COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

ENSAI École nationale de la statistique et de l'analyse de l'information

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE DE LA STATISTIQUE ET DE L'ANALYSE DE L'INFORMATION

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *Mathématiques et leurs Interactions*

Par
Elie CHEDEMAIL

Débruitage de signaux définis sur des graphes de grande taille avec application à la confidentialité différentielle

Thèse présentée et soutenue à Bruz, le 21 décembre 2023
Unité de recherche : Crest-Ensai (UMR CNRS 9194) et Orange Innovation

Rapporteurs avant soutenance :

Pierre BORGNAT Directeur de recherche, ENS Lyon
David SHUMAN Professeur, Franklin W. Olin College of Engineering

Composition du Jury :

Président :	Jalal FADILI	Professeur, Ensicaen
Examinateurs :	Pierre BORGNAT	Directeur de recherche, ENS Lyon
	Salima EL KOLEI	Enseignante-chercheuse, Crest-Ensai
	Erwan LE PENNEC	Professeur, École polytechnique
	David SHUMAN	Professeur, Franklin W. Olin College of Engineering
Directeur de thèse :	Fabien NAVARRO	Maître de conférence, Université Paris 1 Panthéon-Sorbonne
Co-encadrants de thèse :	Basile DE LOYNES	Enseignant-chercheur, Crest-Ensai
	Baptiste OLIVIER	Ingénieur de recherche, IBM

Remerciements

En commençant mon voyage éducatif, je n'avais pas de destination en tête. Mon itinéraire s'est dessiné progressivement au gré de rencontres et d'expériences qui ont nourri ma curiosité, affiné mes désirs et forgé ma volonté. De mes premiers pas en terres mathématiques jusqu'aux contrées inexplorées de la statistique, je mesure à présent le chemin parcouru et souhaite remercier toutes les personnes rencontrées en route.

En premier lieu, j'éprouve une profonde reconnaissance à l'égard de mon directeur et de mes co-encadrants de thèse, Fabien Navarro, Baptiste Olivier et Basile de Loynes. Merci Fabien de m'avoir offert l'opportunité de réaliser un doctorat alors que je faisais mon stage de fin d'études. Je t'avais fait part de cette envie et tu as conçu avec Baptiste et Basile un sujet de recherche correspondant en tout point à mes capacités et ambitions. Cette attention et générosité qui te caractérisent m'ont été d'une grande aide tout au long de ce travail. Merci Baptiste de m'avoir accueilli à Orange et encadré au quotidien avec une grande disponibilité, des idées clairvoyantes et de précieux conseils. Merci Basile de m'avoir transmis ton sens de la précision juste et pour ton soutien continu pendant cette dernière année de rédaction. Vous trois avez su me guider et m'aider à tenir la route jusqu'à l'arrivée, je vous en remercie à nouveau.

Je tiens à remercier les membres du jury de ma soutenance de thèse pour leur temps, la pertinence de leurs remarques et la qualité de notre échange. Merci à Pierre Borgnat et David Shuman d'avoir accepté de rapporter cette thèse, Jalal Fadili d'avoir présidé ce jury et examiné mon travail aux côtés de Erwan Le Pennec et Salima El Kolei qui s'est aussi impliquée dans mon comité de suivi individuel avec Christophe Chesneau que je remercie également.

Au sein d'Orange où j'ai réalisé une grande partie de mon doctorat, je me dois de remercier Adam Ouorou et Françoise Fessant d'avoir participé à mon encadrement et l'ensemble de l'équipe NCP pour leur accueil chaleureux. Je pense en particulier aux divers étudiants et apprentis, Rémi, Adina, Marcel, Zaki, Elie, Corentin, Thomas, Anne, Héloïse, Despoina, Astrid et Divi que j'ai eu le plaisir d'encadrer pour son stage de fin d'études. Chaque jour passé auprès de vous était l'occasion de découvrir de nouveaux univers et la garantie de poursuivre mon chemin en bonne compagnie.

J'exprime toute ma gratitude aux personnes qui composent l'Ensai, mon autre lieu de travail et celui de ma formation initiale d'ingénieur. Un grand merci à l'ensemble de l'équipe pédagogique, du personnel administratif et tout particulièrement Cécile Terrien pour sa disponibilité précieuse et son travail d'accompagnement des doctorants. Ces derniers ont été d'excellents camarades et des soutiens formidables toutes ces années. Steven, Amandine, Camille G, Edouard, Max, Daphné, Sunny, Omar, Hassan, Guillaume Fl, Guillaume Fr et Camille M, je vous souhaite le meilleur et bonne route.

Mon désir de faire de la recherche est né aux côtés de personnes m'ayant fait confiance pour les accompagner dans le cadre de stages de recherche. Je veux donc remercier une fois de plus Audrey Lavenu, Guillaume Bouzillé, Richard Wilkinson ainsi que les doctorants d'alors, Marc, Ségolène, Noureddine, Canelle et Mark. Ces chemins de traverse empruntés avec vous m'ont donné envie de marcher dans vos pas.

Si j'ai pu réaliser ce parcours, c'est aussi grâce aux enseignants des universités de Rennes 1 et Rennes 2 qui m'ont formé en licence. Merci notamment à Magalie Fromont Renoir et Nicolas Jégou d'avoir élargi mon horizon en m'ouvrant la voie de la statistique. De même, ma reconnaissance s'adresse naturellement à mes professeurs des écoles, de collège et de lycée pour leur travail et leur dévouement. Je tiens tout spécialement à remercier Messieurs Eugène et Zaidouni de m'avoir appris à marcher sur les sentiers des mathématiques et offert la capacité de choisir ma route. Grâce à vous, le monde m'est possible.

Enfin, je remercie passionnément ma famille et mes amis pour leur encouragement et réconfort jour après jour. Ce voyage a commencé et se poursuit avec vous.

"Une fois de plus, nos valises cabossées s'empilaient sur le trottoir ; on avait du chemin devant nous. Mais qu'importe : la route, c'est la vie." — Jack Kerouac, *Sur la route (1957)*

Table of Contents

List of Figures

List of Tables

RÉSUMÉ EN FRANÇAIS

Les données acquises à partir de systèmes interactifs à grande échelle, tels que les réseaux informatiques, écologiques, sociaux, financiers ou biologiques, sont de plus en plus répandues et accessibles. Au sein de l'apprentissage automatique moderne, la représentation, le traitement ou l'analyse efficaces de ces données structurées à grande échelle à l'aide de graphes ou de réseaux sont quelques-uns des problèmes clés [99, 16]. Le domaine émergent du traitement du signal sur graphe met en évidence les liens entre les domaines que sont le traitement du signal et la théorie spectrale des graphes [115, 103], tout en établissant des passerelles pour relever ces défis. En effet, le traitement du signal sur graphe a conduit à de nombreuses applications dans le domaine de l'apprentissage automatique : réseau de neurones convolutifs (CNN, *convolutional neural networks*) sur graphe [18, 68, 33], classification semi-supervisée avec CNN sur graphe [77, 64] ou détection de communautés [127], pour n'en citer que quelques-unes. Nous renvoyons le lecteur à [38] pour une analyse récente offrant de nouvelles perspectives sur le traitement du signal sur graphe pour l'apprentissage automatique, dont son rôle important dans certaines des premières conceptions d'architectures de réseaux de neurones sur graphe (GNN, *graph neural networks*). En outre, l'étude récente de [54] montre que les GNN populaires conçus d'un point de vue spectral, tels que les CNN spectraux sur graphe ou les réseaux d'attention sur graphe, résolvent implicitement des problèmes de débruitage de signal sur graphe.

Au cours des dernières décennies, la représentation creuse dans un repère a joué un rôle fondamental dans de nombreux domaines tels que la compression et la restauration de signaux, l'analyse de données et le traitement du signal sur graphe en général. En effet, les représentations surcomplètes telles que les repères d'ondelettes présentent plusieurs avantages et offrent plus de flexibilité que les bases orthonormées. Une famille représentative de systèmes surcomplets dérivée des ondelettes de diffusion (*diffusion wavelets*) orthonormées de [27] est la transformée en ondelettes spectrales sur graphe (SGWT, *spectral graph wavelet transform*) de [65] construite à partir d'un repère d'ondelettes général. Dans un contexte de débruitage, la SGWT a récemment été adaptée par [58] pour former un repère ajusté en utilisant la décomposition de Littlewood-Paley inspirée par [28]. En se basant sur la SGWT, les auteurs de [86] ont proposé un calibrage automatique du paramètre

de seuil en adaptant l'estimateur sans biais du risque de Stein (SURE, *Stein's unbiased risk estimate*) pour un signal bruité défini sur un graphe et décomposé dans un repère ajusté d'ondelettes donné. Même si ce critère de sélection produit des estimateurs efficaces de l'erreur quadratique moyenne inconnue, la principale difficulté est la nécessité d'une décomposition complète de la matrice laplacienne, ce qui est rédhibitoire pour les graphes à grande échelle.

Nous proposons dans cette thèse d'étendre cette méthodologie aux graphes creux de grande taille en évitant cette décomposition afin d'élargir son champ d'application. Différentes stratégies ont été proposées dans le contexte du traitement du signal sur graphe, l'une des plus populaires étant basée sur l'approximation par polynômes de Chebyshev [65]. Cependant, même si cette dernière constitue un bon choix dans de nombreuses situations, les approximations des fonctions discontinues ou non périodiques souffrent du phénomène de Gibbs. Une solution simple couramment utilisée en traitement du signal sur graphe [114] pour réduire les éventuelles oscillations parasites sans coût de calcul supplémentaire est l'introduction de coefficients d'amortissement de Jackson [73, 34], qui permet des ordres d'approximation plus élevés.

Comme le SURE peut être évalué dans le domaine des ondelettes, son calcul bénéficie directement de ces approximations numériques efficaces. Pour qu'il soit adapté aux graphes creux de grande taille, la seule difficulté est le calcul de poids apparaissant dans son expression. En effet, puisque la SGWT n'est pas orthogonale, un bruit blanc gaussien dans le domaine du graphe est transformé en un bruit corrélé, ce qui introduit une pondération par cette covariance dans le terme de divergence du SURE. Ce dernier nécessite donc le calcul explicite et le stockage du repère pour être calculé. Inspirée par l'estimation de la corrélation entre les ondelettes centrées sur différents sommets proposée dans [127], notre contribution consiste à tirer parti de l'interprétation des poids du SURE comme la covariance entre les transformées en ondelettes de signaux aléatoires afin de les estimer avec une approximation de Monte-Carlo. Nous insérons ensuite cet estimateur de poids dans la formule du SURE pour en obtenir un estimateur qui s'adapte bien aux signaux sur graphes de grande taille. En outre, nous fournissons des expressions pour la variance de nos estimateurs proposés et montrons que le tirage d'échantillon de Monte-Carlo à partir de la loi de Rademacher centrée produit une meilleure convergence en comparaison avec la loi gaussienne standard. Notre approche est conforme à d'autres méthodes [107, 138] qui utilisent également l'approximation de Monte-Carlo, mais estiment seulement le terme de divergence du SURE dans le cas d'un bruit non corrélé.

La méthode que nous proposons permet de réduire le bruit de tout signal défini sur un graphe, y compris les images [115] et les maillages 3D [100] qui peuvent comporter un grand nombre de sommets. Nous nous concentrons ici sur une application intéressante en matière de confidentialité différentielle [46, 43, 44] dont l'objectif est de protéger les données sensibles utilisées par les algorithmes. Certains travaux de la littérature se sont intéressés à la protection des sommets [76] et des arêtes [83] d'un graphe dans ce cadre. Pour un signal sur graphe, de telles garanties de confidentialité sont généralement obtenues en lui ajoutant du bruit blanc, ce qui réduit inévitablement son utilité statistique puisque les informations pertinentes qu'il contient sont perturbées. Cette utilité peut être partiellement restaurée par débruitage à condition qu'aucune information sur le signal original ne soit utilisée, des techniques de lissage sur graphe ont d'ailleurs déjà été employées dans ce but [7]. Comme la méthodologie que nous proposons ne dépend que des données observées, elle se prête bien à cette utilisation pour les signaux sur graphes que nous intégrons dans notre application numérique. Celle-ci permet d'évaluer notre estimateur Monte-Carlo du SURE et ses poids, ainsi que la méthodologie globale de débruitage sur des graphes de petite et de grande taille. En résumé, les contributions de cette thèse sont les suivantes :

— Premièrement, nous proposons un estimateur des poids du SURE grâce à leur interprétation en terme de covariance entre transformées en ondelettes de signaux aléatoires sur graphes. Notre approche combinant la transformée rapide basée sur l'approximation par polynômes de Chebyshev avec l'estimation par Monte-Carlo permet de contourner la décomposition de la matrice laplacienne du graphe coûteuse en ressources de calcul. Une fois estimés, nous utilisons ces poids pour construire un estimateur du SURE qui s'adapte bien aux graphes de grande taille.
— Ensuite, nous explicitons les variances de ces deux estimateurs et mettons en évidence une meilleure convergence lorsque l'échantillon de Monte-Carlo provient d'une loi de Rademacher en comparaison avec la loi gaussienne. Nous complétons ce résultat théorique avec une illustration numérique.
— Enfin, nous réalisons une évaluation expérimentale de la méthodologie proposée de débruitage de signal sur graphe en montrant ses performances sur des données réelles protégées par confidentialité différentielle et sur des signaux simulés sur graphes de grande taille.

Cette thèse s'inscrit dans le travail de recherche du groupe *Orange* qui a choisi la cybersécurité comme un de ses axes stratégiques de développement. En effet, les cybermenaces connaissent un développement rapide ces dernières décennies notamment du fait

des nouvelles capacités de calcul et de l'adoption des techniques de la science des données dans ce domaine. Cet opérateur et fournisseur de services a donc pour ambition d'être à la pointe des technologies liées à l'analyse des données pour la sécurité de ses infrastructures et des données de ses clients. En accord avec le règlement général sur la protection des données (RGPD), Orange doit par ailleurs développer cette expertise et ces nouvelles technologies dans le respect des données personnelles de ses clients.

Dans cette optique, il a déjà entamé divers projets de recherche avec sa division *Orange Innovation*. Ses travaux portent notamment sur la protection des données de mobilité [52, 19] et la synthétisation de données [12] dans le cadre de la confidentialité différentielle.

Plan de la thèse

Le Chapitre 1 est consacré à la présentation du domaine du traitement du signal sur graphe et plus précisément à son application dans la réduction du bruit. Dans une première section, nous rappelons les définitions classiques du graphe et des signaux définis sur des graphes, présentons la transformée de Fourier sur graphe et enfin passons en revue la transformée en ondelettes particulière dans le contexte des graphes qu'est la SGWT. Puis dans une deuxième section, nous exposons une procédure de débruitage reposant sur le seuillage des coefficients d'ondelettes de cette transformée, présentons différents processus de seuillage et quelques méthodes de sélection de paramètres de seuillage dont celle de la minimisation du SURE qui nous intéresse spécifiquement.

Le Chapitre 2 porte sur le champ de recherche de la confidentialité différentielle. Nous introduisons en première section cette formalisation du concept de confidentialité dans le contexte des jeux de données statistiques. En particulier, nous présentons différentes définitions de confidentialité différentielle, quelques mécanismes permettant de l'atteindre ainsi que des propriétés intéressantes sur la composition de ces derniers. La section suivante s'intéresse à l'utilisation de la confidentialité différentielle dans le contexte industriel et son application possible aux graphes.

Le Chapitre 3 de ce manuscrit présente la contribution de cette thèse qui propose une solution au problème de complexité de calcul du SURE dans le cadre du seuillage de coefficients d'ondelettes sur graphe. Les résultats qui lui sont associés ont fait l'objet d'une publication dans le journal *IEEE Transactions on Signal and Information Processing over Networks* [23].

Chapter 1

Graph signal processing and application to denoising

We introduce in this chapter the domain of graph signal processing and its application in noise reduction. In this first section, we recall classical definitions of graphs and graph signals, present the graph analogue of the Fourier transform and finally review the spectral graph wavelet transform that will be of interest in this dissertation.

1.1 Graph signal processing

1.1.1 Graphs and graph signals

Graph definition

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a *graph* consisting of a finite set of *vertices* $\mathcal{V}$ connected together by *edges* from a set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. If there exists an edge between two vertices, they are said to be *adjacent* to each other. An edge that connects a vertex to itself is called a *loop*. Unless specified otherwise, we consider graphs without loops in this dissertation. The number of vertices is $n = |\mathcal{V}|$ and we number each vertex $i \in \mathcal{V}$ of the graph from 1 to n.

A *weighted graph* is a graph equipped with a *weight* function $w : \mathcal{E} \to \mathbb{R}$ that assigns to each edge (i, j) a weight w_{ij}. These can be gathered in a square matrix $\mathbf{W}$ of size n called the *weight matrix* whose entries are given by $W_{ij} = w_{ij}$ if the vertices i and j are adjacent, and $W_{ij} = 0$ otherwise. The diagonal elements of the weight matrix are zero for graphs without loops. When a graph has no edge weights, the graph edges are represented by entries $W_{ij} = 1$ and $\mathbf{W}$ is called the *adjacency matrix*. Hereinafter, we indistinctly write $(w_{ij})_{i,j=1,\dots,n}$ to refer to the weight matrix entries.

Graph edges that have an orientation from one vertex i to another j are called *directed edges* and denoted by an ordered pair of vertices (i, j). If a graph contains at least one such edge, it is known as a *directed graph*. This dissertation only covers *undirected graphs* which

are exclusively composed of symmetric edges $\{i, j\}$ that can be seen as edges connecting vertices in both directions. Graphs of this kind have symmetric edge weights $w_{ij} = w_{ji}$ and thus a symmetric weight matrix $\mathbf{W}$. The transformation $\mathbf{W}_{\text{sym}} = \frac{1}{2}(\mathbf{W} + \mathbf{W}^*)$ is commonly used to symmetrize a directed graph.

