Novel Lifting Scheme Constructions for Data Collected from Network Edges

Dingjia Cao

PhD

University of York

Mathematics

September 2024

Abstract

As an emerging research area, analysing functions that arise from network (graph) structures attracts researchers in both statistics and signal processing communities. While current literature is rich when data are collected from the graph vertex space, the data collected from graph edges call for new techniques, and in its turn reaches across many application fields, from traffic networks to neuroscience and hydrology. Wavelets are popular tools for understanding the behaviour of the underlying (edge) functions due to their computational efficiency and robust performance in the presence of discontinuities.

In this thesis, we propose three types of new algorithms that provide a multiscale approach developed through lifting scheme wavelet constructions for data collected from the network edges, useful for data compression and signal denoising. We thoroughly investigate the properties of the proposed algorithms through simulation studies. In addition, we analyse the impact on the method performance of different choices of key quantities, such as prediction/update weights and integrals of the scaling functions.

Finally, we propose non-decimated versions for all of the proposed methods, and these are shown to have a significant impact in the context of denoising problems posed in this thesis. We illustrate the advantages of our non-decimated lifting constructions on a simulated dataset previously introduced in the literature as well as on a new hydrological real dataset. The findings and the comparisons with existing results reported in the literature reinforce the superiority of our techniques as well as the wide-reach of our three method types when considering the extent to which data is available, e.g. full or partial information on edge lengths.

To my dad and my grandparents,
for their constant love at all times.

Contents

List of Figures

List of Tables

Acknowledgement

This thesis would not have been done without the help and support of many lovely people. Firstly, I would like to send my gratitude to my supervisor, Prof. Marina Knight. Thank you, Marina, for the support on academic and some chill chats throughout my PhD period. Your tips on (math) writing, thinking, as well as the suggestions on work-life balance will keep on helping me after my PhD study.

I would like to thank Prof. Wenyang Zhang and Prof. Degui Li for agreeing on being my Thesis Advisory Panel members. I would also like to thank Dr. Ben Powell for many useful chats in either the department or the pubs. Thanks to Dr. Yue Zhao, Dr. Jessica Hargreaves, and Dr. Ben Powell, for letting me do the graduate teaching job for them and trusting me. A big appreciation to everyone in the department of Mathematics, in particular to Prof. Ed Corrigan for many chats and encouragements during my thesis writing period, and to Prof. Kasia Rejzner for many memorable pub times.

I am extremely grateful for the accompany from every friend I meet in York, Ali, Andrew, Jenny, Laura, Vincenzo, Vasilis, Ambroise, Cords, Eva, Diego, David, Peiyun, Jade, Jack, Esther, Samantha, Guy, Emily, Ben G, Sam E, Nick, Ayan, Shefali, Nekhel, Anjali, Lewis, Nikos, Yujia, Simen, Beth, Peter, Simon, Berend, Jintao, Shashank, Shashaank, and Rutvij. Thanks to Diego and Eva for hosting me in London; to Andrew, Cords, Beth, Ambroise (many times), Lewis, and Rutvij for being my housemates.

Thanks to Arnon, Jamie M, Jorjie, Ben C, Vincenzo for many enjoyable badminton games, and of course to the state-level badminton player (and in many other sports) for the guidances.

There were many hard periods during the PhD (especially writing up time) and I always know where to go, the Golden Ball, the best pub in the world. Thanks to all of the staffs and every friend I met there. Thanks to Orion for the many drinks as well.

Big thanks to my friendship back in China, Haoyan, Yingyi, Cissie, thank you all for a lot of late night (British time) conversations.

In the end, a big thank to my family, in particular my dad Bin Cao and my grandma Genmei Feng, for their love and support at all time.

Declaration

This thesis is submitted to the University of York for the degree of Doctor of Philosophy in Mathematics. I declare that the work presented in this thesis is my own original independent research and has not been submitted for a degree at any other university or institution. All sources of information have been properly acknowledged through references.

Introduction

Network structures (or graph structures) have already attracted many mathematicians throughout history. The first milestone in the study of networks has been widely recognised as the proof of the Königsberg bridge problem by Euler, see Newman (2003). Network analysis allows us to understand the behaviour of complex systems, such as climate, as Kolaczyk and Csárdi (2014) put it '*an increasing tendency towards a systems-level perspective in the sciences, away from the reductionism that characterised much of the previous century*'. Some measurements or analysis methods for networks have a large range of applications. For example, Zhang et al. (2003) presented a traffic matrix estimation via an entropy-based method, Zhang et al. (2005) introduced a framework for network anomaly detection and network tomography, Popescul and Ungar (2003) and Taskar et al. (2003) analysed the link prediction problem in network science, Girvan and Newman (2002) investigated community structures in networks and introduced a method for detecting such structures. For a comprehensive introduction to all these topics and works, the reader can refer to Kolaczyk and Csárdi (2014). Recently, many statisticians have been interested in the (statistical) modelling of networks, for example, spatio-temporal modelling on networks, see Mahadevan (2010) and Knight et al. (2019), as well as Park et al. (2022) for a problem in a hydrological context.

Although many efforts have been made to understand the probabilistic aspects of networks, such as random graphs (Erdős et al.; 1960), there are still many interesting statistical research questions to tackle when considering data collected on networks. In general, statistical network analysis can be grouped into two main directions: (i) constructing statistics describing the network structure, e.g., subgraph density (Chang

et al.; 2022); and (ii) analysing functions recorded over the network, e.g., nonparametric regression on networks (Severn et al.; 2021). The second aspect can be subdivided into two areas, when the function is observed from a static network, or the function is observed through time, from a dynamic network. In this work, we are interested in the problem of denoising a function recorded on the edges of a static network (e.g., a time snapshot).

Most of the work in the current literature on 'graph wavelets' among signal processing community is exploring wavelet constructions for the vertex set, see for example, Shuman et al. (2016) and Stanković et al. (2020). In reality, data may naturally come from the edge set rather than from the vertex set, e.g. traffic flow data (Lakhina et al.; 2004), data from river networks (Cressie et al. (2006) and Park et al. (2022)), and fish species distribution (Buisson et al.; 2008). It is not trivial to apply the existing methods to such data, due to edge topology/geometry considerations, as opposed to the much simpler case when data have been collected over one-dimensional locations. Under these circumstances, constructing a multiscale method capable to operate on the edges of a network is desirable, in particular, wavelets ('little waves') can capture space-frequency (scale) localised information for many different underlying functions. To our knowledge, the literature that considers multiscale decompositions for functions defined on the edges of networks is very sparse. Cressie et al. (2006) discussed some statistical methods for spatial prediction of data collected from streams (edges) of a river network, such as the spatial moving average and kriging. Park et al. (2022) proposed an approach that uses the lifting-one-coefficient-at-one-time algorithm (LOCAAT) of Jansen et al. (2009) albeit the algorithm is designed for vertices while the data is recorded at the edges of a river network. The method from Park et al. (2022) does not discuss the stream domain from a geometric point of view and could lead to a mixture of vertex and edge topology. Moreover, their method employs river flow, another potentially noisy signal collected from edges, to construct the prediction weights, which could lead to an inaccurate prediction. Severn et al. (2021) introduced a regression curve estimation method for network data, which they described as manifold-valued data. Their estimator is obtained by solving a minimisation problem involving squared distances based on the graph Laplacian. Sim-

ilar to classic nonparametric regression methods, their approach relies on the choice of kernel and bandwith, and may struggle to estimate network functions with discontinuities. Therefore, designing a wavelet-based method with a theoretical framework that captures information at various edge-driven scales would be valuable. In our work, we will mainly build on the LOCAAT framework from Jansen et al. (2001, 2004, 2009). This has already been proven to be a powerful tool for statistical regression problems (also of interest in this thesis), see Nunes et al. (2006), Knight and Nason (2009) and Knight et al. (2017). Our main target is to construct new multiscale methodologies that are capable to denoise the data collected from edges corrupted by noise.

This thesis is organised as follows. In Chapter 1 we give a review of existing works, which is essential for our work. In particular, we give the necessary background to wavelet theory, including the first-generation (classic) and second-generation (via lifting scheme) wavelets, as well as their applications for signal denoising. Some concepts from graph theory will also be discussed, including the metrized graph from Kuchment (2003) and Baker and Faber (2006), which provide a framework that will allow us to construct multiscale function representations on a graph structure.

In Chapter 2 we propose a line graph lifting-based algorithm (LG-LOCAAT) for dealing with data collected from network edges, where we first carry out a line graph transform, and then use the LOCAAT algorithm (discussed in Chapter 1) on the new vertices of this transformed structure. Using function representations by means of metrized graphs, we also provide the theoretical support for our LG-LOCAAT transform, such as the integral choice and the scale notion. In addition, other wavelet constructions can be extended to the line graph space, such as the diffusion wavelets introduced by Coifman and Maggioni (2006). However, the performance of diffusion wavelets depends on the choice of diffusion operators as well as the underlying graph/manifold structure. For instance, if the graph contains denser connected subgraphs, the diffusion operator may create basis functions that fail to be localised in the observation domain. In contrast, our method can overcome this limitation by utilising the lifting scheme, constructed to generate a set of biorthogonal basis functions.

In Chapter 3 we introduce a new multiscale construction for edge data, which we refer to as E-LOCAAT, for which the lifting scheme is performed on the *original* metrized graph domain instead of executing a line graph transform beforehand. The line graph transformation is a limitation caused by the LG-LOCAAT construction but avoided by the E-LOCAAT proposal, see Section 2.3.4 for more the details. An interesting Haar-like variant will be generated by E-LOCAAT with a self-similarity constraint for the scaling functions, which shows excellent denoising performance when the underlying function is piecewise constant.

In Chapter 4 we introduce a new algorithm for dealing with edge data via the graph Laplacian, which we refer to as the Laplacian-LOCAAT algorithm. This construction is motivated by works from the signal processing community, in which graph Laplacians (for the vertex set) are used for constructing Fourier and wavelet bases, see for example, Hammond et al. (2013), Shuman et al. (2013), and Ortega et al. (2018); and for diffusion wavelets introduced by Coifman and Maggioni (2006). To fit into the edge-based data, we make use of the edge Laplacian, introduced into applications by Zelazo et al. (2007) and Zelazo and Mesbahi (2010). The Laplacian-LOCAAT provides a faster algorithm than LG-LOCAAT and E-LOCAAT, while also enabling different potential future improvements, see Chapters 4 and 6 for more details.

Comprehensive simulation studies and assessment of the proposed methodologies will be presented for the new algorithms in Chapters 2, 3, and 4. In Chapter 5 we introduce non-decimated lifting scheme variants for the best performing algorithms in the simulation studies for previous chapters, followed by a real data analysis. Comparisons with recent works that treat hydrological data denoising on (simulated) river flow data illustrate the superior behaviour of our proposed methodologies, while the analysis of dissolved oxygen in river water, as an indicator of water quality, highlights the wide span of data types (with complete versus incomplete information) that our portfolio of algorithms can successfully deal with.

Chapter 1

Literature Review

In this chapter, we introduce the mathematical and statistical tools which we will use throughout the thesis. In Section 1.1 we will briefly discuss some basic functional analysis terminologies. In Section 1.2 we give a detailed introduction of the first generation wavelets (or classic wavelets) and the second generation wavelets, including the so-called Multiresolution Analysis (MRA) framework, followed by the description of the lifting scheme (Sweldens; 1998) in Section 1.3. Then in Section 1.4, we describe some topics in graph theory which are essential for our analysis. Some related works and terminologies of network data will be discussed. In the rest of this chapter, Section 1.5 describes the **L**ifting **O**ne **C**oefficient **A**t **A** **T**ime (LOCAAT) framework constructed upon the lifting scheme. The description of nonparametric regression, along with the wavelet thresholding methods, will be introduced in Section 1.6.

1.1 Prelude: Concepts for Hilbert Spaces and Basis Representations

1.1.1 Hilbert Spaces

Hilbert spaces are functional spaces which contain the classes of functions with 'the richest geometrical structure' (see Young (1988)). Intuitively, we can consider Hilbert

spaces as generalisations of Euclidean spaces (on functions). **Inner products** are used to define quantities, such as distances, in Hilbert spaces. We denote $\langle \cdot, \cdot \rangle$ as the inner product, each inner product then derives a **norm**, denoted by $\|\cdot\|$. Then a Hilbert space $\mathcal{H}$ can be considered as a complete metric space with the metric induced by its inner product.

Throughout our study, we are mainly concerned with Hilbert spaces for some specific functions or sequences, such as those are squared integrable (summable). For the function version, suppose we have two functions $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$. We say that f belongs to the space of squared integrable functions on $\mathbb{R}$, namely $f \in L^2(\mathbb{R})$, if it satisfies

$$\int_{-\infty}^{\infty} |f(x)|^2 dx < \infty.$$

Then the inner product of f and g is defined as

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(x)g(x)dx.$$

The (induced) L^2-norm of the function f is defined as

$$\|f\|_{L^2} = \langle f, f \rangle^{\frac{1}{2}} = \left(\int_{-\infty}^{\infty} |f(x)|^2 dx \right)^{\frac{1}{2}}.$$

For the sequence version, suppose now we have a sequence $\underline{x} = \{x_n\}_{n \in \mathbb{Z}}$, we say $\underline{x} \in l^2(\mathbb{Z})$ **the space of squared summable sequences** on $\mathbb{Z}$ if it satisfies

$$\sum_{n \in \mathbb{Z}} |x_n|^2 < \infty.$$

Suppose we have another sequence $\underline{y} = \{y_n\}_{n \in \mathbb{Z}}$, then the inner product of $\underline{x}$ and $\underline{y}$ is

$$\langle \underline{x}, \underline{y} \rangle = \sum_{n \in \mathbb{Z}} x_n y_n.$$

The induced norm of the sequence $\underline{x}$ is

$$\|\underline{x}\|_{l^2} = \langle \underline{x}, \underline{x} \rangle^{\frac{1}{2}} = \left(\sum_{n \in \mathbb{Z}} |x_n|^2 \right)^{\frac{1}{2}}.$$

1.1.2 Orthogonality and Bases

The basis of a Hilbert space $\mathcal{H}$ is defined as a set of functions $\{f_k\}_{k\in\mathcal{B}}$, such that for any $f \in \mathcal{H}$, there exist a linear combination such that

$$f = \sum_{k\in\mathcal{B}} \alpha_k f_k,$$

for some constants $\{\alpha_k\}_{k\in\mathcal{B}}$, and $\sum_{k\in\mathcal{B}} \alpha_k f_k = 0$ if and only if $\alpha_k = 0$ for any $k \in \mathcal{B}$. This property can be denoted as $\overline{\text{span}}\{f_k | k \in \mathcal{B}\} = \mathcal{H}$. Here $\overline{A}$ denotes the topological closure of a set A.

An important property is **orthogonality**, and we say that two functions $f, g \in \mathcal{H}$ are orthogonal if and only if their inner product satisfies $\langle f, g\rangle = 0$.

A basis where any two different basis functions are orthogonal to each other is called an orthogonal basis, and a set of functions $\{f_k\}_{k\in\mathcal{B}} \subseteq \mathcal{H}$ forms an **orthonormal basis** if and only if $\langle f_i, f_j\rangle = \delta_{ij}$ for all $i, j \in \mathcal{B}$ and $\overline{\text{span}}\{f_k | k \in \mathcal{B}\} = \mathcal{H}$, where

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

is the Kronecker delta, which implies the normalisation (each element has unit norm) and linear independence of the family $\{f_k\}_{k\in\mathcal{B}}$.

Example: Fourier series

Suppose we have a set of functions $\{f_n(x)\}_{n\in\mathbb{Z}}$, where

$$f_n(x) = \frac{1}{\sqrt{2\pi}} e^{inx}, \quad \text{for} \quad x \in (-\pi, \pi],$$

where we can see the oscillation in these functions according to the well-known *Euler's formula*: $e^{inx} = \cos(nx) + i\sin(nx)$. The functions in this family satisfy both conditions $\langle e_m, e_n\rangle = \delta_{mn}$ and $\overline{\text{span}}\{f_n | n \in \mathbb{Z}\} = L^2((-\pi, \pi])$, hence, for any function $f \in L^2((-\pi, \pi])$, we can rewrite it approximately as

$$f(x) = \sum_{n\in\mathbb{Z}} \langle f, f_n\rangle f_n(x), \tag{1.1.1}$$

where $\widehat{f}(n) = \langle f, f_n \rangle = \int_{-\pi}^{\pi} \frac{1}{\sqrt{2\pi}} f(x) e^{inx} dx$ denotes the Fourier transform of function f. Equation (1.1.1) is commonly known as the *Fourier series* expansion of f.

Fourier analysis is widely used in many areas, from physics to neuroscience, and promoted a mathematical branch called harmonic analysis, see Pereyra and Ward (2012). Moreover, a new philosophy was motivated by orthonormal basis expansion, in analysing functions or signals in a different domain (frequency domain) rather than in observation domain (time domain). For example, in equation (1.1.1), the amplitude and frequency can be indicated by the magnitude of $\langle f, f_n \rangle$ for each n.

In practice, Fourier bases have many desirable properties, for example, they can be computed easily and can effectively extract information from smooth functions. But for these functions with discontinuities, Fourier series usually lead to poor convergence results of neighbouring regions near discontinuities, the so-called **Gibbs phenomenon**, see Pereyra and Ward (2012). Figure 1.1 shows a visualisation of the Gibbs phenomenon, where we can clearly see that the Fourier analysis has difficulty dealing with the discontinuities in the function, even as the number of basis functions increases.

In a nutshell, as classical Fourier analysis is not capable to deal with some local information, Daubechies (1992) refers to this as: 'high frequency bursts cannot be read off easily from $\langle f, f_n \rangle$', which further motivated the development of functional analysis in other domains, such as polynomial and wavelet bases.

Example: Monomial basis

Consider a class of polynomial functions f which have the form

$$f(x) = \sum_{i=0}^{\infty} \alpha_i x^i,$$

where α_i is the coefficient of i-th power polynomial. For example, if f is a cubic function, then $\alpha_3 \neq 0$ and $\alpha_i = 0$ for $i \geq 4$. It is easy to check that the set of polynomials $\{x^0, x^1, x^2, ...\}$ spans the space of all polynomial functions, but orthogonality does not hold. The set $\{x^0, x^1, x^2, ...\}$ is known as a (non-orthogonal) monomial basis.

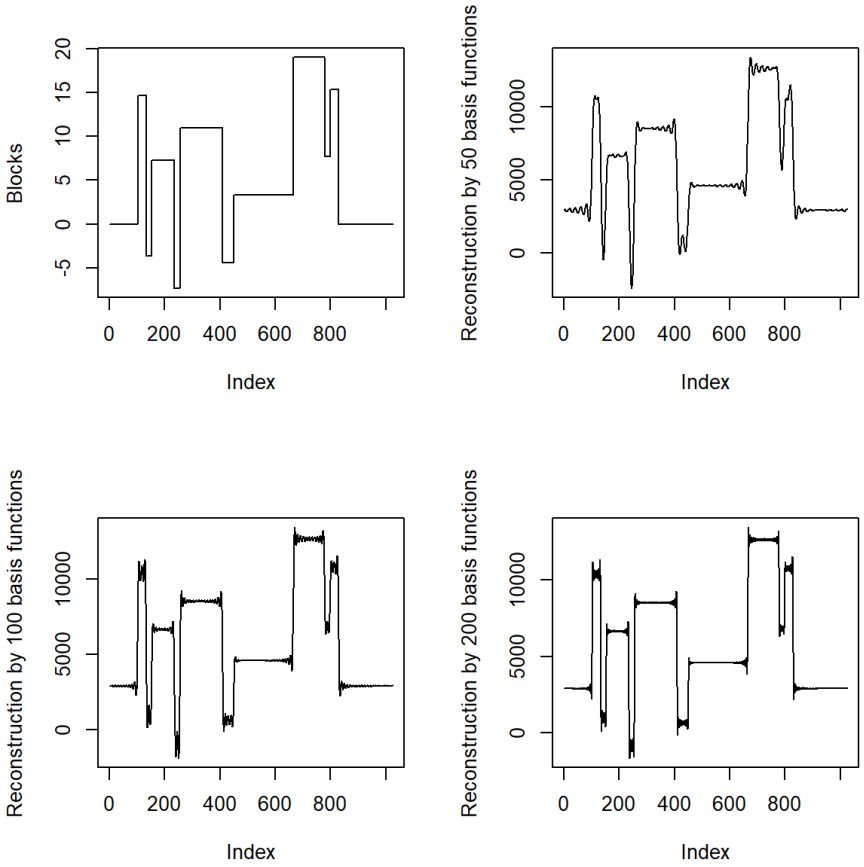

Figure 1.1: Visualisation for the Gibbs phenomenon. **Top Left:** The Blocks function. **Top Right:** The reconstruction of the Blocks function via 50 Fourier basis functions. **Bottom Left:** The reconstruction of the Blocks function via 100 Fourier basis functions. **Bottom Right:** The reconstruction of the Blocks function via 200 Fourier basis functions.

1.2 Wavelets

Since the methodology in this work relies heavily on wavelets, in this section we provide the essential details of wavelet theory. As of now, one may still have the question: *what are wavelets?* Similar to Fourier analysis, wavelet analysis is a class of methods that relies on (wavelet) basis expansion in functional spaces. Unlike Fourier bases, wavelet bases are usually time-scale (frequency) localised rather than only frequency localised.

A mother wavelet (or wavelet function), $\psi(x) \in L^2(\mathbb{R})$, is a function that can generate a basis by using dilations and translations. An element can be expressed as

$\{\psi_{a,b}\}_{a\in\mathbb{R}\setminus\{0\},b\in\mathbb{R}}$, such that

$$\psi_{a,b}(x) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{x-b}{a}\right), \tag{1.2.1}$$

where the dilation parameter a represents the scale of the wavelet, which also ensures a constant norm in L^2 such that $\|\psi_{a,b}\|_{L^2} = \|\psi\|_{L^2}$, while the translation parameter b indicates the location of the wavelet. The continuous wavelet transform (CWT) for a function $f \in L^2(\mathbb{R})$ can be defined as

$$\mathbf{CWT}_f(a,b) = \langle f, \psi_{a,b}\rangle_{L^2} = \frac{1}{\sqrt{|a|}}\int_{-\infty}^{\infty} f(x)\psi\left(\frac{x-b}{a}\right)dx. \tag{1.2.2}$$

Then the function f can be written as the wavelet representation by the Caldéron reproducing formula (Calderón; 1964), such that

$$f(x) = C_\psi^{-1}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{1}{a^2}\mathbf{CWT}_f(a,b)\,\psi_{a,b}(x)\,da\,db, \tag{1.2.3}$$

in which the constant is obtained by $C_\psi = \int_{-\infty}^{\infty}\frac{|\widehat{\psi}(\omega)|^2}{|\omega|}d\omega$, where $\widehat{\psi}$ is the function obtained by taking Fourier transform of ψ, see Section 1.1.2. To ensure the existence of the inverse transform, this constant has to satisfy the *admissibility condition*, such that

$$C_\psi = \int_{-\infty}^{\infty}\frac{|\widehat{\psi}(\omega)|^2}{|\omega|}d\omega < \infty. \tag{1.2.4}$$

Daubechies (1992) shows that an equivalent form for the condition (1.2.4) is

$$\int_{-\infty}^{\infty}\psi(x)dx = 0,$$

which demonstrates the oscillation of wavelets. This oscillation, along with the compact support of wavelets (the Shannon wavelet is one of the exceptions, but it decays to zero fast), brings the name 'wavelets', see for example Figure 1.2.

For a wavelet function ψ, if we have

$$\int_{-\infty}^{\infty}x^l\psi(x)dx = 0,$$

for $l = 0, ..., r-1$, then we say this wavelet has r vanishing moment. Wavelet functions with higher vanishing moments are able to capture smoother patterns. For example,

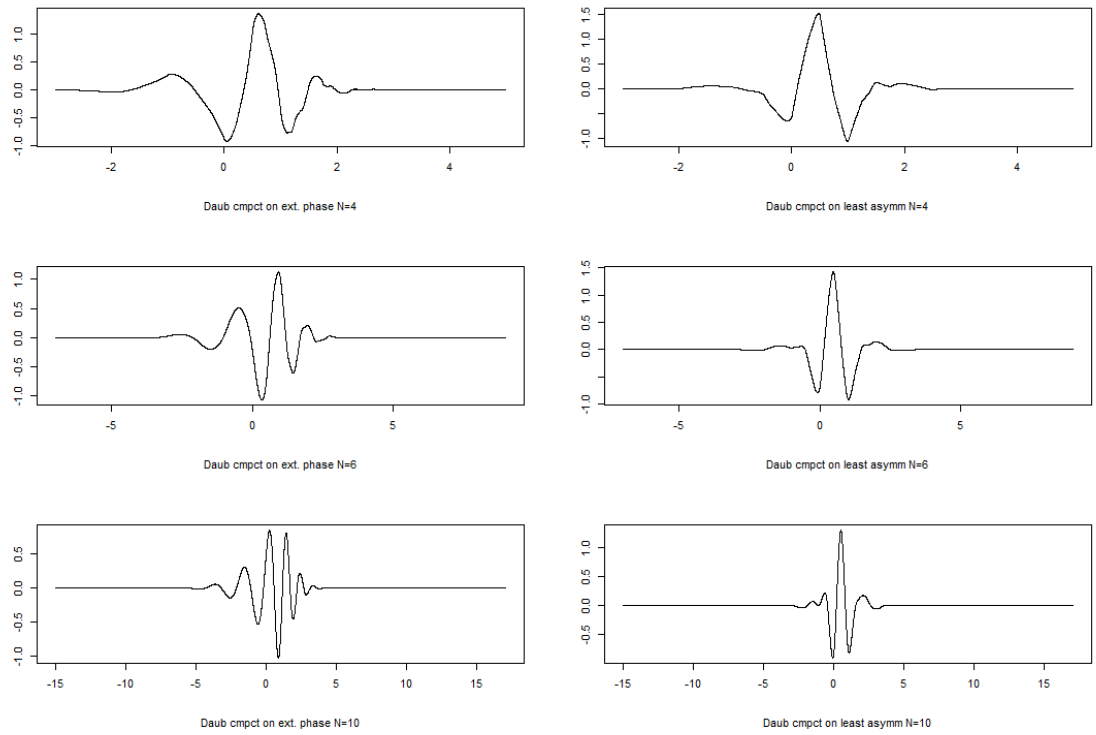

Figure 1.2: Example of wavelets. **Left:** Extremal Phase Daubechies wavelets. **Right:** Least Asymmetric Daubechies wavelets. **Top:** Wavelets with 4 vanishing moments. **Middle:** Wavelets with 6 vanishing moments. **Bottom:** Wavelets with 10 vanishing moments.

suppose that we have a r-polynomial function $f(x) = a_r x^r + a_{r-1}x^{r-1} + \cdots + a_1 x + a_0$, where $a_i \in \mathbb{R}$, for $i = 0, ..., r$. If we have a wavelet $\psi^{(r+1)}$ with $(r+1)$ vanishing moments, then the CWT will transform f into a zero function $\mathbf{CWT}_f(a, b)$, which is a desirable property in many applications.

1.2.1 Discretisation of the CWT

Recall in equation (1.2.2) that taking continuous wavelet transform for a function f results into a function $\mathbf{CWT}_f(a, b)$ that depends on two parameters, $a \in \mathbb{R}\backslash\{0\}$ and $b \in \mathbb{R}$, making the transform redundant. A discretisation for the CWT can be carried out to reduce this redundancy. The discretisation involves choosing discrete values for a, b to reduce the number of wavelet functions $\psi_{a,b}$ and is referred to as *critical sampling*. The choice is not unique, but the most used is to let $a = 2^{-j}$ and $b = 2^{-j}\, k$, where j, $k \in \mathbb{Z}$,

hence we can rewrite equation (1.2.1) into the dyadic decimated form such that

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k). \tag{1.2.5}$$

In addition, non-decimated wavelets can be constructed by letting $a = 2^{-j}$ and $b = k$, for some j, $k \in \mathbb{Z}$, which yields

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j(x - k)).$$

The non-decimated wavelet family provides wavelets at every location $k \in \mathbb{Z}$ instead of a dyadic decimation $2^{-j}\,k$, see Section 1.2.6 for more details.

Hence, a discrete version of the function expansion form for $f \in L^2(\mathbb{R})$ in equation (1.2.3) can be represented by an orthonormal decimated wavelet basis $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ (in equation (1.2.5)), such that

$$f(x) = \sum_{k\in\mathbb{Z}}\sum_{j\in\mathbb{Z}} \langle f, \psi_{j,k}\rangle_{L^2}\, \psi_{j,k}(x),$$

and due to the orthogonality of wavelets, we have

$$d_{j,k} := \langle f, \psi_{j,k}\rangle_{L^2} = \int_{-\infty}^{\infty} f(x)\psi_{j,k}(x)dx,$$

where $\{d_{j,k}\}_{j,k\in\mathbb{Z}}$ is known as the set of wavelet (or detail) coefficients.

1.2.2 Multiresolution Analysis (MRA)

Now let us introduce the important framework that brings wavelet analysis into various applications, known as ***multir****esolution* ***a****nalysis* (MRA). The (orthogonal) multiresolution analysis (Mallat; 1989b,a) is defined as a nested sequence of closed subspaces $\{V_j\}_{j\in\mathbb{Z}}$ in $L^2(\mathbb{R})$, which satisfy that

$$\cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots . \tag{1}$$

$$\bigcap_{j\in\mathbb{Z}} V_j = \{0\} \tag{2}$$

$$\overline{\bigcup_{j\in\mathbb{Z}} V_j} = L^2(\mathbb{R}). \tag{3}$$

$$f(2^j x) \in V_j \quad \text{if and only if} \quad f(x) \in V_0. \tag{4}$$

$$f(x-k) \in V_0 \quad \text{if and only if} \quad f(x) \in V_0. \tag{5}$$

There exists a scaling function $\varphi \in V_0$ with unit integral value $\int_{-\infty}^{\infty} \varphi(x)dx = 1$,

such that $\{\varphi_{0,k}\}_{k\in\mathbb{Z}}$ is an orthonormal basis in V_0, where $\varphi_{0,k}(x) = \varphi(x-k)$. (6)

Conditions (2) and (3) show that the subspaces have a trivial intersection and a dense union in $L^2(\mathbb{R})$. Condition (4) indicates the self-similarity of any two subspaces. The difference between the 'resolution levels' can be observed from different choices of j. For example, from j to $j+1$, finer information is captured on a higher scale 2^{j+1} rather than 2^j. As $V_0 \subset V_1$ and the fact that $\{\varphi_{1,k}\}_{k\in\mathbb{Z}}$ is a orthonormal basis in V_1, then any function $\varphi \in V_0$ can be represented as a linear combination of the functions of V_1. Hence, we have

$$\varphi(x) = \sum_{k\in\mathbb{Z}} h_k \sqrt{2}\varphi(2x-k). \tag{1.2.6}$$

This is called the *dilation* or *refinement equation*. The set of coefficients $\{h_k\}_{k\in\mathbb{Z}}$ is referred to as a *low-pass filter*. As its name suggests, this filter will preserve the low frequency components of a signal and average out the high frequency components.

Daubechies (1992) defined an operator Proj, such that

$$\text{Proj}_j f = \sum_{k\in\mathbb{Z}} \langle f, \varphi_{j,k}\rangle_{L^2}\, \varphi_{j,k}(x), \tag{1.2.7}$$

is the projection of the function $f \in L^2(\mathbb{R})$ on the subspace V_j. Daubechies (1992) also shows the basic idea of MRA as the following theorem.

Theorem 1. If subspaces $\{V_j\}_{j\in\mathbb{Z}}$ with a scaling function $\varphi(x)$ form a multiresolution analysis, then there exists an orthogonal wavelet basis $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$, of $L^2(\mathbb{R})$, which inherits the scaling relations from φ, so, for every $f \in L^2(\mathbb{R})$ and $j \in \mathbb{Z}$,

$$\text{Proj}_{j+1} f = \text{Proj}_j f + \sum_{k\in\mathbb{Z}} \langle f, \psi_{j,k}\rangle_{L^2}\, \psi_{j,k}, \tag{1.2.8}$$

where Proj_{j+1} and Proj_j are projections on V_{j+1} and V_j, respectively.

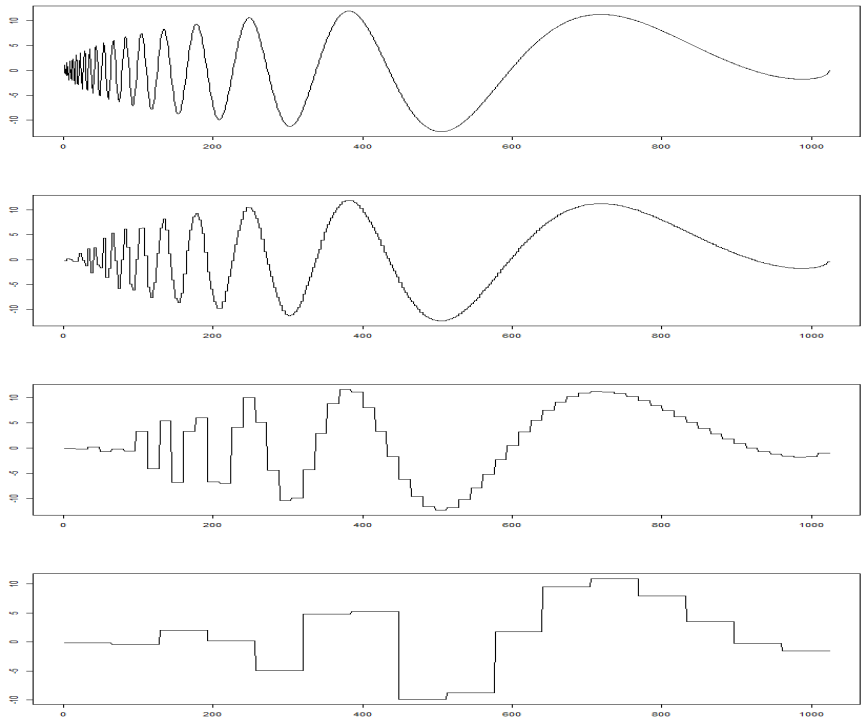

Figure 1.3: Doppler function and the approximations of it into the spaces V_j. **From top to bottom:** Doppler function (V_{10}); approximation of Doppler function in V_8; approximation of Doppler function in V_6; and approximation of Doppler function in V_4.

Figure 1.3 illustrates projections for Doppler function in different spaces V_j using the Haar wavelet. An interpretation of equation (1.2.8) is that the wavelets $\{\psi_{j,k}\}_{k\in\mathbb{Z}}$ at level j can be considered as the information 'loss' when zooming out from Proj_{j+1} to a coarser version Proj_j. Let W_j be the detail space spanned by $\{\psi_{j,k}\}_{k\in\mathbb{Z}}$ for a certain $j \in \mathbb{Z}$, which is the *orthogonal complement* of V_j in V_{j+1} such that $W_j = \{f \in V_{j+1} : \forall g \in V_j,\ \langle f, g\rangle = 0\}$. Then we have

$$V_j = V_{j-1} \oplus W_{j-1}, \tag{1.2.9}$$

where $j \in \mathbb{Z}$ and the symbol '$\oplus$' denotes the *direct sum*, W_j is called the detail space at level-j. If the finest scale is J, then for $i < j \leq J$, we have

$$V_j = V_i \oplus \bigoplus_{n=i}^{j-1} W_n. \tag{1.2.10}$$

Recall the conditions (2) and (3) for the MRA, hence we have

$$L^2(\mathbb{R}) = \bigoplus_{j\in\mathbb{Z}} W_j,$$

which means that a function in $L^2(\mathbb{R})$ can be characterised by the detail spaces. Notice that W_j inherit the scaling property from V_{j+1} (Daubechies; 1992), hence we have the following relations

$$f(x) \in W_0 \iff f(2^j\, x) \in W_j, \tag{1.2.11}$$

$$f(x) \in W_0 \iff f(x-k) \in W_0, \tag{1.2.12}$$

for $j, k \in \mathbb{Z}$. Due to the self-similarity of the detail spaces, the mother wavelets $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ can be obtained by constructing an orthonormal basis ψ for W_0. A possible construction for such a basis is that

$$\psi(x) = \sqrt{2}\sum_{k\in\mathbb{Z}} (-1)^k h_{1-k}\, \varphi(2x-k). \tag{1.2.13}$$

We can see that equation (1.2.13) follows the same thought with the refinement equation (1.2.6) for scaling functions. Let $g_k = (-1)^k h_{1-k}$, then the set of coefficients $\{g_k\}_{k\in\mathbb{Z}}$ is referred to as the high-pass filter, which preserves high frequency components of the signal. The sets $\{h_k\}_{k\in\mathbb{Z}}$ and $\{g_k\}_{k\in\mathbb{Z}}$ are together called *quadrature mirror filters*. Recall that for any function $\varphi(x) \in V_0$, we have $\sqrt{2}\varphi(2x-k) \in V_1$, and $W_0 \subset V_1$, then the refinement can be written as

$$\varphi_{j,k}(x) = \sum_{l\in\mathbb{Z}} h_{l-2k}\varphi_{j+1,l}(x), \tag{1.2.14}$$

$$\psi_{j,k}(x) = \sum_{l\in\mathbb{Z}} g_{l-2k}\varphi_{j+1,k}(x). \tag{1.2.15}$$

By equations (1.2.14) and (1.2.15), the filter design can be expressed as

$$h_{l-2k} = \langle \varphi_{j,k}, \varphi_{j+1,l} \rangle, \tag{1.2.16}$$

$$g_{l-2k} = \langle \psi_{j,k}, \varphi_{j+1,k} \rangle, \tag{1.2.17}$$

for all j, k, l.

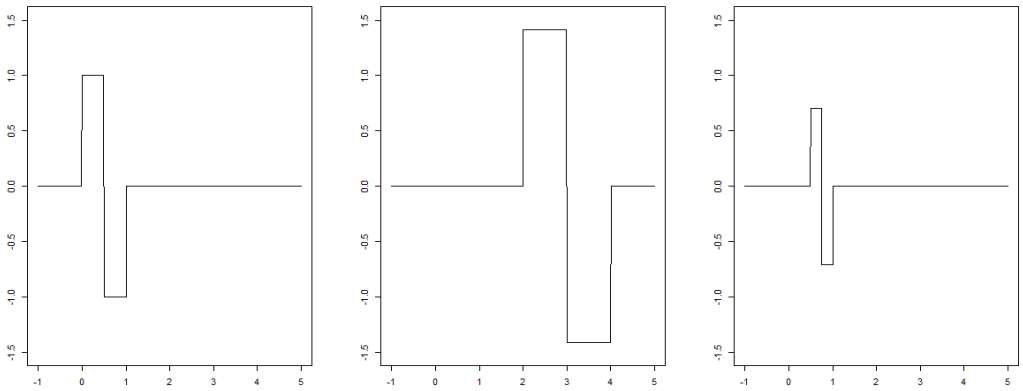

Figure 1.4: **Left:** Haar mother wavelet $\psi_{1,0}^{Haar}$. **Middle:** Haar wavelet $\psi_{2,2}^{Haar}$. **Right:** Haar wavelet $\psi_{\frac{1}{2},\frac{1}{2}}^{Haar}$.

Example: Haar Wavelet

The Haar wavelet, introduced by Alfréd Haar (Haar; 1910) (even before the appearance of the terminology 'wavelet'), is the best example to start the wavelet journey because of its simplicity. The Haar mother wavelet is defined as

$$\psi^{Haar}(x) = \chi_{[0,\frac{1}{2})}(x) - \chi_{[\frac{1}{2},1]}(x) = \begin{cases} 1, & 0 \leq x < \frac{1}{2}; \\ -1, & \frac{1}{2} \leq x \leq 1; \\ 0, & \text{otherwise.} \end{cases}$$

The family of Haar wavelet functions obtained by dilation and translation are

$$\psi_{a,b}^{Haar}(x) = \chi_{[b,\frac{a}{2}+b)}(x) - \chi_{[\frac{a}{2}+b,a+b]}(x) = \begin{cases} \frac{1}{\sqrt{a}}, & b \leq x < \frac{a}{2} + b; \\ -\frac{1}{\sqrt{a}}, & \frac{a}{2} + b \leq x \leq a + b; \\ 0, & \text{otherwise,} \end{cases}$$

where $a \in \mathbb{R}_{>0}$ and $b \in \mathbb{R}$. Figure 1.4 illustrates the Haar wavelets with different dilation and translation.

1.2.3 Function Expansion using Wavelets

Recall that for a scale j, the detail space $W_j \subset V_{j+1}$, and $W_j \perp V_{j'}$ for any $j' \leq j$. Now let us consider a certain level j_0, we have that $V_{j_0} \perp W_j$ and $W_j \perp W_{j'}$, for any $j > j' \geq j_0$. The set $\{\varphi_{j_0,k}\}_{k\in\mathbb{Z}} \bigcup \{\psi_{j,k}\}_{j\geq j_0,k\in\mathbb{Z}}$ is an orthonormal basis for $L^2(\mathbb{R})$, hence, along with equation (1.2.8), a function $f \in L^2(\mathbb{R})$ can be written as

$$f(x) = \sum_{k\in\mathbb{Z}} \langle f, \varphi_{j_0,k}\rangle_{L^2} \varphi_{j_0,k}(x) + \sum_{j\geq j_0} \sum_{k\in\mathbb{Z}} \langle f, \psi_{j,k}\rangle_{L^2} \psi_{j,k}(x). \tag{1.2.18}$$

The first component tells us the information of the function at a coarse level j_0, and the second component consists of the finer level information when moving from Proj_{j+1} to Proj_j, for all $j \geq j_0$, so on and so forth. The limit form of the function expansion in equation (1.2.18) can be written as

$$f(x) = \sum_{j\in\mathbb{Z}} \sum_{k\in\mathbb{Z}} \langle f, \psi_{j,k}\rangle_{L^2} \psi_{j,k}(x).$$

This indicates that any function $f \in L^2(\mathbb{R})$ can be characterised by the set of wavelet functions $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ and the associated wavelet coefficients $d_{j,k} = \langle f, \psi_{j,k}\rangle_{L^2}$. In practice, computing the magnitude of the wavelet coefficients is of interest to us. The approach is called the *discrete wavelet transform*, described as follows.

1.2.4 Discrete Wavelet Transform (DWT)

The discrete wavelet transform provides a fast recursive computation to obtain the scaling and wavelet coefficients, without actually calculating the inner products $\langle f, \varphi_{j,k}\rangle$ and $\langle f, \psi_{j,k}\rangle$. This framework is firstly introduced by Mallat (1989b,a), referred to as *Mallat's Pyramid algorithm*.

Suppose we have a orthonormal basis $\{\varphi_{j,k}\}_{j,k\in\mathbb{Z}}$ with compact supports, which satisfies that $\int_{-\infty}^{\infty} \varphi_{j,k}(x)dx = 1$, for all j, k. Recall that the supports of scaling functions are squeezed from resolution level j to $j+1$, Daubechies (1992) show the fact that $\lim_{j\to\infty} \varphi_{j,k}(x) = \delta(x-k)$, where δ is the Dirac delta. Hence, by selecting a suitable level j, we can start with $\langle f, \varphi_{j,k}\rangle \approx f(k)$, for $k \in \mathbb{Z}$. According to equation (1.2.8), this

indicates that the information of the function can be (mostly) characterised by Proj_j, thus, we have $\langle f, \psi_{j,k} \rangle \approx 0$. This means the function expansion can be written as

$$f(x) \approx \sum_{k \in \mathbb{Z}} c_{j,k} \varphi_{j,k}(x), \tag{1.2.19}$$

for a suitable level j. Hence, a possible construction is to start with the sequence $\{c_{j,k} = f(k)\}_k$, and fully utilise the refinement equations (1.2.14) and (1.2.15). For example, we have that

$$\begin{aligned} c_{j-1,k} &= \langle f, \varphi_{j-1,k} \rangle \\ &= \sum_{l \in \mathbb{Z}} h_{l-2k} \langle f, \varphi_{j,k} \rangle \\ &= \sum_{l \in \mathbb{Z}} h_{l-2k} c_{j,k}. \end{aligned} \tag{1.2.20}$$

Similarly, we have

$$d_{j-1,k} = \sum_{l \in \mathbb{Z}} g_{l-2k} c_{j,k}. \tag{1.2.21}$$

Equations (1.2.20) and (1.2.21) are known as the discrete wavelet transform. A sequence of wavelet coefficients will be obtained if we perform these two equations recursively.

The discrete wavelet transform is also invertible, the associated inverse can be done by recursively applying

$$c_{j,k} = \sum_{l \in \mathbb{Z}} h_{k-2l} c_{j,l} + \sum_{l \in \mathbb{Z}} g_{k-2l} d_{j,l},$$

see Mallat (1989b) for more details. Figure 1.5 illustrates the discrete wavelet transform for Blocks and Doppler functions. The two bottom plots illustrate different degrees of sparsity of the wavelet coefficients associated with these two functions, with most of the wavelet coefficients being zero (or almost zero) for Blocks, hence indicating that the function can be well represented by only a small set of basis functions, see equation (1.2.18). For functions with discontinuities, this property helps avoid the Gibbs phenomenon associated with the Fourier transform. Compared to the wavelet coefficients of the Doppler function, note the Haar wavelet coefficients of the Blocks function are much sparser, indicating a more efficient representation for Blocks than for Doppler, since the function features can be captured using fewer basis functions.

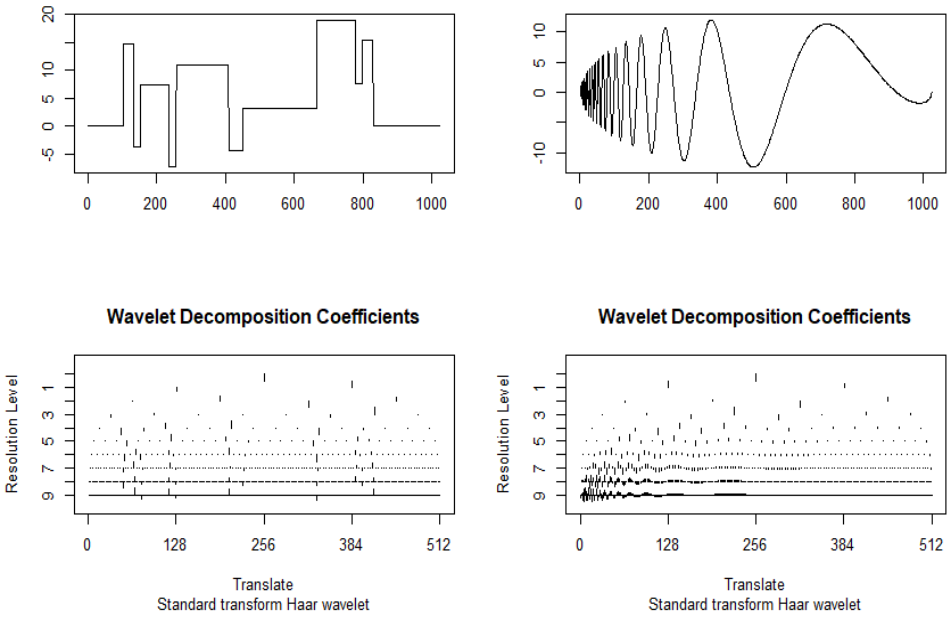

Figure 1.5: An illustration of DWT. **Top left:** Blocks function. **Top right:** Doppler function. **Bottom left:** Wavelet coefficients through different resolution levels obtained by Haar wavelet transform for Blocks function. **Bottom right:** Wavelet coefficients at different resolution levels obtained by Haar wavelet transform for Doppler function.

DWT for Discrete Data

All of the above discussions are for an underlying continuous function $f \in L^2(\mathbb{R})$. However, in practice (at least in statistical applications), the data observed are generally discrete, located on equally-spaced points $\{x_i\}_{i=1}^n$ of length $n = 2^J$ for some $J \in \mathbb{N}$. For the i-th point, there is a function value $f(x_i)$ associated with it. A convenient setup for the starting scaling coeffcients is

$$c_{J,i} = f(x_i). \tag{1.2.22}$$

However, this choice is not accurate since the scaling functions $\{\varphi_{J,i}\}_{i=1}^n$ are not exactly Dirac deltas. As a result, equation (1.2.22) can lead to some error for the approximation in equation (1.2.19), which is referred to as the '*wavelet crime*' in Strang and Nguyen (1996). They suggest to perform a pre-filter for the original data to guarantee equation (1.2.19), for more details the reader can refer to Strang and Nguyen (1996).

Once having chosen the scaling coefficients $\{c_{J,i}\}_{i=1}^{2^J}$, we can compute the wavelet coefficients $\underline{d}_{J-1} = (d_{J-1,1}, ..., d_{J-1,2^J})^T$ and scaling coefficients $\underline{c}_{J-1} = (c_{J-1,1}, ..., c_{J-1,2^J})^T$ at level $(J-1)$ by equations (1.2.20) and (1.2.21). Denote $\underline{f} = (f(x_1), ..., f(x_{2^J}))^T$, and after iterating the procedure, we can obtain the vector

$$\mathbf{DWT}(\underline{f}) = (\underline{c}_{j_0}, \underline{d}_{j_0}, ..., \underline{c}_{J-1}, \underline{d}_{J-1})^T,$$

where j_0 is a chosen stopping time. Since for any $j \in \mathbb{Z}$, and $j_0 \leq j \leq J-1$, the lengths of the associated vectors $\underline{c}_j$ and $\underline{d}_j$ are 2^j, it is easy to check that $\mathbf{DWT}(\underline{f})$ produces an output of the same length as the input (which is 2^J).

The discrete wavelet transform can be represented in a matrix form, such that

$$\mathbf{DWT}(\underline{f}) = R\underline{f},$$

where R is a $2^J \times 2^J$ matrix that characterises the discrete wavelet transform. Recall that the discrete wavelet transform is an orthogonal transform, thus, the matrix R is an orthogonal matrix, such that $R^T R = \mathbf{I}_{2^J}$, where $\mathbf{I}_{2^J}$ is an identity matrix of dimension $2^J \times 2^J$. Hence, we have $R^{-1} = R^T$, which means the inverse transform in matrix form can be obtained by taking the transpose of the matrix R.

1.2.5 Biorthogonal Wavelets

Orthogonal wavelets have many desirable advantages. For example, the wavelet domain preserves the function energy (in terms of L^2 norm). However, (real-valued) orthogonal wavelets with compact support will lead to the absence of symmetry, with the only exception being the Haar wavelet, see Daubechies (1992). This asymmetry can lead to some drawbacks in applications, especially for those involving visualisation, such as image analysis, see Daubechies (1992). Motivated by these application areas, Cohen et al. (1992) introduced a wavelet construction with a relaxation of the orthogonality, which is referred as *biorthogonal wavelet construction*.

Instead of a system of two filters $\{h_k\}_{k\in\mathbb{Z}}$ and $\{g_k\}_{k\in\mathbb{Z}}$, biorthogonal constructions involve four different filters: dual filters $\{\tilde{h}_k\}_{k\in\mathbb{Z}}$ and $\{\tilde{g}_k\}_{k\in\mathbb{Z}}$, and primal filters $\{h_k\}_{k\in\mathbb{Z}}$ and

$\{g_k\}_{k\in\mathbb{Z}}$. The refinement relations for biorthogonal wavelets can be found in Daubechies (1992), which are

$$\tilde{\varphi}_{j,k}(x) = \sum_{l\in\mathbb{Z}} \tilde{h}_{l-2k}\tilde{\varphi}_{j+1,l}(x), \quad \tilde{\psi}_{j,k}(x) = \sum_{l\in\mathbb{Z}} \tilde{g}_{l-2k}\tilde{\varphi}_{j+1,l}(x);$$
$$\varphi_{j,k}(x) = \sum_{l\in\mathbb{Z}} h_{l-2k}\varphi_{j+1,l}(x), \quad \psi_{j,k}(x) = \sum_{l\in\mathbb{Z}} g_{l-2k}\varphi_{j+1,l}(x),$$

along with the condition

$$\langle \varphi_{0,k}, \tilde{\varphi}_{0,k'} \rangle = \delta_{kk'},$$

where $\delta_{kk'}$ is a Kronecker delta. This condition gives rise to the biorthogonality conditions

$$\langle \varphi_{j,k}, \tilde{\varphi}_{j,k'} \rangle = \delta_{kk'}, \tag{1.2.23}$$
$$\langle \psi_{j,k}, \tilde{\psi}_{j',k'} \rangle = \delta_{jj'}\delta_{kk'}. \tag{1.2.24}$$

Biorthogonal wavelets generate two multiresolution analysis ladders (Cohen et al.; 1992)

$$\cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots,$$
$$\cdots \subset \tilde{V}_{-2} \subset \tilde{V}_{-1} \subset \tilde{V}_0 \subset \tilde{V}_1 \subset \tilde{V}_2 \subset \cdots,$$

where $V_j = \overline{\text{span}}\{\varphi_{j,k};\, k \in \mathbb{Z}\}$, $\tilde{V}_j = \overline{\text{span}}\{\tilde{\varphi}_{j,k};\, k \in \mathbb{Z}\}$, and their (non-orthogonal) complement subspaces $W_j = \overline{\text{span}}\{\psi_{j,k};\, k \in \mathbb{Z}\}$, $\tilde{W}_j = \overline{\text{span}}\{\tilde{\psi}_{j,k};\, k \in \mathbb{Z}\}$. Similar to the orthogonal multiresolution analysis ladder, we have $V_{j+1} = V_j \oplus W_j$ and $\tilde{V}_{j+1} = \tilde{V}_j \oplus \tilde{W}_j$. Different from the orthogonal multiresolution analysis, the spaces V_j and W_j (also $\tilde{V}_j$ and $\tilde{W}_j$) are not orthogonal to each other. This can lead to the stability issues, which will be delved into in our work. As a result from equations (1.2.23) and (1.2.24), we have $V_j \perp \tilde{W}_j$ and $\tilde{V}_j \perp W_j$. As described by Daubechies (1992), these two ladders "fit together like a giant zipper". Consequently, the function expansion form (by biorthogonal wavelets) can be written as (Cohen et al.; 1992)

$$\begin{aligned} f(x) &= \sum_{j\in\mathbb{Z}}\sum_{k\in\mathbb{Z}} \langle f, \psi_{j,k}\rangle \tilde{\psi}_{j,k}(x) \\ &= \sum_{j\in\mathbb{Z}}\sum_{k\in\mathbb{Z}} \langle f, \tilde{\psi}_{j,k}\rangle \psi_{j,k}(x), \end{aligned}$$

or it can be written at a certain resolution level j_0, such that

$$\begin{aligned} f(x) &= \sum_{k\in\mathbb{Z}} \langle f, \varphi_{j_0,k}\rangle \tilde{\varphi}_{j_0,k}(x) + \sum_{j\geq j_0}\sum_{k\in\mathbb{Z}} \langle f, \psi_{j,k}\rangle \tilde{\psi}_{j,k}(x) \\ &= \sum_{k\in\mathbb{Z}} \langle f, \tilde{\varphi}_{j_0,k}\rangle \varphi_{j_0,k}(x) + \sum_{j\geq j_0}\sum_{k\in\mathbb{Z}} \langle f, \tilde{\psi}_{j,k}\rangle \psi_{j,k}(x). \end{aligned} \tag{1.2.25}$$

For more details on biorthogonal wavelets, the reader can refer to Cohen et al. (1992), Cohen and Daubechies (1992), and Daubechies (1992).

1.2.6 Non-decimated Discrete Wavelet Transform (NDWT)

Non-decimated wavelet transforms have already been proven to be a powerful tool for regression and time series problems, some of which will be discussed later on. For classic (first generation) wavelets, non-decimated wavelets aim to 'fill in the gap' of the decimation caused by the dilation of multiresolution analysis, see Nason and Silverman (1995). Due to this fact, Percival (1995) called it the maximal overlap estimator. This construction allows us to perform a **translation-invariant** transform, which 'averages out' the translation dependence caused by information shifts at each level, see Coifman and Donoho (1995).

Suppose we have a dyadic sequence $\underline{c} = \{c_0, ..., c_{N-1}\}$, defined on a set of equally spaced one-dimensional points $\underline{x} = \{0, \frac{1}{N}, ..., \frac{N-1}{N}\}$, where $N = 2^J$, for some $J \in \mathbb{N}^+$. We denote $\mathcal{F}^{\text{low}} = \{h_n\}_{n=0}^{N-1}$ and $\mathcal{F}^{\text{high}} = \{g_n\}_{n=0}^{N-1}$ as the low-pass and high-pass filters, respectively. The low-pass and high-pass filters are designed with a discrete analogue of compact support, which means only a few elements are non-zeros. The two filters operate on the original data, such that

$$\begin{aligned} (\mathcal{F}^{\text{low}}\underline{c})_k &= \sum_{n=0}^{N-1} h_{n-k} c_n, \\ (\mathcal{F}^{\text{high}}\underline{c})_k &= \sum_{n=0}^{N-1} g_{n-k} c_n, \end{aligned}$$

see Daubechies (1992) and Nason (2008). Along with a dyadic decimation operator $\mathcal{D}_0$, such that

$$(\mathcal{D}_0\,\underline{c})_k = c_{2k},$$

for $k \in \{0, ..., \frac{N}{2} - 1\}$, the dilation equation can be written as

$$\underline{c}_{J-1} = \mathcal{D}_0\mathcal{F}^{\text{low}}\underline{c}_J,$$
$$\underline{d}_{J-1} = \mathcal{D}_0\mathcal{F}^{\text{high}}\underline{c}_J,$$

see Nason (2008). For any level $j = 0, ..., J-1$, Nason and Silverman (1995) show that the detail and scaling coefficients can be written as

$$\underline{d}_j = \mathcal{D}_0\mathcal{F}^{\text{high}}(\mathcal{D}_0\mathcal{F}^{\text{low}})^{J-j-1}\underline{c}_J,$$
$$\underline{c}_j = (\mathcal{D}_0\mathcal{F}^{\text{low}})^{J-j}\underline{c}_J.$$

The non-decimation can be carried out by considering a shift operator $\mathcal{S}$, such that

$$(\mathcal{S}\underline{c})_k = c_{k+1}.$$

Nason and Silverman (1995) define a shifted-decimation operator $\mathcal{D}_1 = \mathcal{D}_0\mathcal{S}$, such that

$$\begin{aligned}(\mathcal{D}_1\,\underline{c})_k &= (\mathcal{D}_0\mathcal{S}\,\underline{c})_k \\ &= c_{2k+1}. \end{aligned} \tag{1.2.26}$$

Suppose we have an integer S and its binary representation $\varepsilon_{J-1}\varepsilon_{J-2}\cdots\varepsilon_0$, where each $\varepsilon_j \in \{0, 1\}$ is an indicator of the shift operator, such that $\mathcal{D}_{\varepsilon_j}$ is used from level $j+1$ to j. Nason and Silverman (1995) denote this as ε-decimated discrete wavelet transform, and they show that the detail coefficients obtained by the binary sequence $\varepsilon_{J-1}\varepsilon_{J-2}\cdots\varepsilon_0$ is equivalent to performing the usual DWT on the shifted sequence $\mathcal{S}^S\,\underline{c}$. Nason and Silverman (1995) also discussed the inverse of NDWT and point out the possible advantage of NDWT for nonparametric regression, while Coifman and Donoho (1995) provided a detail analysis of NDWT (what they call translation-invariant transform) on the denoising performance, in which they found that NDWT with wavelet thresholding can suppress visual artifacts, such as the Gibbs phenomena.

1.3 The Lifting Scheme

Although first-generation wavelets are widely regarded as powerful tools, constructing first-generation wavelets encounters some practical restrictions. For example, the length of data has to be dyadic (2^J for some $J \in \mathbb{Z}^+$), and observations have to be equally spaced. However, the data observed for most real-life problems does not follow these restrictions. Motivated by this, Sweldens (1995, 1996b, 1998) introduced a new technique called the *lifting scheme*, which can produce wavelet-like bases adapted to more general sampling cases. The lifting scheme naturally leads to biorthogonal bases and does not rely on the Fourier domain. The wavelet bases generated by the lifting scheme are also known as *second generation wavelets*. In this section, we provide a detailed mathematical introduction to the lifting scheme, which forms the foundation construct for this thesis. However, the reader can refer directly to Section 1.5 for the variant of the lifting scheme that is most relevant for the subsequent chapters.

Now suppose we have an observed signal $\{f(x_i)\}_{i=1}^n$, defined on the n different locations $\{x_i\}_{i=1}^n$, where $x_1 < x_2 < \cdots < x_n$. Note that here these locations are no longer assumed to be regularly distributed. The lifting scheme (Sweldens; 1998) consists of three steps: split, prediction, and update. A single step split-predict-update paradigm can be described as follows.

- **Split:** This step is equivalent to the downsampling of the discrete wavelet transform. In the lifting construction, the observation locations $\{x_i\}_{i=1}^n$ are separated into two sets, $\{x_{2i-1}\}_{i=1}^{\lceil \frac{n}{2} \rceil}$ and $\{x_{2i}\}_{i=1}^{\lfloor \frac{n}{2} \rfloor}$, which indicate odd and even positions of the location sequence. A different 'split' strategy will be described in Section 1.5.

- **Predict:** Then we obtain the 'prediction' for the values on the odd positions, $\{f(x_{2i-1})\}_{i=1}^{\lceil \frac{n}{2} \rceil}$, by the information from the even ones, $\{f(x_{2i})\}_{i=1}^{\lfloor \frac{n}{2} \rfloor}$. The difference between the observation values and the prediction values are interpreted as the detail or wavelet coefficients.

- **Update:** The observations on the even positions are then updated based on their current values and the detail coefficients obtained in the prediction step. This step

aims to preserve some measurements of the signal, such as its energy and its mean value.

These three steps will be reiterated to obtain a set of detail coefficients and remaining scaling coefficients (those that have not been predicted).

The lifting scheme is easier to implement than the classic wavelets, and can be flexibly designed (Sweldens and Schröder; 2005). In addition, one of the advantages of the lifting scheme in the theoretical aspect is that it does not rely on the Fourier (frequency) domain and instead, the construction can be carried out entirely in the data domain.

In the following section, the MRA framework for the second generation wavelets will be briefly described, and for more details, the reader can refer to Sweldens (1996a, 1998).

1.3.1 Second Generation Multiresolution Analysis

The second generation MRA can be described as follows (Sweldens; 1998).

Definition 1.3.1. A second generation **primal** multiresolution analysis of the space $L^2(\mathbb{R})$ is (still) defined as a nested sequence of closed subspaces $\{V_j\}_{j\in\mathcal{J}\subset\mathbb{Z}}$ such that

1. Conditions 1-3 in classic multiresolution analysis (Section 1.2.2) still hold.
2. Condition 4-5 in classic multiresolution analysis do not necessarily hold.
3. For each j, the space V_j has a **Riesz basis** given by scaling functions $\{\varphi_{j,k}\}_{k\in\mathcal{S}_j}$, such that for a function $f \in L^2(\mathbb{R})$, there is

$$L\,\|f\|_{L_2}^2 \leq \sum_{k\in\mathcal{S}_j} |\langle f, \varphi_{j,k}\rangle|^2 \leq U\,\|f\|_{L_2}^2, \tag{1.3.1}$$

where $\mathcal{S}_j$ is an index set at level j, with $\mathcal{S}_j \subset \mathcal{S}_{j+1}$, and $0 < L \leq U < \infty$.

Definition 1.3.2. A second generation **dual** multiresolution analysis consists of a nested sequence of closed subspaces $\{\tilde{V}_j\}_{j\in\mathcal{J}\subset\mathbb{Z}}$, where each subspace $\tilde{V}_j$ has the same properties with V_j as described in definition 1.3.1. For each space $\tilde{V}_j$, there is a set of (dual) scaling functions $\{\tilde{\varphi}_{j,k}\}_{k\in\mathcal{S}_j}$, such that

$$\langle \varphi_{j,k}, \tilde{\varphi}_{j,k'}\rangle = \delta_{kk'}, \quad \forall j \in \mathcal{J} \subset \mathbb{Z};\, k, k' \in \mathcal{S}_j. \tag{1.3.2}$$

It is clear that the dual and primal scaling functions satisfy the biorthogonality property as described in Section 1.2.5. The primal and dual scaling functions are designed to have properties such that

$$\int_{-\infty}^{\infty} \tilde{\varphi}_{j,k}(x)dx = 1,$$
$$\sum_{k\in\mathcal{S}_j} \varphi_{j,k}(x) = 1, \quad \forall x \in \mathbb{R},$$

see Sweldens (1998).

Recall that $V_j \subset V_{j+1}$, then we can define a filter with the coefficients $\{h_{j,k,l}\}_{l\in\mathcal{S}_{j+1}}$ for each scaling function $\varphi_{j,k}$, where $j \in \mathcal{J} \subset \mathbb{Z}$ and $k \in \mathcal{S}_j$, such that

$$\varphi_{j,k}(x) = \sum_{l\in\mathcal{S}_{j+1}} h_{j,k,l}\varphi_{j+1,l}(x). \tag{1.3.3}$$

This can be considered as the equivalence of the refinement relation of the classic wavelet theory, since each scaling function at level j can be written as a linear combination of those at level $j+1$. To ensure that equation (1.3.3) is well-defined, the sets $\{k \in \mathcal{S}_j \,|\, h_{j,k,l} \neq 0\}$ and $\{l \in \mathcal{S}_{j+1} \,|\, h_{j,k,l} \neq 0\}$ have to be finite, which can be considered as the 'compact support' in classic wavelet theory. In addition, the values $h_{j,k,l}$, for all j, k, l, have to be uniformly bounded. Similarly, the refinement for the dual scaling function can be constructed by a dual filter, such that

$$\tilde{\varphi}_{j,k}(x) = \sum_{l\in\mathcal{S}_{j+1}} \tilde{h}_{j,k,l}\tilde{\varphi}_{j+1,l}(x), \tag{1.3.4}$$

in which the filter $\{\tilde{h}_{j,k,l}\}_{\forall j,k,l}$ is finite and uniformly bounded as for the primal one.

1.3.2 Constructing Second Generation Wavelets and Filters

For constructing wavelets, we can define another two filters $\{g_{j,m,l}\}$ and $\{\tilde{g}_{j,m,l}\}$, where $j \in \mathbb{Z}$, $m \in \mathcal{D}_j = \mathcal{S}_{j+1}\backslash\mathcal{S}_j$, and $l \in \mathcal{S}_{j+1}$, such that

$$\psi_{j,m}(x) = \sum_{l\in\mathcal{S}_{j+1}} g_{j,m,l}\varphi_{j+1,l}(x), \tag{1.3.5}$$

$$\tilde{\psi}_{j,m}(x) = \sum_{l\in\mathcal{S}_{j+1}} \tilde{g}_{j,m,l}\tilde{\varphi}_{j+1,l}(x). \tag{1.3.6}$$

Similar to the filters h and $\tilde{h}$, here g and $\tilde{g}$ are assumed to be finite and uniformly bounded.

By recursively performing equations (1.3.5) and (1.3.6), we can obtain a set of functions $\{\varphi_{j_0,k}\}_{k\in\mathcal{S}_{j_0}} \bigcup \{\psi_{j,k}\}_{j\geq j_0,\, m\in\mathcal{D}_j}$, or a fully decomposed version $\{\psi_{j,k}\}_{j\in\mathbb{Z},m\in\mathcal{D}_j}$. Note that the second generation MRA generates a biorthogonal basis for $L^2(\mathbb{R})$ (Sweldens; 1998), the function can be written as a similar form as in equation (1.2.25), such that

$$\begin{aligned} f(x) &= \sum_{k\in\mathcal{S}_{j_0}} \langle f, \tilde{\varphi}_{j_0,k}\rangle \varphi_{j_0,k}(x) + \sum_{j\geq j_0} \sum_{m\in\mathcal{D}_j} \langle f, \tilde{\psi}_{j,m}\rangle \psi_{j,m}(x) \\ &= \sum_{j\in\mathbb{Z}} \sum_{m\in\mathcal{D}_j} \langle f, \tilde{\psi}_{j,m}\rangle \psi_{j,m}(x). \end{aligned} \tag{1.3.7}$$

Recall the biorthogonality property from Section 1.2.5, which is $\langle \varphi_{j,k}, \tilde{\varphi}_{j,k'}\rangle = \delta_{kk'}$. Then equations (1.3.3) and (1.3.4) can lead to

$$\sum_{l\in\mathcal{S}_{j+1}} h_{j,k,l}\tilde{h}_{j,k',l} = \delta_{kk'}, \quad \forall j \in \mathbb{Z},\, k, k' \in \mathcal{S}_j. \tag{1.3.8}$$

Note from equation (1.3.3), we have

$$\begin{aligned} \langle \varphi_{j,k}, \tilde{\varphi}_{j+1,l}\rangle &= \left\langle \left(\sum_{l'\in\mathcal{S}_{j+1}} h_{j,k,l'}\, \varphi_{j+1,l'} \right), \tilde{\varphi}_{j+1,l} \right\rangle \\ &= \sum_{l'\in\mathcal{S}_{j+1}} h_{j,k,l'} \langle \varphi_{j+1,l'}, \tilde{\varphi}_{j+1,l}\rangle \\ &= h_{j,k,l}, \end{aligned}$$

which gives us the construction of the filters h. Similarly, the construction for the filters $\tilde{h}$ can be obtained by $\tilde{h}_{j,k,l} = \langle \tilde{\varphi}_{j,k}, \varphi_{j+1,l}\rangle$. Similar relations for filters g and $\tilde{g}$ can be written as

$$\sum_{l\in\mathcal{S}_{j+1}} g_{j,m,l}\tilde{g}_{j,m',l} = \delta_{mm'}, \quad \forall j \in \mathbb{Z},\, m, m' \in \mathcal{D}_j, \tag{1.3.9}$$

where the filters can be obtained by $g_{j,m,l} = \langle \psi_{j,m}, \tilde{\varphi}_{j+1,l}\rangle$, and $\tilde{g}_{j,m,l} = \langle \tilde{\psi}_{j,m}, \varphi_{j+1,l}\rangle$.

Recall that $V_j \perp \tilde{W}_j$ and $\tilde{V}_j \perp W_j$, which indicate that

$$\sum_{l \in \mathcal{S}_{j+1}} h_{j,k,l} \tilde{g}_{j,m,l} = 0, \tag{1.3.10}$$

$$\sum_{l \in \mathcal{S}_{j+1}} \tilde{h}_{j,k,l} g_{j,m,l} = 0, \tag{1.3.11}$$

for all j, k, m.

Definition 1.3.3. A set of filters $\{h, \tilde{h}, g, \tilde{g}\}$ is a set of biorthogonal filters if equations (1.3.8), (1.3.9), (1.3.10), and (1.3.11) hold.

1.3.3 Fast Second Generation Wavelet Transform

The second generation fast wavelet transform is similar in spirit to the first generation one. Denote the level-j scaling coefficient $c_{j,k} = \langle f, \tilde{\varphi}_{j,k} \rangle$, for $k \in \mathcal{S}_j$ and the level-j wavelet coefficients $d_{j,m} = \langle f, \tilde{\psi}_{j,m} \rangle$, for $m \in \mathcal{D}_j$.

