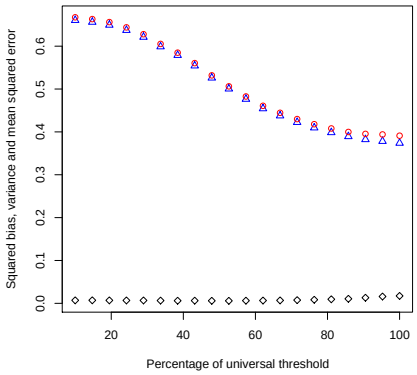
(c) BBC

(d) BBC with mean correction

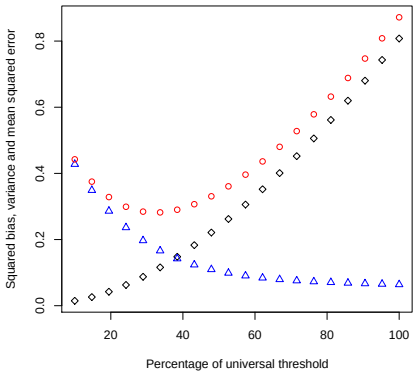
Figure B.2: Various thresholding results for Type-1 network with 50 nodes, Average squared bias(=◇), variance(=△) and mean squared error(=○) against $p\hat{\sigma}\sqrt{2\log n}$, p=varying percentage of universal threshold.



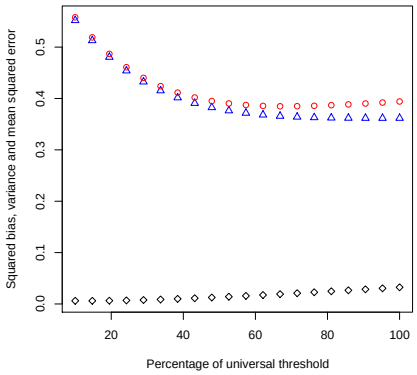
(a) Hard Thresholding



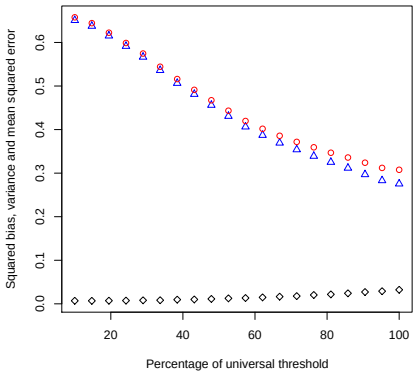
(b) Hard thresholding with mean correction