A *subgraph* $\mathcal{H} = (\mathcal{V}', \mathcal{E}')$ of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph whose sets of vertices and edges are subsets of those of $\mathcal{G}$. One way to form a subgraph is by choosing $\mathcal{V}' \subseteq \mathcal{V}$ and taking all the edges from $\mathcal{E}$ that connect pairs of vertices in $\mathcal{V}'$, that is, $\mathcal{E}' = \mathcal{E} \cap (\mathcal{V}' \times \mathcal{V}')$. A sequence of edges which joins a sequence of distinct vertices is called a *path*. Undirected graphs are *connected* if a path exists between every pair of vertices. The vertex and edge sets of any graph can be partitioned into a unique set of *components* which are the largest connected subgraphs of the graph. Thus, connected graphs have exactly one component and we only consider these throughout this dissertation for the sake of simplicity. An example of such a graph with symmetric, weighted edges is shown on Figure 1.1. All graph figures in this dissertation are produced with the **NetworkX** [62] Python package.

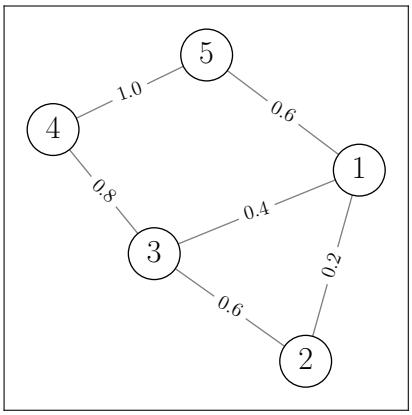


Figure 1.1 – Undirected, connected, weighted graph example

The *degree* $d_i = \sum_{j=1}^{n} w_{ij}$ of a vertex i is the sum of the weights assigned to the edges that connect to it. For unweighted graphs, it corresponds to the number of vertices adjacent to i. We define the *degree matrix* as the matrix whose diagonal elements are the degrees of the graph: $\mathbf{D} = \text{diag}(d_1, \ldots, d_n)$.

Graph construction

In many practical situations, the weight matrix $\mathbf{W}$ is not known from theory, but can be constructed from data points. Here, we illustrate some examples of such constructions.

If each vertex is associated with additional data in the form of space coordinates or feature values, the Euclidean distance $\mathrm{d}(i,j)$ or another one can be computed between each pair of vertices. This distance can then be used to decide whether to draw an edge between two vertices and choose its assigned weight value.

Given a vertex set, selecting edges is commonly done by considering that two distant vertices from one another have little to no influence on each other. One straightforward method is to draw an edge between vertices whose distance from each other is smaller than a threshold κ:

$$w_{ij} = \begin{cases} 1 & \text{if } \mathrm{d}(i,j) \leq \kappa \\ 0 & \text{otherwise.} \end{cases}$$

A second way consists in connecting each vertex i to its k-nearest neighbors $k\mathrm{NN}(i)$ [37]. Since one vertex can belong to the neighbors of another but not vice versa, symmetric edges are obtained by viewing two vertices as adjacent if at least one does:

$$w_{ij} = \begin{cases} 1 & \text{if } i \in k\mathrm{NN}(j) \text{ or } j \in k\mathrm{NN}(i) \\ 0 & \text{otherwise.} \end{cases}$$

In both methods, the choice of the hyperparameter κ or k is decisive as it directly affects the weight matrix sparsity, that is, its proportion of zero elements. Indeed, we will see in this section that signal processing on sparse graphs benefits from reduced computational complexity.

Different approaches exist to define the edge weight values depending on the application. For instance, if the vertices of the graph have space coordinates, the edge weights can directly represent the distance between them: $w_{ij} = \mathrm{d}(i,j)$. This definition is commonly used in geometric applications such as transportation networks or polygon meshes. An inverse approach is to view the edge weights as the strength of connection between the vertices like in communication or social networks for example. This interpretation in terms of similarity can be achieved with a decreasing function of the distance in the feature space [75]. A common choice for this method is the Gaussian kernel: $w_{ij} = \exp\left(-\frac{\mathrm{d}(i,j)^2}{2\alpha^2}\right)$, with $\alpha > 0$ the kernel bandwidth parameter. It produces edge weights whose values range from 0 to 1 and is therefore easily interpreted as a similarity measure. An additional motivation for the Gaussian kernel is its popular use on manifolds [74] which are usually approximated with a graph associated to a sampled point cloud [10]. More graph construction methods for defining both edges and their weights can be found in [60, Chapter 4].

Note that each approach yields non-negative weights that are preferable to work with in graph signal processing. In some applications, negative edge weights can represent the strength of repulsion between vertices such as antagonistic interactions in a network [3] for instance. However, many methods in this domain involve the eigendecomposition of matrices derived from the weight matrix $\mathbf{W}$ for which more theoretical results presented below exist when $w_{ij} \geq 0$. For this reason and the sake of simplicity, we only consider real non-negative edge weights in this dissertation.

Graph Laplacian matrices

The Laplacian matrix and its variants are other matrix representations of a graph whose respective definitions combine the weight matrix $\mathbf{W}$ and the degree matrix $\mathbf{D}$. They play an essential role in the understanding of the graph they are defined from. In particular, their eigenvalues and eigenvectors are studied in the field of spectral graph theory [25, 120] to reveal the principal properties and structure of the graph.

The most simple one is the unnormalized (or combinatorial) *Laplacian matrix* defined as the difference

$$\mathcal{L} = \mathbf{D} - \mathbf{W},$$

whose elements are given by

$$\mathcal{L}_{ij} = \begin{cases} d_i & \text{if } i = j \\ -w_{ij} & \text{otherwise.} \end{cases}$$

Since we are interested in undirected graphs, the graph Laplacian $\mathcal{L}$ is a real symmetric matrix. It is therefore diagonalizable with real eigenvalues and orthonormal eigenvectors that we denote $\lambda_1, \ldots, \lambda_n$ and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$, respectively.

For any vector $\boldsymbol{x} \in \mathbb{R}^n$ and with $w_{ij} \geq 0$, the quadratic form it defines is non-negative:

$$\boldsymbol{x}^\top \mathcal{L} \boldsymbol{x} = \sum_{\{i,j\} \in \mathcal{E}} w_{ij} (x(i) - x(j))^2 \geq 0.$$

As a result, $\mathcal{L}$ is positive semi-definite and thus its eigenvalues are all non-negative. Note that non-negative weights are sufficient but not necessary for the spectrum to be non-negative. In addition, the smallest eigenvalue is zero with multiplicity equal to the number of components of the graph [17, Proposition 1.3.7]. Since we consider connected graphs, zero appears only once in the spectrum. Finally, the authors of [31] show the largest

eigenvalue is bounded from above by twice the maximum degree of the graph: $d_{\max} = \max_i d_i$. We summarize these bounds with the following ordering:

$$0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_n \leq 2d_{\max}.$$

Among the many properties from spectral graph theory worth mentioning, the eigenvector $\boldsymbol{v}_1$ associated with the null eigenvalue λ_1 is constant and valued $\frac{1}{\sqrt{n}}$ at each vertex. Also, the second smallest eigenvalue λ_2 is known as the *algebraic connectivity* of $\mathcal{G}$ [51] whose value is related to the connectedness of the graph. If we gather the eigenvalues and eigenvectors in the respective matrices $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ and $\mathbf{V} = (\boldsymbol{v}_1 | \ldots | \boldsymbol{v}_n)$, the eigendecomposition of the graph Laplacian is given by

$$\mathcal{L} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top.$$

In some applications, it is preferable to normalize the Laplacian matrix such that its diagonal elements are all equal to one. This can be done by dividing each weight w_{ij} by the vertex degrees $\sqrt{d_i d_j}$ which defines the *normalized Laplacian matrix*:

$$\begin{aligned}\mathcal{L}^{\text{norm}} &= \mathbf{D}^{-\frac{1}{2}}\mathcal{L}\mathbf{D}^{-\frac{1}{2}} \\ &= \mathbf{I}_n - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}.\end{aligned}$$

Its elements are given by

$$\mathcal{L}^{\text{norm}}_{ij} = \begin{cases} 1 & \text{if } i = j \\ -\frac{w_{ij}}{\sqrt{d_i d_j}} & \text{otherwise.} \end{cases}$$

This symmetric matrix is well-defined as long as the graph contains no isolated vertices whose degrees are zero. Moreover, non-negative edge weights ensures all degrees are real positive and the normalized Laplacian is a real matrix. It is therefore unitarily diagonalizable with non-negative real eigenvalues $\mu_1, \ldots, \mu_n$ associated with eigenvectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$. The multiplicity of zero is equal to the number of components as with the unnormalized Laplacian matrix and the largest eigenvalue is always smaller than two [25, Lemma 1.7]. This gives the ordering

$$0 = \mu_1 < \mu_2 \leq \cdots \leq \mu_n \leq 2,$$

such that the matrices $\mathbf{M} = \mathrm{diag}(\mu_1, \ldots, \mu_n)$ and $\mathbf{U} = (\boldsymbol{u}_1 | \ldots | \boldsymbol{u}_n)$ produce the spectral decomposition

$$\mathcal{L}^{\mathrm{norm}} = \mathbf{U}\mathbf{M}\mathbf{U}^\top.$$

Both the Laplacian matrix $\mathcal{L}$ and its normalized variant $\mathcal{L}^{\mathrm{norm}}$ can be used in graph signal processing methods based on the graph spectrum. There is no clear indication as to which to choose since they each have their own advantages and disadvantages. For instance, the eigenvector $\boldsymbol{v}_1$ of $\mathcal{L}$ associated with the zero eigenvalue being constant is helpful to extend intuitions about the direct current (DC) component from the theory of classical signal processing [115], that is, the average value of a signal: $\forall \boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x}^\top \boldsymbol{v}_1 = \frac{1}{\sqrt{n}} \sum_{i=1}^n x(i) = \sqrt{n}\bar{\boldsymbol{x}}$. On the other hand, the spectrum of $\mathcal{L}^{\mathrm{norm}}$ benefits from an invariant upper bound that eases the comparison of a given method over graphs of varying maximum degrees.

An alternative normalization of the Laplacian matrix is to multiply it by the inverse of the degree matrix on the left, doing so yields the *random walk Laplacian matrix*

$$\begin{aligned}\mathcal{L}^{\mathrm{rw}} &= \mathbf{D}^{-1}\mathcal{L} \\ &= \mathbf{I}_n - \mathbf{D}^{-1}\mathbf{W},\end{aligned}$$

whose entries are given by

$$\mathcal{L}^{\mathrm{rw}}_{ij} = \begin{cases} 1 & \text{if } i = j \\ -\frac{w_{ij}}{d_i} & \text{otherwise.} \end{cases}$$

This matrix is named after the *random walk matrix* $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ which describes the transitions of a Markov chain defined on the vertices of the graph. Its elements are the transition probabilities of going from a vertex i to another j proportional to the weight of the edge in-between: $P_{ij} = \frac{w_{ij}}{d_i}$. Note that even if the weight matrix $\mathbf{W}$ is symmetric, the same is not necessarily true of $\mathbf{P}$ and thus $\mathcal{L}^{\mathrm{rw}}$. It is only the case for *regular graphs* whose vertices all have equal degrees, that is, $d_i = d_j$ for all i, j. As this type of graph is rarely encountered in graph signal processing applications, the random walk Laplacian matrix is usually asymmetric and its eigenvectors not orthogonal for the usual inner product. It has however proved to be successful in some domains such as spectral clustering [129] and image processing [85] to give a few examples.

Despite its lack of symmetry, the random walk Laplacian matrix is nonetheless diagonalizable as it is similar to the normalized Laplacian: $\mathcal{L}^{\mathrm{norm}} = \mathbf{D}^{\frac{1}{2}}\mathcal{L}^{\mathrm{rw}}\mathbf{D}^{-\frac{1}{2}}$. As a result, they share the same eigenvalues and have related eigenvectors. This is easily observed

from the eigenpairs of $\mathcal{L}^{\text{norm}}$:

$$\begin{aligned}\mathcal{L}^{\text{norm}}\boldsymbol{u}_\ell &= \mu_\ell \boldsymbol{u}_\ell \\ \mathbf{D}^{\frac{1}{2}}\mathcal{L}^{\text{rw}}\mathbf{D}^{-\frac{1}{2}}\boldsymbol{u}_\ell &= \mu_\ell \boldsymbol{u}_\ell \\ \mathcal{L}^{\text{rw}}(\mathbf{D}^{-\frac{1}{2}}\boldsymbol{u}_\ell) &= \mu_\ell(\mathbf{D}^{-\frac{1}{2}}\boldsymbol{u}_\ell),\end{aligned}$$

for all $\ell = 1, \ldots, n$. We see that $\mathcal{L}^{\text{rw}}$ has exactly the same spectrum as $\mathcal{L}^{\text{norm}}$ and its eigenvectors are given by $\mathbf{D}^{-\frac{1}{2}}\boldsymbol{u}_1, \ldots, \mathbf{D}^{-\frac{1}{2}}\boldsymbol{u}_n$. With the appropriate inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathbf{D}} = \boldsymbol{x}^\top \mathbf{D} \boldsymbol{y}$, the latter form an orthonormal basis for $\mathbb{R}^n$.

Graph signals

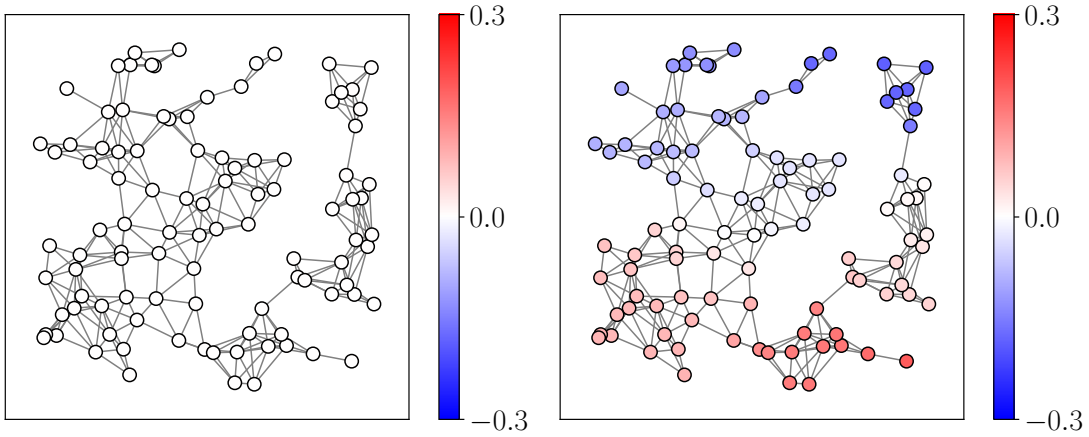
While the notions presented above from spectral graph theory focus on the structure of the graph, they form the basis of *graph signal processing* [115, 102]. This field aims to study signals whose values reside on the graph vertices. Formally, a *graph signal* is a function $f : \mathcal{V} \to \mathbb{R}$ that assigns to each vertex i a real value $f(i)$. Assuming the ordering of the vertices from 1 to n, the graph signal can be represented by a vector $\boldsymbol{f} \in \mathbb{R}^n$ where the i^{th} element is equal to the function value at the i^{th} vertex. Figure 1.2 illustrates examples of signals defined on a same graph composed of 100 vertices and 295 edges.

Notice how these graph signals vary differently from one vertex to another: the first signal (1.2a) is completely constant across all vertices while the last one (1.2d) displays many variations over the whole graph. As for the other two graph signals (1.2b and 1.2c), a relative regularity can be observed where vertices close to each other on the graph have similar associated signal values. Here, the notion of closeness corresponds to vertices being strongly connected by edges and more generally by paths of varying sizes. Thus, considering the underlying structure of a graph is decisive in the proper analysis of any signals defined on its vertices.