By the refinement relations in equations (1.3.4) and (1.3.6), we have that

$$c_{j,k} = \sum_{l \in \mathcal{S}_{j+1}} \tilde{h}_{j,k,l} c_{j+1,l}, \quad \forall j \in \mathbb{Z},\, k \in \mathcal{S}_j,$$

$$d_{j,m} = \sum_{l \in \mathcal{S}_{j+1}} \tilde{g}_{j,m,l} c_{j+1,l}, \quad \forall j \in \mathbb{Z},\, m \in \mathcal{D}_j.$$

Based on equations (1.3.3) and (1.3.5), the inverse transform can be done by recursively applying the following equation

$$c_{j+1,l} = \sum_{k \in \mathcal{S}_j} h_{j,k,l} c_{j,k} + \sum_{m \in \mathcal{D}_j} g_{j,m,l} d_{j,m}. \tag{1.3.12}$$

1.3.4 The Lifting Transform in Practice

Up to this point we have seen the framework of the (biorthogonal) second generation wavelets, but one might have the question on how could we perform the transform in practice? Let us first start with the following theorem.

Theorem 2 (*Primal Lifting* in index notation (Sweldens; 1998))**.** If there is a set of biorthogonal filters $\{h, \tilde{h}, g, \tilde{g}\}$, then a new set of biorthogonal filters $\{h^{\text{new}}, \tilde{h}^{\text{new}}, g^{\text{new}}, \tilde{g}^{\text{new}}\}$ can be constructed via

$$h^{\text{new}}_{j,k,l} = h_{j,k,l}, \tag{1.3.13}$$

$$\tilde{h}^{\text{new}}_{j,k,l} = \tilde{h}_{j,k,l} + \sum_{m} b_{j,k,m}\tilde{g}_{j,m,l}, \tag{1.3.14}$$

$$g^{\text{new}}_{j,m,l} = g_{j,m,l} - \sum_{k} b_{j,k,m}h_{j,k,l}, \tag{1.3.15}$$

$$\tilde{g}^{\text{new}}_{j,m,l} = \tilde{g}_{j,m,l}. \tag{1.3.16}$$

Combining equations (1.3.3)-(1.3.6), with equations (1.3.14) and (1.3.15), we have the dual scaling functions and similarly for the primal wavelets, respectively,

$$\begin{aligned} \tilde{\varphi}^{\text{new}}_{j,k} &= \tilde{\varphi}_{j,k} + \sum_{m\in\mathcal{D}_j} b_{j,k,m}\tilde{\psi}_{j,m}, \\ \psi^{\text{new}}_{j,m} &= \psi_{j,m} + \sum_{k\in\mathcal{S}_j} b_{j,k,m}\varphi_{j,k}. \end{aligned} \tag{1.3.17}$$

Now assume we have a set of observations $\{f_i\}_{i=1}^n$ from a underlying function $f \in L^2(\mathbb{R})$, with one-dimensional locations $x_1 < \cdots < x_n$, where the i-th observation is located at x_i. Then from equation (1.3.17) we have

$$\begin{aligned} \langle f, \tilde{\varphi}^{\text{new}}_{j,k}\rangle &= \left\langle f, \left(\tilde{\varphi}_{j,k} + \sum_{m\in\mathcal{D}_j} b_{j,k,m}\tilde{\psi}_{j,m}\right)\right\rangle \\ \implies c^{\text{new}}_{j,k} &= c_{j,k} + \sum_{m\in\mathcal{D}_j} b_{j,k,m}d_{j,m}. \end{aligned} \tag{1.3.18}$$

It is easy to check that these 'new' filters are finite and uniformly bounded, see Sweldens (1998) for more details. By a similar construction, Sweldens (1998) show that

the *dual lifting* (in index notation) can be presented as

$$h_{j,k,l}^{\text{new}} = h_{j,k,l} + \sum_m a_{j,k,m} g_{j,m,l}, \tag{1.3.19}$$

$$\tilde{h}_{j,k,l}^{\text{new}} = \tilde{h}_{j,k,l}, \tag{1.3.20}$$

$$g_{j,m,l}^{\text{new}} = g_{j,m,l}, \tag{1.3.21}$$

$$\tilde{g}_{j,m,l}^{\text{new}} = \tilde{g}_{j,m,l} - \sum_k a_{j,k,m} \tilde{h}_{j,k,l}. \tag{1.3.22}$$

Similarly, from equation (1.3.22), we can have the dual wavelets

$$\tilde{\psi}_{j,m}^{\text{new}} = \tilde{\psi}_{j,m} - \sum_{k \in \mathcal{S}_j} a_{j,k,m} \tilde{\varphi}_{j,k}. \tag{1.3.23}$$

Then from equation (1.3.23) we have

$$\begin{aligned} \langle f, \tilde{\psi}_{j,m}^{\text{new}} \rangle &= \left\langle f, \left(\tilde{\psi}_{j,m} - \sum_{k \in \mathcal{S}_j} a_{j,k,m} \tilde{\varphi}_{j,k} \right) \right\rangle \\ \implies d_{j,m}^{\text{new}} &= d_{j,m} - \sum_{k \in \mathcal{S}_j} a_{j,k,m} c_{j,k}. \end{aligned} \tag{1.3.24}$$

The lifting scheme provides a different philosophy. Assuming we have a set of observations $\{f_i\}_{i=1}^n$ from an underlying function $f \in L^2(\mathbb{R})$, sampled at one-dimensional locations $x_1 < \cdots < x_n$, where the i-th observation is located at x_i, we then iteratively apply dual lifting followed by primal lifting. For example, we start with a set of primal scaling functions $\{\varphi_{J,i}(x) = \chi_{x_i}(x)\}_{i=1}^n$ and a set of dual scaling functions $\{\tilde{\varphi}_{J,i}(x) = \delta(x - x_i)\}_{i=1}^n$, hence the i-th initial scaling coefficient can be represented as $c_{J,i} = \langle f, \tilde{\varphi}_{J,i} \rangle = f_i$. For obtaining the level-$(J-1)$ coefficients, we perform the following steps.

- **Split**: Split the index set $\mathcal{S}_J = \{1, ..., n\}$ into two sets, $\mathcal{S}_{J-1}$ and $\mathcal{D}_{J-1}$. For $k \in \mathcal{S}_{J-1}$ and $m \in \mathcal{D}_{J-1}$, we denote $c_{J-1,k} = c_{J,k}$ and $d_{J-1,m} = c_{J,m}$, respectively.

- **Predict**: The wavelet coefficients will be obtained by

  $$d_{J-1,m} := d_{J-1,m} - \sum_{k \in \mathcal{S}_{J-1}} a_{J-1,k,m} c_{J-1,k},$$

  for all $k \in \mathcal{S}_{J-1}$.

- **Update**: The scaling coefficients will be updated by

$$c_{J-1,k} := c_{J-1,k} + \sum_{m \in \mathcal{D}_{J-1}} b_{J-1,k,m} d_{J-1,m},$$

  for all $m \in \mathcal{D}_{J-1}$.

If we iterate the above steps until a chosen (primary resolution) level j_0, then a sequence of wavelet coefficients can be obtained as in classic DWT. It is worth to mention that the odd-even split strategy is just one of the possible choice, other choices can be chosen to handle other irregular situations, such as data from high-dimensional spaces, which will be expanded upon in Section 1.5.

1.4 Graphs

The structure of the data collected from networks can be modelled by graphs, the main objects in the study of graph theory. In this section, we will present some background knowledge on graph theory that will contribute to our work. We then introduce an existing scheme that can handle data from the vertex set of a network, see Section 1.5 for more details.

1.4.1 Basics of Graph Theory

Much real-life data can be modelled by graphs, which consist of a set of objects and the pairwise relations among these objects. Mathematically, a graph G can be represented as an ordered pair $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, ..., v_n\}$ is the set of vertices and $\mathcal{E} = \{e_1, ..., e_m\}$ is the set of edges, see e.g., Bondy and Murty (2008). The cardinality of the vertex and the edge sets can be denoted as $|\mathcal{V}| = n$ and $|\mathcal{E}| = m$, respectively. If the k-th edge indicates the connection between i-th and j-th vertices, then we write $e_k = \{v_i, v_j\}$, which is an unordered pair of these two vertices ($\{v_i, v_j\} = \{v_j, v_i\}$). In this case, we say v_j is a neighbour of v_i, and vice versa. We further denote $\mathcal{N}_i^{\mathcal{V}}$ as the set of neighbouring vertices of the vertex v_i, which is defined as $\mathcal{N}_i^{\mathcal{V}} = \{v_j \,|\, v_j \in \mathcal{V}; \{v_i, v_j\} \in \mathcal{E}\}$. The

graphs with all edges as unordered pairs are undirected graphs. A directed graph contains ordered pairs as the edges, where we denote $e_k = (v_i, v_j)$, and $(v_i, v_j) \neq (v_j, v_i)$ if $i \neq j$. Intuitively, the order indicates that the vertex v_i is connected to v_j through the edge e_k, but not the other way around. In this work, when we mention the term 'graph', it always means an undirected graph unless explicitly stated. Graphs can be divided into two main groups: **simple graphs** and **multi-graphs**. A graph is a **simple graph** if: (1) there is no edge starting from a vertex and ending at the same vertex (a loop with only *one* vertex), which means that for any edge $e_k = \{v_i, v_j\}$, we have $i \neq j$; (2) there is no pair of vertices with two or more than two edges linking them; thus, $e_k = e_{k'}$ only if $k = k'$. A graph is a **multi-graph** if it is not a simple graph. In our work, all graph structures we use are simple graphs, for more details on multi-graphs, the reader can refer to Diestel (2005).

Furthermore, it would be helpful to provide some terminology for different types of graphs. Listed below are some common graph structures that we will use in our study. A **complete graph** is one in which every pair of vertices is connected by an unique (undirected) edge. If a complete graph contains n vertices, then the number of edges is $m = n(n-1)/2$. The **degree** of a vertex is measured by the number of its neighbouring vertices. A graph is a d-**regular** graph if every vertex is connected with d other vertices. This structure is very common in many areas, for example, certain chemical or biological structures form a regular graph, see Bonchev (1991). A graph is said to have **loops** if for a vertex or some vertices, there exists a non-overlapping path from one vertex and back to itself. A **tree** is a special case of graph, which is a structure without loops. A graph $G' = (\mathcal{V}', \mathcal{E}')$ is a **subgraph** of a graph $G = (\mathcal{V}, \mathcal{E})$ if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$. In general, for a given set of vertices, any possible graph structure is the subgraph of the complete graph with the same vertices. A **spanning subgraph** is a subgraph that contains the full set of vertices, or it only deletes some edges from the graph associated with the set of vertices; see Bondy and Murty (2008). Thus, a spanning subgraph of the graph $G = (\mathcal{V}, \mathcal{E})$ can be denoted as $G_s = (\mathcal{V}, \mathcal{E}_s)$, where $\mathcal{E}_s \subset \mathcal{E}$. If the subgraph G_s is a tree, then it is known as a **spanning tree**. If for a graph $G = (\mathcal{V}, \mathcal{E})$ there exists a partition of

the vertex set such that $\mathcal{V} = \mathcal{V}_X \bigcup \mathcal{V}_Y$ and $\mathcal{V}_X \bigcap \mathcal{V}_Y = \emptyset$, then the graph is a **bipartite graph**, if any edge $\{x, y\}$ satisfies $x \in \mathcal{V}_X$ and $y \in \mathcal{V}_Y$. A special case of trees is a **star graph**, which is a bipartite graph with one of the partitioning set containing only one element (vertex).

1.4.2 Matrices Associated to Graphs

Here let us introduce two most commonly used matrices, the adjacency matrix and the incidence matrix, used for representing the graph structure. Suppose we have a graph $G = (\mathcal{V}, \mathcal{E})$, which consists of $|\mathcal{V}| = n$ vertices (or nodes) and $|\mathcal{E}| = m$ edges, with $|\cdot|$ representing the set cardinality. Then the (i, j)-th entry of the $n \times n$ vertex **adjacency matrix** $A(G)$ is given by

$$[A(G)]_{ij} = \begin{cases} 1, & \text{if } v_i \leftrightsquigarrow v_j \text{ and } i \neq j; \\ 0, & \text{otherwise,} \end{cases} \tag{1.4.1}$$

where '$\leftrightsquigarrow$' indicates an existing connection (edge) between two vertices. In our case, we assume the graphs we are interested in contain no self-loops, which indicates $[A(G)]_{ii} = 0$ for any adjacency matrix we will use in this work.

As mentioned above, another well-studied matrix in graph theory is the **incidence matrix**, a $n \times m$ matrix whose (i, k)-th entry is

$$[B(G)]_{ik} = \begin{cases} 1, & \text{if } v_i \in e_k; \\ 0, & \text{otherwise,} \end{cases} \tag{1.4.2}$$

For the incidence matrix $B(G)$, the k-th column gives the set of vertices contained in the k-th edge. In this sense, we have $\sum_{i=1}^{n} [B(G)]_{ik} = 2$ for all $k \in \{1, ..., m\}$, since each edge is given by two different vertices. From now on, when it is clear which graph we refer to, we will drop '(G)' for simplification. For example, when using B_{ik} instead of $[B(G)]_{ik}$, we should recall that B is still a function of G rather than only a matrix.

1.4.3 Weighted Graphs

Previous sections described different types of graphs based on their topology, while many real-life problems often require more information, see Bondy and Murty (2008). Weighted graphs can provide such kind of information. Mathematically, a weighted graph G^{ω} is modelled by an ordered pair (G, ω), where G is the graph topology containing the vertex set and their connectivity (edge set), and $\omega : \mathcal{E} \longrightarrow \mathbb{R}$ is a function that associates a weight to each edge, see Bondy and Murty (2008). To fit our framework, we let the weight for each edge $e \in \mathcal{E}$ be non-zero. Furthermore, we let $\omega : \mathcal{E} \longrightarrow \mathbb{R}^{+}$ for any undirected graph (the case with direction/orientation will be discussed in later chapters). For the edge $e_k = \{v_i, v_j\} \in \mathcal{E}$, the notation ω_k or ω_{ij} will be used interchangeably to represent the weight defined on this edge. In the literature on graph theory, there are many different ways to consider the weights associated with edges; for example, weights can be the cost in the travelling salesman problem (Bondy and Murty; 2008), resistance in electronic networks (Bollobás; 1998), or traffic flows in a road network. The size of the weights has different (sometimes opposite) effects on various applications. For example, a large weight on an edge indicates a weak connection between two vertices associated with this edge when the weights represent cost; however, in the case of traffic flows, it indicates a strong link. For the purpose of our work, we let the connection between two vertices v_i and v_j be monotonically increasing with the weight of the edge $e_k = \{v_i, v_j\}$. Therefore, if v_i and v_j are close to each other, then the weight ω_k tends to be large.

1.4.4 Metrized Graphs

Weighted graphs, as we discussed in the previous section, are widely used in many classical graph theory problems. However, recently, more focus has been put on metric graphs, due to their wide ranging applications, for example, for Hilbert spaces defined on graphs (Kuchment; 2003) and for the analysis of electronic networks (Baker and Faber; 2006). In a nutshell, the metrized graph gives geometric properties for a weighted graph, which allows us to define and analyse functions on the graph domain. The following

introduction to the metrized graph theory is mainly based on the work of Baker and Faber (2006). A metrized graph Γ of a weighted graph $G^\omega = (G, \omega)$ arises from the a pair (G, ℓ), where $\ell : \mathcal{E} \longrightarrow \mathbb{R}^+$ is a function that assigns lengths to all edges, in the following way. Here we follow the construction from Baker and Faber (2006) and define the length as the inverse weight, such that $\ell(e) = 1/\omega(e)$, for all $e \in \mathcal{E}$. Then to each edge e, we associate a line segment (or equivalently, an one-dimensional interval) of length $\ell(e)$ and identify the ends of distinct line segments if they correspond to the same vertex $v \in \mathcal{V}$. Sometimes for the length of a certain edge, for example, the k-th edge e_k, we write ℓ_k for convenience. The space $\Gamma(G^\omega)$ is the space that contains the points in any of these line segment intervals, and G^ω is referred to as a model for $\Gamma(G^\omega)$. Sometimes we simply denote $\Gamma(G^\omega)$ as Γ for convenience if there is no ambiguity. The distance between two points in Γ is defined as the length of the shortest path between them along the line segments traversed. This distance is referred to as the path metric, and Γ with this metric form a metrized graph.

Baker and Faber (2006) also introduced a way to define a vertex set and the associated edge set on Γ, we re-state the main ideas of their definitions below.

- **Defining a vertex set on Γ:**

  Let $\mathrm{Vex}(\Gamma)$ be any finite, non-empty subset of Γ, such that $\mathrm{Vex}(\Gamma)$ contains all points with $n_{\mathbf{p}} \neq 2$, where $n_{\mathbf{p}}$ denotes the number of the directions by which a path can leave the point $\mathbf{p}$, which is referred to as the valence in Baker and Faber (2006). Hence, $\Gamma \backslash \mathrm{Vex}(\Gamma)$ is a finite, disjoint union of subspaces $\{U_k\}_k$ isometric to open intervals, where U_k can be considered as a neighbourhood of a point $\mathbf{p}_k \in \Gamma$.

  The reason for selecting the points in $\mathrm{Vex}(\Gamma)$ from any other points in Γ according to their valence is that these are the points that distinguish Γ from an isometry of open interval, see Baker and Faber (2006). For example, consider a vertex v_s, with two distinct neighbouring vertices v_{j_1} and v_{j_2}, through two edges $e_{k_1} = \{v_s, v_{j_1}\}$ and $e_{k_2} = \{v_2, v_{j_2}\}$. As we discussed above, we can consider two line segments $[0, \ell(e_{k_1})]$ and $[0, \ell(e_{k_2})]$ as their isometry, where ℓ is the length function. If we

translate $[0, \ell(e_{k_2})]$ to $[\ell(e_{k_1}), \ell(e_{k_1}) + \ell(e_{k_2})]$, then the interval $[0, \ell(e_{k_1}) + \ell(e_{k_2})]$ can be considered as an isometry of the union of these two edges on Γ. Clearly, this property does not hold for any vertex with $n_{\mathbf{p}} \neq 2$.

- **Defining an edge set on Γ:**

  Once a certain vertex set has been defined, the associated (metrized) edge set can be constructed based on it. Denote by U_k a subset of $\Gamma \backslash \text{Vex}(\Gamma)$, which is isometric to an open interval (as we discussed above), thus its topological closure $\overline{U}_k$ is isometric to a line segment. Then we define $e_k^{\text{met}} = \overline{U}_k$ as the k-th edge of the metrized graph. We also denote the metrized k-th edge e_k^{met} of $e_k = \{v_i, v_j\}$ as $[v_i, v_j]$ for convenience. Additionally, any distinct metrized edges e_k^{met} and $e_{k'}^{\text{met}}$ $(k \neq k')$ intersect in at most one point.

Note that the choice of $\text{Vex}(\Gamma)$ is not unique (Baker and Faber; 2006), and each choice of the vertex set determines a distinct set of metrized edges $\{e_k^{\text{met}}\}$. Specifically, we define $\text{Vex}_G(\Gamma)$ as the vertex set that corresponds to the vertex set of the (original non-metrized) graph $G = (\mathcal{V}, \mathcal{E})$. For two different graphs G and G', we denote $G \sim G'$ if they admit a common refinement. Mathematically, a common refinement $\tilde{G}$ is a graph satisfies that $\text{Vex}_G(\Gamma) \subseteq \text{Vex}_{\tilde{G}}(\Gamma)$ and $\text{Vex}_{G'}(\Gamma) \subseteq \text{Vex}_{\tilde{G}}(\Gamma)$. Without loss of generality, if both G and $\tilde{G}$ are the models of Γ, and $\text{Vex}_G(\Gamma) \subset \text{Vex}_{\tilde{G}}(\Gamma)$, then we call $\tilde{G}$ a refinement (or refined graph) of G and we have $G \sim \tilde{G}$.

The metrized graph and this shortest path distance define a metric space, denoted as $(\Gamma, \text{dist}_{\text{path}})$, see Baker and Faber (2006). For the mathematical details of the path distance, please refer to Appendix A.

A Note for Hilbert Spaces on Metrized Graphs

For constructing a multiresolution analysis, having the knowledge of the underlying function spaces is always desirable. The space $L^2(\Gamma)$ can be defined as follows (Kuchment; 2003).

Definition 1.4.1. For a metrized graph Γ corresponding to the original graph G^{ω}, the space $L^2(\Gamma)$ consists of functions that are measurable and integrable on each metrized edge, such that

$$\|f\|^2_{L^2(\Gamma)} = \sum_{e\in\mathcal{E}} \|f\|^2_{L^2(e^{\mathrm{met}})} < \infty,$$

where the metrized edge e^{met} corresponding to the original edge $e \in \mathcal{E}$ is isometric to a line segment $[0, \ell(e^{\mathrm{met}})]$ and $\ell(e^{\mathrm{met}})$ is the length of this edge.

The space $L^2(\Gamma)$ can be considered as the orthogonal direct sum of $L^2(e^{\mathrm{met}})$, see Kuchment (2003). Consequently, since e^{met} is isometric to a closed interval on $\mathbb{R}$, the inner product and orthogonality can be defined as in $L^2(\mathbb{R})$.

1.5 LOCAAT Transform

The LOCAAT (**L**ifting **O**ne **C**oefficient **A**t **A** **T**ime) transform (Jansen et al.; 2004, 2009) is a variant of the lifting scheme that perform a split stage that isolates one-point at each step, as opposed to odds and evens, and provides a wavelet-like decomposition for various types of data. In addition to the one-dimensional case, LOCAAT can also work for complex data topology, such as spatial points and network vertices. The LOCAAT scheme (for graph/network data) relies only on the topology/connectiveness and the inter-vertex distance information. The original work of graph-LOCAAT (Jansen et al.; 2004, 2009) deals with the case when observations (with/without coordinates) come from an Euclidean space $\mathbb{R}^d$, where $d \geq 2$ and $d \in \mathbb{Z}$, and the graph may be constructed by performing the minimal spanning tree among these points. Many works extended the scope of LOCAAT to other cases, for example, one-dimensional data (Nunes et al.; 2006; Knight and Nason; 2006) and non-tree networks (Mahadevan; 2010). For fitting into our framework, we use the case in which data is endowed with interpoint distance information.

Suppose we have a tree graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, ..., v_n\}$ and $\mathcal{E} = \{e_1, ..., e_m\}$, and the inter-vertex distance measure $\mathrm{dist}(v_i, v_j)$, $\forall v_i, v_j \in \mathcal{V}$ is known. Additionally,

1. $\text{dist}(v_i, v_i) = 0$, $\forall v_i \in \mathcal{V}$,

2. $\text{dist}(v_i, v_j) > 0$, $\forall v_i, v_j \in \mathcal{V}$ and $i \neq j$,

3. $\text{dist}(v_i, v_j) \leq \text{dist}(v_i, v_k) + \text{dist}(v_j, v_k)$, $\forall v_i, v_j, v_k \in \mathcal{V}$,

to guarantee that the set of vertices and their distance measure form a metric space. Then suppose that we have a set of values $\{f_1^{\mathcal{V}}, ..., f_n^{\mathcal{V}}\}$ of a function $f^{\mathcal{V}} \in L^2(\mathbb{R}^d)$ defined on the vertex set, where $f_i^{\mathcal{V}}$ is the value associated with the vertex v_i, viewed as a 'site' in $\mathbb{R}^d$. The target is to transform these values into a set of wavelet and scaling coefficients (in the second generation wavelet domain). From the aspect of function representation, initially we have the finest level expansion of function approximation such that for $v \in \mathcal{V}$,

$$f^{\mathcal{V}}(v) = \sum_{i=1}^{n} c_{i,n}^{\mathcal{V}} \varphi_{i,n}^{\mathcal{V}}(v),$$

where $c_{i,n}^{\mathcal{V}} := f_i^{\mathcal{V}}$ and $\{\varphi_{i,n}^{\mathcal{V}}\}$ form a basis for the space $L^2(\mathbb{R}^d)$. Here $\{\varphi_{i,n}^{\mathcal{V}}(v_k) = \delta_{ik}\}_{i=1}^n$ is the set of initial (or finest) scaling functions at stage-n, which can be defined as Kronecker deltas. The LOCAAT transform follows the split-predict-update paradigm from the lifting scheme of Sweldens (1998); Sweldens and Schröder (2005), along with an extra relinkage procedure. At the initial stage-n, a single-step LOCAAT is as follows.

- **Split:** As the name 'lifting one coefficient at a time' suggests, one coefficient (corresponding to one vertex) will be chosen to be removed and to be predicted. This coefficient will be chosen based on the integral values of the initial primal scaling function. Jansen et al. (2009) define the integral of a function as a weighted sum along with a set of weights $\{w_i\}_{i=1}^n$. For example, the integral of the function f in the domain $\mathcal{V}$ will be $\int_{\mathcal{V}} f^{\mathcal{V}}(v)dv = I_{f^{\mathcal{V}}}^{\mathcal{V}} := \sum_{i=1}^n w_i f_i^{\mathcal{V}}$. The weights proposed in Jansen et al. (2009) are the sum or average of the distances between a vertex and its immediate neighbours, where the immediate neighbours at stage-n are defined as the vertices connected with the chosen vertex by an edge, denoted as

$$\mathcal{N}_{i,n}^{\mathcal{V}} = \{v_j \,|\, \{v_i, v_j\} \in \mathcal{E}\}$$

for i-th vertex at stage-n. Mathematically, for the i-th vertex, the weight w_i as the sum of distances can be presented as

$$w_i^{\text{sum}} = \sum_{j:\, v_j \in \mathcal{N}_{i,n}^{\mathcal{V}}} \text{dist}(v_i, v_j). \tag{1.5.1}$$

In addition, an average version has also been proposed as the weight definition (Jansen et al.; 2009), such that for i-th vertex,

$$w_i^{\text{ave}} = \frac{1}{|\mathcal{N}_{i,n}^{\mathcal{V}}|} \sum_{j:\, v_j \in \mathcal{N}_{i,n}^{\mathcal{V}}} \text{dist}(v_i, v_j). \tag{1.5.2}$$

Then for the associated i-th initial scaling function, we have its integral to be

$$\begin{aligned} I_{i,n}^{\mathcal{V}} : &= \sum_{k=1}^{n} w_k \delta_{ik} \\ &= w_i, \end{aligned}$$

where w_i can be designed by equation (1.5.1) or equation (1.5.2). The choice of vertex to be lifted is then

$$v_{i_n} = \arg\min_{v_i \in \mathcal{V}} \{I_i^{\mathcal{V}}\}.$$

The reason for choosing the vertex with the smallest integral value is that it is associated to the highest sampling frequency in the first steps of the algorithm (Nunes et al.; 2006).

- **Predict:** Once we have chosen the vertex v_{i_n} to be removed, the detail coefficient can be obtained using the information in the neighbouring set, namely

$$d_{i_n}^{\mathcal{V}} = c_{i_n,n}^{\mathcal{V}} - \sum_{j:\, v_j \in \mathcal{N}_{i_n,n}^{\mathcal{V}}} a_{j,n}^{\mathcal{V}} c_{j,n}^{\mathcal{V}}, \tag{1.5.3}$$

where $\{a_{j,n}^{\mathcal{V}}\}_{j:\, v_j \in \mathcal{N}_{i_n,n}^{\mathcal{V}}}$ are the prediction filters at stage-n.

- **Update:** The scaling coefficients associated with the neighbouring vertices of v_{i_n} will be updated from stage-n to stage-$(n-1)$ as follows

$$c_{j,n-1}^{\mathcal{V}} = c_{j,n}^{\mathcal{V}} + b_{j,n}^{\mathcal{V}} d_{i_n}^{\mathcal{V}}, \;\; \forall j \in \mathcal{N}_{i_n,n}^{\mathcal{V}}, \tag{1.5.4}$$

and the remaining scaling coefficients will not be changed. Thus, $\forall k \notin \mathcal{N}^{\mathcal{V}}_{i_n,n}$, we let $c^{\mathcal{V}}_{k,n-1} := c^{\mathcal{V}}_{k,n}$. In addition, the neighbouring integral values will also be updated to account for the loss of vertex v_{i_n}, such that

$$I^{\mathcal{V}}_{j,n-1} = I^{\mathcal{V}}_{j,n} + a^{\mathcal{V}}_{j,n} I^{\mathcal{V}}_{i_n,n}, \ \ \forall j \in \mathcal{N}^{\mathcal{V}}_{i_n,n}. \tag{1.5.5}$$

Similarly, $\forall k \notin \mathcal{N}^{\mathcal{V}}_{i_n,n}$, we let $I^{\mathcal{V}}_{k,n-1} := I^{\mathcal{V}}_{k,n}$.

- **Relink:** Unlike the structure of the one-dimensional case, a tree will be divided into several different sub-trees after performing a split-predict-update step. Then it is essential to have an extra step to relink these sub-trees. Once the vertex v_{i_n} has been removed, Jansen et al. (2004, 2009) suggest performing a minimum spanning tree among the neighbouring node set $\{v_j \in \mathcal{N}^{\mathcal{V}}_{i_n,n}\}$ for relinkage.

- **Iterate :** Then we repeat the three steps above until a stopping time τ that we have set in advance. Generally, we iterate the above steps $(n-2)$ times, resulting in $(n-2)$ detail coefficients and $\tau = 2$ scaling coefficients. At each stage-r, the split-predict-update-relink procedure is of the same form as the one presented above (stage-n), and we only need to change all the notations n to r, for example, i_r indicates the vertex chosen at stage-r.

We let $\mathcal{D}_r = \{i_n, ..., i_{r+1}\}$ be the set of indices of the removed vertices at the beginning of the stage-r, and $\mathcal{S}_r = \{1, ..., n\} \backslash \mathcal{D}_r$ as the set of indices of the remaining vertices. Without loss of generality, we let $\mathcal{D}_n = \emptyset$ and $\mathcal{S}_n = \{1, ..., n\}$. Then at stage-$(r-1)$, the corresponding expansion form for functional approximation can be written as

$$f^{\mathcal{V}}(v) = \sum_{s \in \mathcal{S}_{r-1}} c^{\mathcal{V}}_{s,r-1} \varphi^{\mathcal{V}}_{s,r-1}(v) + \sum_{l \in \mathcal{D}_{r-1}} d^{\mathcal{V}}_l \psi^{\mathcal{V}}_l(v), \tag{1.5.6}$$

where $\{\varphi^{\mathcal{V}}_{s,r-1}\}_s$ and $\{\psi^{\mathcal{V}}_l\}_l$ are (primal) scaling and wavelet functions built 'behind the scenes' by the lifting algorithm, as follows.

1.5.1 MRA Framework for LOCAAT

For $\forall r \in \{n, ..., 3\}$ (we stop at stage-τ) the stage-r dual scaling and detail coefficients are obtained by

$$c_{s,r}^{\mathcal{V}} = \langle f^{\mathcal{V}}, \tilde{\varphi}_{s,r}^{\mathcal{V}} \rangle, \ \ \forall s \in \mathcal{S}_r;$$
$$d_l^{\mathcal{V}} = \langle f^{\mathcal{V}}, \tilde{\psi}_l^{\mathcal{V}} \rangle, \ \ \forall l \in \mathcal{D}_r.$$

If we plug these two inner products into the stage-r predict and update procedure, then we have

$$\langle f^{\mathcal{V}}, \tilde{\psi}_{i_r}^{\mathcal{V}} \rangle = \langle f^{\mathcal{V}}, \tilde{\varphi}_{i_r,r}^{\mathcal{V}} \rangle - \sum_{j:\, v_j \in \mathcal{N}_{i_r,r}^{\mathcal{V}}} a_{j,r}^{\mathcal{V}} \langle f^{\mathcal{V}}, \tilde{\varphi}_{j,r}^{\mathcal{V}} \rangle; \tag{1.5.7}$$

$$\langle f^{\mathcal{V}}, \tilde{\varphi}_{j,r-1}^{\mathcal{V}} \rangle = \langle f^{\mathcal{V}}, \tilde{\varphi}_{j,r}^{\mathcal{V}} \rangle + b_{j,r}^{\mathcal{V}} \langle f^{\mathcal{V}}, \tilde{\psi}_{i_r}^{\mathcal{V}} \rangle, \ \ \forall j \in \mathcal{N}_{i_r,r}^{\mathcal{V}}. \tag{1.5.8}$$

From equations (1.5.7) and (1.5.8), we can see that the dual MRA framework for LOCAAT can be represented as

$$\tilde{\psi}_{i_r}^{\mathcal{V}}(v) = \tilde{\varphi}_{i_r,r}^{\mathcal{V}}(v) - \sum_{j:\, v_j \in \mathcal{N}_{i_r,r}^{\mathcal{V}}} a_{j,r}^{\mathcal{V}} \tilde{\varphi}_{j,r}^{\mathcal{V}}(v);$$
$$\tilde{\varphi}_{j,r-1}^{\mathcal{V}}(v) = \tilde{\varphi}_{j,r}^{\mathcal{V}}(v) + b_{j,r}^{\mathcal{V}} \tilde{\psi}_{i_r}^{\mathcal{V}}(v), \ \ \forall j \in \mathcal{N}_{i_r,r}^{\mathcal{V}}.$$

For obtaining the primal MRA framework, we consider the special case that $f^{\mathcal{V}}(v) = \varphi_{s,r-1}^{\mathcal{V}}(v)$ for one $s \in \mathcal{N}_{i_r,r}^{\mathcal{V}}$. According to the expansion form (1.5.6), we have the following conditions

$$c_{s,r-1}^{\mathcal{V}} = 1;$$
$$c_{s',r-1}^{\mathcal{V}} = 0, \ \ \forall s' \neq s;$$
$$d_l^{\mathcal{V}} = 0, \ \ \forall l \in \mathcal{D}_{r-1} = \{i_n, ..., i_r\}.$$

Then inverting the update step (1.5.4) at stage-r, we have

$$c_{s,r}^{\mathcal{V}} = 1;$$
$$c_{s',r}^{\mathcal{V}} = 0, \ \ \forall s' \neq s.$$

Next we invert the predict step (1.5.3) at stage-r, we have

$$c^{\mathcal{V}}_{i_r,r} = a^{\mathcal{V}}_{s,r}.$$

Thus, we have

$$\begin{aligned}\varphi^{\mathcal{V}}_{s,r-1}(v) = f^{\mathcal{V}}(v) &= \sum_{l\in\mathcal{D}_r} d^{\mathcal{V}}_l \psi^{\mathcal{V}}_l(v) + \sum_{s\in\mathcal{S}_r} c^{\mathcal{V}}_{s,r}\varphi^{\mathcal{V}}_{s,r}(v)\\ &= c^{\mathcal{V}}_{s,r}\varphi^{\mathcal{V}}_{s,r}(v) + a^{\mathcal{V}}_{s,r}\varphi^{\mathcal{V}}_{i_r,r}(v).\end{aligned}$$

Hence for $\forall j \in \mathcal{N}^{\mathcal{V}}_{i_r,r}$, the 'primal update' can be represented as

$$\varphi^{\mathcal{V}}_{j,r-1}(v) = \varphi^{\mathcal{V}}_{j,r}(v) + a^{\mathcal{V}}_{j,r}\varphi^{\mathcal{V}}_{i_r,r}(v), \ \ \forall j \in \mathcal{N}^{\mathcal{V}}_{i_r,r}. \tag{1.5.9}$$

If we integrate equation (1.5.9), it will give us the relationship for updating the integrals (as in equation (1.5.5)). Now let us consider another special case that $f^{\mathcal{V}}(v) = \psi^{\mathcal{V}}_{i_r}(v)$. Then again according to equation (1.5.6), we have

$$\begin{aligned} d^{\mathcal{V}}_{i_r} &= 1;\\ d^{\mathcal{V}}_l &= 0, \ \ \forall l \in \mathcal{D}_r = \{i_n, ..., i_{r+1}\};\\ c^{\mathcal{V}}_{s,r-1} &= 0, \ \ \forall s \in \mathcal{S}_{r-1}.\end{aligned}$$

Then we invert the update step (1.5.4) at stage-r, hence we have

$$c^{\mathcal{V}}_{j,r} = -b^{\mathcal{V}}_{j,r}, \ \ \forall j \in \mathcal{N}^{\mathcal{V}}_{i_r,r},$$

and inverting the predict step (1.5.3) at stage-r, we obtain

$$c^{\mathcal{V}}_{i_r,r} = 1 - \sum_{j:\, v_j\in\mathcal{N}^{\mathcal{V}}_{i_r,r}} a^{\mathcal{V}}_{j,r} b^{\mathcal{V}}_{j,r}.$$

Hence, we have

$$\begin{aligned}
\psi_{i_r}^{\mathcal{V}}(v) = f^{\mathcal{V}}(v) &= \sum_{l\in\mathcal{D}_r} d_l^{\mathcal{V}}\psi_l^{\mathcal{V}}(v) + \sum_{s\in\mathcal{S}_r} c_{s,r}^{\mathcal{V}}\varphi_{s,r}^{\mathcal{V}}(v) \\
&= \left(1 - \sum_{j:\, v_j\in\mathcal{N}_{i_r,r}^{\mathcal{V}}} a_{j,r}^{\mathcal{V}} b_{j,r}^{\mathcal{V}}\right)\varphi_{i_r,r}^{\mathcal{V}}(v) + \sum_{j:\, v_j\in\mathcal{N}_{i_r,r}^{\mathcal{V}}} -b_{j,r}^{\mathcal{V}}\varphi_{j,r}^{\mathcal{V}}(v) \\
&= \varphi_{i_r,r}^{\mathcal{V}}(v) - \sum_{j:\, v_j\in\mathcal{N}_{i_r,r}^{\mathcal{V}}} b_{j,r}^{\mathcal{V}}\left(\varphi_{j,r}^{\mathcal{V}}(v) + a_{j,r}^{\mathcal{V}}\varphi_{i_r,r}^{\mathcal{V}}(v)\right) \qquad (1.5.10) \\
&= \varphi_{i_r,r}^{\mathcal{V}}(v) - \sum_{j:\, v_j\in\mathcal{N}_{i_r,r}^{\mathcal{V}}} b_{j,r}^{\mathcal{V}}\varphi_{j,r-1}^{\mathcal{V}}(v). \qquad (1.5.11)
\end{aligned}$$

From (1.5.10) to (1.5.11) above, we simply apply equation (1.5.9). Then the 'primal predict' can be represented as

$$\psi_{i_r}^{\mathcal{V}}(v) = \varphi_{i_r,r}^{\mathcal{V}}(v) - \sum_{j:\, v_j\in\mathcal{N}_{i_r,r}^{\mathcal{V}}} b_{j,r}^{\mathcal{V}}\varphi_{j,r-1}^{\mathcal{V}}(v). \qquad (1.5.12)$$

For clarity, we summarise the results above as

$$\tilde{\psi}_{i_r}^{\mathcal{V}} = \tilde{\varphi}_{i_r,r}^{\mathcal{V}} - \sum_{j:\, v_j\in\mathcal{N}_{i_r,r}^{\mathcal{V}}} a_{j,r}^{\mathcal{V}}\tilde{\varphi}_{j,r}^{\mathcal{V}}; \qquad (1.5.13)$$

$$\tilde{\varphi}_{j,r-1}^{\mathcal{V}} = \tilde{\varphi}_{j,r}^{\mathcal{V}} + b_{j,r}^{\mathcal{V}}\tilde{\psi}_{i_r}^{\mathcal{V}},\ \ \forall j\in\mathcal{N}_{i_r,r}^{\mathcal{V}}; \qquad (1.5.14)$$

$$\varphi_{j,r-1}^{\mathcal{V}} = \varphi_{j,r}^{\mathcal{V}} + a_{j,r}^{\mathcal{V}}\varphi_{i_r,r}^{\mathcal{V}},\ \ \forall j\in\mathcal{N}_{i_r,r}^{\mathcal{V}}; \qquad (1.5.15)$$

$$\psi_{i_r}^{\mathcal{V}} = \varphi_{i_r,r}^{\mathcal{V}} - \sum_{j:\, v_j\in\mathcal{N}_{i_r,r}^{\mathcal{V}}} b_{j,r}^{\mathcal{V}}\varphi_{j,r-1}^{\mathcal{V}}. \qquad (1.5.16)$$

Equations (1.5.13)-(1.5.16) are linked to the MRA framework for the LOCAAT transform (Jansen et al.; 2009).

1.5.2 Filter Design

For the construction of the stage-r prediction filters $\{a_{j,r}^{\mathcal{V}}\}_{j\in\mathcal{N}_{i_r,r}^{\mathcal{V}}}$, Jansen et al. (2009) suggested to adopt the normalised inverse distance, such that

$$a_{j,r}^{\mathcal{V}} = \frac{(\mathrm{dist}(v_{i_r}, v_j))^{-1}}{\sum_{k:\, v_k\in\mathcal{N}_{i_r,r}^{\mathcal{V}}}(\mathrm{dist}(v_{i_r}, v_k))^{-1}},\ \ \forall j:\, v_j\in\mathcal{N}_{i_r,r}^{\mathcal{V}}.$$

The normalisation guarantees that $\sum_{j:\, v_j \in \mathcal{N}^{\mathcal{V}}_{i_r,r}} a^{\mathcal{V}}_{j,r} = 1$. The reason for using inverse distance as the weight construction is that inverse distance weighted averaging has a correspondence with the linear interpolation for the embedding of the tree in a Euclidean space (Jansen et al.; 2009).

Different from the prediction filters, the update filters $\{b^{\mathcal{V}}_{j,r}\}_{j \in \mathcal{N}^{\mathcal{V}}_{i_r,r}}$ are constructed based upon the primal framework. If we integrate equation (1.5.16), we have

$$0 = I^{\mathcal{V}}_{i_r,r} - \sum_{j:\, v_j \in \mathcal{N}^{\mathcal{V}}_{i_r,r}} b^{\mathcal{V}}_{j,r} I^{\mathcal{V}}_{j,r-1}. \tag{1.5.17}$$

The reason for the left-hand side to be zero is the admissibility condition for wavelets, $\int_{\mathcal{V}} \psi^{\mathcal{V}}_{i_r}(v)dv = 0$. We can see that when $|\mathcal{N}^{\mathcal{V}}_{i_r,r}| \geq 2$, then equation (1.5.17) is an underdetermined system. Jansen et al. (2004, 2009) suggested to solve the equation by the minimum norm solution, such that

$$b^{\mathcal{V}}_{j,r} = \frac{I^{\mathcal{V}}_{i_r,r} I^{\mathcal{V}}_{j,r-1}}{\sum_{k:\, v_k \in \mathcal{N}^{\mathcal{V}}_{i_r,r}} \left(I^{\mathcal{V}}_{j,r-1}\right)^2}, \;\; \forall j :\, v_j \in \mathcal{N}^{\mathcal{V}}_{i_r,r},$$

which provides a stabilised update step.

1.5.3 Scales and Artificial Levels

Compared with classical wavelets and the general lifting scheme, the LOCAAT framework lacks the notion of a dyadic scale. Jansen et al. (2009) suggested to consider scale as a continuous concept instead, and a convenient measure is the scaling function integral discussed above. Therefore, for the detail coefficient $d^{\mathcal{V}}_{i_r}$ obtained at stage-r, its scale is $I^{\mathcal{V}}_{i_r,r}$. Then artificial levels can be constructed, thus providing a partitioning of detail coefficients based on their scales $\{I^{\mathcal{V}}_{i_r,r}\}_{r \in \{n,\ldots,3\}}$. The artificial levels are constructed as follows. Let α_J be the $(1 - 2^{-J})$-quantile for some $J \in \mathbb{Z}^+$, e.g., α_1 is the median of the sequence. Without loss of generality, we denote $\alpha_0 = 0$, then the J-th artificial level is constructed as the range $(\alpha_{J-1}, \alpha_J]$ and we say a detail coefficient $d^{\mathcal{V}}_{i_r}$ belongs to the J-th artificial level if and only if $I^{\mathcal{V}}_{i_r} \in (\alpha_{J-1}, \alpha_J]$.

1.5.4 Variance Approximation

Now let us consider the variance of the detail coefficients given by the LOCAAT transform. Suppose the initial scaling coefficients $\{c^{\mathcal{V}}_{i,n}\}$ are *i.i.d* normally distributed random variables with finite variance $\sigma^2 < \infty$. For the k-th initial scaling coefficient, we denote its variance as $V_{k,n}\,\sigma^2$, for all $k \in \{1, ..., n\}$. Recall that the prediction step in equation (1.5.3) is linear, then after the prediction step, we have

$$\begin{aligned} V^*_{i_n}\,\sigma^2 := \mathrm{Var}(d^{\mathcal{V}}_{i_n}) &= \mathrm{Var}\left(c^{\mathcal{V}}_{i_n,n} - \sum_{j:\, v_j \in \mathcal{N}^{\mathcal{V}}_{i_n,n}} a^{\mathcal{V}}_{j,n} c^{\mathcal{V}}_{j,n}\right) \\ &= \mathrm{Var}(c^{\mathcal{V}}_{i_n,n}) + \sum_{j:\, v_j \in \mathcal{N}^{\mathcal{V}}_{i_n,n}} (a^{\mathcal{V}}_{j,n})^2\,\mathrm{Var}(c^{\mathcal{V}}_{j,n}) \\ &= \left(V_{i_n,n} + \sum_{j:\, v_j \in \mathcal{N}^{\mathcal{V}}_{i_n,n}} (a^{\mathcal{V}}_{j,n})^2\, V_{j,n}\right)\sigma^2. \\ \Rightarrow V^*_{i_n} &= V_{i_n,n} + \sum_{j:\, v_j \in \mathcal{N}^{\mathcal{V}}_{i_n,n}} (a^{\mathcal{V}}_{j,n})^2\, V_{j,n}. \end{aligned} \tag{1.5.18}$$

Since $d^{\mathcal{V}}_{i_n}$ is correlated with $c^{\mathcal{V}}_{j,n}$, $\forall j :\, v_j \in \mathcal{N}^{\mathcal{V}}_{i_n,n}$, then we have

$$\mathrm{cov}(d^{\mathcal{V}}_{i_n}, c^{\mathcal{V}}_{j,n}) = -a^{\mathcal{V}}_{j,n} V_{j,n}\,\sigma^2, \;\; \forall j :\, v_j \in \mathcal{N}^{\mathcal{V}}_{i_n,n}. \tag{1.5.19}$$

According to the update step as in equation (1.5.4), along with the variance in equation (1.5.18) and the covariance in equation (1.5.19), we have that

$$\begin{aligned} V_{j,n-1}\,\sigma^2 := \mathrm{Var}(c^{\mathcal{V}}_{j,n-1}) &= \mathrm{Var}\left(c^{\mathcal{V}}_{j,n} + b^{\mathcal{V}}_{j,n} d^{\mathcal{V}}_{i_n}\right) \\ &= \mathrm{Var}(c^{\mathcal{V}}_{j,n}) + (b^{\mathcal{V}}_{j,n})^2 \mathrm{Var}(d^{\mathcal{V}}_{i_n}) + 2b^{\mathcal{V}}_{j,n}\mathrm{cov}(d^{\mathcal{V}}_{i_n}, c^{\mathcal{V}}_{j,n}) \\ &= V_{j,n}\,\sigma^2 + (b^{\mathcal{V}}_{j,n})^2 V^*_{i_n}\,\sigma^2 + 2b^{\mathcal{V}}_{j,n}(-a^{\mathcal{V}}_{j,n} V_{j,n}\,\sigma^2) \\ &= \left((1 - 2a^{\mathcal{V}}_{j,n} b^{\mathcal{V}}_{j,n}) V_{j,n} + (b^{\mathcal{V}}_{j,n})^2 V^*_{i_n}\right)\,\sigma^2, \;\; \forall j :\, v_j \in \mathcal{N}^{\mathcal{V}}_{i_n,n}. \\ \Rightarrow V_{j,n-1} &= (1 - 2a^{\mathcal{V}}_{j,n} b^{\mathcal{V}}_{j,n}) V_{j,n} + (b^{\mathcal{V}}_{j,n})^2 V^*_{i_n}. \end{aligned} \tag{1.5.20}$$

Note that the variance magnitudes $\{V^*_{i_r}\,\sigma^2\}_{r\in\{n,...,3\}}$ for wavelet coefficients $\{d^{\mathcal{V}}_{i_r}\}_{r\in\{n,...,3\}}$ are different. However, for wavelet thresholding (which will be discussed in Section 1.6.1), the homogeneity of variance is normally essential. Thus, we have to normalise the variance after the lifting scheme. One possible choice is to normalise the wavelet coefficients

such that the i_r-th normalised wavelet coefficient is $\tilde{d}^{\mathcal{V}}_{i_r} = d^{\mathcal{V}}_{i_r}/\sqrt{V^*_{i_r}}$. Hence, all normalised wavelet coefficients are with (approximately) equal variances, such that $\mathrm{Var}(\tilde{d}^{\mathcal{V}}_{i_r}) \approx \sigma^2$. The inaccuracy happens since we ignore the covariance at the next stage. Nevertheless, this strategy will only lose a small precision. Another advantage of this strategy is the computational feasibility for obtaining $\{V^*_{i_r}\}_{r\in\{n,\dots,3\}}$, we only have to perform the lifting scheme (at the same time with the scaling coefficients) for the vector $(V_{1,n},\dots,V_{n,n})^T = \mathbf{1}$.

1.6 Nonparametric Regression

Consider we have the following data set $\{(x_i, f_i)\}_{i=1}^n$, which can be modelled as being additively contaminated by noise $\{\epsilon_i\}_{i=1}^n$

$$f_i = g(x_i) + \epsilon_i, \tag{1.6.1}$$

where $x_i \in \Omega$ is the data location from a certain bounded domain Ω, and $g : \Omega \longrightarrow \mathbb{R}$ is a 'true function'. The noise is assumed to be a set of *i.i.d* normal distribution, $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$, where σ^2 is finite. The bounded domain can be one-dimensional line (Nunes et al.; 2006), high-dimensional Euclidean space, or graph/network domain (Jansen et al. (2009) and Mahadevan (2010)). For example, for the LOCAAT framework we discussed above, the domain is $\Omega = \mathcal{V}$ for a graph $G = (\mathcal{V}, \mathcal{E})$. Our goal is to denoise f, and obtain an estimator $\hat{g}$ for the true (unknown) function g.

1.6.1 Estimation by Wavelet Shrinkage

Donoho and Johnstone (1994) introduced a non-linear smoothing method based on wavelet techniques, which is wavelet shrinkage. This method has become very popular among both the signal processing and statistics communities in the past thirty years due to its theoretical support and the convenience of implementation. In this section, we briefly describe some well-established classic wavelet shrinkage methods, which are designed for data satisfies that

- The number of the observations is dyadic, denoted as 2^J, for some $J \in \mathbb{Z}^+$.

- The locations of the observations are equally spaced in one dimension, for example, $x_i = i/n$ in the interval $[0, 1]$.

For the set of (noisy) observations $\underline{f} = \{f_i\}_{i=1}^n$ which follow the model (1.6.1), the wavelet shrinkage is described by the following steps.

1. We first perform the discrete wavelet transform to the observations $\underline{f}$, which gives us two sets of (wavelet and scaling) coefficients, $\{d_{j,k}\}_{j_0 \le j < J;\, j,k \in \mathbb{Z}}$ and $\{c_{j_0,k}\}_{k \in \mathbb{Z}}$, where $j_0 \in \mathbb{Z}$ is the primary resolution. For e.g. the Daubechies' family of wavelets, which generates orthonormal bases, the wavelet coefficients can be represented as

$$d_{j,k} = d^\star_{j,k} + \varepsilon_{j,k}, \quad j, k \in \mathbb{Z};\, j \ge j_0 \tag{1.6.2}$$

   where $d^\star_{j,k}$ is the true wavelet coefficients and due to the orthonormality, we have $\varepsilon_{j,k} \overset{\text{iid}}{\sim} N(0, \sigma^2)$, see Donoho and Johnstone (1994).

2. An estimator of each $d^\star_{j,k}$, denoted as $\hat{d}^\star_{j,k}$, will be obtained by a chosen thresholding rule, then we will have the set of the estimated wavelet coefficients $\{\hat{d}^\star_{j,k}\}_{j \ge j_0;\, j,k \in \mathbb{Z}}$.

3. Perform the inverse transform for $\{\hat{d}^\star_{j,k}\}_{j \ge j_0;\, j,k \in \mathbb{Z}}$ and obtain the estimates $\{\hat{g}_i\}_{i=1}^n$ for the unknown function g.

1.6.2 Thresholding Strategy

Now let us give some examples of existing popular thresholding strategies.

1.6.2.1 Hard- and Soft-thresholding

The hard- and soft-thresholding were introduced by Donoho and Johnstone (1994, 1995) and have the form

$$\hat{d}^{\text{hard}}_{j,k} = d_{j,k} \cdot \mathbb{I}(|d_{j,k}| > \lambda),$$
$$\hat{d}^{\text{soft}}_{j,k} = \text{sgn}(d_{j,k})(|d_{j,k}| - \lambda)\mathbb{I}(|d_{j,k}| > \lambda),$$

where 'sgn' is the signum function, $\lambda > 0$ is a certain threshold value, and $\mathbb{I}(\cdot)$ is the indicator function. Both hard- and soft-thresholding are non-linear smoothing methods depending on the threshold value λ. The popular choice is the *universal threshold* introduced by Donoho and Johnstone (1994), which is

$$\lambda^u = \sigma\sqrt{2\log n}, \tag{1.6.3}$$

where σ is the standard deviation of the noise, and n is the number of data points. Vidakovic (2009) recalled the following theorem from Pickands (1967).

Theorem 3. If $\{Z_i\}_{i\in\mathbb{N}}$ is a stationary Gaussian process, where $\mathbb{E}(Z_i) = 0$, $\mathbb{E}(Z_i^2) = 1$ and $\mathbb{E}(Z_i\, Z_{i+k}) = \gamma(k)$ with $\lim_{k\to\infty}\gamma(k) = 0$, then we have $(\max_{i=1}^n |Z_i|)/\sqrt{2\log n} \to 1$, almost surely, when $n \to \infty$.

Hence, for the noise $\varepsilon_{j,k} \overset{\text{iid}}{\sim} N(0,\sigma^2)$ in the wavelet domain, by Theorem 3, we have

$$\lim_{n\to\infty} \mathbb{P}(\max_{j,k} |\varepsilon_{j,k}| > \sigma\sqrt{2\log n}) = 0.$$

Notice that this result holds for dependent noise, see Theorem 3.

1.6.2.2 Empirical Bayes Thresholding

Note that the basic idea behind wavelet thresholding is to find an oracle such that the wavelet coefficients can be separated into two groupings: the wavelet coefficients we should choose for function reconstruction (likely to have high true absolute value in wavelet domain), and those negligible (likely to have low true absolute value in wavelet domain), see Donoho and Johnstone (1994) and Nason (2008). It is easy to see that the threshold choice is very important in this process. Johnstone and Silverman (2004) show that the optimal size of the threshold is highly dependent on the sparsity of the true wavelet coefficients. However, this sparsity is usually unknown in applications. Hence, a thresholding strategy which is self-adaptive to different sparsity levels (in the wavelet domain) will be desirable.

Johnstone and Silverman (2004) proposed the *empirical Bayes thresholding* to tackle this problem. Based on equation (1.6.2) and $\varepsilon_{j,k} \overset{\text{iid}}{\sim} N(0,\sigma^2)$, we have that $d_{j,k}|d^\star_{j,k} \sim$

$N(d^\star_{j,k}, \sigma^2)$. Johnstone and Silverman (2004) noticed that in practice, the significant wavelets coefficients are likely to come from a heavy-tailed distribution, hence the prior distribution f_{prior} for $d^\star_{j,k}$ can be written as

$$f_{\text{prior}}(d^\star) = (1-\nu)\delta_0(d^\star) + \nu\pi(d^\star), \tag{1.6.4}$$

where $0 < \nu < 1$ is a constant, δ_0 is the Dirac delta for modelling the zero-valued wavelet coefficients. The non-zero wavelet coefficients are captured by a fixed unimodal symmetric density $\pi(\cdot)$. Unlike other Bayesian based shrinkage methods, Johnstone and Silverman (2004) suggested to model these non-zero wavelet coefficients by a heavy-tailed distribution instead of a zero-mean normal distribution, for example, Laplace distribution or Cauchy distribution.

An alternative strategy is to allow a parametric dependence, hence the prior distribution will be determined by a level-dependent ratio ν_j, see Johnstone and Silverman (2004) and Jansen et al. (2009). In this work, we only consider the case that ν_j is a constant within levels.

Once the distribution is fixed, we need to estimate the ratio ν. Johnstone and Silverman (2004) suggest performing marginal maximum likelihood estimation for ν. Let h be the convolution of the density π and a standard normal density ϕ, then the marginal density of $Z \sim N(d^\star, 1)$ can be written as

$$(1-\nu)\phi(z) + \nu h(z),$$

Then the estimator $\hat{\nu}$ can be obtained by the maximizer of the marginal log-likelihood, see Johnstone and Silverman (2004). Then the posterior distribution can be calculated for each observation Z_i, which is $f_{\text{post}}(d^\star | Z_i = z_i)$. The estimator $\hat{d^\star}$ for the true wavelet coefficients $d^\star$ can be obtained by hard/soft thresholding, or posterior mean/median, see Johnstone and Silverman (2004) for more details.

1.6.2.3 Estimating the Noise Level

In practice, the noise has an unknown variance, which in turn has to be first estimated before applying thresholding. We follow the suggestion from Donoho and Johnstone

(1994), to estimate the standard deviation by the median absolute deviation (MAD) of the finest level-$(J-1)$ wavelet coefficients, which is

$$\hat{\sigma} = \mathbf{MAD}\{d_{J-1,k}\} = \text{median}\{|d_{J-1,k} - \text{median}\{d_{J-1,k}\}|\}/0.6745,$$

where $d_{J-1,k}$ is the finest level (observed) wavelet coefficients. The advantage of the MAD estimator is its robustness. For the LOCAAT algorithm, the standard deviation can be estimated by using the largest artificial level instead of the 'finest level'.

Chapter 2

Line Graph LOCAAT

This chapter introduces an approach to carry out multiscale analysis on data collected from the edges of a network. The method we propose involves initially applying a line graph transform in order to shift emphasis from edges to (new) vertices, followed by a version of the LOCAAT on the vertices in the line graph domain. Therefore, we refer to our proposed technique as **L**ine **G**raph LOCAAT, or LG-LOCAAT.

This chapter will be organised as follows. We firstly introduce our proposed LG-LOCAAT algorithm with both theoretical and computational aspects. Then a simulation study will be implemented to test the performance of our algorithm. Finally, we will discuss the advantages and disadvantages of our methods based on both theory and simulation results.

For the methodology of LG-LOCAAT, we first focus on the line graph transform and the associated metric defined on the line graph. We then present the proposed LG-LOCAAT transform. In the line graph domain, the algorithm can (mostly) follow the steps of LOCAAT; however, the relink step will be emphasised in this context, since the transformation of the line graph can produce a non-tree graph.

A simulation study is presented that consists of evaluating three different aspects of our algorithm's performance: (i) data compression ability (via sparsity plots); (ii) stability of the transform (via the condition number); and (iii) denoising performance

(via the average mean squared error (AMSE)). Details on these related measures will be discussed in the associated sections in this chapter.

Crovella and Kolaczyk (2003) was one of the first works that came up with the idea of representing edge data by line graphs without fully developing the idea further. Evans and Lambiotte (2009) consider line graph as a tool for edge clustering, enhancing our confidence on the potential of wavelets lifting on line graph since multiscale methods have a strong connection with cluster analysis, see Starck et al. (1998). Now let us start by introducing essential knowledge on line graphs.

2.1 Line Graph Transform

In this section, we introduce the line graph transform, which leads to a graph structure that fits the existing LOCAAT algorithm that operates on vertices. The term 'line graph' is sometimes used to mean the general 'graph' structure ('line' represents 'edge') in some of the works from the signal processing community, see for example, Ortega et al. (2018). In our work, the term 'line graph' will be used to represent the definition within graph theory, see for example, Bondy and Murty (2008), Bollobás (1998) and Diestel (2005).

Specifically, the line graph of a graph G, denoted $\mathbf{LG}(G)$, is the graph whose vertex set is bijective to the edge set $\mathcal{E}$ of G. For convenience, we write $\mathbf{LG}(G) = G^* = (\mathcal{V}^*, \mathcal{E}^*)$, and we have $\mathcal{V}^* \longleftrightarrow \mathcal{E}$, where '$\longleftrightarrow$' indicates a bijection between two sets. Moreover, we simply let a new vertex v_k^* correspond to the k-th original edge $e_k \in \mathcal{E}$, for any $k \in \{1, ..., |\mathcal{E}|\}$. Although the correspondence of subscripts can be any permutation, it will not affect the result of our work since LOCAAT does not rely on this ordering. Using notation established in Section 1.4.1, the new edge set $\mathcal{E}^*$ can be defined as

$$\mathcal{E}^* = \left\{ \{v_k^*, v_l^*\} \,|\, e_k, e_l \in \mathcal{E} \text{ and } |e_k \cap e_l| = 1 \right\}.$$

Here the cardinality of the intersection of two edges being equal to one indicates that e_k and e_l share only one common vertex. Figure 2.1 shows an example of the line graph transformation.

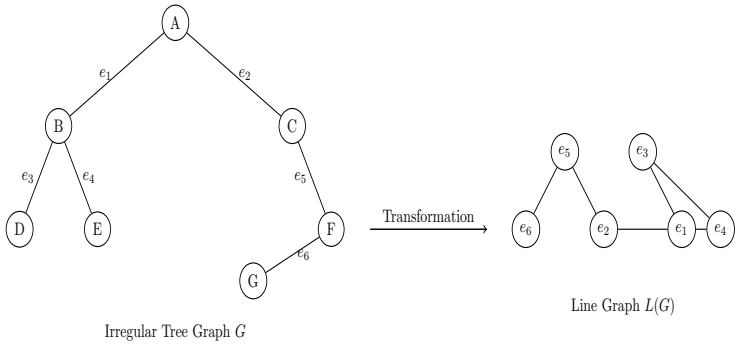

Figure 2.1: A visualisation for the line graph transform.

In addition, the neighbourhood $\mathcal{N}_k^{\mathcal{V}^*}$ of a vertex v_k^* in the line graph is defined as the set of the vertices which are connected with v_k^* by an edge $e^* \in \mathcal{E}^*$. Mathematically, for the edge v_k^*, its neighbourhood $\mathcal{N}_k^{\mathcal{V}^*}$ can be represented as

$$\mathcal{N}_k^{\mathcal{V}^*} := \{v_s^* \in \mathcal{V}^* \,|\, \{v_k^*, v_s^*\} \in \mathcal{E}^*\} \tag{2.1.1}$$

$$= \{v_s^* \in \mathcal{V}^* \,|\, s \neq k;\, e_k \cap e_s \neq \emptyset;\, e_k, e_s \in \mathcal{E}\} \tag{2.1.2}$$

Line graphs and their associated properties have been well-studied in the graph theory literature and for more details the reader can refer to Bondy and Murty (2008) and Harary (2018).

2.2 Line Graph Distance Measure

Note that line graphs only reveal combinatorial information between edges. To perform LOCAAT on a line graph, we need to define a 'distance' between new vertices (edges), as we recall from Section 1.5, e.g., the prediction weight is designed by means of the inverse distance between the vertices (here, in the new line graph). We distinguish between the following two situations:

1. We have the coordinate information for the graph G.

2. Coordinate information is not available, but the lengths of the edges in $\mathcal{E}$ are available.

For the first case, the original network can be naturally assumed to be located in a Euclidean space, while for the second one, the associated metrized (original) graph can be obtained. Recall a Euclidean space along with the Euclidean distance forms a metric space, and the metrized graph with shortest path distance forms a metric space. We remind the reader the definition of metric subspace (O'Searcoid; 2006).

Definition 2.2.1. If we have a set A and (A, dist) is a metric space, let $B \subset A$ be a subset of A, and $\text{dist}_{|B}$ as a restriction of dist, such that $\text{dist}_{|B}$ holds only for the set B, and for any $b_1, b_2 \in B$, we have $\text{dist}_{|B}(b_1, b_2) = \text{dist}(b_1, b_2)$. Then $(B, \text{dist}_{|B})$ is a metric subspace.

Since a metric subspace is itself a metric space (O'Searcoid; 2006), a convenient way to define the distance measure for $\mathcal{E}$ is to find a set of points $\{\mathbf{p}_k\}_{k=1}^{|\mathcal{E}|}$ (points in a Euclidean space for case 1, and points in the metrized graph space Γ for case 2) to represent $\mathcal{E}$, and to take the associated distance. For each edge e_k, we use its middle point, denoted as $\mathbf{p}_k$, to represent this edge. Then for the first case, let $S = \{\mathbf{p}_k\}_{k=1}^{|\mathcal{E}|}$, and we have that $(S, \text{dist}_{\text{E},|S})$ forms a metric space, where 'E' indicates Euclidean distance; for the second case, the shortest path distance will be used to determine the metric space. As S has a bijection with the new vertex set $\mathcal{V}^*$, we let v_k^* be the new vertex corresponding to $\mathbf{p}_k$. We further define a distance $\text{dist}_{\mathcal{V}^*}$, where $\text{dist}_{\mathcal{V}^*}(v_i^*, v_j^*) = \text{dist}_{\text{E},|S}(\mathbf{p}_i, \mathbf{p}_j)$ for the first case, or $\text{dist}_{\mathcal{V}^*}(v_i^*, v_j^*) = \text{dist}_{\text{path},|S}(\mathbf{p}_i, \mathbf{p}_j)$ for the second case. Then the pair $(\mathcal{V}^*, \text{dist}_{\mathcal{V}^*})$ is a metric space.

The rationale behind using the Euclidean distance is the fact that a graph can be considered as the discrete approximation of a manifold, which is a topological space that is locally homeomorphic to an Euclidean space $\mathbb{R}^p$ for some p, see Tu (2011) for more details on the manifold structure, and see Zhou and Burges (2008) and Cao et al. (2024) for further discussion on the connection between graphs and manifolds.

However, these are not the only two ways to define the distance between edges. For example, if we consider each metrized edge as a planar curve, then one might choose

Fréchet distance or Hausdorff distance (Alt et al.; 2004). For more details about various distance measures, the reader can refer to Dryden and Mardia (2016).

2.3 Line Graph LOCAAT (LG-LOCAAT)

In this section, we will describe in detail our proposed LG-LOCAAT algorithm, including some theoretical characteristics in terms of the associated function expansion, sparsity and stability, as well as the computational aspects of the algorithm.

2.3.1 Function Representation and Initial Lifting Functions Setup

Before discussing the line graph LOCAAT algorithmic steps, the details must first be understood from the function representation perspective. Following the notation of Diestel (2005), suppose that we have a real-valued function $g^{\mathcal{E}} \in \mathfrak{E}$, where $\mathfrak{E}$ is the vector space that contains all functions that map $\mathcal{E}$ to $\mathbb{R}$, where $\mathcal{E}$ is the edge set of a graph $G = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = \{v_1, ..., v_n\}$ and $\mathcal{E} = \{e_1, ..., e_m\}$. The associated line graph $\mathbf{LG}(G) = (\mathcal{V}^*, \mathcal{E}^*)$ can be obtained by taking the line graph transform as we discussed in Section 2.1, and we denote $\mathbf{LG}(G)$ as G^* for convenience. Similarly, a vector space $\mathfrak{V}^*$ containing all functions defined on the new vertex set can also be defined. The bijection between $\mathcal{E}$ and $\mathcal{V}^*$ indicates that the two vector spaces $\mathfrak{E}$ and $\mathfrak{V}^*$ are isomorphic, denoted as $\mathfrak{E} \approx \mathfrak{V}^*$, see Roman et al. (2005). Thus, we can find a function $g^{\mathcal{V}^*} \in \mathfrak{V}^*$, where $g^{\mathcal{V}^*}(v_k^*) = g^{\mathcal{E}}(e_k)$ if v_k^* is the image of e_k according to the line graph transform. We denote $g^{\mathcal{V}^*} \equiv g^{\mathcal{E}}$ if they satisfy this condition. This isomorphism has many advantages for our construction. First of all, since additivity and scalar multiplication are maintained in isomorphic vector spaces (Roman et al.; 2005), performing a linear combination of a set of vectors $\{\mathbf{u}_s \in \mathfrak{V}^*\}_{s=1}^p$ is equivalent to performing the same linear combination of a set of vectors $\{\mathbf{w}_s \in \mathfrak{E}\}_{s=1}^p$, where $\mathbf{u}_s \equiv \mathbf{w}_s$. Secondly, if $\{\mathbf{u}_s\}_{s=1}^p$ is a basis defined on $\mathfrak{V}^*$, then $\{\mathbf{w}_s\}_{s=1}^p$ is a basis defined on $\mathfrak{E}$, and $\mathbf{u}_s \equiv \mathbf{w}_s$. As a result, it is feasible to design a multiresolution analysis for $g^{\mathcal{E}}$ based on $\mathcal{V}^*$.

Now let us consider the design for the initial primal and dual scaling functions. Incorporating the results from Sweldens (1996b, 1998) and the LOCAAT framework (Section 1.5), we summarise the conditions for suitably designing primal and dual scaling functions (for the vertex set $\mathcal{V}^*$ of the graph G^*) as follows. Firstly, the set of dual scaling functions $\{\tilde{\varphi}_{k,m}^{\mathcal{V}^*}\}_{k=1}^m$ should satisfy $\langle \tilde{\varphi}_{k,m}^{\mathcal{V}^*}, g^{\mathcal{V}^*} \rangle = g_k^{\mathcal{V}^*}$, where $\{g_k^{\mathcal{V}^*}\}_{k=1}^m$ is the set of observations on the function $g^{\mathcal{V}^*}$, defined on the new vertex set $\mathcal{V}^*$. This condition allows us to start with the observation values, as the initial scaling coefficients, and then perform the lifting scheme. The primal scaling functions $\{\varphi_{k,m}^{\mathcal{V}^*}\}_{k=1}^m$ are designed such that for each $k \in \{1, ..., m\}$, we have $\varphi_{k,m}^{\mathcal{V}^*}(v_k^*) = 1$ and $\varphi_{k,m}^{\mathcal{V}^*}(v_s^*) = 0$ for all $s \neq k$. The set of primal and dual initial scaling functions has to satisfy the biorthogonality property, such that $\langle \tilde{\varphi}_{k,m}^{\mathcal{V}^*}, \varphi_{k',m}^{\mathcal{V}^*} \rangle = \delta_{kk'}$, where $\delta_{kk'}$ is the Kronecker delta.