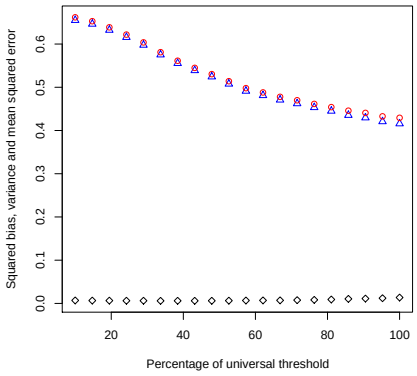
(c) Soft Thresholding



(d) Soft Thresholding with mean correction

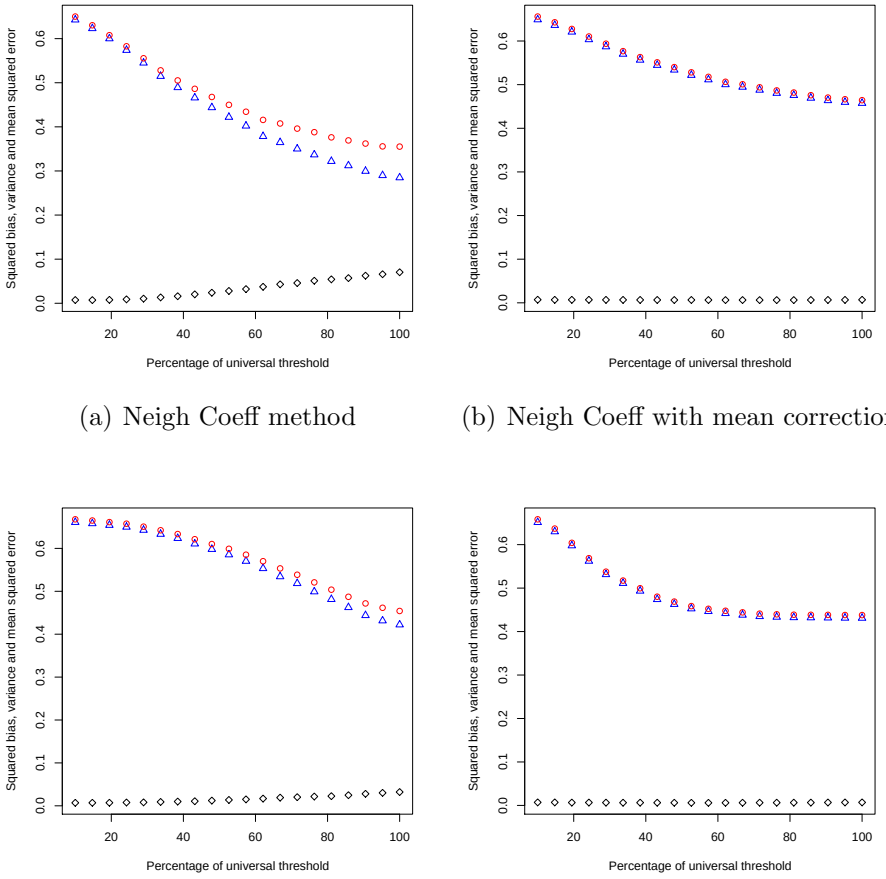


(e) Block Thresholding



(f) Block Thresholding with mean correction

Figure B.3: Various thresholding results for Type-1 network with 150 nodes, Average squared bias(=$\diamond$), variance(=$\triangle$) and mean squared error(=$\circ$) against $p\hat{\sigma}\sqrt{2\log n}$, p=varying percentage of universal threshold.



(a) Neigh Coeff method

(b) Neigh Coeff with mean correction

(c) BBC

(d) BBC with mean correction

Figure B.4: Various thresholding results for Type-1 network with 150 nodes, Average squared bias(=◇), variance(=△) and mean squared error(=○) against $p\hat{\sigma}\sqrt{2\log n}$, p=varying percentage of universal threshold.

B.2 Optimised risk values

Table B.1 shows the λ^* that minimse risk estimates SURE, MOCV, and GCV with T-2 network.

Signals	λ^*			Efficiency		
	SURE	GCV	MOCV	SURE	GCV	MOCV
g^1	0.391	3.304	2.056	1.187	1.295	1.537
g^2	3.144	1.623	1.530	0.097	0.545	0.710
maartenfunc	2.090	1.513	0.670	1.016	1.270	1.662
2D Doppler	3.151	1.844	1.314	0.240	0.649	0.847

Table B.1: Optimising the risk and corresponding efficiencies for type-2 (T-2) network (spatial network) with SNR = 2 and n = 500.

Appendix C

Qualnet

Below is a very brief introduction to Qualnet and for more detailed description refer to programmer's guide and introduction manuals provided by Scalable Network Technologies.

Qualnet is a commercial version of the free network simulation tool *glomosim*. Qualnet has a Graphical User Interface (GUI) and is easy to use. Execution of scenarios on *glomosim* is done via command line execution. Qualnet can execute both through command line and a GUI. The GUI-based tool enables animations which give a better understanding of the tool and communication protocols. The GUI-based execution is slow under heavy simulations. Thus, it is recommended, once the user is familiar with the tool, they are encouraged to use command line execution.

C.1 Changes made

Windows based Qualnet uses *qualnet.exe* therefore any changes made to the source files has to be recompiled in order for the changes to take effect. The main source files are written in *c++*. The files modified in order to collect necessary information required are, partition.cpp, aodv.cpp, node.cpp. The function *nodeprocessevent* in node.cpp in order to dump the statistics.

C.2 R

R [76] is a language and environment for statistical computing. It is free and can be installed on any operating system. It can be obtained from http://www.R-project.org. Throughout my research R is the main program used for analysis and processing and transform of data. The tree based LOCAAT tranform introduced in [46] is implemented by Professor Guy Nason on R and the package *NetTree* is available on request from Professor Guy Nason. R scripts I have written are available on request.