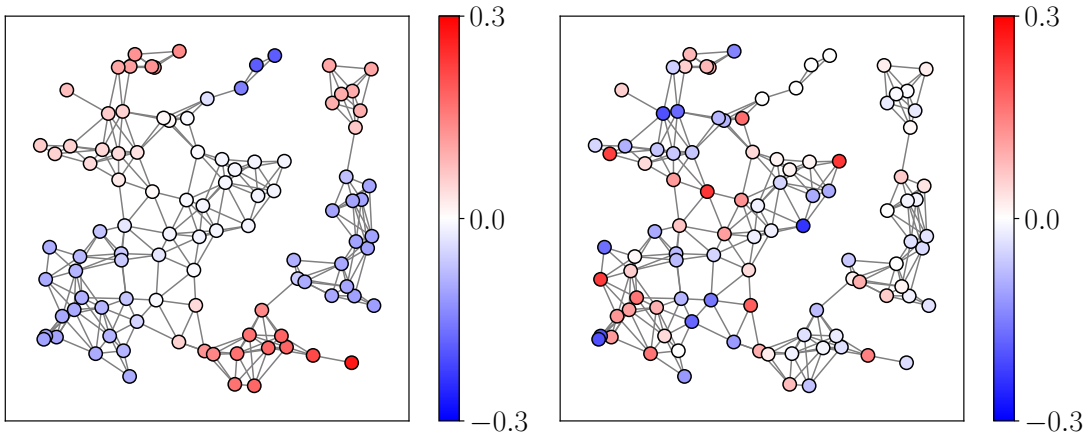
A general measure for the global smoothness of a graph signal $\boldsymbol{f}$ is given by the *discrete p-Dirichlet form* [115]:

$$S_p(\boldsymbol{f}) = \frac{1}{p} \sum_{i \in \mathcal{V}} \left(\sum_{j \in \mathcal{N}_i} w_{ij} (f(i) - f(j))^2 \right)^{\frac{p}{2}},$$

where $\mathcal{N}_i = \{ j \in \mathcal{V} \mid (i, j) \in \mathcal{E} \}$ is the neighborhood of vertex i, that is, the set of vertices connected to it by an edge. This form is commonly used with two particular values of p



(a) $R_{\mathcal{L}}(\boldsymbol{f}_1) = 0$ (b) $R_{\mathcal{L}}(\boldsymbol{f}_2) = 0.1$



(c) $R_{\mathcal{L}}(\boldsymbol{f}_3) = 0.21$ (d) $R_{\mathcal{L}}(\boldsymbol{f}_4) = 4.73$

Figure 1.2 – Examples of graph signals with their respective Rayleigh quotient.

that yield well-known quantities: $S_1(\boldsymbol{f})$ is the *total variation* of the signal $\boldsymbol{f}$ on the graph whereas $S_2(\boldsymbol{f})$ equates to the *Laplacian quadratic form* as seen above:

$$\begin{aligned} S_2(\boldsymbol{f}) &= \frac{1}{2} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} w_{ij} (f(i) - f(j))^2 \\ &= \sum_{\{i,j\} \in \mathcal{E}} w_{ij} (f(i) - f(j))^2 = \boldsymbol{f}^\top \mathcal{L} \boldsymbol{f}. \end{aligned}$$

This specific measure of the signal smoothness can be understood as the weighted cumulative energy of the signal changes across the graph vertices. It is no coincidence that the

Laplacian matrix $\mathcal{L}$ appears here because, by definition, it is a differential operator:

$$(\mathcal{L}\boldsymbol{f})(i) = \sum_{j \in \mathcal{N}_i} w_{ij}(f(i) - f(j)).$$

Normalizing the Laplacian quadratic form $S_2(\boldsymbol{f})$ with the signal energy $\boldsymbol{f}^\top \boldsymbol{f} = \|\boldsymbol{f}\|_2^2$ gives the *Rayleigh quotient* $R_{\mathcal{L}}(\boldsymbol{f}) = \frac{\boldsymbol{f}^\top \mathcal{L} \boldsymbol{f}}{\boldsymbol{f}^\top \boldsymbol{f}}$. This measure is particularly useful to compare the smoothness of signals with distinct norms defined on a same graph. Like $S_p(\boldsymbol{f})$, it is equal to zero for constant signals across all vertices and increases with the number of oscillations in the signal as illustrated by Figure 1.2. More precisely, it is small when the signal vary slowly between vertices connected by an edge with a large weight.

The Rayleigh quotient is related to the spectrum of the Laplacian matrix in several ways. First, it takes values between the smallest and largest eigenvalues of $\mathcal{L}$:

$$\lambda_1 = \min_{\boldsymbol{f} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}} R_{\mathcal{L}}(\boldsymbol{f}), \qquad \lambda_n = \max_{\boldsymbol{f} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}} R_{\mathcal{L}}(\boldsymbol{f}),$$

where the minimum and maximum are reached for the eigenvectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_n$, respectively. Moreover, the Rayleigh quotient can be used to iteratively identify all eigenvalues and associated eigenvectors through the Courant-Fischer theorem [70, Theorem 4.2.6]:

$$\lambda_\ell = \min_{\substack{\boldsymbol{f} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\} \\ \boldsymbol{f} \perp \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{\ell-1}}} R_{\mathcal{L}}(\boldsymbol{f}), \quad \ell = 2, \ldots, n,$$

for which the eigenvector $\boldsymbol{v}_\ell$ is the solution of the ℓ^{th} problem. In words, each consecutive eigenvector is the smoothest nonzero vector orthogonal to its predecessors. In addition, from $\mathcal{L}\boldsymbol{v}_\ell = \lambda_\ell \boldsymbol{v}_\ell$ and $\boldsymbol{v}_\ell^\top \boldsymbol{v}_\ell = 1$, we have

$$R_{\mathcal{L}}(\boldsymbol{v}_\ell) = \frac{\boldsymbol{v}_\ell^\top \mathcal{L} \boldsymbol{v}_\ell}{\boldsymbol{v}_\ell^\top \boldsymbol{v}_\ell} = \lambda_\ell \boldsymbol{v}_\ell^\top \boldsymbol{v}_\ell = \lambda_\ell.$$

Thus, each eigenvalue precisely measures the smoothness of its corresponding eigenvector and therefore carries a notion of frequency: small (resp. large) eigenvalues λ_ℓ are associated with slowly (resp. quickly) oscillating eigenvectors $\boldsymbol{v}_\ell$ over the graph vertices. This relationship proves useful to define graph analogues of the Fourier and wavelet transforms using the Laplacian eigenvectors.

Graph Laplacian convergence

Several convergence results of the graph Laplacian towards the manifold Laplace-Beltrami operator have been established in the literature [10, 116, 126]. Let $x_1, \dots, x_n$ be uniformly sampled points from a manifold $\mathcal{M} \subset \mathbb{R}^n$ and let $g \in \mathcal{C}^\infty(\mathcal{M})$ be a function defined on this manifold. We construct an associated graph whose vertices are the n sampled points and edge weights are computed with a Gaussian kernel of the Euclidean distance as presented above: $w_{ij} = \exp\left(-\frac{\mathrm{d}(x_i,x_j)^2}{2\alpha^2}\right)$, with x_i the coordinates of vertex i and $\alpha > 0$. The *point cloud Laplace operator* is defined by

$$\mathbf{L}_n g(x) = g(x)\frac{1}{n}\sum_j \exp\left(-\frac{\mathrm{d}(x,x_j)^2}{2\alpha^2}\right) - \frac{1}{n}\sum_j g(x_j)\exp\left(-\frac{\mathrm{d}(x,x_j)^2}{2\alpha^2}\right),$$

where x is an arbitrary point on the manifold. This Laplacian associated to the point cloud is an extension of the graph Laplacian applied to the graph signal $\boldsymbol{g} = (g(x_1), \dots, g(x_n))^\top$:

$$\begin{aligned}\mathbf{L}_n g(x_i) &= g(x_i)\frac{1}{n}\sum_j w_{ij} - \frac{1}{n}\sum_j g(x_j)w_{ij} \\ &= \frac{1}{n}\sum_j w_{ij}(g(x_i) - g(x_j)) = \frac{1}{n}(\mathcal{L}\boldsymbol{g})(i).\end{aligned}$$

Under certain assumptions, the point cloud Laplace operator $\mathbf{L}_n g$ converges in probability to the Laplace-Beltrami operator on the manifold $\Delta_{\mathcal{M}} g$ as the sampling density increases, that is, as the sample size $n \to \infty$ and the kernel bandwidth $\alpha \to 0$.

1.1.2 Graph Fourier transform

Laplacian based graph Fourier transform

In classical signal processing, the Fourier transform decomposes time continuous signals with oscillating components called the *Fourier modes*. These are the eigenfunctions g of the Laplace operator Δ which, together with the associated eigenvalues λ satisfy the Helmholtz equation: $-\Delta g = \lambda g$. In the one-dimensional case, we have

$$-\Delta g(t) = -\nabla^2 g(t) = \lambda g(t) \Leftrightarrow \frac{\partial^2}{\partial t^2} g(t) = -\lambda g(t) \Leftrightarrow g(t) = e^{i\sqrt{\lambda}t},$$

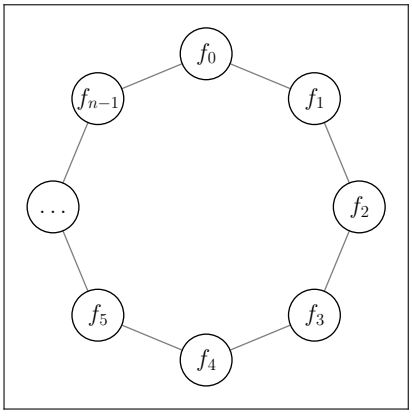
where ∇ is the gradient operator and $t \in \mathbb{R}$. The eigenvalues and eigenfunctions of the Laplace operator are respectively $\{(2\pi\nu)^2\}_\nu$ and $\{e_\nu(t) = e^{i2\pi\nu t}\}_\nu$, where $\nu \in \mathbb{R}$ is the

frequency. Hence, rapidly oscillating continuous Fourier modes go with large eigenvalues as the frequency increases in absolute value. Inversely, low frequencies close to zero produce smooth Fourier modes and small associated eigenvalues. The classical Fourier transform corresponds to the projection of an absolutely integrable function f on the *Fourier basis* $\{e_\nu(t)\}_{\nu\in\mathbb{R}}$ using the Hermitian inner product $\langle f, g\rangle = \int_{\mathbb{R}} f(t)\overline{g(t)}dt$.

Definition 1.1.1 (Fourier transform)**.** The Fourier transform $\hat{f} \in L^1(\mathbb{R})$ at frequency ν of a function $f \in L^1(\mathbb{R})$ and its inverse are given by

$$\hat{f}(\nu) = \langle f, e_\nu\rangle = \int_{\mathbb{R}} f(t)e^{-i2\pi\nu t}dt$$
$$f(t) = \langle \hat{f}, \bar{e}_\nu\rangle = \int_{\mathbb{R}} \hat{f}(\nu)e^{i2\pi\nu t}d\nu.$$

A common approach to define the Fourier transform of a graph signal is to consider the eigenvectors of the Laplacian matrix as a Fourier basis. Their analogous behavior in relation to the eigenvalues intuitively makes them suitable candidates for Fourier modes in the graph domain. Moreover, the discrete Fourier transform of a time series is directly linked to the Laplacian matrix of the cycle graph. This particular type of graph composed of n vertices connected by n edges in a closed chain can be used to represent a periodic time series f_t of period n as shown on Figure 1.3.



$$\mathcal{L} = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & -1 \\ -1 & 2 & -1 & \ddots & & 0 \\ 0 & -1 & 2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & \ddots & \ddots & -1 \\ -1 & 0 & \cdots & 0 & -1 & 2 \end{pmatrix}$$

Figure 1.3 – Cycle graph representation of an n-periodic time series

It is well known that the Laplacian eigenbasis of the cycle graph is identical to the Fourier modes of the discrete Fourier transform [61] given by $\hat{f}(\nu_\ell) = \frac{1}{\sqrt{n}}\sum_{t=0}^{n-1} f_t e^{-i2\pi\nu_\ell t}$, with $\nu_\ell = \frac{\ell-1}{n}$ and $\ell = 1, \ldots, n$. Indeed, the Laplacian eigenvalues and eigenvectors are respectively $\lambda_\ell = 2 - 2\cos(2\pi\nu_\ell)$ and $\boldsymbol{v}_\ell = (1, \zeta_\ell^1, \ldots, \zeta_\ell^{n-1})^\top$, where $\zeta_\ell = e^{i2\pi\nu_\ell}$ [17, Section

1.4.3]. This result further supports the definition of the *graph Fourier transform* as the projection of a signal on the eigenvectors of the Laplacian matrix.

Definition 1.1.2 (Graph Fourier transform)**.** The graph Fourier transform $\hat{f}(\lambda_\ell)$ at frequency λ_ℓ of a signal $\boldsymbol{f} \in \mathbb{R}^n$ defined on a graph $\mathcal{G}$ and its inverse are given by

$$\hat{f}(\lambda_\ell) = \langle \boldsymbol{f}, \boldsymbol{v}_\ell \rangle = \sum_{i=1}^{n} f(i) v_\ell(i)$$
$$f(i) = \sum_{l=1}^{n} \hat{f}(\lambda_\ell) v_\ell(i),$$

where λ_ℓ and $\boldsymbol{v}_\ell$ are the eigenvalues and eigenvectors of the graph Laplacian matrix.

Through this definition, we see that the graph Fourier transform is a linear operator which can be represented by a matrix: $\hat{\boldsymbol{f}} = \mathbf{V}^\top \boldsymbol{f}$, with $\mathbf{V} = (\boldsymbol{v}_1 | \dots | \boldsymbol{v}_n)$. Because $\mathbf{V}$ is an orthogonal matrix, the graph Fourier transform verifies the Plancherel theorem like its classical analogue:

$$\langle \hat{\boldsymbol{f}}, \hat{\boldsymbol{g}} \rangle = \left(\mathbf{V}^\top \boldsymbol{f}\right)^\top \mathbf{V}^\top \boldsymbol{g} = \boldsymbol{f}^\top \mathbf{V}\mathbf{V}^\top \boldsymbol{g} = \langle \boldsymbol{f}, \boldsymbol{g} \rangle,$$

where $\boldsymbol{f}, \boldsymbol{g} \in \mathbb{R}^n$. As a result, the transform also preserves the graph signal energy as stated by Parseval's identity:

$$\|\hat{\boldsymbol{f}}\|_2^2 = \|\boldsymbol{f}\|_2^2.$$

The graph Fourier transform provides a way to view a signal as a function of the Laplacian eigenvalues. A given signal can thus be represented in the graph spectral domain in addition to the usual vertex domain. Figure 1.4 shows an example of a graph signal $\boldsymbol{f}$ represented in both domains: its values on the vertices of a graph are represented on the left and its graph Fourier coefficients $\hat{f}(\lambda_\ell)$ on the right.

Similarly to the classical Fourier transform, the graph Fourier coefficients of a smooth signal decay when the frequency increases. As illustrated, the signal visible smoothness on the vertices translates to most of its energy being concentrated in the low frequencies. This means its projection on the Fourier basis is mainly supported by the slowly oscillating eigenvectors. Consequently, a smooth graph signal is *compressible* in the sense that just a few Fourier coefficients are needed to approximate it as it is the case with classical signal processing [140].

The interaction between the smoothness of a signal and its graph Fourier transform

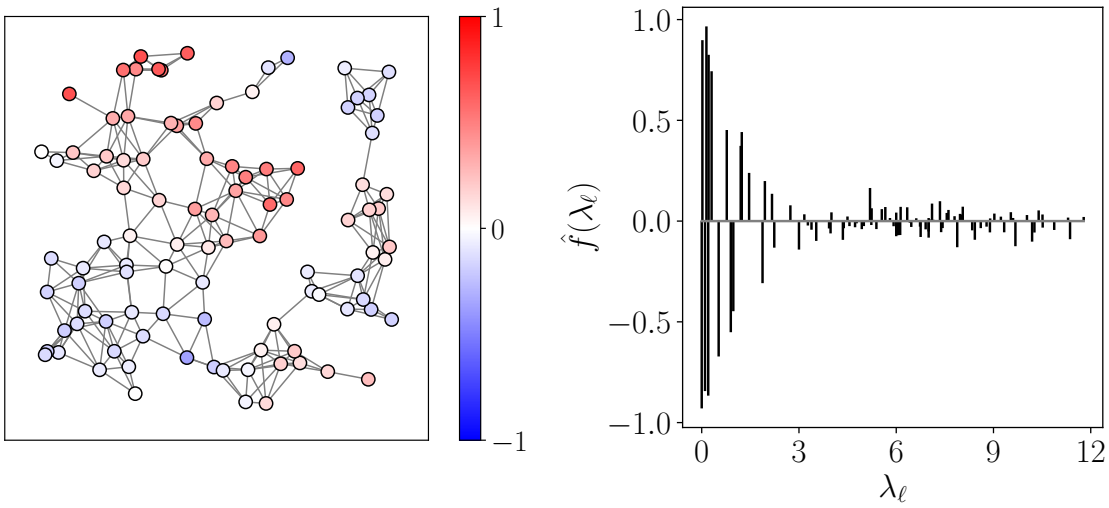


Figure 1.4 – Equivalent representations of a graph signal in the vertex (left) and graph spectral (right) domains.

is also observed in the Rayleigh quotient. Indeed, with Parseval's identity and the eigendecomposition $\mathcal{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, we have the alternative formula

$$R_\mathcal{L}(\boldsymbol{f}) = \frac{\boldsymbol{f}^\top \mathcal{L} \boldsymbol{f}}{\boldsymbol{f}^\top \boldsymbol{f}} = \frac{\hat{\boldsymbol{f}}^\top \mathbf{\Lambda} \hat{\boldsymbol{f}}}{\hat{\boldsymbol{f}}^\top \hat{\boldsymbol{f}}} = \frac{\sum_{l=1}^n \lambda_\ell \hat{f}(\lambda_\ell)^2}{\sum_{l=1}^n \hat{f}(\lambda_\ell)^2},$$

where the Rayleigh quotient appears as a weighted average of the Laplacian eigenvalues by the Fourier coefficients. Here again, we see that for smooth signals the average is weighted towards the lower eigenvalues and thus the resulting Rayleigh quotient is small.

Other graph Fourier transforms

Other definitions of the graph Fourier transform have been proposed. We briefly present two of them based on the eigenvectors of other graph matrix representations.

First, the Laplacian matrix can be replaced with its normalized variant $\mathcal{L}^{\text{norm}}$ whose eigenvalues μ_ℓ and eigenvectors $\boldsymbol{u}_\ell$ provide a comparable notion of frequency [57]. As we have $R_{\mathcal{L}^{\text{norm}}}(\boldsymbol{u}_\ell) = \mu_\ell$, the same relationship where large eigenvalues are associated with quickly oscillating eigenvectors is observed. The *normalized Laplacian matrix based graph Fourier transform* is given by

$$\hat{\boldsymbol{f}} = \mathbf{U}^\top \boldsymbol{f},$$

where $\mathbf{U} = (\boldsymbol{u}_1 | \dots | \boldsymbol{u}_n)$. As with the standard Laplacian matrix $\mathcal{L}$, the normalized Lapla-

cian matrix has been proved to converge to the manifold Laplace-Beltrami operator under certain conditions [10]. Additionally, it is likewise limited to graphs composed of symmetric edges with non-negative weights to benefit from the useful theoretical results on its spectrum presented in Section 1.1.1.