Recall that for a graph G, we can obtain a line graph $\mathbf{LG}(G) = (\mathcal{V}^*, \mathcal{E}^*)$ with an associated metric space $(\mathcal{V}^*, \mathrm{dist}_{\mathcal{V}^*})$. Then naturally we can generate the metrized version of $\mathbf{LG}(G)$, denoted as $\Gamma(\mathbf{LG}(G))$, or Γ^* for convenience, along with the distance measure $\mathrm{dist}_{\mathcal{V}^*}(v_i^*, v_j^*)$ as the length, ℓ_k, of the metrized version, $e_k^{*\mathrm{met}}$, for the edge $e_k^* = \{v_i^*, v_j^*\}$, which gives us a continuous analogue of the line graph $\mathbf{LG}(G)$. We denote a point on the metrized graph space as $\mathbf{p}^* \in \Gamma^*$, and particularly, we denote $\mathbf{p}_{v_k^*}^*$ as the point that corresponds to v_k^*. Then we can define the set of partitionings, $\{\mathbf{P}_s^*\}_{s\in\{1,...,m\}}$, of the metrized line graph such that

$$\mathbf{P}_s^* = \left\{ \mathbf{p}^* \,|\, \mathbf{p}^* \in \Gamma^*;\, \mathrm{dist}(\mathbf{p}^*, \mathbf{p}_{v_s^*}^*) < \mathrm{dist}(\mathbf{p}^*, \mathbf{p}_{v_{s'}^*}^*),\ \forall s' \in \{1, ..., m\} \backslash \{s\} \right\}. \tag{2.3.1}$$

Clearly, the middle point of each metrized edge $e^{*\mathrm{met}} \in \Gamma^*$ separates this edge into two distinct partitionings. Then based on the partitioning and the conditions above, we let the dual and primal scaling functions to be

$$\begin{aligned} \tilde{\varphi}_{s,m}^{\Gamma^*}(\mathbf{p}^*) &= \delta(\mathbf{p}^* - \mathbf{p}_{v_s^*}^*); \\ \varphi_{s,m}^{\Gamma^*}(\mathbf{p}^*) &= \chi_{\mathbf{P}_s^*}(\mathbf{p}^*), \end{aligned}$$

where $\chi_{\mathbf{P}_s^*}$ is the characteristic function defined on the s-th block of the partitioning, and $\delta(\mathbf{p}^* - \mathbf{p}_{v_s^*}^*)$ is the Dirac delta with the energy centred on the point $\mathbf{p}_{v_s^*}^*$. We write

Γ^* the superscript for these scaling functions to point out that they are defined on the metrized line graph space. The advantage of the characteristic functions is that they can be used to reveal the geometric information of the partitionings. We also consider another setup for the dual and primal scaling functions, which is

$$\tilde{\varphi}^{\Gamma^*}_{s,m}(\mathbf{p}^*) = \delta(\mathbf{p}^* - \mathbf{p}^*_{v^*_s}); \tag{2.3.2}$$

$$\varphi^{\Gamma^*}_{s,m}(\mathbf{p}^*) = \delta_{\mathbf{p}^*_{v^*_s},\mathbf{p}^*}, \tag{2.3.3}$$

where $\delta_{\mathbf{p}^*_{v^*_s},\mathbf{p}^*}$ is the Kronecker delta. The construction using equations (2.3.2) and (2.3.3) can be considered akin to the *lazy wavelets* introduced by Sweldens (1996b), but on the (new) vertex set. Then the initial expansion form of the function approximation can be written as

$$\begin{aligned} g^{\mathcal{E}}(e) \equiv g^{\mathcal{V}^*}(v^*) &= g^{\Gamma^*}(\mathbf{p}^*) \\ &= \sum_{s=1}^{m} c^{\Gamma^*}_{s,m}\varphi^{\Gamma^*}_{s,m}(\mathbf{p}^*). \end{aligned} \tag{2.3.4}$$

where $\mathbf{p}^* \in \Gamma^*$ and g^{Γ^*} is the metrized analogue of the function $g^{\mathcal{V}^*}$, and the (initial) scaling coefficients are $c^{\Gamma^*}_{s,m} = \langle g^{\Gamma^*}, \tilde{\varphi}^{\Gamma^*}_{s,m}\rangle = g^{\Gamma^*}_s = g^{\mathcal{E}}_s$. From this point onwards, we use $g^{\mathcal{V}^*}$ to be interchangeable with g^{Γ^*}. The aim is to find the function approximation as follows

$$g^{\Gamma^*}(\mathbf{p}^*) = \sum_{s\in\mathcal{S}_2} c^{\Gamma^*}_{s,2}\varphi^{\Gamma^*}_{s,2}(\mathbf{p}^*) + \sum_{l\in\mathcal{D}_2} d^{\Gamma^*}_l \psi^{\Gamma^*}_l(\mathbf{p}^*), \tag{2.3.5}$$

where $\mathcal{D}_2 = \{k_m, ..., k_3\}$, and $\mathcal{S}_2 = \{1, ..., m\}\backslash\mathcal{D}_2$. The set $\{d^{\Gamma^*}_l\}_{l\in\mathcal{D}_2}$ holds the detail (wavelet) coefficients, which can be obtained at each stage-r along with the scaling coefficients $\{c^{\Gamma^*}_{s,r}\}_{s\in\mathcal{S}_r}$ by our proposed algorithm next detailed.

2.3.2 LG-LOCAAT Algorithm

In this section, we present our proposed algorithm in terms of the split-predict-update-relink procedure. Suppose that we have a set of observations, $\{g^{\mathcal{E}}_k\}_{k\in\{1,...,m\}}$, which are collected from the graph edges. As we discussed in the previous section, the observations can be considered as $\{g^{\Gamma^*}_k\}_{k\in\{1,...,m\}}$ in the metrized line graph domain. Similar to Jansen

et al. (2009), we start from stage-m, corresponding to the original edge-based observations, and the initial scaling coefficients are defined as $c^{\Gamma^*}_{k,m} := \langle g^{\Gamma^*}, \tilde{\varphi}^{\Gamma^*}_{s,m} \rangle = g^{\Gamma^*}_k = g^{\mathcal{E}}_k$. The *initial* neighbourhood structure can be represented as described in equation (2.1.1) or (2.1.2). Here, we write $\mathcal{N}^{\mathcal{V}^*}_{k,m}$ instead of $\mathcal{N}^{\mathcal{V}^*}_k$ since the neighbourhood structure will be changed as the algorithm progresses through stage-m, stage-$(m-1)$, and so on.

- **Split:** In line with Jansen et al. (2009), the criterion of the split step is to choose the new vertex, denote it by $v^*_{k_m}$, according to minimum integral value for the *primal* scaling function associated with its metrized point, $\mathbf{p}^*_{v^*_{k_m}}$. When the initial primal scaling functions are defined as characteristic functions on the partitionings of the metrized line graph, the initial integral values are

$$\begin{aligned} I^{\Gamma^*,\text{sum}}_{k,m} &= \int_{\Gamma^*} \varphi^{\Gamma^*}_{k,m}(\mathbf{p}^*) d\mathbf{p}^* \\ &= \int_{\Gamma^*} \chi_{\mathbf{P}^*_k}(\mathbf{p}^*) d\mathbf{p}^* \\ &= \mu(\mathbf{P}^*_k) \\ &= \sum_{s: v^*_s \in \mathcal{N}^{\mathcal{V}^*}_{k,m}} \frac{\ell^*_s}{2} \\ &\propto \sum_{s: v^*_s \in \mathcal{N}^{\mathcal{V}^*}_{k,m}} \text{dist}(v^*_k, v^*_s), \end{aligned} \tag{2.3.6}$$

  where we use the Lebesgue measure μ for a union of intervals (or line segments) to be the summation of their lengths, and '$\propto$' means 'proportional to'. The resulting integral is proportional to the sum of distances to the new neighbouring vertices of the k-th new vertex. Computationally, using a proportional sum of distances will give the same result (in terms of detail coefficients) as using the sum of distances according to the following proposition.

  Proposition 2.3.1. Suppose we have an integral sequence $\underline{I}^* = \{I^*_{k,m}\}_{k \in \{1,\dots,m\}}$, and a constant $C > 0$. Then, performing the LOCAAT algorithm with $C \cdot \underline{I}^*$ as integrals will yield the same detail coefficients and the same prediction/update filters, as performing LOCAAT with $\underline{I}^*$.

The proof is presented in Appendix C.1. Jansen et al. (2009) also suggested the corresponding average distance as a possible integral determination for initial scaling functions. The average distance can be obtained by

$$I_{k,m}^{\Gamma^*,\text{ave}} = \frac{1}{2|\mathcal{N}_{k,m}^{\mathcal{V}^*}|} \sum_{s:v_s^* \in \mathcal{N}_{k,m}^{\mathcal{V}^*}} \text{dist}(v_k^*, v_s^*). \tag{2.3.7}$$

In our implementation, we will also test the performance of the algorithm using the average distance as the integral since previous literature indicates its good performance, see also Mahadevan (2010).

As we discussed in the previous subsection, the initial primal scaling functions can also be set as Kronecker delta, which leads to a variant of the lazy wavelets introduced in Sweldens (1998). However, the Kronecker delta is a discrete paradigm which does not have an 'integral'. For tackling this problem, we use the inner product instead of the integral. Let $\underline{\delta}_s = (0, \cdots, 0, 1, 0, \cdots, 0)$ as a canonical basis representation for v_s^*, such that only its s-th element is non-zero (one). Then we have

$$\begin{aligned} I_{k,m}^{\Gamma^*,\text{Delta}} &= \langle \underline{\delta}_s, \mathbb{1}_m \rangle \\ &= 1, \end{aligned} \tag{2.3.8}$$

where $\mathbb{1}_m$ is a vector of ones of length m. This inner product can be considered as a discrete analogue of the 'integral of Kronecker delta'.

In the simulation study we will test the performance of these three integral value choices, namely sum of distances (equation (2.3.6), 'sum'), average distance (equation (2.3.7), 'ave'), and starting with a vector of ones (equation (2.3.8), 'Delta'). Once the integral values (of the initial scaling functions) have been decided, choose the new vertex removal corresponding to the minimum integral value. Additionally, if there exist multiple new vertices with the minimum integral value, then we randomly pick one of these vertices. The sequence of integrals will be updated through the stages of the algorithm (which we will discuss in the update step). The LOCAAT framework follows the principle of recursive wavelet construction, which

only needs us to fix the initial scaling function, and the recursive computation of the integrals will be carried out through the iterative process, see Sweldens (1998). Therefore, in what follows, we refer to a general stage-r instead of the original stage-m. The stage-r removal (new) vertex will be denoted by $v^*_{k,r}$ and the integral value of the k-th scaling function will be denoted as $I^*_{k,r}$. From now on, we will skip the superscript indicating the integral determination (sum/ave/Delta) unless necessary.

- **Predict:** Recall that we have a set of function values defined on the new vertices of the metrized line graph Γ^*, which are $\{g_k^{\Gamma^*}\}_{k\in\{1,\dots,m\}}$. Since the initial dual scaling functions are set as Dirac deltas, we can start with $c_{k,m}^{\Gamma^*} := g_k^{\Gamma^*}$, and perform the prediction of the same form as in equation (1.5.3). Thus, the detail coefficient obtained at stage-r is

$$d_{k_r}^{\Gamma^*} = c_{k_r,r}^{\Gamma^*} - \sum_{s:v_s^*\in\mathcal{N}_{k_r,r}^{\mathcal{V}^*}} a_{s,r}^{\Gamma^*} c_{s,r}^{\Gamma^*}, \tag{2.3.9}$$

where $\{a_{s,r}^{\Gamma^*}\}_{s:v_s^*\in\mathcal{N}_{k_r,r}^{\mathcal{V}^*}}$ are the prediction weights. The MRA framework associated with the prediction step can be expressed as in equation (3.2.3),

$$\tilde{\psi}_{k_r}^{\Gamma^*} = \tilde{\varphi}_{k_r}^{\Gamma^*} - \sum_{s:v_s^*\in\mathcal{N}_{k_r,r}^{\mathcal{V}^*}} a_{s,r}^{\Gamma^*} \tilde{\varphi}_{s,r}^{\Gamma^*}.$$

Integrating and letting the left hand side be zero, the prediction weights satisfy

$$\sum_{s:v_s^*\in\mathcal{N}_{k_r,r}^{\mathcal{V}^*}} a_{s,r}^{\Gamma^*} = 1.$$

Then, as suggested in Jansen et al. (2009), we construct the prediction weights by means of the normalised inverse distances (defined as in Section 2.2), where

$$a_{s,r}^{\Gamma^*} = \frac{1/\mathrm{dist}(v_{k_r}^*, v_s^*)}{\sum_{t:v_t^*\in\mathcal{N}_{k_r,r}^{\mathcal{V}^*}} 1/\mathrm{dist}(v_{k_r}^*, v_t^*)}. \tag{2.3.10}$$

In addition to the inverse distance weights, we will also test the simple moving average prediction, where the s-th prediction weight at stage-r is determined as

$$a_{s,r}^{\Gamma^*} = \frac{1}{|\mathcal{N}_{k_r,r}^{\mathcal{V}^*}|}, \tag{2.3.11}$$

where $|\mathcal{N}_{k_r,r}^{\mathcal{V}^*}|$ is the cardinality of the set $\mathcal{N}_{k_r,r}^{\mathcal{V}^*}$.

- **Update:** The update will be performed firstly for the integrals associated with the neighbourhood, such that for $v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}$, we have

$$I_{s,r-1}^{\Gamma^*} = a_{s,r}^{\Gamma^*} I_{k_r,r}^{\Gamma^*} + I_{s,r}^{\Gamma^*}, \quad \text{for } s : v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}.$$

  The second part to be updated is the set of scaling coefficient values corresponding to the neighbourhood $\mathcal{N}_{k_r,r}^{\mathcal{V}^*}$. Thus, we have

$$c_{s,r-1}^{\Gamma^*} = c_{s,r}^{\Gamma^*} + b_{s,r}^{\Gamma^*} d_{k_r}^{\Gamma^*}, \quad \text{for } s : v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}, \tag{2.3.12}$$

  where the update coefficients, $\{b_{s,r}^{\Gamma*}\}$, are obtained by the minimum norm solution as suggested in Jansen et al. (2009), such that

$$b_{s,r}^{\Gamma^*} = \frac{I_{s,r-1}^{\Gamma^*} I_{k_r,r}^{\Gamma^*}}{\sum_{t:v_t^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}} (I_{t,r-1}^{\Gamma^*})^2}, \quad \text{for } s : v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}. \tag{2.3.13}$$

  These update coefficients along with the condition $\sum_{s:v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}} a_{s,r}^* = 1$, guarantee the stability of the transform, see Jansen and Oonincx (2005) and Jansen et al. (2009) for more details.

- **Relink:** A further requirement through the lifting steps is to perform a relink of the graph structure, since the removal of a vertex (and of the associated edges) might disconnect the graph structure. For a tree graph, it is easy to see that the tree will be disconnected by removing a vertex which is not on the boundary (the boundary of a graph consists of those vertices with only one neighbouring vertex). However, for a non-tree graph (graphs satisfying $|\mathcal{E}| \geq |\mathcal{V}|$), the remaining subgraph after removing a vertex and associated edges might still be connected. For example, Figure 2.2 is a toy network from Knight et al. (2019), to which the authors refer as 'fiveNet'. Its graph structure is not that of a tree, and its structure will result in two separate components if we remove the 1-st vertex and its associated edges, but the remaining graph will still be connected if we remove the 2-nd vertex and its associated edges.

  In such a case, if we still perform the relinking procedure introduced by Jansen et al. (2009) (which is designed for tree structures) after removing $v_{k_r}^*$, we may

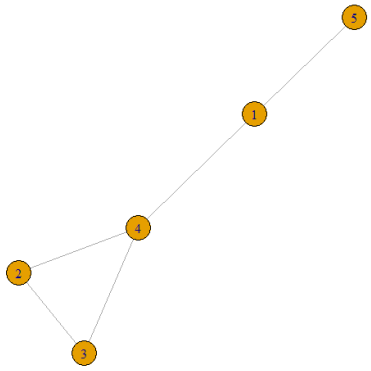

Figure 2.2: An undirected network structure 'fiveNet' with five nodes from Knight et al. (2019).

introduce artificial connections for these vertices $\{v_s^*\}_{s:v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}}$, which might have consequent stability issues since the wavelet bases will have larger overlaps (Jansen and Bultheel; 1998). Thus, a new relinkage scheme is desirable to be adapted to general graphs. Suppose we are at stage-r, then the way we design the relinkage part of the algorithm is as follows.

1. Remove $v_{k_r}^*$ and all edges $\{e_l^* \,|\, v_{k_r}^* \in e_l^*\}$.
2. Test the connectivity of the subgraph consisting of $\mathcal{N}_{k_r,r}^{\mathcal{V}^*}$ as vertices. If connected, then the relinkage will not be performed.
3. If the subgraphs are not connected, then find the minimum spanning tree and *embed* this spanning tree into the existing graph structure.

Once we complete the relinkage, the neighbourhood structure will be updated to be $\mathcal{N}_{k,r-1}^{\mathcal{V}^*}$ for all $k \in \{1, ..., m\} \backslash \{k_m, ..., k_r\}$. We denote the line graph structure at stage-r as $G_r^* = (\mathcal{V}_r^*, \mathcal{E}_r^*)$, and the one after relinkage as $G_{r-1}^* = (\mathcal{V}_{r-1}^*, \mathcal{E}_{r-1}^*)$, where $\mathcal{V}_r^* = \mathcal{V}_{r-1}^* \cup \{v_{k_r}^*\}$.

Figure 2.3 helps visualise the proposed relinkage method compared to the method from Jansen et al. (2009). We can see that our proposed relinkage method results

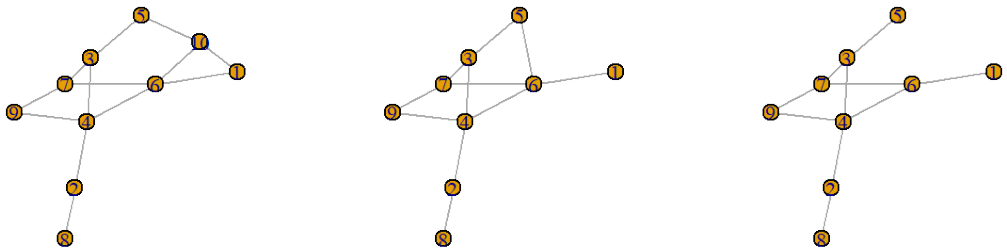

Figure 2.3: Proposed relinkage method versus the relinkage method from Jansen et al. (2009). **Left:** A toy network. **Middle:** The next-stage network after removing the 10-th node via the relinkage from Jansen et al. (2009). **Right:** The next-stage network after removing the 10-th node via the proposed relinkage.

in a sparser graph structure for the next stage and avoids adding redundant edges, such as the edge $\{v_5, v_6\}$ in the middle plot.

- **Iterate:** We iterate the split-predict-update procedures discussed above to obtain a sequence of detail coefficients $\{d^{\Gamma^*}_{k_m}, ..., d^{\Gamma^*}_{k_{\tau+1}}\}$, where τ is the stopping time, which indicates the number of line graph vertices that will not be removed. In our work, we set $\tau = 2$ as recommended in the literature, see for example, Jansen et al. (2009) and Nunes et al. (2006). For details on the stopping time problem for LOCAAT, the reader can refer to Nunes (2006) and Mahadevan (2010).

- **Inverse:** A lifting scheme such as the one discussed above is a linear transform, which guarantees a perfect reconstruction, see Sweldens (1998). Thus, the inverse transform can be done by *undoing* the lifting steps in equation (2.3.9) and (2.3.12), from stage-$(r-1)$ to stage-r, which are as follows

$$c^{\Gamma^*}_{s,r} = c^{\Gamma^*}_{s,r-1} - b^{\Gamma^*}_{s,r} d^{\Gamma^*}_{k_r}, \quad \text{for } s : v^*_s \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r} \tag{2.3.14}$$

$$c^{\Gamma^*}_{k_r,r} = d^{\Gamma^*}_{k_r} + \sum_{s:v^*_s \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}} a^{\Gamma^*}_{s,r} c^{\Gamma^*}_{s,r}. \tag{2.3.15}$$

From a computation viewpoint, a *lifting coefficient array* has to be stored after every stage to allow the inverse transform. Consider first the inverse transform for a tree structure as in Jansen et al. (2009). At stage-$(r-1)$, we start with G^*_{r-1} and disconnect the set of edges $\{e^*_k = \{v^*_i, v^*_j\} \,|\, e^*_k \in \mathcal{E}^*_{r-1} \text{ and } v^*_i, v^*_j \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}\}$. Then add the vertex $v^*_{k_r}$ and connect it to all vertices in $\mathcal{N}^{\mathcal{V}^*}_{k_r,r}$, thus, obtaining G^*_r. For non-tree cases, it is possible that we do not have to disconnect some of the edges in the graph G^*_{r-1} as discussed in relinkage. Therefore, we have to preserve more information than the lifting array in Jansen et al. (2009). We first propose the lifting array as

$$k_r \quad |\mathcal{N}^{\mathcal{V}^*}_{k_r,r}| \quad S^*_r \quad \underline{a}^{\Gamma^*}_r \quad \underline{b}^{\Gamma^*}_r,$$

where S^*_r is the set consists of all s such that $v^*_s \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}$, and $\underline{a}^*_r$; $\underline{b}^*_r$ are sequences of predict and update coefficients. In addition, we construct the following further list at every stage. For stage-r, the component of the 'extra' list consists of all pairs (s, s'), where $s \neq s'$ and $s, s' \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}$, such that $\{v^*_s, v^*_{s'}\} \in \mathcal{E}^*_r$. Thus, for the inverse transform, from stage-$(r-1)$ to stage-r, the edges in this list will not be disconnected.

2.3.3 LG-LOCAAT Properties

In this section, we give some aspects of the theory behind the LG-LOCAAT transform. We mainly focus on sparsity, stability conditions, and the scale interpretation, which will all contribute to the computational performance of our approach. Some results are the generalised version of those obtained by Jansen et al. (2009).

Wavelet Coefficient Magnitude

For multiresolution analysis, the term 'sparsity' usually indicates that after a wavelet transform, the detail coefficients form a sparse sequence. This indicates that the energy of the observations is concentrated in a relative small amount of coefficients, which allows us to represent the function by a small set of the coefficients and wavelet functions.

Recall that we are interested in a function g^{Γ^*} defined on the metrized graph Γ^*, where the data locations $\{\mathbf{p}^*_{v^*_k}\}_{k=1}^m$ are associated with new vertices $v^*_k \in \mathcal{V}^*$, along with the $\text{dist}_{\mathcal{V}^*}$ (generated by $\text{dist}_{\text{path}}$) give us a metric space. Then a Euclidean analogue of a Lipschitz continuity on a graph can be generalised as follows.

Definition 2.3.1. A function g^{Γ^*} defined on Γ^* has a point $\mathbf{p}^*_L$ of Lipschitz continuity if there exists an interval $\mathbf{I}^* \subset \Gamma^*$, such that for all $\mathbf{p}^* \in \mathbf{I}^*$,

$$\left|g^{\Gamma^*}(\mathbf{p}^*_L) - g^{\Gamma^*}(\mathbf{p}^*)\right| \leq C\,\text{dist}(\mathbf{p}^*_L, \mathbf{p}^*),$$

where $0 < C < \infty$ is a constant.

This definition is motivated by the spatial Lipschitz continuous function used in Jansen et al. (2009) and the Hölder class for tree graphs defined by Gavish et al. (2010). Based on this definition, we have the following proposition.

Proposition 2.3.2. For a stage-r prediction step, if for all $v^*_k \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r} \cup \{v^*_{k_r}\}$, we have $c^{\Gamma^*}_{k,r} = g^{\Gamma^*}(\mathbf{p}^*_{v^*_k})$, and g^{Γ^*} is Lipschitz continuous with an interval that contains all these metrized points associated with $\mathcal{N}^{\mathcal{V}^*}_{k_r,r}$, then the detail coefficient obtained by equation (2.3.9) at stage-r satisfies that

$$\left|d^{\Gamma^*}_{k_r}\right| \leq C\,\frac{\sum_{s:v^*_s \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}} \text{dist}(v^*_{k_r}, v^*_s)}{|\mathcal{N}^{\mathcal{V}^*}_{k_r,r}|}. \tag{2.3.16}$$

For the proof, the reader can refer to Appendix C.2. Nonetheless, acquiring a precise bound for all detail coefficients is not a straightforward task for iterative methods. Let us consider the following situation, suppose at stage-r, the function underpinning the scaling coefficients $(c^{\Gamma^*}_{k,r})$ is Lipschitz continuous, which indicates that

$$\left|c^{\Gamma^*}_{k,r} - c^{\Gamma^*}_{s,r}\right| \leq C\,\text{dist}(v^*_{k_r}, v^*_s),$$

Now let us suppose that $s : v^*_s \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}$, and $k : v^*_k \notin \mathcal{N}^{\mathcal{V}^*}_{k_r,r}$, then from stage-$r$ to stage-$(r-1)$, we have that

$$\begin{aligned} c^{\Gamma^*}_{k,r-1} &= c^{\Gamma^*}_{k,r}, \\ c^{\Gamma^*}_{s,r-1} &= c^{\Gamma^*}_{s,r} + b^{\Gamma^*}_{s,r} d^{\Gamma^*}_{k_r}. \end{aligned}$$

We further assume that the detail coefficient satisfies the Proposition 2.3.2, then the bound for the absolute difference between $c^{\Gamma^*}_{k,r-1}$ and $c^{\Gamma^*}_{s,r-1}$ becomes

$$\begin{aligned}\left|c^{\Gamma^*}_{k,r-1} - c^{\Gamma^*}_{s,r-1}\right| &= \left|\left(c^{\Gamma^*}_{k,r} - c^{\Gamma^*}_{s,r}\right) - b^{\Gamma^*}_{s,r} d^{\Gamma^*}_{k_r}\right| \\ &\leq \left|c^{\Gamma^*}_{k,r} - c^{\Gamma^*}_{s,r}\right| + b^{\Gamma^*}_{s,r}\left|d^{\Gamma^*}_{k_r}\right| \\ &\leq C\,\mathrm{dist}(v^*_k, v^*_s) + b^{\Gamma^*}_{s,r} C\,\frac{\sum_{t:v^*_t \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}} \mathrm{dist}(v^*_{k_r}, v^*_t)}{|\mathcal{N}^{\mathcal{V}^*}_{k_r,r}|}.\end{aligned}$$

Hence, there is no guarantee that the function underpinning $c^{\Gamma*}_{k,r-1}$ is still Lipschitz continuous with the same constant C. This indicates that if $v^*_{k_r}$ and $v^*_{k_{r-1}}$ are 'close' to each other for some stage-r, there will be an unwanted effect on the compression ability.

Stability

The wavelet functions generated by the lifting scheme (including LOCAAT-based approaches) are no longer orthogonal, thus, the algorithm stability may become an issue. One way to guarantee the stability of the transform is to ensure that both dual and primal wavelet functions form Riesz bases, such that

$$L\,\|g^{\Gamma^*}\|^2_{L_2} \leq \sum_{k\in\mathcal{D}_r} |\langle g^{\Gamma^*}, \psi^{\Gamma^*}_k\rangle|^2 \leq U\,\|g^{\Gamma^*}\|^2_{L_2}, \tag{2.3.17}$$

$$\tilde{L}\,\|g^{\Gamma^*}\|^2_{L_2} \leq \sum_{k\in\mathcal{D}_r} |\langle g^{\Gamma^*}, \tilde{\psi}^{\Gamma^*}_k\rangle|^2 \leq \tilde{U}\,\|g^{\Gamma^*}\|^2_{L_2}, \tag{2.3.18}$$

where $\mathcal{D}_r = \{k_m, ..., k_{r+1}\}$. If we can find the upper bounds, then the lower bound can be obtained automatically by the duality, such that $L = \tilde{U}^{-1}$ and $\tilde{L} = U^{-1}$ holds, see Cohen et al. (1993) and Daubechies (1992). Nonetheless, as pointed out by Jansen et al. (2009), it is challenging to verify whether a set of bases satisfies the Riesz condition on a global scale, especially in irregular scenarios. Simoens and Vandewalle (2003) and Jansen and Oonincx (2005) presented a necessary but not sufficient condition for the Riesz condition is that each one-step transform and its inverse are uniformly bounded. This can be articulated as the one-level lifting operator and its inverse being bounded, see Simoens and Vandewalle (2003). For the LOCAAT-based algorithm, it means that the quantities resulting from the predict (equation (2.3.9)), the update (equation 2.3.12),

undo update (equation (2.3.14)), and undo predict (equation (2.3.15)) should be bounded in norm. For the forward prediction, given that $\sum_{k:v_k^*\in\mathcal{N}_{k_r,r}^{\mathcal{V}^*}} a_{s,r}^{\Gamma^*} = 1$ and $a_{s,r}^{\Gamma^*} \geq 0$, we have that

$$\begin{aligned} |d_{k_r}^{\Gamma^*}|^2 &= |c_{k_r,r}^{\Gamma^*} - \sum_{s:v_s^*\in\mathcal{N}_{k_r,r}^{\mathcal{V}^*}} a_{s,r}^{\Gamma^*} c_{s,r}^{\Gamma^*}|^2 \\ &\leq (1 + \sum_{s:v_s^*\in\mathcal{N}_{k_r,r}^{\mathcal{V}^*}} |a_{s,r}^{\Gamma^*}|^2) \sum_{k:v_k^*\in\mathcal{N}_{k_r,r}^{\mathcal{V}^*}\cup\{k_r\}} |c_{k,r}^{\Gamma^*}|^2, \qquad (2.3.19) \\ &\leq 2 \sum_{k:v_k^*\in\mathcal{N}_{k_r,r}^{\mathcal{V}^*}\cup\{k_r\}} |c_{k,r}^{\Gamma^*}|^2. \end{aligned}$$

by the Cauchy-Schwarz inequality. Since each update coefficient satisfies that $0 < b_{s,r}^{\Gamma^*} \leq \frac{1}{2}$, for all $v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}$, see Jansen et al. (2009), this guarantees that $c_{s,r-1}^{\Gamma^*}$ is bounded after the update (equation (2.3.12)) for all $v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}$. For the one-level undo lifting, the boundness can be immediately obtained by the duality, see Jansen et al. (2009). From the equation (2.3.19), we can see that the upper bound is given by the value $(1 + \sum_{s:v_s^*\in\mathcal{N}_{k_r,r}^{\mathcal{V}^*}} |a_{s,r}^{\Gamma^*}|^2)$. Hence, if we lift a new vertex which only has one neighbouring new vertex, the bound tends to be the maximum (which is exactly 2). This matches with the practical sensitive points phenomena observed by Jansen and Bultheel (1998) and Mahadevan (2010), where the term 'sensitive points' indicates that such boundary points contain high energy and have a significant influence on recovering the signal. On the other hand, if a new vertex with a large size of neighbourhood has been lifted, and the prediction weights are almost evenly distributed (e.g. moving average), then the upper bound tends to be small.

Scale Interpretation

Having the notion of scale is essential for any wavelet-based method. The scale is a measurement related to the level of detail, also known as the resolution, and has a strong connection with the Fourier-based frequency notion, see Nason (2008) or Vidakovic (2009). Recall that in the classical wavelet methods, the scales can be generated by the dilation relation. Within the framework based on LOCAAT, the concept of scale

is not directly discernible due to the absence of a dilation relation. As in the first generation wavelets literature, the scale can be viewed as a parameter of a power function, which signifies the reduction of the detail coefficients, see Daubechies (1992). Inspired by this, the quantities in equation (2.3.16) can help in determining the 'scale' notion. Notice that its expression is very similar to the 'sum of distances' and the 'average distance integral' as we discussed in Section 2.3. Thus, for computational convenience, we simply define the scale of the detail coefficient $d_{k_r}^{\Gamma^*}$ obtained at stage-r as

$$\text{scale}_{k_r}^{\Gamma^*} = I_{k_r,r}^{\Gamma^*} \tag{2.3.20}$$

The integral satisfies that $I_{k_{r-1},r-1}^{\Gamma^*} \geq I_{k_r,r}^{\Gamma_*}$, which implies the correspondence between removal order and scale. Note that when using the integral values as a sequence of ones, the potential power of this integral choice is that for the initial recursions of lifting, the next stage-$(r-1)$ removal choice $v_{k_{r-1}}^*$ cannot be a neighbouring vertex of $v_{k_r}^*$. For example, once we remove $v_{k_m}^*$, after the integral update, the integral values of its neighbourhood will exceed the value one, which guarantees that another vertex (not in the neighbourhood) will be picked for removal, hence the space is explored more efficiently.

2.3.4 Original Domain Transformation

Through the iterations of the LG-LOCAAT steps discussed above, after stage-r, the expansion of the function approximation can be written as (Jansen et al.; 2009)

$$g^{\Gamma^*}(\mathbf{p}^*) = \sum_{s\in\mathcal{S}_{r-1}} c_{s,r-1}^{\Gamma^*}\varphi_{s,r-1}^{\Gamma^*}(\mathbf{p}^*) + \sum_{l\in\mathcal{D}_{r-1}} d_l^{\Gamma^*}\psi_l^{\Gamma^*}(\mathbf{p}^*), \tag{2.3.21}$$

where $\mathcal{D}_{r-1} = \{k_m, ..., k_r\}$ are the wavelet coefficient indices, and $\mathcal{S}_{r-1} = \{1, ..., m\}\backslash\mathcal{D}_{r-1}$ are the scaling indices. Then functions $\{\psi_l^{\Gamma^*}\}_{l\in\mathcal{D}_{r-1}}$ are the wavelets functions and are obtained recursively as detailed in Section 1.5. Recall that the LG-LOCAAT is designed as a transform in the *line graph space* (G^*, or the metrized version Γ^*) of the original graph. However, we are primarily interested in the function approximation in the original graph space. In particular, understanding the topology of the set of scaling and wavelet

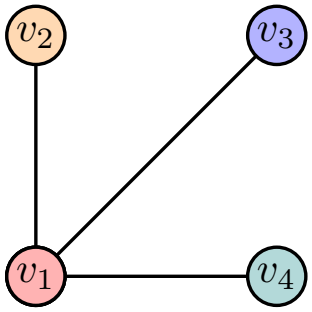

Figure 2.4: An example of a claw graph.

functions is of interest. We can conceptualise our line graph transform as being associated to a map $\mathbf{LG}$, such that $\mathbf{LG} : G \longrightarrow G^*$.

A question of interest is whether an inverse transform $\mathbf{LG}^{-1}$ exists, at any stage-r. However, *the line graph transform is not always invertible*. This is since while *any graph has its unique corresponding line graph, but not all graphs are line graphs*, see Bondy and Murty (2008). As an example, let us consider the case where we lifted a new vertex $v^*_{k_r}$ at stage-r, and its neighbourhood is denoted as $\mathcal{N}^{\mathcal{V}^*}_{k_r,r}$. Therefore, we have the subgraph $G^{*\text{supp}}_r = (\mathcal{V}^{*\text{supp}}_r, \mathcal{E}^{*\text{supp}}_r)$, where $\mathcal{V}^{*\text{supp}}_r = \{v^*_{k_r}\} \cup \mathcal{N}^{\mathcal{V}^*}_{k_r,r}$ and $\mathcal{E}^{*\text{supp}}_r = \{\{v^*_{k_r}, v^*_s\}\}_{s:\, v^*_s \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}}$. Recall that the prediction step at stage-r is performed on this subgraph $G^{*\text{supp}}_r$, hence the analytic form of the wavelet function $\varphi^{\Gamma^*}_{k_r}$ is defined on the topology of $G^{*\text{supp}}_r$. Now if we want to obtain the associated analytical form defined on the original graph G for $\varphi^{\Gamma^*}_{k_r}$, we have to find the inverse of $G^{*\text{supp}}_r$. Let us consider a special case, suppose there are three component in the neighbourhood $\mathcal{N}^{\mathcal{V}^*}_{k_r,r}$. This graph topology is called 'claw graph' in graph theory literature, see Bondy and Murty (2008). Figure 2.4 gives a visualisation of a claw graph. Unfortunately, claw graphs are not line graphs, see Bondy and Murty (2008) and Chudnovsky and Seymour (2005), which indicates that $G^{*\text{supp}}_r$ has no interpretation in the original graph domain if it is a claw graph. Moreover, any star graph (see Section 1.4.1) with more than four vertices is not a line graph, see Bondy and Murty (2008). Hence, at any stage-r, if there are more than three vertices used for prediction, then the associated topology $G^{*\text{supp}}_r$ will have no interpretation. Moreover, the stage-$(r-1)$ graph after relinkage may also not be a line graph. To check whether the resulting graph at stage-$(r-1)$ is a line graph, one could use the algorithm introduced by

Roussopoulos (1973), which has the computational cost $\mathcal{O}(\max\{n^*_{r-1}, m^*_{r-1}\})$, where n^*_{r-1} and m^*_{r-1} indicate the number of vertices and edges of the relinked graph at stage-$(r-1)$, respectively. However, this algorithm is beyond the scope of this thesis, and we consider it as a direction for future research. Nonetheless, we find it worth mentioning that if the original graph is very dense, for example, if G is close to a complete graph, such that m approaches to $n(n-1)/2$, then the computational cost will increase significantly. For the purpose of this work, recall the equivalence between the information in the original edge domain and the line graph vertex domain, also represented in equation (2.3.4) as $g^{\mathcal{E}}(e) \equiv g^{\mathcal{V}^*}(v^*)$, which we will assume to hold at each stage.

2.4 Simulation Testbed

In this section, we present a comprehensive simulation study in order to investigate the behaviour of our LG-LOCAAT algorithm. The simulation consists of three parts: stability (assessed via the condition number), compression ability (assessed via the sparsity plot), and denoising performance (assessed via the average mean squared error). Before moving onto the simulation setting, let us first describe the sampling for our test functions and toy models.

2.4.1 Test Functions

In our work, the set of functions from Jansen et al. (2009) will be used in our simulation study, which are two-dimensional analogues of the test functions from Donoho and Johnstone (1994), along with the two-dimensional *maartenfunc* (`mfc`) introduced by Jansen et al. (2009). 'Blocks' is a function with several blocks of different function values, but within each block, the value is the same. 'Doppler' is a coordinate-based sine function, which can be considered as a smooth function without discontinuities. 'Bumps' is a function that has several spikes as discontinuities. 'Heavisine' is a smooth function but with high variation. '`mfc`' is a function that has a line segment as a set of discontinuities, which divides the domain into two parts, and each of them is a smooth

function. We also test the performance of our method on the g_1 function introduced in Mahadevan (2010), which has a similar form to `mfc`, but with some more 'obvious' jump discontinuities. Figure 2.5 gives the visualisation of these functions on a $[0,1] \times [0,1]$ square. The formulae can be found in Appendix B, see also Mahadevan (2010).

2.4.1.1 Sampling Network Structure

Since the test functions are defined over the square $[0,1] \times [0,1]$ in Euclidean space, we sample this space by means of a network structure. We first sample n points $\{(x_i, y_i)\}_{i=1}^n$, where $x_i, y_i \sim \text{Unif}(0,1)$. These points are fixed as the graph vertices $\{v_i = (x_i, y_i)\}_{i=1}^n$ of the network G. Then the graph edges are obtained by the minimum spanning tree, which gives us a set of m ($m = n - 1$) connections between vertices. We let these be the set of edges for the graph G, and the length of each edge is simply given by the Euclidean distance of the two vertices associated with this edge.

2.4.1.2 Embedding the Function Values

Within our simulation study, we will determine the function values at the set of edges using two methods. The first method is to simply select the value that corresponds to the coordinate at the midpoint of each edge. Let us assume we have a function g^{test} from the previously mentioned collection of test functions. Consequently, for $e_k = \{v_i, v_j\}$, where $v_i = (x_i, y_i)$ and $v_j = (x_j, y_j)$, the corresponding 'true' observation value will be

$$g_k^{\mathcal{E}} = g_k^{\mathcal{V}^*} := g^{\text{test}}\left(\frac{x_i + x_j}{2}, \frac{y_i + y_j}{2}\right). \tag{2.4.1}$$

Nonetheless, this function's definition still hinges on the pointwise perspective. We also explore an alternative method for designating the function values, which incorporates the geometric details of the edges. Inspired by the cell average function from Donoho (1997), we suggest a different approach for adjusting the function values. We refer to these functions as 'edge averaging', bearing a resemblance to the cell averaging but defined on a set of line segments rather than a set of squares. For $e_k = \{v_i, v_j\}$, the

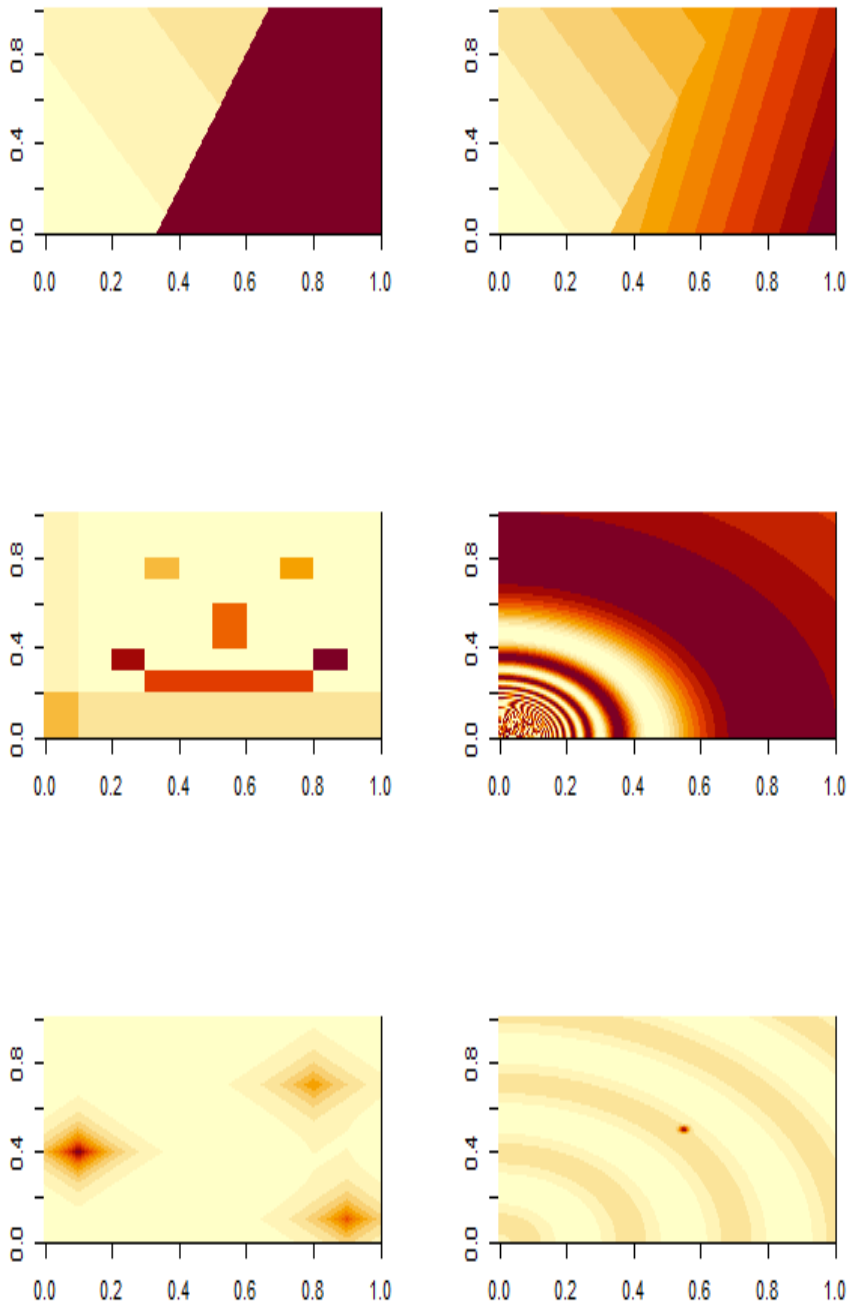

Figure 2.5: Heat maps for the test functions used in simulation. From left to right on *top row*: g_1, maartenfunc; *middle row*: Blocks, Doppler; *bottom row*: Bumps, Heavisine.

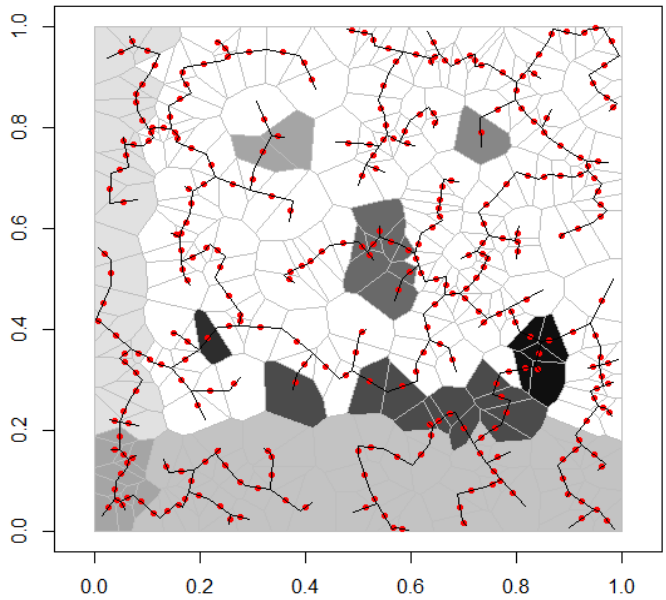

Figure 2.6: Heat map for a set of function values defined on Voronoi polygons of the vertex set of the line graph. The red points are the middle points of network edges (hence, the vertices of the line graph), and each polygon represents the function value of the corresponding new vertex (original edge). The test function here is the edge averaging Blocks function, the values are obtained by pointwise functions. The Voronoi polygons are generated by the new vertices (red points).

function value will be

$$g_k^{\mathcal{E}} = g_k^{\mathcal{V}^*} = \left(\sum_{h=0}^{N-1} g^{\text{test}} \left(x_i + \frac{h}{N-1}(x_j - x_i), y_i + \frac{h}{N-1}(y_j - y_i) \right) \right) \Big/ N, \qquad (2.4.2)$$

where g^{test} is one of the two-dimensional test functions. Here we simply let $N = 100$. Note that this edge averaging function depends on the geometric location of the edges. Figure 2.4.1 and Figure 2.4.2 show the visualisations of the 'pointwise' Blocks function and 'edge averaging' Blocks function, respectively. Note that compared with the pointwise function, the edge averaging is smoother when the edges cross blocks with different values.

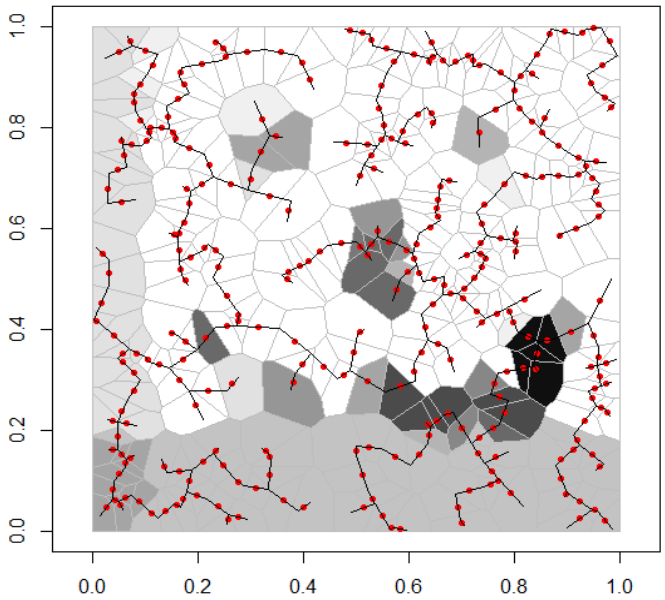

Figure 2.7: Heat map for a set of function values defined on Voronoi polygons of the vertex set of the line graph. The red points are the middle points of network edges (hence, the vertices of the line graph), and each polygon represents the function value of the corresponding new vertex (original edge). The test function here is the Blocks function, the values are obtained by edge averaging functions. The Voronoi polygons are generated by the new vertices (red points).

2.5 Simulation Results

In this section, we will provide numerical results for our simulation study. Various proposed methods based on different combinations of measurements, such as initial scaling function integral and prediction weights, will be presented. Acronyms will be used to identify the approaches we explore, and their descriptions can be found in Table 2.1.

2.5.1 Stability

The *condition number* will be used to represent the stability of the transform. Before discussing the condition number, we first have to construct the matrix associated to the transform. In the remainder of this section, we drop the superscripts and subscripts of

Acronym	Proposed LG-LOCAAT variant
LG-Sid-c	S: sum of distances as integral (equation (2.3.6)); id: inverse distance prediction (equation (2.3.10)); c: coordinate information available.
LG-Aid-c	A: average distance as integral (equation (2.3.7)); id: inverse distance prediction (equation (2.3.10)); c: coordinate information available.
LG-Did-c	D: a sequence of ones as integrals (equation (2.3.8)); id: inverse distance prediction (equation (2.3.10)); c: coordinate information available.
LG-Snw-c	S: sum of distances as integral (equation (2.3.6)); nw: moving average prediction (equation (2.3.11)); c: coordinate information available.
LG-Anw-c	A: average distance as integral (equation (2.3.7)); nw: moving average prediction (equation (2.3.11)); c: coordinate information available.
LG-Dnw-c	D: a sequence of ones as integrals (equation (2.3.8)); nw: moving average prediction (equation (2.3.11)); c: coordinate information available.
LG-Sid-p	S: sum of distances as integral (equation (2.3.6)); id: inverse distance prediction (equation (2.3.10)); p: path length available.
LG-Aid-p	A: average distance as integral (equation (2.3.7)); id: inverse distance prediction (equation (2.3.10)); p: path length available.
LG-Did-p	D: a sequence of ones as integrals(equation (2.3.8)); id: inverse distance prediction (equation (2.3.10)); p: path length available.
LG-Snw-p	S: sum of distances as integral (equation (2.3.6)); nw: moving average prediction (equation (2.3.11)); p: path length available.
LG-Anw-p	A: average distance as integral (equation (2.3.7)); nw: moving average prediction (equation (2.3.11)); p: path length available.
LG-Dnw-p	D: a sequence of ones as integrals (equation (2.3.8)); nw: moving average prediction (equation (2.3.11)); p: path length available.

Table 2.1: Acronyms and algorithm descriptions for different parameter choices of LG-LOCAAT.

the function values, coefficients, and wavelet/scaling functions, since the construction of the lifting matrix is general for any scheme. Computationally, the lifting scheme can be represented as a matrix multiplication, $\underline{d} = \tilde{R}\underline{g}$, where $\underline{g}$ is the observation sequence, $\tilde{R}$ denotes the forward matrix generated by lifting scheme, and $\underline{d}$ is the detail coefficient sequence.

Let vector $\underline{\tilde{\psi}}_{k_r}$ be the filter associated with the k_r-th dual wavelet function $\tilde{\psi}_{k_r}$, then the forward matrix can be represented as

$$\tilde{R} = \begin{bmatrix} \underline{\tilde{\psi}}_1^T \\ \vdots \\ \underline{\tilde{\psi}}_m^T \end{bmatrix}. \tag{2.5.1}$$

Then we denote the k_r-th row of the matrix $\tilde{R}$ as $\text{row}_{k_r}(\tilde{R}) = \underline{\tilde{\psi}}_{k_r}$, can be considered as the vector form of the k_r-th dual wavelet, $\tilde{\psi}_{k_r}$. The inverse matrix R, is obtained by solving the equation $R\tilde{R} = \tilde{R}R = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix of the same dimension as R and $\tilde{R}$. Notice that for any $k, k' \in \{1, ..., m\}$, there is

$$\langle \text{row}_k(\tilde{R}), \text{col}_{k'}(R) \rangle = \delta_{kk'}.$$

Thus, the k_r-th column of the matrix R, can be considered as the vector form of the k_r-th primal wavelet, ψ_{k_r}. The inverse matrix can be represented as

$$R = \left[\underline{\psi}_1, \cdots, \underline{\psi}_m \right]. \tag{2.5.2}$$

Now, if we let the vector $\boldsymbol{u}_s = (\underbrace{0, ..., 0}_{s-1}, 1, \underbrace{0, ..., 0}_{m-s})^T$ as the detail coefficient sequence, which indicates that $d_s = 1$ and $d_{s'} = 0$ for all $s' \neq s$, and perform inverse transform for this vector $\boldsymbol{u}_s$, this will yield the vector form for the s-th wavelet, $\underline{\psi}_s$. Then we will obtain the inverse matrix R by performing the procedure above for $s \in \{1, ..., m\}$. The (forward) lifting matrix can be calculated by taking the inverse of R.

The condition number of the lifting matrix is then measured by

$$\kappa(\tilde{R}) = \kappa(R) = \|\tilde{R}\|_2 \|R\|_2,$$

where $\|\cdot\|_2$ is the L_2-norm of the matrix. This norm is sometimes been called the spectral norm of the matrix, and can be found by the *singular value decomposition* (SVD) of the matrix (Trefethen and Bau III; 1997). For example, for our $m \times m$ matrix $\tilde{R}$, we can obtain m ordered singular values through SVD. Since the forward matrix of our LG-LOCAAT is invertible, then these singular values are all positive values. We further denote by $\{\rho_i\}_{i\in\{1,\dots m\}}$ the set of singular values ordered according to their magnitude, which means that ρ_1 is the maximum singular value and ρ_m is the minimum one. Then we have $\|\tilde{R}\|_2 = \rho_1$. As the matrix $\tilde{R}$ is non-singular, we have $\|R\|_2 = \|(\tilde{R})^{-1}\|_2 = 1/\rho_m$. Thus,

$$\kappa(\tilde{R}) = \frac{\rho_1}{\rho_m}.$$

Notice that the condition number satisfies $\kappa(\tilde{R}) \geq 1$. A transform is more stable than another one if its condition number is closer to one, see Higham (2002). The condition number and stabilised transform will be discussed in more detail in the later chapters.

Condition Number	Max	75%	Median	25%	Min
LG-Sid-c	14.5314	12.9183	12.4962	11.9702	11.1918
LG-Aid-c	15.0632	13.2504	12.5252	11.9598	11.1996
LG-Did-c	13.9051	12.5774	11.5010	11.0700	10.6420
LG-Snw-c	13.5717	12.3798	11.7343	11.3743	10.8643
LG-Anw-c	12.7559	11.4412	11.0405	10.5475	10.0281
LG-Dnw-c	**12.1607**	**11.0283**	**10.5684**	**10.2285**	**9.9500**

Table 2.2: Condition number for LG-LOCAAT with coordinate information.

Condition Number	Max	75%	Median	25%	Min
LG-Sid-p	13.4308	12.3485	11.7877	11.3759	10.7340
LG-Aid-p	12.7611	11.4225	10.9914	10.5397	10.0863
LG-Did-p	12.6918	11.6651	10.7789	10.3077	10.0251
LG-Snw-p	13.2506	12.0824	11.5843	11.2258	10.6871
LG-Anw-p	12.5608	11.3372	10.7768	10.3567	10.0530
LG-Dnw-p	**12.2235**	**11.0813**	**10.5528**	**10.1628**	**9.9535**

Table 2.3: Condition number for LG-LOCAAT using the path length.

Tables 2.2 and 2.3 give the quantiles of the condition numbers of the associated transform. We can see that overall, performing the scheme with a sequence of ones as

the integral will result in a lower condition number than using sum of distances or average distance, moving average prediction provides a more stabilised transform than inverse distance prediction. Reassuringly, it does not make a significant difference if the LG-LOCAAT is based on the path length. The scheme based only on the path length gives a comparably stabilised transform. These results are in agreement with our intuition set out in the discussion in Section 2.3.3.

2.5.2 Sparsity

A desirable property of the wavelet transform is that a function can be transformed into a sparse set of detail coefficients if this function belongs to a certain class of functions, see Meyer (1992). Thus, the energy of the signal is concentrated in a small set of detail coefficients after the transform, which means that the function can be well-approximated by only a few detail coefficients.

The tool we use to assess the performance of compression is the sparsity plot, and its construction proceeds as follows. First, we perform the LG-LOCAAT transform for the true values $\{g_k^{\mathcal{V}^*}\}_{k=1}^m$. This will yield a corresponding vector which contains two scaling coefficients and $(m-2)$ detail coefficients. We begin with the case with just two scaling coefficients and assume all the detail coefficients are zero. Then to perform the inverse transform for this vector, we obtain an estimator $\hat{g}^{\mathcal{V}^*}(t)$ for the true function $g^{\mathcal{V}^*}$. Here $(t-1)$ gives the number of detail coefficients used in reconstruction, such that $\hat{g}^{\mathcal{V}^*}(1)$ is the reconstruction with only two scaling coefficients, while $\hat{g}^{\mathcal{V}^*}(2)$ means the reconstruction with two scaling coefficients and one detail coefficient with the largest absolute value, and so on and so forth. For any step, we will introduce one remaining detail coefficient with the largest absolute value into the vector for reconstruction. Then following the process above, we can obtain a different estimation in each step until all detail coefficients have been used. For the sparsity plot, the value on the y-axis is the integrated squared error (ISE), defined as follows:

$$\mathrm{ISE}(t) = P^{-1} \sum_{p=1}^{P} \sum_{k=1}^{m} \left(\hat{g}_k^{\mathcal{V}^{*(p)}}(t) - g_k^{\mathcal{V}^{*(p)}} \right)^2,$$

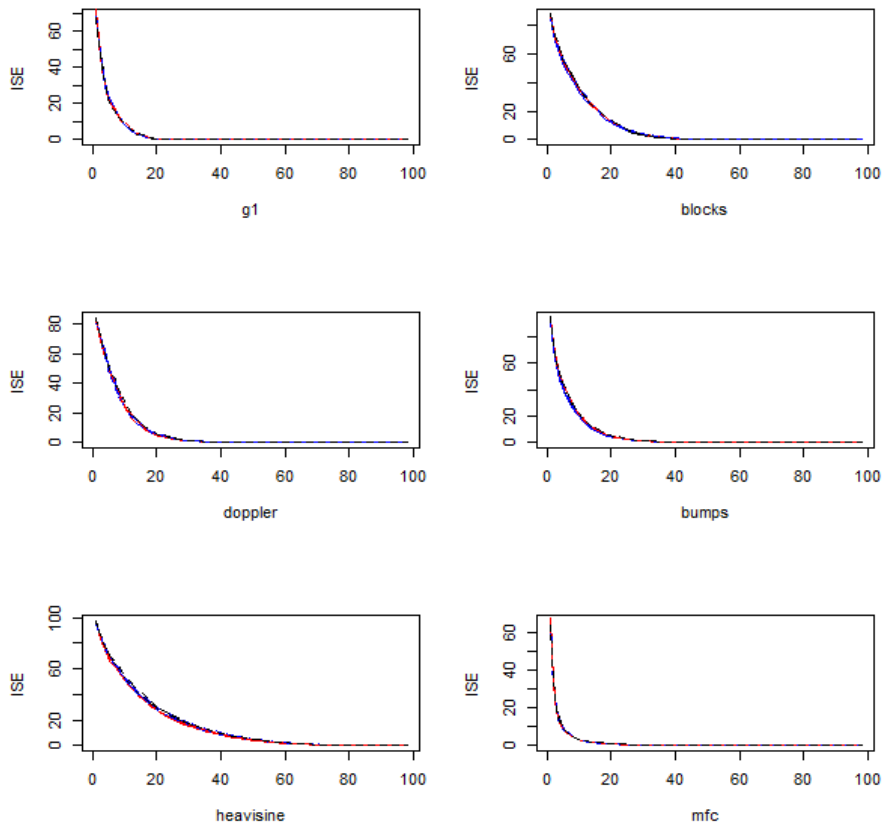

Figure 2.8: Sparsity plots for the test functions used in simulation by the equation (2.4.1). The scheme is based on coordinate information. From left to right on *top row*: g_1, Blocks; *middle row*: Doppler, Bumps; *bottom row*: Heavisine, maartenfunc. **Black line**: LG-Sid-c; **red line**: LG-Aid-c; **blue line**: LG-Did-c; **dashed black line**: LG-Snw-c; **dashed red line**: LG-Anw-c; **dashed blue line**: LG-Dnw-c.

where $g_k^{\mathcal{V}^{*(p)}}$ and $\hat{g}_k^{\mathcal{V}^{*(p)}}$ denote the true observation at new vertex 'v_k^*' and its reconstruction using $(t-1)$ detail coefficients for the p-th network. In total, we generate $P = 50$ different networks with $n = 100$ vertices (and $m = 99$ edges) for our study. The x-axis is the 't'-argument, as we mentioned above. If the ISE decays to zero fast (if for small t, the ISE is already close to zero), then the algorithm leads to a highly sparse result for the target function.

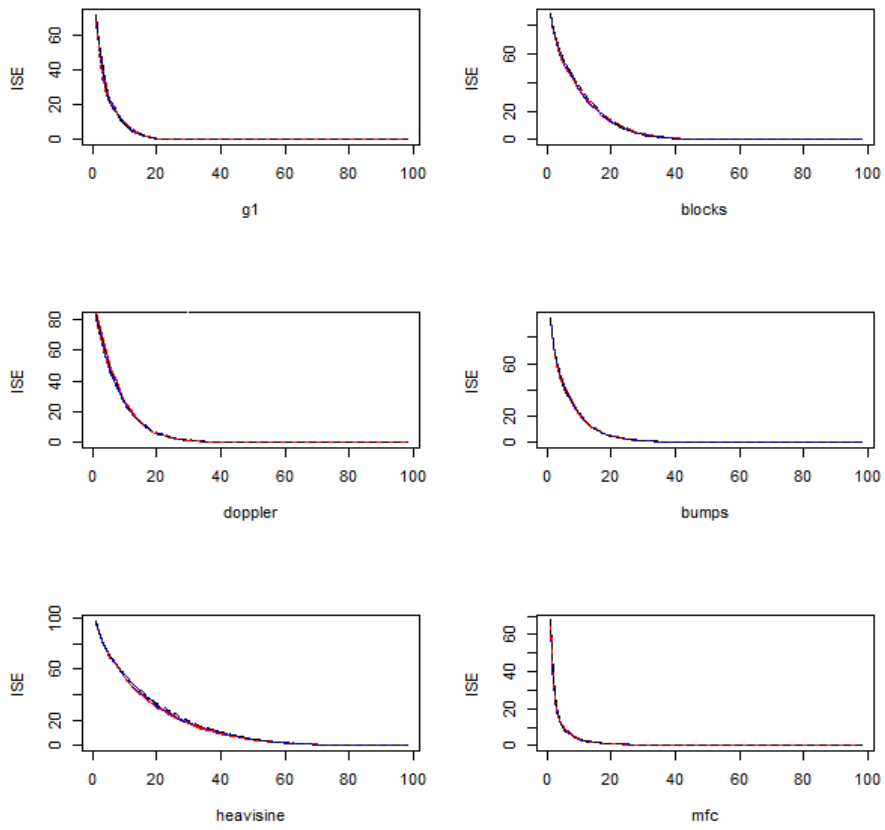

Figure 2.9: Sparsity plots for the test functions used in simulation by the equation (2.4.1). The scheme is based on path distance. From left to right on *top row*: g_1, Blocks; *middle row*: Doppler, Bumps; *bottom row*: Heavisine, maartenfunc. **Black line**: LG-Sid-p; **red line**: LG-Aid-p; **blue line**: LG-Did-p; **dashed black line**: LG-Snw-p; **dashed red line**: LG-Anw-p; **dashed blue line**: LG-Dnw-p.

2.5.2.1 Sparsity Results for Pointwise Functions

Figure 2.8 shows the sparsity results for different (pointwise) test functions, all with schemes being based on the coordinate information. We can see that the compression ability of the algorithm on g_1 and `mfc` surpass any other test function. The results for Bumps and Doppler functions display similar sparsity results, while for Blocks function the results are inferior. The compression of the Heavisine function is the weakest compared with other functions. Reassuring, there is no significant difference among different choices of integral values and prediction weights.

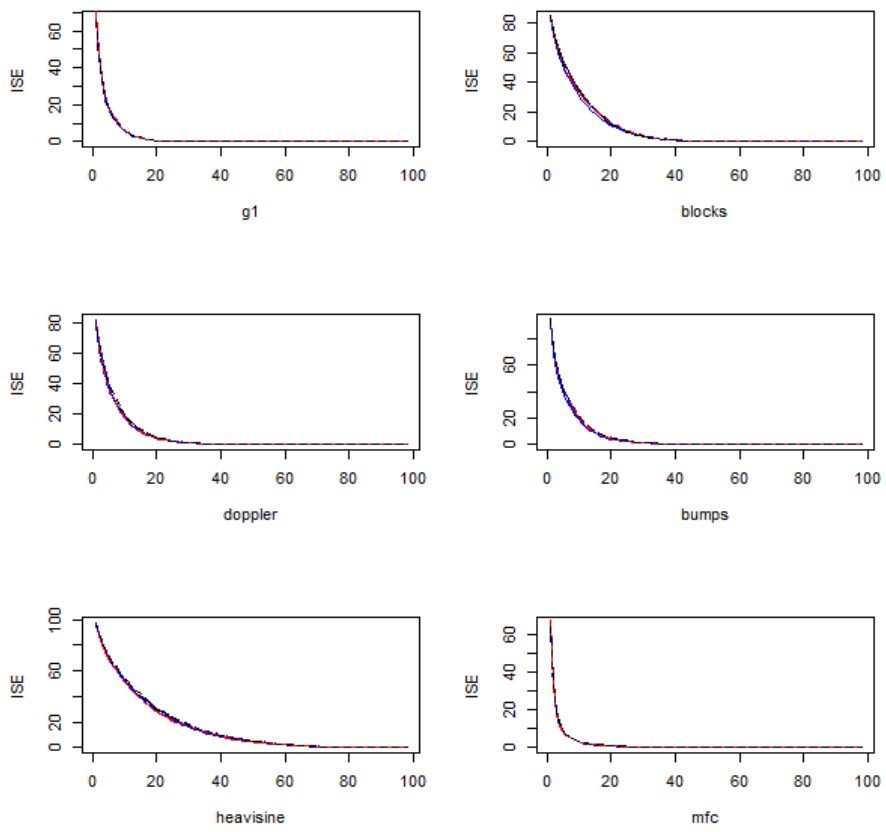

Figure 2.10: Sparsity plots for the test functions used in simulation by the equation (2.4.2). The scheme is based on coordinate information. From left to right on *top row*: g_1, Blocks; *middle row*: Doppler, Bumps; *bottom row*: Heavisine, maartenfunc. **Black line**: LG-Sid-c; **red line**: LG-Aid-c; **blue line**: LG-Did-c; **dashed black line**: LG-Snw-c; **dashed red line**: LG-Anw-c; **dashed blue line**: LG-Dnw-c.

Figure 2.9 shows the results for the same functions while using path length instead of the coordinates. For data compression, there is no significant difference between using path length and coordinate information.

2.5.2.2 Sparsity Results for Edge Averaging Functions

Figure 2.10 shows the sparsity results for different edge averaging functions. The compression ability follows the similar patterns to pointwise functions. Along with Figure 2.11, again there is no evidence of a significant difference between using path length

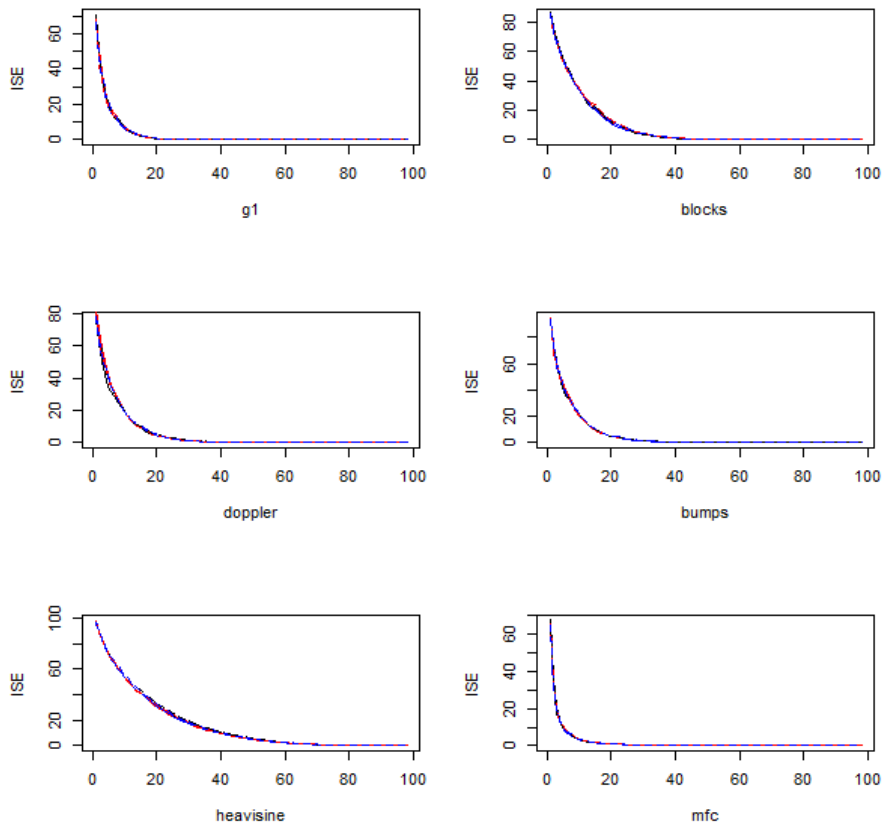

Figure 2.11: Sparsity plots for the test functions used in simulation by the equation (2.4.2). The scheme is based on path distance. From left to right on *top row*: g_1, Blocks; *middle row*: Doppler, Bumps; *bottom row*: Heavisine, maartenfunc. **Black line**: LG-Sid-p; **red line**: LG-Aid-p; **blue line**: LG-Did-p; **dashed black line**: LG-Snw-p; **dashed red line**: LG-Anw-p; **dashed blue line**: LG-Dnw-p.

and coordinate information. Both sets of plots indicate that different choices of integral values and prediction weights will not change the sparsity results significantly.