Appendix D

Some R codes

```
dopplerfunc <- function (x,y) {
r <- sqrt(x^2 + y^2)
f <- sin(1/(r^2))
f }

blockfunc <- function(x,y){
function(x,y){
f <- rep(0, length(x)) sv <- x < 0.1 f[sv] <- f[sv] + 1 sv <- y <
0.2 f[sv] <- f[sv] + 2 sv <- (x>0.3) & (x < 0.4) & (y<0.8) &
(y>0.7) f[sv] <- f[sv] + 3 sv <- (x>0.7) & (x < 0.8) & (y<0.8) &
(y>0.7) f[sv] <- f[sv] + 4 sv <- (x>0.5) & (x < 0.6) & (y<0.6) &
(y>0.4) f[sv] <- f[sv] + 5 sv <- (x>0.3) & (x < 0.8) & (y<0.3) &
(y>0.2) f[sv] <- f[sv] + 6 sv <- (x>0.2) & (x < 0.3) & (y<0.4) &
(y>0.3) f[sv] <- f[sv] + 7 sv <- (x>0.8) & (x < 0.9) & (y<0.4) &
(y>0.3) f[sv] <- f[sv] + 8 f
}

heavisinefunc <- function(x,y, pp=0.005, sd=0.01, freq=20){
r <- sqrt(x^2 + y^2)
f1<- sin(freq*r)
```

```
f2 <- pp*dnorm(x,0.55,sd=sd)*dnorm(y,0.5, sd=sd)
f1+f2
}

bumpsfunc <- function (x,y) { xc <- c(0.1, 0.8, 0.9) yc <- c(0.4,
0.7, 0.1) vc <- c(0.01,0.02, 0.015) nc <- length(xc) ans <- rep(0,
length(y)) for(i in 1:nc)
ans <- ans + doubexp(x, mean=xc[i], rate=sqrt(vc[i])) *
doubexp(y, mean=yc[i], rate=sqrt(vc[i]))
ans }
```

where the function `doubexp` is defined as,

`function (x, mean=0, rate=1) { exp( - abs(x-mean)/rate)/(2*rate)}`.

`maartenfunc`$(x, y) = (2x + y)\mathbb{I}((3x - y) < 1) + (5x - y)\mathbb{I}((3x - y) \geq 1)$.

Bibliography

[1] F. Abramovich, T. Sapatinas, and B. W. Silverman, "Wavelet thresholding via a Bayesian approach," *Journal of the Royal Statistical Society*, vol. 60, no. 4, pp. 725–749, 1998.

[2] P. Abry, P. Goncalvès, and P. Flandrin, "Wavelet-based spectral analysis of l/f processes," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, pp. 237–240, April 1993.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communication Magazine*, vol. 40, no. 8, pp. 102–114, August 2002.

[4] J. N. Al-Karai and A. E. Kamal, "Routing techniques in wireless sensor networks: A survey," *Wireless Communications, IEEE*, vol. 11, pp. 6– 28, December 2004.

[5] A. Antoniadis, E. Paparoditis, and T. Sapatinas, "A functional wavelet-kernel approach for time series prediction," *Journal of the Royal Statistical Society B*, vol. 68(part 5), pp. 837–857, 2006.

[6] S. Basagni, M. Conti, S. Giordano, and I. Stojmenović, *Mobile ad hoc networking*. IEEE, 2004.

[7] J. F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal, *Numerical Optimization: Theoretical and Practical Aspects*. Springer, 2003.

[8] G. Box, G. Jenkins, and G. Reinsel, *Time Series Analysis: Forecasting and Control*, 3rd ed. Prentice Hall, 1994.

[9] C. S. Burrus, R. A. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelets Transforms : A Primer*. Prentice Hall, Inc, 1998.

[10] T. T. Cai and B. W. Silverman, "Incorporating information on neighboring coefficients into wavelet estimation," *Sankhya*, vol. 63, pp. 127–148, 2001.

[11] T. Cai, "Minimax wavelet estimation via block thresholding," *Technical Report # 96-41, Department of Statistics, Purdue University*, 1996.

[12] ——, "Wavelet regression via block thresholding: adaptivity and the choice of block size and threshold level," *Technical Report # 99-14, Department of Statistics, Purdue University*, 1999.