Another graph Fourier transform definition based on algebraic signal processing theory [106] is proposed by the authors of [111]. Their approach is built on the observation that the weight matrix $\mathbf{W}$ of the directed cycle graph, whose elements are $w_{ij} = \mathbf{1}_{\{i-j\equiv 1 \mod n\}}$, matches the time shift operator of periodic time series: $(\mathbf{W}\boldsymbol{f})(i) = \boldsymbol{f}(i-1)$. For any directed, weighted graph, this operation is called the *graph shift* and replaces a graph signal value on a given vertex with the weighted linear combination of the signal values on the adjacent vertices: $\tilde{f}(i) = (\mathbf{W}\boldsymbol{f})(i) = \sum_{j\in\mathcal{N}_i} w_{ij} f(j)$. The eigenbasis of the weight matrix is used as a Fourier basis to define the *weight matrix based graph Fourier Transform*:

$$\hat{\boldsymbol{f}} = \mathbf{T}^{-1}\boldsymbol{f},$$

with $\mathbf{T}$ the matrix of eigenvectors of $\mathbf{W}$. Because this definition applies to directed graphs, the lack of symmetry of $\mathbf{W}$ implies $\mathbf{T}$ is not orthogonal and thus no explicit formula can be given as with the previous graph Fourier transform definitions. Moreover, the Plancherel theorem and Parseval's identity are not verified for the same reasons. Another disadvantage of this approach comes from the difficult interpretation of the weight matrix complex-valued eigenvalues as frequencies. In the particular case of a real spectrum, the authors provide an analysis [112] through a definition of the graph signal total variation based on the graph shift operator: $\mathrm{TV}(\boldsymbol{f}) = \|\boldsymbol{f} - \mathbf{W}\boldsymbol{f}\|_1$. They show that eigenvectors associated with large eigenvalues vary slowly between the graph vertices and vice versa. With this limited result, the problem of interpreting complex eigenvalues however remains to be addressed. On the other hand, this graph Fourier transform benefits from a wide range of possible applications because its definition is not restricted to graphs with non-negative real edge weights like the Laplacian matrix based ones are. This approach is therefore an appropriate tool in the study of signals on negatively or even complex weighted graphs.

All three graph Fourier transforms presented above share a common problem: they involve the eigendecomposition of a matrix whose size directly depends on the size of the graph. For a graph composed of n vertices, the diagonalization of either Laplacian matrices or the weight matrix has a computational complexity of $\mathcal{O}(n^3)$ with most eigen-

value algorithms. Hence, these approaches to compute the graph Fourier transform are limited to graphs with less than a few thousand vertices. As some applications in signal and image processing can involve data with millions of dimension or more, approximate computational methods are required to extend this transform to large-scale graphs. We can cite the *approximate fast graph Fourier transform* [78] which is based on an approximate diagonalization of the graph Laplacian matrix performed with a modified Jacobi eigenvalue algorithm. Alternatively, other methods [88] allow for a fast and exact graph Fourier transform providing that the graph has suitable symmetry properties.

1.1.3 Spectral graph wavelet transform

Although the Laplacian based graph Fourier transform is a straightforward method for the study of graph signals, it lacks convenient properties to get a finer analysis such as localization. Indeed, the Fourier transform of a graph signal is global in the sense that most of its Fourier modes, that is, the Laplacian matrix eigenvectors are not localized in the vertex domain of the graph.

Localized, multiscale transforms for signals on graphs

Different localized transforms have been proposed to locally study a graph signal around a vertex through the values it takes on the neighbor vertices at different scales. These methods are usually adaptations of wavelet transforms from classical signal processing which can localize signal information in both the time and frequency domains. Correspondingly, graph wavelet transforms are designed so that graph signal information is localized in both the vertex and graph spectral domains.

A graph signal $\boldsymbol{f}$ is said to be localized in the vertex domain when most of its energy $\|\boldsymbol{f}\|_2^2$ is concentrated on the relatively close neighbors of a given vertex. The size of the neighborhood generally depends on a scale parameter which controls the degree of localization. In the same way, we can say that a graph signal is localized in the spectral domain if its projection $\hat{\boldsymbol{f}}$ on the graph Laplacian eigenvectors has most of its energy $\|\hat{\boldsymbol{f}}\|_2^2$ concentrated in a band of frequencies centered around a given frequency. Localization in the graph spectral domain is illustrated on Figure 1.4 where the energy is mostly supported by the low frequencies. On the contrary, the distribution of the graph signal energy is more scattered across the graph vertices and thus does not portray vertex domain localization. The graph wavelet transform designs are of two types: vertex domain designs and graph

spectral domain designs.

The vertex domain designs of graph wavelets are based on spatial features of the graph like how connected together or distant are its vertices. Some examples of such designs are *random multiscale representations* [131] and *graph wavelets* [29] for unweighted graphs, along with *lifting based wavelets* [97, 113] and *multiscale tree wavelets* [56] for weighted graphs.

Whereas the graph spectral properties of the vertex domain designs are not explicitly designed, the graph spectral domain designs of graph wavelets are based on spectral features of the graph. These are encoded in the eigenvalues and eigenvectors of one of the graph matrix representations like the graph Laplacian. Designs of this type aim to build localized bases in both the vertex and graph spectral domains. Examples of graph spectral domain designs include *graph wavelet filterbanks* [98, 96], *diffusion wavelets* [89, 26] and the *spectral graph wavelet transform* [65] which we describe in more detail below. The *graph wavelet filterbanks* extend the multiscale analysis based on filterbanks from classical signal processing to graph signal processing. They apply to bipartite graphs whose vertex set can be partitioned into two subsets such that all edges lie between those sets. Under certain conditions, the defined filterbanks are critically sampled, provide perfect reconstruction and are either orthogonal [98] or biorthogonal with localized basis functions on a compact support [96]. The *diffusion wavelets* depend on compressed representations of the powers of a diffusion operator like the graph Laplacian to build a multiscale transform. The localized basis functions at each scale are then downsampled and orthogonalized through a variation of the Gram-Schmidt process.

Spectral graph wavelet transform

The theory of wavelets was originally created for square integrable functions defined on the real line [92, 128]. We first recall the classical wavelet transform in the one-dimensional case before presenting the spectral graph wavelet transform introduced in [65].

The continuous wavelet transform is usually generated by a single *mother wavelet* function $\psi \in L^2(\mathbb{R})$. Wavelets at scale $s > 0$ and location $a \in \mathbb{R}$ are then defined by appropriately scaling and shifting the mother wavelet:

$$\psi_{s,a}(t) = \frac{1}{s}\psi\left(\frac{t-a}{s}\right).$$

It is possible to directly define these wavelets in the Fourier domain as

$$\psi_{s,a}(t) = \int_{\mathbb{R}} \hat{\psi}(s\nu) e^{-i\nu a} e^{i2\pi\nu t} d\nu,$$

where $\hat{\psi}$ is the Fourier transform of the mother wavelet and $\nu \in \mathbb{R}$ is the frequency. In this expression, the wavelet is represented on the eigenfunctions $e_\nu(t) = e^{i2\pi\nu t}$ of the Laplace operator Δ. We also observe that scaling ψ with $\frac{1}{s}$ is equivalent to scaling $\hat{\psi}$ by s in the Fourier domain. Finally, shifting the mother wavelet to location a is done by multiplying its Fourier transform by $e^{-i\nu a}$.

This expression of the continuous wavelet serves as a basis in [65] to define an analogue on graphs. Precisely, the *spectral graph wavelets* at scale s and vertex $a \in \mathcal{V}$ are designed in the graph spectral domain as

$$\psi_{s,a}(i) = \sum_{\ell=1}^{n} g(s\lambda_\ell) v_\ell(a) v_\ell(i),$$

where $i \in \mathcal{V}$. In comparison to the previous expression, the mother wavelet Fourier transform $\hat{\psi}$ is replaced by a wavelet generating function $g : \mathbb{R}_+ \to \mathbb{R}_+$ that behaves as a scaled bandpass filter such that $g(0) = 0$ and $\lim_{x\to\infty} g(x) = 0$. Similarly to the graph Fourier transform, the eigenvalue λ_ℓ now acts as the frequency and the wavelet is represented on the eigenvectors $v_\ell(i)$ of the graph Laplacian $\mathcal{L}$ instead of the eigenfunctions $e_\nu(t)$. Last, shifting the wavelet to vertex a is performed by substituting $e^{-i\nu a}$ with $v_\ell(a)$.

These spectral graph wavelets are accompanied by *spectral graph scaling functions* which help ensure stable representation of the low frequency of the signal. They are similarly constructed by a scaling generating function $h : \mathbb{R}_+ \to \mathbb{R}$ which acts as a low-pass filter such that $h(0) > 0$ and $\lim_{x\to\infty} h(x) = 0$. The spectral graph scaling functions at vertex $a \in \mathcal{V}$ are given by

$$\psi_{0,a}(i) = \sum_{\ell=1}^{n} h(\lambda_\ell) v_\ell(a) v_\ell(i).$$

It should be noted that for all vertices $a \in \mathcal{V}$, the spectral graph wavelets and scaling functions correspond to functional calculus on $\mathcal{L}$. Indeed, for any function $\rho : \mathrm{sp}(\mathcal{L}) \to \mathbb{R}$ defined on the spectrum of $\mathcal{L}$, we have the general formula $\rho(\mathcal{L}) = \sum_{l=1}^{n} \rho(\lambda_\ell) \langle \boldsymbol{v}_\ell, \cdot \rangle \boldsymbol{v}_\ell$. As a result, with the matrix representation $\mathbf{G}_s = \mathrm{diag}(g(s\lambda_1), \ldots, g(s\lambda_n))$, the *wavelet frame*

at scale s reads as

$$\boldsymbol{\Psi}_s = (\boldsymbol{\psi}_{s,1}|\dots|\boldsymbol{\psi}_{s,n}) = \mathbf{V}\mathbf{G}_s\mathbf{V}^\top = g(s\mathcal{L}),$$

where $\boldsymbol{\psi}_{s,a} = (\psi_{s,a}(1),\dots,\psi_{s,a}(n))^\top \in \mathbb{R}^n$ is one spectral graph wavelet. Likewise, the *scaling function frame* is given by

$$\boldsymbol{\Psi}_0 = (\boldsymbol{\psi}_{0,1}|\dots|\boldsymbol{\psi}_{0,n}) = \mathbf{V}\mathbf{H}\mathbf{V}^\top = h(\mathcal{L}),$$

where $\mathbf{H} = \mathrm{diag}(h(\lambda_1),\dots,h(\lambda_n))$.

In the continuous setting, the wavelet coefficients of a function $f \in L^2(\mathbb{R})$ are given by the Hermitian inner product with the wavelets:

$$(\mathcal{W}f)(s,a) = \langle f, \psi_{s,a}\rangle = \int_{\mathbb{R}} f(t)\overline{\psi_{s,a}(t)}dt.$$

By analogy, the wavelet and scaling function coefficients of a graph signal $\boldsymbol{f} \in \mathbb{R}^n$ are given by its projection on the spectral graph wavelets and scaling functions, respectively:

$$(\mathcal{W}\boldsymbol{f})(s,a) = \langle \boldsymbol{f}, \boldsymbol{\psi}_{s,a}\rangle = \boldsymbol{\psi}_{s,a}^\top \boldsymbol{f},$$
$$(\mathcal{W}\boldsymbol{f})(0,a) = \langle \boldsymbol{f}, \boldsymbol{\psi}_{0,a}\rangle = \boldsymbol{\psi}_{0,a}^\top \boldsymbol{f}.$$

In practice, the continuous scaling parameter s is sampled to a finite number of J scales $\mathcal{S} = \{s_j\}_{j=1}^J$. This yields a collection of nJ wavelets $\boldsymbol{\psi}_{s_j,a}$ and n scaling functions $\boldsymbol{\psi}_{0,a}$ that can be gathered in a matrix $\boldsymbol{\Psi} = (\boldsymbol{\Psi}_0|\boldsymbol{\Psi}_{s_1}|\dots|\boldsymbol{\Psi}_{s_J}) \in \mathbb{R}^{n\times n(J+1)}$ to represent the entire wavelet frame. In this case, the *spectral graph wavelet transform* (SGWT) is a linear operator:

$$\mathcal{W}\boldsymbol{f} = \boldsymbol{\Psi}^\top \boldsymbol{f} \in \mathbb{R}^{n(J+1)}.$$

Construction of Parseval frames

At discretized scales, the spectral graph wavelets as first introduced by [65] can be considered as a frame of which we review the definition [11]. A family of wavelets $\{\boldsymbol{\psi}_{s,a}\}_{s\in\mathcal{S},\, a\in\mathcal{V}}$ forms a *frame* of $\mathbb{R}^n$, if there exist frame bounds $A, B \in (0,\infty)$ satisfying for all $\boldsymbol{f} \in \mathbb{R}^n$

$$A\|\boldsymbol{f}\|_2^2 \le \sum_{s,a} |\langle \boldsymbol{f}, \boldsymbol{\psi}_{s,a}\rangle|^2 \le B\|\boldsymbol{f}\|_2^2.$$

The frame bounds A and B give information about the numerical stability of recov-

ering the graph signal $\boldsymbol{f}$ from the wavelet coefficients $\mathcal{W}\boldsymbol{f}$. They can be optimized with an appropriate choice of the wavelet generating function g. The authors of the SGWT originally proposed the following filter:

$$g(x;\alpha,\beta,x_1,x_2) = \begin{cases} x_1^{-\alpha}x^{\alpha} & \text{for } x < x_1 \\ p(x) & \text{for } x_1 \leq x \leq x_2 \\ x_2^{\beta}x^{-\beta} & \text{for } x > x_2, \end{cases}$$

where the integers α and β, and the transition points x_1 and x_2 are parameters of the filter. It behaves as a monic power near the origin and decays as a power law for large x. The transition is done with the unique cubic polynomial $p(x)$ that respects the continuity of g and its derivative g'. Figure 1.5 shows wavelet generating functions $g(s\cdot)$ for $J = 4$ different scales, $p(x) = -5 + 11x - 6x^2 + x^3$ and $\alpha = \beta = 2$, $x_1 = 1$, $x_2 = 2$ parameter value examples as in [65] on the interval $[0, 4]$. The scaling generating function is given by $h(x) = \gamma \exp\left(-(\frac{x}{0.12})^4\right)$, where γ is chosen such that $h(0) = \max g(x)$.

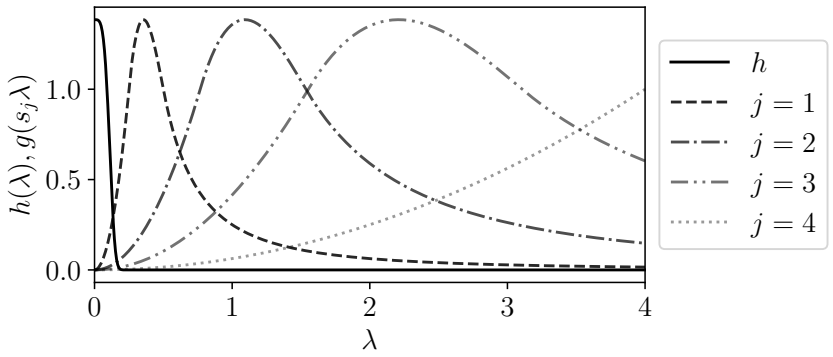


Figure 1.5 – Scaling and wavelet generating functions from [65] for 4 scales.

Other generating functions have since been introduced in the literature to achieve better frame bounds A and B than the ones associated with the SGWT from [65]. One type of frame particularly sought after is the *tight frame* which corresponds to equal bounds. More specifically, when $A = B = 1$, the family of wavelets $\{\boldsymbol{\psi}_{s,a}\}_{s\in\mathcal{S},\, a\in\mathcal{V}}$ forms a *Parseval frame* and preserves the energy of all graph signal $\boldsymbol{f} \in \mathbb{R}^n$ in the wavelet domain:

$$\|\boldsymbol{f}\|_2^2 = \sum_{s,a} |\langle \boldsymbol{f}, \boldsymbol{\psi}_{s,a}\rangle|_2^2 = \|\mathcal{W}\boldsymbol{f}\|_2^2.$$

In addition, Parseval frames benefit from a simple reconstruction formula similar to

that of an orthonormal basis. A natural choice for the inverse transform in the general case [65] is the pseudoinverse

$$\mathcal{W}^{-1} = (\mathcal{W}^*\mathcal{W})^{-1}\mathcal{W}^* = (\mathbf{\Psi}\mathbf{\Psi}^\top)^{-1}\mathbf{\Psi}.$$

In the specific case of Parseval frames, the wavelet frame $\mathbf{\Psi}$ is a semi-orthogonal matrix which simplifies the inverse transform to

$$\mathcal{W}^{-1} = \mathcal{W}^* = \mathbf{\Psi}.$$

Any graph signal thus can easily be recovered from its frame decomposition:

$$\boldsymbol{f} = \mathcal{W}^*\mathcal{W}\boldsymbol{f} = \mathbf{\Psi}\mathbf{\Psi}^\top\boldsymbol{f} = \sum_{s,a}\langle \boldsymbol{f}, \boldsymbol{\psi}_{s,a}\rangle \boldsymbol{\psi}_{s,a}.$$

Extensions of the original SGWT to form a Parseval frame have been proposed by [81] and more recently [58] which we present here. Both approaches use a partition of unity to define the wavelet generating functions of the spectral graph wavelets. The set $\{\phi_j\}_{j=0}^J$ of functions $\phi_j : [0, \lambda_n] \to [0, 1]$ is a finite partition of unity if it satisfies $\sum_{j=0}^J \phi_j(\lambda) = 1$ for all $\lambda \in [0, \lambda_n]$. At scale index $j = 0, \ldots, J$ and vertex $a \in \mathcal{V}$, the spectral graph wavelets from [58] are given by

$$\boldsymbol{\psi}_{j,a} = \sum_{\ell=1}^{n} \sqrt{\phi_j(\lambda_\ell)} v_\ell(a) \boldsymbol{v}_\ell.$$

With the matrix representation $\mathbf{\Phi}_j = \mathrm{diag}(\sqrt{\phi_j(\lambda_1)}, \ldots, \sqrt{\phi_j(\lambda_n)})$ of these new wavelet generating functions, the wavelet frame at scale j becomes

$$\mathbf{\Psi}_j = (\boldsymbol{\psi}_{j,1}| \ldots |\boldsymbol{\psi}_{j,n}) = \mathbf{V}\mathbf{\Phi}_j\mathbf{V}^\top = \sqrt{\phi_j}(\mathcal{L}).$$

Proving that the family of wavelets $\{\boldsymbol{\psi}_{j,a}\}_{j,a}$ forms a Parseval frame consists in showing

that the wavelet frame matrix $\mathbf{\Psi} = (\mathbf{\Psi}_0 | \dots | \mathbf{\Psi}_J)$ is semi-orthogonal:

$$\begin{aligned}\mathbf{\Psi}\mathbf{\Psi}^\top &= \sum_{j=0}^{J} \mathbf{\Psi}_j \mathbf{\Psi}_j^\top \\ &= \sum_{j=0}^{J} \mathbf{V}\mathbf{\Phi}_j \mathbf{V}^\top \mathbf{V}\mathbf{\Phi}_j \mathbf{V}^\top \\ &= \sum_{j=0}^{J} \mathbf{V}\mathbf{\Phi}_j \mathbf{\Phi}_j \mathbf{V}^\top \\ &= \mathbf{V}\mathbf{V}^\top = \mathbf{I}_n.\end{aligned}$$

The second equality is given by the symmetry of the wavelet frame $\mathbf{\Psi}_j$, the third and last by the orthogonality of the eigenvector matrix $\mathbf{V}$ while the fourth is obtained from the partition of unity property: $\sum_{j=0}^{J} \mathbf{\Phi}_j \mathbf{\Phi}_j = \mathbf{I}_n$.