2.5.3 Denoising Performance

In this section, we investigate the behaviour of LG-LOCAAT in the context of nonparametric regression problem. Suppose we have the model,

$$\begin{aligned} f_k^{\mathcal{E}} &= g^{\mathcal{E}}(e_k) + \epsilon_k, \\ &= g_k^{\mathcal{E}} + \epsilon_k. \end{aligned} \tag{2.5.3}$$

where $g^{\mathcal{E}}$ is a true function defined on the edge set $\mathcal{E} = \{e_k\}_{k\in\{1,\dots,m\}}$, and $\{\epsilon_k\}$ are independent and identically distributed random variables (noise), assumed to follow a normal distribution $N(0, \sigma^2)$. Thus, $\underline{f}^{\mathcal{E}} = \{f_k^{\mathcal{E}}\}_{k\in\{1,\dots,m\}}$ is the set of observation values on the edges of our graph G that are corrupted by noise, and our purpose is to obtain an estimator $\hat{g}^{\mathcal{E}}$ of the true (unknown) function $g^{\mathcal{E}}$ evaluated at the observed edges. For our simulation study, the true function $g^{\mathcal{E}}$ is generated by the test functions as in Appendix B. Recall that the line graph transform allows us to represent $g^{\mathcal{E}}(e_k) = g^{\mathcal{V}^*}(v_k^*)$ (and similarly for the observations $\{f_k^{\mathcal{E}}\}_{k\in\{1,\dots,m\}}$). Thus, equation (2.5.3) can be rewritten as

$$\begin{aligned} f_k^{\mathcal{V}^*} &= g^{\mathcal{V}^*}(v_k^*) + \epsilon_k \\ &= g_k^{\mathcal{V}^*} + \epsilon_k. \end{aligned}$$

Then we can perform our proposed LG-LOCAAT algorithm for the observations $\underline{f}^{\mathcal{V}^*}$, in order to obtain a sequence of detail coefficients $\underline{d}^*$. Next, the detail coefficients will be separated into different artificial levels by taking the median and different quantiles of the size of the measurement $I^*_{k_r,r}$ associated with the scale at stage-r for all r in the algorithm, see Jansen et al. (2004, 2009) and Nunes et al. (2006). Subsequently, wavelet thresholding will be performed on these detail coefficients. Following the finding from Mahadevan (2010), which is that performing thresholding for around 80-90% detail coefficients seems an optimal choice, we perform the wavelet thresholding for all detail coefficients except for those that have been allocated at the two coarsest artificial levels. The thresholding approach used in this thesis will be empirical Bayes thresholding, unless we mention it specifically. The non-zero part of the prior density will be modelled as the 'quasi-Cauchy' distribution from Johnstone and Silverman (2004), since it has been proved to have good performance for LOCAAT-based approaches, see Nunes et al. (2006) and Jansen et al. (2009). A set of estimated coefficients, $\hat{\underline{d}}^*$, will be obtained following thresholding and we then perform the inverse transform to obtain an estimate of the true, unknown function. We denote the corresponding estimates as $\hat{g}_k^{\mathcal{V}^*}$ for all $k \in \{1, ..., m\}$. Recall from Section 2.3.4, the equivalence between the estimates $\{\hat{g}_k^{\mathcal{V}^*}\}_{k\in\{1,\dots,m\}}$ defined on the line graph G_m^*, and the original graph domain, such that we have $\hat{g}^{\mathcal{E}} = \hat{g}^{\mathcal{V}^*}$.

Our simulation will be performed on three different scales of the signal-to-noise ratio, SNR = 3, 5, and 7. This ratio is measured by SNR = $\sqrt{\text{var}(g^{\mathcal{E}})}/\sigma$, where var($g^{\mathcal{E}}$) is the variance of the simulated true function, and σ is the standard deviation of the noise. For a clear comparison, the true function will be normalised so that var($g^{\mathcal{E}}$) = 1. Thus, the noise will be generated with $\sigma = 1/3, 1/5$, and $1/7$, respectively. The simulation is designed as follows: we sample $p = 1, ..., P = 50$ different graph structures, $G^{(p)}$ (whose associated line graph will be denoted as $G^{*(p)}$). For each of them, we simulated $r = 1, ..., R = 100$ different noise sequences, of a certain choice of σ as we discussed above. Following the estimation procedure, we calculate the average mean squared error (AMSE) defined as

$$\text{AMSE} = (PRm)^{-1} \sum_{p=1}^{P} \sum_{r=1}^{R} \sum_{k=1}^{m} \left(\hat{g}^{\mathcal{E}}_{k,p,r} - g^{\mathcal{E}}_{k,p}\right)^2,$$

where $g^{\mathcal{E}}_{k,p}$ is the true edge function value corresponding to the k-th 'new' vertex on the line graph $G^{*(p)}$, while $\hat{g}^{\mathcal{E}}_{k,p,r}$ is the estimate of $g^{\mathcal{E}}_{k,p}$ when the true function is corrupted by the r-th noise sequence. Figure 2.12 shows a visualisation of one realisation of denoising for three different test functions for LG-LOCAAT with 'LG-Aid-p'.

We will also calculate the squared bias and variance, and therefore the corresponding average mean squared errors (AMSE) to check the performance of our methods. The variance is calculated as follows.

$$\text{Var} = (PRm)^{-1} \sum_{p=1}^{P} \sum_{k=1}^{m} \sum_{r=1}^{R} \left(\hat{g}^{\mathcal{E}}_{k,p,r} - \bar{g}^{\mathcal{E}}_{k,p}\right)^2,$$

where $\bar{g}_{k,p} = \frac{1}{R} \sum_{r=1}^{R} \hat{g}^{\mathcal{E}}_{k,p,r}$ and the squared bias is calculated as

$$\text{bias}^2 = (Pm)^{-1} \sum_{p=1}^{P} \sum_{k=1}^{m} \left(\bar{g}^{\mathcal{E}}_{k,p} - g^{\mathcal{E}}_{k,p}\right)^2.$$

2.5.3.1 Denoising Pointwise Functions

In this section, we provide the AMSE results for the pointwise functions generated as described in equation (2.4.1) using $R = 100$ noise sequences over $P = 50$ graph structures each with $m = 99$ edges.

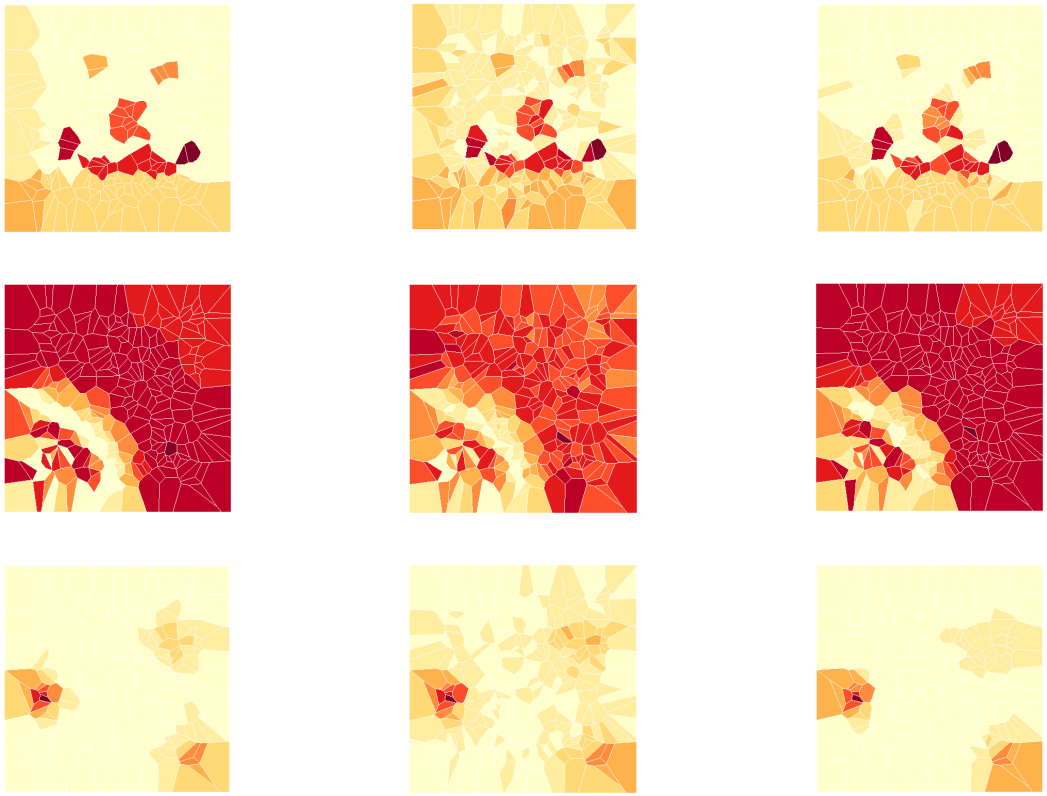

Figure 2.12: Visualisation for the LG-LOCAAT estimation of three test functions. The denoising is done by 'LG-Aid-p', with the average of 10 runs. From top to bottom on *left column*: true Blocks, Doppler, Bumps functions; *middle column*: their noisy versions; *right column*: denoised signals.

We can see that in terms of AMSE, using average distance as the integral value almost surpasses the choice of sequence of ones or sum integral choices. 'Did' and 'Dnw' perform well for most of the function except Blocks and Heavisine. 'Sid' and 'Snw' never appear to be the optimal choice unless the function is the Blocks (piecewise constant). Note that the results for the Heavisine function are comparatively less competitive than for the other test functions, and one reason could be that the 99 edges are below the anti-aliasing sampling rate (equivalent to the Nyquist rate in the case of equally spaced data). As a result, the reconstructed function may contain lower-frequency components instead of the true underlying high-frequency components. This problem can likely be solved by increasing the size of the network sampling, but is outside our scope for investigation here.

AMSE$\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Sid-c	65 (19)	114 (44)	109 (39)	92 (26)	272 (99)	52 (12)
LG-Aid-c	**62** (18)	**113** (48)	**92** (29)	**79** (19)	**198** (66)	46 (10)
LG-Did-c	64 (18)	120 (46)	93 (31)	80 (20)	235 (81)	**45** (11)
LG-Snw-c	66 (19)	117 (47)	109 (38)	97 (28)	282 (103)	53 (12)
LG-Anw-c	64 (18)	116 (48)	**92** (31)	81 (20)	205 (66)	46 (10)
LG-Dnw-c	67 (19)	126 (51)	98 (33)	83 (21)	261 (87)	47 (11)
SNR=5						
LG-Sid-c	23 (7)	**41** (16)	44 (17)	45 (17)	206 (88)	26 (6)
LG-Aid-c	23 (7)	42 (18)	38 (13)	**38** (12)	**138** (48)	**22** (5)
LG-Did-c	23 (7)	46 (20)	**37** (13)	39 (12)	178 (76)	**22** (5)
LG-Snw-c	**22** (7)	42 (17)	44 (17)	47 (18)	217 (91)	26 (6)
LG-Anw-c	23 (8)	43 (19)	38 (14)	39 (12)	147 (48)	**22** (5)
LG-Dnw-c	24 (8)	48 (21)	39 (14)	41 (13)	208 (81)	23 (5)
SNR=7						
LG-Sid-c	11 (3)	**21** (9)	24 (10)	28 (13)	184 (83)	17 (4)
LG-Aid-c	11 (3)	22 (10)	**21** (7)	**23** (8)	**120** (43)	**14** (3)
LG-Did-c	11 (3)	24 (11)	**21** (7)	24 (9)	161 (75)	**14** (3)
LG-Snw-c	**10** (3)	**21** (9)	25 (10)	29 (14)	195 (86)	18 (4)
LG-Anw-c	**10** (3)	22 (12)	22 (8)	24 (8)	130 (43)	**14** (3)
LG-Dnw-c	11 (3)	25 (11)	22 (8)	25 (9)	191 (79)	15 (3)

Table 2.4: AMSE for LG-LOCAAT on a tree structure with 100 nodes and 99 edges. The functions follow the pointwise construction. We assume the coordinate information is available. The values in parentheses are the standard deviations ($\times 10^3$) of the AMSE results across the $P \times R = 50 \times 100$ replications.

From Table 2.5, we can see that the integral choice 'A' provides the best results for variance control, while the choice 'S' provides similar and competitive results, too. Performing the algorithm with the integral 'D' gives a relatively high variance compared with the other two choices. The reason for this might be the dual wavelet functions have fewer overlaps than any other scaling function construction choices (recall that the integral choice 'D' allows us to capture the information more uniformly on the graph structure).

For the bias results, integral choice 'D' surpasses the other methods for most of the functions. The choice 'A' seems competitive for the Heavisine function, which contains high frequency components when compared with other functions. The choice 'S' intro-

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Sid-c	48	65	61	51	68	39
LG-Aid-c	**46**	**64**	**55**	**48**	**64**	**36**
LG-Did-c	49	81	66	56	101	37
LG-Snw-c	49	66	61	52	69	39
LG-Anw-c	**46**	**64**	**55**	**48**	**64**	**36**
LG-Dnw-c	52	86	71	60	113	39
SNR=5						
LG-Sid-c	**18**	**24**	23	22	31	15
LG-Aid-c	**18**	**24**	22	**20**	27	**14**
LG-Did-c	20	33	27	26	67	**15**
LG-Snw-c	19	25	23	22	31	15
LG-Anw-c	**18**	**24**	**21**	**20**	**26**	**14**
LG-Dnw-c	21	34	30	28	78	17
SNR=7						
LG-Sid-c	**9**	**13**	**12**	12	18	9
LG-Aid-c	**9**	**13**	**12**	**11**	**15**	**8**
LG-Did-c	10	18	15	16	57	9
LG-Snw-c	**9**	**13**	**12**	12	18	9
LG-Anw-c	**9**	**13**	**12**	12	**15**	**8**
LG-Dnw-c	10	18	16	17	68	10

Table 2.5: Variance for LG-LOCAAT on a tree structure with 100 nodes and 99 edges. The functions follow the pointwise construction. We assume the coordinate information is available. The values in parentheses are the standard deviations ($\times 10^3$) of the AMSE results across the $P \times R = 50 \times 100$ replications.

duces high bias for the smoother functions (Doppler, Heavisine, `mfc`). Hence, when using coordinate information, overall 'LG-Aid-c' appears to be a balanced choice that delivers competitive results irrespective of signal smoothness and noise contamination level. Although 'LG-Did-c' does not give as good results as 'LG-Aid-c' in terms of AMSE (especially for Blocks and Heavisine) and variance, it is still worth emphasising on since it gives low bias. Let us next investigate the impact of not accessing the coordinate information.

We can see that when performing the algorithm by using the path distance instead of coordinate information, the AMSE results are remarkably comparable to those when coordinate information is available, see Table 2.4. The variance and bias patterns in

Bias$^2 \times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Sid-c	17	49	48	41	204	13
LG-Aid-c	17	49	37	**32**	**134**	9
LG-Did-c	**14**	**39**	**27**	24	**134**	**8**
LG-Snw-c	17	51	48	44	213	14
LG-Anw-c	18	51	37	33	141	10
LG-Dnw-c	15	40	**27**	**23**	148	**8**
SNR=5						
LG-Sid-c	4	17	20	23	175	10
LG-Aid-c	5	18	16	17	112	7
LG-Did-c	**3**	**13**	**10**	**13**	**111**	**6**
LG-Snw-c	4	17	21	25	186	11
LG-Anw-c	5	19	17	18	121	8
LG-Dnw-c	**3**	**13**	**10**	**13**	129	**6**
SNR=7						
LG-Sid-c	2	9	12	16	166	9
LG-Aid-c	2	9	9	11	105	**6**
LG-Did-c	**1**	**7**	**5**	**8**	**104**	**5**
LG-Snw-c	**1**	9	12	16	177	9
LG-Anw-c	2	10	10	12	115	6
LG-Dnw-c	**1**	**7**	**5**	**8**	124	**5**

Table 2.6: Squared bias for LG-LOCAAT on a tree structure with 100 nodes and 99 edges. The functions follow the pointwise construction. We assume the coordinate information is available. The values in parentheses are the standard deviations ($\times 10^3$) of the AMSE results across the $P \times R = 50 \times 100$ replications.

Tables 2.5 and 2.6 appear similar to previous methods. However, should coordinate information be available, this may be the better choice. This may be justified by the design of the function values based on a two-dimensional Euclidean space and it will be essential to inspect the algorithm performance for the edge averaging functions (see next section).

Additionally, the average distance for integral and the inverse distance prediction ('LG-Aid-p') also arise as a strong choice in this context, with the 'Did' choice a close competitor in terms of bias control, and 'Sid' a close match particularly for variance results. The performance of 'LG-Sid-p' is very similar to 'LG-Aid-p' except for Heavisine. So using 'average distances' as integral is advantageous when dealing with high-frequency

AMSE$\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Sid-p	67 (21)	**116** (45)	109 (39)	93 (25)	274 (96)	54 (12)
LG-Aid-p	**65** (20)	117 (48)	**94** (31)	84 (21)	**209** (73)	**47** (10)
LG-Did-p	**65** (19)	122 (49)	95 (31)	**83** (22)	253 (93)	**47** (11)
LG-Snw-p	68 (21)	118 (48)	110 (36)	96 (26)	288 (102)	54 (12)
LG-Anw-p	66 (20)	124 (55)	95 (31)	84 (22)	215 (76)	48 (10)
LG-Dnw-p	68 (20)	129 (54)	100 (33)	86 (22)	276 (93)	48 (11)
SNR=5						
LG-Sid-p	**23** (8)	**42** (17)	44 (17)	45 (16)	208 (83)	27 (7)
LG-Aid-p	**23** (8)	44 (19)	**38** (13)	**40** (13)	**152** (62)	**23** (5)
LG-Did-p	24 (8)	47 (21)	**38** (13)	**40** (13)	192 (86)	**23** (5)
LG-Snw-p	**23** (8)	**42** (19)	44 (16)	47 (17)	220 (87)	27 (6)
LG-Anw-p	24 (8)	46 (22)	39 (13)	**40** (13)	159 (61)	**23** (5)
LG-Dnw-p	25 (8)	49 (22)	40 (14)	42 (13)	219 (89)	24 (5)
SNR=7						
LG-Sid-p	**11** (3)	22 (10)	24 (10)	28 (12)	186 (78)	18 (5)
LG-Aid-p	**11** (3)	23 (11)	**21** (7)	**24** (9)	**135** (58)	**15** (3)
LG-Did-p	**11** (4)	25 (12)	**21** (7)	25 (9)	172 (83)	**15** (3)
LG-Snw-p	**11** (3)	**21** (10)	25 (10)	29 (13)	197 (80)	18 (5)
LG-Anw-p	**11** (3)	24 (13)	22 (8)	**24** (9)	142 (55)	**15** (3)
LG-Dnw-p	**11** (4)	26 (12)	22 (8)	26 (10)	201 (87)	16 (4)

Table 2.7: AMSE for LG-LOCAAT on a tree structure with 100 nodes and 99 edges. The functions follow the pointwise construction. The path distance is used. The values in parentheses are the standard deviations ($\times 10^3$) of the AMSE results across the $P \times R = 50 \times 100$ replications.

information, as opposed to using 'sum of distances' as integral. However, we have to note that using the average distances as integral values is still without theoretical support.

2.5.3.2 Denoising Edge Averaging Functions

Note from Tables 2.10 and 2.13 that when employing our algorithm on edge averaging functions, 'Aid' algorithm is the optimal choice in terms of the AMSE when the coordinate information is known, while 'Did' performs better if only path lengths are known. However, 'Aid' surpasses the other methods for high frequency Heavisine function. Choosing a sequence of ones as the starting integrals decreases the bias for all functions, and the improvements are more significant when we use the path length instead

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Sid-p	49	66	61	52	69	39
LG-Aid-p	**47**	**64**	57	**49**	**63**	**36**
LG-Did-p	51	83	69	60	113	39
LG-Snw-p	49	67	61	52	70	39
LG-Anw-p	**47**	65	**56**	50	64	37
LG-Dnw-p	53	88	73	63	121	40
SNR=5						
LG-Sid-p	19	25	23	22	31	16
LG-Aid-p	**18**	**24**	**22**	**21**	**26**	**15**
LG-Did-p	21	34	29	28	77	17
LG-Snw-p	19	25	23	22	31	16
LG-Anw-p	**18**	25	**22**	**21**	**26**	**15**
LG-Dnw-p	21	35	30	29	85	18
SNR=7						
LG-Sid-p	**9**	**13**	12	**12**	18	9
LG-Aid-p	**9**	**13**	**11**	**12**	**15**	**8**
LG-Did-p	10	18	16	17	65	10
LG-Snw-p	**9**	**13**	12	13	19	9
LG-Anw-p	**9**	**13**	12	**12**	**15**	**8**
LG-Dnw-p	10	19	17	18	74	11

Table 2.8: Variance for LG-LOCAAT on a tree structure with 100 nodes and 99 edges. The functions follow the pointwise construction. The path distance is used. The values in parentheses are the standard deviations ($\times 10^3$) of the AMSE results across the $P \times R = 50 \times 100$ replications.

of the coordinate information. Again, 'Aid' is the optimal choice in terms of variance control. The methods with coordinate information still slightly surpass the corresponding results by the path distance, except for mfc. Hence we recommend to make use of the coordinate information, if it is available. According to AMSE, the performances of 'Did' and 'Aid' are close except when the underlying function has high frequency (e.g., Heavisine).

So overall, we recommend the use of coordinate information when available, as implemented in the methods, LG-Aid-c and LG-Did-c, if the underlying function are spatial coordinate dependent. Should such coordinate information not be available, LG-Aid-p and LG-Did-p are also very competitive throughout the board.

Bias^2	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Sid-p	17	50	48	42	205	14
LG-Aid-p	18	53	37	35	145	11
LG-Did-p	**14**	**39**	**26**	**23**	**140**	**8**
LG-Snw-p	19	51	49	43	218	15
LG-Anw-p	19	58	39	35	151	11
LG-Dnw-p	15	41	27	**23**	155	**8**
SNR=5						
LG-Sid-p	4	17	21	24	177	11
LG-Aid-p	5	20	16	19	126	8
LG-Did-p	**3**	**13**	**9**	**12**	**115**	**6**
LG-Snw-p	4	17	21	25	188	12
LG-Anw-p	5	21	17	19	132	9
LG-Dnw-p	**3**	**13**	**9**	**12**	133	**6**
SNR=7						
LG-Sid-p	2	9	12	16	168	9
LG-Aid-p	2	10	10	12	120	6
LG-Did-p	**1**	**7**	**5**	**8**	**107**	**5**
LG-Snw-p	2	9	12	16	178	9
LG-Anw-p	2	11	10	13	127	7
LG-Dnw-p	**1**	**7**	**5**	**8**	127	**5**

Table 2.9: Squared bias for LG-LOCAAT on a tree structure with 100 nodes and 99 edges. The functions follow the pointwise construction. The path distance is used. The values in parentheses are the standard deviations ($\times 10^3$) of the AMSE results across the $P \times R = 50 \times 100$ replications.

Empirical Computational Cost

The overall running times of the proposed LG-LOCAAT algorithms are recorded as follows. For each algorithm, we report the median running time (hours:minutes:seconds), rounded up to the nearest 30 seconds, and refer it to as the *standard running time* of the algorithms. According to our simulation study, the standard running time is 00:49:30 for 50×100 replications, with 90% of the algorithms completing within 00:52:00. All simulations were conducted using the York Viking high-performance computing facility. Detailed system information can be found at `https://vikingdocs.york.ac.uk/`.

AMSE$\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Sid-c	64 (19)	126 (47)	103 (32)	92 (25)	281 (96)	51 (12)
LG-Aid-c	**59** (16)	**120** (48)	89 (27)	**79** (18)	**206** (70)	45 (10)
LG-Did-c	**59** (16)	**120** (43)	**88** (26)	**79** (19)	228 (77)	**44** (10)
LG-Snw-c	65 (19)	128 (46)	104 (32)	96 (27)	292 (101)	52 (12)
LG-Anw-c	60 (16)	125 (46)	90 (28)	80 (19)	215 (70)	45 (10)
LG-Dnw-c	61 (16)	128 (46)	92 (28)	82 (20)	255 (83)	46 (11)
SNR=5						
LG-Sid-c	26 (9)	50 (22)	45 (17)	45 (16)	220 (86)	25 (6)
LG-Aid-c	**24** (8)	**49** (23)	39 (14)	**38** (11)	**147** (53)	**21** (4)
LG-Did-c	25 (7)	51 (21)	**38** (14)	**38** (12)	176 (71)	**21** (4)
LG-Snw-c	26 (9)	51 (22)	46 (18)	47 (17)	230 (90)	26 (6)
LG-Anw-c	25 (9)	51 (24)	40 (14)	39 (12)	158 (53)	22 (5)
LG-Dnw-c	26 (8)	54 (23)	40 (14)	40 (12)	204 (76)	22 (5)
SNR=7						
LG-Sid-c	**13** (4)	**27** (12)	26 (11)	28 (12)	199 (81)	17 (4)
LG-Aid-c	**13** (4)	**27** (13)	**22** (8)	**23** (8)	**129** (48)	14 (3)
LG-Did-c	**13** (4)	28 (13)	**22** (8)	24 (8)	160 (69)	**13** (3)
LG-Snw-c	**13** (4)	**27** (12)	26 (11)	29 (13)	211 (86)	18 (5)
LG-Anw-c	**13** (5)	**27** (15)	23 (9)	24 (8)	141 (47)	14 (3)
LG-Dnw-c	**13** (5)	30 (14)	23 (8)	25 (9)	190 (74)	15 (3)

Table 2.10: AMSE for LG-LOCAAT on a tree structure with 100 nodes and 99 edges. The functions follow the edge-averaging construction. We assume the coordinate information is available. The values in parentheses are the standard deviations ($\times 10^3$) of the AMSE results across the $P \times R = 50 \times 100$ replications.

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Sid-c	45	65	56	51	67	38
LG-Aid-c	**42**	**62**	52	**47**	**64**	**36**
LG-Did-c	46	78	60	55	99	37
LG-Snw-c	46	65	56	52	67	38
LG-Anw-c	**42**	63	**51**	48	**64**	**36**
LG-Dnw-c	48	84	65	59	111	39
SNR=5						
LG-Sid-c	18	26	23	22	30	15
LG-Aid-c	**17**	**25**	**21**	**20**	27	**14**
LG-Did-c	20	35	27	26	65	15
LG-Snw-c	18	26	23	22	30	15
LG-Anw-c	18	26	**21**	**20**	**26**	**14**
LG-Dnw-c	21	37	29	28	77	17
SNR=7						
LG-Sid-c	10	14	**12**	12	18	8
LG-Aid-c	**9**	**13**	**12**	**11**	**15**	8
LG-Did-c	11	19	15	16	55	9
LG-Snw-c	10	14	**12**	12	17	9
LG-Anw-c	**9**	**13**	**12**	**11**	**15**	8
LG-Dnw-c	11	21	17	17	67	10

Table 2.11: Variance for LG-LOCAAT on a tree structure with 100 nodes and 99 edges. The functions follow the edge-averaging construction. We assume the coordinate information is available. The values in parentheses are the standard deviations ($\times 10^3$) of the AMSE results across the $P \times R = 50 \times 100$ replications.

Bias^2	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Sid-c	19	61	47	41	215	12
LG-Aid-c	17	58	37	31	142	9
LG-Did-c	**13**	**42**	**28**	**23**	**130**	**7**
LG-Snw-c	20	63	48	44	224	13
LG-Anw-c	18	62	39	32	152	9
LG-Dnw-c	**13**	44	**28**	**23**	144	**7**
SNR=5						
LG-Sid-c	7	25	22	24	190	10
LG-Aid-c	7	24	18	18	120	7
LG-Did-c	**5**	**17**	**11**	**12**	**111**	**6**
LG-Snw-c	7	25	23	25	201	11
LG-Anw-c	8	25	19	19	131	7
LG-Dnw-c	**5**	**17**	**11**	**12**	128	**6**
SNR=7						
LG-Sid-c	3	13	14	16	181	9
LG-Aid-c	3	13	11	12	114	6
LG-Did-c	**2**	**9**	**6**	**8**	**105**	**4**
LG-Snw-c	3	14	14	17	193	9
LG-Anw-c	4	14	11	12	126	6
LG-Dnw-c	**2**	**9**	**6**	**8**	123	5

Table 2.12: Squared bias for LG-LOCAAT on a tree structure with 100 nodes and 99 edges. The functions follow the edge-averaging construction. We assume the coordinate information is available. The values in parentheses are the standard deviations ($\times 10^3$) of the AMSE results across the $P \times R = 50 \times 100$ replications.

AMSE$\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
			SNR=3			
LG-Sid-p	66 (20)	127 (47)	104 (32)	93 (25)	282 (95)	53 (12)
LG-Aid-p	62 (17)	123 (47)	91 (28)	83 (21)	**218 (76)**	**46 (10)**
LG-Did-p	**60** (17)	**122** (45)	**90** (27)	**82** (21)	247 (87)	**46** (11)
LG-Snw-p	66 (21)	129 (48)	104 (30)	96 (26)	296 (101)	53 (12)
LG-Anw-p	62 (17)	133 (51)	92 (28)	84 (22)	225 (77)	47 (10)
LG-Dnw-p	62 (17)	130 (49)	94 (28)	85 (22)	270 (89)	47 (11)
			SNR=5			
LG-Sid-p	26 (9)	**51** (23)	45 (17)	46 (16)	222 (85)	26 (7)
LG-Aid-p	26 (9)	52 (24)	**39** (14)	40 (12)	**162 (64)**	**22 (5)**
LG-Did-p	**25** (8)	52 (22)	**39** (14)	**39 (12)**	190 (82)	**22 (5)**
LG-Snw-p	26 (9)	52 (23)	46 (17)	47 (17)	233 (88)	27 (6)
LG-Anw-p	26 (9)	55 (27)	41 (14)	40 (12)	170 (63)	23 (5)
LG-Dnw-p	26 (8)	55 (25)	41 (15)	41 (13)	217 (85)	23 (5)
			SNR=7			
LG-Sid-p	**13** (5)	**27** (13)	26 (11)	28 (12)	202 (81)	18 (5)
LG-Aid-p	**13** (4)	28 (15)	**22** (8)	**24** (9)	**145** (60)	**14** (3)
LG-Did-p	**13** (4)	29 (14)	**22** (8)	**24** (9)	173 (78)	**14** (3)
LG-Snw-p	**13** (5)	**27** (13)	26 (11)	29 (12)	212 (81)	18 (5)
LG-Anw-p	14 (5)	29 (16)	23 (8)	25 (9)	154 (57)	15 (3)
LG-Dnw-p	**13** (5)	31 (15)	23 (9)	26 (9)	201 (83)	16 (3)

Table 2.13: AMSE for LG-LOCAAT on a tree structure with 100 nodes and 99 edges. The functions follow the edge-averaging construction. The path distance is used. The values in parentheses are the standard deviations ($\times 10^3$) of the AMSE results across the $P \times R = 50 \times 100$ replications.

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Sid-p	46	65	56	52	67	39
LG-Aid-p	**43**	**63**	53	**49**	**63**	**36**
LG-Did-p	48	81	63	59	111	39
LG-Snw-p	46	65	56	52	69	39
LG-Anw-p	**43**	64	**52**	**49**	64	**36**
LG-Dnw-p	49	86	67	62	119	40
SNR=5						
LG-Sid-p	19	26	23	22	30	**15**
LG-Aid-p	**18**	**25**	**21**	**21**	**26**	**15**
LG-Did-p	20	36	29	28	76	17
LG-Snw-p	19	26	23	22	30	16
LG-Anw-p	**18**	26	22	**21**	**26**	**15**
LG-Dnw-p	21	38	30	29	84	18
SNR=7						
LG-Sid-p	10	**14**	**12**	**12**	18	9
LG-Aid-p	**9**	**14**	**12**	**12**	**15**	**8**
LG-Did-p	11	20	16	17	65	10
LG-Snw-p	10	**14**	13	**12**	18	9
LG-Anw-p	10	**14**	**12**	**12**	**15**	**8**
LG-Dnw-p	12	21	17	18	74	11

Table 2.14: Variance for LG-LOCAAT on a tree structure with 100 nodes and 99 edges. The functions follow the edge-averaging construction. The path distance is used. The values in parentheses are the standard deviations ($\times 10^3$) of the AMSE results across the $P \times R = 50 \times 100$ replications.

Bias^2	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Sid-p	20	62	47	42	215	14
LG-Aid-p	18	61	38	34	155	10
LG-Did-p	**13**	**41**	**27**	**23**	**136**	**7**
LG-Snw-p	20	64	48	44	227	14
LG-Anw-p	20	69	40	35	162	10
LG-Dnw-p	**13**	44	28	**23**	151	**7**
SNR=5						
LG-Sid-p	8	25	22	24	192	11
LG-Aid-p	8	26	18	19	135	8
LG-Did-p	**4**	**16**	**11**	**12**	**115**	**6**
LG-Snw-p	8	26	23	25	203	11
LG-Anw-p	8	29	19	19	144	8
LG-Dnw-p	5	17	**11**	**12**	133	**6**
SNR=7						
LG-Sid-p	3	14	13	16	185	9
LG-Aid-p	4	15	11	12	130	6
LG-Did-p	**2**	**9**	**6**	**7**	**108**	**4**
LG-Snw-p	3	14	14	17	194	9
LG-Anw-p	4	15	12	13	138	6
LG-Dnw-p	**2**	**9**	**6**	8	127	5

Table 2.15: Squared bias for LG-LOCAAT on a tree structure with 100 nodes and 99 edges. The functions follow the edge-averaging construction. The path distance is used. The values in parentheses are the standard deviations ($\times 10^3$) of the AMSE results across the $P \times R = 50 \times 100$ replications.

Chapter 3

E-LOCAAT: An Edge-Centred Scheme

In the previous chapter we constructed a variant of LOCAAT algorithm, which we called line graph LOCAAT (LG-LOCAAT), for dealing with the data collected from the edge set. The proposed LG-LOCAAT relies on a transformation from the graph G to the associated line graph G^*. We pointed out that, at a particular stage, the relinkage of the LG-LOCAAT might lead to a graph which cannot be inverted into the original graph domain. Therefore, we further seek an alternative algorithm which allows us to perform the lifting scheme directly on the original edge-domain.

In this chapter, we introduce a new edge-centred scheme, E-LOCAAT, which adapts the lifting-one-coefficient-at-a-time paradigm but works directly for the edge collected data on the original graph G.

3.1 E-LOCAAT Framework and Setup

Before formally starting, we first remark that the distance we employ in E-LOCAAT will be determined by the path metric described in Section 1.4.4, as the construction of the E-LOCAAT framework is heavily based on the metrized version of the original graph. Our work is partially motivated by the previous work from Schröder and

Sweldens (1995a) that discussed two different approaches for representing functions defined on non-regular topology by wavelet lifting (specifically, sphere triangulation), which they refer to as interpolating-point bases and face bases. Wavelet constructions generated by interpolating scaling functions and characteristic functions are applied to these two approaches (vertex-based and face-based), respectively. By taking advantage of the recursive construction of the LOCAAT framework, we will see later that linear interpolating functions (as initial scaling functions) or characteristic functions (as initial scaling functions) will give the same result (detail coefficient sequence). We will also introduce a biorthogonal Haar construction by imposing a condition on the scaling functions at different stages (based on characteristic functions, see Section 3.2). For avoiding confusion, where appropriate we use the terminology 'interpolating point' instead of 'vertex' in our work.

Similar in structure to the discussion in Section 2.3, our work will start from the function expansion form and then move onto the proposed algorithm. Our aim is to construct a method that works on the original graph space, which we define in terms of the metrized graph space introduced in Section 1.4.4, instead of using a representation in a different space such as the line graph space. Such a construction may circumvent problems associated to the line graph space, refer to the invertibility problem discussed in Section 2.3.4. First, let us start by explaining the setup for our study, particularly the approach for defining the initial scaling functions.

3.1.1 Interpolating-point Bases and Function Representations

Recall we have the graph (tree) $G = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = \{v_1, ..., v_n\}$ and $\mathcal{E} = \{e_1, ..., e_m\}$, where $m = n - 1$. We further assume that the length of each edge is known. Thus, a metrized graph $\Gamma(G)$ (or put simply, Γ) can be obtained. We also simply denote the vertex set of the metrized original graph be $\mathrm{Vex}_G(\Gamma) = \mathcal{V}^{\mathrm{met}}$. Then we consider the following steps.

1. For the metrized form $e_k^{\text{met}} = [v_i, v_j]$ of an edge $e_k = \{v_i, v_j\}$, we interpolate the midpoint $\mathbf{p}_{ij}$ and assume this has an equivalent representation $\mathbf{p}_k$ on e_k^{met} (sometimes we use both notations interchangeably).

2. We carry out a subdivision by considering the set of points $\{\mathbf{p}_k\}_{k\in\{1,\dots,m\}}$ as a set of interpolating vertices. Thus, we have a new graph with $\mathcal{V}'^{\text{met}} = \mathcal{V}^{\text{met}} \cup \{\mathbf{p}_k\}_{k\in\{1,\dots,m\}}$ as the vertex set, and the cardinality of the vertex set $\mathcal{V}'$ is $(n+m)$. It is easy to see that each interpolating point separates the metrized edge into two different edges. For each metrized edge $e_k^{\text{met}} = [v_i, v_j]$, we denote $e'^{\text{met}}_{2k-1} = [\mathbf{p}_{v_i}, \mathbf{p}_k]$ and $e'^{\text{met}}_{2k} = [\mathbf{p}_k, v_j]$ as the new edges resulting from the subdivision of e_k^{met} (regardless of which one corresponds to either v_i or v_j, while $i \neq j$). Then we denote the set of edges following the construction of interpolating vertices as $\mathcal{E}'^{\text{met}} = \{e'^{\text{met}}_{k'}\}_{k'\in\{1,\dots,2m\}}$, and further denote the corresponding resulting graph as $G' = (\mathcal{V}', \mathcal{E}')$. Using $n = m+1$, the graph G' is still a tree since $2m = (n+m) - 1$. Note that the total length of the edges is preserved; thus G' is also a model of the metrized graph Γ, and since $\mathcal{V}^{\text{met}} \subset \mathcal{V}'^{\text{met}}$, G' is a refinement of G, denoted as $G \sim G'$, and this scheme produces a subdivision.

As the refined graph G' and the original graph G both correspond to the same metrized graph space Γ, we can construct a set of *interpolating-point-based* scaling functions for $\{\mathbf{p}_k\}_{k\in\{1,\dots,m\}}$. To be specific, we let the scaling functions on $\Gamma(G')$ satisfy the following conditions.

1. The initial scaling function $\varphi^{\Gamma}_{k,m}$ has to be interpolating among the set $\mathcal{V}^{\text{met}} \cup \{\mathbf{p}_k\}_{k=1}^m$. Therefore, $\varphi^{\Gamma}_{k,m}(\mathbf{p}_k) = 1$, and $\varphi^{\Gamma}_{k,m}(\mathbf{p}) = 0$ for those $\mathbf{p} \in \mathcal{V}^{\text{met}} \cup \{\mathbf{p}_k\}_{k=1}^m$; $\mathbf{p} \neq \mathbf{p}_k$.

2. The closure of the union of the support of all scaling functions associated with $\{\mathbf{p}_k\}_{k=1}^m$ covers the space Γ.

Note that we write $\varphi^{\Gamma}_{k,m}$ to indicate that the analytical form of the scaling functions is built upon the metrized graph domain Γ. An immediate way to ensure these conditions is

to define a partitioning and design primal scaling functions as the characteristic functions of associated partitioning blocks. A possible choice is to simply let the partition be $\{e_k^{\text{met}}\}_{k=1}^m$. Akin to the idea of Schröder and Sweldens (1995a), we propose a set of interpolating-point bases, which are generalised triangular functions defined on edges (see the detail below).

For the refined graph G', we construct initial scaling functions based on the set of interpolating points $\{\mathbf{p}_k\}_{k\in\{1,\dots,m\}}$ since each interpolating point can represent an associated edge. Since there is an isometry between e_k^{met} and the interval $[0, \ell_k]$, we can first design the scaling function (for the k-th edge) on the interval $[0, \ell_k]$, and then map it onto e_k^{met}. Hence, first we can design a triangular function centred at the middle point of the interval $[0, \ell_k]$, with the interval as the support. A standard triangular function is defined as

$$\mathbf{tri}(x) = \begin{cases} 1 - |x|, & \text{if } -1 \le x \le 1; \\ 0, & \text{otherwise.} \end{cases}$$

For fitting its domain into the interval $[0, \ell_k]$, we only have to perform the scaling and the translation of the function to obtain $\mathbf{tri}(y)$, where $y = \frac{\ell_k}{2}(x+1)$. Thus, we have that

$$\mathbf{tri}_{k,m}(y) = \begin{cases} 1 - \frac{|y-\ell_k/2|}{\ell_k/2}, & \text{if } 0 \le y \le \ell_k; \\ 0, & \text{otherwise.} \end{cases}$$

Notice that $|y - \ell_k/2|$ can be considered as the distance between the point y and the middle point $(\ell_k/2)$ on the support interval. Then if we substitute it by the associated path distance on Γ, the triangular function can be generalised to the one on the metrized edge e_k^{met}. We take this triangular function on e_k^{met} as the k-th initial primal scaling function, and we define

$$\tilde{\varphi}_{k,m}^{\Gamma,\text{vertex}}(\mathbf{p}) = \delta(\mathbf{p} - \mathbf{p}_k); \tag{3.1.1}$$

$$\varphi_{k,m}^{\Gamma,\text{vertex}}(\mathbf{p}) = \begin{cases} 1 - \frac{\text{dist}_{\text{path}}^{\Gamma}(\mathbf{p},\mathbf{p}_k)}{\ell_k/2}, & \text{if } \mathbf{p} \in e_k^{\text{met}}; \\ 0, & \text{if } \mathbf{p} \notin e_k^{\text{met}}, \end{cases} \tag{3.1.2}$$

where $\delta(\mathbf{p}-\mathbf{p}_k)$ is the Dirac delta, and '$\text{dist}^{\Gamma}_{\text{path}}(\mathbf{p},\mathbf{p}_k)$' is the path distance between $\mathbf{p}$ and $\mathbf{p}_k$ on the metrized graph Γ, as described in Section 1.4.4. Since we have that $\varphi^{\Gamma,\text{vertex}}_{k,m}(\mathbf{p}_k)=1$ (by the fact that $\text{dist}^{\Gamma}_{\text{path}}(\mathbf{p}_k,\mathbf{p}_k)=0$), and $\varphi^{\Gamma,\text{vertex}}_{k,m}(\mathbf{p}_{k'})=0$ if $k'\neq k$ ($\mathbf{p}_{k'}\notin e^{\text{met}}_k$ if $k'\neq k$), the initial primal scaling functions are interpolating. The dual scaling functions and primal scaling functions also satisfy the biorthogonality property, such that $\langle\tilde{\varphi}^{\Gamma,\text{vertex}}_{k,m},\varphi^{\Gamma,\text{vertex}}_{k',m}\rangle=\delta_{kk'}$, where $\delta_{kk'}$ is the Kronecker delta. For the function defined on the edges, $g^{\mathcal{E}}$, now we can represent it by the metrized graph, g^{Γ}, where $g^{\mathcal{E}}_k=g^{\Gamma}_k=g^{\Gamma}(\mathbf{p}_k)$, for $k\in\{1,...,m\}$. Hence $\langle\tilde{\varphi}^{\Gamma,\text{vertex}}_{k,m},g^{\Gamma}\rangle=g^{\Gamma}_k$, which allows us to have the function representation

$$g^{\mathcal{E}}(e)=g^{\Gamma}(\mathbf{p})=\sum_{k=1}^{m}c^{\Gamma}_{k,m}\varphi^{\Gamma}_{k,m}(\mathbf{p}), \tag{3.1.3}$$

where $\mathbf{p}\in\Gamma$, and the initial scaling coefficient is given by $c^{\Gamma}_{k,m}:=\langle\tilde{\varphi}^{\Gamma}_{k,m},g^{\Gamma}\rangle=g^{\Gamma}_k$. In addition, we also construct a similar 'lazy wavelet' to the construction described in Section 2, whose initial dual and primal scaling functions are given by

$$\tilde{\varphi}^{\Gamma,\text{Delta}}_{k,m}(\mathbf{p})=\delta(\mathbf{p}-\mathbf{p}_k); \tag{3.1.4}$$

$$\varphi^{\Gamma,\text{Delta}}_{k,m}(\mathbf{p})=\delta_{\mathbf{p}_k,\mathbf{p}}, \tag{3.1.5}$$

where $\delta_{\mathbf{p}_k,\mathbf{p}}$ is the Kronecker delta, which fulfills the biorthogonality requirement between dual and primal scaling functions. Thus, the initial function expansion form can be written exactly the same with equation (3.1.3). Note that in contrast to LG-LOCAAT, the scaling functions are defined on the metrized graph Γ associated to the original graph G instead of the line graph. Thus, the analytical forms of the scaling and wavelet functions are defined in the original domain.

3.1.2 Edge Bases and Function Representations

Now let us introduce a construction of edge bases, as a variant of the face bases introduced by Schröder and Sweldens (1995a).

We still start with the metrized graph Γ for the graph G, and recall that for all $k\in\{1,...,m\}$, we denote $e^{\text{met}}_k=[v_i,v_j]$ as the metrized version of the k-th edge

$e_k = \{v_i, v_j\}$, and we further assume that the function values satisfy that $g^{\mathcal{E}}(e_k) = g^{\Gamma}(\mathbf{p})$, if $\mathbf{p} \in e_k^{\text{met}} \backslash \{\mathbf{p}_{v_i}, \mathbf{p}_{v_j}\}$. As a convention, we let $g^{\Gamma}(\mathbf{p}_{v_i}) = 0$ for all i such that $v_i \in \mathcal{V}$, otherwise some singularities might appear since two metrized edges might have one point intersection. Before designing the scaling functions, we have to find a suitable partitioning for the metrized graph domain $\Gamma(G)$, since the domain of each scaling function should lie in one of the partitionings, see Jawerth and Sweldens (1994). For the k-th edge $e_k = \{v_i, v_j\}$, let $e_k^{\text{met}} = [\mathbf{p}_{v_i}, \mathbf{p}_{v_j}]$ be its metrized version, then denote $\mathbf{P}_k = \text{int}(e_k^{\text{met}})$ as the interior of e_k^{met}, which is $e_k^{\text{met}} \backslash \{\mathbf{p}_{v_i}, \mathbf{p}_{v_j}\}$. Then the set $\{\mathbf{P}_k\}_{k\in\{1,\dots,m\}}$ forms a partitioning of the metrized graph $\Gamma(G)$, since the set satisfies that

$$\begin{aligned} &\mathbf{P}_k \cap \mathbf{P}_{k'} = \emptyset, \quad \text{if } k \neq k'; \\ &\overline{\bigcup_{k=1}^{m} \mathbf{P}_k} = \Gamma. \end{aligned}$$

Recall that (from Section 1.4.4) there is an isometry of $e_k^{\text{met}} = [\mathbf{p}_{v_i}, \mathbf{p}_{v_j}]$ with $[0, \ell_k]$, where ℓ_k is the length of the k-th edge. Then similarly, there is an isometry of $\mathbf{P}_k = \text{int}(e_k^{\text{met}})$ with $(0, \ell_k)$, where $(0, \ell_k)$ is the open interval associated with 0 and ℓ_k. The reason for taking the interior is to make sure that the linear combination of primal scaling function will not create spikes at these metrized vertex points. Then we consider the dual and primal scaling functions as

$$\tilde{\varphi}_{k,m}^{\Gamma,\text{edge}}(\mathbf{p}) = \frac{1}{\mu(\mathbf{P}_k)} \chi_{\mathbf{P}_k}(\mathbf{p}) := \frac{1}{\ell_k} \chi_{(0,\ell_k)}, \tag{3.1.6}$$

$$\varphi_{k,m}^{\Gamma,\text{edge}}(\mathbf{p}) = \chi_{\mathbf{P}_k}(\mathbf{p}) := \chi_{(0,\ell_k)}, \tag{3.1.7}$$

where χ is the characteristic function, and $\mu(\mathbf{P}_k)$ is the Lebesgue measure of the interval $(0, \ell_k)$, which is simply the length of its closure. Hence, we have

$$\begin{aligned} \langle \tilde{\varphi}_{k,m}^{\Gamma,\text{edge}}, \varphi_{k,m}^{\Gamma,\text{edge}} \rangle &= \int_{\Gamma} \tilde{\varphi}_{k,m}^{\Gamma,\text{edge}}(\mathbf{p}) \varphi_{k,m}^{\Gamma,\text{edge}}(\mathbf{p}) d\mathbf{p} \\ &= \int_{\mathbb{R}} \frac{1}{\ell_k} \chi_{(0,\ell_k)}^2(y) dy \\ &= \frac{1}{\ell_k} \int_0^{\ell_k} 1 dy \\ &= 1, \end{aligned}$$

and for $k \neq k'$, we have $\langle \tilde{\varphi}^{\Gamma,\text{edge}}_{k,m}, \varphi^{\Gamma,\text{edge}}_{k',m} \rangle = 0$ since $\tilde{\varphi}^{\Gamma,\text{edge}}_{k,m}$ and $\varphi^{\Gamma,\text{edge}}_{k',m}$ have non-overlapping support. This guarantees that the dual and primal scaling functions are still biorthogonal. Moreover, we have that

$$\begin{aligned}
\langle \tilde{\varphi}^{\Gamma,\text{edge}}_{k,m}, g^{\Gamma} \rangle &= \int_{\Gamma} \tilde{\varphi}^{\Gamma,\text{edge}}_{k,m}(\mathbf{p}) g^{\Gamma}(\mathbf{p}) d\mathbf{p} \\
&= \int_{\mathbb{R}} g^{\Gamma}_{k} \frac{1}{\ell_k} \chi_{(0,\ell_k)}(y) dy \\
&= g^{\Gamma}_{k} \left(\frac{1}{\ell_k} \int_{0}^{\ell_k} 1 dy \right) \\
&= g^{\Gamma}_{k},
\end{aligned}$$

which allows us to start with the initial scaling coefficients defined as $c^{\Gamma}_{k,m} := g^{\Gamma}_{k}$, and the functional expansion at stage-m can be written similarly to equation (3.1.3). Compared with interpolating-point bases, edge bases do not use points as a representation of edges, but treat each edge as a set of points instead. We will see later that edge bases will lead to some interesting constructions.

3.1.3 E-LOCAAT Algorithm Steps

From now on in this chapter, we denote the length of k-th edge as $\ell_{k,m}$ initially (at stage-m), since their values will be updated through the algorithm. The initial neighbourhood structure for k-th edge is defined as

$$\mathcal{N}^{\mathcal{E}}_{k,m} = \left\{ e_s \,\middle|\, |e_k \cap e_s| = 1 \right\},$$

which encompasses the edges that have a common vertex with the edge e_k. Although we discussed and designed the initial scaling function by means of two different aspects (interpolating-point- and edge-bases), there are only a few steps where these choices impact the lifting algorithm. Thus, we will point out the differences where these occur. The E-LOCAAT algorithm steps can be formalised as follows.

- **Split:** Recall that for the LOCAAT-based algorithm, the task for the 'split' step is to first determine the integral values of the scaling functions at the initial stage,

which is stage-m here. For the interpolating-point bases, we have that

$$\begin{aligned} I_{k,m}^{\Gamma,\text{vertex}} &= \int_{\Gamma} \varphi_{k,m}^{\Gamma,\text{vertex}}(\mathbf{p}) d\mathbf{p} \\ &\equiv \int_{0}^{\ell_{k,m}} \mathbf{tri}_{k,m}(y) dy \\ &= \int_{0}^{\ell_{k,m}} \left(1 - \frac{|y - \ell_{k,m}/2|}{\ell_{k,m}/2}\right) dy \\ &= 2\int_{\ell_{k,m}/2}^{\ell_{k}} \left(1 - \frac{y - \ell_{k,m}/2}{\ell_{k,m}/2}\right) dy \\ &= 2\int_{\ell_{k,m}/2}^{\ell_{k,m}} \left(2 - \frac{2y}{\ell_{k,m}}\right) dy \\ &= 2\left[2\left(\ell_{k,m} - \frac{\ell_{k,m}}{2}\right) - \frac{2}{\ell_{k,m}}\left(\ell_{k,m}^2 - \frac{\ell_{k,m}^2}{4}\right)\right] \\ &= \frac{\ell_{k,m}}{2}. \end{aligned} \tag{3.1.8}$$

The integral values of the initial primal scaling functions can also be set as

$$I_{k,m}^{\Gamma,\text{Delta}} = 1, \tag{3.1.9}$$

for the case that Kronecker Delta as the initial primal scaling functions.

For our proposed edge bases construction with the choices of the primal scaling functions mentioned above, the initial integral value of the k-th edge scaling function can be obtained by

$$\begin{aligned} I_{k,m}^{\Gamma,\text{edge}} &= \int_{\Gamma} \varphi_{k,m}^{\Gamma,\text{edge}}(\mathbf{p}) d\mathbf{p} \\ &= \int_{\mathbb{R}} \chi_{[0,\ell_{k,m}]}(y) dy \\ &= \ell_{k,m}. \end{aligned} \tag{3.1.10}$$

Note that according to Proposition 2.3.1, the choices of scaling functions for vertex and edge bases will not impact the lifting steps, hence we will use equation (3.1.8). Thus, from now on, we skip the superscript (vertex/edge/Delta) for convenience.

- **Predict:** Recall that for each $e_k \in \mathcal{E}$, we have an associated initial scaling coefficient value $c_{k,m}^{\Gamma} := g_k^{\Gamma}$ on the metrized graph domain Γ. To ensure full generality, we

present this algorithm at stage-r. Similar to LOCAAT and LG-LOCAAT, the prediction of the removal edge, e_{k_r}, at stage-r can be obtained by a linear combination of the values of neighbouring coefficients $\{c^{\Gamma}_{s,r}\}_{s:e_s\in\mathcal{N}^{\mathcal{E}}_{k_r,r}}$, where

$$\mathcal{N}^{\mathcal{E}}_{k_r,r} = \{e_s \,|\, e_s \text{ and } e_{k_r} \text{ share only one common vertex at stage-}r\},$$

is the set of edge(s) neighbouring e_{k_r}. The detail coefficient $d^{\Gamma}_{k_r}$ is then obtained by taking the difference between the k_r-th coefficient at stage-r and its prediction, which is

$$d^{\Gamma}_{k_r} = c^{\Gamma}_{k_r,r} - \sum_{s:\, e_s\in\mathcal{N}^{\mathcal{E}}_{k_r,r}} a^{\Gamma}_{s,r}c^{\Gamma}_{s,r}, \tag{3.1.11}$$

where $\sum_{s:\, e_s\in\mathcal{N}^{\mathcal{E}}_{k_r,r}} a^{\Gamma}_{s,r}c^{\Gamma}_{s,r}$ is the prediction for $c^{\Gamma}_{k_r,r}$, and $\{a^{\Gamma}_{s,r}\}_{s:\, e_s\in\mathcal{N}^{\mathcal{E}}_{k_r,r}}$ are the prediction weights. The criterion for the prediction weights is that $\sum_{s:\, e_s\in\mathcal{N}^{\mathcal{E}}_{k_r,r}} a^{\Gamma}_{s,r} = 1$ and we still consider the normalised inverse distances as the prediction weights suggested by Jansen et al. (2009). However, the interpolating-point bases and edge bases will give rise to different distance measures, as follows.

– **Distance for interpolating-point bases:**

  For the interpolating-point bases, the distance between two neighbouring edges e_k and e_s, can be simply represented as the path distance between their metrized version $\mathbf{p}_k$ and $\mathbf{p}_s$, on the metrized graph Γ. Instead of fixing the distance measure at the beginning as we did in LG-LOCAAT, we define the stage-r distance as the path distance such that

$$\mathrm{dist}^{\Gamma,r}_{\mathrm{path}}(\mathbf{p}_k,\mathbf{p}_s) = \frac{\ell_{k,r}+\ell_{s,r}}{2}, \tag{3.1.12}$$

  for all s,k, such that e_k and e_s that neighbouring to each other. Notice that we will only use the edges in the neighbourhood of e_{k_r} at stage-r, thus we do not have to calculate the distance between two non-neighbouring edges at each stage.

– **Distance for edge bases:**

Recall that for edge bases, we consider each edge as a set of points (as its metrized version). Thus, we inevitably have to find a metric which gives not only distance between two points, but preferably between two sets. A popular choice for such metric is the *Hausdorff distance*. Hausdorff distance and some of its variants are widely used in image analysis, see Huttenlocher et al. (1993), Zhao et al. (2005), and Karimi and Salcudean (2019). Hausdorff distance is a metric for measuring the distance between non-empty sets (O'Searcoid; 2006), which is defined as follows.

Definition 3.1.1 (Hausdorff Distance)**.** If (X, dist) is a metric space, let $\mathcal{A}$ and $\mathcal{B}$ be two non-empty closed bounded subsets of X, then the Hausdorff distance between $\mathcal{A}$ and $\mathcal{B}$ is defined as

$$\mathrm{dist}_H(\mathcal{A},\mathcal{B}) = \max\left\{ \sup_{x\in\mathcal{A}}\left\{ \inf_{y\in\mathcal{B}}\{\mathrm{dist}(x,y)\}\right\}, \sup_{y\in\mathcal{B}}\left\{ \inf_{x\in\mathcal{A}}\{\mathrm{dist}(x,y)\}\right\}\right\},$$

where the H of dist_H stands for Hausdorff.

If we denote $\mathcal{K}(X)$ as the collection of all subsets of X, then $(\mathcal{K}(X), \mathrm{dist}_H)$ forms a metric space, see O'Searcoid (2006). Denote by Γ_r the metrized version for the graph at stage-r, and by $\mathrm{dist}_{\mathrm{path}}^{\Gamma_r,r}$ the path metric defined as in equation (3.1.12). Since $(\Gamma_r, \mathrm{dist}_{\mathrm{path}}^{\Gamma_r,r})$ is a metric space, then at stage-r, if $e_s = \{v_j, v_l\}$ is neighbouring $e_{k_r} = \{v_i, v_j\}$, then the Hausdorff distance between their metrized versions is

$$\begin{aligned}
&\mathrm{dist}_H(e_{k_r}^{\mathrm{met}}, e_s^{\mathrm{met}}) \\
&= \max\left\{ \sup_{\mathbf{p}\in e_{k_r}^{\mathrm{met}}}\left\{ \inf_{\mathbf{p}'\in e_s^{\mathrm{met}}}\left\{\mathrm{dist}_{\mathrm{path}}^{\Gamma_r,r}(\mathbf{p},\mathbf{p}')\right\}\right\}, \sup_{\mathbf{p}'\in e_s^{\mathrm{met}}}\left\{ \inf_{\mathbf{p}\in e_{k_r}^{\mathrm{met}}}\left\{\mathrm{dist}_{\mathrm{path}}^{\Gamma_r,r}(\mathbf{p},\mathbf{p}')\right\}\right\}\right\} \\
&= \max\left\{ \sup_{\mathbf{p}\in e_{k_r}^{\mathrm{met}}}\left\{\mathrm{dist}_{\mathrm{path}}^{\Gamma_r,r}(\mathbf{p},\mathbf{p}_{v_j})\right\}, \sup_{\mathbf{p}'\in e_s^{\mathrm{met}}}\left\{\mathrm{dist}_{\mathrm{path}}^{\Gamma_r,r}(\mathbf{p}',\mathbf{p}_{v_j})\right\}\right\} \\
&= \max\left\{\ell_{k_r,r},\, \ell_{s,r}\right\},
\end{aligned}$$

which is easily dealt with computationally.

Once we obtain the distance measure, the prediction weights can then be constructed as

$$a_s^{\Gamma,\text{vertex}} = \frac{1/(l_{k_r,r} + l_{s,r})}{\sum_{j:\, e_j \in \mathcal{N}^{\mathcal{E}}_{k_r,r}} 1/(l_{k_r,r} + l_{j,r})}, \quad \text{for } s:\, e_j \in \mathcal{N}^{\mathcal{E}}_{k_r,r}, \tag{3.1.13}$$

$$a_s^{\Gamma,\text{edge}} = \frac{1/\max\{\ell_{k_r,r},\, \ell_{s,r}\}}{\sum_{j:\, e_j \in \mathcal{N}^{\mathcal{E}}_{k_r,r}} 1/\max\{\ell_{k_r,r},\, \ell_{j,r}\}}, \quad \text{for } s:\, e_j \in \mathcal{N}^{\mathcal{E}}_{k_r,r}. \tag{3.1.14}$$

The superscript 'vertex' and 'edge' indicate interpolating-point bases and edge bases, respectively. Recall that for the edge bases, the removal choice is based on the edge with the minimal length, such that $\ell_{k_r,r} \leq \ell_{t,r}$ for all $t \in \mathcal{N}^{\mathcal{E}}_{k_r,r}$. Thus, the prediction weights for edge bases can simply be represented as the normalised inverse length, which is

$$a_s^{\Gamma,\text{edge}} = \frac{1/\ell_{s,r}}{\sum_{j:\, e_j \in \mathcal{N}^{\mathcal{E}}_{k_r,r}} 1/\ell_{j,r}}, \quad \text{for } s:\, e_j \in \mathcal{N}^{\mathcal{E}}_{k_r,r}. \tag{3.1.15}$$

Note that if the edge removal choice is not determined by the length as the integral measure, then the prediction weights will not be exactly inverse length as in equation (3.1.15), for example, if we perform the split along with a sequence of ones as the integral values. Nevertheless, we still construct the prediction weights as in equation (3.1.15) because of its computational convenience, and similar to LG-LOCAAT, we will perform the moving average in the proposed E-LOCAAT, in which the prediction weights are

$$a_s^{\Gamma,\text{edge}} = \frac{1}{|\mathcal{N}^{\mathcal{E}}_{k_r,r}|}, \quad \text{for } s:\, e_j \in \mathcal{N}^{\mathcal{E}}_{k_r,r}. \tag{3.1.16}$$

- **Update:** For LOCAAT-based methods for vertex-based data, there are three quantities to be updated at each stage: the integral values (of the scaling functions), the scaling coefficient values and the graph structure. Our E-LOCAAT is no exception, with the lengths of the neighbouring edges also being updated.

  Let us first update the integral values (associated with the neighbourhood of e_{k_r}) from stage-r to stage-$(r-1)$ by

$$I^{\Gamma}_{s,r-1} = I^{\Gamma}_{s,r} + a^{\Gamma}_{s,r} I^{\Gamma}_{k_r,r}, \quad \text{for } s : e_s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}.$$

Since all prediction weights are positive values, the integral values of the scaling functions will be non-decreasing from stage-r to stage-$(r-1)$.

The coefficient values will be updated as

$$c^{\Gamma}_{s,r-1} = c^{\Gamma}_{s,r} + b^{\Gamma}_{s,r} d^{\Gamma}_{k_r}, \quad \text{for } s : e_s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}. \tag{3.1.17}$$

The values of the update coefficients will be given by taking the minimum norm solution of underdetermined linear system $\sum_{s:e_s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}} b^{\Gamma}_{s,r} I^{\Gamma}_{s,r-1} = I^{\Gamma}_{k_r,r}$, see Jansen et al. (2009). Therefore, the update coefficient will be obtained by

$$b^{\Gamma}_{s,r} = \frac{I^{\Gamma}_{k_r,r} I^{\Gamma}_{s,r-1}}{\sum_{t:e_t \in \mathcal{N}^{\mathcal{E}}_{k_r,r}} \left(I^{\Gamma}_{t,r-1}\right)^2}, \quad \text{for } s : e_s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}.$$

For E-LOCAAT, updating the graph structure by minimal spanning tree, as for vertex-based LOCAAT, might not be a suitable choice, for both theoretical and interpretation reasons, especially since we no longer transform edges into vertices (as for LG-LOCAAT). Instead, we equate the removal of an edge $e_{k_r} = \{v_{i_r}, v_{j_r}\}$ with its two endpoints (vertices) fusing. Therefore, different from the relinkage in LG-LOCAAT, relinkage in E-LOCAAT is a natural part of the update procedure. If we consider that these two vertices fuse to the middle point of the associated metrized edge, then the updated neighbouring lengths can be obtained by

$$\ell_{s,r-1} = \ell_{s,r} + \frac{1}{2}\ell_{k_r,r}, \quad \text{for } s : e_s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}. \tag{3.1.18}$$

Another possible length update is to use the same scheme as for the integral value update, which is given by

$$\ell_{s,r-1} = \ell_{s,r} + a^{\mathcal{E}}_{s,r}\ell_{k_r,r}, \quad \text{for } s : e_s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}. \tag{3.1.19}$$

Both methods ensure that the lengths are non-decreasing from stage-r to stage-$(r-1)$. The one in equation (3.1.18) is more intuitive, while the one in equation (3.1.19) keeps the consistence between lengths and integrals, which leads to a lower computational effort (we only have to store one of them). We will refer to these two different length update procedures as 'unweighted' (equation (3.1.18)) and 'weighted' (equation (3.1.19)) length update, respectively.

- **Iterate:** We reiterate the split-predict-update procedures and obtain the detail coefficients, $\{d^{\Gamma}_{k_m}, ..., d^{\Gamma}_{k_{\tau+1}}\}$, where $\tau \in \mathbb{Z}$ is the number of edges not being removed (stopping time). Note that this set of detail coefficients is defined directly on the edge topology, which allows us to avoid the problems discussed in Section 2.3.4.

- **Inverse:** The inverse E-LOCAAT transform can be carried out by undoing equation (3.1.17) and then undoing equation (3.1.11), as follows,

$$\begin{aligned} c^{\Gamma}_{s,r} &= c^{\Gamma}_{s,r-1} - b^{\Gamma}_{s,r} d^{\Gamma}_{k_r}, \\ c^{\Gamma}_{k_r,r} &= d^{\Gamma}_{k_r} + \sum_{s:e_s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}} a^{\Gamma}_{s,r} c^{\Gamma}_{s,r}. \end{aligned}$$

  Computationally, we store the set of edges as a list, each element of the list consists of two numbers indicating two vertices associated with that edge. For example, the k_r-th element of the list is (i_r, j_r). Then the graph structure update is performed by replacing all scripts j_r with i_r in the list except the k_r-th element (or replacing i_r with j_r, which will not influence the algorithm result). The k_r-th element will be replaced by (j_r, j_r). Although it is not removed through the algorithm, it will not have any influence after stage-r since none of the other elements contain j_r. For E-LOCAAT, a lifting array will be constructed as

$$k_r \quad |\mathcal{N}^{\mathcal{E}}_{k_r,r}| \quad \mathcal{S}^{\mathcal{E}}_r \quad \underline{a}^{\Gamma}_r \quad \underline{b}^{\Gamma}_r \quad i_r \quad j_r \quad \mathrm{sub}_{j_r}(\mathcal{N}^{\mathcal{E}}_{k_r,r}),$$

  where $\mathcal{S}^{\mathcal{E}}_r$ is the set consists of all s such that $v^{\mathcal{E}}_s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}$; $\underline{a}^{\Gamma}_r$, $\underline{b}^{\Gamma}_r$ are sequences of predict and update coefficients; $i_r \in e_{k_r}$ is the vertex for replacing j_r at stage-r as we described in the update part, and j_r is the one being replaced; $\mathrm{sub}_{j_r}(\mathcal{S}^{\mathcal{E}}_r)$ is the subset of $\mathcal{S}^{\mathcal{E}}_r$, which is defined as

$$\mathrm{sub}_{j_r}(\mathcal{S}^{\mathcal{E}}_r) = \{t \,|\, e_t \in \mathcal{N}^{\mathcal{E}}_{k_r,r};\ j_r \in e_t\}.$$

  Thus, for recovering an edge k_r-th edge through the inverse transform, we first turn the k_r-th element of the list (as we discussed in the update part) from (j_r, j_r) back to (i_r, j_r). For all s-th elements, where $s \in \mathrm{sub}_{j_r}(\mathcal{S}^{\mathcal{E}}_r)$, we replace the i_r by the j_r. Then we say both coefficient values and the graph structure have been inverted.

3.2 Biorthogonal Haar E-LOCAAT

In this section, we delve into a variant of the E-LOCAAT framework, which is a unique instance of the edge bases we outlined in Section 3.1.2. We retain the term 'biorthogonal Haar' as coined by Schröder and Sweldens (1995a), this name was chosen as at each stage, the wavelet function takes the form of a Haar-like step function on the metrized edges.