[13] ——, "On block thresholding in wavelet regression: Adaptivity, block size, and threshold level," *Statistica Sinica*, vol. 12, pp. 1241–1273, 2002.

[14] C. Chatfield, *The Analysis of Time Series An Introduction*, 6th ed. Chapman & Hall/CRC, 2004.

[15] E. Chicken and T. T. Cai, "Block thresholding for density estimation: local and global adaptivity," *Journal of Multivariate Analysis*, vol. 95, p. 76106, 2005.

[16] R. Christensen, *Linear Models for Multivariate Time Series and Spatial Data*. Springer, 1991.

[17] C. K. Chui, L. Montefusco, and L. Puccio, *Wavelets: theory, algorithms, and applications*. Academic Press, Inc., 1994.

[18] A. Ciancio and A. Ortega, "A distributed wavelet compression algorithm for wireless multihop sensor networks using lifting," *IEEE International*

Conference on Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05), vol. 4, pp. 825 – 828, March 2005.

[19] W. S. Cleveland, "Robust locally weighted regression and smoothing scatterplots," *Journal of the American Statistical Association*, vol. 74, no. 368, pp. 829–836, December 1979.

[20] W. S. Cleveland and S. J. Devlin, "Locally weighted regression: An approach to regression analysis by local fitting," *Journal of the American Statistical Association*, vol. 83, no. 403, pp. 596–610, September 1988.

[21] W. S. Cleveland, S. J. Devlin, and E. Grosse, "Regression by local fitting: methods, properties and computational algorithms," *Journal of Econometrics*, vol. 37, pp. 87–114, January 1988.

[22] W. S. Cleveland and E. Grosse, "Computational methods for local regression," *Statistics and Computing*, vol. 1, no. 1, pp. 47–62, September 1991.

[23] M. Clyde and E. I. George, "Flexible empirical Bayes estimation for wavelets," *Journal of the Royal Statistical Society*, vol. 62, no. 4, pp. 681–698, 2000.

[24] N. Cressie and D. M. Hawkins, "Robust estimation of the variogram: I," *Mathematical geology*, vol. 12, no. 2, pp. 115–125, 1980.

[25] N. A. Cressie, *Statistics for Spatial Data*. John Wiley & Sons, Inc., 1993.

[26] I. Daubechies, *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics (SIAM), 1992.

[27] ——, *Different perspectives on wavelets*. American Mathematical Society, 1994, vol. 47.

[28] V. Delouille, M. Jansen, and R. von Sachs, "Second generation wavelet methods for denoising of irregularly spaced data in two dimensions," *Signal Processing*, vol. 86, pp. 1435 – 1450, July 2006.

[29] V. Delouille, J. Simoens, and R. von Sachs, "Smooth design-adapted wavelets for nonparametric stochastic regression," *Journal of the American Statistical Association*, vol. 99, pp. 643 – 658, 2004.

[30] V. Delouille and R. von Sachs, "Smooth design-adapted wavelets for half-regular designs in two dimensions," *Discussion Paper 0226, Institut de Statistique, Université Catholique de Louvain, Belgium.*, October, 2002.

[31] P. D. Dolan, S. S. Agaian, and J. Noonan, "An examination of decomposition sparsity," *Digital Signal Processing*, vol. 14, pp. 125–137, March 2004.

[32] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 613–627, 1995.

[33] D. L. Donoho and I. M. Johnstone, "Adapting to unknown smoothness via wavelet shrinkage," *Journal of the American Statistical Association*, vol. 90, no. 432, pp. 1200 – 1224, 1995.

[34] ——, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, August, 1994.

[35] P. Fryzlewicz, S. V. Bellegem, and R. V. Sachs, "Forecasting non-stationary time series by wavelet process modelling," *Annals of the Institute of Statistical Mathematics*, vol. 55, no. 4, pp. 737–764, 2003.

[36] D. Ganesan, A. Cerpa, W. Ye, Y. Yu, J. Zhao, and D. Estrin, "Networking issues in wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 64, pp. 799–814, July 2004.

[37] D. Ganesan, D. Estrin, and J. Heidemann, "Dimensions: Why do we need a new data handling architecture for sensor networks," *ACM, Computer Communication Review*, vol. 33, pp. 143–148, October 2002.