To build a semi-orthogonal SGWT, the authors of [58] choose a sequence of functions $\{\phi_j\}_{j=0}^{J}$ called a *multiscale bandpass filter*. It is inspired by [28] which develops a theory of multiscale frame analysis on very general metric spaces. This construction is known as a smooth Littlewood-Paley decomposition in the context of functional analysis. Let $\omega : \mathbb{R}_+ \to [0, 1]$ be some continuous function with support in $[0, 1]$, satisfying $\omega(x) = 1$ for $x \in [0, b^{-1}]$, for some constant $b > 1$. For $j = 0, \dots, J$, the functions $\phi_j : \mathbb{R}_+ \to [0, 1]$ are defined by

$$\phi_j(x) = \begin{cases} \omega(x) & \text{if } j = 0 \\ \omega(b^{-j}x) - \omega(b^{-j+1}x) & \text{if } j > 0. \end{cases}$$

This multiscale bandpass filter forms a partition of unity as the condition $\sum_{j=0}^{J} \phi_j(\lambda) = 1$ is satisfied. For $j > 0$, each of these functions are supported on respective intervals $[b^{j-2}, b^j]$, which results in the graph wavelets from [58] to be localized in the spectral domain. Indeed, for a fixed scale index j and all vertices $a \in \mathcal{V}$, the spectral graph wavelets $\boldsymbol{\psi}_{j,a}$ are linear combinations of the eigenvectors $\boldsymbol{v}_\ell$ associated with the eigenvalues $\lambda_\ell \in [b^{j-2}, b^j]$, exclusively. Figure 1.6 illustrates an example of a multiscale bandpass filter on the interval $[0, 4]$ for $J + 1 = 4$ scales, $b = 2$ and a particular ω function. The latter is given by $\omega(x) = z(x + 1)z(1 - x)$, where $z(x) = y(x)/(y(x) + y(\frac{1}{2} - x))$ and $y(x) = e^{-1/x}\mathbf{1}_{\{x>0\}}$.

Along with spectral localization, the other property expected from graph wavelets is to be localized in the vertex domain. Spectral graph wavelets fit this criterion as most of their energy is centered around the vertex they are defined on. This is observed in practice on Figure 1.7 where such a wavelet $\boldsymbol{\psi}_{j,a}$ at scale indices $j = 0, \dots, 3$ and vertex

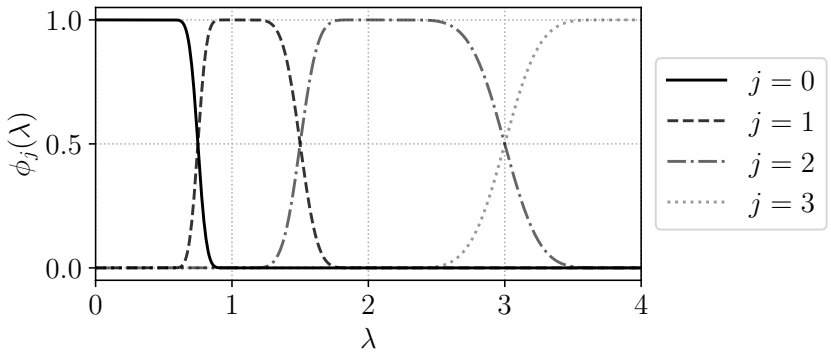


Figure 1.6 – Multiscale bandpass filter from [58] for 4 scales.

$a = 20$ of a cycle graph of size 40 is represented. The wavelet generating function used is the multiscale bandpass filter example given above.

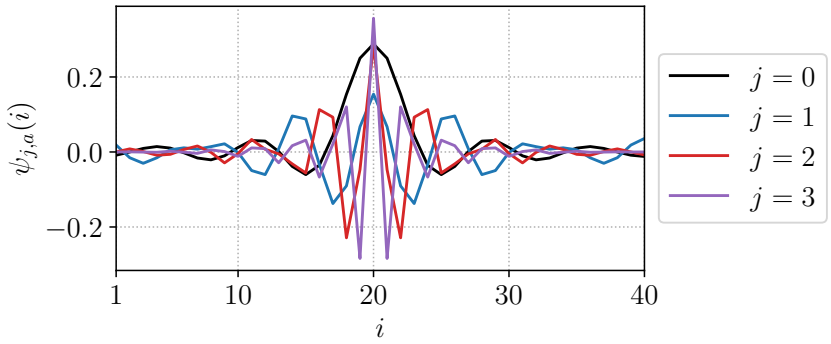


Figure 1.7 – Spectral graph wavelet from [58] at vertex $a = 20$ for 4 scales.

Notice how the energy of the spectral graph wavelet $\boldsymbol{\psi}_{j,a}$ is focused on the neighborhood of vertex a at each scale index j. In addition, we see the latter increases together with the number of oscillations in each wavelet as a result of their spectral localization. This clearly illustrates that the parameters j and a in $\boldsymbol{\psi}_{j,a}$ are respectively a spectral scale parameter and a spatial localization parameter.

Figure 1.8 shows the spectral graph wavelet coefficients at first scale index $(\mathcal{W}\boldsymbol{f})(0) = \boldsymbol{\Psi}_0^\top \boldsymbol{f} \in \mathbb{R}^n$ of a signal $\boldsymbol{f}$ defined on a cycle graph of size $n = 40$. The wavelet frame $\boldsymbol{\Psi}_0$ is constructed with the wavelet generating function ϕ_0 pictured in Figure 1.6.

We can observe that at scale index $j = 0$, the wavelet transform of a signal acts as a smoother that reduces its irregularities. This is due to function ϕ_0 defined as a low-pass filter since it is supported on the interval $[0, 1]$ which includes the eigenvalue $\lambda_1 = 0$ associated with the constant eigenvector $\boldsymbol{v}_1$.

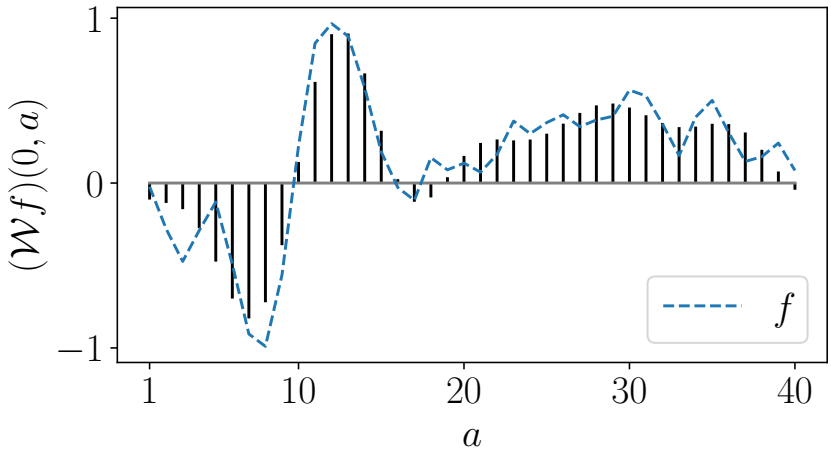


Figure 1.8 – Spectral graph wavelet coefficients at scale index $j = 0$ (black) of a graph signal example (dashed blue).

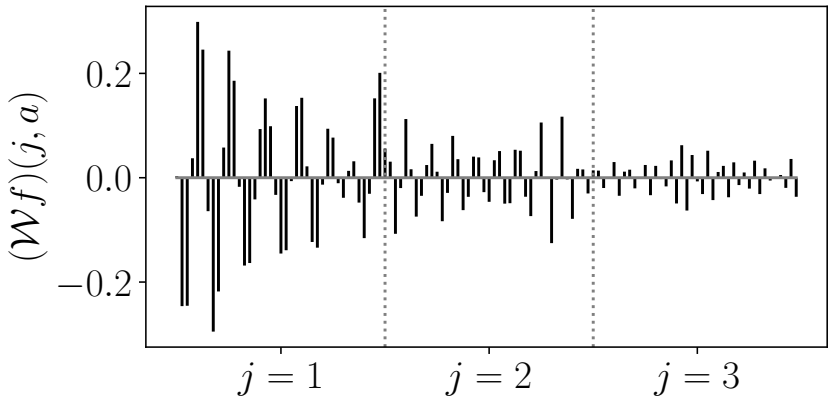


Figure 1.9 – Spectral graph wavelet coefficients at scale indices $j \geq 1$.

The spectral graph wavelet coefficients of the same graph signal at the remaining scale indices $j \geq 1$ are portrayed on Figure 1.9. With the wavelet generating functions ϕ_j being supported on larger eigenvalues, the resulting wavelet transform at these scale indices is supported by eigenvectors that oscillate more quickly which enhances the decomposition of the signal.

SGWT polynomial approximation

Direct computation of the SGWT entails functional calculus on the graph Laplacian matrix $\mathcal{L}$ and thus the computation of its eigenvectors and eigenvalues. Identically to the graph Fourier transform, this limits applications to reasonably sized graphs that have

less than a few thousand vertices. For larger ones, the computationally expensive eigen-decomposition can be avoided through a fast transform based on Chebyshev polynomial approximation. We briefly present this circumvention first introduced by the authors of the original SGWT [65] and reviewed in [72].

Their approach is to directly approximate the wavelet transform of a graph signal $\boldsymbol{f}$ at each scale $(\mathcal{W}\boldsymbol{f})(s) = \boldsymbol{\Psi}_s^\top \boldsymbol{f} = g(s\mathcal{L})^\top \boldsymbol{f} \in \mathbb{R}^n$ by using a polynomial approximation given by the truncated Chebyshev polynomial expansion of the functional calculus $g(s\mathcal{L})$. As these polynomials of $\mathcal{L}$ can be computed through simple matrix-vector multiplications, the approximation is more efficient when $\mathcal{L}$ is sparse, in other words, when the graph contains a small number of edges. An overview of Chebyshev polynomial approximation can be found in [104], we recall here a basic definition and properties useful for the fast SGWT.

The Chebyshev polynomials of the first kind $T_k(x)$ are obtained from the stable recurrence relation $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0(x) = 1$ and $T_1(x) = x$. For $x \in [-1, 1]$, they satisfy the trigonometric expression $T_k(x) = \cos(k \arccos(x))$, which means that $T_k(x)$ is bounded between -1 and 1 as well. The Chebyshev polynomials form an orthogonal basis for $L^2([-1, 1], dx/\sqrt{1-x^2})$, that is, the Hilbert space of square integrable functions with respect to the measure $dx/\sqrt{1-x^2}$.

Any filter $\rho : \mathbb{R}_+ \to \mathbb{R}_+$, and thus any wavelet generating function can be approximated with the truncated polynomial Chebyshev expansion of degree K:

$$\rho_K(\mathcal{L}) = \sum_{k=0}^{K} \theta_k(\tilde{\rho}) T_k(\widetilde{\mathcal{L}}),$$

where $\theta_k(\tilde{\rho})$ is the k^{th} coefficient of the Chebyshev expansion of $\tilde{\rho}(x) = \rho(\frac{\lambda_n}{2}(x+1))$ and $T_k(\widetilde{\mathcal{L}})$ is the Chebyshev polynomial of degree k computed for $\widetilde{\mathcal{L}} = \frac{2}{\lambda_n}\mathcal{L} - \mathbf{I}_n$. This transformation of $\mathcal{L}$ extends the expansion to any Laplacian matrix by mapping $[0, \lambda_n]$ into $[-1, 1]$. According to [65], for all filter ρ defined on $\mathrm{sp}(\mathcal{L})$ and all signal $\boldsymbol{f}$, the approximation $\rho_K(\mathcal{L})\boldsymbol{f}$ is close to $\rho(\mathcal{L})\boldsymbol{f} = (\mathcal{W}\boldsymbol{f})(s)$.

The utility of this fast SGWT relies on the efficient computation of $T_k(\widetilde{\mathcal{L}})\boldsymbol{f}$ whose computational cost is dominated by a single matrix-vector multiplication by $\widetilde{\mathcal{L}}$:

$$T_k(\widetilde{\mathcal{L}})\boldsymbol{f} = 2\widetilde{\mathcal{L}}\left(T_{k-1}(\widetilde{\mathcal{L}})\boldsymbol{f}\right) - T_{k-2}(\widetilde{\mathcal{L}})\boldsymbol{f},$$

where $T_0(\widetilde{\mathcal{L}}) = \mathbf{I}_n$ and $T_1(\widetilde{\mathcal{L}}) = \widetilde{\mathcal{L}}$. As mentioned above, this polynomial approximation is

particularly interesting for sparse Laplacian matrices since the computational complexity of applying $\mathcal{L}$ to any vector is proportional to m, the number of edges in the graph. Finally, the overall complexity of the fast SGWT requires $\mathcal{O}(mK + n(J+1)K)$ operations [65] which is greatly more efficient than the computationally expensive eigendecomposition of order $\mathcal{O}(n^3)$.

While this first approximation is more practical than the direct SGWT, it is subjected to the *Gibbs phenomenon*. The latter is more or less pronounced depending on the wavelet generating functions used to form the wavelet frame. Indeed, spectral localization is improved with functions that properly separate frequencies into distinct decomposition scales, the most efficient method being to slice the spectrum with indicator functions. As it is difficult to approximate these with polynomials, most wavelet generating functions found in the literature reach a compromise between spectral localization and approximation suitability in the case of large-scale graphs.

A solution to reduce the negative impact of the Gibbs phenomenon on the approximation error is to include Jackson coefficients g_k^K as damping multipliers in the Chebyshev expansion:

$$\rho_K(\mathcal{L}) = \sum_{k=0}^{K} g_k^K \theta_k(\tilde{\rho}) T_k(\widetilde{\mathcal{L}}).$$

An expression of these damping factors can be found in [73], a shorter form proposed in [34] is given by

$$g_k^K = \frac{\sin(k+1)\alpha_K}{(K+2)\sin(\alpha_K)} + \left(1 - \frac{k+1}{K+2}\right)\cos(k\alpha_K),$$

where $\alpha_K = \pi/(K+2)$. This Chebyshev-Jackson polynomial approximation reduces Gibbs oscillations resulting in a better convergence as the degree K increases. We illustrate this gain on Figure 1.10 where the low-pass filter ϕ_0 from Figure 1.6 is approximated with both standard and damped Chebyshev polynomials of degree $K = 20$.

On this particular example, we see that the Gibbs phenomenon is greatly reduced by including the Jackson coefficients in the polynomial approximation.

1.2 Thresholding methods for graph signal denoising

Classical wavelets transforms have been an influential complement to the traditional Fourier transform as they enable an analysis of a signal into localized oscillating components. Consequently, they were proved to be particularly useful in the common signal

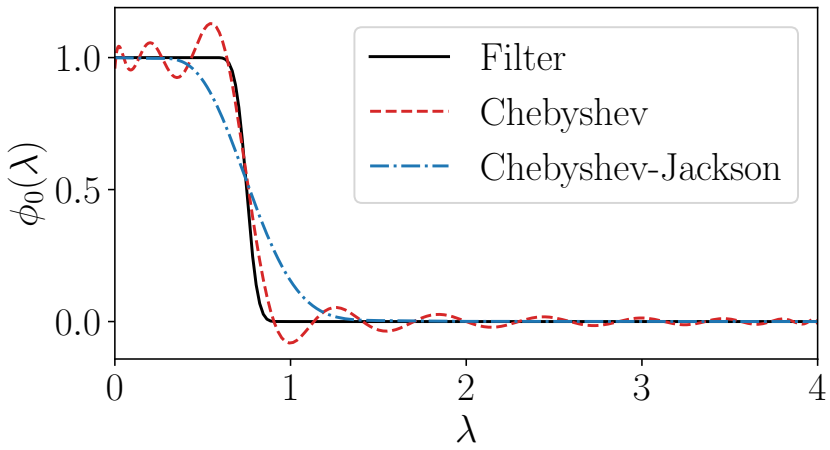


Figure 1.10 – Low-pass filter polynomial approximation

processing application that consists in removing noise from a given signal. Introductions and references may be found in the books [32], [91] and [66].

A notable *denoising* method developed by the authors of [40, 39] is based on the fact that a sparse signal can be represented with only a few wavelets coefficients. On the contrary, noise cannot be compressed in the wavelet domain considering its erratic nature. Their procedure is to apply a thresholding process to recover those few signal coefficients.