Let us consider the scaling function choices of Section 3.1.2,

$$\varphi^{\Gamma}_{k,m}(\mathbf{p}) = \chi_{\mathbf{P}_k}(\mathbf{p}) \cong \chi_{(0,\ell_k)}(y), \tag{3.2.1}$$

$$\tilde{\varphi}^{\Gamma}_{k,m}(\mathbf{p}) = \ell_k^{-1}\chi_{\mathbf{P}_k}(\mathbf{p}) \cong \ell_k^{-1}\chi_{(0,\ell_k)}(y). \tag{3.2.2}$$

As we previously mentioned, these initial scaling functions are inspired by the construction in Schröder and Sweldens (1995a). Their construction relies on the cascade algorithm, therefore, through their construction, the scaling functions at different levels are *always* assumed to be characteristic functions. This condition guarantees the self-similarity of the scaling functions (and the associated wavelet functions). Now, let us consider this as a constraint for our E-LOCAAT. Firstly, note that our proposed E-LOCAAT construction leads to an MRA of a similar form to the one derived in Jansen et al. (2009), summarised here as an edge case, such that

$$\tilde{\psi}^{\Gamma}_{k_r,r} = \tilde{\varphi}^{\Gamma}_{k_r,r} - \sum_{s\in\mathcal{N}^{\mathcal{E}}_{k_r,r}} a^{\Gamma}_{s,r}\tilde{\varphi}^{\Gamma}_{s,r} \tag{3.2.3}$$

$$\tilde{\varphi}^{\Gamma}_{s,r-1} = \tilde{\varphi}^{\Gamma}_{s,r} + b^{\Gamma}_{s,r}\tilde{\psi}^{\Gamma}_{k_r,r}, \qquad \text{for } s \in \mathcal{N}^{\mathcal{E}}_{k_r,r} \tag{3.2.4}$$

$$\varphi^{\Gamma}_{s,r-1} = \varphi^{\Gamma}_{s,r} + a^{\Gamma}_{s,r}\varphi^{\Gamma}_{k_r,r}, \qquad \text{for } s \in \mathcal{N}^{\mathcal{E}}_{k_r,r} \tag{3.2.5}$$

$$\psi^{\Gamma}_{k_r,r} = \varphi^{\Gamma}_{k_r,r} - \sum_{s\in\mathcal{N}^{\mathcal{E}}_{k_r,r}} b^{\Gamma}_{s,r}\varphi^{\Gamma}_{s,r-1}. \tag{3.2.6}$$

If we plug equation (3.2.3) into equation (3.2.4), then the dual scaling function at stage-r can be rewritten for $s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}$ as

$$\begin{aligned}\tilde{\varphi}^{\Gamma}_{s,r-1} &= \tilde{\varphi}^{\Gamma}_{s,r} + b^{\Gamma}_{s,r}(\tilde{\varphi}^{\Gamma}_{k_r,r} - \sum_{t\in\mathcal{N}^{\mathcal{E}}_{k_r,r}} a^{\Gamma}_{t,r}\tilde{\varphi}^{\Gamma}_{t,r}) \\ &= (1 - a^{\Gamma}_{s,r}b^{\Gamma}_{s,r})\tilde{\varphi}^{\Gamma}_{s,r} + b^{\Gamma}_{s,r}\tilde{\varphi}^{\Gamma}_{k_r,r} - \sum_{t'\in\mathcal{N}^{\mathcal{E}}_{k_r,r}\backslash\{s\}} b^{\Gamma}_{s,r}a^{\Gamma}_{t',r}\tilde{\varphi}^{\Gamma}_{t',r}. \end{aligned} \tag{3.2.7}$$

Here we want the stage-$(r-1)$ dual scaling function to be a characteristic function of the form

$$\tilde{\varphi}^{\Gamma}_{s,r-1} = \ell^{-1}_{s,r-1}\chi_{(0,\ell_{s,r-1})}.$$

One way for equation (3.2.7) to lead to a characteristic function is if we consider letting $a^{\Gamma}_{t',r} = 0$ for all $t' \in \mathcal{N}^{\mathcal{E}}_{k_r,r}\backslash\{s\}$, i.e. using only one neighbour for prediction. Then equation (3.2.7) can be rewritten as

$$\begin{aligned}\tilde{\varphi}^{\Gamma}_{s,r-1} &= (1 - a^{\Gamma}_{s,r}b^{\Gamma}_{s,r})\tilde{\varphi}^{\Gamma}_{s,r} + b^{\Gamma}_{s,r}\tilde{\varphi}^{\Gamma}_{k_r,r} \\ &= (1 - a^{\Gamma}_{s,r}b^{\Gamma}_{s,r})\ell^{-1}_{s,r}\chi_{(0,\ell_{s,r})} + b^{\Gamma}_{s,r}\ell^{-1}_{k_r,r}\chi_{(0,\ell_{k_r,r})}. \end{aligned} \tag{3.2.8}$$

Since e_{k_r} and e_s are neighbours, we can translate the characteristic function part of $\tilde{\varphi}^{\Gamma}_{s,r}$ from $\chi_{(0,\ell_{s,r})}$ to $\chi_{(\ell_{k_r,r},\ell_{k_r,r}+\ell_{s,r})}$. Therefore, if the condition $(1 - a^{\Gamma}_{s,r}b^{\Gamma}_{s,r})\ell^{-1}_{s,r} = b^{\Gamma}_{s,r}\ell^{-1}_{k_r,r}$ is satisfied, it ensures that the right-hand side of equation (3.2.8) can be represented as a constant times a characteristic function on the support $(0, \ell_{k_r,r}+\ell_{s,r})$, since $(0, \ell_{k_r,r}+\ell_{s,r})$ can be represented as $\mathbf{int}\left(\overline{(0,\ell_{k_r,r}) \cup (\ell_{k_r,r}, \ell_{k_r,r}+\ell_{s,r})}\right)$, where $\overline{\mathcal{A}}$ is the closure of a set $\mathcal{A}$, and $\mathbf{int}(\mathcal{A})$ is the interior of the set $\mathcal{A}$. If we keep the length as the measure, and let $\tilde{\varphi}^{\Gamma}_{s,r-1} \propto \chi_{(0,\ell_{s,r-1})}$, then naturally, from stage-r to stage-$(r-1)$, the s-th edge length can be updated by

$$\ell_{s,r-1} = \ell_{k_r,r} + \ell_{s,r}.$$

Based on this, we propose to obtain the update coefficient $b^{\Gamma}_{s,r}$ by

$$\begin{aligned}\ell^{-1}_{s,r-1} &= b^{\Gamma}_{s,r}\ell^{-1}_{k_r,r} \\ \Rightarrow b^{\Gamma}_{s,r} &= \frac{\ell_{k_r,r}}{\ell_{s,r-1}} = \frac{\ell_{k_r,r}}{\ell_{k_r,r}+\ell_{s,r}}. \end{aligned}$$

The prediction weight $a^{\Gamma}_{s,r}$ is then derived to be

$$\begin{aligned}
&\ell^{-1}_{s,r-1} = (1 - a^{\Gamma}_{s,r} b^{\Gamma}_{s,r})\ell^{-1}_{s,r} \\
\Rightarrow &1 - a^{\Gamma}_{s,r} b^{\Gamma}_{s,r} = \frac{\ell_{s,r}}{\ell_{s,r-1}} \\
\Rightarrow &a^{\Gamma}_{s,r} = (1 - \frac{\ell_{s,r}}{\ell_{s,r-1}}) \, (b^{\Gamma}_{s,r})^{-1} \\
&\quad = \frac{\ell_{k_r,r}}{\ell_{s,r-1}} \, (b^{\Gamma}_{s,r})^{-1} \\
&\quad = 1.
\end{aligned}$$

It is easy to see from equation (3.2.5) that $a^{\Gamma}_{s,r} = 1$ guarantees that the s-th primal scaling function is still characteristic function from stage-r to stage-$(r-1)$. Notice that at every stage-r, only one edge from the neighbourhood of e_{k_r} will be used to carry out the prediction for the value of e_{k_r}, with all of the scaling functions at any stage as characteristic functions, while the forms of the wavelet functions look like step functions. Therefore, this proposed construction generates a set of Haar-like biorthogonal wavelets (since the lifting scheme generates biorthogonal bases). Different from the original biorthogonal Haar from Schröder and Sweldens (1995a), our construction (since it relies on recursive lifting construction) does not rely on the cascade algorithm. Recall that performing a prediction step by only one neighbouring element will result in a high bound for the L^2-norm of the prediction weights, which makes the transform less stable (Section 2.3.3). For the overall stability of the transform, the split will choose the edge with the shortest length at each stage-r, such that the condition $b^{\Gamma}_{s,r} \leq \frac{1}{2}$ is always guaranteed. The edge selected for prediction is the one with the shortest length in the set $\mathcal{N}^{\mathcal{E}}_{k_r,r}$, as it is nearest to e_{k_r} regardless of the distance metric employed.

3.3 Simulation Study

In this section, we perform a new simulation study to investigate the newly proposed schemes. The datasets, network generating methods, and noise contamination are all as

described in Section 2.4.1. A table consisting the acronyms of the algorithm is provided in Table 3.1.

Acronyms	
E-Lid-wu	L: lengths as initial integral values (equation (3.1.8)); id: inverse distance prediction (equation (3.1.13)); wu: weighted update for edge lengths (equation (3.1.19))
E-Lid-nwu	L: lengths as initial integral values (equation (3.1.8)); id: inverse distance prediction (equation (3.1.13)); nwu: equally-weighted update for edge lengths (equation (3.1.18))
E-Lil-wu	L: lengths as initial integral values (equation (3.1.8)); il: inverse length prediction (equation (3.1.15)); wu: weighted update for edge lengths (equation (3.1.19))
E-Lil-nwu	L: lengths as initial integral values (equation (3.1.8)); il: inverse length prediction (equation (3.1.15)); nwu: equally-weighted update for edge lengths (equation (3.1.18))
E-Did-wu	D: sequence of ones as initial integral values (equation (3.1.9)); id: inverse distance prediction (equation (3.1.13)); wu: weighted update for edge lengths (equation (3.1.19))
E-Did-nwu	D: sequence of ones as initial integral values (equation (3.1.9)); id: inverse distance prediction (equation (3.1.13)); nwu: equally-weighted update for edge lengths (equation (3.1.18))
E-Dil-wu	D: sequence of ones as initial integral values (equation (3.1.9)); il: inverse length prediction (equation (3.1.15)); wu: weighted update for edge lengths (equation (3.1.19))
E-Dil-nwu	D: sequence of ones as initial integral values (equation (3.1.9)); il: inverse length prediction (equation (3.1.15)); nwu: equally-weighted update for edge lengths (equation (3.1.18))
E-Lnw-wu/nwu	L: lengths as initial integral values (equation (3.1.8)); nw: moving average prediction (equation (3.1.16)). Here we combine wu/nwu since they produce similar results.
E-Dnw-wu/nwu	D: sequence of ones as initial integral values (equation (3.1.9)); nw: moving average prediction (equation (3.1.16)). Here we combine wu/nwu since they produce similar results.
Bio-Haar	Biorthogonal Haar E-LOCAAT

Table 3.1: Acronyms and algorithm descriptions for different parameter choices of E-LOCAAT.

3.3.1 Stability

As already discussed in the previous chapter, the stability of the transform is measured by the condition number of the associated lifting matrix. The condition numbers for the different E-LOCAAT schemes are reported in the following table.

Condition Number	Max	75%	Median	25%	Min
E-Lid-wu	13.0416	11.8697	11.5001	11.1103	10.6264
E-Lid-nwu	12.5082	11.7017	11.4390	11.1276	10.6339
E-Lil-wu	12.7565	12.0485	11.6578	11.3073	10.7940
E-Lil-nwu	12.6003	11.7388	11.5613	11.2099	10.5744
E-Did-wu	11.5659	10.7700	10.5015	10.1513	9.9615
E-Did-nwu	11.4153	**10.6699**	10.3878	10.1631	10.0230
E-Dil-wu	11.9683	11.4165	10.9500	10.5821	10.1335
E-Dil-nwu	11.8750	11.1817	10.8991	10.5558	10.1561
E-Lnw-wu/nwu	12.9201	12.1043	11.5347	11.0867	10.5442
E-Dnw-wu/nwu	**11.1841**	10.7376	**10.3361**	**10.0381**	**9.9513**
Bio-Haar	14.4716	13.2060	12.7001	12.2815	11.4737

Table 3.2: Condition number for E-LOCAAT on a tree structure.

From Table 3.2, we note that unweighted prediction provides a more stable transform than the algorithms involving different parameter choices, its magnitude is similar to the (best) condition number in LG-LOCAAT. There is almost no difference in the stability achieved via the equally weighted and weighted update strategies. Overall, choosing a sequence of ones as the integral values produces a more stable transform. The biorthogonal Haar LOCAAT has higher condition number than any other scheme, even though it only chooses one neighbour at each stage. However, when compared with the performance of LG-LOCAAT, we note that overall E-LOCAAT has lower condition number, as measured by their interquartile range.

3.3.2 Sparsity

The sparsity plots for E-LOCAAT will be constructed as described for LG-LOCAAT (see Section 2.4). Just as for LG-LOCAAT, different method choices do not appear to change the algorithm sparsity property significantly. The biorthogonal Haar E-LOCAAT also

produces reasonable results in terms of data compression. All methods have very similar decay rates, within each function type. The most important factors in achieving good compression properties are the decomposing function's properties, such as smoothness, with the most competitive results being obtained for '`mfc`' and 'g_1'.

3.3.2.1 Sparsity for Pointwise Functions

We can see from Figures 3.1, 3.2, and 3.3 that different choices of initial integral determination, prediction methods, and length update methods will not improve the algorithm's compression ability. The sparsity is more likely to be influenced by the function types. Biorthogonal Haar E-LOCAAT appears to have a slightly better compression ability for 'Blocks' and 'g_1'.

3.3.2.2 Sparsity for Edge Averaging Functions

Similar to the sparsity plots of E-LOCAAT for pointwise functions, there are no significant differences between different choices of initial integral determination, prediction methods, and length update methods for the edge averaging functions by E-LOCAAT, see Figures 3.4, 3.5, and 3.6.

3.3.3 Denoising Performance

3.3.3.1 Denoising Pointwise Functions

Compared with the AMSE, (squared) bias, and variance tables in LG-LOCAAT, the results for E-LOCAAT are more varied, see Tables 3.3, 3.4, and 3.5. In order to ensure a fair comparison, we focus on comparing E-LOCAAT results to those obtained by LG-LOCAAT using path information, as opposed to full coordinate information. Overall, the best results in LG-LOCAAT (those associated with 'LG-Aid', which stands for using average distances as the initial integral values and whose prediction weights are constructed via inverse shortest path distance) are better than the best ones in E-LOCAAT (those associated with 'E-Did', which stands for using a sequence of ones as the initial integral

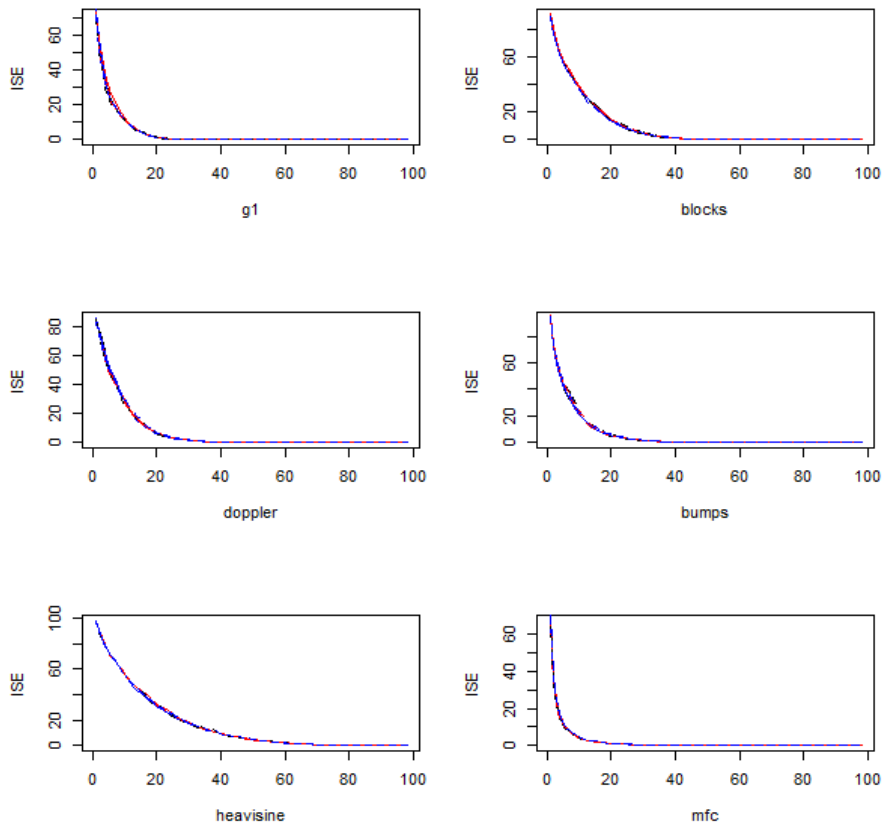

Figure 3.1: Sparsity plots for the test functions used in simulation by the equation (2.4.1). The scheme is based on the unweighted length update. From left to right on *top row*: g_1, blocks; *middle row*: doppler, bumps; *bottom row*: heavisine, maartenfunc. **Black line**: E-Lid-nwu; **red line**: E-Lil-nwu; **blue line**: E-Lnw-wu/nwu; **dashed black line**: E-Did-nwu; **dashed red line**: E-Dil-nwu; **dashed blue line**: E-Dnw-wu/nwu.

values and whose prediction weights are constructed via inverse shortest path distance) in terms of AMSE, except for the biorthogonal Haar E-LOCAAT which significantly surpasses every other method for g_1 and Blocks function. The source of the higher AMSE when compared to LG-LOCAAT, appears to be traced from the variance of the estimator, while the bias results of E-LOCAAT are competitive compared to LG-LOCAAT. When the level of noise contamination is high, E-LOCAAT produces a lower bias compared to LG-LOCAAT. Overall, amongst the E-LOCAAT methods we observe a clear bias-variance tradeoff. Biorthogonal Haar E-LOCAAT beats other schemes in both variance and bias, for g_1 and Blocks functions. However, the biorthogonal Haar LOCAAT

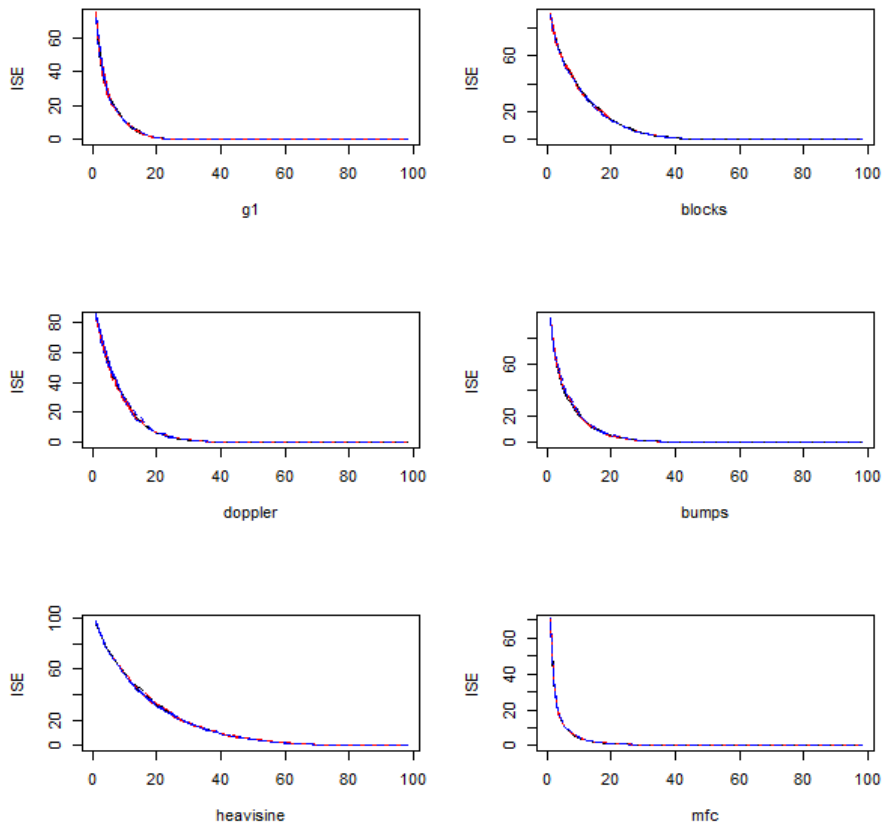

Figure 3.2: Sparsity plots for the test functions used in simulation by the equation (2.4.1). The scheme is based on the weighted length update. From left to right on *top row*: g_1, blocks; *middle row*: doppler, bumps; *bottom row*: heavisine, maartenfunc. **Black line**: E-Lid-nwu; **red line**: E-Lil-nwu; **blue line**: E-Lnw-wu/nwu; **dashed black line**: E-Did-nwu; **dashed red line**: E-Dil-nwu; **dashed blue line**: E-Dnw-wu/nwu.

does not work well for Bumps and `mfc`. If we have the knowledge that the underlying true function is a piecewise function, then biorthogonal Haar will be the optimal choice; otherwise it does not perform as well as other E-LOCAAT methods.

Among the tested methods, 'E-Did-nwu' seems an optimal choice for the remaining functions (except g_1 and Blocks). It is also interesting that 'E-Lil-nwu' always performs better than any other E-LOCAAT algorithms (except when SNR=3) for Heavisine function, in terms of both AMSE and variance. Hence, the 'E-Lil' algorithm might be the optimal choice for high-frequency functions. Overall, as a general rule of thumb, the 'D' methods (choosing a sequence of ones as initial integral values) are preferable.

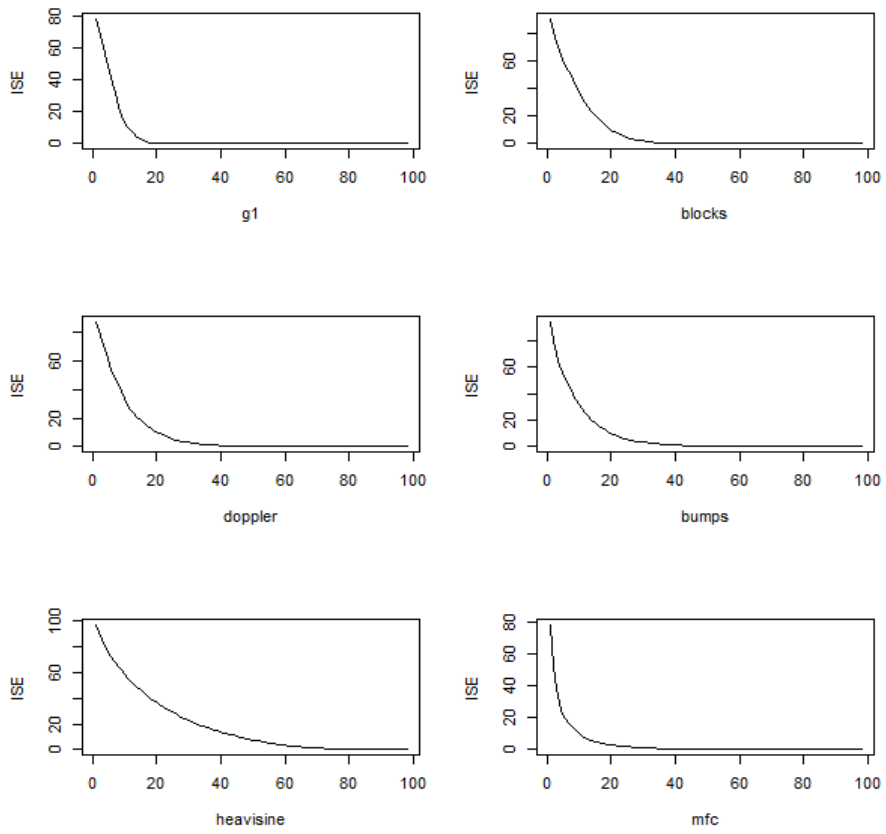

Figure 3.3: Sparsity plots for the test functions used in simulation by the equation (2.4.1). The results are obtained by biorthogonal Haar LOCAAT. From left to right on *top row*: g_1, blocks; *middle row*: doppler, bumps; *bottom row*: heavisine, maartenfunc.

Note that the algorithm performance is consistent through different SNR levels. Similarly to the results obtained by the LG-LOCAAT algorithm, the denoising performance is highly related to the type of underlying function. For the function with smooth patterns almost everywhere, e.g., 'Doppler' and 'Bumps' (see Appendix B), E-LOCAAT is competitive compared with LG-LOCAAT, but for functions with many discontinuities ('Blocks', g_1, and `mfc`), E-LOCAAT is less competitive when compared to LG-LOCAAT. The reason might be that E-LOCAAT uses larger neighbourhood sets through the algorithm. As for LG-LOCAAT, E-LOCAAT also does not perform well when the underlying function has high frequency, e.g., 'Heavisine'.

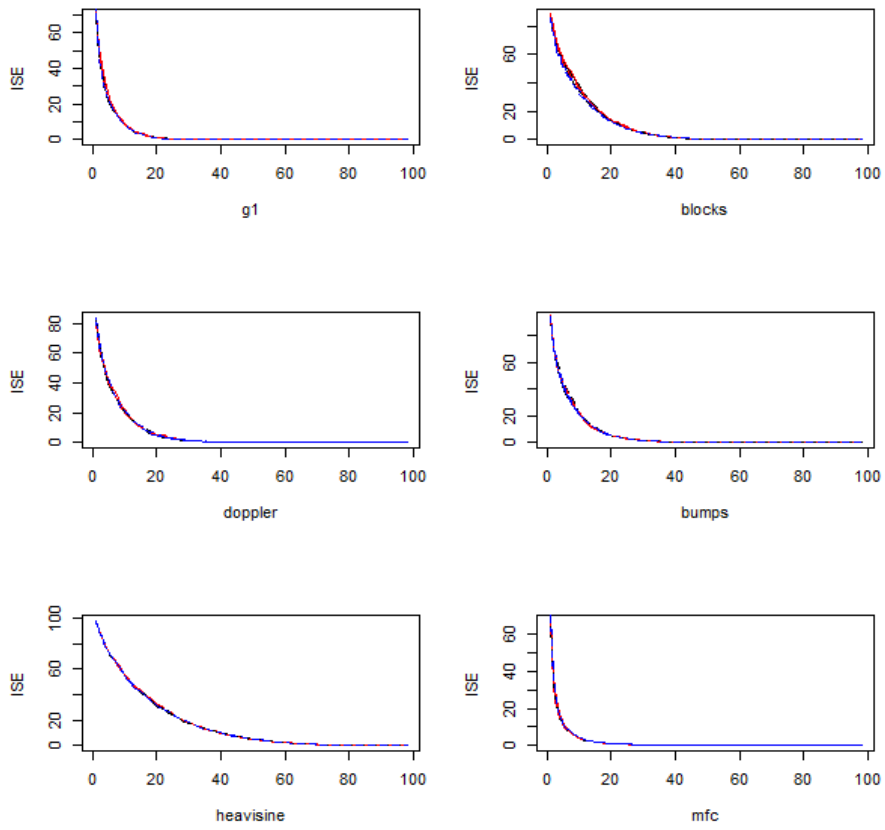

Figure 3.4: Sparsity plots for the test functions used in simulation by the equation (2.4.2). The scheme is based on the unweighted length update. From left to right on *top row*: g_1, blocks; *middle row*: doppler, bumps; *bottom row*: heavisine, maartenfunc. **Black line**: E-Lid-wu/nwu; **red line**: E-Lil-nwu; **blue line**: E-Lnw-nwu; **dashed black line**: E-Did-nwu; **dashed red line**: E-Dil-nwu; **dashed blue line**: E-Dnw-wu/nwu.

If we compare the algorithms with the same choice of the initial integral values and prediction weights, but with different length update paradigm, similar denoising results can be observed, which indicates that there is no significance between these two choices (as in equations (3.1.18) and (3.1.19)). In practice the choice associated with equation (3.1.19) will be preferred since it keeps the construction consistent with the integral value update.

AMSE$\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
			SNR=3			
E-Lid-wu	74 (22)	136 (59)	109 (38)	93 (27)	232 (76)	53 (12)
E-Lid-nwu	73 (23)	137 (59)	109 (38)	93 (27)	234 (77)	53 (12)
E-Lil-wu	73 (23)	134 (56)	108 (37)	94 (27)	232 (78)	53 (12)
E-Lil-nwu	73 (22)	134 (56)	107 (37)	94 (27)	**228 (72)**	53 (12)
E-Did-wu	68 (19)	127 (51)	**98 (33)**	85 (21)	251 (91)	50 (11)
E-Did-nwu	68 (19)	127 (51)	**98 (32)**	**84 (21)**	252 (92)	**49 (11)**
E-Dil-wu	67 (19)	122 (48)	**98 (33)**	86 (22)	251 (92)	50 (11)
E-Dil-nwu	67 (20)	122 (48)	**98 (33)**	86 (22)	252 (93)	50 (11)
E-Lnw-wu/nwu	75 (24)	139 (63)	110 (38)	94 (27)	243 (81)	53 (12)
E-Dnw-wu/nwu	70 (20)	132 (54)	102 (34)	87 (22)	274 (94)	50 (11)
Bio-Haar	**57 (20)**	**87 (33)**	116 (41)	108 (31)	298 (99)	65 (13)
			SNR=5			
E-Lid-wu	27 (10)	50 (24)	44 (17)	44 (15)	166 (54)	26 (6)
E-Lid-nwu	27 (10)	50 (25)	44 (17)	43 (15)	167 (55)	26 (6)
E-Lil-wu	27 (10)	50 (24)	44 (16)	43 (15)	166 (57)	26 (5)
E-Lil-nwu	27 (10)	50 (23)	43 (16)	43 (15)	**163 (52)**	25 (5)
E-Did-wu	24 (8)	49 (22)	**39 (13)**	**40 (13)**	191 (83)	**24 (5)**
E-Did-nwu	25 (8)	49 (22)	**39 (13)**	**40 (13)**	191 (83)	**24 (5)**
E-Dil-wu	24 (8)	47 (20)	**39 (14)**	41 (13)	189 (84)	25 (5)
E-Dil-nwu	24 (8)	47 (20)	**39 (14)**	41 (13)	189 (85)	**24 (5)**
E-Lnw-wu/nwu	27 (11)	50 (25)	45 (18)	45 (17)	179 (59)	26 (6)
E-Dnw-wu/nwu	25 (8)	50 (23)	40 (14)	42 (13)	218 (87)	25 (5)
Bio-Haar	**18 (5)**	**29 (10)**	49 (20)	52 (18)	232 (82)	33 (8)
			SNR=7			
E-Lid-wu	12 (5)	26 (13)	24 (10)	27 (10)	147 (46)	17 (4)
E-Lid-nwu	12 (4)	26 (13)	24 (10)	27 (10)	149 (47)	17 (4)
E-Lil-wu	13 (4)	26 (13)	24 (9)	26 (10)	146 (50)	**16 (4)**
E-Lil-nwu	12 (4)	26 (12)	23 (9)	26 (10)	**144 (46)**	**16 (4)**
E-Did-wu	12 (4)	25 (12)	**21 (7)**	**25 (9)**	171 (80)	**16 (4)**
E-Did-nwu	12 (4)	25 (12)	**21 (7)**	**25 (9)**	172 (80)	**16 (4)**
E-Dil-wu	12 (3)	25 (11)	22 (8)	**25 (9)**	171 (82)	**16 (4)**
E-Dil-nwu	12 (3)	25 (11)	22 (8)	**25 (9)**	170 (82)	**16 (4)**
E-Lnw-wu/nwu	12 (4)	25 (13)	25 (10)	28 (11)	161 (51)	17 (4)
E-Dnw-wu/nwu	12 (4)	27 (13)	22 (8)	26 (10)	200 (84)	**16 (4)**
Bio-Haar	**9 (2)**	**14 (5)**	28 (12)	33 (13)	212 (77)	21 (6)

Table 3.3: The AMSE table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1).

Var$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
E-Lid-wu	51	72	**61**	**54**	70	**40**
E-Lid-nwu	51	71	**61**	**54**	70	**40**
E-Lil-wu	51	71	**61**	55	70	41
E-Lil-nwu	51	71	**61**	**54**	**69**	**40**
E-Did-wu	53	87	72	62	116	41
E-Did-nwu	53	87	71	61	115	**40**
E-Dil-wu	53	85	72	63	116	42
E-Dil-nwu	53	84	71	62	115	41
E-Lnw-wu/nwu	51	72	**61**	**54**	**69**	**40**
E-Dnw-wu/nwu	54	91	74	63	123	41
Bio-Haar	**49**	**61**	63	58	73	43
SNR=5						
E-Lid-wu	20	27	**24**	23	**29**	**16**
E-Lid-nwu	20	27	**24**	**22**	**29**	**16**
E-Lil-wu	20	27	**24**	23	**29**	**16**
E-Lil-nwu	20	27	**24**	23	**29**	**16**
E-Did-wu	21	35	29	29	78	18
E-Did-nwu	21	35	29	28	77	18
E-Dil-wu	21	34	30	29	78	19
E-Dil-nwu	21	34	30	29	77	18
E-Lnw-wu/nwu	21	27	**24**	23	**29**	**16**
E-Dnw-wu/nwu	22	37	31	30	86	18
Bio-Haar	**16**	**22**	25	24	31	19
SNR=7						
E-Lid-wu	10	14	**13**	**13**	**16**	**9**
E-Lid-nwu	10	14	**13**	**13**	**16**	**9**
E-Lil-wu	10	14	**13**	**13**	17	**9**
E-Lil-nwu	10	14	**13**	**13**	**16**	**9**
E-Did-wu	11	19	16	17	66	11
E-Did-nwu	11	19	16	17	65	11
E-Dil-wu	11	18	16	17	66	11
E-Dil-nwu	11	18	16	17	65	11
E-Lnw-wu/nwu	10	14	**13**	**13**	**16**	**9**
E-Dnw-wu/nwu	11	20	17	18	75	11
Bio-Haar	**8**	**11**	**13**	14	19	11

Table 3.4: The variance table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1).

bias^2 $\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
E-Lid-wu	23	65	48	40	163	13
E-Lid-nwu	23	65	48	39	164	13
E-Lil-wu	22	63	47	39	162	12
E-Lil-nwu	22	63	46	40	159	12
E-Did-wu	15	40	**26**	**23**	**135**	**9**
E-Did-nwu	15	40	**26**	**23**	137	**9**
E-Dil-wu	14	37	**26**	**23**	**135**	**9**
E-Dil-nwu	14	38	27	24	136	**9**
E-Lnw-wu/nwu	24	67	49	40	174	13
E-Dnw-wu/nwu	15	41	27	24	151	**9**
Bio-Haar	**8**	**26**	53	50	225	22
SNR=5						
E-Lid-wu	7	23	20	21	138	9
E-Lid-nwu	7	23	20	21	139	9
E-Lil-wu	7	23	19	21	136	9
E-Lil-nwu	7	23	19	21	134	9
E-Did-wu	3	13	**9**	**12**	113	**6**
E-Did-nwu	3	14	**9**	**12**	114	**6**
E-Dil-wu	3	13	**9**	**12**	**112**	**6**
E-Dil-nwu	3	13	10	**12**	**112**	**6**
E-Lnw-wu/nwu	7	23	21	23	151	10
E-Dnw-wu/nwu	3	14	**9**	**12**	132	**6**
Bio-Haar	**2**	**8**	24	28	201	14
SNR=7						
E-Lid-wu	2	12	11	14	131	7
E-Lid-nwu	2	12	11	14	132	7
E-Lil-wu	3	12	11	14	130	7
E-Lil-nwu	3	12	11	13	128	7
E-Did-wu	**1**	7	**5**	**8**	106	**5**
E-Did-nwu	**1**	7	**5**	**8**	107	**5**
E-Dil-wu	**1**	6	**5**	**8**	**105**	**5**
E-Dil-nwu	**1**	7	**5**	**8**	106	**5**
E-Lnw-wu/nwu	2	12	12	15	145	8
E-Dnw-wu/nwu	**1**	7	**5**	**8**	125	**5**
Bio-Haar	**1**	**3**	15	20	192	11

Table 3.5: The squared bias table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1).

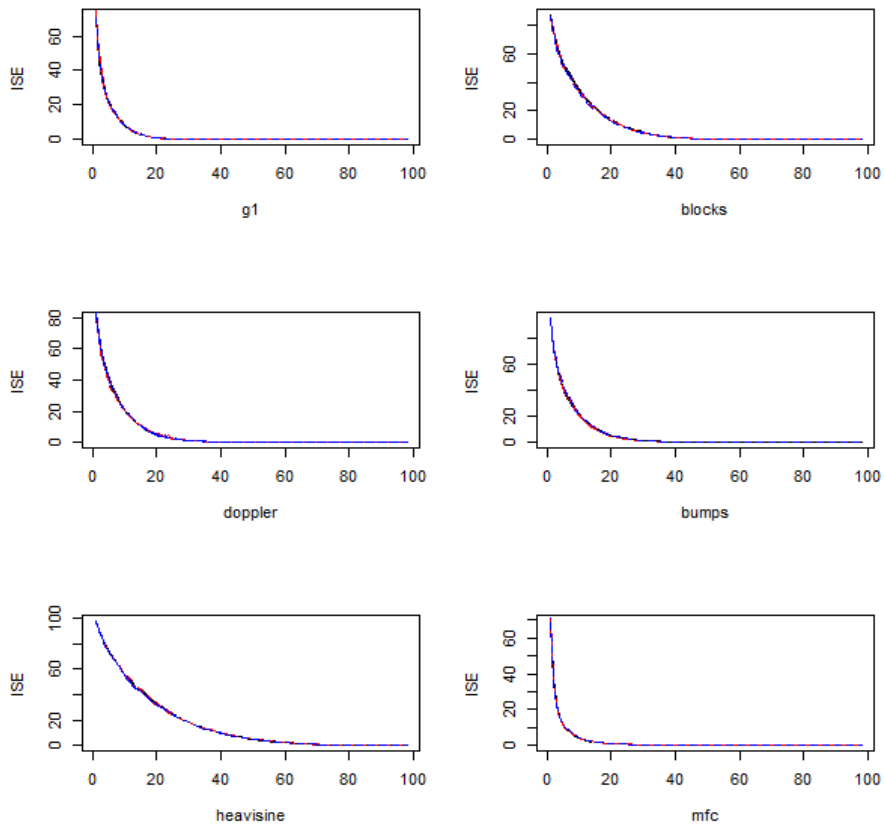

Figure 3.5: Sparsity plots for the test functions used in simulation by the equation (2.4.2). The scheme is based on the unweighted length update. From left to right on *top row*: g_1, blocks; *middle row*: doppler, bumps; *bottom row*: heavisine, maartenfunc. **Black line**: E-Lid-nwu; **red line**: E-Lil-nwu; **blue line**: E-Lnw-wu/nwu; **dashed black line**: E-Did-nwu; **dashed red line**: E-Dil-nwu; **dashed blue line**: E-Dnw-wu/nwu.

3.3.3.2 Denoising Edge Averaging Function

The results for edge averaging functions detailed in Tables 3.6, 3.7, and 3.8 are more competitive when compared to LG-LOCAAT. Here, the 'D' methods are still preferred (especially 'E-Did-nwu'), since they have better AMSE results and lower bias. For Blocks and g_1 functions, biorthogonal Haar LOCAAT still outperforms the other proposals.

It is desirable that the 'Did-nwu' method is consistently better performing, regardless of the data generating process, while for the line graph algorithm the best performing method switched from 'Aid-p' to 'Did-p' when moving from pointwise to edge averaging.

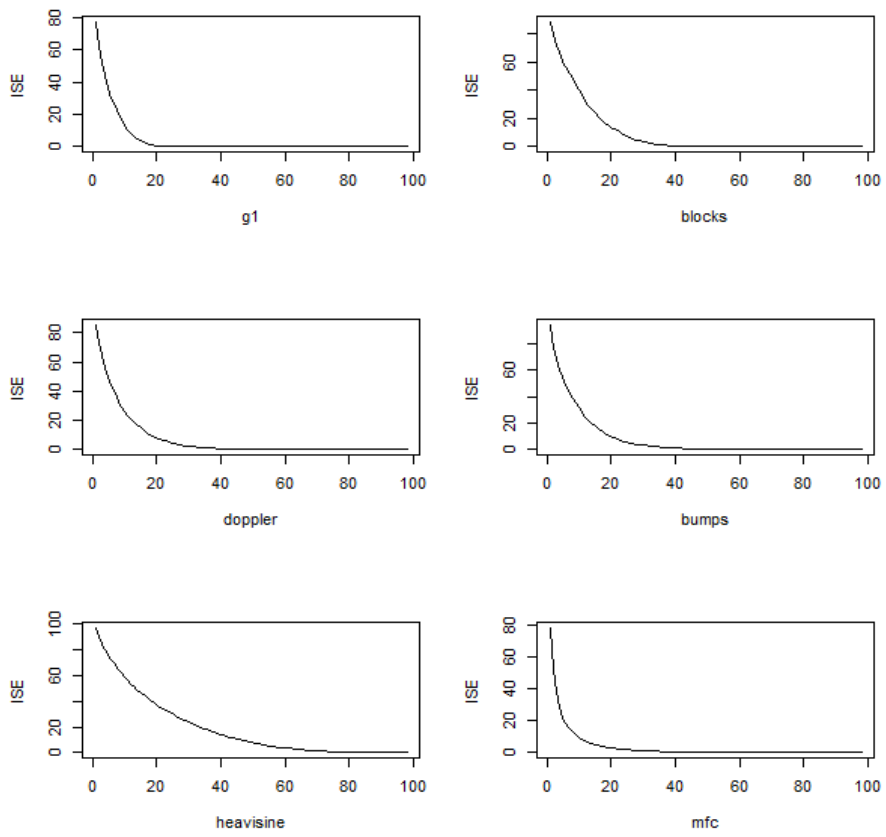

Figure 3.6: Sparsity plots for the test functions used in simulation by the equation (2.4.2). The results are obtained by biorthogonal Haar LOCAAT. From left to right on *top row*: g_1, blocks; *middle row*: doppler, bumps; *bottom row*: heavisine, maartenfunc.

Similarly, with increasing SNR, the least competitive results are for 'Heavisine' while all others are comparable.

Empirical Computational Cost

The standard running time for E-LOCAAT algorithms is 00:10:30, with 90% of the algorithms completing within 00:12:00. Compared to the computational cost of LG-LOCAAT (see Section 2.5.3), E-LOCAAT affords a much faster computation.

AMSE$\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
E-Lid-wu	70 (20)	148 (59)	106 (34)	94 (27)	251 (83)	52 (12)
E-Lid-nwu	69 (20)	147 (58)	105 (33)	93 (27)	252 (83)	52 (12)
E-Lil-wu	69 (20)	147 (57)	105 (33)	94 (27)	251 (85)	52 (12)
E-Lil-nwu	69 (20)	147 (58)	104 (33)	94 (27)	246 (79)	52 (12)
E-Did-wu	**62 (17)**	126 (45)	**92 (28)**	84 (21)	**243 (84)**	49 (11)
E-Did-nwu	**62 (17)**	125 (45)	**92 (28)**	**83 (21)**	**243 (85)**	**48 (11)**
E-Dil-wu	**62 (17)**	124 (45)	93 (28)	85 (21)	245 (86)	50 (11)
E-Dil-nwu	**62 (17)**	123 (45)	93 (28)	85 (21)	245 (86)	49 (11)
E-Lnw-wu/nwu	70 (20)	150 (60)	107 (34)	94 (27)	263 (88)	52 (12)
E-Dnw-wu/nwu	64 (17)	132 (48)	96 (29)	86 (21)	267 (90)	49 (11)
Bio-Haar	66 (21)	**110 (47)**	114 (38)	109 (31)	317 (103)	64 (13)
SNR=5						
E-Lid-wu	31 (11)	61 (28)	46 (18)	44 (16)	187 (63)	25 (5)
E-Lid-nwu	31 (11)	62 (29)	46 (17)	44 (16)	189 (63)	25 (5)
E-Lil-wu	31 (11)	61 (28)	46 (17)	44 (15)	187 (66)	25 (5)
E-Lil-nwu	30 (11)	61 (28)	45 (17)	44 (15)	**184 (61)**	25 (5)
E-Did-wu	26 (8)	53 (22)	**40 (14)**	**40 (13)**	187 (77)	**24 (5)**
E-Did-nwu	26 (8)	53 (23)	**40 (14)**	**40 (12)**	188 (77)	**24 (5)**
E-Dil-wu	26 (8)	51 (22)	**40 (14)**	**40 (13)**	188 (79)	**24 (5)**
E-Dil-nwu	26 (8)	51 (21)	**40 (14)**	**40 (12)**	188 (79)	**24 (5)**
E-Lnw-wu/nwu	31 (12)	63 (31)	47 (18)	45 (16)	199 (66)	26 (5)
E-Dnw-wu/nwu	27 (9)	57 (25)	41 (15)	42 (13)	214 (82)	**24 (5)**
Bio-Haar	**23 (8)**	**38 (14)**	51 (21)	53 (18)	252 (90)	33 (8)
SNR=7						
E-Lid-wu	16 (7)	34 (17)	26 (11)	27 (10)	169 (56)	16 (4)
E-Lid-nwu	16 (7)	34 (18)	26 (10)	27 (10)	170 (56)	16 (4)
E-Lil-wu	16 (6)	34 (17)	25 (10)	27 (10)	168 (60)	16 (4)
E-Lil-nwu	16 (6)	34 (17)	25 (10)	26 (10)	**166 (55)**	16 (4)
E-Did-wu	14 (5)	30 (14)	**22 (8)**	**24 (9)**	170 (74)	**15 (4)**
E-Did-nwu	13 (5)	30 (14)	**22 (8)**	**24 (9)**	170 (74)	**15 (3)**
E-Dil-wu	13 (4)	28 (13)	23 (8)	25 (9)	172 (76)	**15 (4)**
E-Dil-nwu	13 (4)	28 (13)	23 (8)	25 (9)	171 (77)	**15 (3)**
E-Lnw-wu/nwu	16 (7)	35 (19)	26 (11)	28 (11)	181 (58)	17 (4)
E-Dnw-wu/nwu	14 (5)	32 (15)	23 (9)	26 (9)	198 (79)	16 (4)
Bio-Haar	**11 (3)**	**20 (7)**	29 (12)	34 (13)	232 (85)	22 (6)

Table 3.6: The AMSE table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2).

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
E-Lid-wu	47	70	58	54	70	**40**
E-Lid-nwu	47	69	**57**	**53**	70	**40**
E-Lil-wu	47	69	58	54	70	**40**
E-Lil-nwu	47	69	**57**	54	**69**	**40**
E-Did-wu	50	84	66	61	113	41
E-Did-nwu	49	84	65	61	113	**40**
E-Dil-wu	50	84	67	62	114	42
E-Dil-nwu	50	83	66	62	113	41
E-Lnw-wu/nwu	**46**	69	**57**	**53**	70	**40**
E-Dnw-wu/nwu	50	88	68	63	121	41
Bio-Haar	49	**66**	60	58	73	43
SNR=5						
E-Lid-wu	20	28	**24**	23	29	**16**
E-Lid-nwu	20	28	**24**	23	29	**16**
E-Lil-wu	20	28	**24**	23	29	**16**
E-Lil-nwu	20	28	**24**	23	**28**	**16**
E-Did-wu	21	37	29	28	76	18
E-Did-nwu	21	37	29	28	75	18
E-Dil-wu	21	36	30	29	77	18
E-Dil-nwu	21	36	30	29	76	18
E-Lnw-wu/nwu	20	28	**24**	**22**	29	**16**
E-Dnw-wu/nwu	22	40	31	29	84	18
Bio-Haar	**18**	**24**	25	24	32	18
SNR=7						
E-Lid-wu	11	15	**13**	**13**	**16**	**9**
E-Lid-nwu	11	15	**13**	**13**	**16**	**9**
E-Lil-wu	11	15	**13**	**13**	17	**9**
E-Lil-nwu	11	15	**13**	**13**	**16**	**9**
E-Did-wu	12	21	17	17	65	11
E-Did-nwu	12	21	17	17	64	11
E-Dil-wu	11	20	17	17	65	11
E-Dil-nwu	11	20	17	17	64	11
E-Lnw-wu/nwu	11	15	**13**	**13**	**16**	**9**
E-Dnw-wu/nwu	12	22	17	18	74	11
Bio-Haar	**9**	**12**	**13**	14	19	11

Table 3.7: The variance table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2).

bias$^2 \times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
E-Lid-wu	23	78	48	40	182	12
E-Lid-nwu	23	78	48	40	183	12
E-Lil-wu	23	78	47	40	181	12
E-Lil-nwu	23	78	46	40	177	11
E-Did-wu	**13**	41	**27**	**22**	**129**	**8**
E-Did-nwu	**13**	42	**27**	23	131	**8**
E-Dil-wu	**13**	**40**	**27**	23	130	**8**
E-Dil-nwu	**13**	**40**	**27**	23	132	**8**
E-Lnw-wu/nwu	24	80	50	41	193	12
E-Dnw-wu/nwu	**13**	44	28	23	146	**8**
Bio-Haar	16	47	54	51	244	22
SNR=5						
E-Lid-wu	11	33	22	21	159	9
E-Lid-nwu	11	34	22	21	160	9
E-Lil-wu	11	33	22	21	158	9
E-Lil-nwu	11	33	21	21	155	9
E-Did-wu	5	16	**10**	**11**	**111**	**6**
E-Did-nwu	5	16	**10**	12	113	**6**
E-Dil-wu	**4**	15	11	12	**111**	**6**
E-Dil-nwu	5	15	11	12	112	**6**
E-Lnw-wu/nwu	11	35	23	23	171	9
E-Dnw-wu/nwu	5	18	11	12	130	**6**
Bio-Haar	5	**14**	26	29	220	15
SNR=7						
E-Lid-wu	5	19	13	14	153	7
E-Lid-nwu	5	19	13	14	154	7
E-Lil-wu	5	19	13	14	152	7
E-Lil-nwu	5	19	13	14	150	7
E-Did-wu	**2**	9	**6**	**7**	**105**	**4**
E-Did-nwu	**2**	9	**6**	**7**	106	5
E-Dil-wu	**2**	8	**6**	**7**	106	**4**
E-Dil-nwu	**2**	8	**6**	**7**	107	**4**
E-Lnw-wu/nwu	5	20	14	15	165	8
E-Dnw-wu/nwu	**2**	10	**6**	8	125	5
Bio-Haar	**2**	**7**	16	20	213	11

Table 3.8: The squared bias table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2).

3.4 Remarks

We proposed two algorithms aimed at building wavelet bases for the graph edge set on the original graph domain through two different approaches, namely interpolation point bases and edge bases. Although algorithmically they are similar (see Section 3.1.3), they do have different interpretations. Thus, in this section, we will provide some remarks on these aspects (including the biorthogonal Haar E-LOCAAT).

For the interpolating-point bases, we start with an idea similar to the LG-LOCAAT, which is to find a set of points that represent all edges (referred to as 'new vertices' in LG-LOCAAT). The advantage of using vertex representation is the convenience of function representation, and the possibility for constructing 'smoother' wavelets (by means of higher vanishing moments), see Schröder and Sweldens (1995a), which can be an interesting future task.

Although we did not put a lot of emphasis on the analysis of the function spaces of the true functions (there is not much literature that has discussed the functional analysis for edge-based functions), understanding the behaviour (e.g., smoothness and variation) of the functions in edge-space is of interest as a future task, which allows derivation for the detail coefficient decay rate, Riesz bounds, and estimation error. For achieving these theoretical results, biorthogonal Haar type multiresolution analysis might be the most amenable approach, see Schröder and Sweldens (1995a). In addition, the self-similarity of the scaling/wavelet functions of the biorthogonal Haar E-LOCAAT is more convenient than general LG-LOCAAT and E-LOCAAT (both use inverse distance prediction, which is a linear interpolating scheme), which raises a potential contribution to some applications, such as long-memory estimation (Knight et al.; 2017). Zhang et al. (2008) shows that the biorthogonal Haar framework has potential for removing Poisson noise.

Compared to LG-LOCAAT, although the constructions of E-LOCAAT scaling functions are very different, the filters follow similar constructions. For example, for E-LOCAAT (except biorthogonal Haar E-LOCAAT), the construction of prediction filters $\{a^{\Gamma}\}$ is based on the distance measure between different edges, and the construction of

update filters $\{b^{\Gamma}\}$ relies on the minimal norm solution, which guarantees the stability of the transform.

Although for both LG-LOCAAT and E-LOCAAT, the distance can be chosen as the path distance between two neighbouring edges, they result in very different algorithms and interpretations. Once we perform the line graph transform with a chosen metric, the distances between each pair of new vertices are then determined. However, for E-LOCAAT, the distance measure between neighbouring edges depends on the edges at the corresponding stage. Thus, the distance measure will be changed (for the edges in $\mathcal{N}^{\mathcal{E}}_{k_r,r}$) after each stage-$r$. Note that the distance between any two edges is non-decreasing through the algorithm, hence, E-LOCAAT has a better ability to prevent the compression issue from the 'update effect' as discussed in Section 2.3.3.

The relinkage of E-LOCAAT normally generates a less sparse structure compared with LG-LOCAAT, which means E-LOCAAT tends to use a larger size of neighbourhood compared with LG-LOCAAT. Thus, E-LOCAAT displays a better stability performance (see Section 2.3.3 for the discussion of the influence of the neighbourhood size), but it might introduce more scale-mixing at the same time. This might be the underlying reason for the denoising performance of LG-LOCAAT to be better than most proposed E-LOCAAT algorithms.

For the function representation aspects, E-LOCAAT has more potential than LG-LOCAAT, since the (scaling/wavelet) function representations can always be obtained in the original domain (which is not the case for LG-LOCAAT, see Section 2.3.4).

Chapter 4

Laplacian-LOCAAT Construction

In Chapters 2 and 3 we proposed two algorithms that provide multiscale decompositions for data collected from the network edges. One difficulty we recognised is to determine the most suitable distance measure among the set of edges, a problem which is not as straightforward as measuring the distance among nodes. Thus, an algebraic approach, rather than a functional one, would be desirable for addressing this limitation.

Hence, in this chapter, we introduce a new LOCAAT-based algorithm for dealing with edge data, which we refer to as Laplacian-LOCAAT, since it is derived from the graph edge-Laplacian. The chapter is organised as follows. We first introduce some essential background knowledge on graph Laplacian and its edge variant. This is followed by a section introducing the connection between graph Laplacians and the lifting steps, and some recent topics of topological data analysis related to the high-order Laplacians. Then, similar to the previous two chapters, a detailed discussion of the proposed algorithm and the simulation results will be given.

4.1 Graph Laplacian

The graph Laplacian was originally studied in spectral graph theory, which focused on the analysis of eigenvalues and eigenvectors of the associated graph matrices. Bollobás (1998) gives a description of spectral graph theory, details can also be found in Chung

(1996). Brouwer and Haemers (2011) gives some advanced topics in relation to algebra and topology. For introductory material on spectral graph theory, the reader can refer to Spielman (2007, 2012). In the following two subsections, we will review literature on two different types of graph Laplacian, corresponding to graphs whose edges are with and without orientation, while mainly focusing on the relevant aspects of spectral graph theory which contribute to our work. The reader can refer to the literature mentioned above for further details. In this section, we preserve the notation from previous chapters, i.e., the graph G, the number of the vertices (n), and the number of edges (m).

4.1.1 Laplacian Construction Using an Oriented Incidence Matrix

Let us recall the adjacency matrix $A(G)$ in equation (1.4.1) and the incidence matrix $B(G)$ in equation (1.4.2) discussed in Section 1.4.2. These matrices play an important role in spectral graph theory, and in particular in the construction of the graph Laplacian. The matrix $B(G)$ introduced in equation (1.4.2) is the non-oriented incidence matrix for an undirected graph G, see Diestel (2005). In a lot of literature, for example, Godsil and Royle (2001) and Diestel (2005), an *orientation* is given to the incidence matrix, by assigning '1' and '-1' to the corresponding elements (initial vertex and terminal vertex) of the incidence matrix. For an undirected graph, the assignation could be arbitrary, see Zelazo et al. (2007) and Zelazo and Mesbahi (2010). Here we further let the edge $e_k = \{v_i, v_j\}$, with v_i be the initial vertex if $i < j$, and v_j be the terminal vertex. Then, similarly to the notion used in Zelazo et al. (2007), the (i,k)-th element of oriented incidence matrix $\overrightarrow{B}(G)$ is defined as

$$\left[\overrightarrow{B}(G)\right]_{ik} = \begin{cases} 1, & \text{if } v_i \text{ is the initial vertex of edge } e_k; \\ -1, & \text{if } v_i \text{ is the terminal vertex of edge } e_k; \\ 0, & \text{otherwise.} \end{cases} \tag{4.1.1}$$

Unlike the non-oriented incidence matrix B, each element of the oriented incidence matrix $\overrightarrow{B}$ takes a value from the set $\{0, 1, -1\}$ rather than $\{0, 1\}$. One might notice

that the orientation can give 'directions' to an undirected graph, but oriented graphs and directed graphs are *different*, see Diestel (2005). An edge could have two directions in a directed graph but this is not allowed in oriented graph. Although the oriented incidence matrix gives a unique direction for each edge, some consequent constructs will not be changed, for example, when obtaining the graph Laplacian; see Godsil and Royle (2001), Zelazo et al. (2007) or Zelazo and Mesbahi (2010). We will discuss this later in this section. A relevant matrix related to the graph structure is the **degree matrix**, which can be written as a $n \times n$-dimensional diagonal matrix for the graph G, defined as

$$D(G) = \begin{pmatrix} \deg(v_1) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \deg(v_n) \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{n} A_{1j} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \sum_{j=1}^{n} A_{nj} \end{pmatrix}, \tag{4.1.2}$$

where $\deg(v_i)$ is the degree of the i-th vertex defined to be simply the number of the neighbours of the i-th vertex (or the number of edges containing this vertex).

4.1.1.1 Non-weighted Version

The core notion for the study of spectral graph theory is the **graph Laplacian matrix** (see Chung (1996)), which is a $n \times n$ matrix given by

$$\mathcal{L}^{\mathcal{V}}(G) = D(G) - A(G). \tag{4.1.3}$$

Here we use the superscript $\mathcal{V}$ to represent that this Laplacian is a matrix established upon the vertex set (since we will discuss the so-called edge Laplacian later). Alternatively, we can express it in an element-wise form, such that the (i, j)-th element corresponding to the vertices v_i and v_j is

$$\left[\mathcal{L}^{\mathcal{V}}(G)\right]_{ij} = \begin{cases} -1, & \text{if } i \neq j \text{ and } \{v_i, v_j\} \in \mathcal{E}; \\ \deg(v_i), & \text{if } i = j; \\ 0, & \text{otherwise.} \end{cases} \tag{4.1.4}$$

Moreover, the graph Laplacian matrix has a natural relation with the oriented incidence matrix $\overrightarrow{B}$, see Merris (1994) or Zelazo and Mesbahi (2010). Namely, the Laplacian is

obtained by

$$\mathcal{L}^{\mathcal{V}}(G) = \overrightarrow{B}(G)\overrightarrow{B}(G)^T, \tag{4.1.5}$$

where each element B_{ik}, as we mentioned in equation (4.1.1), is taking values from the set $\{0, 1, -1\}$. In this case, we could see that $\left[\mathcal{L}^{\mathcal{V}}(G)\right]_{ij} = \sum_{t=1}^{m} \overrightarrow{B}_{it}\overrightarrow{B}_{jt}$. Since $\overrightarrow{B}_{it}, \overrightarrow{B}_{jt} \in \{0, 1, -1\}$ and $\overrightarrow{B}_{it} = \overrightarrow{B}_{jt}$ will occur only in two cases, which are $\overrightarrow{B}_{it} = \overrightarrow{B}_{jt} = 0$ or $i = j$, then we can easily see that $\left[\mathcal{L}^{\mathcal{V}}(G)\right]_{ij} \leq 0$ if $i \neq j$. The graph Laplacian is a symmetric matrix, thus, all of its eigenvalues are real and non-negative, see Chung (1997). From equations (4.1.2) and (4.1.3), we can see that the graph Laplacian matrix satisfies that $\sum_{j=1}^{n} \left[\mathcal{L}^{\mathcal{V}}(G)\right]_{ij} = 0$ for any $i \in \{1, ..., n\}$. We will see later that this property is desirable for our work.

Recall that the incidence matrix brings up the connection between vertices and edges, since the rows and columns represent vertices and edges, respectively. Consequently, an edge variant of the graph Laplacian was introduced by Zelazo et al. (2007) and Zelazo and Mesbahi (2010), which will be our main tool in this chapter. From now on, we will distinguish the terminology 'graph Laplacian' and 'edge Laplacian' as the Laplacian matrix for vertex set and edge set, respectively. Starting from equation (4.1.5) with the oriented $n \times m$ incidence matrix B in equation (4.1.1), the edge variant of the graph Laplacian, briefly, the $m \times m$ **edge Laplacian matrix**, is obtained by simply commuting the product terms in equation (4.1.5), such that

$$\mathcal{L}^{\mathcal{E}}(G) = \overrightarrow{B}(G)^T\overrightarrow{B}(G). \tag{4.1.6}$$

It is easy to see that the edge Laplacian $\mathcal{L}^{\mathcal{E}}(G)$ is a real-valued $m \times m$ matrix symmetric matrix with the following two properties (Zelazo et al.; 2007):

- For the Laplacian matrices $\mathcal{L}^{\mathcal{V}}$ and $\mathcal{L}^{\mathcal{E}}$ of the same graph structure G, the non-zero eigenvalues are the same.
- The non-zero eigenvalues of $\mathcal{L}^{\mathcal{V}}$ and $\mathcal{L}^{\mathcal{E}}$ are equal to the square of the non-zero singular values of $\overrightarrow{B}$.

These two properties guarantee that the edge Laplacian can preserve measures of the graph structure, such as connectivity (see Godsil and Royle (2001)) and the trace of the matrices. The edge Laplacian can be written in an elementwise form for the k-th and l-th edges, e_k and e_l, such that

$$\left[\mathcal{L}^{\mathcal{E}}(G)\right]_{kl} = \sum_{p=1}^{n} \overrightarrow{B}_{pk} \overrightarrow{B}_{pl}. \tag{4.1.7}$$

Here $\{\overrightarrow{B}_{pk}\}_{p=1}^{n}$ are all the entries of k-th column, namely, describing the relationship the set of vertices $\{v_p\}_{p=1}^{n}$ has with edge e_k. Recall that the column of the oriented incidence matrix $\overrightarrow{B}$ only contains two nonzero elements that indicate the two vertices that belong to the associated edge (see equation (4.1.1)) since we only consider simple graphs. Therefore, if $k = l$, we have

$$\begin{aligned}\left[\mathcal{L}^{\mathcal{E}}(G)\right]_{kk} &= \sum_{p=1}^{n} (\overrightarrow{B}_{pk})^2 \\ &= 1^2 + (-1)^2 \\ &= 2.\end{aligned}$$

For $k \neq l$, we have $\left[\mathcal{L}^{\mathcal{E}}(G)\right]_{kl} \in \{1, -1, 0\}$ because distinct edges cannot share exactly the same set of vertices. Furthermore, for a pair of neighbouring edges e_k and e_l, we say that they form a path if their common vertex is initial for one of e_k or e_l, and terminal for the other. Thus the edge Laplacian can also be written entry-wise, namely

$$\left[\mathcal{L}^{\mathcal{E}}(G)\right]_{kl} = \begin{cases} 1, & \text{if } e_k \text{ and } e_l \text{ do not form a path but have a common vertex;} \\ -1, & \text{if } e_k \text{ and } e_l \text{ form a path;} \\ 2, & \text{if } k = l; \\ 0, & \text{otherwise (no vertex in common).} \end{cases} \tag{4.1.8}$$

Hence the diagonal of the edge Laplacian is a vector of dimension m, populated with 2's. A crucial algebraic property of $\mathcal{L}^{\mathcal{E}}$ is the connection with the line graph $\mathbf{LG}(G)$, see Godsil and Royle (2001) and Zelazo and Mesbahi (2010), which is

$$A(\mathbf{LG}(G)) = \left|\mathcal{L}^{\mathcal{E}}(G) - 2\mathbf{I}_m\right|, \tag{4.1.9}$$

here $|\cdot|$ means taking the absolute value for all entries in the matrix, and $\mathbf{I}_m$ is the $m \times m$ identity matrix.

4.1.1.2 Weighted Version

Both the graph Laplacian and edge Laplacian defined above are derived from degree matrices, adjacency matrices, and incidence matrices, which contain only graph connectivity information. However, a weighted Laplacian version is needed in order to show the geometric information of a weighted graph. The weighted Laplacian was used in many previous works from signal processing or wavelet communities, see for example, Coifman and Maggioni (2006), Hammond et al. (2011), Hammond et al. (2013) and Chen and Liu (2017). In our case, a weighted edge Laplacian will be preferred in order to show the geometric information.

Similarly to the established literature, we begin with the vertex case. In general, the graph Laplacian can also be embedded with a weight function $\omega : \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$ if further information is available. We let this weight function to be symmetric, thus, $\omega(v_i, v_j) = \omega(v_j, v_i), \forall i, j$, and assign it the role of an edge weight function as described in Section 1.4.3. In this case, we define the Laplacian for a weighted graph, or the weighted Laplacian (see Godsil and Royle (2001)), as

$$\left[\mathcal{L}^{\mathcal{V},\omega}(G)\right]_{ij} = \begin{cases} -\omega(v_i, v_j), & \text{if } \{v_i, v_j\} \in \mathcal{E}; \\ \sum_{s \in \{1,\dots,n\}\setminus\{i\}} \omega(v_i, v_s), & \text{if } i = j; \\ 0, & \text{otherwise.} \end{cases} \tag{4.1.10}$$

Sometimes we may write ω_{ij} instead of $\omega(v_i, v_j)$ for simplification. We could also construct the weighted Laplacian matrix by generalising equation (4.1.5) (Godsil and Royle; 2001), such that

$$\mathcal{L}^{\mathcal{V},\omega}(G) = \overrightarrow{B}(G)\, W(G)\, \overrightarrow{B}(G)^T, \tag{4.1.11}$$

where $W(G)$ (or put simply, W) is a $m \times m$ diagonal weight matrix such that

$$W_{kk'} = \begin{cases} \omega_{ij}, & \text{if } k = k' \text{ and } e_k = \{v_i, v_j\}; \\ 0, & \text{if } k \neq k'. \end{cases} \tag{4.1.12}$$

This weight matrix could be designed flexibly, for example, the weights can be treated as an intrinsic quantity related to the network, see Kolaczyk and Csárdi (2014). For the river network motivating example, the construction of the weights using the streamflow data (Park et al.; 2022) can be fitted as a weight matrix. In general, the weight matrix W is not necessarily diagonal. Here, we assume that W is diagonal and that each weight is positive with the k-th entry of the diagonal defined as the inverse length of the k-th edge, $\frac{1}{\ell_k}$, which can be further handled both theoretically and computationally. Thus, we define its square root as

$$\left[W^{1/2}(G)\right]_{kk'} = \sqrt{W_{kk'}} = \begin{cases} \sqrt{\omega_{ij}}, & \text{if } k = k' \text{ and } e_k = \{v_i, v_j\}; \\ 0, & \text{if } k \neq k'. \end{cases}$$

Then the weighted Laplacian in equation (4.1.11) could be rewritten as

$$\begin{aligned} \mathcal{L}^{\mathcal{V},\omega}(G) &= \overrightarrow{B} W \overrightarrow{B}^T \\ &= \overrightarrow{B} W^{1/2} \left(W^{1/2}\right)^T \overrightarrow{B}^T \\ &= \left(\overrightarrow{B} W^{1/2}\right) \left(\overrightarrow{B} W^{1/2}\right)^T \\ &= \overrightarrow{B}^{\omega} \overrightarrow{B}^{\omega^T}, \end{aligned} \tag{4.1.13}$$

where

$$\begin{aligned} \left[\overrightarrow{B}^{\omega}\right]_{ik} :&= \left[\overrightarrow{B} W^{1/2}\right]_{ik} \\ &= \sum_{q=1}^{m} \overrightarrow{B}_{iq} \left[W^{1/2}\right]_{qk} \\ &= \overrightarrow{B}_{ik} \sqrt{W_{kk}} \end{aligned}$$

is the weighted incidence matrix, of the same dimension $(n \times m)$ as $\overrightarrow{B}$. Thus we rewrite the general version of equation (4.1.1) as

$$\left[\overrightarrow{B}^{\omega}\right]_{ik} = \begin{cases} \sqrt{\omega_{ij}}, & \text{if } v_i \text{ is the initial vertex of edge } e_k \text{ and } e_k = \{v_i, v_j\}; \\ -\sqrt{\omega_{ij}}, & \text{if } v_i \text{ is the terminal vertex of edge } e_k \text{ and } e_k = \{v_i, v_j\}; \\ 0, & \text{otherwise.} \end{cases} \tag{4.1.14}$$

The properties of the unweighted oriented incidence matrix $\overrightarrow{B}$ (defined as in equation (4.1.1)) are well studied in some literature, see e.g., Godsil and Royle (2001) for algebraic properties, Zelazo et al. (2007) and Zelazo and Mesbahi (2010) used the edge Laplacian to analyse the agreement protocol of multi-agent systems, Barbarossa et al. (2018) treated the edge Laplacian as a key to describe features of the high-order simplicial complexes, which is the core of the topological data analysis, Schaub and Segarra (2018) gave a framework of smoothing functions defined on edges by edge Laplacian and pointed out the difference between edge Laplacian and Laplacian for line graphs. Overall, the weighted edge Laplacian has received much less attention in the literature. In our context, the weighted edge Laplacian will play an important role since it allows us to understand higher-order interactions in real-life complex systems, see Schaub et al. (2021).

As we can see in equation (4.1.13), the weighted graph Laplacian $\mathcal{L}^{\mathcal{V},\omega}$ is determined by the weighted incidence matrix B_ω, and similarly, the weights could be introduced into edge case by plugging equation (4.1.13) into equation (4.1.6), such that

$$\mathcal{L}^{\mathcal{E},\omega}(G) = \overrightarrow{B}^{\omega^T}\overrightarrow{B}^{\omega}. \tag{4.1.15}$$

Then similar to equation (4.1.7), each entry of the edge Laplacian can be obtained by

$$\left[\mathcal{L}^{\mathcal{E},\omega}(G)\right]_{kl} = \sum_{p=1}^{n}\left[\overrightarrow{B}^{\omega}\right]_{pk}\left[\overrightarrow{B}^{\omega}\right]_{pl}, \tag{4.1.16}$$

The weighted edge Laplacian could also be presented similarly to equation (4.1.8):

$$\left[\mathcal{L}^{\mathcal{E},\omega}(G)\right]_{kl} = \begin{cases} \sqrt{W_{kk}W_{ll}}, & \text{if } e_k \text{ and } e_l \text{ do not form a path but have a common vertex;} \\ -\sqrt{W_{kk}W_{ll}}, & \text{if } e_k \text{ and } e_l \text{ form a path;} \\ W_{kk}+W_{ll}=2W_{kk}, & \text{if } k=l; \\ 0, & \text{otherwise.} \end{cases} \tag{4.1.17}$$

We can see that the orientation does have an influence on the sign of the non-zero non-diagonal components, but some of the properties are still preserved by construction, for example, symmetry.

4.1.2 Laplacian Construction Using a Non-oriented Incidence Matrix

In the previous section we discussed the matrices related to graph structures, mainly focusing on graph Laplacian and edge Laplacian, and their weighted versions. The construction for Laplacian matrices heavily relies on the incidence matrix with an orientation (4.1.1). For real data cases, such as the streamflow data used in Park et al. (2022), the graph can be modelled by the Laplacian with orientation since each stream segment has a natural unique direction. However, having a natural orientation does not apply to all real-life problems. For instance, the features of the social network data from Zachary (1977) and the traffic network data from Deri and Moura (2015) cannot be captured by giving an orientation to the incidence matrix. Since both social data and traffic data are generally bidirectional, it is also of interest to also consider a Laplacian construction that uses a non-oriented incidence matrix.

4.1.2.1 Non-weighted Version

In this section we mainly focus on constructing the (edge) Laplacian matrices in terms of the non-oriented incidence matrix (see equation (1.4.2)). Recall the matrix B is the non-oriented incidence matrix given by

$$B_{ik} = \begin{cases} 1, & \text{if } v_i \in e_k \text{ and } e_k = \{v_i, v_j\}; \\ 0, & \text{otherwise,} \end{cases} \tag{4.1.18}$$

and define the graph and edge-Laplacian as

$$\mathcal{Q}^{\mathcal{V}}(G) = B(G)B(G)^T, \tag{4.1.19}$$

$$\mathcal{Q}^{\mathcal{E}}(G) = B(G)^T B(G). \tag{4.1.20}$$

Therefore the (i, j)-th element of $\mathcal{Q}^{\mathcal{V}}$ will be

$$\left[\mathcal{Q}^{\mathcal{V}}\right]_{ij} = \begin{cases} 1, & \text{if } i \neq j \text{ and } \{v_i, v_j\} \in \mathcal{E}; \\ \deg(v_i), & \text{if } i = j; \\ 0, & \text{otherwise,} \end{cases} = [D + A]_{ij},$$

This version of Laplacian is referred to as the 'signless Laplacian' in the literature, see for example, Godsil and Royle (2001) and Cvetković et al. (2007). Since there is no distinction between the initial vertex and the terminal vertex for non-oriented incidence matrix, then the (k,l)-th element of $\mathcal{Q}^{\mathcal{E}}$ would be

$$\left[\mathcal{Q}^{\mathcal{E}}(G)\right]_{kl} = \begin{cases} 1, & \text{if } e_k \text{ and } e_l \text{ share one common vertex and } k \neq l; \\ 2, & \text{if } k = l; \\ 0, & \text{otherwise.} \end{cases}$$

4.1.2.2 Weighted Version

As in Section 4.1.1, the weight matrix (4.1.12) can be plugged into the graph Laplacian to give rise to the weighted version

$$\begin{aligned} \mathcal{Q}^{\mathcal{V},\omega} &= BWB^T \\ &= (BW^{1/2})(BW^{1/2})^T \\ &= B^{\omega}(B^{\omega})^T. \end{aligned} \tag{4.1.21}$$

As in equation (4.1.14), here we have

$$[B^{\omega}]_{ik} = \begin{cases} \sqrt{\omega_{ij}}, & \text{if } v_i \in e_k \text{ and } e_k = \{v_i, v_j\}; \\ 0, & \text{otherwise.} \end{cases} \tag{4.1.22}$$

By commuting these two matrices in equation (4.1.21), the signless weighted edge Laplacian then could be obtained by

$$\mathcal{Q}^{\mathcal{E},\omega} = (B^{\omega})^T B^{\omega}, \tag{4.1.23}$$

or equivalently,

$$\left[\mathcal{Q}^{\mathcal{E},\omega}\right]_{kl} = \begin{cases} \sqrt{W_{kk}W_{ll}}, & \text{if } e_k \text{ and } e_l \text{ share one common vertex and } k \neq l; \\ W_{kk} + W_{ll} = 2W_{kk}, & \text{if } k = l; \\ 0, & \text{otherwise.} \end{cases} \tag{4.1.24}$$

Compared to the signed edge Laplacian (equation (4.1.17)), we can see that the signless one (equation (4.1.24)) contains no negative elements. The signless Laplacian has some potential advantages for spectral graph theory, see Cvetković et al. (2007). Nevertheless, Section 4.1.2 is just for completeness, and the Laplacian generated by an oriented incidence matrix will be used in our proposal since the oriented version is a better fit for our framework.