[38] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann, "An evaluation of multi-resolution storage for sensor networks," *Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 89 – 102, November 2003.

[39] P. Hall, G. Kerkyacharian, and D. Picard, "On the minimax optimality of block thresholded wavelet estimators," *Statistica Sinica*, vol. 9, pp. 33–49, 1999.

[40] ——, "Block threshold rules for curve estimation using kernel and wavelet methods," *The Annals of Statistics*, vol. 26, no. 3, pp. 922–942, June 1998.

[41] D. M. Hawkins and N. Cressie, "Robust kriging - a proposal," *Mathematical geology*, vol. 16, no. 1, pp. 3–18, 1984.

[42] H.-C. Huang and N. Cressie, "Deterministic/stochastic wavelet decomposition for recovery of signal from noisy data," *Technometrics*, vol. 42, no. 3, pp. 262 – 276, August, 2000.

[43] M. Jansen and A. Bultheel, "Smoothing irregularly sampled signals using wavelets and cross validation," Department of Computer Science, K.U.Leuven, Tech. Rep., 1999.

[44] M. Jansen, M. Malfait, and A. Bultheel, "Generalized cross validation for wavelet thresholding," *Signal Processing*, vol. 56, pp. 33–44, 1995.

[45] M. Jansen, G. P. Nason, and B. W. Silverman, "Scattered data smoothing by empirical Bayesian shrinkage of second-generation wavelet coefficients," *Wavelet Applications in Signal and Image Processing IX, Proceedings of SPIE*, vol. 4478, pp. 87 – 97, 2001.

[46] ——, "Multiscale methods for data on graphs and irregular multidimensional situations," *Journal of the Royal Statistical Society*, vol. 71, no. 1, pp. 97–125, 2009.

[47] ——, "Simulations and examples for multivariate nonparametric regression using lifting," *Technical Report 04:18, Department of Mathematics, University of Bristol, UK.*, November, 2004.

[48] I. M. Johnstone and B. W. Silverman, "Wavelet threshold estimators for data with correlated noise," *Journal of the Royal Statistical Society*, vol. 59, no. 2, pp. 319–351, 1997.

[49] I. M. Johnstone, "Wavelets and the theory of nonparametric function estimation," *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, vol. 357, no. 1760, pp. 2475–2493, September 1999.

[50] I. M. Johnstone and B. Silverman, "Ebayesthresh: R and s-plus programs for empirical Bayes thresholding," 2002.

[51] ——, "Needles and straw in haystacks: Empirical Bayes estimates of possibly sparse sequences," *The Annals of Statistics*, vol. 32, no. 4, pp. 1594–1649, 2004.

[52] ——, "Empirical Bayes selection of wavelet thresholds," *The Annals of Statistics*, vol. 33, no. 4, pp. 1700–1752, 2005.

[53] H. Karl and A. Willig, *Protocols and architectures for wireless sensor networks*. John Wiley & Sons, Inc., 2005.

[54] W. S. Kerwin and J. L. Prince, "The Kriging update model and recursive space-time function estimation," *IEEE Transactions on Signal Processing*, vol. 47, no. 11, pp. 2942 – 2952, November 1999.

[55] A. Kovac, "Wavelet thresholding for unequally spaced data," Ph.D. dissertation, University of Bristol, March 1998.

[56] A. Kovac and B. W. Silverman, "Extending the scope of wavelet regression methods by coefficient-dependent thresholding," *Journal of the American Statistical Association*, vol. 95, pp. 172–183, 2000.

[57] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674–693, July 1989.

[58] ——, *A Wavelet Tour of Signal Processing*. Academic Press, 1998.

[59] G. Matheron, "Principles of geostatistics," *Economic geology*, vol. 58, pp. 1246–1266, 1963.

[60] E. J. McCoy and A. T. Walden, "Wavelet analysis and synthesis of stationary long-memory processes," *Journal of Computational and Graphical Statistics*, vol. 5, no. 1, pp. 26–56, March 1996.

[61] Y. Meyer, *Wavelets and operators*. Cambdridge University Press, 1992.

[62] E. A. Nadaraya, "On estimating regression," *Theory of Probability and its Applications*, vol. 9, pp. 141–142, January 1964.