Analogously, sparsity allows for efficient representation and processing of signals on graphs. We can thus perform denoising on graph signals using the tools presented in the previous section. We now review a corresponding procedure that aims to reduce noise by thresholding the SGWT coefficients, we also present different thresholding processes and threshold selection methods. This section closes with an introduction to Stein's unbiased risk estimate in which we are particularly interested.

1.2.1 Denoising procedure

Let $\boldsymbol{f} \in \mathbb{R}^n$ be an unknown graph signal that we wish to recover from an observed signal $\tilde{\boldsymbol{f}}$. The latter is considered as a noisy variant of the former and their relationship is described by the following noise corruption model:

$$\tilde{\boldsymbol{f}} = \boldsymbol{f} + \boldsymbol{\xi},$$

where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$. The noise is thus composed of n independent and identically distributed Gaussian random variables with variance σ^2 added to each coordinate $f(i)$ of

the original signal.

The purpose of denoising is to build an estimator $\hat{\boldsymbol{f}}$ of $\boldsymbol{f}$ that depends only on $\tilde{\boldsymbol{f}}$. We present here an approach based on the sparse representation of the observed signal in an appropriate transformed domain. As seen above, the energy of a relatively smooth graph signal is mostly concentrated in the low frequencies of the spectrum. Therefore, the projection of this signal on a multiscale frame whose elements are localized in the spectral domain is itself mainly supported by the lower scales. On the other hand, the energy of the transformed noise is more distributed along the scales. As a result, the intuition is to reduce this noise background by applying a thresholding process in the transformed domain that separates it from the original signal as much as possible.

The spatial and spectral localization properties of the SGWT are particularly useful to process signals with heterogeneous regularity in the vertex domain compared to other transforms like the graph Fourier transform. Our denoising procedure consists in thresholding the spectral graph wavelet coefficients of a graph signal in the transformed domain. This can be viewed as an extension of the classical wavelet methodology from [40, 39] to the SGWT. It is summarized as follows:

1. Compute the SGWT of the observed signal: $\mathcal{W}\tilde{\boldsymbol{f}} = \boldsymbol{\Psi}^\top \tilde{\boldsymbol{f}}$
2. Apply a given thresholding process h to the coefficients: $\mathcal{W}\hat{\boldsymbol{f}} = h(\mathcal{W}\tilde{\boldsymbol{f}})$
3. Compute the inverse SGWT to get an estimate: $\hat{\boldsymbol{f}} = \mathcal{W}^*\mathcal{W}\hat{\boldsymbol{f}}$

Because the SGWT is a linear operator, the noise corruption model $\tilde{\boldsymbol{f}} = \boldsymbol{f} + \boldsymbol{\xi}$ translates in the wavelet domain to the following model:

$$\widetilde{\boldsymbol{F}} = \boldsymbol{F} + \boldsymbol{\Xi},$$

where $\widetilde{\boldsymbol{F}} = \mathcal{W}\tilde{\boldsymbol{f}}$, $\boldsymbol{F} = \mathcal{W}\boldsymbol{f}$ and $\boldsymbol{\Xi} = \mathcal{W}\boldsymbol{\xi}$ are respectively the spectral graph wavelet coefficients of the noisy signal, original signal and added noise. By linearity, the SGWT of $\boldsymbol{\xi}$ is also a centered Gaussian random vector whose covariance matrix is given by $\mathbb{V}(\boldsymbol{\Xi}) = \mathbb{V}(\boldsymbol{\Psi}^\top\boldsymbol{\xi}) = \boldsymbol{\Psi}^\top\mathbb{V}(\boldsymbol{\xi})\boldsymbol{\Psi} = \sigma^2\boldsymbol{\Psi}^\top\boldsymbol{\Psi}$. It is important noting that $\boldsymbol{\Psi}^\top\boldsymbol{\Psi}$ is not the identity matrix but an orthogonal projector of rank n in $\mathbb{R}^{n(J+1)}$ as the wavelet frame $\boldsymbol{\Psi}$ is only a semi-orthogonal matrix. In other words, the transformed noise $\boldsymbol{\Xi}$ obtained with the SGWT is both autocorrelated and heteroscedastic contrary to the classical wavelet transform or the graph Fourier transform which are orthogonal.

This feature is inherent to any localized, multiscale transform due to the overcompleteness of its associated frame, that is, the family of wavelets $\{\boldsymbol{\psi}_i\}_{i=1,\dots,n(J+1)}$ in the case of

the SGWT. Autocorrelation given by $\mathrm{cov}(\Xi(i), \Xi(j)) = \sigma^2 \boldsymbol{\psi}_i^\top \boldsymbol{\psi}_j$ arises from the spectral graph wavelets encoding the topological information about the neighborhood of their corresponding vertex. Since the graph vertices share a common environment, autocorrelation is particularly notable within each scale and can be present in between. Incidentally, such correlated graph wavelets find an application in community detection [127] where vertices that share a similar environment are grouped together. From a spectral point of view, the graph wavelets of a given scale are naturally expected to be correlated considering they are linear combinations of the same eigenvectors.

As for hereroscedasticity, the different variance values $\mathbb{V}(\Xi(i)) = \sigma^2 \boldsymbol{\psi}_i^\top \boldsymbol{\psi}_i = \sigma^2 \|\boldsymbol{\psi}_i\|_2^2$ are a product of the spectral graph wavelets each having a distinct energy that depends both on the scale and associated vertex. This heterogeneity of variance is observed on Figure 1.11 where the SGWT coefficients of the realization of a centered Gaussian random vector with variance $\sigma^2 = 0.04$ are illustrated. The graph and wavelet frame are the same as the ones from the previous examples.

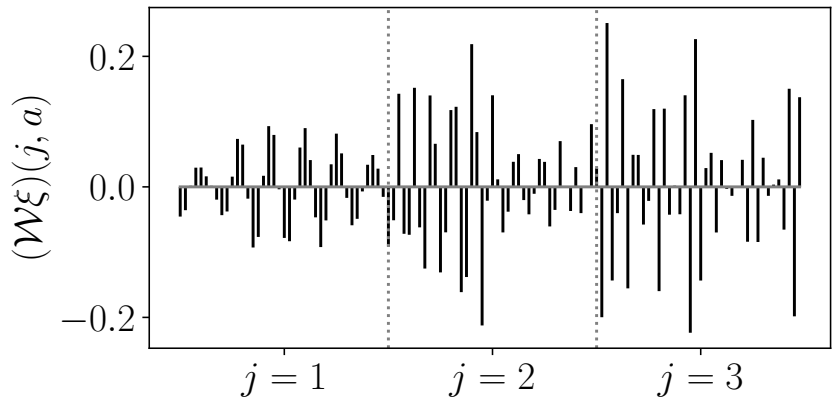


Figure 1.11 – Spectral graph wavelet coefficients at scale indices $j \geq 1$ of a centered Gaussian random vector realization.

Specifically, we notice that the variability of the noise wavelet coefficients intensifies as the scale index increases. This configuration where the energy of the transformed noise is mostly supported by the finer decomposition scales is conducive to an effective separation between the signal and noise in the wavelet domain. It shows a suitable decomposition frame is necessary in the first place before attempting to reduce noise.

1.2.2 Diagonal estimation

The main issue in the denoising procedure described above is to choose an appropriate thresholding process h in regard to both the noisy signal and the underlying graph. Before introducing this kind of process, we first present a linear map $h : \mathbb{R}^{n(J+1)} \to \mathbb{R}^{n(J+1)}$ which attenuates or removes individual wavelet coefficients by application of a diagonal matrix:

$$\widehat{\boldsymbol{F}} = h(\widetilde{\boldsymbol{F}}) = \mathbf{H}\widetilde{\boldsymbol{F}},$$

where $\widehat{\boldsymbol{F}}$ is the estimator in the wavelet domain and $\mathbf{H} = \operatorname{diag}(h_1, \ldots, h_{n(J+1)})$. The resulting estimator $\hat{\boldsymbol{f}} = \mathcal{W}^* \widehat{\boldsymbol{F}}$ is fittingly called a *diagonal estimator*. In the case where the elements of the diagonal operator $\mathbf{H}$ are restricted to $h_i \in \{0, 1\}$ for all $i = 1, \ldots, n(J+1)$, the process is equivalent to linear projection.

If the h_i are taken in the interval $[0, 1]$ and chosen such that the risk $\mathbb{E}\left[\|\boldsymbol{f} - \hat{\boldsymbol{f}}\|_2^2\right]$ is minimized, the produced estimator corresponds to the *Wiener filter*. The attenuating coefficients h_i of this estimator are given by

$$h_i = \frac{|F(i)|^2}{|F(i)|^2 + \sigma^2},$$

in which case the risk is minimum [91]:

$$r_{\text{inf}}(\boldsymbol{f}) = \sum_{i=1}^{n(J+1)} \frac{|F(i)|^2 \sigma^2}{|F(i)|^2 + \sigma^2}.$$

The Wiener filter cannot be computed in practice since it depends on the unknown signal $\boldsymbol{f}$ and is known as an *oracle* estimator for this reason. The risk $r_{\text{inf}}(\boldsymbol{f})$ is thus a lower bound that is unreachable but can be approached with specific thresholding estimators in the specific case of classical wavelet coefficients.

1.2.3 Thresholding estimation

When the map h is replaced by a thresholding process, the denoising procedure yields a *thresholding estimator* in line with the classical wavelet approach from [40, 39]. This process can either perform *coordinatewise* or *block* thresholding. We focus here on coordinatewise thresholding of spectral graph wavelet coefficients. More information on the block thresholding process can be found in [20] where it is applied in conjunction with

the classical wavelet transform and in [86] which extends it to the SGWT.

A coordinatewise thresholding process h is of the form $h(\boldsymbol{x}) = (\tau(x_i, t_i))_{i=1,\dots,n(J+1)}$ where the $(t_i)_{i=1,\dots,n(J+1)} \in \mathbb{R}_+^{n(J+1)}$ are threshold values and τ is a thresholding function. These can be fine-tuned to take into account the transformed noise heteroscedasticity with a suitable adjustment based on the spectral graph wavelets: $t_i' = t_i\sqrt{\mathbb{V}(\Xi(i))}/\sigma = t_i\|\boldsymbol{\psi}_i\|_2$. In the case of a uniform threshold along every coefficient: $t_i = t \in \mathbb{R}_+$ for all i, we say that h is a *global* thresholding process. When the threshold has different values $\boldsymbol{t}_0, \dots, \boldsymbol{t}_J$ with $\boldsymbol{t}_j \in \mathbb{R}_+^n$ on each scale j of the transform, the thresholding process is said to be *level-dependent*. The lowest scale index $j = 0$ is usually thresholded contrary to the classical wavelet methodology where the coefficients of the first decomposition scale are constant.

The most common thresholding functions in the literature are given by

$$\tau_\beta(x, t) = x \max\{1 - \frac{t^\beta}{|x|^\beta}, 0\},$$

with $\beta \geq 1$. Different choices for parameter β produce popular functions that include soft thresholding ($\beta = 1$), James-Stein thresholding ($\beta = 2$) and hard thresholding ($\beta = \infty$) which is defined by $\tau(x, t) = x\mathbf{1}_{\{|x|>t\}}$.

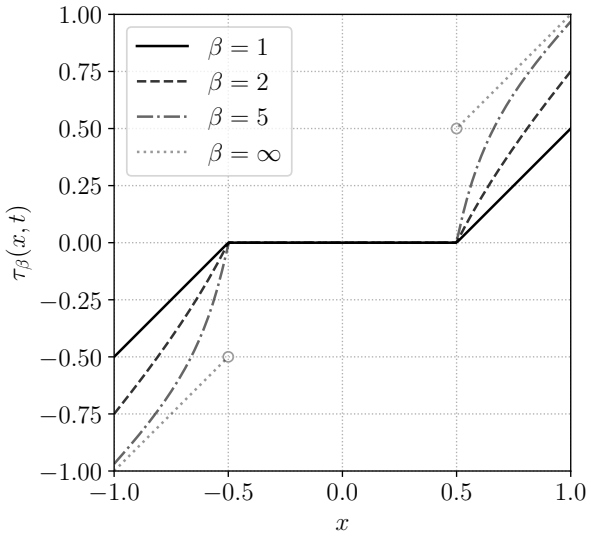


Figure 1.12 – Examples of thresholding functions

These thresholding functions illustrated on Figure 1.12 with a threshold $t = 0.5$ seek to extract noise from the signal by removing the small spectral graph wavelet coefficients and attenuating the large ones. Whereas hard thresholding leaves the wavelet coefficients

unchanged beyond the threshold value, the remaining thresholding functions do decrease their amplitude and even more so β is small.

1.2.4 Threshold selection

The choice of the threshold values t_i is crucial as these determine to what extent the wavelet coefficients should be attenuated through the thresholding functions presented above. We now give a few existing methods to select appropriate thresholds.

Universal threshold

As mentioned above, the authors of [40] provide a classical wavelet theorem which proves that for an appropriate choice of t, the risk of a soft or hard thresholding estimator is close to the minimum risk of the Wiener filter. This particular global threshold is called the *universal threshold* and is given by $t_{\text{univ}} = \sigma\sqrt{2\ln N}$, where N is the sample size. It is based on the result [13] that the maximum amplitude of the noise has a very high probability of being just below t_{univ}.

This theorem is however conditional on the transformed noise being a decorrelated and homoscedastic Gaussian random vector, that is, when the frame is orthogonal like in the case of the classical wavelet transform. Since the SGWT is only semi-orthogonal, the universal threshold gives no guarantee on the risk of the thresholding estimators presented above. Furthermore, it should be noted that this threshold is not optimal as the risk can be reduced in general by choosing a smaller threshold. The minimization of a performance measure therefore remains an effective strategy to empirically select optimal threshold values.

Mean squared error

The empirical risk or *mean squared error* (MSE) is an oracle performance measure because it depends on the unknown original signal, its minimization thus produces an oracle estimator. The MSE between the original signal $\boldsymbol{f}$ and the denoised signal $\hat{\boldsymbol{f}}$ is given by

$$\mathbf{MSE}(\boldsymbol{f}, \hat{\boldsymbol{f}}) = \frac{1}{n}\|\boldsymbol{f} - \hat{\boldsymbol{f}}\|_2^2 = \frac{1}{n}\sum_{i=1}^{n}(f(i) - \hat{f}(i))^2.$$

Because the SGWT and its inverse verify Parseval's identity, the energy of the signal is preserved in both the vertex and wavelet domains. This means the optimization to

determine a suitable thresholding process can directly be performed in the transformed domain of the wavelet coefficients. Indeed, we have

$$\begin{aligned}\|\boldsymbol{f} - \hat{\boldsymbol{f}}\|_2^2 &= \|\boldsymbol{f} - \mathcal{W}^* h(\mathcal{W}\tilde{\boldsymbol{f}})\|_2^2 \\ &= \|\mathcal{W}^*(\mathcal{W}\boldsymbol{f} - h(\mathcal{W}\tilde{\boldsymbol{f}}))\|_2^2 \\ &= \|\mathcal{W}\boldsymbol{f} - h(\mathcal{W}\tilde{\boldsymbol{f}})\|_2^2 \\ &= \|\boldsymbol{F} - h(\widetilde{\boldsymbol{F}})\|_2^2 \\ &= \|\boldsymbol{F} - \widehat{\boldsymbol{F}}\|_2^2.\end{aligned}$$

As a result, it is not necessary to apply the inverse SGWT at each optimizing step in order to evaluate the empirical performance of the estimator $\hat{\boldsymbol{f}}$ in the vertex domain. This significantly reduces the complexity of the optimization process and grants more resources towards parameter tuning.

Stein's unbiased risk estimate

The MSE can be substituted with another performance measure called *Stein's unbiased risk estimate* (SURE). Introduced by [121], it estimates with no bias the risk of a nearly arbitrary, nonlinear biased estimator and only depends on the noisy signal and the noise variance. SURE has been used in the literature [39] as the optimization criterion of a thresholding estimator with classical wavelet coefficients and is extendable to the SGWT according to the following theorem from [86].

Theorem 1.2.1 (h-SURE)**.** *Let h be a weakly differentiable thresholding process for the denoising problem $\widetilde{\boldsymbol{F}} = \boldsymbol{F} + \boldsymbol{\Xi}$. Then the theoretical risk is given by*

$$\mathbb{E}\left[\|\boldsymbol{F} - h(\widetilde{\boldsymbol{F}})\|_2^2\right] = \mathbb{E}\left[-n\sigma^2 + \|\widetilde{\boldsymbol{F}} - h(\widetilde{\boldsymbol{F}})\|_2^2 + 2\sigma^2 \sum_{i,j=1}^{n(J+1)} \gamma_{ij}\partial_j h_i(\widetilde{\boldsymbol{F}})\right],$$

where $\gamma_{ij} = (\boldsymbol{\Psi}^\top\boldsymbol{\Psi})_{ij}$ and h_i is the i^{th} component of h.