4.1.3 Remarks

4.1.3.1 A Natural Connection between the Laplacian and LOCAAT

Here we briefly discuss the connection between the Laplacian (with the oriented incidence matrix) and the LOCAAT. Suppose we have a function defined on the vertex set of a graph $G = (\mathcal{V}, \mathcal{E}, \omega)$, let us denote it by $g^{\mathcal{V}} : \mathcal{V} \to \mathbb{R}$. Let us denote the vector form of $g^{\mathcal{V}}$ as $\underline{g}^{\mathcal{V}} = (g_1^{\mathcal{V}}, ..., g_n^{\mathcal{V}})^T$, where $g_i^{\mathcal{V}}$ denotes the value of the function at vertex v_i. Then we can show that the i-th element of the vector $\mathcal{L}^{\mathcal{V},\omega}\,\underline{g}^{\mathcal{V}}$ is

$$\begin{aligned}
\left[\mathcal{L}^{\mathcal{V},\omega}\,\underline{g}^{\mathcal{V}}\right]_i &= \sum_{j=1}^{n} \left[\mathcal{L}^{\mathcal{V},\omega}\right]_{ij} g_j^{\mathcal{V}} \\
&= \left[\mathcal{L}^{\mathcal{V},\omega}\right]_{ii} g_i^{\mathcal{V}} + \sum_{\substack{j\in\{1,\dots,n\} \\ j\neq i}} \left[\mathcal{L}^{\mathcal{V},\omega}\right]_{ij} g_j^{\mathcal{V}} \\
&= \sum_{j\in\mathcal{N}_i^{\mathcal{V}}} \omega_{ij} g_i^{\mathcal{V}} + \sum_{j\in\mathcal{N}_i^{\mathcal{V}}} -\omega_{ij} g_j^{\mathcal{V}} \qquad (4.1.25) \\
&= \sum_{j\in\mathcal{N}_i^{\mathcal{V}}} \omega_{ij} \left(g_i^{\mathcal{V}} - g_j^{\mathcal{V}}\right), \qquad (4.1.26)
\end{aligned}$$

where $\mathcal{N}_i^{\mathcal{V}} = \{j \mid \left[\mathcal{L}^{\mathcal{V},\omega}\right]_{ij} \neq 0 \text{ and } j \neq i\}$ is the vertex-neighbourhood index set which denotes the non-zero components on the i-th row of the graph Laplacian, which can be considered as the neighbourhood. Equation (4.1.26) has been discussed in many works but not precisely connected to lifting, the reader can refer to Chung (1996) or Spielman (2007, 2012) for more details. As a special case, for the unweighted graph Laplacian $\mathcal{L}^{\mathcal{V}}$, we have $\left[\mathcal{L}^{\mathcal{V}}\underline{g}^{\mathcal{V}}\right]_i = \sum_{j\in\mathcal{N}_i^{\mathcal{V}}} \left(g_i^{\mathcal{V}} - g_j^{\mathcal{N}}\right)$. Since the Laplacian satisfies that $\forall i$, $\sum_{j=1}^{n} \left[\mathcal{L}^{\mathcal{V}}\right]_{ij} = 0$ and $\left[\mathcal{L}^{\mathcal{V}}\right]_{ij} \leq 0$ if $i \neq j$, then we can see that each row (or column,

due to its symmetry) has a similar role to that of a dual wavelet function, which we recall has the property that $\int_{v\in\mathcal{V}} \tilde{\psi}_i(v)dv = 0$. Hence, along with the unit energy condition $\int_{v\in\mathcal{V}} \tilde{\varphi}_{i,n}(v)dv = 1$, where $\tilde{\varphi}_{i,n}$ is the i-th initial scaling function, the Laplacian $\mathcal{L}^{\mathcal{V},\omega}$ can be considered as a matrix that provides possible filter constructions (every row/column). Moreover, if we normalise equation (4.1.25) as

$$\left[\mathcal{L}^{\mathcal{V},\omega}\, \underline{g}^{\mathcal{V}}\right]_i \Big/ \Big(\sum_{j\in\mathcal{N}_i^{\mathcal{V}}} \omega_{ij}\Big) = g_i^{\mathcal{V}} - \sum_{s\in\mathcal{N}_i^{\mathcal{V}}} \frac{\omega_{is}}{\sum_{j\in\mathcal{N}_i^{\mathcal{V}}} \omega_{ij}} g_s^{\mathcal{V}}, \tag{4.1.27}$$

and (as proposed in Section 4.1), let ω_{ij} be the inverse distance between vertex v_i and vertex v_j, then we note that equation (4.1.27) is of the same form as the prediction step in the LOCAAT transform of Jansen et al. (2009). Then the Laplacian can be considered as a generalised version for the prediction filter design, which gives the weights explicitly.

4.1.3.2 Generalisation for higher-order networks

Recall that one of the most difficult (but important) parts of constructing a multiscale method for edge data is to find appropriate inter-edge distance measures (or inter-edge weights), see Chapters 2 and 3. In both LG-LOCAAT and E-LOCAAT, we used the 'distance' between interpolating points, since the distance measure is natural in this context. However, we might consider whether there is an alternative approach to introducing the edge-related geometric information via the inter-vertex distance measure. The Hodge Laplacian along with the simplicial complexes could be a suitable tool to consider. Simplicial complexes provide a representation of topological space and have been used as a tool by the topological signal processing community, see Robinson (2014). Recently, it has been introduced for some potential applications for data analysis, Kook and Lee (2018) and Devriendt (2022) analysed the effective resistance problem on networks based on simplicial complexes; Schaub et al. (2020) constructed random walk matrices based on simplicial complexes with Hodge theory; Schaub et al. (2021), Battiloro et al. (2023), Bick et al. (2023) and Sardellitti and Barbarossa (2024) discussed signal processing techniques based on simplicial complexes. Besides these works, Lim (2020) gives an introduction of the Hodge theory in linear algebra language, Grady and Polimeni (2010)

gives a framework for defining calculus and functions on discrete structures based upon simplicial complexes and Hodge theory. In our work, we will only introduce the details which will contribute to our framework.

Suppose we have a weighted graph $G = (\mathcal{V}, \mathcal{E}, \omega)$, where $\mathcal{V} = \{v_1, ..., v_n\}$, $\mathcal{E} = \{e_1, ..., e_m\}$, and ω is a weight function that assigns positive values to edges. Recall that an (undirected) edge is an unordered pair of two adjacent vertices. We can naturally consider edges as a 'higher-order structure' compared to vertices. Then intuitively, we can define another higher-order structure, which is referred to as faces in graph theory, see Bondy and Murty (2008), Godsil and Royle (2001) and Diestel (2005). In a nutshell, an abstract face set $\mathcal{T}$ can be defined as Lim (2020) and Bick et al. (2023),

$$\{v_i, v_j, v_k\} \in \mathcal{T} \quad \text{if and only if } \{v_i, v_j\}, \{v_i, v_k\}, \{v_j, v_k\} \in \mathcal{E}. \tag{4.1.28}$$

The unordered set $\{v_i, v_j, v_k\}$ is then called a face. Typically, faces are also referred to as 2-cells, which can represent discrete domains; see Grady and Polimeni (2010). Similarly, the edges and vertices are 1-cells and 0-cells, respectively. Higher-order cells can also be constructed using the above strategy. However, we will not discuss details for any higher-order structure in our work, the reader can refer to Lim (2020) or Bick et al. (2023) for more details. Let $\mathcal{C}_p$ denote the p-order cells, for example, $\mathcal{C}_0 = \mathcal{V}$, $\mathcal{C}_1 = \mathcal{E}$, and $\mathcal{C}_2 = \mathcal{T}$. Then the associated p-order simplicial complex is defined as $\mathfrak{C}^p = \bigcup_{q=0}^{p} \mathcal{C}_q$. For any $(p-1)$- and p-cells $\mathcal{C}_{p-1}$ and $\mathcal{C}_p$, where $p \in \mathbb{Z}^+$, an incidence matrix $\overrightarrow{B}^p$ can be constructed with an orientation. In this chapter, we will set subscripts as the order instead of the vertex or edge set to describe the cell chain clearly. For example, $\overrightarrow{B}^1$ is the oriented incidence matrix $\overrightarrow{B}$ as we described in Section 4.1, whose rows represent vertices and columns represent edges. Similarly, $\overrightarrow{B}^2$ is the oriented incidence matrix whose rows represent edges and columns represent faces. Here we skip the details of the orientation of faces since it will not affect our framework. The incidence matrices play an important role in mapping information between different order structures, for example,

Chung (1997) consider $\overrightarrow{B}^1$ and $\overrightarrow{B}^{1^T}$ as two operators such that

$$\mathcal{E} = \mathcal{C}_1 \underset{\overrightarrow{B}^{1^T}}{\overset{\overrightarrow{B}^1}{\rightleftarrows}} \mathcal{C}_0 = \mathcal{V}\,.$$

Only in this section, we denote $\cdot^p$ as the (weight, incidence or Laplacian) matrix associated with p-order cells, for some $p \in \mathbb{Z}^+$; and let $(\cdot)^q$ as the q-th power of a matrix. One way for obtaining the interaction among a certain simplicial complex $\mathfrak{C}^p$ is to construct the Hodge Laplacian, see Grady and Polimeni (2010), Lim (2020), and Bick et al. (2023), such that

$$\mathbf{L}^p = W^{p+1}\overrightarrow{B}^{p^T}(W^p)^{-1}\overrightarrow{B}^p + \overrightarrow{B}^{p+1}W^{p+2}\overrightarrow{B}^{p+1^T}(W^{p+1})^{-1},$$

where $\overrightarrow{B}^p$ and $\overrightarrow{B}^{p+1}$ are incidence matrices and W^p, W^{p+1}, and W^{p+2} are diagonal weight matrices. Now let us consider the case that $p = 0$, and we let $\overrightarrow{B}^0 = \mathbf{0}$, see Chung (1997) and Bick et al. (2023). Then the general form for the vertex Laplacian becomes

$$\mathbf{L}^0 = \overrightarrow{B}^1 W^2 \overrightarrow{B}^{1^T}(W^1)^{-1}.$$

Thus, the general weighted (vertex) graph Laplacian $\mathcal{L}^{\mathcal{V}}$ can be obtained by setting $W^1 = \mathbf{I}$, and $W^2 = W$ as the edge weight matrix (see Section 4.1). Now, consider the case where $p = 1$, we have $\overrightarrow{B}^2 = \mathbf{0}$, since there is no face in a tree structure. Then, in this case, we can obtain

$$\begin{aligned}\mathbf{L}^1 &= W^2\overrightarrow{B}^{1^T}(W^1)^{-1}\overrightarrow{B}^1 \\ &= W\overrightarrow{B}^{1^T}\overrightarrow{B}^1.\end{aligned}$$

Notice that this matrix is not symmetric, a similarity transform $(W^2)^{-1/2}\mathbf{L}^1(W^2)^{1/2}$ can be performed to normalise it, see Grady and Polimeni (2010) and Bick et al. (2023). Hence, we have

$$\begin{aligned}(W^2)^{-1/2}\mathbf{L}^1(W^2)^{1/2} &= (W^2)^{-1/2}W^2\overrightarrow{B}^{1^T}\overrightarrow{B}^1(W^2)^{1/2} \\ &= (W^2)^{1/2}\overrightarrow{B}^{1^T}\overrightarrow{B}^1(W^2)^{1/2} \\ &= \mathcal{L}^{\mathcal{E}},\end{aligned}$$

which gives us the weighted edge Laplacian introduced in previous Section 4.1.1.2. This indicates that the inter-edge interaction information (weights) can be well-described by the corresponding entries of the edge Laplacian.

4.2 Proposed Laplacian-LOCAAT Framework

As described in Chapters 2 and 3, we assume we have a weighted graph $G = (\mathcal{V}, \mathcal{E}, \omega)$ with $|\mathcal{V}| = n$ and $|\mathcal{E}| = m$, and we are interested in the construction of a basis on which to represent the function $g^{\mathcal{E}} : \mathcal{E} \longrightarrow \mathbb{R}$, where $\{g_k^{\mathcal{E}}\}_{k=1}^m$ is the set of observations on graph edge set $\{e_k\}_{k=1}^m$. In this section, we propose two algorithms, associated with the edge Laplacian and the line graph Laplacian. We will next address the component of the algorithm pertaining to the edge Laplacian.

4.2.1 Proposed LOCAAT via the Edge Laplacian

In our proposed edge Laplacian-LOCAAT framework, we follow the steps of E-LOCAAT, but the edge Laplacian will be used for the construction of prediction filters, as shown next. The algorithm works on the original graph edge domain, so the initial scaling functions can be set as described in Sections 3.1.1 and 3.1.2.

Split Step

Since the initial stage-m scaling functions are defined as in Sections 3.1.1 and 3.1.2, integral values associated to each edge e_k are

$$I_{k,m}^{\Gamma,\text{edge}} = \ell_k,$$
$$I_{k,m}^{\Gamma,\text{Delta}} = 1,$$

where ℓ_k is the initial length of the k-th edge. The choice of the removal edge at stage-r is based on the minimum integral value. Similar to previous chapters, the neighbouring integrals will be updated at each stage during the update step.

Predict Step

Let us assume that edge e_{k_r} was chosen for removal, hence we aim to predict its corresponding function value. The prediction step is such that for stage-r, we ensure the prediction weights associated with k-th edge at stage-r fulfill the condition

$$\sum_{s=1}^{m} a_{s,r}^{\mathcal{E}} = 1. \tag{4.2.1}$$

This condition guarantees that the wavelets satisfy the admissibility condition and the transform is stable. Recall that the $m \times m$ edge Laplacian matrix can be obtained by $\mathcal{L}^{\mathcal{E},\omega}(G) = \overrightarrow{B}^{\omega^T}(G)\overrightarrow{B}^{\omega}(G)$. Note that the non-zero off-diagonal entries of the matrix $\mathcal{L}^{\mathcal{E},\omega}(G)$ show the interactions (in terms of weights) among neighbouring edges (every pair of edges that share only one common vertex), thus, these non-zero values can be naturally considered as the prediction weights. We skip the notation G, and denote the edge Laplacian matrix corresponding to the graph at stage-r as $\mathcal{L}_r^{\mathcal{E},\omega}$. We further denote using the stage-r Laplacian $\mathcal{N}_{k_r,r}^{\mathcal{E}} = \{e_j \mid \left[\mathcal{L}_r^{\mathcal{E},\omega}\right]_{k_r,j} \neq 0 \text{ and } j \neq k_r\}$ as the neighbourhood of the edge e_{k_r} at stage-r. Notice that for the edge Laplacian with an orientation, the off-diagonal components of k_r-th row (or column) are not necessarily sharing the same sign. Stability issues will arise if we predict using both positive and negative prediction weights. For example, negative prediction weights will have an impact on the update of the integral by decreasing it as opposed to increasing it, see Nunes et al. (2006), and the integral may even become zero or negative. This may have a large influence on the compression ability of the algorithm because of the 'update effect' discussed in Section 2.3.3. In order to avoid such problems, we propose to use the prediction weight $a_{s,r}^{\mathcal{E}}$ as

$$a_{s,r}^{\mathcal{E}} = \frac{\left|\left[\mathcal{L}_r^{\mathcal{E},\omega}\right]_{k_r,s}\right|}{\sum_{t:e_t \in \mathcal{N}_{k_r,r}^{\mathcal{E}}} \left|\left[\mathcal{L}_r^{\mathcal{E},\omega}\right]_{k_r,t}\right|}. \tag{4.2.2}$$

Specifically, for the initial stage-m, defining $W^m := W$, the prediction weights can be written as

$$\begin{aligned} a^{\mathcal{E}}_{s,m} &= \frac{\left|\left[\mathcal{L}^{\mathcal{E},\omega}_m\right]_{k_m,s}\right|}{\sum_{t:e_t\in\mathcal{N}^{\mathcal{E}}_{k_m,m}}\left|\left[\mathcal{L}^{\mathcal{E},\omega}_m\right]_{k_m,t}\right|} \\ &= \frac{\sqrt{[W_m]_{k_mk_m}[W_m]_{ss}}}{\sum_{t:e_t\in\mathcal{N}^{\mathcal{E}}_{k_m,m}}\sqrt{[W_m]_{k_mk_m}[W_m]_{tt}}} \\ &= \frac{\sqrt{[W_m]_{ss}}}{\sum_{t:e_t\in\mathcal{N}^{\mathcal{E}}_{k_m,m}}\sqrt{[W_m]_{tt}}}. \end{aligned}$$

Note that if we let the entries of the weight matrix to be inverse distance, where $[W_m]_{kk} = 1/\ell_k$, then as a result, we have $a^{\mathcal{E}}_{s,m} = \frac{\sqrt{1/\ell_s}}{\sum_{t:e_t\in\mathcal{N}^{\mathcal{E}}_{k_m,m}}\sqrt{1/\ell_t}}$, which is the normalised square root of the inverse length. However, this does not hold at every stage, since either the weight matrix or the edge Laplacian matrix will be updated throughout the algorithm (see below). Then the detail coefficient associated to the removed edge ℓ_{k_r} can be obtained as in E-LOCAAT, such that

$$d^{\mathcal{E}}_{k_r} = c^{\mathcal{E}}_{k_r,r} - \sum_{s:\, e_s\in\mathcal{N}^{\mathcal{E}}_{k_r,r}} a^{\mathcal{E}}_{s,r}c^{\mathcal{E}}_{s,r},$$

where $c^{\mathcal{E}}_{k,r}$ is the scaling coefficient of the k-th edge at stage-r, and we set $c^{\mathcal{E}}_{k,m} := g^{\mathcal{E}}_k$ initially.

Update Step

1. **Update for Function and Integral Values**

   The integral values of the primal scaling function will be first updated. When progressing from stage-r to stage-$(r-1)$, we have as in Chapters 2 and 3,

   $$I^{\mathcal{E}}_{s,r-1} = I^{\mathcal{E}}_{s,r} + a^{\mathcal{E}}_{s,r}I^{\mathcal{E}}_{k_r,r}, \quad \forall s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}.$$

   The neighbouring coefficients are also updated by means of

   $$c^{\mathcal{E}}_{s,r-1} = c^{\mathcal{E}}_{s,r} + b^{\mathcal{E}}_{s,r}d_{k_r}, \ \ \forall s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}, \tag{4.2.3}$$

where the coefficients $\left\{b^{\mathcal{E}}_{s,r}\right\}$ are still obtained by taking the minimum norm solution of the update condition mentioned by Jansen et al. (2004, 2009), such that

$$b^{\mathcal{E}}_{s,r} = \frac{I^{\mathcal{E}}_{k_r,r}\, I^{\mathcal{E}}_{s,r-1}}{\sum_{t:e_t\in\mathcal{N}^{\mathcal{E}}_{k_r,r}} \left(I^{\mathcal{E}}_{t,r-1}\right)^2}.$$

2. **Update for the Edge Laplacian and Relinkage via Schur Complement**

Recall that in E-LOCAAT, we update the lengths of the neighbouring edges of the removed edge, and this is in turn will have an influence on the prediction in the next stages. In our construction here, the update step has to consider *the update of the edge Laplacian matrix*, since the edge Laplacian directly contributes to the prediction filter construction (see equation (4.2.2)). We next discuss the possibility to update the Laplacian matrix by taking the Schur complement. This linear algebra tool has been studied along with the Laplacian for electrical circuits which form a graph, see Devriendt (2022). Suppose we have a $(p+q)\times(p+q)$ matrix M, which can be represented as a block-partitioned matrix as

$$M = \left(\begin{array}{c|c} E & F \\ \hline G & H \end{array}\right), \tag{4.2.4}$$

where E, F, G, and H are $p\times p$, $p\times q$, $q\times p$, and $q\times q$ matrices, respectively. If H is invertible, then the Schur complement of M with respect to H, denoted as M/H, is defined as

$$M/H = E - FH^{-1}G. \tag{4.2.5}$$

The Schur complement has been used in many areas of numerical analysis as a tool to reduce the size of a linear system, see Zhang (2006). Dorfler and Bullo (2012) consider Schur complement as a tool for analysing the circuit theory after removing a set of nodes in electronic networks Kron reduction. In our framework, suppose at stage-r we have the edge Laplacian $\mathcal{L}^{\mathcal{E},\omega}_r$ and we split the k_r-th edge and perform the prediction by the k_r-th row (or column) of $\mathcal{L}^{\mathcal{E},\omega}_r$. Then we perform a permutation for both rows and columns of $\mathcal{L}^{\mathcal{E},\omega}_r$, so that the order sequence of each row and column

(as an edge version) becomes $(e_1, ..., e_{k_r-1}, e_{k_r+1}, ..., e_m \,|\, e_{k_r})$ instead of $(e_1, ..., e_m)$. Denote the permuted stage-r edge Laplacian matrix by $\mathbf{P}(\mathcal{L}_r^{\mathcal{E},\omega})$, which can be written in block-partitioned form as

$$\mathbf{P}(\mathcal{L}_r^{\mathcal{E},\omega}) = \left(\begin{array}{c|c} \mathcal{L}_r^{\mathcal{E}\setminus\{e_{k_r}\},\omega} & \mathcal{J}_r \\ \hline \mathcal{J}_r^T & \left[\mathcal{L}_r^{\mathcal{E},\omega}\right]_{k_r,k_r} \end{array} \right), \tag{4.2.6}$$

where $\mathcal{L}_r^{\mathcal{E}\setminus\{e_{k_r}\},\omega}$ is the submatrix of $\mathcal{L}_r^{\mathcal{E},\omega}$, associated with the edge sequence without e_{k_r}, which is $(e_1, ..., e_{k_r-1}, e_{k_r+1}, ..., e_m)$; $\mathcal{J}_r$ denotes the k_r-th column of $\mathcal{L}_r^{\mathcal{E},\omega}$, but without the k_r-th entry; $\left[\mathcal{L}_r^{\mathcal{E},\omega}\right]_{k_r,k_r}$ is the k_r-th diagonal entry of $\mathcal{L}_r^{\mathcal{E},\omega}$. The matrix in the bottom left block is the transpose of the one in the top right block because of the symmetry of the generalised edge Laplacian. Notice that $\left[\mathcal{L}_r^{\mathcal{E},\omega}\right]_{k_r,k_r}$ is a special case (1×1) of matrices, whose inverse can be obtained by taking its reciprocal if it is non-zero. Let us first suppose that $\left[\mathcal{L}_r^{\mathcal{E},\omega}\right]_{k_r,k_r}$ is non-zero, then we define the stage-$(r-1)$ edge Laplacian as the Schur complement associated to stage-r edge Laplacian with respect to its k-th diagonal entry, namely

$$\begin{aligned} \mathcal{L}_{r-1}^{\mathcal{E},\omega} &:= \mathbf{P}(\mathcal{L}_r^{\mathcal{E},\omega}) / \left[\mathcal{L}_r^{\mathcal{E},\omega}\right]_{k_r,k_r} \\ &= \mathcal{L}_r^{\mathcal{E}\setminus\{e_{k_r}\},\omega} - \mathcal{J}_r \left(\left[\mathcal{L}_r^{\mathcal{E},\omega}\right]_{k_r,k_r}\right)^{-1} \mathcal{J}_r^T. \end{aligned} \tag{4.2.7}$$

Note that for $\mathcal{J}_r$, only the entries that correspond to $e_s \in \mathcal{N}_{k_r,r}^{\mathcal{E}}$ are non-zero (recall that we define the neighbourhood by the non-zero entries of the k_r-th row or column at stage-r). For the matrix $\mathcal{J}_r \left(\left[\mathcal{L}_r^{\mathcal{E},\omega}\right]_{k_r,k_r}\right)^{-1} \mathcal{J}_r^T$, we first consider its diagonal components. Let $k < k_r$, and then we have

$$\begin{aligned} &\left[\mathcal{J}_r \left(\left[\mathcal{L}_r^{\mathcal{E},\omega}\right]_{k_r,k_r}\right)^{-1} \mathcal{J}_r^T\right]_{kk} \\ &= \begin{cases} \frac{\left(\left[\mathcal{L}_r^{\mathcal{E},\omega}\right]_{k_r,k}\right)^2}{\left[\mathcal{L}_r^{\mathcal{E},\omega}\right]_{k_r,k_r}}, & \text{if } e_k \text{ and } e_{k_r} \text{ are neighbouring edges;} \\ 0, & \text{otherwise;} \end{cases} \end{aligned}$$

For $k > k_r$, we only have to substitute k by $(k-1)$, while the form for off-diagonal entries is cumbersome. Fortunately, some of the properties of the Schur complement can help us to avoid analysing the new edge Laplacian matrix by elements. Let us firstly consider the matrices $\mathcal{L}_m^{\mathcal{E},\omega}$.

Lemma 4. $\mathcal{L}_m^{\mathcal{E},\omega}$ is a positive-definite matrix.

Proposition 4.2.1. If we have a positive-definite matrix that can be written in block partitioning form as in equation (4.2.4), then its Schur complement as in equation (4.2.5) is also positive-definite.

The details of the proofs can be found in Appendices C.3 and C.4, respectively. As a result of Proposition 4.2.1, the matrix $\mathcal{L}_r^{\mathcal{E},\omega}$ we obtain at any stage-r is positive-definite. This in fact leads to many fascinating properties for $\mathcal{L}_r^{\mathcal{E},\omega}$. Firstly, the positive-definiteness ensures that our algorithm is valid. Note that all of the diagonal entries of a positive-definite matrix have to be positive, which indicates that $[\mathcal{L}_r^{\mathcal{E},\omega}]_{k_r,k_r}^{-1}$ exists. Secondly, the positive-definiteness ensures that at any stage-r, the associated matrix can be decomposed as $\mathcal{L}_r^{\mathcal{E},\omega} = [\overrightarrow{B}_r^{\omega}]^T [\overrightarrow{B}_r^{\omega}]$. Hence, the matrix $\mathcal{L}_r^{\mathcal{E},\omega}$ is still an 'edge Laplacian', although obtaining the exact form for stage-r incidence matrix $\overrightarrow{B}_r^{\omega}$ is not straightforward.

3. **Update for the Edge Laplacian via Incidence Matrix**

   - **Updating the graph structure (relinkage):** Similar to LOCAAT proposed by Jansen et al. (2004, 2009), a relinking step will be performed for the graph structure at the end of every stage. As we mentioned in the edge Laplacian form, $\mathcal{L}^{\mathcal{E},\omega} = W^{1/2}\overrightarrow{B}^T\overrightarrow{B}W^{1/2}$, the information on the edges of the graph is given by the matrix W, while the incidence matrix $\overrightarrow{B}$ indicates the connectivity and orientation of the graph. Thus, here we consider performing the relinkage by evolving the matrix $\overrightarrow{B}_r$ at stage-r to $\overrightarrow{B}_{r-1}$ at stage-$(r-1)$. As discussed in E-LOCAAT, the relinkage procedure for a tree graph could be divided into two steps, which are *edge removal* and *vertex merging*. Thus, if we have a matrix of dimension $p \times q$, then after one-step relinkage, the resulting matrix will be a $(p-1) \times (q-1)$ matrix. It is clearer to start from stage-m, where we recall that the i-th row of the incidence matrix $\overrightarrow{B}$ represents the incidence information of the vertex v_i, while the k-th column represents the

incidence information of edge e_k. Suppose from stage-m to stage-$(m-1)$, the edge to be lifted is $e_{k_m} = \{v_{i_m}, v_{j_m}\}$. The edge removal simply means deleting the k_m-th column of $\overrightarrow{B}_m$, and the vertex merging is to combine v_{i_m} and v_{j_m} such that they become a new vertex.

For computational reasons, here we introduce an alternative design, which is to keep one vertex of $\{v_{i_m}, v_{j_m}\}$. Suppose that $i_m < j_m$, then we firstly transform all incidence information of v_{j_m} to v_{i_m}, in terms of a row addition of the i_m-th and the j_m-th row of $\overrightarrow{B}_m$. Since each edge only contains two distinct vertices, if the k-th element of the i_m-th row is non-zero (where $k \neq k_m$, indicates that $v_{i_m} \in e_k$), then the k-th element of the j_m-th row must be zero. Therefore, adding the j_m-th row to the i_m-th row and then deleting the j_m-th row will be our relinkage strategy. In this case, the incidence matrix $\overrightarrow{B}_{m-1}$ at stage-$(m-1)$ is a $(n-1) \times (m-1)$ matrix, where $n = m+1$ for a tree graph. The rows of $\overrightarrow{B}_{m-1}$ represent the incidence information of $\{v_1, ..., v_{j_m-1}, v_{j_m+1}, ..., v_n\}$, therefore, we can construct a map from this set of remaining vertices to the index set of the incidence matrix $\overrightarrow{B}_{m-1}$, which is $\{1, ..., m-1\}$, so that the s-th row of $\overrightarrow{B}_{m-1}$ indicates the incidence information of v_s if $s < j$, or indicates the incidence information of v_{s+1} if $s \geq j$. A similar map can be defined for the set of remaining edges $\{e_1, ..., e_{k_m-1}, e_{k_m+1}, ..., e_m\}$, such that the l-th column of $\overrightarrow{B}_{m-1}$ indicates the incidence information of the edge e_l if $l < k_m$ or indicates the incidence information of the edge e_{l+1} if $l \geq k_m$. According to this construction, we could define the relinkage as a

transform from $\overrightarrow{B}_r$ to $\overrightarrow{B}_{r-1}$, where $e_{k_r} = (v_{i_r}, v_{j_r})$ and $i_r < j_r$, such that

$$\left[\overrightarrow{B}_{r-1}\right]_{ik} = \begin{cases} \left[\overrightarrow{B}_r\right]_{ik}, & \text{if } i < j_r;\ i \neq i_r \text{ and } k < k_r; \\ \left[\overrightarrow{B}_r\right]_{i,k+1}, & \text{if } i < j_r;\ i \neq i_r \text{ and } k \geq k_r; \\ \left[\overrightarrow{B}_r\right]_{i+1,k}, & \text{if } i \geq j_r \text{ and } k < k_r; \\ \left[\overrightarrow{B}_r\right]_{i+1,k+1}, & \text{if } i \geq j_r \text{ and } k \geq k_r; \\ \left[\overrightarrow{B}_r\right]_{ik} + \left[\overrightarrow{B}_r\right]_{j_r,k}, & \text{if } i = i_r \text{ and } k < k_r; \\ \left[\overrightarrow{B}_r\right]_{i,k+1} + \left[\overrightarrow{B}_r\right]_{j_r,k+1}, & \text{if } i = i_r \text{ and } k \geq k_r. \end{cases} \tag{4.2.8}$$

- **Updating the weight matrix:** Recall that our construction is based on the edge Laplacian of the form $\mathcal{L}^{\mathcal{E},\omega} = (\overrightarrow{B}W^{1/2})^T(\overrightarrow{B}W^{1/2}) = W^{1/2}\overrightarrow{B}^T\overrightarrow{B}W^{1/2}$. Therefore we have a further term to be updated, which is the weight matrix W_r at stage-r (here, $W_m = W$). The update of weight matrix could be flexibly completed to account of some real-life measures, such as traffic flow (Deri and Moura; 2015) or streamflow strength (Park et al.; 2022). Our focus here is to describe possible ways of updating the weight matrices and not to find the optimal solution.

  An interesting observation is that many different designs of the prediction and update weights result from different choices of W plugged into the Laplacian-based method, e.g, if we let $[W]_{kk} = 1/\ell_k^2$ for all k, then it initially results in the inverse length prediction weight construction (equation 3.1.15) as discussed in Section 3; if we let $[W]_{kk} = 1$ for all k, then it results in the moving average case. In our work, the initial diagonal entries of W are set as $[W]_{kk} = 1/\ell_k$ as we discussed before in prediction step. We additionally introduce a further approach for the update of the weight matrix from stage-r to stage-$(r-1)$, which is as follows (further options are possible and easily

implemented too),

$$[W_{r-1}]_{kk'} = \begin{cases} 0, & \text{if } k \neq k'; \\ \alpha_{k,r}\,[W_r]_{kk'}, & \text{if } k = k' < k_r \text{ and } k \in \mathcal{N}^{\mathcal{E}}_{k_r,r}; \\ [W_r]_{kk'}, & \text{if } k = k' < k_r \text{ and } k \notin \mathcal{N}^{\mathcal{E}}_{k_r,r}; \\ \alpha_{k,r}\,[W_r]_{k+1,k'+1}, & \text{if } k = k' \geq k_r \text{ and } k \in \mathcal{N}^{\mathcal{E}}_{k_r,r}; \\ [W_r]_{k+1,k'+1}, & \text{if } k = k' \geq k_r \text{ and } k \notin \mathcal{N}^{\mathcal{E}}_{k_r,r}, \end{cases}$$

where $\alpha_{k,r}$ is a shrinkage parameter satisfies $0 < \alpha_{k,r} < 1$. The reason we decrease the size of the weights associated with the neighbourhood set $\mathcal{N}^{\mathcal{E}}_{k_r,r}$ is that larger weights cause issues with the algorithm stability and scale-mixing, see Jansen and Oonincx (2005). In this work, we set $\alpha_{k,r} = 1/2$ to test the performance of our method as this aligns well with the proposed recursive construction for the edge Laplacian. An interesting question for the future is to construct an optimal value $\alpha_{k,r}$ by means of exploiting quantities such as the Laplacian distance mentioned in Severn et al. (2021), or eigenvalue/eigenvector properties described by Chung (1997).

4. **Computational Aspects of the Update Step**

As the algorithm progresses, we will decrease the dimension of incidence matrices and weight matrices, as they are updated through the algorithm. Computationally, it is easier to instead keep the initial dimension and replace the corresponding rows and columns with zero vectors. In this section, we will mainly discuss the computational update steps based on the weight matrix and incidence matrix, since the only thing that has to be considered in the update step by Schur complement is the permutation of the rows/columns. Then the incidence matrix relinkage could be divided into two steps as follows:

- **Vertex merging:** Let us now define the dimension-preserving oriented incidence matrix $\overrightarrow{\tilde{B}}_r \in \mathbb{R}^{n\times m}$ for all $r \in \{m, ..., 2\}$, and the removal edge at

this stage is $e_{k_r} = \{v_{i_r}, v_{j_r}\}$. The vertex-merging could be completed by combining the i_r-th and j_r-th rows of $\overrightarrow{\tilde{B}}_r$. Denote $\text{row}_{i_r}(\overrightarrow{\tilde{B}}_r)$ as the vector of dimension m representing the i-th row of $\overrightarrow{\tilde{B}}_r$, then the vertex merging could be expressed as

$$\begin{aligned} \text{row}_{i_r}\left(\overrightarrow{\tilde{B}}_{r-1}\right) &= \text{row}_{i_r}\left(\overrightarrow{\tilde{B}}_r\right) + \text{row}_{j_r}\left(\overrightarrow{\tilde{B}}_r\right). \\ \text{row}_{j_r}\left(\overrightarrow{\tilde{B}}_{r-1}\right) &= \mathbf{0}, \end{aligned} \tag{4.2.9}$$

where $\mathbf{0}$ is the zero-vector of the same dimension as $\text{row}_{j_r}\left(\overrightarrow{\tilde{B}}_r\right)$. There is no significance in the choice of i_r and j_r, but here we always assume that $i_r < j_r$, thus we always plug the information from j_r-th row into i_r-th row. We notice that the i_r-th row of the matrix $\overrightarrow{\tilde{B}}_r$ is the incidence information of the vertex v_{i_r}, see the matrix form (1.4.2). Due to the fact that two distinct vertices, for example, v_{i_r} and v_{j_r}, cannot simultaneously belong to more than one edge, $\text{row}_{i_r}(\overrightarrow{\tilde{B}}_r)$ and $\text{row}_{j_r}(\overrightarrow{\tilde{B}}_r)$ only have one common non-zero element at the same location, which is the k_r-th element, then the k-th element of $\text{row}_{i_r}(\overrightarrow{\tilde{B}}_{r-1})$ can be written as

$$[\text{row}_{i_r}(\overrightarrow{\tilde{B}}_{r-1})]_k = \begin{cases} 0, & \text{if } k = k_r; \\ 1, & \text{if } k \neq k_r \text{ and if } v_{i_r} \in e_k \text{ or } v_{j_r} \in e_{k_r}; \\ 0, & \text{otherwise.} \end{cases} \tag{4.2.10}$$

The reason for the k-th element becoming zero if $k = k_r$ is that one of $[\overrightarrow{\tilde{B}}]_{i_r,k_r}$ and $[\overrightarrow{\tilde{B}}]_{j_r,k_r}$ is 1 and another is -1.

- **Edge Removal:** Once we lifted an edge e_{k_r} at stage-r, we have to remove the corresponding column from the whole matrix incidence structure. For achieving this, we could just simply remove the k_r-th column from the dimension-preserving incidence matrix $\overrightarrow{\tilde{B}}_r$. Notice that after performing vertex merging (equation (4.2.10)), the k_r-th column immediately becomes a vector of all zeros, since at stage-r, only the i_r-th and j_r-th element of k_r-th column are non-zero.

A similar update will also be performed for the weight matrix computation. The construction for the weight matrix is more intuitive than for the incidence matrix, where we only have to update the weights associated with the neighbourhood $\mathcal{N}^{\mathcal{E}}_{k_r,r}$. The element indicating the weight of k_r-th edge, $[W_r]_{k_r,k_r}$, will not contribute to the edge Laplacian at stage-$(r-1)$ because the k_r-th column of $\overrightarrow{\tilde{B}}_{r-1}$ is $\mathbf{0}$, see the discussion above.

Inverse Transform

The inverse transform can be done by simply undoing the predict and update steps, which are

$$c^{\mathcal{E}}_{j,r} = c^{\mathcal{E}}_{j,r-1} - b^{\mathcal{E}}_{j,r}\, d^{\mathcal{E}}_{k_r}.$$
$$c^{\mathcal{E}}_{k_r,r} = d^{\mathcal{E}}_{k_r} + \sum_{j \in \mathcal{N}^{\mathcal{E}}_{k_r,r}} a^{\mathcal{E}}_{j,r} c^{\mathcal{E}}_{j,r}.$$

Recall the lifting coefficient array designed for the inverse transform, see Section 2.3.2, which consists of indicators for the removal edge and its neighbourhood, and the (prediction and update) weights at each stage. In their construction, there is no other information that has to be stored in this array because of the inverse of the minimal spanning tree relinkage is uniquely defined, see Jansen et al. (2009) for more details. However, this is not the case with our method. For the ability to recover the graph structure, in terms of recovering from $\overrightarrow{\tilde{B}}_{m-\tau}$ (τ is the number of lifted edges) to $\overrightarrow{\tilde{B}}_{m}$, more information must be stored in the lifting array. We set our recovery for the incidence matrix as follows. Firstly, we want to have a map that recovers $\mathrm{row}_{j_r}\left(\overrightarrow{\tilde{B}}_r\right)$ from $\mathrm{row}_{j_r}\left(\overrightarrow{\tilde{B}}_{r-1}\right)$. Through this step, we could see that the essential items for recovery are the k_r-th element value of $\mathrm{row}_{j_r}\left(\overrightarrow{\tilde{B}}_r\right)$, and the index j_r for the vertex v_{j_r} and a subset of $\mathcal{N}^{\mathcal{E}}_{k_r,r}$, which indicates those edges that contain the vertex v_{j_r} at stage-r. Let us denote this by $\mathrm{sub}_{j_r}(\mathcal{N}^{\mathcal{E}}_{k_r,r}) = \{l \mid e_l \in \mathcal{N}^{\mathcal{E}}_{k_r,r} \text{ and } v_{j_r} \in e_l\}$. This subset can be easily determined by the locations of those non-zero elements in $\mathrm{row}_{j_r}\left(\overrightarrow{\tilde{B}}_r\right)$ except k_r while performing the forward transform. Similarly, the value $\left[\mathrm{row}_{j_r}\left(\overrightarrow{\tilde{B}}_r\right)\right]_{k_r}$ will be stored throughout the

algorithm. The next step will be obtaining $\mathrm{row}_{i_r}\left(\overrightarrow{\tilde{B}}_r\right)$ by

$$\mathrm{row}_{i_r}\left(\overrightarrow{\tilde{B}}_r\right) := \mathrm{row}_{i_r}\left(\overrightarrow{\tilde{B}}_{r-1}\right) - \mathrm{row}_{j_r}\left(\overrightarrow{\tilde{B}}_r\right) \tag{4.2.11}$$

Through this step we do not have to store any other information, the k_r-th column is automatically recovered through the equation (4.2.11). The associated vertices v_{i_r} and v_{j_r} (indices i_r and j_r) have to be stored as well. Thus, the lifting array in our method is

$$k_r \quad |\mathcal{N}^{\mathcal{E}}_{k_r,r}| \quad S^{\mathcal{E}}_r \quad |\mathrm{sub}_{j_r}\left(\mathcal{N}^r_{k_r}\right)| \quad \mathrm{sub}_{j_r}\left(\mathcal{N}^r_{k_r}\right) \quad \underline{a}^{\mathcal{E}}_r \quad \underline{b}^{\mathcal{E}}_r \quad i_r \quad j_r \quad \left[\mathrm{row}_{j_r}\left(\overrightarrow{\tilde{B}}_r\right)\right]_{k_r}.$$

where $S^{\mathcal{E}}_r$ is the set consists of all s such that $e_s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}$, and $\underline{a}^{\mathcal{E}}_r$; $\underline{b}^{\mathcal{E}}_r$ are sequences of predict and update coefficients.

The inverse transform corresponding to using the Schur complement is more straightforward than the one via updating incidence and weight matrices. Since the Schur complement naturally gives us the structure update (relinkage), we can simply set the lifting array as

$$k_r \quad |\mathcal{N}^{\mathcal{E}}_{k_r,r}| \quad S^{\mathcal{E}}_r \quad \underline{a}^{\mathcal{E}}_r \quad \underline{b}^{\mathcal{E}}_r.$$

where $S^{\mathcal{E}}_r$ is the set consists of all s such that $e_s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}$, and $\underline{a}^{\mathcal{E}}_r$; $\underline{b}^{\mathcal{E}}_r$ are sequences of predict and update coefficients. This lifting array enables us to recover the signal, for recovering the whole edge Laplacian structure, we only have to store the value $\left[\mathcal{L}^{\mathcal{E},\omega}_r\right]_{k_r,k_r}$ and the associated non-zero values of the vector $\mathcal{J}_r$ (see equation (4.2.7)), which is of the same size of the neighbourhood $\mathcal{N}^{\mathcal{E}}_{k_r,r}$. Then the Schur complement can be obtained by undoing equation (4.2.7).

4.2.2 Proposed LOCAAT via the Line Graph Laplacian

Recalling the algebraic relation between the line graph and corresponding edge Laplacian of the original graph (equation (4.1.9)) suggests to propose rewriting equation (4.1.9) in a weighted form, specifically

$$A(\mathbf{LG}(G^\omega)) = \left|\mathcal{L}^{\mathcal{E},\omega}(G^\omega) - 2\,W\right|.$$

Here the weight matrix W is used instead of the identity matrix $\mathbf{I}$, since its role is simply to turn the diagonal entries to be all zeros. We next define the degree matrix as

$$D(\mathbf{LG}(G^{\omega})) = \begin{pmatrix} \sum_{j=1}^{n}[A(\mathbf{LG}(G^{\omega}))]_{1j} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \sum_{j=1}^{n}[A(\mathbf{LG}(G^{\omega}))]_{nj} \end{pmatrix},$$

thus, the line graph Laplacian can be obtained by

$$\mathfrak{L}(\mathbf{LG}(G^{\omega})) = D(\mathbf{LG}(G^{\omega})) - A(\mathbf{LG}(G^{\omega})). \tag{4.2.12}$$

Since $D(\mathbf{LG}(G^{\omega}))$ is determined by $A(\mathbf{LG}(G^{\omega}))$, hence, the graph information can be encoded into the edge adjacency matrix. The matrix $A^{\hat{\mathcal{V}},\omega} = A(\mathbf{LG}(G^{\omega}))$ is the algebraic equivalent to the edge adjacency matrix, but defined on the set of line graph vertices $\hat{\mathcal{V}}$. Therefore, the matrix $\mathfrak{L}(\mathbf{LG}(G^{\omega}))$ (or $\mathfrak{L}^{\hat{\mathcal{V}},\omega}$) can lead to a variant of the Laplacian-LOCAAT transform, in which the line graph Laplacian is used instead of the edge Laplacian.

The lengths of the new edges correspond to the reciprocal of their weights, which can be written as $\ell'(\hat{e}_{ks}) = 1/[A(\mathbf{LG}(G^{\omega}))]_{ks}$, where $\hat{e}_{ks} = \{\hat{v}_k, \hat{v}_s\}$. We denote $\hat{\Gamma}$ as the metrized graph space for the graph given by the line graph Laplacian $\mathfrak{L}^{\hat{\mathcal{V}},\omega}$, then the partitionings and the functional forms can be constructed as in Section 2.3.1.

We propose the following algorithm.

Split

Consider the construction of the scaling functions as in Chapter 2, then for the new line graph Laplacian-LOCAAT, we can start with their associated choices of initial scaling function integral values. Then we can have the same initial integral measure as in LG-LOCAAT, where the sum of distances can be represented as in equation (2.3.6), such that

$$I_{k,m}^{\hat{\mathcal{V}},\text{sum}} = \sum_{s \in \mathcal{N}_{k,m}^{\hat{\mathcal{V}}}} \ell'(\hat{e}_{ks}),$$

where $\mathcal{N}_{k,m}^{\hat{\mathcal{V}}}$ is determined by the non-zero entries of $A^{\hat{\mathcal{V}},\omega}$.

Similarly, the average distance can be written as in equation (2.3.7), such that

$$I_{k,m}^{\hat{\mathcal{V}},\text{ave}} = \frac{I_{k,m}^{\hat{\mathcal{V}},\text{sum}}}{|\mathcal{N}_{k,m}^{\hat{\mathcal{V}}}|},$$

We can also set initial integral values corresponding to the lazy lifting, such that

$$I_{k,m}^{\hat{\mathcal{V}},\text{Delta}} = 1,$$

as in equation (2.3.8).

The split strategy will still based on the line graph vertex with the minimum integral value, and a random choice when several integral values are the same.

Predict

We propose to alter the prediction weights by substituting the stage-r edge Laplacian with stage-r line graph Laplacian $\mathfrak{L}_r^{\hat{\mathcal{V}},\omega}$. Thus, for the chosen edge e_{k_r} and its neighbourhood $\mathcal{N}_{k_r,r}^{\hat{\mathcal{V}}}$ the prediction weights are given by

$$a_{s,r}^{\hat{\mathcal{V}}} = \frac{\left|\left[\mathfrak{L}_r^{\hat{\mathcal{V}},\omega}\right]_{k_r,s}\right|}{\sum_{t:e_t\in\mathcal{N}_{k_r,r}^{\hat{\mathcal{V}}}}\left|\left[\mathfrak{L}_r^{\hat{\mathcal{V}},\omega}\right]_{k_r,t}\right|}.$$

and the detail coefficient can be obtained by

$$d_{k_r}^{\hat{\mathcal{V}}} = c_{k_r,r}^{\hat{\mathcal{V}}} - \sum_{s:\, e_s\in\mathcal{N}_{k_r,r}^{\hat{\mathcal{V}}}} a_{s,r}^{\hat{\mathcal{V}}} c_{s,r}^{\hat{\mathcal{V}}}.$$

Update

Note that the initial line graph Laplacian $\mathfrak{L}_m^{\hat{\mathcal{V}},\omega}$ cannot be represented by the incidence matrix $\overrightarrow{B}$ and the weight matrix W of the original graph. In fact, we have to construct a new incidence matrix and an associated weight matrix of higher dimensions, since the line graph normally consists of more edges than the original graph, which might cause computational inefficiency. Moreover, note that the relinkage strategy through updating

the incidence matrix is computationally non-problematic because we do not have to add any structure. Removing edges and combining vertices can be simply coded as deleting and performing addition for associated rows and columns in the incidence matrix. However, it is impossible to complete the relinkage for the line graph without checking the graph connectivity, hence structure updating through the incidence matrix and weight matrix might not be a good choice when considering the associated computational effort. However, updating structure by the Schur complement is still valid, and in fact, taking the Schur complement of a graph Laplacian has many desirable properties, as follows.

Theorem (Devriendt (2022))**.** The Schur complement of a graph Laplacian is a graph Laplacian.

We also have the following theorem assuring the connectivity of the stage-r line graph structure.

Theorem (Dorfler and Bullo (2012))**.** The Schur complement of an irreducible graph Laplacian is also irreducible.

The irreduciblity of a graph Laplacian is equivalent to the connectivity of the vertex set, see Dorfler and Bullo (2012). For an irreducible graph Laplacian, the diagonal components have to be non-zero positive values. Hence, performing the Schur complement as the structure update (and relinkage) is still valid, as discussed in Section 4.2.1. The algorithm then consists of re-iterating the split-predict-update steps we described above.

4.3 Simulation Study

In this section, we test the proposed algorithm on the simulation test functions described in Chapter 2.4.1. A list consisting of the descriptions and acronyms of our chosen algorithms is provided in Table 4.1.

Acronym	
EL-SC-L	LOCAAT algorithm with edge Laplacian, Laplacian updated by Schur complement, the initial weight matrix is given by $W_{kk}^m = 1/\ell_k$, and initial scaling function integral value determined by lengths.
EL-W-L	LOCAAT algorithm with edge Laplacian, Laplacian updated by weight matrix and incidence matrix, the initial weight matrix is given by $W_{kk}^m = 1/\ell_k$, and initial scaling function integral value determined by lengths.
EL-SC-D	LOCAAT algorithm with edge Laplacian, Laplacian updated by Schur complement, the initial weight matrix is given by $W_{kk}^m = 1/\ell_k$, and initial scaling function integral value determined by a sequence of ones.
EL-W-D	LOCAAT algorithm with edge Laplacian, Laplacian updated by weight matrix and incidence matrix, the initial weight matrix is given by $W_{kk}^m = 1/\ell_k$, and initial scaling function integral value determined by a sequence of ones.
LGL-SC-S	LOCAAT algorithm with Line graph Laplacian, Laplacian updated by Schur complement, sum of distances as the integrals of the initial primal scaling functions.
LGL-SC-A	LOCAAT algorithm with Line graph Laplacian, Laplacian updated by Schur complement, average distances as the integrals of the initial primal scaling functions.
LGL-SC-D	LOCAAT algorithm with Line graph Laplacian, Laplacian updated by Schur complement, a sequence of ones as the integrals of the initial primal scaling functions.

Table 4.1: Acronyms and algorithm descriptions for different parameter choices of LG-LOCAAT.

4.3.1 Stability

The stability study will be presented in this section. We note that the stability results are quite similar for most of the algorithms, except for 'EL-SC-L' and 'LGL-LG-S-Rem', which appears less stable than other algorithms, namely, 'EL-SC-D'. The sparsity results are slightly better than LG-LOCAAT, but slightly less stable when compared with E-LOCAAT.

Condition Number	Max	75%	Median	25%	Min
EL-SC-L	14.0715	11.9665	11.5194	11.0676	10.7022
EL-W-L	13.1193	11.8048	11.4902	11.0826	10.5132
EL-SC-D	**12.0302**	**11.1227**	**10.7234**	**10.2282**	**9.9567**
EL-W-D	12.1063	11.0290	10.7075	10.3864	10.0916
LGL-SC-S	13.5790	12.4173	11.8303	11.4840	10.8405
LGL-SC-A	13.2385	11.8782	11.1536	10.8220	10.2659
LGL-SC-D	12.9910	12.1666	11.5973	11.2536	10.5370

Table 4.2: Condition number for E-LOCAAT on a tree structure.

4.3.2 Sparsity

Similar to Sections 2 and 3, the sparsity plots for Laplacian-LOCAAT will be constructed in the same way. The results will be presented for both pointwise functions and the edge averaging function as we discussed in Section 2.4.1.

According to these figures, we can see that the sparsity for each function follows a similar pattern to the LG-LOCAAT and E-LOCAAT. The compression ability for most of the functions is good, however it is still difficult to attain a good compression for the Heavisine function.

4.3.2.1 Sparsity Plot for Pointwise Functions

In this section, we provide the sparsity plots for different schemes performed on the pointwise functions as described in equation (2.4.1).

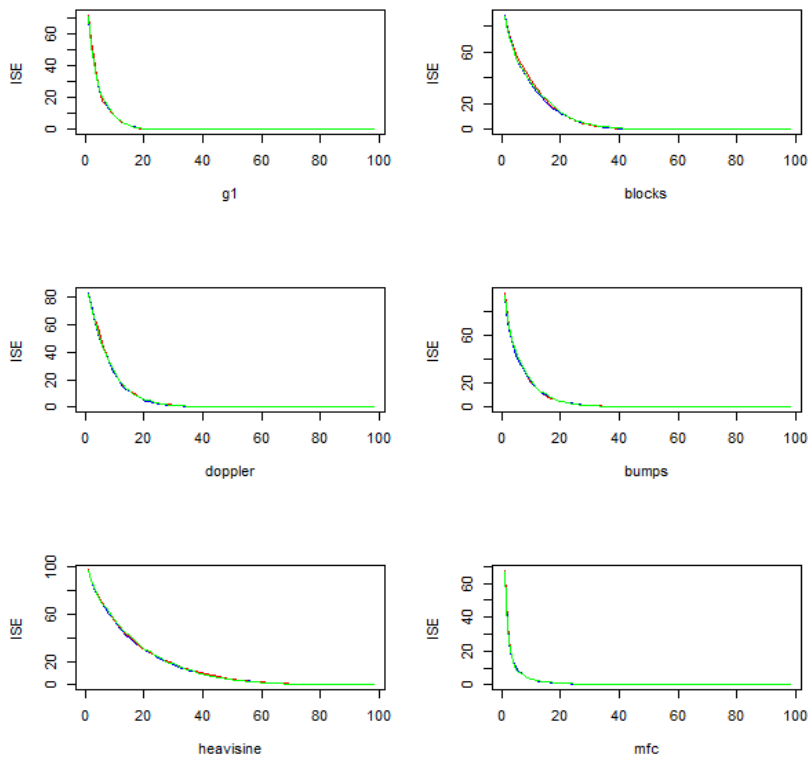

Figure 4.1: Sparsity plots for the test functions used in simulation by the equation (2.4.1). The scheme is based on the edge Laplacian, and updated by Schur complement, and incidence and weight matrix. From left to right on *top row*: g_1, blocks; *middle row*: doppler, bumps; *bottom row*: heavisine, maartenfunc. **Black line**: EL-SC-L; **Red line**: EL-W-L. **Blue line**: EL-SC-D; **Green line**: EL-W-D.

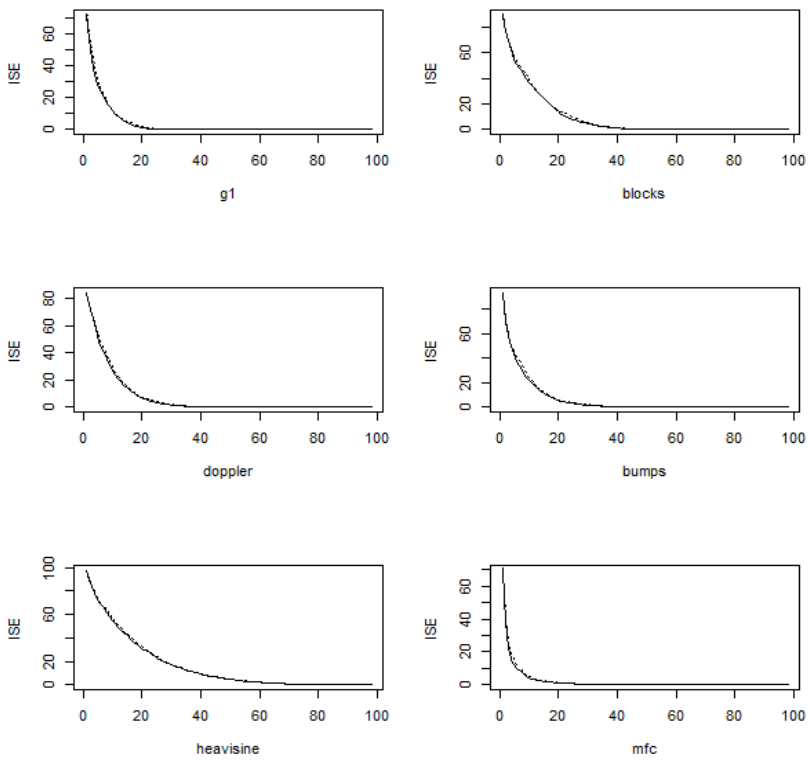

Figure 4.2: Sparsity plots for the test functions used in simulation by the equation (2.4.1). The scheme is based on the line graph Laplacian, and updated by Schur complement. From left to right on *top row*: g_1, blocks; *middle row*: doppler, bumps; *bottom row*: heavisine, maartenfunc. **Black line**: LGL-SC-S; **dashed black line**: LGL-SC-A; **dotted black line**: LGL-SC-D.

4.3.2.2 Sparsity Plot for Edge Averaging Functions

In this section, we provide the sparsity plots for different schemes performed on the pointwise functions as described in equation (2.4.2).

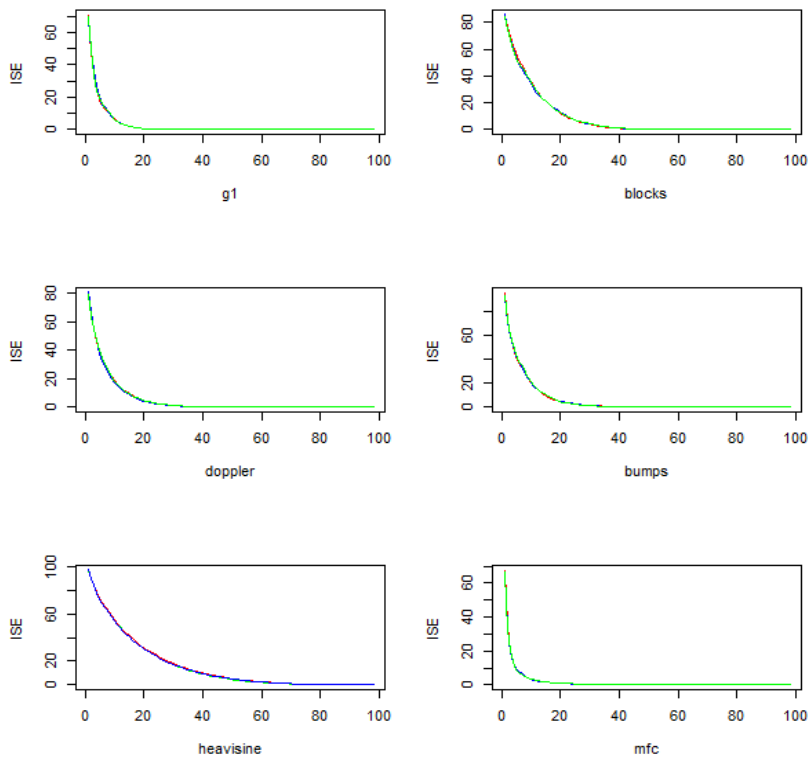

Figure 4.3: Sparsity plots for the test functions used in simulation by the equation (2.4.2). The scheme is based on the edge Laplacian, and updated by Schur complement, and incidence and weight matrix. From left to right on *top row*: g_1, blocks; *middle row*: doppler, bumps; *bottom row*: heavisine, maartenfunc. **Black line**: EL-SC-L; **Red line**: EL-W-L. **Blue line**: EL-SC-D; **Green line**: EL-W-D.

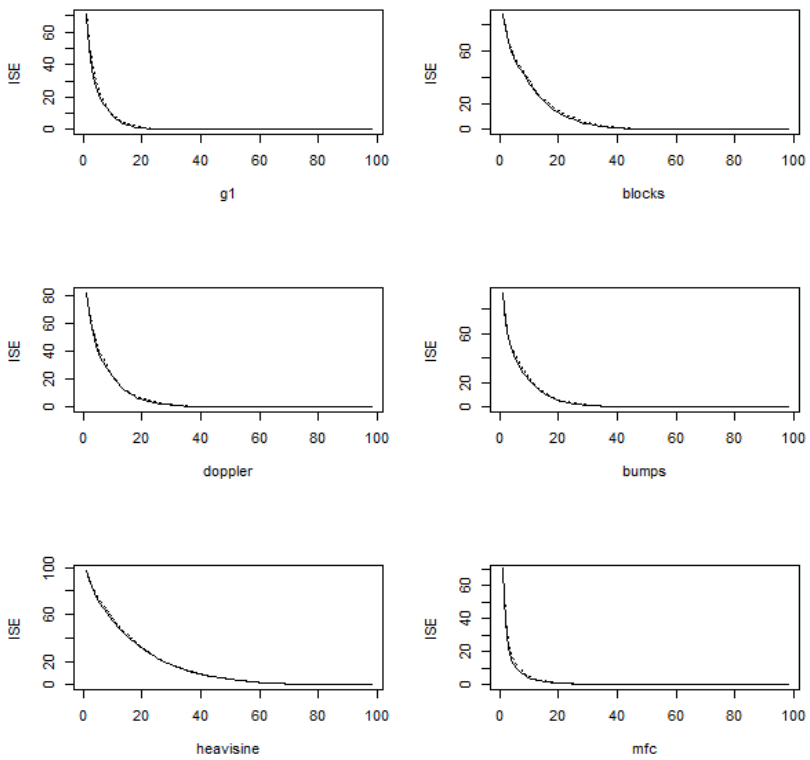

Figure 4.4: Sparsity plots for the test functions used in simulation by the equation (2.4.2). The scheme is based on the line graph Laplacian, and updated by Schur complement. From left to right on *top row*: g_1, blocks; *middle row*: doppler, bumps; *bottom row*: heavisine, maartenfunc. **Black line**: LGL-SC-S; **dashed black line**: LGL-SC-A; **dotted black line**: LGL-SC-D.

4.3.3 Denoising Performance

In this section, we will provide a simulation study for the functions also tested in Chapters 2 and 3.

4.3.3.1 Denoising Pointwise Functions

From Tables 4.3, 4.4, and 4.5, we can see that the methods with 'D' (a sequence of ones) as the initial integral values are competitive. However, the two proposed methods of updating the coefficients (through the Schur complement and through incidence and weight matrices) with both edge Laplacian or line graph Laplacian yield similar results,

and close to the best results obtained across all proposed algorithms. Compared with E-LOCAAT and LG-LOCAAT, the biorthogonal Haar is still the best choice when the underlying function is mostly piecewise constant (g_1 and Blocks). We can see that the algorithms with line graph Laplacian ('LGL') show similar patterns to the results reported in LG-LOCAAT. 'EL-SC-D', 'EL-W-D', and 'LGL-SC-D' deserve more emphasis, since they are totally algebraic, which works even when the data do not form a metric space.

AMSE$\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
EL-SC-L	72 (22)	135 (58)	108 (38)	94 (27)	237 (78)	52 (12)
EL-W-L	73 (22)	138 (60)	109 (38)	94 (27)	237 (78)	54 (12)
EL-SC-D	67 (19)	126 (50)	97 (32)	84 (21)	251 (92)	48 (11)
EL-W-D	67 (19)	125 (50)	98 (32)	85 (21)	251 (91)	49 (11)
LGL-SC-S	68 (21)	**121 (50)**	115 (41)	100 (29)	292 (101)	56 (13)
LGL-SC-A	67 (20)	123 (47)	100 (34)	87 (23)	**216 (76)**	48 (11)
LGL-SC-D	**66 (19)**	123 (48)	**96 (31)**	**83 (21)**	250 (89)	**47 (11)**
SNR=5						
EL-SC-L	27 (10)	49 (22)	44 (18)	44 (15)	171 (58)	25 (5)
EL-W-L	27 (10)	50 (24)	43 (17)	44 (15)	171 (57)	26 (6)
EL-SC-D	25 (8)	49 (22)	**39 (13)**	**40 (13)**	190 (84)	24 (5)
EL-W-D	25 (8)	48 (21)	**39 (13)**	**40 (13)**	190 (84)	24 (5)
LGL-SC-S	**23 (8)**	**44 (18)**	46 (17)	49 (18)	225 (89)	27 (6)
LGL-SC-A	24 (8)	46 (20)	40 (15)	42 (14)	**154 (59)**	**23 (5)**
LGL-SC-D	24 (8)	49 (21)	**39 (14)**	**40 (13)**	191 (83)	**23 (5)**
SNR=7						
EL-SC-L	13 (5)	25 (12)	24 (9)	27 (10)	152 (52)	16 (4)
EL-W-L	12 (5)	26 (12)	24 (9)	27 (10)	151 (49)	17 (4)
EL-SC-D	12 (4)	26 (12)	**21 (7)**	**25 (9)**	171 (80)	**15 (4)**
EL-W-D	12 (4)	25 (12)	**21 (7)**	**25 (9)**	171 (81)	16 (4)
LGL-SC-S	**11 (3)**	**22 (9)**	26 (10)	31 (13)	203 (85)	18 (5)
LGL-SC-A	**11 (4)**	24 (11)	22 (8)	**25 (10)**	**136 (53)**	**15 (3)**
LGL-SC-D	12 (4)	26 (12)	**21 (8)**	**25 (9)**	173 (81)	**15 (3)**

Table 4.3: The AMSE table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1).

4.3.3.2 Denoising Edge Averaging Function

For the edge averaging functions, the results yield similar inferences when compared to those associated to pointwise functions.

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
EL-SC-L	50	71	61	54	70	40
EL-W-L	51	72	62	54	70	41
EL-SC-D	52	86	70	61	113	40
EL-W-D	53	86	72	62	115	41
LGL-SC-S	50	68	63	54	70	41
LGL-SC-A	**48**	**66**	**59**	**51**	**66**	**38**
LGL-SC-D	51	83	68	58	107	39
SNR=5						
EL-SC-L	20	27	24	**22**	29	16
EL-W-L	20	27	24	23	29	16
EL-SC-D	21	35	29	28	75	17
EL-W-D	21	34	30	28	76	18
LGL-SC-S	20	**25**	24	23	32	16
LGL-SC-A	**19**	**25**	**23**	**22**	**28**	**15**
LGL-SC-D	21	34	28	26	71	16
SNR=7						
EL-SC-L	10	14	13	13	**16**	**9**
EL-W-L	10	14	13	13	**16**	**9**
EL-SC-D	11	19	16	17	64	11
EL-W-D	11	18	16	17	65	11
LGL-SC-S	**9**	**13**	13	13	19	**9**
LGL-SC-A	**9**	**13**	**12**	**12**	**16**	**9**
LGL-SC-D	11	19	15	16	60	10

Table 4.4: The variance table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1).

Empirical Computational Cost

The standard running time for Laplacian-LOCAAT is 00:08:30, and 90% of the algorithms completing within 00:11:00. Hence the algebraic construction at the heart of the Laplacian-LOCAAT algorithms offers a faster computation than for both LG-LOCAAT and E-LOCAAT.

bias$^2 \times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
EL-SC-L	22	64	48	40	167	12
EL-W-L	23	66	47	40	167	13
EL-SC-D	15	40	27	24	138	9
EL-W-D	**14**	**39**	**26**	**23**	**136**	**8**
LGL-SC-S	17	53	53	46	222	15
LGL-SC-A	19	57	41	37	150	10
LGL-SC-D	**14**	40	28	25	143	8
SNR=5						
EL-SC-L	7	23	20	22	142	9
EL-W-L	7	23	19	22	142	10
EL-SC-D	**3**	14	**9**	**12**	114	**6**
EL-W-D	**3**	**13**	**9**	**12**	**113**	**6**
LGL-SC-S	5	18	22	27	193	11
LGL-SC-A	6	21	17	20	126	8
LGL-SC-D	4	14	10	13	120	**6**
SNR=7						
EL-SC-L	3	12	11	15	136	7
EL-W-L	2	12	11	14	135	7
EL-SC-D	**1**	**7**	**5**	**8**	107	**5**
EL-W-D	**1**	**7**	**5**	**8**	**106**	**5**
LGL-SC-S	2	9	13	18	183	9
LGL-SC-A	2	11	10	13	120	**5**
LGL-SC-D	1	**7**	6	9	112	**5**

Table 4.5: The squared bias table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1).

AMSE$\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
			SNR=3			
EL-SC-L	69 (19)	148 (60)	105 (35)	95 (28)	257 (86)	52 (12)
EL-W-L	70 (20)	151 (62)	105 (34)	94 (27)	256 (87)	53 (12)
EL-SC-D	62 (17)	125 (46)	92 (28)	83 (21)	243 (85)	47 (11)
EL-W-D	62 (17)	125 (45)	93 (28)	84 (21)	243 (85)	48 (11)
LGL-SC-S	67 (21)	134 (53)	109 (34)	101 (29)	302 (100)	55 (13)
LGL-SC-A	62 (17)	131 (49)	95 (29)	87 (24)	**228 (78)**	48 (11)
LGL-SC-D	**61 (16)**	**123 (43)**	**91 (27)**	**82 (20)**	242 (82)	**46 (11)**
			SNR=5			
EL-SC-L	31 (11)	62 (30)	46 (18)	45 (16)	193 (69)	25 (5)
EL-W-L	31 (11)	63 (30)	45 (17)	45 (15)	193 (68)	26 (5)
EL-SC-D	**26 (8)**	**53 (23)**	**40 (14)**	40 (12)	187 (77)	23 (5)
EL-W-D	**26 (8)**	**53 (22)**	**40 (14)**	40 (12)	187 (77)	24 (5)
LGL-SC-S	27 (9)	54 (24)	48 (18)	50 (19)	240 (91)	27 (6)
LGL-SC-A	**26 (8)**	55 (25)	41 (15)	42 (14)	**168 (64)**	**22 (5)**
LGL-SC-D	**26 (8)**	**53 (23)**	**40 (14)**	**39 (12)**	189 (77)	**22 (5)**
			SNR=7			
EL-SC-L	16 (6)	34 (17)	26 (10)	27 (11)	174 (61)	16 (4)
EL-W-L	16 (6)	34 (18)	25 (10)	27 (11)	174 (61)	17 (4)
EL-SC-D	**14 (5)**	30 (14)	**22 (8)**	**24 (9)**	170 (75)	15 (3)
EL-W-D	**14 (4)**	**29 (14)**	**22 (8)**	**24 (9)**	170 (75)	15 (4)
LGL-SC-S	**14 (5)**	**29 (14)**	27 (11)	31 (13)	220 (86)	18 (5)
LGL-SC-A	**14 (5)**	30 (15)	23 (9)	26 (10)	**150 (57)**	**14 (3)**
LGL-SC-D	**14 (4)**	30 (14)	**22 (9)**	**24 (9)**	173 (76)	**14 (3)**

Table 4.6: The AMSE table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2).