[63] G. P. Nason, "Wavelet shrinkage using cross-validation," *Journal of the Royal Statistical Society*, vol. 58, no. 2, pp. 463–479, 1996.

[64] ——, "Choice of wavelet smoothness, primary resolution and threshold in wavelet shrinkage," *Statistics and Computing*, vol. 12, pp. 219–227, 2002.

[65] ——, *Wavelet Methods in Statistics with R*. Springer, 2008.

[66] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, January 1965.

[67] J. Nocedal and S. J. Wright, *Numerical Optimization.* Springer, 1999.

[68] M. A. Nunes, M. I. Knight, and G. P. Nason, "Adaptive lifting for nonparametric regression," *Statistics and Computing*, vol. 16, no. 2, pp. 143–159, June, 2006.

[69] M. A. Nunes, "Some new multiscale methods for curve estimation and binomial data," Ph.D. dissertation, University of Bristol, May 2006.

[70] M. Oliver, R. Webster, and J. Gerrard, "Geostatistics in physical geography. i: Theory," *Transactions of the Institute of British Geographers*, vol. 14, no. 3, pp. 259–269, 1989.

[71] ——, "Geostatistics in physical geography. ii: Applications," *Transactions of the Institute of British Geographers*, vol. 14, no. 3, pp. 270–286, 1989.

[72] O.Renaud, J.-L. Starck, and F. Murtagh, "Wavelet-based forecasting of short and long memory time series," University of Geneva, Tech. Rep., May 2002.

[73] H. M. Polchlopek and J. P. Noonan, "Wavelets, detection, estimation, and sparsity," *Digital Signal Processing*, vol. 7, pp. 28–36, January 1997.

[74] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, 2007.

[75] M. B. Priestley and M. T. Chao, "Non-parametric function fitting," *Journal of the Royal Statistical Society*, vol. 34, no. 3, pp. 385–392, 1972.

[76] R Development Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2004, iSBN 3-900051-00-3. [Online]. Available: http://www.R-project.org

[77] C. H. Reinsch, "Smoothing by spline functions," *Numerische Mathematik*, vol. 10, no. 3, pp. 947–950, October 1967.

[78] O. Rioul and M. Vetterli, "Wavelets and signal processing," *IEEE Signal Processing Magazine*, vol. 8, pp. 14–38, 1991.

[79] I. J. Schoenberg, "Spline functions and the problem of graduation," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 52, no. 4, pp. 947–950, October 1964.

[80] B. W. Silverman, "Wavelets in statistics: Beyond standard assumptions," *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, vol. 357, no. 1760, pp. 2459–2473, September 1999.

[81] C. M. Stein, "Estimation of the mean of a multivariate normal distribution," *The Annals of Statistics*, vol. 9, no. 6, pp. 1135–1151, November 1981.

[82] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," *Wavelet Applications in Signal and Image Processing III*, vol. 2569, pp. 68–79, 1995.

[83] ——, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Applied and Computational Harmonic Analysis*, vol. 3, pp. 186–200, March 1996.

[84] ——, "The lifting scheme: A construction of second generation wavelets," *SIAM Journal on Mathematical Analysis*, vol. 29, no. 2, pp. 511–546, March 1998.

[85] R. Wagner, S. Sarvotham, and R. Baraniuk, "A multiscale data representation for distributed sensor networks," *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05)*, vol. 4, pp. 549 – 552, March 2005.

[86] R. Wagner, S. Sarvotham, H. Choi, and R. Baraniuk, "Distributed multiscale data analysis and processing for sensor networks," *Rice University ECE Technical Report*, February 2005.

[87] A. T. Walden, E. J. McCoy, and D. B. Percival, "The effective bandwidth of a multitaper spectral estimator," *Biometrika*, vol. 82, no. 1, pp. 201–214, March 1995.

[88] M. Wand and M. Jones, *Kernel Smoothing*. Chapman & Hall/CRC, 1995.

[89] G. Watson, "Smooth regression analysis," *Sankhya*, no. 26, pp. 359–372, January 1969.

[90] B. Whitcher, "Wavelet-based estimation for seasonal long-memory processes," National Center for Atmospheric Research, Tech. Rep., March 2001.