Once a thresholding process h is chosen and the noise variance σ^2 is estimated or known, SURE of h in the transformed domain reads as

$$\mathbf{SURE}(h) = -n\sigma^2 + \|\widetilde{\boldsymbol{F}} - h(\widetilde{\boldsymbol{F}})\|_2^2 + 2\sigma^2 \sum_{i,j=1}^{n(J+1)} \gamma_{ij}\partial_j h_i(\widetilde{\boldsymbol{F}}).$$

This performance measure is defined for any weakly differentiable thresholding process which covers both diagonal estimators and thresholding estimators with the exception of hard thresholding. In particular, when h is a coordinatewise thresholding process that applies the thresholding function τ_β with $\beta \in [1, \infty)$ to each wavelet coefficient, the formula becomes

$$\begin{aligned}\mathbf{SURE}(h) = -n\sigma^2 + \sum_{i=1}^{n(J+1)} \widetilde{F}(i)^2 \left[1 \wedge \frac{t_i^\beta}{|\widetilde{F}(i)|^\beta}\right]^2 \\ + 2\sigma^2 \sum_{i=1}^{n(J+1)} \gamma_{ii} \left[1 + \frac{(\beta-1)t_i^\beta}{|\widetilde{F}(i)|^\beta}\right] \mathbf{1}_{[t_i,\infty)}(|\widetilde{F}(i)|).\end{aligned}$$

Despite the fact the SURE performance measure does not depend on the original signal, its expression includes the noise variance σ^2. In the case where it is not known for a particular application, we present here a naive estimator from [86] and [87] which stems from the following observation:

$$\begin{aligned}\mathbb{E}\left[\tilde{\boldsymbol{f}}^\top \mathcal{L} \tilde{\boldsymbol{f}}\right] &= \boldsymbol{f}^\top \mathcal{L} \boldsymbol{f} + \mathbb{E}\left[\boldsymbol{\xi}^\top \mathcal{L} \boldsymbol{\xi}\right] \\ &= S_2(\boldsymbol{f}) + \sigma^2 \operatorname{tr}(\mathcal{L}),\end{aligned}$$

where $S_2(\boldsymbol{f})$ is the Laplacian quadratic form presented in Section 1.1.1 that measures the smoothness of $\boldsymbol{f}$. If the original signal is relatively smooth such that $S_2(\boldsymbol{f})$ is negligible compared to $\operatorname{tr}(\mathcal{L})$, the noise variance can reasonably be approximated with the biased estimator

$$\hat{\sigma}^2 = \frac{\tilde{\boldsymbol{f}}^\top \mathcal{L} \tilde{\boldsymbol{f}}}{\operatorname{tr}(\mathcal{L})} = \frac{\sum_{\{i,j\} \in \mathcal{E}} w_{ij} (\tilde{f}(i) - \tilde{f}(j))^2}{\sum_{i,j=1}^n w_{ij}}.$$

This quantity corresponds to a graph analogue of the von Neumann estimator [130]. It has the benefit of only relying on known values but suffers from a bias strongly dependent on the underlying signal smoothness.

Figure 1.13 shows a comparison between minimizing the MSE and SURE for different threshold values. The graph signal and noise coefficients are the ones from Figures 1.8, 1.9 and 1.11. Noise is reduced by coordinatewise soft thresholding where a global threshold t is applied to all spectral graph wavelet coefficients.

Although the risk is overestimated by SURE here, the optimal threshold value is relatively close to the one obtained with the oracle estimator which minimizes the MSE.

Lastly, an important limitation is the numerical complexity regarding the weights

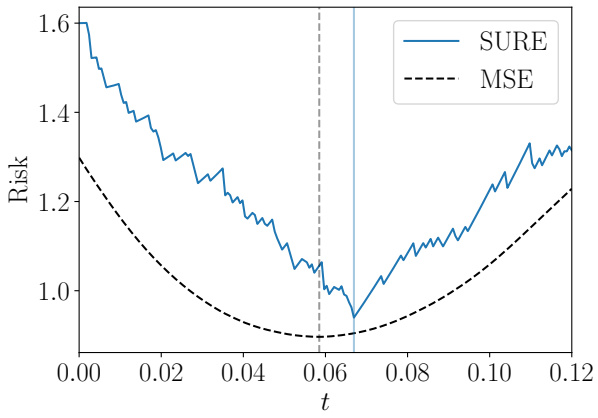


Figure 1.13 – MSE and SURE minimization

$\gamma_{ij} = (\mathbf{\Psi}^\top \mathbf{\Psi})_{ij}$ contained in the SURE formula. They entirely depend on the wavelet frame matrix and thus the computationally expensive eigendecomposition of the graph Laplacian matrix $\mathcal{L}$. For this reason, directly computing SURE is not tractable for signals defined on large graphs.

A solution is to take advantage of their relationship with the covariance between transformed Gaussian random vectors: $\gamma_{ij} = \mathrm{cov}(\Xi(i), \Xi(j))/\sigma^2$. By combining the SGWT polynomial approximation presented in Section 1.1.3 with the Monte Carlo method, it is possible to estimate these weights before plugging them in the SURE expression. A more detailed presentation and evaluation of this approach is given in Chapter 3.

Chapter 2

Differential privacy and application to graphs

In this chapter, we focus on the research field of differential privacy, its use in industry, and possible applications to graphs. This first section introduces different definitions of differential privacy, some mechanisms for achieving it, and interesting properties about their composition. In particular, differential privacy guarantees remain unaffected by any post-processing such as the denoising procedure presented in the previous chapter. In a second section, we present how differential privacy is used in industry and some of its applications to graphs.

2.1 Introduction to differential privacy

2.1.1 Pure differential privacy

In this subsection, we give the initial definition of ε-differential privacy proposed by the authors of [46, 43], present a few common privacy mechanisms that fall within and summarize some interesting properties.

Definition

Suppose a number of m individuals each owns a piece of sensitive information we wish to use for statistical analysis and protect at the same time through a *privacy mechanism*. We model these private data as a random vector $(X_1, \ldots, X_m)$ where each component X_i is a random variable. This information is collected in a dataset $X = (X_1, \ldots, X_m)$ that serves as an input to the privacy mechanism M which returns a *sanitized* output $Z = (Z_1, \ldots, Z_k) = M(X)$ that preserves the privacy of each individual. This output can be of different size from the protected dataset. For instance, a sanitized summary statistic corresponds to a single released value ($k = 1$) while a sanitized version of the whole dataset

is the same size ($k = m$). Let $(\mathcal{X}^m, \mathcal{A}^m)$ and $(\mathcal{Z}, \mathcal{B})$ be the measurable spaces where X and Z respectively take values. The privacy mechanism distribution $Q(\cdot|X)$ corresponds to the conditional distribution of Z given X and is supposed regular. That is, the map $Q(\cdot|\cdot) : \mathcal{B} \times \mathcal{X}^m \to [0, 1]$ satisfies (i) for all $x \in \mathcal{X}^m$, $Q(\cdot|x)$ is a probability measure on $(\mathcal{Z}, \mathcal{B})$ and (ii) for all $B \in \mathcal{B}$, $Q(B|\cdot)$ is $\mathcal{A}^m$-measurable. Thus, Q is a Markov kernel.

To define *differential privacy*, we need a measure of distance between two datasets that gives the number of entries in which they *differ*. The Hamming distance fulfills this function when applied to two datasets X, X' of same size m, we define it as follows: $\mathrm{d}(X, X') \coloneqq \#\{i \in \{1, \dots, m\} \mid X_i \neq X'_i\}$. When $\mathrm{d}(X, X') = 1$, the datasets differ in one entry and are said to be *adjacent*.

We now give the formal definition of ε-differential privacy (or *pure* differential privacy) first introduced in [46].

Definition 2.1.1 (ε-differential privacy)**.** Let $\varepsilon \geq 0$. The privacy mechanism M is said to satisfy ε-differential privacy if

$$Q(B|x) \leq e^{\varepsilon} Q(B|x'), \quad \forall B \in \mathcal{B}, \forall x, x' \in \mathcal{X}^m : \mathrm{d}(x, x') = 1.$$

The intuition behind this definition is that the distribution $Q(\cdot|X)$ of the sanitized output Z does not change much when the observed dataset x is altered on a single entry x_i. In other words, differential privacy ensures the inclusion or removal of any given individual has a negligible impact on the privacy mechanism results. As such, the promise is that their private information is protected as their participation does not make too much of a difference.

In the case of ε-differential privacy, this variation in the output distribution is bounded from above by e^{ε}. As this quantity comes close to 1, the distributions behave similarly and the privacy guarantee is stronger. The privacy parameter ε (also called *privacy budget*) thus indicates how much privacy-preserving the mechanism M is: the closer it is to 0, the safer is the sensitive information. This measure of privacy is a helpful tool when comparing different mechanisms. For instance, one can choose a method that provides better privacy among several with similar performance.

This standard definition uses the Hamming distance to control that X and X' are adjacent input datasets. Alternative definitions of differential privacy can be formulated with different notions of adjacency. Furthermore, extending the distance to an arbitrary metric is a way to generalize differential privacy to domains other than datasets [22].

In practice, differential privacy is achieved by introducing randomness in the process. We now present a first ε-differentially private mechanism and observe how the degrees of randomness and privacy relate to each other.

Randomized response

Assume you wish to survey a population to estimate the proportion of people who have a certain property. If the latter is associated with embarrassing or illegal behavior, the participants may be reluctant to give an honest answer. This can result in a biased estimate due to untruthful responses or even refusal of being surveyed. A solution proposed by [135] consists in randomizing the response of the participant to guarantee their privacy. One way to perform this is by letting the respondent answer honestly or lie depending on the result of a random experiment. We give an example of such a *randomized response* mechanism.

Example 2.1.1. Participants are asked to follow this procedure before answering whether they have the property of interest:

1. Roll a die
2. If the result is between **1** and **4**, then respond truthfully.
3. If the result is **5** or **6**, then lie.

With this method, it is not possible to know if a given respondent has the property since there is a chance they made a wrong statement. The randomized response mechanism gives them *plausible deniability* as they can rightfully claim the die modified their true answer. Although the information is uncertain on the individual level, it is still possible to estimate the true proportion of the population. Indeed, the randomness in the procedure having known results and corresponding probabilities, we can use a statistical model for this purpose.

Let X be a random variable which takes the value 1 with probability p if the participant has the property of interest and 0 otherwise. This corresponds to a Bernoulli distribution with parameter p that represents the proportion we want to estimate: $X \sim \text{Bernoulli}(p)$. Let another random variable Y be the result of the die roll that follows a discrete uniform distribution: $Y \sim \mathcal{U}_{\{1,6\}}$. Finally, the answer given by the respondent at the end of the procedure is represented by a third random variable $Z = X\mathbf{1}_{\{Y \leq 4\}} + (1 - X)\mathbf{1}_{\{Y > 4\}}$ which takes the value 1 if the final answer is "Yes" and 0 if "No" is given. To better visualize the randomized response mechanism, Figure 2.1 represents its associated tree diagram.

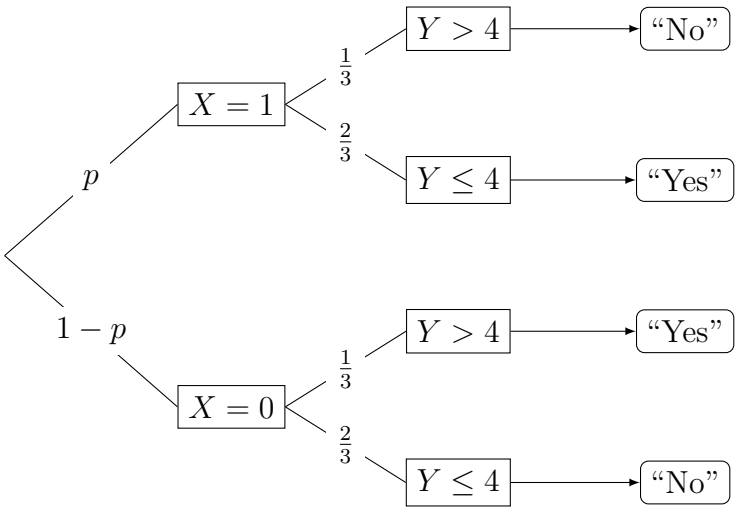


Figure 2.1 – Randomized response tree diagram

As X and Y are independent, we can easily compute the probability of a participant answering "Yes": $\mathbb{P}(Z = 1) = \mathbb{P}(X = 1)\mathbb{P}(Y \leq 4) + \mathbb{P}(X = 0)\mathbb{P}(Y > 4) = \frac{2p}{3} + \frac{1-p}{3} = \frac{1+p}{3}$. The random variable Z thus also follows a Bernoulli distribution with parameter and expected value $\frac{1+p}{3}$. If m individuals are surveyed with this procedure, we obtain a random sample $Z_1, \ldots, Z_m$ of responses whose sample mean $\bar{Z}_m = \frac{1}{m}\sum_{i=1}^{m} Z_i$ is a consistent estimator of $\frac{1+p}{3}$ according to the law of large numbers. With the appropriate transformation, we can therefore directly estimate the proportion p with $3\bar{Z}_m - 1$.

This example shows how the randomized response mechanism protects the private data $X_1, \ldots, X_m$ from disclosure and at the same time makes estimation from the sanitized output $Z_1, \ldots, Z_m$ possible. While this method was introduced more than 40 years before ε-differential privacy, it does fit its definition. Theoretical analysis under differential privacy of this mechanism and its variants has been conducted since, see for example [136] and [133].

We can determine the privacy budget ε for the specific randomized response example presented above by first computing the probabilities $Q(z|x) = \mathbb{P}(Z = z|X = x)$ for $z, x \in \{0, 1\}$:

$$\mathbb{P}(Z = 1|X = 1) = \mathbb{P}(Z = 0|X = 0) = \mathbb{P}(Y \leq 4) = 2/3,$$
$$\mathbb{P}(Z = 1|X = 0) = \mathbb{P}(Z = 0|X = 1) = \mathbb{P}(Y > 4) = 1/3.$$

Then, we have the probability ratios

$$\frac{\mathbb{P}(Z=1|X=1)}{\mathbb{P}(Z=1|X=0)} = \frac{\mathbb{P}(Z=0|X=0)}{\mathbb{P}(Z=0|X=1)} = \frac{2/3}{1/3} = 2.$$

This gives the upper bound

$$\frac{Q(z|x)}{Q(z|x')} \leq e^{\ln 2}, \quad \forall z \in \{0,1\}, \forall x, x' \in \{0,1\} : \mathrm{d}(x,x') = 1.$$

As a result, this randomized response mechanism provides ln(2)-differential privacy.

Let us see how the privacy budget ε behaves when the randomness in the mechanism changes. Suppose now the participants are asked to lie only if they roll a 6. In this case the distribution of Z is modified and the privacy budget becomes $\ln \frac{5/6}{1/6} = \ln 5$. This higher value of ε corresponds to a lower degree of privacy as the respondents are more likely to make a true statement. On the other hand, the estimator of the proportion p benefits from a smaller variance since more information is collected over the m individuals.

We can better study the relationship between privacy and noise in this mechanism by writing T the random variable which takes the value 1 if the participant answers truthfully and 0 if not. We generalize the probability of a honest answer being given to π so that $T \sim \text{Bernoulli}(\pi)$. For $\pi \in [\frac{1}{2}, 1]$, the privacy budget is given by $\varepsilon = \ln \frac{\mathbb{P}(T=1)}{\mathbb{P}(T=0)} = \ln \frac{\pi}{1-\pi}$. As π approaches 1, the procedure comes closer to a regular survey with no privacy protection and ε tends to ∞. In that situation, the answers give the most information and the proportion estimator has the lowest variance. If instead, the respondents are asked to tell the truth half the time on average ($\pi = \frac{1}{2}$) with a coin toss for example, then privacy is at its highest degree with $\varepsilon = 0$. No information is revealed from the responses as they are no longer related to the proportion $p : \mathbb{P}(Z=1) = \frac{1}{2}$. Statistical analysis in this extreme case is impossible because the sanitized data are useless. Note that for $\pi \in [0, \frac{1}{2}]$, the degree of privacy becomes $\varepsilon = \ln \frac{1-\pi}{\pi}$ and thus behaves the same as π goes from $\frac{1}{2}$ to 0. The privacy budget here depends in fact on the variance of T given by $\mathbb{V}(T) = \pi(1-\pi)$ which measures the amount of noise in the procedure.

In general, the more randomness there is in a privacy mechanism, the more protection it provides to the sensitive data. The randomized response example above also shows that this relationship necessarily implies a compromise between statistical utility and privacy. Caution must be taken when sanitizing data so that enough relevant information is preserved in the process.

Laplace mechanism

Besides binary data, differential privacy can also be used to sanitize numeric queries on datasets. For instance, we can be interested by the empirical means of the n variables of a dataset composed of m individuals. Assuming the results of the queries are real-valued, we want to protect the output $f(X)$, where $f : \mathcal{X}^m \to \mathbb{R}^k$ is the query function. The *Laplace mechanism* [46] does so by adding noise drawn from a centered Laplace distribution whose scale is calibrated to a quantity called the ℓ_1-sensitivity of f.

Definition 2.1.2 (ℓ_1-sensitivity)**.** The ℓ_1-sensitivity of a function $f : \mathcal{X}^m \to \mathbb{R}^k$ is

$$\Delta_1 f = \max_{\substack{x,x' \in \mathcal{X}^m \\ \mathrm{d}(x,x')=1}} \|f(x) - f(x')\|_1.$$

As it appears from this definition, this value measures how much a single entry from any dataset can affect the query result at most. That is, the biggest difference we can obtain in the output by including or removing an individual. Note that the ℓ_1-sensitivity does not depend on the observed dataset, it is a theoretical quantity intrinsic to the function f. This gives an upper bound on the amount of randomness that must be introduced by the mechanism to preserve privacy. Indeed, if the added noise is enough to protect against the largest shift an individual can cause, then all of them are covered.

Example 2.1.2. We now provide a few examples of ℓ_1-sensitivity for usual queries on datasets.

— Count: "How many women are there in the dataset?"
A counting query counts the number of dataset entries that have a certain property. It can be seen as a sum $f(x) = \sum_{i=1}^m x_i$ where $x \in \mathcal{X}^m = \{0,1\}^m$.

$$\Delta_1 f = \max \left| \sum_{i=1}^m x_i - \sum_{i=1}^m x'_i \right| = \max \left| x_{i_0} - x'_{i_0} \right| = 1, \quad \text{for some } i_0 \in \{1, \dots, m\}.$$

— Sum: "What is the total income of all people in the dataset?"
If there is no information on the range of values an attribute $x \in \mathbb{R}^m$ can take, the resulting sensitivity of $f(x) = \sum_{i=1}^m x_i$ is unbounded: $\Delta_1 f = \max |x_{i_0} - x'_{i_0}| = +\infty$. When the x_i are known to lie in an interval $[a, b]$, the sensitivity is $\Delta_1 f = b - a$.