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
EL-SC-L	46	69	57	54	70	40
EL-W-L	47	70	58	53	70	41
EL-SC-D	49	83	65	60	111	40
EL-W-D	49	83	66	61	112	41
LGL-SC-S	47	67	58	53	68	41
LGL-SC-A	**44**	**65**	**54**	**50**	**66**	**38**
LGL-SC-D	47	81	62	57	104	39
SNR=5						
EL-SC-L	19	28	24	22	29	16
EL-W-L	20	28	24	23	29	16
EL-SC-D	21	37	29	28	74	17
EL-W-D	21	37	29	28	75	18
LGL-SC-S	19	27	24	23	31	16
LGL-SC-A	**18**	**26**	**22**	**21**	**28**	**15**
LGL-SC-D	21	36	28	26	70	16
SNR=7						
EL-SC-L	11	15	13	13	16	9
EL-W-L	11	15	13	13	17	9
EL-SC-D	12	21	16	17	63	10
EL-W-D	12	21	17	17	64	11
LGL-SC-S	**10**	**14**	13	13	19	9
LGL-SC-A	**10**	**14**	**12**	**12**	**16**	**8**
LGL-SC-D	11	20	16	16	59	9

Table 4.7: The variance table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2).

bias$^2 \times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
EL-SC-L	23	79	48	41	187	12
EL-W-L	23	80	47	41	187	12
EL-SC-D	**13**	42	**27**	23	133	**8**
EL-W-D	**13**	**41**	**27**	**22**	**131**	**8**
LGL-SC-S	20	67	51	47	233	14
LGL-SC-A	18	66	41	37	163	10
LGL-SC-D	**13**	43	28	25	138	**8**
SNR=5						
EL-SC-L	11	34	22	22	164	9
EL-W-L	11	34	22	22	164	9
EL-SC-D	**5**	**16**	11	**12**	113	**6**
EL-W-D	**5**	**16**	**10**	**12**	**112**	**6**
LGL-SC-S	8	28	24	27	209	11
LGL-SC-A	8	29	19	20	140	8
LGL-SC-D	**5**	17	12	13	119	**6**
SNR=7						
EL-SC-L	6	19	13	15	157	7
EL-W-L	5	20	13	15	157	7
EL-SC-D	**2**	**9**	**6**	8	107	**4**
EL-W-D	**2**	**9**	**6**	**7**	**106**	**4**
LGL-SC-S	4	15	14	18	201	9
LGL-SC-A	4	16	11	13	134	6
LGL-SC-D	2	9	7	9	113	5

Table 4.8: The squared bias table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2).

Chapter 5

Hydrological Data Analysis via Non-decimated Algorithms

In this chapter, we propose non-decimated versions for the algorithms we introduced in Chapters 2, 3, and 4. We show that the non-decimated lifting transform can be easily implemented in computation, as shown in Knight and Nason (2009) for the one-dimensional case, and the resulting algorithms have competitive properties when evaluated on simulated flow data and dissolved oxygen (DO) dataset. Chapters 2 and 5 have been (partially) packaged as a preprint (Cao et al.; 2024) and submitted for publication.

5.1 Non-decimated Lifting Transform

To bring the 'non-decimation' concept into the irregular data setup, Knight and Nason (2009) introduced a non-decimated lifting transform (NLT) based on the LOCAAT framework from Jansen et al. (2004, 2009). Instead of using the shift operator, the non-decimation is introduced via different 'trajectories' of removal order.

The framework of the NLT can be described as follows. Suppose we are still following the nonparametric regression problem and that there is an underlying true, unknown function g, sampled at a set of irregular locations $\underline{x} = \{x_1, ..., x_n\}$, which gives us a set of noisy observations $\underline{f} = \{f_1, ..., f_n\}$. Unlike the general LOCAAT transform, the split step

of NLT is determined by using a predefined removal order. We denote $T = (x_{o_1}, ..., x_{o_n})$ as a trajectory, where $(o_1, ..., o_n)$ is a permutation of $(1, ..., n)$. The transform starts with the removal of the observation located at x_{o_1}, and the prediction will be performed for $f_{x_{o_1}}$, followed by the update step for relevant quantities (observation values and the integral values) of its neighbours. Then we perform the predict and update steps for x_{o_2} and so on. After the full algorithm iteration, a sequence of detail coefficients $\{d_{o_1}, d_{o_2}..., d_{o_{n-2}}\}$ will be obtained. Following this idea, we generate P different trajectories, denoted as $\{T_p\}_{p=1}^{P}$. Then for *each* p-th trajectory, we will obtain a detail coefficient sequence $\underline{d}^p$. As a result of the transform across all P trajectories, we will have a set of detail coefficients $\{d_i^p\}_{p\in\{1,...,P\}, i\in\{1,...n\}}$. The main difference when compared with the NDWT is that NLT will give several detail coefficients for each location across artificial levels, as opposed to exactly one, see Knight and Nason (2009). Hence, in NLT case, the problem 'to fill in the gap' of the decimation can be expressed as to select P 'well-behaved' trajectories. As Knight and Nason (2009) pointed out, a proper selection of trajectories should have the ability to explore the trajectory space consisting of all $(n!)$ permutations of $\{1, ..., n\}$.

Once we obtain the detail coefficients, the denoising strategy assuming additive noise is as follows. For the p-th trajectory T_p, we threshold the detail coefficient sequence $\underline{d}^p$ by a chosen shrinkage method (in our work, empirical Bayes thresholding). Then the inverse transform will be performed to obtain an estimate $\hat{g}^{(p)}$ of the function g. Denote $\hat{g}^{(p)}(x_i)$ as the estimate of the observation g_i at location x_i, obtained by the p-th trajectory. Then the NLT-based averaged estimator of g can be written as

$$\hat{\bar{g}}(x_i) = \frac{1}{P}\sum_{p=1}^{P} \hat{g}^{(p)}(x_i),$$

for all $i \in \{1, ..., n\}$. This aggregate estimator has the ability to improve performance of individual (path) estimators as well as lower their variability (Knight and Nason; 2009).

5.2 NLT for our Proposed Algorithms

In this section, we will discuss how to introduce a non-decimated transform stemming from a selection of our proposed algorithms that have shown good performance in the previous chapters. We will also put emphasis on the biorthogonal Haar construction, since as we will see later, it can provide different trajectory choices.

Similarly to the previous chapters, suppose we have a graph $G = (\mathcal{V}, \mathcal{E})$, and a function $g^{\mathcal{E}} : \mathcal{E} \longrightarrow \mathbb{R}$, with $\{f_k^{\mathcal{E}}\}_{k=1}^m$ the set of noisy observations on the edge set $\{e_k\}_{k=1}^m$. Then we naturally follow the idea from Knight and Nason (2009), and generate P trajectories $\{T_p\}_{p=1}^P$, where $T_p = (e_{o_1}, ..., e_{o_m})$ and $(o_1, ..., o_m)$ is a permutation of the order sequence $(1, ..., m)$. For each trajectory, let us say, the p-th one T_p, a detail coefficient sequence $\underline{d}^{\mathcal{E},(p)}$ can be obtained by any of (as long as it is the same) the transforms we have proposed in previous chapters. Then similarly, an estimate $\hat{g}^{\mathcal{E},(p)}$ will be obtained. Hence, the average estimator can be written as

$$\hat{\bar{g}}_k^{\mathcal{E}} = \frac{1}{P} \sum_{p=1}^{P} \hat{g}_k^{\mathcal{E},(p)},$$

for $k \in \{1, ..., m\}$. In the work from Knight and Nason (2009), a genetic algorithm (Lucasius and Kateman; 1993, 1994) is applied to generate 'well-behaved' trajectories, which are those likely to have low variations. In the context of applying the LOCAAT algorithm, for river network applications, Park et al. (2022) suggest to perform several permutations within each stream cluster for each trajectory (see Section 5.4 later for the discussion on stream clusters). In this work, we assume that there is no prior information on the underlying function, thus, the trajectories are chosen by random permutations, but note that further improvements may be possible as suggested by Knight and Nason (2009) and Park et al. (2022).

5.2.1 Non-decimated 'Lazy' Lifting Transform

Notice that for the NLT, there is a possible stability issue caused by the use of a random permutation $(o_1, ..., o_m)$. Recall that (see Section 2.3.3 or Jansen et al. (2009)) the

transform is well-bounded if the size of update coefficients is in $(0, \frac{1}{2}]$. This condition is always satisfied if the edge associated with the minimal scaling function integral value is removed, and the update weights are constructed via the minimum norm solution, see Jansen et al. (2009) for a detailed discussion. However, it is possible that an edge with relative large (compared with its neighbouring edges) scaling function integral value is chosen for removal by a random trajectory, which might cause an instability issue. Consider for example that e_{k_r} is removed at stage-r, and it has only one neighbouring edge, denote it as e_s. If $I^{\mathcal{E}}_{k_r,r} > I^{\mathcal{E}}_{s,r}$ holds, where $I^{\mathcal{E}}_{k_r,r}$ and $I^{\mathcal{E}}_{s,r}$ are the stage-r scaling function integral values associated with e_{k_r} and e_s, respectively, then the update coefficient obtained by the minimum norm solution is

$$\begin{aligned} b^{\mathcal{E}}_{s,r} &= \frac{I^{\mathcal{E}}_{k_r,r} I^{\mathcal{E}}_{s,r-1}}{(I^{\mathcal{E}}_{s,r-1})^2} \\ &= \frac{I^{\mathcal{E}}_{k_r,r}}{I^{\mathcal{E}}_{s,r-1}}. \end{aligned}$$

Since the prediction weight is $a^{\mathcal{E}}_{s,r} = 1$ for the one neighbour case, we have $I^{\mathcal{E}}_{s,r-1} = I^{\mathcal{E}}_{s,r} + I^{\mathcal{E}}_{k_r,r}$, which yields

$$\begin{aligned} b^{\mathcal{E}}_{s,r} &= \frac{I^{\mathcal{E}}_{k_r,r}}{I^{\mathcal{E}}_{s,r} + I^{\mathcal{E}}_{k_r,r}} \\ &> \frac{I^{\mathcal{E}}_{k_r,r}}{I^{\mathcal{E}}_{k_r,r} + I^{\mathcal{E}}_{k_r,r}} \\ &= \frac{1}{2}. \end{aligned}$$

When the detail coefficient $d^{\mathcal{E}}_{k_r}$ obtained at stage-r is not small (for example, if there is a discontinuity between the values on e_{k_r} and e_s), then it is likely that after update step, the scaling coefficient $c^{\mathcal{E}}_{s,r-1}$ will be dominated by $c^{\mathcal{E}}_{k_r,r}$ (see equation (2.3.19)), which normally leads to large overlap between wavelet functions (Jansen and Bultheel; 1998) and may cause some unacceptable artifacts (Jansen and Oonincx; 2005). Although the artifacts will be 'averaged out' through the non-decimated wavelet transform (see Coifman and Donoho (1995) and Knight and Nason (2009)), we conjecture that the denosing performance could be further improved if such instability issues are avoided.

The 'lazy lifting' approach introduced in chapter can be considered as a way to solve this issue. The key is that the lazy lifting itself has natural randomness embedded in the trajectory choice when scaling function integral values equal one. Hence, instead of generating random trajectories, we propose to perform several lazy lifting transforms (with Kronecker deltas as the initial primal scaling functions, see in Chapter 2, 3 and 4). We then record the average of their denoised edge-functions as the NLT estimator.

5.2.2 Non-decimated Biorthogonal Haar Transform

Recall from Chapter 2.3.3 that only one of the neighbouring edges is chosen in the prediction and update steps through the biorthogonal Haar E-LOCAAT. This gives us more options to introduce the randomness into different trajectories.

We consider choosing the removal order (the trajectories $\{T_p\}_{p=1}^{P}$) randomly as proposed in Knight and Nason (2009). Recall that for biorthogonal Haar E-LOCAAT, the one-step algorithm as proposed can be expressed as follows.

1. At stage-r, we remove the edge with the shortest length, denoted as e_{k_r}.

2. Among the neighbourhood of the chosen edge, the edge associated to the shortest length will be chosen for performing the prediction. Then at stage-r, the prediction is

$$d_{k_r}^{\Gamma} = c_{k_r,r}^{\Gamma} - c_{s,r}^{\Gamma},$$

   where Γ is the metrized graph for the original graph G, and $s = \arg\min_{t:v_t \in \mathcal{N}_{k_r,r}^{\mathcal{E}}} \{\ell_{t,r}\}$.

3. The value $c_{s,r}^{\Gamma}$ will then be updated by

$$c_{s,r-1}^{\Gamma} = c_{s,r}^{\Gamma} + b_{s,r}^{\Gamma} d_{k_r}^{\Gamma},$$

   where the update weight $b_{s,r}^{\Gamma}$ is obtained by

$$b_{s,r}^{\Gamma} = \frac{\ell_{k_r,r}}{\ell_{k_r,r} + \ell_{s,r}}.$$

4. Relink (see Section 3) and then iterate the steps to obtain the detail coefficients.

We can see that the stability issue comes up if $\ell_{k_r,r} > \ell_{s,r}$. On the other hand, choosing a neighbouring edge e_s with $\ell_{s,r} \gg \ell_{k_r,r}$ will result in an update weight close to zero. Recall that the MRA framework for biorthogonal Haar E-LOCAAT can be expressed as

$$\begin{aligned}\varphi^{\Gamma}_{s,r-1} &= \varphi^{\Gamma}_{s,r} + \varphi^{\Gamma}_{k_r,r},\\ \psi^{\Gamma}_{k_r,r} &= \varphi^{\Gamma}_{k_r,r} - b^{\Gamma}_{s,r}\varphi^{\Gamma}_{s,r-1}.\end{aligned}$$

Note that the primal wavelets are generated by the update steps, see Sweldens (1998). Hence, if $b^{\Gamma}_{s,r}$ is close to zero, then intuitively $\psi^{\Gamma}_{k_r,r}$ almost locates in the vector space spanned by $\varphi^{\Gamma}_{k_r,r}$, meanwhile the detail coefficient $d^{\Gamma}_{k_r}$ is still obtained by the difference between scaling coefficients, $c^{\Gamma}_{k_r,r} - c^{\Gamma}_{s,r}$. Recall that in the initial function expansion form, the contribution for e_{k_m} is given by $c^{\Gamma}_{k_m,m}\varphi^{\Gamma}_{k_m,m}$, while at stage-$r$, it is given by $d^{\Gamma}_{k_r,r}\psi^{\Gamma}_{k_r,r}$. Consider the situation that the coefficient and scaling function associated with e_{k_r} have not been updated for any stage-r', where $r' > r$. Then if $b^{\Gamma}_{s,r}$ is close to zero, we will have that $\psi^{\Gamma}_{k_r,r}$ is approximately equal to $\varphi^{\Gamma}_{k_r,r}$, however, $d^{\Gamma}_{k_r}$ is not typically equal to $c^{\Gamma}_{k_r,r}$ since $d^{\Gamma}_{k_r}$ also highly depends on the value of $c^{\Gamma}_{s,r}$. This suggests a loss of the local information for the function expansion. As a result, for biorthogonal Haar E-LOCAAT, a desirable design is to construct the update weight with a not negligible size, but bounded by $\frac{1}{2}$. Then we propose the designated neighbour $e_s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}$ to be chosen by the criterion

$$s = \underset{s:\, e_s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}}{\arg\min}\{\ell_{s,r} | \ell_{s,r} > \ell_{k_r,r}\}.$$

If this criterion cannot be satisfied, the edge e_s will be chosen by

$$s = \underset{s:\, e_s \in \mathcal{N}^{\mathcal{E}}_{k_r,r}}{\arg\max}\{\ell_{s,r}\}, \tag{5.2.1}$$

in which the update weight will fall in the interval $(\frac{1}{2}, 1)$, but closer to $\frac{1}{2}$ than any other choice of the neighbouring edges.

We will also test a completely random biorthogonal Haar NLT algorithm, in which *both* the removal edge and the neighbour used for prediction are chosen randomly.

5.3 Simulation Study for Denoising Performance

In this section, as we did in previous chapters, a simulation study and the associated results to the non-decimated version of our algorithms will be presented. Note that both NLT and NDWT are over-determined transforms and they do not have an unique inverse transform. Hence, implementing sparsity plots and condition numbers for the overall NLT transform is of no benefit. Our main focus in this simulation is the denoising performance of our proposed NLT variants, by means of the AMSE value. Recall that our averaged estimator has the form

$$\hat{\bar{g}}_k^{\mathcal{E}} = \frac{1}{P}\sum_{p=1}^{P} \hat{g}_k^{\mathcal{E},(p)},$$

for $k \in \{1, ..., m\}$, where $\{\hat{g}_k^{\mathcal{E},(p)}\}_{k=1}^m$ is the denoised version of the observed data $\{f_k^{\mathcal{E},\text{obs}}\}_{k=1}^m$ corresponding to the p-th removal trajectory. The AMSE is obtained by simulating $R = 100$ noise structures over $Q = 50$ different networks

$$\text{AMSE} = (QRm)^{-1}\sum_{q=1}^{Q}\sum_{r=1}^{R}\sum_{k=1}^{m}\left(\hat{\bar{g}}_{k,q,r}^{\mathcal{E}} - g_{k,q}^{\mathcal{E}}\right)^2,$$

For the simulation study, we choose the algorithms from the previous chapters that displayed best results. Specifically, the non-decimated versions of LG-LOCAAT that will be tested are 'LG-Aid-c' and 'LG-Did-c'. Similarly, the non-decimated versions of 'E-Lil-nwu', 'E-Did-nwu', and 'E-Dil-nwu' in E-LOCAAT will also be implemented. For Laplacian-LOCAAT, 'EL-SC-D' and 'LGL-SC-D' will be chosen. As we discussed before, the stabilised version of all these methods with (acronyms using 'D') as the choice for the initial integral values and biorthogonal Haar NLT will be presented as special cases.

For the benefit of the reader, the acronyms and their meanings can be found in Tables 2.1, 3.1, and 4.1.

The algorithms whose acronyms end in the term 'stable' are the ones introduced in Section 5.2.1. For the biorthogonal Haar construction, the algorithms that end in the term 'random' indicate those with stable prediction as discussed in Section 5.2.2, while those labelled with 'mix.random' are the ones that include randomness on both split and prediction steps.

5.3.1 Denoising Pointwise Functions

According to Tables 5.1-5.12, we can see that all of the non-decimated lifting algorithms give more competitive results than those using one trajectory, albeit carefully chosen as described in previous chapters. The only exception is the non-decimated biorthogonal Haar algorithm for Heavisine function. This indicates that biorthogonal Haar construction does not work well for functions that display relative high variations. However, for g_1 and Blocks function, the non-decimated biorthogonal Haar construction is still the most competitive. In addition, we see that if we only introduce the randomness into the split step only, the non-decimated biorthogonal Haar performs better than the one with randomnesses on both split and prediction step, which indicates that the magnitude of the update coefficients does matter.

Contrary to our conjecture, the NLT algorithm with pruned trajectories for improved stability does not improve the overall denoising performance, except for Heavisine function.

Overall, we can see that non-decimated lifting has the power to improve the network edge denoising performance. Namely, for example, the AMSE for Blocks function denoising via bio-Haar-random (30) in Table 5.7 has a 26.4% drop compared with the one-trajectory bio-Haar algorithm. According to Tables 5.1-5.12, we can see that the algorithms with $P = 30$ always performs better than those with $P = 10$. In conjunction with the results in Knight and Nason (2009), we conjecture that increasing trajectories has a high chance to improve the denoising performance in terms of AMSE.

AMSE$\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Aid-c (10)	46 (13)	86 (27)	68 (19)	58 (13)	174 (46)	34 (8)
LG-Aid-c (30)	**42 (12)**	79 (25)	62 (18)	**53 (12)**	**164 (41)**	**31 (7)**
LG-Did-c (10)	46 (12)	84 (26)	67 (18)	58 (13)	188 (49)	34 (8)
LG-Did-c (30)	**42 (12)**	**77 (23)**	**61 (17)**	**53 (12)**	178 (46)	**31 (7)**
LG-Did-c-stable (10)	50 (14)	88 (28)	70 (20)	61 (14)	181 (49)	37 (9)
LG-Did-c-stable (30)	49 (13)	85 (27)	68 (19)	59 (13)	176 (46)	36 (9)
SNR=5						
LG-Aid-c (10)	17 (5)	34 (11)	26 (7)	27 (7)	127 (41)	16 (3)
LG-Aid-c (30)	**15 (4)**	31 (10)	**24 (6)**	**24 (6)**	**120 (37)**	**15 (3)**
LG-Did-c (10)	16 (5)	33 (10)	26 (7)	26 (7)	140 (46)	16 (3)
LG-Did-c (30)	**15 (4)**	**30 (9)**	**24 (6)**	**24 (6)**	133 (42)	**15 (3)**
LG-Did-c-stable (10)	18 (5)	33 (11)	27 (8)	28 (8)	134 (45)	17 (4)
LG-Did-c-stable (30)	17 (5)	32 (10)	26 (7)	27 (7)	131 (43)	17 (4)
SNR=7						
LG-Aid-c (10)	8 (2)	18 (6)	14 (4)	16 (4)	113 (39)	**10 (2)**
LG-Aid-c (30)	**7 (2)**	17 (6)	**13 (3)**	**14 (4)**	**107 (36)**	**10 (2)**
LG-Did-c (10)	8 (2)	18 (6)	14 (3)	16 (4)	125 (45)	**10 (2)**
LG-Did-c (30)	**7 (2)**	**16 (5)**	**13 (3)**	**14 (4)**	119 (41)	**10 (2)**
LG-Did-c-stable (10)	9 (2)	17 (5)	15 (4)	17 (5)	118 (44)	11 (2)
LG-Did-c-stable (30)	8 (2)	17 (5)	14 (4)	17 (5)	115 (42)	11 (2)

Table 5.1: The AMSE table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Aid-c (10)	32	47	41	36	48	26
LG-Aid-c (30)	30	**42**	**37**	**32**	40	24
LG-Did-c (10)	32	47	41	36	47	26
LG-Did-c (30)	**29**	**42**	**37**	**32**	**39**	**23**
LG-Did-c-stable (10)	36	50	44	37	47	29
LG-Did-c-stable (30)	35	47	42	36	44	29
SNR=5						
LG-Aid-c (10)	13	20	17	16	24	11
LG-Aid-c (30)	**12**	**17**	**15**	**14**	**17**	**10**
LG-Did-c (10)	13	19	17	15	24	11
LG-Did-c (30)	**12**	**17**	**15**	**14**	**17**	**10**
LG-Did-c-stable (10)	14	20	17	16	21	11
LG-Did-c-stable (30)	14	19	17	15	18	11
SNR=7						
LG-Aid-c (10)	7	11	9	9	16	6
LG-Aid-c (30)	**6**	10	**8**	**8**	**10**	**5**
LG-Did-c (10)	7	11	9	9	16	6
LG-Did-c (30)	**6**	**9**	**8**	**8**	**10**	**5**
LG-Did-c-stable (10)	7	11	9	9	14	6
LG-Did-c-stable (30)	7	11	9	**8**	**10**	6

Table 5.2: The variance table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

bias$^2 \times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Aid-c (10)	13	39	26	22	126	**8**
LG-Aid-c (30)	**12**	37	25	**21**	**124**	**8**
LG-Did-c (10)	13	38	26	22	141	**8**
LG-Did-c (30)	**12**	**35**	**24**	**21**	139	**8**
LG-Did-c-stable (10)	14	38	27	24	134	**8**
LG-Did-c-stable (30)	14	37	26	23	133	**8**
SNR=5						
LG-Aid-c (10)	**3**	14	9	11	104	6
LG-Aid-c (30)	**3**	**13**	9	**10**	**103**	**5**
LG-Did-c (10)	**3**	**13**	9	11	117	6
LG-Did-c (30)	**3**	**13**	**8**	**10**	116	6
LG-Did-c-stable (10)	**3**	**13**	10	13	112	6
LG-Did-c-stable (30)	**3**	**13**	10	13	113	6
SNR=7						
LG-Aid-c (10)	**1**	7	**5**	**7**	97	**4**
LG-Aid-c (30)	**1**	7	**5**	**7**	**96**	**4**
LG-Did-c (10)	**1**	7	**5**	**7**	109	**4**
LG-Did-c (30)	**1**	7	**5**	**7**	109	**4**
LG-Did-c-stable (10)	**1**	7	**5**	8	105	5
LG-Did-c-stable (30)	**1**	**6**	**5**	8	105	5

Table 5.3: The squared bias table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

AMSE$\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
E-Did-nwu (10)	48 (13)	88 (28)	68 (18)	58 (13)	207 (56)	35 (8)
E-Did-nwu (30)	**44 (12)**	**80 (25)**	**62 (17)**	**53 (12)**	196 (51)	**32 (7)**
E-Did-nwu-stable (10)	50 (14)	89 (29)	70 (20)	60 (14)	185 (50)	36 (9)
E-Did-nwu-stable (30)	48 (13)	86 (27)	67 (19)	58 (13)	**180 (48)**	35 (8)
E-Lil-nwu (10)	50 (14)	98 (35)	73 (21)	62 (15)	212 (55)	37 (8)
E-Lil-nwu (30)	46 (13)	90 (32)	66 (19)	56 (13)	200 (50)	34 (8)
SNR=5						
E-Did-nwu (10)	17 (5)	34 (11)	26 (7)	26 (6)	158 (53)	17 (4)
E-Did-nwu (30)	**16 (5)**	**31 (10)**	**24 (6)**	**24 (6)**	150 (49)	**16 (3)**
E-Did-nwu-stable (10)	18 (5)	34 (11)	27 (7)	28 (8)	136 (46)	18 (4)
E-Did-nwu-stable (30)	17 (5)	32 (10)	26 (7)	27 (7)	**133 (44)**	17 (4)
E-Lil-nwu (10)	18 (5)	39 (15)	28 (8)	28 (7)	161 (51)	18 (4)
E-Lil-nwu (30)	17 (5)	35 (14)	25 (7)	26 (7)	152 (45)	17 (4)
SNR=7						
E-Did-nwu (10)	**8 (2)**	18 (6)	14 (3)	16 (4)	142 (53)	11 (2)
E-Did-nwu (30)	**8 (2)**	**17 (5)**	**13 (3)**	**14 (4)**	135 (48)	**10 (2)**
E-Did-nwu-stable (10)	9 (2)	18 (6)	15 (4)	17 (5)	121 (45)	11 (2)
E-Did-nwu-stable (30)	**8 (2)**	**17 (5)**	14 (4)	16 (4)	**117 (43)**	11 (2)
E-Lil-nwu (10)	9 (2)	21 (9)	15 (4)	17 (5)	145 (49)	12 (2)
E-Lil-nwu (30)	**8 (2)**	19 (8)	14 (4)	15 (4)	137 (44)	11 (2)

Table 5.4: The AMSE table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
E-Did-nwu (10)	33	48	42	37	48	26
E-Did-nwu (30)	**30**	**42**	**37**	**33**	**39**	**24**
E-Did-nwu-stable (10)	36	50	44	37	49	28
E-Did-nwu-stable (30)	34	47	41	36	43	27
E-Lil-nwu (10)	35	50	44	38	51	28
E-Lil-nwu (30)	31	44	39	34	40	25
SNR=5						
E-Did-nwu (10)	14	20	17	16	25	11
E-Did-nwu (30)	**13**	**18**	**16**	**14**	**17**	**10**
E-Did-nwu-stable (10)	15	21	18	16	22	11
E-Did-nwu-stable (30)	14	19	17	15	18	11
E-Lil-nwu (10)	14	21	18	17	26	12
E-Lil-nwu (30)	**13**	**18**	**16**	15	18	**10**
SNR=7						
E-Did-nwu (10)	**7**	11	10	9	18	7
E-Did-nwu (30)	**7**	**10**	**9**	**8**	**11**	**6**
E-Did-nwu-stable (10)	**7**	11	10	9	14	7
E-Did-nwu-stable (30)	**7**	**10**	**9**	9	**11**	**6**
E-Lil-nwu (10)	8	12	10	10	19	7
E-Lil-nwu (30)	**7**	**10**	**9**	**8**	**11**	**6**

Table 5.5: The variance table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

bias$^2 \times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
E-Did-nwu (10)	15	40	26	21	159	9
E-Did-nwu (30)	**14**	**38**	**24**	**20**	157	9
E-Did-nwu-stable (10)	**14**	39	26	23	**137**	**8**
E-Did-nwu-stable (30)	**14**	39	26	23	**137**	**8**
E-Lil-nwu (10)	16	48	29	24	161	9
E-Lil-nwu (30)	15	46	27	23	160	9
SNR=5						
E-Did-nwu (10)	**3**	14	9	**10**	133	**6**
E-Did-nwu (30)	**3**	14	**8**	**10**	133	**6**
E-Did-nwu-stable (10)	**3**	**13**	9	12	**114**	**6**
E-Did-nwu-stable (30)	**3**	**13**	9	12	**114**	**6**
E-Lil-nwu (10)	4	18	10	12	135	**6**
E-Lil-nwu (30)	4	17	9	11	134	**6**
SNR=7						
E-Did-nwu (10)	**1**	**7**	5	**6**	125	**4**
E-Did-nwu (30)	**1**	**7**	**4**	**6**	124	**4**
E-Did-nwu-stable (10)	**1**	**7**	5	8	**107**	5
E-Did-nwu-stable (30)	**1**	**7**	5	7	**107**	5
E-Lil-nwu (10)	**1**	9	5	7	126	5
E-Lil-nwu (30)	**1**	9	5	7	126	**4**

Table 5.6: The squared bias table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

AMSE$\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
bio-Haar-random (10)	41 (13)	70 (24)	84 (27)	77 (20)	326 (73)	45 (9)
bio-Haar-random (30)	**37 (12)**	**64 (23)**	**76 (24)**	**69 (18)**	313 (67)	**40 (9)**
bio-Haar-mix.random (10)	73 (22)	88 (30)	114 (34)	88 (26)	324 (62)	84 (24)
bio-Haar-mix.random (30)	61 (17)	80 (26)	100 (30)	77 (21)	**307 (54)**	67 (16)
SNR=5						
bio-Haar-random (10)	13 (3)	24 (7)	35 (12)	37 (12)	288 (76)	23 (5)
bio-Haar-random (30)	**12 (3)**	**22 (6)**	**32 (11)**	**34 (10)**	**277 (68)**	**21 (5)**
bio-Haar-mix.random (10)	30 (11)	37 (13)	57 (21)	49 (18)	301 (65)	52 (19)
bio-Haar-mix.random (30)	23 (6)	33 (11)	48 (17)	41 (13)	285 (56)	38 (11)
SNR=7						
bio-Haar-random (10)	**6 (2)**	12 (3)	20 (8)	24 (9)	276 (77)	16 (3)
bio-Haar-random (30)	**6 (2)**	**11 (3)**	**18 (7)**	**21 (8)**	**265 (69)**	**14 (3)**
bio-Haar-mix.random (10)	21 (9)	22 (9)	39 (17)	36 (16)	294 (66)	41 (18)
bio-Haar-mix.random (30)	15 (5)	19 (7)	31 (13)	29 (11)	278 (57)	19 (9)

Table 5.7: The AMSE table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
bio-Haar-random (10)	35	46	46	42	47	31
bio-Haar-random (30)	**31**	**41**	**40**	**36**	34	**26**
bio-Haar-mix.random (10)	51	49	54	49	47	55
bio-Haar-mix.random (30)	41	**41**	41	38	**30**	39
SNR=5						
bio-Haar-random (10)	12	17	19	19	28	14
bio-Haar-random (30)	**11**	**16**	**17**	**16**	17	**12**
bio-Haar-mix.random (10)	23	22	28	26	32	32
bio-Haar-mix.random (30)	16	17	19	18	**16**	19
SNR=7						
bio-Haar-random (10)	6	9	11	11	23	8
bio-Haar-random (30)	**5**	**8**	**9**	**9**	**11**	**7**
bio-Haar-mix.random (10)	15	13	19	18	28	25
bio-Haar-mix.random (30)	10	10	12	11	12	13

Table 5.8: The variance table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

bias$^2 \times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
bio-Haar-random (10)	6	24	38	35	278	**14**
bio-Haar-random (30)	**5**	**23**	**36**	**33**	279	**14**
bio-Haar-mix.random (10)	21	39	60	39	**277**	29
bio-Haar-mix.random (30)	21	39	58	39	**277**	28
SNR=5						
bio-Haar-random (10)	**1**	**7**	**15**	19	**260**	**9**
bio-Haar-random (30)	**1**	**7**	**15**	**18**	**260**	**9**
bio-Haar-mix.random (10)	7	15	29	23	269	19
bio-Haar-mix.random (30)	7	15	28	23	269	19
SNR=7						
bio-Haar-random (10)	**0**	**3**	**9**	13	**253**	**7**
bio-Haar-random (30)	**0**	**3**	**9**	**12**	254	**7**
bio-Haar-mix.random (10)	5	9	20	18	266	16
bio-Haar-mix.random (30)	5	9	19	18	266	16

Table 5.9: The squared bias table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

AMSE$\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
EL-SC-D (10)	47 (13)	87 (27)	68 (18)	59 (13)	208 (55)	35 (8)
EL-SC-D (30)	43 (12)	79 (25)	**62 (17)**	**53 (12)**	197 (50)	**32 (7)**
EL-SC-D-stable (10)	50 (14)	89 (29)	70 (20)	61 (14)	186 (50)	37 (9)
EL-SC-D-stable (30)	48 (13)	86 (27)	67 (19)	59 (13)	**182 (49)**	36 (8)
LGL-SC-D (10)	46 (13)	85 (27)	69 (19)	60 (13)	203 (55)	35 (8)
LGL-SC-D (30)	**42 (12)**	**78 (24)**	63 (17)	55 (13)	192 (51)	**32 (8)**
LGL-SC-D-stable (10)	50 (14)	89 (29)	72 (21)	63 (15)	190 (52)	37 (9)
LGL-SC-D-stable (30)	49 (13)	86 (27)	69 (20)	61 (14)	187 (50)	37 (8)
SNR=5						
EL-SC-D (10)	17 (5)	34 (11)	26 (7)	27 (6)	160 (53)	17 (4)
EL-SC-D (30)	**15 (4)**	**31 (10)**	**24 (6)**	**24 (6)**	151 (48)	**16 (3)**
EL-SC-D-stable (10)	50 (14)	89 (29)	70 (20)	61 (14)	186 (50)	37 (9)
EL-SC-D-stable (30)	48 (13)	86 (27)	67 (19)	59 (13)	182 (49)	36 (8)
LGL-SC-D (10)	17 (5)	34 (11)	27 (7)	27 (7)	154 (53)	17 (4)
LGL-SC-D (30)	**15 (4)**	**31 (9)**	25 (6)	25 (6)	147 (49)	**16 (3)**
LGL-SC-D-stable (10)	18 (5)	34 (11)	28 (8)	29 (8)	142 (48)	18 (4)
LGL-SC-D-stable (30)	17 (5)	33 (10)	27 (7)	28 (8)	**139 (46)**	17 (4)
SNR=7						
EL-SC-D (10)	8 (2)	19 (6)	14 (3)	16 (4)	144 (53)	11 (2)
EL-SC-D (30)	8 (2)	**17 (6)**	**13 (3)**	**14 (4)**	136 (48)	**10 (2)**
EL-SC-D-stable (10)	9 (2)	18 (6)	15 (4)	17 (5)	121 (45)	11 (2)
EL-SC-D-stable (30)	8 (2)	**17 (5)**	14 (4)	16 (5)	**118 (43)**	11 (2)
LGL-SC-D (10)	8 (2)	18 (6)	15 (4)	17 (5)	139 (52)	11 (2)
LGL-SC-D (30)	**7 (2)**	**17 (5)**	**13 (3)**	15 (4)	132 (48)	**10 (2)**
LGL-SC-D-stable (10)	9 (2)	18 (6)	15 (4)	18 (6)	127 (48)	11 (2)
LGL-SC-D-stable (30)	8 (2)	**17 (6)**	15 (4)	17 (5)	124 (46)	11 (2)

Table 5.10: The AMSE table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
EL-SC-D (10)	32	47	42	36	48	26
EL-SC-D (30)	**29**	**41**	**37**	**32**	38	**23**
EL-SC-D-stable (10)	36	50	44	37	48	28
EL-SC-D-stable (30)	34	47	42	36	43	27
LGL-SC-D (10)	32	47	41	36	48	26
LGL-SC-D (30)	**29**	**41**	**37**	**32**	**38**	**23**
LGL-SC-D-stable (10)	36	50	44	38	47	29
LGL-SC-D-stable (30)	35	47	42	36	43	28
SNR=5						
EL-SC-D (10)	14	20	17	16	25	11
EL-SC-D (30)	**12**	**17**	**15**	14	**17**	**10**
EL-SC-D-stable (10)	36	50	44	37	48	28
EL-SC-D-stable (30)	34	47	42	36	43	27
LGL-SC-D (10)	13	20	17	16	24	11
LGL-SC-D (30)	**12**	**17**	**15**	**13**	**17**	**10**
LGL-SC-D-stable (10)	14	20	18	16	22	11
LGL-SC-D-stable (30)	14	19	17	15	18	11
SNR=7						
EL-SC-D (10)	7	11	10	9	18	**6**
EL-SC-D (30)	7	10	9	**8**	11	**6**
EL-SC-D-stable (10)	7	11	10	9	14	**6**
EL-SC-D-stable (30)	7	10	9	**8**	11	**6**
LGL-SC-D (10)	7	11	9	9	17	**6**
LGL-SC-D (30)	**6**	**9**	**8**	**8**	**10**	**6**
LGL-SC-D-stable (10)	7	11	10	9	14	**6**
LGL-SC-D-stable (30)	7	10	9	**8**	**10**	**6**

Table 5.11: The variance table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

bias$^2 \times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
EL-SC-D (10)	14	40	26	22	160	9
EL-SC-D (30)	**13**	37	**25**	**21**	158	9
EL-SC-D-stable (10)	14	39	26	23	**137**	**8**
EL-SC-D-stable (30)	14	39	26	23	139	**8**
LGL-SC-D (10)	14	39	27	24	155	9
LGL-SC-D (30)	**13**	**36**	26	23	154	9
LGL-SC-D-stable (10)	14	39	28	25	143	**8**
LGL-SC-D-stable (30)	14	39	27	25	144	**8**
SNR=5						
EL-SC-D (10)	**3**	14	9	11	135	**6**
EL-SC-D (30)	**3**	14	**8**	**10**	134	**6**
EL-SC-D-stable (10)	14	39	26	23	137	8
EL-SC-D-stable (30)	14	39	26	23	139	8
LGL-SC-D (10)	4	14	10	12	130	**6**
LGL-SC-D (30)	**3**	**13**	9	11	130	**6**
LGL-SC-D-stable (10)	4	14	10	14	**121**	**6**
LGL-SC-D-stable (30)	**3**	14	10	13	**121**	**6**
SNR=7						
EL-SC-D (10)	**1**	8	5	7	126	**4**
EL-SC-D (30)	**1**	**7**	**4**	**6**	126	**4**
EL-SC-D-stable (10)	**1**	**7**	5	8	**107**	5
EL-SC-D-stable (30)	**1**	**7**	5	8	108	5
LGL-SC-D (10)	**1**	**7**	5	8	122	5
LGL-SC-D (30)	**1**	**7**	5	7	121	5
LGL-SC-D-stable (10)	**1**	**7**	6	9	113	5
LGL-SC-D-stable (30)	**1**	**7**	6	9	113	5

Table 5.12: The squared bias table for different schemes. The test functions are the pointwise ones defined in equation (2.4.1). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

5.3.2 Denoising Edge Averaging Functions

From Table 5.13-5.24, we can see that for the edge averaging functions, similar conclusions can be drawn similar to those in Section 5.3.1.

AMSE$\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Aid-c (10)	42 (11)	87 (25)	65 (18)	60 (14)	190 (46)	34 (8)
LG-Aid-c (30)	38 (11)	79 (23)	60 (17)	54 (13)	178 (42)	**31 (7)**
LG-Did-c (10)	41 (11)	85 (25)	64 (18)	59 (14)	185 (45)	34 (8)
LG-Did-c (30)	**37 (10)**	**78 (23)**	**59 (16)**	**53 (12)**	**174 (41)**	**31 (7)**
LG-Did-c-stable (10)	50 (14)	88 (28)	70 (20)	61 (14)	181 (49)	37 (9)
LG-Did-c-stable (30)	43 (12)	86 (27)	66 (20)	59 (13)	168 (43)	36 (8)
SNR=5						
LG-Aid-c (10)	17 (5)	37 (12)	27 (8)	27 (7)	143 (43)	17 (4)
LG-Aid-c (30)	**15 (4)**	34 (11)	**25 (7)**	25 (6)	135 (39)	**15 (3)**
LG-Did-c (10)	17 (5)	36 (11)	27 (8)	27 (7)	139 (42)	17 (4)
LG-Did-c (30)	**15 (4)**	**33 (10)**	**25 (7)**	**24 (6)**	131 (38)	**15 (3)**
LG-Did-c-stable (10)	18 (5)	37 (14)	29 (9)	28 (7)	127 (41)	18 (4)
LG-Did-c-stable (30)	17 (5)	36 (13)	27 (8)	27 (7)	**124 (38)**	17 (4)
SNR=7						
LG-Aid-c (10)	9 (2)	21 (7)	15 (4)	16 (4)	128 (42)	11 (2)
LG-Aid-c (30)	**8 (2)**	19 (7)	**14 (4)**	15 (4)	121 (38)	**10 (2)**
LG-Did-c (10)	9 (2)	20 (7)	15 (4)	16 (4)	124 (42)	11 (2)
LG-Did-c (30)	**8 (2)**	**18 (6)**	**14 (4)**	**14 (4)**	117 (37)	**10 (2)**
LG-Did-c-stable (10)	9 (3)	21 (9)	16 (5)	17 (5)	113 (40)	11 (2)
LG-Did-c-stable (30)	9 (2)	20 (8)	15 (5)	16 (4)	**110 (38)**	11 (2)

Table 5.13: The AMSE table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Aid-c (10)	30	45	37	36	48	26
LG-Aid-c (30)	27	**39**	**33**	32	**38**	**23**
LG-Did-c (10)	29	44	37	36	48	25
LG-Did-c (30)	**26**	**39**	**33**	32	**38**	**23**
LG-Did-c-stable (10)	36	50	44	**31**	47	29
LG-Did-c-stable (30)	31	44	37	35	43	27
SNR=5						
LG-Aid-c (10)	13	20	16	16	25	11
LG-Aid-c (30)	**11**	**17**	**14**	**14**	**17**	10
LG-Did-c (10)	12	20	16	16	25	11
LG-Did-c (30)	**11**	**17**	**14**	**14**	**17**	**9**
LG-Did-c-stable (10)	13	20	17	16	22	11
LG-Did-c-stable (30)	13	19	16	15	18	11
SNR=7						
LG-Aid-c (10)	7	11	9	9	18	6
LG-Aid-c (30)	**6**	**10**	**8**	**8**	11	**5**
LG-Did-c (10)	7	11	9	9	17	6
LG-Did-c (30)	**6**	**10**	**8**	**8**	**10**	**5**
LG-Did-c-stable (10)	7	11	9	9	14	6
LG-Did-c-stable (30)	7	**10**	9	**8**	**10**	6

Table 5.14: The variance table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

bias$^2 \times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
LG-Aid-c (10)	12	42	28	24	142	9
LG-Aid-c (30)	**11**	40	27	**22**	140	**8**
LG-Did-c (10)	12	41	27	23	137	9
LG-Did-c (30)	**11**	39	**26**	**22**	136	**8**
LG-Did-c-stable (10)	14	**38**	27	24	134	**8**
LG-Did-c-stable (30)	12	43	29	23	**126**	9
SNR=5						
LG-Aid-c (10)	**4**	17	11	**11**	118	**6**
LG-Aid-c (30)	**4**	17	11	**11**	117	**6**
LG-Did-c (10)	**4**	17	11	**11**	114	**6**
LG-Did-c (30)	**4**	**16**	**10**	**11**	114	**6**
LG-Did-c-stable (10)	**4**	17	12	13	**106**	**6**
LG-Did-c-stable (30)	**4**	17	12	12	**106**	**6**
SNR=7						
LG-Aid-c (10)	**2**	10	**6**	**7**	110	5
LG-Aid-c (30)	**2**	**9**	**6**	**7**	110	**4**
LG-Did-c (10)	**2**	**9**	**6**	**7**	107	**4**
LG-Did-c (30)	**2**	**9**	**6**	**7**	107	**4**
LG-Did-c-stable (10)	**2**	10	7	8	**100**	5
LG-Did-c-stable (30)	**2**	10	7	8	**100**	5

Table 5.15: The squared bias table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

AMSE×10^3 (sd×10^3)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
E-Did-nwu (10)	43 (11)	87 (25)	64 (16)	57 (12)	201 (52)	34 (8)
E-Did-nwu (30)	**40 (11)**	**79 (23)**	**59 (16)**	**52 (12)**	190 (48)	**31 (7)**
E-Did-nwu-stable (10)	46 (12)	88 (26)	66 (17)	60 (13)	178 (46)	36 (8)
E-Did-nwu-stable (30)	44 (11)	85 (24)	64 (17)	57 (12)	**174 (45)**	35 (8)
E-Lil-nwu (10)	46 (13)	97 (30)	70 (19)	62 (14)	215 (53)	36 (8)
E-Lil-nwu (30)	42 (12)	89 (27)	64 (18)	56 (13)	203 (48)	33 (8)
SNR=5						
E-Did-nwu (10)	17 (5)	37 (11)	27 (7)	26 (6)	155 (50)	17 (3)
E-Did-nwu (30)	**16 (4)**	**33 (10)**	**24 (6)**	**24 (5)**	148 (46)	**15 (3)**
E-Did-nwu-stable (10)	18 (5)	36 (11)	28 (8)	27 (7)	134 (43)	17 (4)
E-Did-nwu-stable (30)	18 (5)	35 (10)	26 (7)	26 (6)	**131 (41)**	17 (4)
E-Lil-nwu (10)	19 (5)	42 (14)	29 (8)	28 (7)	166 (50)	18 (4)
E-Lil-nwu (30)	17 (5)	38 (13)	26 (7)	25 (6)	157 (45)	16 (4)
SNR=7						
E-Did-nwu (10)	9 (2)	21 (7)	15 (4)	15 (4)	141 (50)	11 (2)
E-Did-nwu (30)	**8 (2)**	**19 (6)**	**13 (3)**	**14 (3)**	134 (46)	**10 (2)**
E-Did-nwu-stable (10)	9 (3)	20 (6)	15 (4)	16 (5)	121 (43)	11 (2)
E-Did-nwu-stable (30)	9 (2)	**19 (6)**	15 (4)	16 (4)	**117 (41)**	11 (2)
E-Lil-nwu (10)	10 (3)	24 (9)	16 (4)	17 (5)	151 (49)	11 (2)
E-Lil-nwu (30)	9 (3)	22 (8)	14 (4)	15 (4)	143 (44)	**10 (2)**

Table 5.16: The AMSE table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
E-Did-nwu (10)	31	45	38	37	48	26
E-Did-nwu (30)	**28**	**40**	**34**	**33**	**38**	**23**
E-Did-nwu-stable (10)	33	47	40	37	47	28
E-Did-nwu-stable (30)	32	44	38	35	42	27
E-Lil-nwu (10)	32	45	40	38	50	27
E-Lil-nwu (30)	29	41	35	**33**	40	25
SNR=5						
E-Did-nwu (10)	13	20	17	16	25	11
E-Did-nwu (30)	**12**	**17**	**15**	**14**	**17**	**10**
E-Did-nwu-stable (10)	14	20	17	16	22	11
E-Did-nwu-stable (30)	13	19	16	15	18	11
E-Lil-nwu (10)	14	21	17	17	26	12
E-Lil-nwu (30)	**12**	18	**15**	**14**	18	**10**
SNR=7						
E-Did-nwu (10)	**7**	11	9	9	17	7
E-Did-nwu (30)	**7**	**10**	**8**	**8**	**10**	**6**
E-Did-nwu-stable (10)	8	11	9	9	14	**6**
E-Did-nwu-stable (30)	**7**	**10**	9	**8**	**10**	**6**
E-Lil-nwu (10)	8	12	10	10	19	7
E-Lil-nwu (30)	**7**	**10**	**8**	**8**	11	**6**

Table 5.17: The variance table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

bias$^2 \times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
E-Did-nwu (10)	13	41	26	21	153	**8**
E-Did-nwu (30)	**12**	**39**	**25**	**20**	152	**8**
E-Did-nwu-stable (10)	13	40	26	23	**131**	**8**
E-Did-nwu-stable (30)	13	40	26	22	132	**8**
E-Lil-nwu (10)	14	51	30	24	165	9
E-Lil-nwu (30)	13	48	28	22	163	9
SNR=5						
E-Did-nwu (10)	**4**	17	10	10	131	**5**
E-Did-nwu (30)	**4**	**16**	**9**	**9**	131	**5**
E-Did-nwu-stable (10)	5	**16**	10	12	**112**	6
E-Did-nwu-stable (30)	**4**	**16**	10	11	113	6
E-Lil-nwu (10)	5	21	11	11	140	6
E-Lil-nwu (30)	5	20	11	11	140	6
SNR=7						
E-Did-nwu (10)	**2**	**9**	**5**	**6**	124	**4**
E-Did-nwu (30)	**2**	**9**	**5**	**6**	124	**4**
E-Did-nwu-stable (10)	**2**	**9**	6	7	**107**	**4**
E-Did-nwu-stable (30)	**2**	**9**	6	7	**107**	**4**
E-Lil-nwu (10)	**2**	12	6	7	132	**4**
E-Lil-nwu (30)	**2**	12	6	7	132	**4**

Table 5.18: The squared bias table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

AMSE$\times 10^3$ (sd$\times 10^3$)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
bio-Haar-random (10)	45 (13)	89 (33)	82 (25)	77 (20)	339 (68)	44 (9)
bio-Haar-random (30)	**41 (13)**	**82 (31)**	**75 (23)**	**70 (18)**	327 (62)	**40 (9)**
bio-Haar-mix.random (10)	76 (22)	110 (36)	110 (32)	88 (25)	334 (56)	84 (24)
bio-Haar-mix.random (30)	64 (17)	100 (32)	96 (27)	77 (20)	**317 (48)**	67 (16)
SNR=5						
bio-Haar-random (10)	15 (4)	33 (12)	36 (13)	38 (11)	306 (72)	23 (5)
bio-Haar-random (30)	**14 (4)**	**31 (11)**	**33 (11)**	**34 (10)**	**295 (65)**	**21 (5)**
bio-Haar-mix.random (10)	33 (12)	51 (21)	58 (22)	50 (18)	313 (60)	51 (19)
bio-Haar-mix.random (30)	26 (7)	45 (18)	49 (17)	41 (13)	297 (51)	38 (11)
SNR=7						
bio-Haar-random (10)	8 (2)	18 (6)	21 (8)	24 (8)	296 (74)	16 (3)
bio-Haar-random (30)	**7 (2)**	**16 (6)**	**19 (7)**	**22 (7)**	**285 (66)**	**14 (3)**
bio-Haar-mix.random (10)	23 (10)	33 (16)	41 (18)	37 (16)	306 (61)	41 (18)
bio-Haar-mix.random (30)	16 (5)	28 (13)	32 (13)	29 (11)	290 (52)	29 (9)

Table 5.19: The AMSE table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
bio-Haar-random (10)	35	48	42	42	46	30
bio-Haar-random (30)	**31**	42	**37**	**36**	33	**26**
bio-Haar-mix.random (10)	51	51	52	49	46	55
bio-Haar-mix.random (30)	40	**41**	39	38	**29**	39
SNR=5						
bio-Haar-random (10)	13	20	19	19	28	14
bio-Haar-random (30)	**12**	**17**	**16**	**16**	**16**	**11**
bio-Haar-mix.random (10)	24	25	28	26	32	32
bio-Haar-mix.random (30)	17	19	19	18	**16**	19
SNR=7						
bio-Haar-random (10)	7	11	11	11	22	8
bio-Haar-random (30)	**6**	**9**	**9**	**9**	**11**	**7**
bio-Haar-mix.random (10)	16	16	20	19	28	25
bio-Haar-mix.random (30)	10	11	12	11	12	13

Table 5.20: The variance table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

bias$^2 \times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
bio-Haar-random (10)	10	41	40	35	293	14
bio-Haar-random (30)	**9**	**40**	**38**	**34**	294	**13**
bio-Haar-mix.random (10)	25	59	59	39	288	29
bio-Haar-mix.random (30)	24	59	58	39	**287**	28
SNR=5						
bio-Haar-random (10)	**2**	14	**17**	19	**278**	10
bio-Haar-random (30)	**2**	**13**	**17**	**18**	279	**9**
bio-Haar-mix.random (10)	9	27	30	23	281	19
bio-Haar-mix.random (30)	9	27	30	24	281	19
SNR=7						
bio-Haar-random (10)	**1**	**7**	11	**13**	**273**	**7**
bio-Haar-random (30)	**1**	**7**	**10**	**13**	274	**7**
bio-Haar-mix.random (10)	6	17	21	18	278	16
bio-Haar-mix.random (30)	6	17	21	18	279	16

Table 5.21: The squared bias table for different schemes. The test functions are theedge averaging ones defined in equation (2.4.2). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

AMSE×10^3 (sd×10^3)	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
EL-SC-D (10)	43 (11)	87 (25)	64 (17)	58 (13)	202 (52)	34 (8)
EL-SC-D (30)	**40 (11)**	**80 (23)**	**59 (16)**	**53 (12)**	191 (47)	**31 (7)**
EL-SC-D-stable (10)	46 (12)	88 (26)	67 (18)	60 (13)	179 (47)	36 (8)
EL-SC-D-stable (30)	44 (11)	85 (25)	65 (17)	58 (12)	**175 (45)**	35 (8)
LGL-SC-D (10)	43 (11)	87 (25)	66 (17)	59 (13)	197 (51)	34 (8)
LGL-SC-D (30)	**40 (11)**	**80 (23)**	60 (16)	54 (12)	187 (47)	32 (7)
LGL-SC-D-stable (10)	47 (13)	90 (27)	68 (18)	62 (14)	184 (48)	37 (9)
LGL-SC-D-stable (30)	45 (12)	86 (25)	66 (18)	60 (13)	180 (46)	36 (8)
SNR=5						
EL-SC-D (10)	18 (5)	37 (11)	27 (7)	26 (6)	157 (50)	16 (3)
EL-SC-D (30)	**16 (4)**	**34 (10)**	**24 (7)**	**24 (6)**	149 (46)	**15 (3)**
EL-SC-D-stable (10)	46 (12)	88 (26)	67 (18)	60 (13)	179 (47)	36 (8)
EL-SC-D-stable (30)	44 (11)	85 (25)	65 (17)	58 (12)	175 (45)	35 (8)
LGL-SC-D (10)	18 (5)	37 (11)	28 (8)	27 (7)	153 (50)	16 (3)
LGL-SC-D (30)	**16 (5)**	**34 (11)**	25 (7)	25 (6)	145 (46)	**15 (3)**
LGL-SC-D-stable (10)	19 (5)	37 (12)	29 (8)	29 (8)	141 (45)	17 (4)
LGL-SC-D-stable (30)	18 (5)	36 (11)	28 (8)	28 (7)	**138 (43)**	17 (4)
SNR=7						
EL-SC-D (10)	9 (2)	21 (7)	15 (4)	16 (4)	143 (50)	**10 (2)**
EL-SC-D (30)	**8 (2)**	**19 (6)**	**14 (3)**	**14 (4)**	136 (46)	**10 (2)**
EL-SC-D-stable (10)	9 (3)	20 (6)	15 (4)	17 (5)	121 (42)	11 (2)
EL-SC-D-stable (30)	9 (2)	**19 (6)**	15 (4)	16 (4)	**118 (40)**	**10 (2)**
LGL-SC-D (10)	9 (3)	21 (7)	15 (4)	16 (4)	139 (50)	11 (2)
LGL-SC-D (30)	**8 (2)**	**19 (6)**	**14 (4)**	15 (4)	132 (46)	**10 (2)**
LGL-SC-D-stable (10)	10 (3)	20 (7)	16 (5)	18 (5)	127 (45)	11 (2)
LGL-SC-D-stable (30)	9 (3)	20 (6)	15 (4)	17 (5)	124 (43)	11 (2)

Table 5.22: The AMSE table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

Variance$\times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
EL-SC-D (10)	30	45	38	36	47	26
EL-SC-D (30)	**27**	**40**	34	**32**	**38**	**23**
EL-SC-D-stable (10)	33	48	40	37	47	28
EL-SC-D-stable (30)	32	45	38	36	42	27
LGL-SC-D (10)	30	45	38	36	47	26
LGL-SC-D (30)	**27**	**40**	**33**	**32**	**38**	**23**
LGL-SC-D-stable (10)	34	47	40	37	46	29
LGL-SC-D-stable (30)	32	45	38	36	42	28
SNR=5						
EL-SC-D (10)	13	20	17	16	24	11
EL-SC-D (30)	12	**17**	**15**	**14**	**17**	**10**
EL-SC-D-stable (10)	33	48	40	37	47	28
EL-SC-D-stable (30)	32	45	38	36	42	27
LGL-SC-D (10)	13	20	16	15	24	11
LGL-SC-D (30)	**11**	**17**	**15**	**14**	**17**	**10**
LGL-SC-D-stable (10)	14	20	17	16	21	11
LGL-SC-D-stable (30)	13	19	16	15	**17**	11
SNR=7						
EL-SC-D (10)	7	11	9	9	17	6
EL-SC-D (30)	7	**10**	**8**	**8**	**10**	6
EL-SC-D-stable (10)	8	11	9	9	14	6
EL-SC-D-stable (30)	7	**10**	9	**8**	**10**	6
LGL-SC-D (10)	7	11	9	9	16	6
LGL-SC-D (30)	**6**	**10**	**8**	**8**	**10**	**5**
LGL-SC-D-stable (10)	8	11	9	9	13	6
LGL-SC-D-stable (30)	7	**10**	9	**8**	**10**	6

Table 5.23: The variance table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

bias$^2 \times 10^3$	g_1	Blocks	Doppler	Bumps	Heavisine	mfc
SNR=3						
EL-SC-D (10)	13	42	27	22	155	**8**
EL-SC-D (30)	**12**	**40**	**25**	**20**	153	**8**
EL-SC-D-stable (10)	13	**40**	27	23	**132**	**8**
EL-SC-D-stable (30)	13	**40**	26	22	133	**8**
LGL-SC-D (10)	13	43	28	23	150	**8**
LGL-SC-D (30)	13	**40**	27	22	149	**8**
LGL-SC-D-stable (10)	13	42	28	25	138	**8**
LGL-SC-D-stable (30)	13	42	28	24	139	**8**
SNR=5						
EL-SC-D (10)	5	17	**10**	**10**	133	6
EL-SC-D (30)	**4**	**16**	**10**	**10**	132	**5**
EL-SC-D-stable (10)	13	40	27	23	132	8
EL-SC-D-stable (30)	13	40	26	22	133	8
LGL-SC-D (10)	5	17	11	12	129	6
LGL-SC-D (30)	5	17	11	11	129	6
LGL-SC-D-stable (10)	5	17	12	13	**120**	6
LGL-SC-D-stable (30)	5	17	12	13	**120**	6
SNR=7						
EL-SC-D (10)	**2**	10	**5**	**6**	126	**4**
EL-SC-D (30)	**2**	**9**	**5**	**6**	126	**4**
EL-SC-D-stable (10)	**2**	**9**	6	8	**107**	**4**
EL-SC-D-stable (30)	2	**9**	6	7	108	**4**
LGL-SC-D (10)	**2**	10	6	7	122	**4**
LGL-SC-D (30)	**2**	**9**	6	7	122	**4**
LGL-SC-D-stable (10)	**2**	**9**	7	9	114	5
LGL-SC-D-stable (30)	**2**	**9**	7	8	114	5

Table 5.24: The squared bias table for different schemes. The test functions are the edge averaging ones defined in equation (2.4.2). The numbers following algorithm types indicate the number of used trajectories, namely $P = 10$ and 30.

5.4 Flow-based Function Denoising

We now investigate our proposed algorithms and their non-decimated versions on the piecewise function introduced by Park et al. (2022) on a simulated river tree network from Gallacher et al. (2017), see Figure 5.1 for a visualisation. The river network contains 80 vertices and 79 edges, and its edge set is separated into seven different clusters. The function is constructed as follows and described by Park et al. (2022). Firstly, the function values for every stream are set as 9. Then we randomly pick a cluster, and the function values of every stream in this cluster will be randomly chosen from $\{12, 15, 18\}$. We

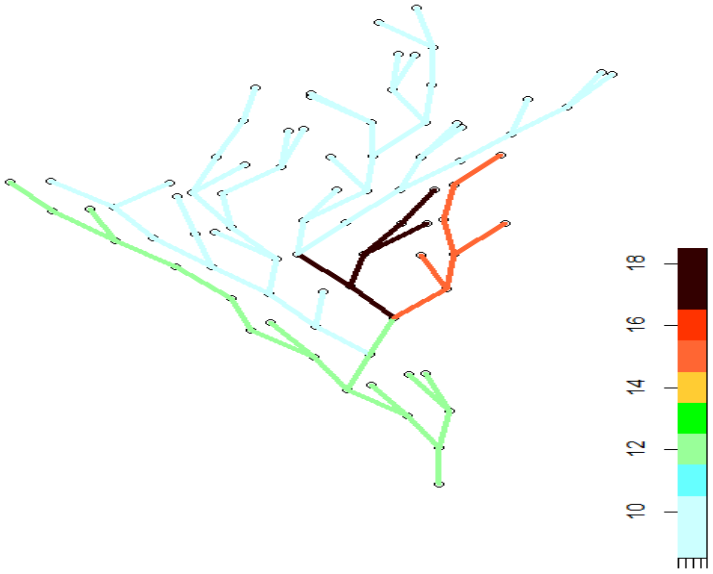

Figure 5.1: The simulated data for river flow, the network structure is introduced in Gallacher et al. (2017), the test function construction is from Park et al. (2022).

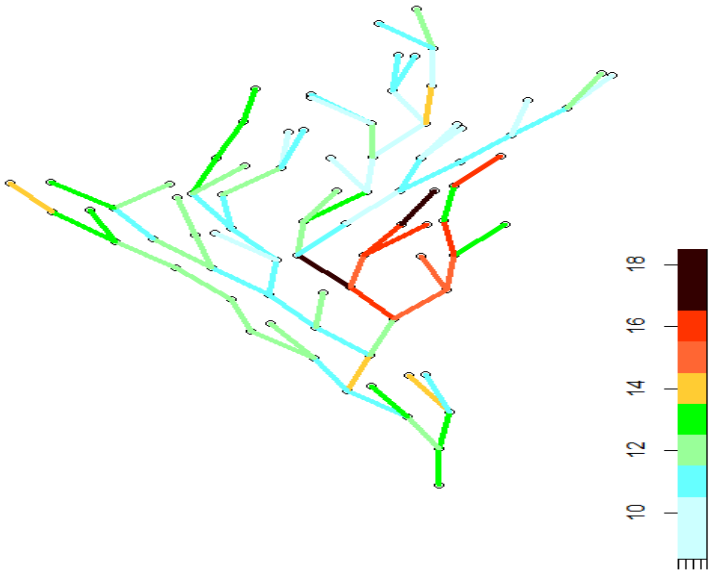

Figure 5.2: The flow data corrupted by noise $\epsilon \sim N(0, 4)$.

continue this procedure until there are more than 30 edges containing a value larger than 9. Figure 5.2 shows the simulated flow data corrupted by noise $\epsilon \sim N(0, 4)$. Table 5.25 shows the results from some of our algorithms which demonstrated to be competitive

in previous chapters, as well as their non-decimated versions (acronyms ending in 'nlt' in Table 5.25). We also display the results of the methods from Park et al. (2022).

AMSE (Std. error)	80 obs ($\sigma = 1$)	80 obs ($\sigma = 1.5$)	80 obs ($\sigma = 2$)
LG-Sid-p	0.6637 (0.0937)	0.9770 (0.1428)	1.2651 (0.1844)
LG-Sid-p-nlt (30)	0.5704 (0.0779)	0.8132 (0.1127)	1.0346 (0.1433)
LG-Aid-p	0.7183 (0.1139)	1.0484 (0.1480)	1.3429 (0.1932)
LG-Aid-p-nlt (30)	0.5645 (0.0786)	**0.8050 (0.1144)**	**1.0231 (0.1456)**
E-Lil-nwu	0.7832 (0.1317)	1.1240 (0.1930)	1.4027 (0.2256)
E-Lil-nwu-nlt (30)	0.6141 (0.0875)	0.8669 (0.1235)	1.0868 (0.1529)
E-Did-nwu	0.7271 (0.1118)	1.0314 (0.1498)	1.3179 (0.1897)
E-Did-nwu-nlt (30)	0.6149 (0.0856)	0.8662 (0.1215)	1.0845 (0.1522)
Bio-Haar	0.6786 (0.1086)	1.0222 (0.1568)	1.3495 (0.1962)
Bio-Haar-nlt-random (30)	**0.5569 (0.0754)**	0.8323 (0.1195)	1.0784 (0.1551)
EL-SC-D	0.7229 (0.1097)	1.0287 (0.1403)	1.3047 (0.1818)
EL-SC-D-nlt (30)	0.6047 (0.0858)	0.8512 (0.1210)	1.0708 (0.1488)
LGL-SC-D	0.7207 (0.1052)	1.0320 (0.1320)	1.3227 (0.1711)
LGL-SC-D-nlt (30)	0.5983 (0.0879)	0.8435 (0.1192)	1.0638 (0.1469)
Proposed (Median) from Park et al. (2022)	0.7265 (0.1212)	1.0249 (0.1510)	1.2818 (0.2599)
Proposed (Hard) from Park et al. (2022)	0.7396 (0.1317)	1.0666 (0.1678)	1.3705 (0.2495)
Proposed (Median, nlt) from Park et al. (2022)	0.7162 (0.1219)	1.0106 (0.1678)	1.2816 (0.2712)
O'Donnell	1.2698 (0.1251)	1.3815 (0.1727)	1.5421 (0.2017)

Table 5.25: AMSE for different methods performed on simulated flow data.

For LG-LOCAAT, we perform the algorithms 'LG-Sid-p' and 'LG-Aid-p' with the path length instead of those with the coordinate information since the algorithms in Park et al. (2022) are also based on path length metric and therefore using path length as the metric will give a fair comparison with their results.

For E-LOCAAT algorithms, we tested 'E-Did-nwu' and 'E-Lil-nwu' since the former one was shown to yield good results for test functions which display similar characteristics to the function here, and 'Lil' gives an algorithm with a different focus, treating each edge as a set of points (see Section 3.1.2). Biorthogonal Haar is also tested since it works well for piecewise constant functions (such as 'Blocks').

For Laplacian-LOCAAT, we mainly focus on 'EL-SC-D' and 'LGL-SC-D', since these two algorithms are totally algebraic and gave competitive results.

We can see that for the simulated flow data with the initial choice of trajectory, 'LG-Sid-p' results surpass all other (decimated) results, while 'Bio-Haar' and 'EL-SC-D' are also pretty competitive. The 'Bio-Haar' algorithm works particularly well when the noise level is not very high. Notably, with only one choice of trajectory, we obtain better

results than the non-decimated results (50 trajectories) from Park et al. (2022), while 'nlt' has been shown in general to drastically improve denosing results (see Knight and Nason (2009)).

For the non-decimated lifting algorithms, biorthogonal Haar appears to be an optimal choice when the noise level is not very high. The line graph algorithms work better than other choices, especially 'LG-Aid-p-nlt', which gives consistently competitive results, regardless of the noise level, as (very closely) does 'LG-Sid-p-nlt'. The Laplacian-based algorithms, 'EL-SC-D-nlt' and 'LGL-SC-D-nlt' also offer competitive choices compared to what previous literature offered, albeit somewhat weak performances than the line graph-based algorithms.

We also note that here the usefulness of introducing the random trajectory choice via non-decimated through the use of using averaged estimates. In particular, exploring 30 trajectories has a significant impact on lowering the AMSE especially for lower signal-to-noise ratios. For the highest reported noise level, the percentage improvement by introducing several trajectories is around 23.8% for 'LG-Aid-p-nlt', followed by 'Bio-Haar-nlt-random' at around 20.1% and the Laplacian-based 'LGL-SC-D-nlt' at around 19.6%. The percentage improvement decreases with the decreasing noise level, but it is still substantial for our competitor methods, ranging from around 21.4% for 'LG-Aid-p-nlt' to around 17.9% for 'Bio-Haar-nlt-random' and over 16% for Laplacian-based methods. The lowest improvement is for 'LG-Sid-p-nlt' still sizeable at 14.1%, 16.7%, and 18.2% corresponding to the reported noise level ordering in Table 5.25.

Figure 5.3 shows the denoised version of the noisy signal in Figure 5.2. The denoising is done by one of our non-decimated lifting algorithm ('LG-Aid-p-nlt'). From this figure, we can see that our algorithm is able to capture the piecewise continuous structure when the noise level is pretty high, which demonstrates that our algorithms work well for the simulated data.

Figure 5.4 shows a comparison of 'LG-Aid-p-nlt' and 'Bio-Haar-nlt-random' denoised networks when the noise level is set at $\sigma = 1.5$. Recall both methods demonstrated superiority over other methods in terms of bias and variance. The observed network

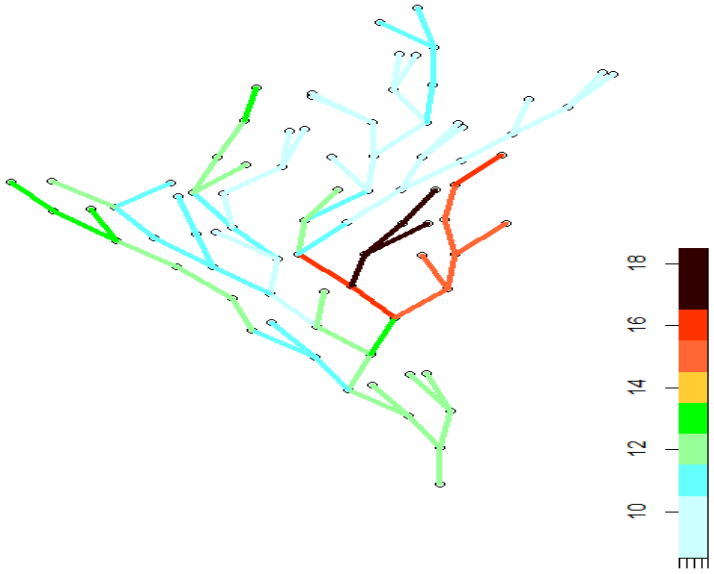

Figure 5.3: The denoised version of the simulated noisy data for river flow in Figure 5.2, via 'LG-Aid-p-nlt (30)'.

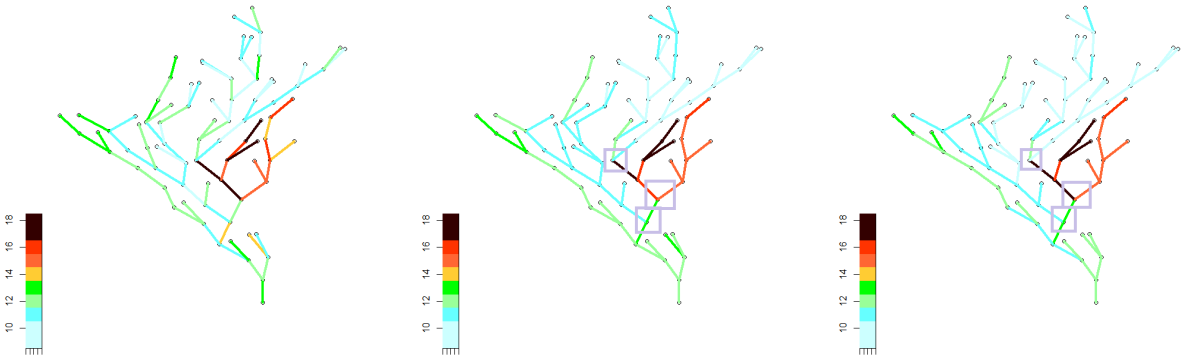

Figure 5.4: This is a new figure added to the thesis **Left:** The flow data corrupted by noise $\epsilon \sim (0, 1.5^2)$. **Middle:** The denoised river flow data by the non-decimated lifting algorithm 'LG-Aid-p-nlt' of Cao et al. (2024), using 30 trajectories. **Right:** The denoised river flow data by the proposed non-decimated lifting algorithm 'Bio-Haar-nlt-random' with 30 trajectories.

appears in the left hand panel, while the middle and right panels of Figure 5.4 illustrate our methods' denoising performance. We highlight the areas containing discontinuities via the rectangles. It can be observed that (although 'LG-Aid-p-nlt' performs better in terms of the AMSE), the 'Bio-Haar-nlt-random' demonstrates better performance for detecting discontinuities in the true function compared to 'LG-Aid-p-nlt'. Additionally,

for edge function values at the boundary (namely, edges with only one neighbour), 'Bio-Haar-nlt-random' achieves better recovery.

5.5 Real Data Analysis

In this section, we perform some of our proposed algorithms ('LG-Aid-nlt' and 'LG-Did-nlt') on a hydrology dataset from an open-resource website (`https://environment.data.gov.uk/hydrology/explore`)). The dataset consists of the dissolved oxygen (DO) measured by the amount of oxygen in the water, which is an indicator of water quality. For convenience, we quote the formal description for the DO data as follows.

'Dissolved oxygen (DO) is the measure of how much gaseous oxygen has been dissolved into the water and, therefore, how much is available to aquatic organisms residing in the body of water. Diurnal and seasonal patterns are often observed in this parameter. DO can be increased by events such as photosynthesis, aeration from the atmosphere and turbulent flow. Processes such as respiration of aquatic organisms and the decay of organic matter reduce the concentration of DO in water. DO saturation is a measure of the dissolved oxygen concentration in proportion to the maximum concentration that can be dissolved and is therefore represented as a percentage.'.

The details can be found on the same website in (`https://environment.data.gov.uk/hydrology/explore`). Understanding the DO data can allow us to know the influence of the weather and human behaviours (e.g., seasonal changes and pollution, respectively) on the river ecology. However, the data is normally corrupted by noise in reality, which is the motivation for applying our methods on this DO dataset.

The dataset is collected by 60 different stations near rivers, at two different time snapshots (10 May 2024 and 05 June 2024). There are some missing observations for the second time snapshot caused by the shutdown of the stations and/or some technical problems. Figure 5.5 gives the visualisation of the complete river network in England. The river system can be separated into 10 river basins, and the 60 sampled stations are distributed in 9 out of 10 river basins. The size of red circles in Figure 5.5 are

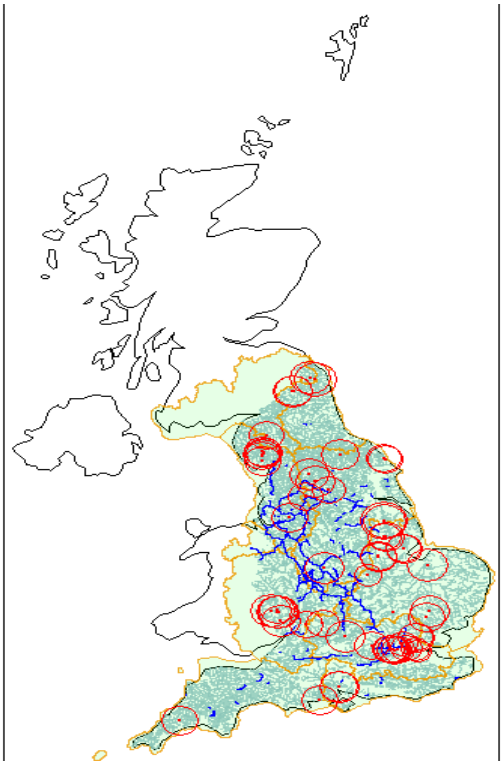

Figure 5.5: The river network geometry of England. The green-coloured areas bounded by orange curves are different river basins. The light-blue/grey curves are the river water bodies, and the blue ones are canal water bodies. The red points are the stations that collected data, and the red circle indicates the DO data value associated with the stations (larger circle indicates larger value) collected on 10th of May in 2024.

determined by the value of the DO data, where larger circle indicates larger value. The river network geometry can be found in another open-resource website (`https://environment.data.gov.uk/catchment-planning/`).

Recalling the discussion of LG-LOCAAT in Chapter 2, it is natural to consider the data collected from these stations as the observations from the edges, since each station is close to a certain river body. Note that we also need to define the vertices for obtaining the whole network structure. A natural choice is to consider river conjunctions, the sources and mouths of rivers as the vertices, and each river will be recognised as an edge. However, by this construction, the England river network will be a tree structure with a huge number of vertices, while we only have 60 observations for a single time snapshot, hence, sub-sampling the network size is needed. Fortunately, this process can

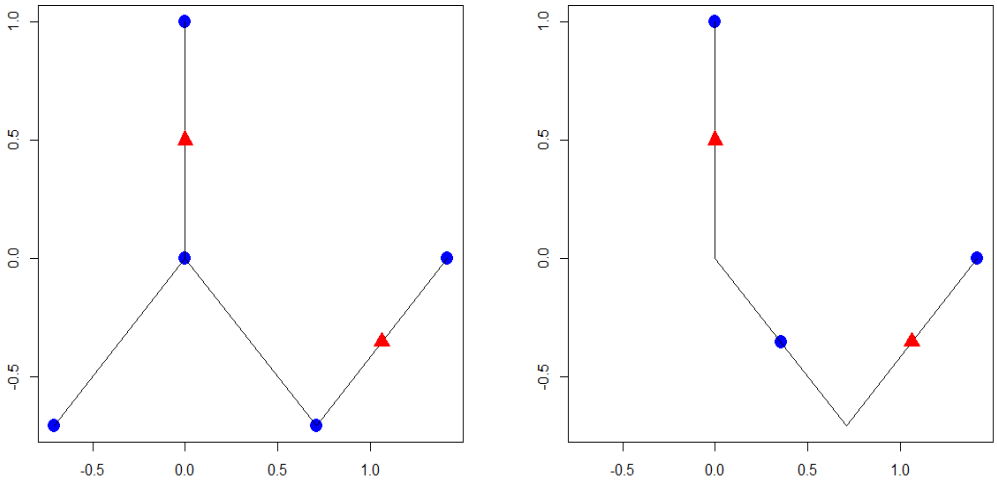

Figure 5.6: A visualised description for the toy network vertex sub-sampling. **Left:** Original toy network. **Right:** Toy network after sub-sampling. **Blue filled dot points**: network vertices. **Red filled triangle points**: stations on edges.

be handled by the power of refinement for metrized graphs, see Section 1.4.4. Consider the toy model showed on the left in Figure 5.6, where each (blue) dot point represents a vertex (river confluence), the lines between two adjacent vertices are the edges (rivers), and the (red) triangles are observations from two stations. The sub-sampling of the network is done as follows.

- If an edge (river) is not on the shortest paths of any two locations (stations), then we remove this edge and fuse its two vertices.

- If there is one or more edges between two locations (exclude those two edges associated to these two locations), then we remove all those edges in between them and fuse all those vertices together.

See Figure 5.6 for an intuitive visualisation for sub-sampling a network.

Moreover, another possible scenario is that there might be more than one station located around one river. This situation can be solved by taking the refinement of such an edge, by means of inserting a point in between two adjacent stations.

Subsequently, a network with 61 vertices (stations) and 60 edges (rivers after sub-sampling and refinement) can be obtained, which allows us to perform the methods we proposed. However, even if the whole network structure has been constructed, some of the information is still difficult to get, for example, the length of an edge (river), e.g., each river is a curve instead of a line segment. Hence, we propose to use line-graph-based algorithms proposed in Chapter 2, which crucially enables us to use the stations as *line graph vertices*, along with their coordinates. Recall in Chapter 2 new vertices are constructed as middle points of metrized edges, this construction can be easily generalised to any point with the exception of the two endpoints on that metrized edge. Specifically, in our context here, we propose to consider the stations and their associated latitude and longitude as the (metrized) new vertices and their coordinates. Recall in Figure 5.6, the vertex (the blue dot in between those two red triangles) is normally relocated if we remove the edge (or some edges) in between two stations. Another advantage of LG-based algorithms is that we do not have to know the exact location (coordinate) of this relocated vertex (blue dot), hence, LG-based methods are computationally feasible and efficient.

As we already mentioned, the second time snapshot (05/Jun/2024) has some missing observations, and we further perform the network sub-sampling to obtain a new network and then perform our algorithm.

Remarks for Neighbourhood Structure

In the work from Park et al. (2022), the neighbourhood selection is based on the concept of 'flow-connected' introduced by Hoef et al. (2006). Under that concept, if the intersection of upstreams of two stations is a non-empty set, then they are defined as neighbouring each other.

In our work, we conjecture that river quality related indices are highly dependent on both river network structure and Euclidean space (for example, human and animal behaviours, which are more likely to be on a two-/three-dimensional Euclidean space instead of the network domain have influences). Hence, defining neighbourhood struc-

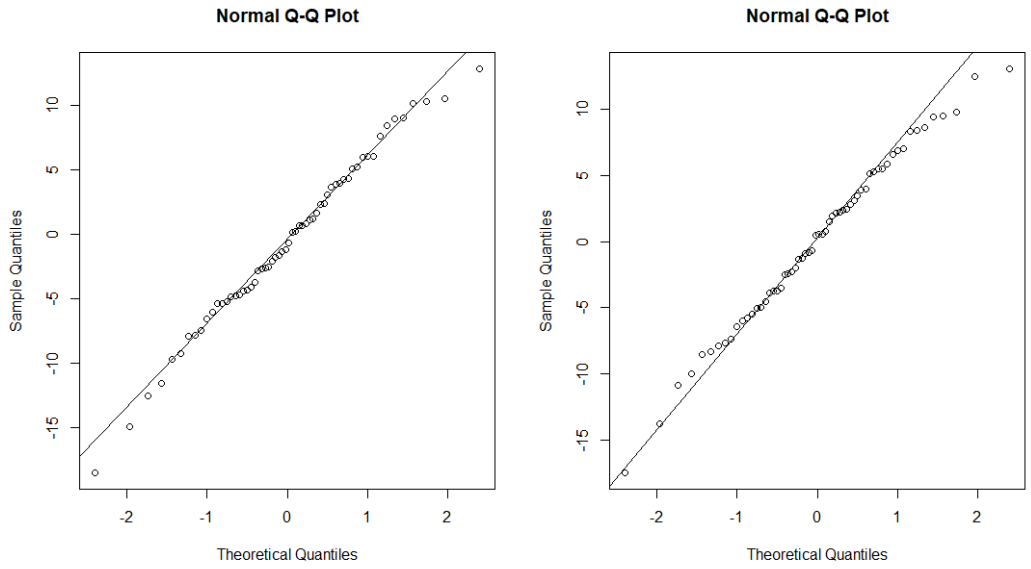

Figure 5.7: **Left:** Residual Q-Q plot of the DO data analysis (the data is collected on 10/May/2024, there are observations from all 60 stations) with algorithm 'LG-Aid-c-nlt'. **Right:** Residual Q-Q plot obtained using algorithm 'LG-Did-c-nlt'.

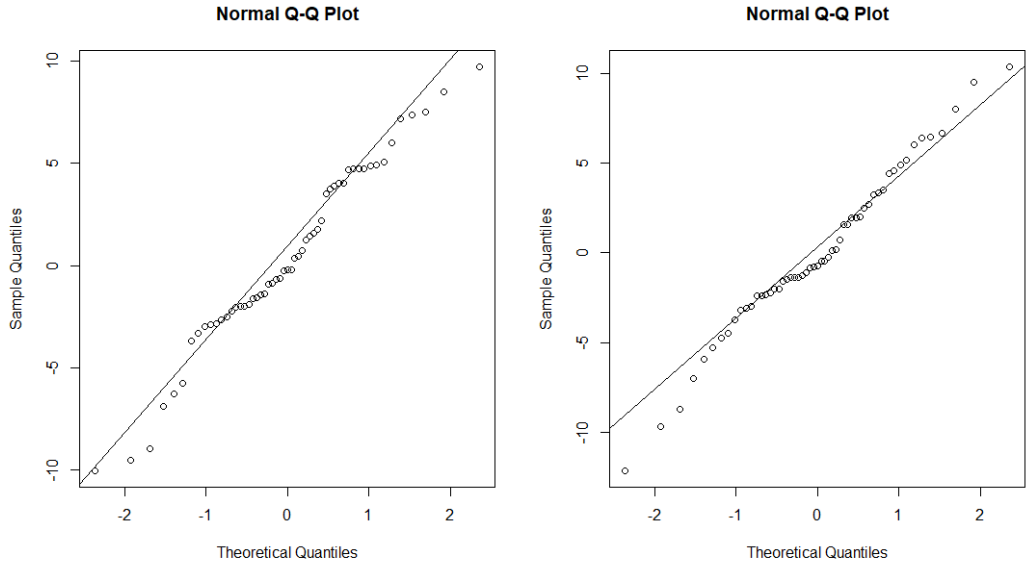

Figure 5.8: **Left:** Residual Q-Q plot of the DO data analysis (the data is collected on 05/Jun/2024, there are 55 observations from 60 stations with 5 missing observations) with algorithm 'LG-Aid-c-nlt'. **Right:** Residual Q-Q plot obtained using algorithm 'LG-Did-c-nlt'.

ture as in equation (2.1.1), and performing the LG-LOCAAT algorithm with coordinate information is also useful since it enables us to capture those information.

5.5.1 Results

Guided by the results from previous simulation study, the non-decimated 'LG-Aid-c' and 'LG-Did-c' will be performed. From Section 5.3.1, we conclude that increasing the number of trajectories in non-decimated lifting increases the denoising performance, hence, we perform a 100-trajectory non-decimated lifting for the real data case.

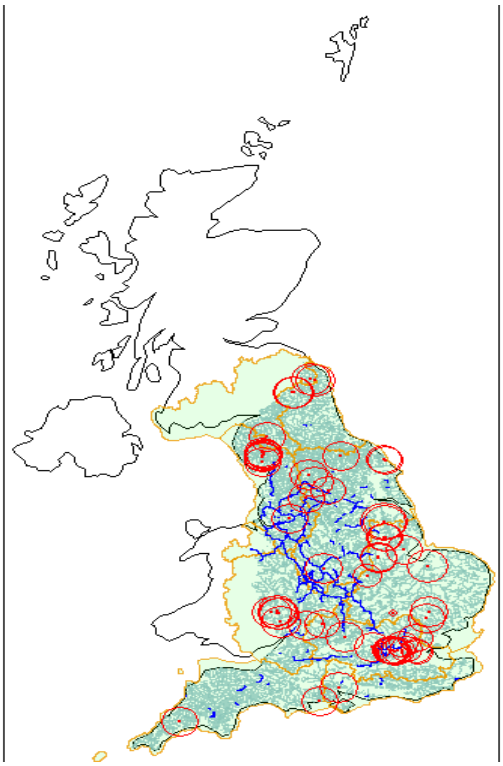

Figure 5.9: The denoised version for the data show in Figure 5.5. The algorithm used is 'LG-Did-c-nlt', with 100 trajectories.

Figure 5.9 shows the denoised version of the data shown in Figure 5.5, and Figure 5.10 shows the visualisation of the residuals after denoising. We observe that our methods tend to smooth the observations based on the network structure, see the red circle chain from near Welsh to the southeast of England, and the values in northern England. We can see in Figure 5.9 that the DO values tend to be large around the countryside area

and the area close to the sea, and many small values are around the London area or the Manchester area, which follows the intuition that big cities might have a big impact on the water quality.

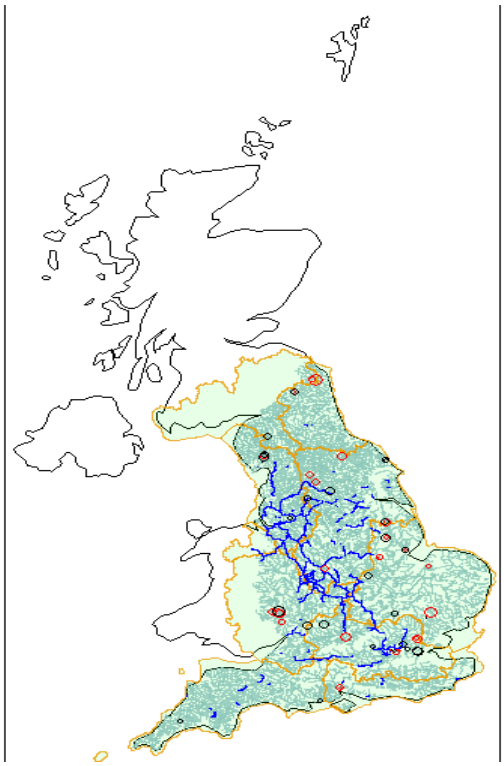

Figure 5.10: The visualisation for the residuals. Positive residuals are represented by the red circles, while negative residuals are represented by the black circles. The size of the circle are determined by the absolute values of the residuals.

Figures 5.7 and 5.8 show the residual Q-Q plots for the DO data analysis at two different time snapshots. We can see that the residuals for the data collected on 10/May/2024 appear to follow the normal distribution. Although the residual Q-Q plots for the data collected on 05/Jun/2024 slightly diverge from the Q-Q lines, we can still agree on the residuals also appear to follow the normal distribution in consideration of 5 missing values (in which we may have to connect two stations that are not very close to each other to ensure a connected network). The results give us the confidence that our algorithms work well for this real-data.

Chapter 6

Conclusions and Future Work

This thesis has proposed three novel lifting-scheme-based algorithms which provide multiscale methods for data collected from network edges. We now draw conclusions for all these methods and propose some possible future work.

LG-LOCAAT

In Chapter 2, we discussed the first proposal that consists of transforming the original graph into its associated line graph, which allows us to use the LOCAAT transform on the line graph domain. In this context, we also brought the 'metrized graph' (Baker and Faber; 2006) into the wavelet construction, which provided advantages in the wavelet context too, since it has been proven by Baker and Faber (2006) that Fourier analysis can be generalised onto the metrized graph. For example, rather than 'defining' the integral values for initial primal scaling functions, the metrized graph provides a function representation which is isometric to the direct sum of functions defined on multiple one-dimensional axes. Based on this, we illustrate the wavelet coefficient magnitude decay as a function of scale (which is defined as the primal integral value when the associated vertex has been predicted), see equation (2.3.16). The upper bound of the wavelet coefficient magnitudes also provide an explanation of why using average distances as initial primal scaling function integrals results in good performances.

Throughout the simulation studies, we demonstrate that our proposed LG-LOCAAT algorithm works well with the variants of the test functions used in Jansen et al. (2009).

As illustrated by the real data example, the line graph construction renders itself useful for particular datasets, where the edge lengths might not be immediately available but (new) vertices of the line graph are endowed with full information.

E-LOCAAT

In Chapter 3, we proposed the E-LOCAAT algorithm for directly dealing with the problem of finding edge-based function expansions on the original graph domain. The need for a method capable of working in the original domain is determined by certain tasks that might require the ability to capture the graph structure at different resolutions. For example, in the context of the edge clustering problem discussed in Evans and Lambiotte (2009), if we apply the E-LOCAAT algorithm to the dataset, then the wavelets corresponding to relatively small scales (support) can be interpreted as corresponding to highly connected small groups within the community. In contrast, the LG-LOCAAT does not have the ability to render such representations, as we discussed in Section 2.3.4.

Our work is inspired by the spherical wavelets constructed by Schröder and Sweldens (1995a,b), but for a network scenario instead. Since the scheme does not rely on global distance information (it only needs the distance measure between the an edge and its neighbouring edges at each stage) and the relinkage is trivial, the E-LOCAAT provides a faster algorithm compared to LG-LOCAAT. The algorithm also works for the situation when the vertex Euclidean coordinates are not available.

Similar to Schröder and Sweldens (1995a), a biorthogonal 'Haar-like' construction has been introduced in our work. This biorthogonal Haar E-LOCAAT shares some of the advantages of LOCAAT (ability to deal with graph-type data, provides continuous scale notion), and at the same time it inherits the self-similarity (all scaling functions are of the same shape) from the cascade algorithm (although the construction does not rely on it).

Throughout the simulation studies, E-LOCAAT also shows competitive denoising performance, in particular the biorthogonal Haar when the underlying function is piecewise continuous.

Laplacian-LOCAAT

In Chapter 4, we construct a lifting scheme based on the Laplacian matrices of graph. This is mainly motivated by works in the signal processing community, see for example, Crovella and Kolaczyk (2003), Hammond et al. (2011), Shuman et al. (2013), Shuman et al. (2016), Ortega et al. (2018), and Stanković et al. (2020). These works explored the harmonic analysis via graph Laplacians. The edge variant of the Laplacian, described in Zelazo et al. (2007), Zelazo and Mesbahi (2010), Schaub and Segarra (2018), and Schaub et al. (2020) enables us to deal with the edge lifting construction. The advantage of this construction is that the observation space does not have to be a metric space. The scheme is valid as long as neighbourhood structures can be identified and the weights between each pair of neighbouring edges are available/can be obtained. The algorithm using edge Laplacian provides a faster transform since the relinkage via the incidence matrix is convenient.

Non-decimated Versions and Real Data Analysis

The non-decimated lifting construction from Knight and Nason (2009) can be easily adapted into our algorithms, which introduces a better performance in denoising problems than for the one-trajectory algorithms. The biorthogonal Haar construction highlights the importance of the update coefficient values and although large update coefficient values lead to a unstable transform, low update coefficient values do not yield a further improved denoising performance.

The real data analysis highlights the versatility of our algorithms that allows the three algorithms we proposed to deal with many real-life data situations even when there is a lack of some particular information type (for example, lack of edge lengths).

Method Comparison

Throughout the simulation studies and real data analysis in Chapters 2, 3, 4, and 5, we have illustrated the advantages and the competitive performance of our proposed methods. Here we provide an overall discussion and guidance on our proposed methods.

For LG-LOCAAT, the line graph transform allows us to take full advantage of the Euclidean coordinates. In the context of our real data analysis, due to the geometrical complexity of the river network, the length of each edge is difficult to determine, whereas the coordinates of each station are immediately available. Thus, LG-LOCAAT can smoothly address this issue, while both E-LOCAAT and Laplacian-LOCAAT will struggle with it. In contrast, E-LOCAAT and Laplacian-LOCAAT algorithms could be applied to datasets where the measure of edges is available, such as traffic data, where the edge length corresponds to the length of each road.

If feasible, the biorthogonal Haar construction is recommended for scenarios where the underlying true function contains discontinuities and the local smooth parts exhibit relatively small variations. We conjecture that biorthogonal Haar construction may have potential for network anomaly detection problems.

Under certain circumstances, more trajectories may be preferred for designing non-decimated versions. Therefore, E-LOCAAT and Laplacian-LOCAAT may be preferred because of their higher computational efficiency.

6.1 Future Work

There are several future research directions that can be related to the topics covered in this thesis, and here we discuss some possibilities.

The first concentrates on the underlying 'true' (edge) functions and on the usefulness of being able to describe various properties of the functions, such as their Lipschitz continuity (for vertex-based graph functions) as we described in Section 2.3.3. For the edge scenarios, describing functions becomes more complex and a possible future contribution could be to develop the connection between edge bases and the *cartoon function class*. A function is referred to as a cartoon function if it can be written as

$$f = f_1 + \sum_{i=2}^{p} \chi_{\mathcal{B}_i} f_i, \tag{6.1.1}$$

for some p, where $\mathcal{B}_i$ is a compact domain in space such as $\mathbb{R}^d$. The reader can refer to Donoho (1997, 2001) and Grohs et al. (2016) for the details about this function class, in which the function class is connected to image data.

Due to the iterative nature of LOCAAT-based algorithms, some theoretical aspects are not easy to verify, and this could be an interesting direction for future research. Additional theoretical background could be considered, such as the derivation of exact bounds for the condition number of the lifting matrix and the distribution of the associated wavelet coefficients.

To find a motivating real-world or simulated example where Laplacian-LOCAAT is well-suited, but both LG-LOCAAT and E-LOCAAT struggle to deal is another interesting problem. A possible scenario is when certain covariates are used to construct the prediction weights, which could naturally be incorporated as associated elements into the (edge) Laplacian matrices.

Other variants of the lifting scheme can be extended to the edge-scenario, for instance, the adaptive lifting introduced by Nunes et al. (2006). A key question is how to define quadratic and cubic prediction in a network setting. A potential approach is to leverage powers of the graph (or edge) Laplacian matrices to construct filters. For instance, the filters for quadratic case can be designed based on $\left(\mathcal{L}^{\mathcal{E}}\right)^2 = \mathcal{L}^{\mathcal{E}}\,\mathcal{L}^{\mathcal{E}}$. Alternatively, the adaptive neighbourhood selection (Nunes et al.; 2006) can also be extended to the network edge regression problem. In the case of a single chosen neighbour, the algorithm can be considered as the adaptive version of our biorthogonal Haar E-LOCAAT design,

which may have the potential for denoising edge signals collected from a denser graph, given the finding that biorthogonal Haar E-LOCAAT performs better than any other algorithms for those signals.

To further generalise the lifting construction for a direct multi-graph, would allow for more application scenarios, and the reader can refer to Sevi et al. (2023) for an overview. Here, the metrized graph construction cannot be applied to a direct graph as even the vertices might not form a metric space, thus, constructing initial scaling functions and function expansions becomes an issue. For this case, a Laplacian-based construction might provide the way forward, along with the discrete calculus notions introduced by Grady and Polimeni (2010).

For the nonparametric regression problem on the network edge functions, going beyond the normality assumption could be another future research direction. As mentioned in Jansen et al. (2009), the krill data they used does not appear to be Gaussian. Although it was a node scenario, it is likely that this might also hold for some edge data. For example, in the case of traffic flow data, a framework based on Poisson random variables is more commonly used in the literature, see Kolaczyk and Csárdi (2014). Hence, one might consider adapting the Haar-Fisz transform Fryzlewicz and Nason (2004) to our biorthogonal Haar framework. Another interesting future research direction is the study of correlated noise in network structures.

In particular, new methods that are able to cope with edge data collected from a network over a long time span can be another future direction. For instance, a spatio-temporal network model can be modelled similarly to the one introduced in Mahadevan (2010), by assigning a distance metric to both spatial and time dimensions. Then a set of wavelet coefficients can be obtained by means of our algorithms, or alternatively, one can model the wavelet coefficients using the GNAR framework introduced by Knight et al. (2019). A further improvement could be to leverage the GNAR-edge model introduced by Mantziou et al. (2023), which allows for time-varying edge weights, including edges dropping in or out. Such a framework would allow us to forecast the time-varying network data and to compute the confidence intervals could be another future research direction.

Finally, constructing wavelets with higher vanishing moments is always of interest. In particular, for constructing wavelets on a graph structure, we conjecture that some regularisation might be useful for different application scenarios, which might suggest constructing wavelets in a weighted inner product space instead of the general L^2, under which the work in Sweldens (1996a) will be helpful. For the regularisation of graph structure, the reader can refer to Chung (1997) and Smola and Kondor (2003).

Appendix A

Path Distance

In graph theory, a (combinatorial) path of length k (where $k \in \mathbb{Z}$ and $k \geq 2$) in a graph $G = (\mathcal{V}, \mathcal{E})$ is defined by a chain of distinct vertices (Diestel; 2005)

$$P_{v_{p_1}, v_{p_k}} = v_{p_1}\, v_{p_2}\, \cdots\, v_{p_k}, \tag{A.0.1}$$

where $p_s \neq p_{s'}$ if $s \neq s'$, and $v_{p_i} \leftrightsquigarrow v_{p_{i+1}}$ for all $i \in \{1, ..., k-1\}$. Following the notation in Diestel (2005), we refer to the path in equation (A.0.1) as a v_{p_1}-v_{p_k} path. For this v_{p_1}-v_{p_k} path, there is an alternative edge representation such that

$$P_{v_{p_1}, v_{p_k}} = e_{q_1}\, e_{q_2}\, \cdots\, e_{q_{k-1}},$$

where $e_{q_t} = \{v_{p_t}, v_{p_{t+1}}\}$, for $t \in \{1, ..., k-1\}$. Note that there could be more than one v_{p_1}-v_{p_k} path in the graph, with each of them being assigned a quantity based on graph-defined measures, for example, the weight associated with each edge (see Section 1.4.3). We denote $\mathbf{p}_{v_i} \in \Gamma$ as the point corresponding to v_i, then the path $P^{\Gamma}_{\mathbf{p},\mathbf{p}'}$ between two points $\mathbf{p}, \mathbf{p}' \in \Gamma$ on the metrized graph space Γ is the generalised form of equation (A.0.1), such that

$$P^{\Gamma}_{\mathbf{p},\mathbf{p}'} = \begin{cases} \mathbf{p}\, \mathbf{p}_{v_{p_1}}\, \cdots\, \mathbf{p}_{v_{p_k}}\, \mathbf{p}', & \text{if } \mathbf{p} \text{ and } \mathbf{p}' \text{ belong to different metrized edges;} \\ \mathbf{p}\, \mathbf{p}', & \text{if } \mathbf{p} \text{ and } \mathbf{p}' \text{ belong to the same metrized edge,} \end{cases} \tag{A.0.2}$$

where $\{\mathbf{p}_{v_{p_i}}\}_{i=1}^{k}$ are the points (corresponding to the vertices) that the $\mathbf{p}$-$\mathbf{p}'$ path passes through, and for $\mathbf{p}_{v_{p_i}}$ and $\mathbf{p}_{v_{p_{i+1}}}$, their corresponding vertices are neighbouring to each

other. Similarly, the edge representation of equation (A.0.2) can be written as

$$P^{\Gamma}_{\mathbf{p},\mathbf{p}'} = \begin{cases} \mathbf{p}\, e^{\mathrm{met}}_{q_1} \cdots e^{\mathrm{met}}_{q_{k-1}}\, \mathbf{p}', & \text{if } \mathbf{p} \text{ and } \mathbf{p}' \text{ belong to different metrized edges;} \\ \mathbf{p}\, \mathbf{p}', & \text{if } \mathbf{p} \text{ and } \mathbf{p}' \text{ belong to the same metrized edge,} \end{cases}$$

where $e^{\mathrm{met}}_{q_t} = [v_{p_t}, v_{p_{t+1}}]$. Notice that $\{\mathbf{p}, \mathbf{p}'\} \cup \mathrm{Vex}_G(\Gamma)$ is a vertex set which defines a refinement of G. Hence, for the case that $\mathbf{p}$ and $\mathbf{p}'$ belong to different metrized edges, the intervals $[\mathbf{p}, \mathbf{p}_{v_{p_1}}]$ and $[\mathbf{p}_{v_{p_k}}, \mathbf{p}']$ can be considered as two metrized edges and measured by the length function, denoted as $\ell_{\mathbf{p},\mathbf{p}_{v_{p_1}}}$ and $\ell_{\mathbf{p}_{v_{p_k}},\mathbf{p}'}$, respectively. If $\mathbf{p}, \mathbf{p}'$ are located on the same metrized edge e^{met}_k, then we denote $\ell([\mathbf{p}, \mathbf{p}'])$ as the length of the sub-interval of e^{met}_k bounded by these two points. Then the length of $\mathbf{p}$-$\mathbf{p}'$ path can be obtained as

$$\ell(P^{\Gamma}_{\mathbf{p},\mathbf{p}'}) = \begin{cases} \ell_{\mathbf{p},\mathbf{p}_{v_{p_1}}} + \ell_{\mathbf{p}_{v_{p_k}},\mathbf{p}'} + \sum_{t=1}^{k-1} \ell_{q_t}, & \text{if } \mathbf{p} \text{ and } \mathbf{p}' \text{ belong to different metrized edges;} \\ \ell([\mathbf{p}, \mathbf{p}']), & \text{if } \mathbf{p} \text{ and } \mathbf{p}' \text{ belong to the same metrized edge,} \end{cases}$$

Note that for a non-tree graph, there can be more than one path between two points if they belong to different edges. Denote $\mathcal{P}^{\Gamma}_{\mathbf{p},\mathbf{p}'}$ as the set that contains all path between $\mathbf{p}$ and $\mathbf{p}'$, the path distance is defined as

$$\mathrm{dist}_{\mathrm{path}}(\mathbf{p}, \mathbf{p}') = \min_{P \in \mathcal{P}^{\Gamma}_{\mathbf{p},\mathbf{p}'}} \{\ell(P)\},$$

which is the shortest path length on Γ between $\mathbf{p}$ and $\mathbf{p}'$.

Appendix B

Formulae of Test Functions

$$
\begin{aligned}
g_1 \equiv g(x,y) &= (2x+y)\mathbb{1}((3x-y)<1) + (10-x)\mathbb{1}((3x-y)\geq 1) \\
\text{mfc} \equiv g(x,y) &= (2x+y)\mathbb{1}((3x-y)<1) + (5x-y)\mathbb{1}((3x-y)\geq 1) \\
\text{Blocks} \equiv g(x,y) &= 1\cdot\mathbb{1}(0\leq x<0.1) + 2\cdot\mathbb{1}(0\leq y<0.2) \\
&\quad + 3\cdot\mathbb{1}(0.3<x<0.4)\mathbb{1}(0.7<y<0.8) + 4\cdot\mathbb{1}(0.7<x<0.8)\mathbb{1}(0.7<y<0.8) \\
&\quad + 5\cdot\mathbb{1}(0.5<x<0.6)\mathbb{1}(0.4<y<0.6) + 6\cdot\mathbb{1}(0.3<x<0.8)\mathbb{1}(0.2<y<0.3) \\
&\quad + 7\cdot\mathbb{1}(0.2<x<0.3)\mathbb{1}(0.3<y<0.4) + 8\cdot\mathbb{1}(0.8<x<0.9)\mathbb{1}(0.3<y<0.4) \\
\text{Doppler} \equiv g(x,y) &= \sin(1/(x^2+y^2)) \\
\text{Bumps} \equiv g(x,y) &= \exp((-|x-0.1|-|y-0.4|)/0.1)/0.04 \\
&\quad + \exp((-|x-0.8|-|y-0.7|)/\sqrt{0.02})/0.08 \\
&\quad + \exp((-|x-0.9|-|y-0.1|)/\sqrt{0.015})/0.06 \\
\text{Heavisine} \equiv g(x,y) &= \sin(20\sqrt{x^2+y^2}) + \frac{1}{0.04\pi}\exp\left(\frac{-(x-0.55)^2-(y-0.5)^2}{0.0002}\right)
\end{aligned}
$$

Appendix C

Proofs

C.1 Proof for Proposition 2.3.1

Starting with the integrals $C \cdot \underline{I}^* = \{CI^*_{k,m}\}_{k\in\{1,\dots m\}}$ will not change the stage-m split step as long as $C > 0$, and the same new vertex $v^*_{k_m}$ will be lifted at this stage. Since the predict step is independent of the integral values, the value of the detail coefficient $d^*_{k_m}$ and the prediction weights remain the same. For any $s \in \mathcal{N}^{\mathcal{V}^*}_{k_m,m}$, the update of the integral sequence is now

$$\begin{aligned} & CI^*_{s,m} + a^*_{s,m} CI^*_{k_m,m} \\ =& C(I^*_{s,m} + a^*_{s,m} I^*_{k_m,m}) \\ =& CI^*_{s,m-1}, \end{aligned}$$

which indicates that in the stage-$(m-1)$, the integral sequence is proportional to the integral with the same multiplier constant C. The s-th update coefficient is determined by the minimum norm solution such that

$$\begin{aligned} b^{\Gamma^*}_{s,m} &= \frac{CI^*_{s,m-1} CI^*_{k_m,m}}{\sum_{t:v^*_t \in \mathcal{N}^{\mathcal{V}^*}_{k_m,m}} (CI^*_{t,m-1})^2} \\ &= \frac{I^*_{s,m-1} I^*_{k_m,m}}{\sum_{t:v^*_t \in \mathcal{N}^{\mathcal{V}^*}_{k_m,m}} (I^*_{t,m-1})^2}, \end{aligned}$$

which coincides with the result when starting with $\underline{I}^*$. Thus, repeating the procedures above and by induction, we conclude that the detail coefficients and the filters do not change upon a proportional change in the integral values.

C.2 Proof for Proposition 2.3.2

Recall the detail coefficient is obtained by the prediction step, in which we have

$$\left|d_{k_r}^{\Gamma^*}\right| = \left|c_{k_r,r}^{\Gamma^*} - \sum_{s:v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}} a_{s,r}^{\Gamma^*} c_{s,r}^{\Gamma^*}\right|. \tag{C.2.1}$$

If for all $t : v_t^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*} \cup \{v_{k_r}^*\}$, we have that $c_{t,r}^{\Gamma^*} = g_t^{\Gamma^*}$, then equation (C.2.1) can be written as

$$\begin{aligned}
\left|d_{k_r}^{\Gamma^*}\right| &= \left|g_{k_r}^{\Gamma^*} - \sum_{s:v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}} a_{s,r}^{\Gamma^*} g_s^{\Gamma^*}\right| \\
&= \left|\sum_{s:v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}} a_{s,r}^{\Gamma^*} (g_{k_r}^{\Gamma^*} - g_s^{\Gamma^*})\right| \\
&\leq \sum_{s:v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}} a_{s,r}^{\Gamma^*} \left|g_{k_r}^{\Gamma^*} - g_s^{\Gamma^*}\right|,
\end{aligned} \tag{C.2.2}$$

since $a_{s,r}^{\Gamma^*} \geq 0$ for all $s : v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}$. Then because the function g^{Γ^*} is Lipschitz with a constant $0 < C < \infty$, then we have

$$\begin{aligned}
\left|d_{k_r}^{\Gamma^*}\right| &\leq \sum_{s:v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}} a_{s,r}^{\Gamma^*} \left|g_{k_r}^{\Gamma^*} - g_s^{\Gamma^*}\right| \\
&\leq \sum_{s:v_s^* \in \mathcal{N}_{k_r,r}^{\mathcal{V}^*}} a_{s,r}^{\Gamma^*} C \operatorname{dist}(\mathbf{p}_{v_{k_r}^*}^*, \mathbf{p}_{v_s^*}^*).
\end{aligned} \tag{C.2.3}$$

Recall that the prediction weights are normalised inverse distances, $a_{s,r}^{\Gamma^*} = \frac{1/\text{dist}(\mathbf{p}^*_{v^*_{k_r}}, \mathbf{p}^*_{v^*_s})}{\sum_{t:v^*_t \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}} 1/\text{dist}(\mathbf{p}^*_{v^*_{k_r}}, \mathbf{p}^*_{v^*_t})}$. Plugging this into the inequality (C.2.3), we have

$$\begin{aligned}
\left|d_{k_r}^{\Gamma^*}\right| &\leq \sum_{s:v^*_s \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}} \frac{1/\text{dist}(\mathbf{p}^*_{v^*_{k_r}}, \mathbf{p}^*_{v^*_s})}{\sum_{t:v^*_t \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}} 1/\text{dist}(\mathbf{p}^*_{v^*_{k_r}}, \mathbf{p}^*_{v^*_t})} C \,\text{dist}(\mathbf{p}^*_{v^*_{k_r}}, \mathbf{p}^*_{v^*_s}) \\
&= \sum_{s:v^*_s \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}} \frac{1}{\sum_{t:v^*_t \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}} 1/\text{dist}(\mathbf{p}^*_{v^*_{k_r}}, \mathbf{p}^*_{v^*_t})} C \\
&= \frac{1}{|\mathcal{N}^{\mathcal{V}^*}_{k_r,r}|} \sum_{s:v^*_s \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}} \frac{|\mathcal{N}^{\mathcal{V}^*}_{k_r,r}|}{\sum_{t:v^*_t \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}} 1/\text{dist}(\mathbf{p}^*_{v^*_{k_r}}, \mathbf{p}^*_{v^*_t})} C \\
&\leq \frac{1}{|\mathcal{N}^{\mathcal{V}^*}_{k_r,r}|} \sum_{s:v^*_s \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}} \frac{\sum_{t:v^*_t \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}} \text{dist}(\mathbf{p}^*_{v^*_{k_r}}, \mathbf{p}^*_{v^*_t})}{|\mathcal{N}^{\mathcal{V}^*}_{k_r,r}|} C \\
&= \frac{\sum_{t:v^*_t \in \mathcal{N}^{\mathcal{V}^*}_{k_r,r}} \text{dist}(\mathbf{p}^*_{v^*_{k_r}}, \mathbf{p}^*_{v^*_t})}{|\mathcal{N}^{\mathcal{V}^*}_{k_r,r}|} C, \qquad \text{(C.2.4)}
\end{aligned}$$

since for positive $x_1, ..., x_n$, we have

$$\frac{n}{1/x_1 + ... + 1/x_n} \leq \frac{x_1 + ... + x_n}{n}. \qquad \text{(C.2.5)}$$

C.3 Proof for Lemma 4

The first part of this proof (the equivalence of null spaces) can also be found in Zelazo and Mesbahi (2010) and Schaub and Segarra (2018). Suppose we have a $n \times m$ oriented weighted incidence matrix $\overrightarrow{B}^*_\omega$. Hence, we have $\mathcal{L}^{\mathcal{E},\omega}_m = {\overrightarrow{B}^*_\omega}^T \overrightarrow{B}^*_\omega$. The null space of $\mathcal{L}^{\mathcal{E},\omega}$ is defined as

$$\textbf{Null}\left(\mathcal{L}^{\mathcal{E},\omega}_m\right) = \left\{\underline{x} \in \mathbb{R}^m \,|\, \mathcal{L}^{\mathcal{E},\omega}\underline{x} = \underline{0}_m\right\},$$

where $\underline{0}_m$ is a vector with m zeros. Similarly, the null space of the incidence matrix $\overrightarrow{B}^*_\omega$ can be written as

$$\textbf{Null}\left(\overrightarrow{B}^*_\omega\right) = \left\{\underline{x} \in \mathbb{R}^m \,|\, \overrightarrow{B}^*_\omega\underline{x} = \underline{0}_n\right\},$$

where $\underline{0}_n$ is a vector with n zeros. An alternative form for the null space of $\mathcal{L}^{\mathcal{E},\omega}_m$ is

$$\textbf{Null}\left(\mathcal{L}^{\mathcal{E},\omega}_m\right) = \textbf{Null}\left({\overrightarrow{B}^*_\omega}^T \overrightarrow{B}^*_\omega\right) = \left\{\underline{x} \in \mathbb{R}^m \,|\, {\overrightarrow{B}^*_\omega}^T \overrightarrow{B}^*_\omega\underline{x} = \underline{0}\right\}.$$

Assume we have a vector $\underline{v} \in \mathbf{Null}\left(\overrightarrow{B}_{\omega}^{*}\right)$, then $\overrightarrow{B}_{\omega}^{*}\underline{v} = \underline{0}$ holds. Thus, immediately we have ${\overrightarrow{B}_{\omega}^{*}}^{T}\overrightarrow{B}_{\omega}^{*}\underline{v} = 0$, which indicates that

$$\mathbf{Null}\left(\overrightarrow{B}_{\omega}^{*}\right) \subseteq \mathbf{Null}\left(\mathcal{L}_{m}^{\mathcal{E},\omega}\right). \tag{C.3.1}$$

On the other hand, suppose we have a vector $\underline{u} \in \mathbf{Null}\left(\mathcal{L}_{m}^{\mathcal{E},\omega}\right)$, then ${\overrightarrow{B}_{\omega}^{*}}^{T}\overrightarrow{B}_{\omega}^{*}\underline{u} = 0$ holds, which leads to the following derivation.

$$\begin{aligned} {\overrightarrow{B}_{\omega}^{*}}^{T}\overrightarrow{B}_{\omega}^{*}\underline{u} = \underline{0} \;&\Rightarrow\; \underline{u}^{T}{\overrightarrow{B}_{\omega}^{*}}^{T}\overrightarrow{B}_{\omega}^{*}\underline{u} = \underline{0} \\ &\Rightarrow\; \left(\overrightarrow{B}_{\omega}^{*}\underline{u}\right)^{T}\left(\overrightarrow{B}_{\omega}^{*}\underline{u}\right) = \underline{0} \\ &\Rightarrow\; \overrightarrow{B}_{\omega}^{*}\underline{u} = \underline{0}. \end{aligned}$$

Hence, we also have

$$\mathbf{Null}\left(\mathcal{L}_{m}^{\mathcal{E},\omega}\right) \subseteq \mathbf{Null}\left(\overrightarrow{B}_{\omega}^{*}\right). \tag{C.3.2}$$

Combining (C.3.1) and (C.3.2), we have

$$\mathbf{Null}\left(\mathcal{L}_{m}^{\mathcal{E},\omega}\right) = \mathbf{Null}\left(\overrightarrow{B}_{\omega}^{*}\right). \tag{C.3.3}$$

Suppose we have a vector $\underline{y}$, then $\underline{y}^{T}\mathcal{L}_{m}^{\mathcal{E},\omega}\underline{y} = 0$ if and only if $\underline{y} \in \mathbf{Null}\left(\overrightarrow{B}_{\omega}^{*}\right)$, as we discussed above. The null space of the incidence matrix $\overrightarrow{B}_{\omega}^{*}$ is also called the flow (or cycle) space, whose dimension equals to the number of cycles (loops) in the associated graph G, see Godsil and Royle (2001) and Schaub and Segarra (2018). The dimension of the flow space obeys the following theorem (see **Theorem 14.2.1** in Godsil and Royle (2001)).

Theorem. Let G be a graph with n vertices, m edges, and c connected components, then the dimension of its flow space is $m - n + c$.

It is easy to see that for a connected tree graph G, the dimension of flow space is 0. Hence, we have

$$\mathbf{Null}\left(\overrightarrow{B}_{\omega}^{*}\right) = \{\underline{0}_{m}\}. \tag{C.3.4}$$

Hence, for any $\underline{y} \in \mathbb{R}^{m}\backslash\{\underline{0}_{m}\}$, we have $\underline{y}^{T}\mathcal{L}_{m}^{\mathcal{E},\omega}\underline{y} > 0$. Therefore, $\mathcal{L}_{m}^{\mathcal{E},\omega}$ is positive-definite.

C.4 Proof for Proposition 4.2.1

Suppose we have a $(p+q)\times(p+q)$ matrix M assumed to be (semi-) positive definite, which can be represented as a block partitioned matrix as

$$M = \left(\begin{array}{c|c} E & F \\ \hline F^T & H \end{array}\right), \tag{C.4.1}$$

where E, F, G, and H are $p\times p$, $p\times q$, $q\times p$, and $q\times q$ matrices, respectively. Recall that if H is invertible (and we further assume H is symmetric), then the Schur complement of E with respect to H, is given by

$$M/H = E - FH^{-1}F^T. \tag{C.4.2}$$

A block-partitioning matrix (**Section 1** in (Zhang; 2006)) contains M/H as one of the blocks can be obtained by

$$\begin{aligned} &\left(\begin{array}{c|c} \mathbf{I} & -FH^{-1} \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right)\left(\begin{array}{c|c} E & F \\ \hline F^T & H \end{array}\right)\left(\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline -H^{-1}F^T & \mathbf{I} \end{array}\right) \qquad \text{(C.4.3)} \\ &= \left(\begin{array}{c|c} E - FH^{-1}F^T & \mathbf{0} \\ \hline F^T & H \end{array}\right)\left(\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline -H^{-1}F^T & \mathbf{I} \end{array}\right) \\ &= \left(\begin{array}{c|c} E - FH^{-1}F^T & \mathbf{0} \\ \hline \mathbf{0} & H \end{array}\right) \\ &= \left(\begin{array}{c|c} M/H & \mathbf{0} \\ \hline \mathbf{0} & H \end{array}\right). \qquad \text{(C.4.4)} \end{aligned}$$

By the fact that H is symmetric, then we have $(-FH^{-1})^T = -H^{-1}F^T$. Then we can see that

$$\left(\begin{array}{c|c} \mathbf{I} & -FH^{-1} \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right) = \left(\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline -H^{-1}F^T & \mathbf{I} \end{array}\right)^T. \tag{C.4.5}$$

Then we have

$$\underline{y}^T M \underline{y} = \underline{x}^T \left(\begin{array}{c|c} M/H & \mathbf{0} \\ \hline \mathbf{0} & H \end{array}\right) \underline{x}, \tag{C.4.6}$$

where

$$\underline{y} = \left(\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline -H^{-1}F^T & \mathbf{I} \end{array}\right) \underline{x} \tag{C.4.7}$$

$$= \left(\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline -H^{-1}F^T & \mathbf{I} \end{array}\right) (\underline{x}_l^T, \underline{x}_r^T)^T$$

$$= (\underline{x}_l^T,\ (\underline{x}_r - H^{-1}F^T\underline{x}_l)^T)^T, \tag{C.4.8}$$

for any $\underline{x} \in \mathbb{R}^{p+q}$, $\underline{x}_l^T \in \mathbb{R}^p$, and $\underline{x}_r^T \in \mathbb{R}^q$. We can see that if $\underline{y} = \underline{0}_{p+q}$, we have that $\mathbf{x}_l = \underline{0}_p$ and $\underline{x}_r = H^{-1}F^T\underline{x}_l = \underline{0}_q$. Hence, we have

$$\underline{y} = \underline{0}_{p+q} \ \Rightarrow\ \underline{x} = \underline{0}_{p+q}, \tag{C.4.9}$$

From the above calculation, we also have

$$\underline{y}^T \left(\begin{array}{c|c} E & F \\ \hline F^T & H \end{array}\right) \underline{y} = \underline{x}^T \left(\begin{array}{c|c} M/H & \mathbf{0} \\ \hline \mathbf{0} & H \end{array}\right) \underline{x}$$

$$= (\underline{x}_l^T\,(M/H)\,\underline{x}_l,\ \underline{x}_r\,H\,\underline{x}_l) \tag{C.4.10}$$

Hence, immediately we can see that M/H is also (semi-) positive-definite. As a result, the Schur complement used in our work will keep the semi-positive/positive definite property of the stage-m edge Laplacian matrix.

Bibliography

Alt, H., Knauer, C. and Wenk, C. (2004). Comparison of distance measures for planar curves, *Algorithmica* **38**: 45–58.

Baker, M. and Faber, X. (2006). Metrized graphs, Laplacian operators, and electrical networks, *Contemporary Mathematics* **415**(15-34): 2.

Barbarossa, S., Sardellitti, S. and Ceci, E. (2018). Learning from signals defined over simplicial complexes, *2018 IEEE Data Science Workshop (DSW)*, IEEE, pp. 51–55.

Battiloro, C., Sardellitti, S., Barbarossa, S. and Di Lorenzo, P. (2023). Topological signal processing over weighted simplicial complexes, *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp. 1–5.

Bick, C., Gross, E., Harrington, H. A. and Schaub, M. T. (2023). What are higher-order networks?, *SIAM Review* **65**(3): 686–731.

Bollobás, B. (1998). *Modern Graph Theory*, Vol. 184, Springer Science & Business Media.

Bonchev, D. (1991). *Chemical Graph Theory: Introduction and Fundamentals*, Vol. 1, CRC Press.

Bondy, J. A. and Murty, U. S. R. (2008). *Graph Theory*, Springer Publishing Company, Incorporated.

Brouwer, A. E. and Haemers, W. H. (2011). *Spectra of Graphs*, Springer Science & Business Media.

Buisson, L., Blanc, L. and Grenouillet, G. (2008). Modelling stream fish species distribution in a river network: The relative effects of temperature versus physical factors, *Ecology of Freshwater Fish* **17**(2): 244–257.

Calderón, A. (1964). Intermediate spaces and interpolation, the complex method, *Studia Mathematica* **24**(2): 113–190.

Cao, D., Knight, M. I. and Nason, G. P. (2024). A multiscale method for data collected from network edges via the line graph, *arXiv preprint arXiv:2410.13693* .

Chang, J., Kolaczyk, E. D. and Yao, Q. (2022). Estimation of subgraph densities in noisy networks, *Journal of the American Statistical Association* **117**(537): 361–374.

Chen, P.-Y. and Liu, S. (2017). Bias-variance tradeoff of graph Laplacian regularizer, *IEEE Signal Processing Letters* **24**(8): 1118–1122.

Chudnovsky, M. and Seymour, P. D. (2005). The structure of claw-free graphs., *BCC*, pp. 153–171.

Chung, F. R. (1996). Laplacians of graphs and Cheeger's inequalities, *Combinatorics, Paul Erdos is Eighty* **2**(157-172): 13–2.

Chung, F. R. (1997). *Spectral Graph Theory*, Vol. 92, American Mathematical Society.

Cohen, A. and Daubechies, I. (1992). A stability criterion for biorthogonal wavelet bases and their related subband coding scheme.

Cohen, A., Daubechies, I. and Feauveau, J.-C. (1992). Biorthogonal bases of compactly supported wavelets, *Communications on Pure and Applied Mathematics* **45**(5): 485–560.

Cohen, A., Daubechies, I. and Vial, P. (1993). Wavelets on the interval and fast wavelet transforms, *Applied and Computational Harmonic Analysis* .

Coifman, R. R. and Donoho, D. L. (1995). Translation-invariant de-noising, *Wavelets and Statistics*, Springer, pp. 125–150.

Coifman, R. R. and Maggioni, M. (2006). Diffusion wavelets, *Applied and Computational Harmonic Analysis* **21**(1): 53–94.

Cressie, N., Frey, J., Harch, B. and Smith, M. (2006). Spatial prediction on a river network, *Journal of Agricultural, Biological, and Environmental Statistics* **11**: 127–150.

Crovella, M. and Kolaczyk, E. (2003). Graph wavelets for spatial traffic analysis, *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, Vol. 3, IEEE, pp. 1848–1857.

Cvetković, D., Rowlinson, P. and Simić, S. K. (2007). Signless Laplacians of finite graphs, *Linear Algebra and Its Applications* **423**(1): 155–171.

Daubechies, I. (1992). *Ten Lectures on Wavelets*, SIAM.

Deri, J. A. and Moura, J. M. (2015). Taxi data in New York city: A network perspective, *2015 49th Asilomar Conference on Signals, Systems and Computers*, IEEE, pp. 1829–1833.

Devriendt, K. (2022). Effective resistance is more than distance: Laplacians, simplices and the Schur complement, *Linear Algebra and Its Applications* **639**: 24–49.

Diestel, R. (2005). *Graph Theory 3rd ed*, Graduate Texts in Mathematics.

Donoho, D. L. (1997). Cart and best-ortho-basis: A connection, *The Annals of Statistics* **25**(5): 1870–1911.

Donoho, D. L. (2001). Sparse components of images and optimal atomic decompositions, *Constructive Approximation* **17**: 353–382.

Donoho, D. L. and Johnstone, I. M. (1994). Ideal spatial adaptation by wavelet shrinkage, *Biometrika* **81**(3): 425–455.

Donoho, D. L. and Johnstone, I. M. (1995). Adapting to unknown smoothness via wavelet shrinkage, *Journal of the American Statistical Association* **90**(432): 1200–1224.

Dorfler, F. and Bullo, F. (2012). Kron reduction of graphs with applications to electrical networks, *IEEE Transactions on Circuits and Systems I: Regular Papers* **60**(1): 150–163.

Dryden, I. L. and Mardia, K. V. (2016). *Statistical Shape Analysis: With Applications in R*, Vol. 995, John Wiley & Sons.

Erdős, P., Rényi, A. et al. (1960). On the evolution of random graphs, *Publ. Math. Inst. Hung. Acad. Sci* **5**(1): 17–60.

Evans, T. S. and Lambiotte, R. (2009). Line graphs, link partitions, and overlapping communities, *Physical Review E* **80**(1): 016105.

Fryzlewicz, P. and Nason, G. P. (2004). A Haar-Fisz algorithm for Poisson intensity estimation, *Journal of Computational and Graphical Statistics* **13**(3): 621–638.

Gallacher, K., Miller, C., Scott, E., Willows, R., Pope, L. and Douglass, J. (2017). Flow-directed PCA for monitoring networks, *Environmetrics* **28**(2): e2434.

Gavish, M., Nadler, B. and Coifman, R. R. (2010). Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi-supervised learning, *ICML*.

Girvan, M. and Newman, M. E. (2002). Community structure in social and biological networks, *Proceedings of the National Academy of Sciences* **99**(12): 7821–7826.

Godsil, C. and Royle, G. F. (2001). *Algebraic Graph Theory*, Vol. 207, Springer Science & Business Media.

Grady, L. J. and Polimeni, J. R. (2010). *Discrete Calculus: Applied Analysis on Graphs for Computational Science*, Vol. 3, Springer.

Grohs, P., Wiatowski, T. and Bölcskei, H. (2016). Deep convolutional neural networks on cartoon functions, *2016 IEEE International Symposium on Information Theory (ISIT)*, IEEE, pp. 1163–1167.

Haar, A. (1910). Zur theorie der orthogonalen funktionensysteme, *Math. Annal.* **69**: 331–371.

Hammond, D. K., Gur, Y. and Johnson, C. R. (2013). Graph diffusion distance: A difference measure for weighted graphs based on the graph Laplacian exponential kernel, *2013 IEEE Global Conference on Signal and Information Processing*, IEEE, pp. 419–422.

Hammond, D. K., Vandergheynst, P. and Gribonval, R. (2011). Wavelets on graphs via spectral graph theory, *Applied and Computational Harmonic Analysis* **30**(2): 129–150.

Harary, F. (2018). *Graph Theory (on Demand Printing Of 02787)*, CRC Press.

Higham, N. J. (2002). *Accuracy and Stability of Numerical Algorithms*, SIAM.

Hoef, J. M. V., Peterson, E. and Theobald, D. (2006). Spatial statistical models that use flow and stream distance, *Environmental and Ecological Statistics* **13**(4): 449–464.

Huttenlocher, D. P., Klanderman, G. A. and Rucklidge, W. J. (1993). Comparing images using the Hausdorff distance, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**(9): 850–863.

Jansen, M. and Bultheel, A. (1998). Smoothing non-equidistantly sampled data using wavelets and cross validation, *Proceedings of the IEEE Benelux Signal Processing Symposium*, Citeseer, pp. 111–114.

Jansen, M. H., Nason, G. and Silverman, B. (2004). Multivariate nonparametric regression using lifting, *Technical Report 04-17*, University of Bristol.

Jansen, M. H. and Oonincx, P. J. (2005). *Second Generation Wavelets and Applications*, Springer Science & Business Media.

Jansen, M., Nason, G. P. and Silverman, B. W. (2001). Scattered data smoothing by empirical Bayesian shrinkage of second-generation wavelet coefficients, *Wavelets: Applications in Signal and Image Processing IX*, Vol. 4478, International Society for Optics and Photonics, pp. 87–97.

Jansen, M., Nason, G. P. and Silverman, B. W. (2009). Multiscale methods for data on graphs and irregular multidimensional situations, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **71**(1): 97–125.

Jawerth, B. and Sweldens, W. (1994). An overview of wavelet based multiresolution analyses, *SIAM Review* **36**(3): 377–412.

Johnstone, I. M. and Silverman, B. W. (2004). Needles and straw in haystacks: Empirical Bayes estimates of possibly sparse sequences, *The Annals of Statistics* **32**(4): 1594–1649.

Karimi, D. and Salcudean, S. E. (2019). Reducing the Hausdorff distance in medical image segmentation with convolutional neural networks, *IEEE Transactions on Medical Imaging* **39**(2): 499–513.

Knight, M. I., Leeming, K., Nason, G. and Nunes, M. (2019). Generalised Network Autoregressive Processes and the GNAR package, *Journal of Statistical Software* .

Knight, M. I. and Nason, G. P. (2006). Improving prediction of hydrophobic segments along a transmembrane protein sequence using adaptive multiscale lifting, *Multiscale Modeling & Simulation* **5**(1): 116–129.

Knight, M. I. and Nason, G. P. (2009). A 'nondecimated' lifting transform, *Statistics and Computing* **19**(1): 1–16.

Knight, M. I., Nason, G. P. and Nunes, M. A. (2017). A wavelet lifting approach to long-memory estimation, *Statistics and Computing* **27**(6): 1453–1471.

Kolaczyk, E. D. and Csárdi, G. (2014). *Statistical Analysis of Network Data with R*, Vol. 65, Springer.

Kook, W. and Lee, K.-J. (2018). Simplicial networks and effective resistance, *Advances in Applied Mathematics* **100**: 71–86.

Kuchment, P. (2003). Quantum graphs: I. Some basic structures, *Waves in Random Media* **14**(1): S107.

Lakhina, A., Papagiannaki, K., Crovella, M., Diot, C., Kolaczyk, E. D. and Taft, N. (2004). Structural analysis of network traffic flows, *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems*, pp. 61–72.

Lim, L.-H. (2020). Hodge Laplacians on graphs, *Siam Review* **62**(3): 685–715.

Lucasius, C. B. and Kateman, G. (1993). Understanding and using genetic algorithms Part 1. Concepts, properties and context, *Chemometrics and Intelligent Laboratory Systems* **19**(1): 1–33.

Lucasius, C. B. and Kateman, G. (1994). Understanding and using genetic algorithms Part 2. Representation, configuration and hybridization, *Chemometrics and Intelligent Laboratory Systems* **25**(2): 99–145.

Mahadevan, N. (2010). *Multiscale, Multi-Dimensional Space and Space-Time Function Estimation for Irregular Network Data*, PhD thesis, University of Bristol.

Mallat, S. G. (1989a). Multiresolution approximations and wavelet orthonormal bases of $L^2(\mathbb{R})$, *Transactions of the American Mathematical Society* **315**(1): 69–87.

Mallat, S. G. (1989b). A theory for multiresolution signal decomposition: The wavelet representation, *IEEE Transactions on Pattern Analysis & Machine Intelligence* **11**(07): 674–693.

Mantziou, A., Cucuringu, M., Meirinhos, V. and Reinert, G. (2023). The GNAR-edge model: A network autoregressive model for networks with time-varying edge weights, *arXiv preprint arXiv:2305.16097* .

Merris, R. (1994). Laplacian matrices of graphs: A survey, *Linear Algebra and Its Applications* **197**: 143–176.

Meyer, Y. (1992). *Wavelets and Operators: Volume 1*, number 37, Cambridge University Press.

Nason, G. P. (2008). *Wavelet Methods in Statistics with R*, Vol. 574, Springer.

Nason, G. P. and Silverman, B. W. (1995). The stationary wavelet transform and some statistical applications, *Wavelets and Statistics*, Springer, pp. 281–299.

Newman, M. E. (2003). The structure and function of complex networks, *SIAM Review* **45**(2): 167–256.

Nunes, M. A. (2006). *Some New Multiscale Methods for Curve Estimation and Binomial Data*, PhD thesis, University of Bristol.

Nunes, M. A., Knight, M. I. and Nason, G. P. (2006). Adaptive lifting for nonparametric regression, *Statistics and Computing* **16**(2): 143–159.

Ortega, A., Frossard, P., Kovačević, J., Moura, J. M. and Vandergheynst, P. (2018). Graph signal processing: Overview, challenges, and applications, *Proceedings of the IEEE* **106**(5): 808–828.

O'Searcoid, M. (2006). *Metric Spaces*, Springer Science & Business Media.

Park, S., Oh, H.-S. et al. (2022). Lifting scheme for streamflow data in river networks, *Journal of the Royal Statistical Society Series C* **71**(2): 467–490.

Percival, D. P. (1995). On estimation of the wavelet variance, *Biometrika* **82**(3): 619–631.

Pereyra, M. C. and Ward, L. A. (2012). *Harmonic analysis: From Fourier to wavelets*, Vol. 63, American Mathematical Society.

Pickands, J. (1967). Maxima of stationary Gaussian processes, *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* **7**: 190–223.

Popescul, A. and Ungar, L. H. (2003). Statistical relational learning for link prediction, *IJCAI Workshop on Learning Statistical Models from Relational Data*, Vol. 2003.

Robinson, M. (2014). *Topological Signal Processing*, Vol. 81, Springer.

Roman, S., Axler, S. and Gehring, F. (2005). *Advanced Linear Algebra*, Vol. 3, Springer.

Roussopoulos, N. D. (1973). A $\max\{m, n\}$ algorithm for determining the graph H from its line graph G, *Information Processing Letters* **2**(4): 108–112.

Sardellitti, S. and Barbarossa, S. (2024). Topological signal processing over generalized cell complexes, *IEEE Transactions on Signal Processing* .

Schaub, M. T., Benson, A. R., Horn, P., Lippner, G. and Jadbabaie, A. (2020). Random walks on simplicial complexes and the normalized Hodge 1-Laplacian, *SIAM Review* **62**(2): 353–391.

Schaub, M. T. and Segarra, S. (2018). Flow smoothing and denoising: Graph signal processing in the edge-space, *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, IEEE, pp. 735–739.

Schaub, M. T., Zhu, Y., Seby, J.-B., Roddenberry, T. M. and Segarra, S. (2021). Signal processing on higher-order networks: Livin'on the edge... and beyond, *Signal Processing* **187**: 108149.

Schröder, P. and Sweldens, W. (1995a). Spherical wavelets: Efficiently representing functions on the sphere, *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pp. 161–172.

Schröder, P. and Sweldens, W. (1995b). Spherical wavelets: Texture processing, *Rendering Techniques' 95: Proceedings of the Eurographics Workshop in Dublin, Ireland, June 12–14, 1995 6*, Springer, pp. 252–263.

Severn, K. E., Dryden, I. L. and Preston, S. P. (2021). Non-parametric regression for networks, *Stat* **10**(1): e373.

Sevi, H., Rilling, G. and Borgnat, P. (2023). Harmonic analysis on directed graphs and applications: From Fourier analysis to wavelets, *Applied and Computational Harmonic Analysis* **62**: 390–440.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A. and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, *IEEE Signal Processing Magazine* **30**(3): 83–98.

Shuman, D. I., Ricaud, B. and Vandergheynst, P. (2016). Vertex-frequency analysis on graphs, *Applied and Computational Harmonic Analysis* **40**(2): 260–291.

Simoens, J. and Vandewalle, S. (2003). A stabilized lifting construction of wavelets on irregular meshes on the interval, *SIAM Journal on Scientific Computing* **24**(4): 1356–1378.

Smola, A. J. and Kondor, R. (2003). Kernels and regularization on graphs, *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings*, Springer, pp. 144–158.

Spielman, D. A. (2007). Spectral graph theory and its applications, *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, IEEE, pp. 29–38.

Spielman, D. A. (2012). Spectral graph theory, *Combinatorial Scientific Computing* **18**.

Stanković, L., Mandic, D., Daković, M., Scalzo, B., Brajović, M., Sejdić, E. and Constantinides, A. G. (2020). Vertex-frequency graph signal processing: A comprehensive review, *Digital Signal Processing* **107**: 102802.

Starck, J.-L., Murtagh, F. D. and Bijaoui, A. (1998). *Image Processing and Data Analysis: The Multiscale Approach*, Cambridge University Press.

Strang, G. and Nguyen, T. (1996). *Wavelets and Filter Banks*, SIAM.

Sweldens, W. (1995). Lifting scheme: A new philosophy in biorthogonal wavelet constructions, *Wavelet Applications in Signal and Image Processing III*, Vol. 2569, International Society for Optics and Photonics, pp. 68–79.

Sweldens, W. (1996a). Compactly supported wavelets which are biorthogonal with respect to a weighted inner product, *Proceedings of the 14th IMACS World Congress*.

Sweldens, W. (1996b). The lifting scheme: A custom-design construction of biorthogonal wavelets, *Applied and Computational Harmonic Analysis* **3**(2): 186–200.

Sweldens, W. (1998). The lifting scheme: A construction of second generation wavelets, *SIAM Journal on Mathematical Analysis* **29**(2): 511–546.

Sweldens, W. and Schröder, P. (2005). Building your own wavelets at home, *Wavelets in the Geosciences* pp. 72–107.

Taskar, B., Wong, M.-F., Abbeel, P. and Koller, D. (2003). Link prediction in relational data, *Advances in Neural Information Processing Systems* **16**.

Trefethen, L. N. and Bau III, D. (1997). *Numerical Linear Algebra*, Vol. 50, SIAM.

Tu, L. W. (2011). Manifolds, *An Introduction to Manifolds*, Springer, New York, pp. 47–83.

Vidakovic, B. (2009). *Statistical Modeling by Wavelets*, Vol. 503, John Wiley & Sons.

Young, N. (1988). *An Introduction to Hilbert Space*, Cambridge University Press.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups, *Journal of Anthropological Research* **33**(4): 452–473.

Zelazo, D. and Mesbahi, M. (2010). Edge agreement: Graph-theoretic performance bounds and passivity analysis, *IEEE Transactions on Automatic Control* **56**(3): 544–555.

Zelazo, D., Rahmani, A. and Mesbahi, M. (2007). Agreement via the edge Laplacian, *2007 46th IEEE Conference on Decision and Control*, IEEE, pp. 2309–2314.

Zhang, B., Fadili, M., Starck, J.-L. and Digel, S. W. (2008). Fast Poisson noise removal by biorthogonal Haar domain hypothesis testing, *Statistical Methodology* **5**(4): 387–396.

Zhang, F. (2006). *The Schur Complement and Its Applications*, Vol. 4, Springer Science & Business Media.

Zhang, Y., Ge, Z., Greenberg, A. and Roughan, M. (2005). Network anomography, *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, pp. 30–30.

Zhang, Y., Roughan, M., Lund, C. and Donoho, D. (2003). An information-theoretic approach to traffic matrix estimation, *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 301–312.

Zhao, C., Shi, W. and Deng, Y. (2005). A new Hausdorff distance for image matching, *Pattern Recognition Letters* **26**(5): 581–586.

Zhou, D. and Burges, C. J. (2008). High-order regularization on graphs, *Proceedings of the 6th International Workshop on Mining and Learning with Graphs*.