— Average: "What is the average screen time of all users in the dataset?"
For $\mathcal{X}^m = [a, b]^m$, the potential impact of each individual on the averaging query $f(x) = \frac{1}{m} \sum_{i=1}^m x_i$ depends on the dataset size: $\Delta_1 f = \max \frac{1}{m} |x_{i_0} - x'_{i_0}| = \frac{b-a}{m}$.

These examples show that prior knowledge or hypotheses on the range of possible values are necessary to bound the sensitivity of some queries. Reasonable intervals can be found in some cases (ex: body height or a specific time) while in others it might be more difficult (ex: market capitalization of a company). A solution is to enforce lower and upper bounds by *clipping* the data we wish to protect at the cost of losing statistical utility. Removing extreme values is an effective way to reduce the sensitivity and thus the amount of noise needed to ensure privacy. However, the chosen interval must not depend on the private data otherwise the differential privacy guarantee would be degraded as the bounds themselves could reveal something about the data. In practice, the clipping bounds are hyperparameters of the privacy mechanism tuned according to preexisting knowledge on the data when available.

The ℓ_1 distance between the query results was purposely chosen to go with the Laplace distribution. The probability density function of a centered Laplace distribution with scale b is given by $g(y) = (1/2b)\exp(-|y|/b)$ and we write $Y \sim \mathrm{Lap}(b)$.

Definition 2.1.3 (Laplace mechanism)**.** Given any function $f : \mathcal{X}^m \to \mathbb{R}^k$, the Laplace mechanism is defined as:

$$Z = f(X) + Y,$$

where $Y = (Y_1, \ldots, Y_k)$ and $Y_j \overset{\text{i.i.d.}}{\sim} \mathrm{Lap}(\Delta_1 f/\varepsilon)$.

This mechanism can be proven to satisfy ε-differential privacy by showing that the distributions $Q(\cdot|X)$ and $Q(\cdot|X')$ are close in the sense of Definition 2.1.1. Let $q(z|x)$ denote the probability density function of Z given $X = x$, for all $z \in \mathbb{R}^k$ it reads:

$$q(z|x) = \prod_{j=1}^{k} \frac{\varepsilon}{2\Delta_1 f} \exp\left(-\frac{\varepsilon|z_j - f(x)_j|}{\Delta_1 f}\right).$$

Thus, for all $x, x' \in \mathcal{X}^m$ such that $\mathrm{d}(x, x') = 1$, we have the ratio

$$\begin{aligned}\frac{q(z|x)}{q(z|x')} &= \prod_{j=1}^{k} \exp\left(\frac{\varepsilon(|z_j - f(x')_j| - |z_j - f(x)_j|)}{\Delta_1 f}\right) \\ &\leq \prod_{j=1}^{k} \exp\left(\frac{\varepsilon|f(x)_j - f(x')_j|}{\Delta_1 f}\right) \\ &= \exp\left(\frac{\varepsilon\|f(x) - f(x')\|_1}{\Delta_1 f}\right) \\ &\leq \exp\left(\frac{\varepsilon\Delta_1 f}{\Delta_1 f}\right) = \exp(\varepsilon),\end{aligned}$$

where the first inequality is given by the triangle inequality and the second by the sensitivity bound $\|f(x) - f(x')\|_1 \leq \Delta_1 f$. Symmetrically, we can find the lower bound $\exp(-\varepsilon)$.

Properties

Differential privacy has several attractive properties that are useful when designing more complex privacy mechanisms. We present here three of them: sequential composition, parallel composition and post-processing. Finally, we mention a few studies about the statistical properties of differential privacy.

Statistical analysis often requires to consult the data multiple times. The *sequential composition* property of differential privacy exactly quantifies the privacy degree of a sequence of privacy mechanisms. The first sequential composition theorem for differential privacy was introduced by [47, Theorem 1], a more complete one is given in [44, Theorem 3.14, Corollary 3.15].

Theorem 2.1.1 (Sequential composition)**.** *If each privacy mechanism $M_j(X)$, chosen sequentially or adaptively, satisfies ε_j-differential privacy for $j = 1, \ldots, k$, then the sequence $M(X) = (M_1(X), \ldots, M_k(X))$ satisfies $(\sum_{j=1}^{k} \varepsilon_j)$-differential privacy.*

As one would expect, the privacy guarantee degrades as the same dataset X appears several times in the sequence. The main convenience of differential privacy is the additive accumulation of the privacy budgets in the worst case.

Example 2.1.3 (Average and variance)**.** This property is particularly useful when multiple differentially private statistics are simultaneously released. Suppose we are interested in both the average $f_1(x) = \frac{1}{m}\sum_{i=1}^{m} x_i$ and variance $f_2(x) = \sum_{i=1}^{m} \frac{1}{m}(x_i - f_1(x))^2$ of an

attribute $x \in \mathcal{X}^m$. Sanitized versions $M_1(X)$ and $M_2(X)$ of these queries can be obtained by adding Laplace noise such that they each satisfy ε-differential privacy. According to sequential composition, the mechanism $M(X) = (M_1(X), M_2(X))$ which releases both statistics is 2ε-differentially private. This guarantee is valid whether the sanitized variance is computed independently from the sanitized average or uses its output: $M_2(X, M_1(X))$.

The privacy bound can be further improved when the dataset domain is partitioned into disjoint sets. If different privacy mechanisms are run on these subsets of data, *parallel composition* ensures the privacy guarantee of their sequence is limited by the weakest of each mechanism. The following theorem was proven for ε-differential privacy by [94].

Theorem 2.1.2 (Parallel composition)**.** *Let $\mathcal{X}_j$ be arbitrary disjoint subsets of the input dataset domain $\mathcal{X}^m$ such that $\bigcup_{j=1}^{k} \mathcal{X}_j^m = \mathcal{X}^m$. If each privacy mechanism $M_j(X_j)$ satisfies ε_j-differential privacy for $j = 1, \ldots, k$ with $X_j = X \cap \mathcal{X}_j^m$, then the sequence $M(X) = (M_1(X_1), \ldots, M_k(X_k))$ satisfies $(\max_j \varepsilon_j)$-differential privacy.*

As the information of each individual is used only once in the sequence of mechanisms, the privacy budget does not depend on their total number. The collective upper bound is thus given by the maximum budget value which separately covers all the subsets of data.

Example 2.1.4 (Histogram)**.** This property finds an application when releasing a differentially private histogram. Suppose a number of k functions $f_j(x) = \sum_{i=1}^{m} \mathbf{1}_{\{x_i \in \mathcal{X}_j\}}$ count how many entries belong to each bin $\mathcal{X}_j$. Considering these functions are counting queries with equal ℓ_1-sensitivities $\Delta_1 f_j = 1$, we can use the Laplace mechanism to obtain the sanitized bin counts $Z_j = f_j(X) + Y_j$, where $Y_j \overset{\text{i.i.d.}}{\sim} \text{Lap}(1/\varepsilon)$. Since each of them provides ε-differential privacy and every dataset entry X_i necessarily fall into exactly one bin, parallel composition guarantees the sanitized histogram $Z = (Z_1, \ldots, Z_k)$ is also ε-differentially private.

Another valuable aspect of differential privacy is its property to remain unaffected by any *post-processing* of the sanitized output [44, Proposition 2.1]. As long as no additional knowledge about the private data is used, manipulating the results of a privacy mechanism cannot degrade the privacy guarantee.

Proposition 2.1.3 (Post-processing)**.** *Let $g : \mathcal{Z} \to \mathcal{Z}'$ be an arbitrary randomized function. If the privacy mechanism $M(X)$ satisfies ε-differential privacy, then the composition $g(M(X))$ also satisfies ε-differential privacy.*

This immunity to post-processing is particularly useful as it permits the application of denoising methods to the sanitized output with no loss of privacy. In this manner, the statistical utility lost in the sanitization process can be partially recovered by reducing the introduced noise.

In addition, differential privacy also features statistical properties. A few articles have studied statistical inference problems under differential privacy constraints. Research on this topic comprises [137, 117, 118, 63] and more recently [21].

The trade-off between statistical accuracy and privacy mentioned above is investigated in [21] for mean estimation and linear regression. In [63], methods to sanitize functional data are developed and kernel density estimation is discussed as the main example. The authors of [117] show that in some parametric problems, it is possible to build a differentially private estimator whose distribution converges to that of the maximum likelihood estimator. Specifically, this sanitized estimator is efficient and asymptotically unbiased. They show afterward in [118] that different point estimators can satisfy differential privacy with asymptotically optimal parametric rates of convergence.

In [137], the authors present differential privacy through the lens of statistics. In particular, they compare different privacy mechanisms by computing the rate of convergence of distributions and densities based on the sanitized data. This article also initiated nonparametric density estimation under differentially privacy constraints. Notably, it studies the differentially private estimation of Lipschitz-continuous density functions through histograms perturbed with Laplace noise. Furthermore, the authors provide an hypothesis testing interpretation of ε-differential privacy with the following result [137, Theorem 2.4].

Theorem 2.1.4. *Suppose the sanitized data Z, distribution of the private data X and ε-differentially private mechanism M are known. If $d(X, X') = 1$, any test of significance level γ of $H_0 : Z \sim Q(\cdot|X)$ against $H_1 : Z \sim Q(\cdot|X')$ has power bounded above by γe^{ε}.*

This means that given the full knowledge of every entry of the private dataset X except one X_i, how these data were sanitized through the privacy mechanism M and the sanitized output Z, it is virtually impossible to determine whether X_i belongs to a particular individual because the power of such a test is nearly equal to its level.

2.1.2 Approximate differential privacy

We introduce in this subsection the broader definition of (ε, δ)-differential privacy in which we are interested in this dissertation, as well as some of its strengths and weaknesses.

We then present two privacy mechanisms that satisfy it: the classical Gaussian mechanism and the analytic Gaussian mechanism which requires less noise for the same privacy guarantee.

Definition

Different relaxations of differential privacy have been proposed since the introduction of ε-differential privacy. The most common and popular one, named (ε, δ)-differential privacy [47], simply introduces an additive parameter δ in the inequality of Definition 2.1.1 to consider more privacy mechanisms that cannot meet pure differential privacy.

Definition 2.1.4 ((ε, δ)-differential privacy)**.** Let $\varepsilon, \delta \geq 0$. The privacy mechanism M is said to satisfy (ε, δ)-differential privacy if

$$Q(B|x) \leq e^{\varepsilon} Q(B|x') + \delta, \quad \forall B \in \mathcal{B}, \forall x, x' \in \mathcal{X}^m : \mathrm{d}(x, x') = 1.$$

The inclusion of this new privacy parameter permits a failure of the ε-differential privacy promise with probability δ. Intuitively, a value $\delta \geq 1$ voids the guarantee for any privacy budget ε while $\delta = 0$ yields the standard definition of ε-differential privacy. For $\delta \in (0, 1)$, a (ε, δ)-differentially private mechanism has a probability $1 - \delta$ to satisfy ε-differential privacy and a remaining probability δ to provide a weaker to no guarantee at all. For instance, a mechanism that randomly releases one dataset entry is $(0, 1/m)$-differentially private, where m is the number of individuals. To guard against any chance of private information disclosure, the privacy parameter is required to be very small. The authors of [93] discussed meaningful values for the δ parameter and showed that $\delta \ll 1/m$ is a necessary condition.

This *approximate* differential privacy shares similar properties with pure differential privacy such as sequential composition [44, Theorem 3.16] where the privacy parameters δ_j accumulate additively in the same way as the privacy budgets ε_j.

Theorem 2.1.5 (Sequential composition)**.** *If each privacy mechanism $M_j(X)$, chosen sequentially or adaptively, satisfies $(\varepsilon_j, \delta_j)$-differential privacy for $j = 1, \ldots, k$, then the sequence $M(X) = (M_1(X), \ldots, M_k(X))$ satisfies $(\sum_{j=1}^{k} \varepsilon_j, \sum_{j=1}^{k} \delta_j)$-differential privacy.*

Along with parallel composition and post-processing, approximate differential privacy also benefits from an advanced composition theorem [45, Theorem III.3] which gives a lower bound on the privacy budget ε for a slightly larger value of δ. It applies for sequences

of mechanisms $M_1, \ldots, M_k$ obtained from k-fold adaptive composition: each mechanism M_j adapts to the outputs of previous mechanisms $M_1, \ldots, M_{i-1}$ and the input to each mechanism is composed of the private dataset X and the outputs of previous mechanisms.

Theorem 2.1.6 (Advanced composition)**.** *If each privacy mechanism $M_j(X)$ satisfies (ε, δ)-differential privacy for $j = 1, \ldots, k$, then for all $\delta' > 0$, their k-fold adaptive composition satisfies $(\varepsilon', k\delta + \delta')$-differential privacy, where $\varepsilon' = \sqrt{2k \ln(1/\delta')}\varepsilon + k\varepsilon(e^{\varepsilon} - 1)$.*

Gaussian mechanism

The initial motivation for the (ε, δ)-differential privacy relaxation was to use noise drawn from a Gaussian distribution to sanitize a query output $f(X)$. Among the benefits of such a privacy mechanism, we can note the faster decay of the Gaussian distribution tails compared to those of the Laplace distribution. In addition, the inherent error in the dataset attributes or resulting from the query (ex: average) may itself be Gaussian.

Definition 2.1.5 (Gaussian mechanism)**.** Given any function $f : \mathcal{X}^m \to \mathbb{R}^k$, the Gaussian mechanism is defined as:

$$Z = f(X) + Y,$$

where $Y = (Y_1, \ldots, Y_k)$ and $Y_j \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$.

The probability density function of Z given $X = x$, for all $z \in \mathbb{R}^k$ reads:

$$q(z|x) = \prod_{j=1}^{k} \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(z_j - f(x)_j)^2}{2\sigma^2}\right\}.$$

Contrary to the Laplace mechanism, no distance appears in this expression and thus the triangle inequality cannot be used to find an upper bound for the ratio $q(z|x)/q(z|x')$. For this reason, the Gaussian mechanism does not meet ε-differential privacy for any privacy budget ε but instead satisfies (ε, δ)-differential privacy. To fit the definition of approximate differential privacy, the standard deviation σ of the added noise is calibrated to the ℓ_2-sensitivity of the function f in place of the ℓ_1-sensitivity.

Definition 2.1.6 (ℓ_2-sensitivity)**.** The ℓ_2-sensitivity of a function $f : \mathcal{X}^m \to \mathbb{R}^k$ is

$$\Delta_2 f = \max_{\substack{x, x' \in \mathcal{X}^m \\ \mathrm{d}(x, x') = 1}} \|f(x) - f(x')\|_2.$$

Given that the ℓ_2 distance is smaller than the ℓ_1 distance for $k \geq 2$, the corresponding ℓ_2-sensitivity has a lower value for the same function f. The original (or *classical*) Gaussian mechanism [47] uses a Gaussian tail approximation to obtain a bound for the standard deviation:

Theorem 2.1.7 (Classical Gaussian mechanism)**.** *For any $\varepsilon, \delta \in (0,1)$, the Gaussian mechanism satisfies (ε, δ)-differential privacy if $\sigma \geq \Delta_2 f \sqrt{2\ln(1.25/\delta)}/\varepsilon$.*

Note that this formula only holds for values of ε smaller than 1, which are generally considered as decent privacy guarantees. In [7], the authors show that the rate $\sigma = \Theta(1/\varepsilon)$ cannot be extended beyond the interval $(0,1)$ to the low privacy regime. They also prove that this value of σ is not optimal in the high privacy regime and can be further improved to reduce the amount of noise needed to achieve the same degree of privacy. Their approach uses numerical evaluations of the Gaussian cumulative distribution function $\Phi(x) = \mathbb{P}(\mathcal{N}(0,1) \leq x)$ to determine an optimal variance and define an *analytic* Gaussian mechanism.

Theorem 2.1.8 (Analytic Gaussian mechanism)**.** *For any $\varepsilon \geq 0$ and $\delta \in [0,1]$, the Gaussian mechanism satisfies (ε, δ)-differential privacy if and only if*

$$\Phi\left(\frac{\Delta_2 f}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_2 f}\right) - e^{\varepsilon}\Phi\left(-\frac{\Delta_2 f}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_2 f}\right) \leq \delta.$$

Privacy profiles

The guarantee provided by approximate differential privacy depends on the values of the privacy parameters ε and δ whose choice is nontrivial. In the case of pure differential privacy ($\delta = 0$), selecting an appropriate value for ε is possible given a practical privacy standard and the knowledge of the dataset domain along with the query function [79]. Furthermore, an optimal choice for δ can be determined for each value of ε with the *privacy profile* of a mechanism that we now present.

As it can be seen in theorems 2.1.7 and 2.1.8, the required inequalities for approximate differential privacy to be met both establish a relationship between the privacy parameters ε and δ. This shows that it is possible to decrease one and accordingly increase the other while maintaining a differential privacy guarantee for the Gaussian mechanism. The advanced composition theorem presented above indicates that a similar trade-off can be obtained for every (ε, δ)-differentially private mechanisms. Consequently, a single

mechanism actually satisfies a curve of (ε, δ)-differential privacy guarantees that contains an infinity of valid privacy parameter settings.

This more complete characterization of a mechanism privacy properties is found in the privacy profiles proposed by the authors of [6]. Their method captures the entire set of privacy guarantees satisfied by a given mechanism M using a function $\delta_M(\varepsilon)$ defined as follows:

Definition 2.1.7 (Privacy profile)**.** The privacy profile of a mechanism M is the function $\delta_M : [0, \infty) \to [0, 1]$ given by

$$\delta_M(\varepsilon) = \sup_{\substack{x,x' \in \mathcal{X}^m \\ \mathrm{d}(x,x')=1}} \sup_{B \in \mathcal{B}} (Q(B|x) - e^{\varepsilon} Q(B|x')).$$

For each privacy budget $\varepsilon \geq 0$, the privacy profile δ_M returns the smallest privacy parameter $\delta \in [0, 1]$ that can be achieved under the definition of approximate differential privacy. This function thus defines a curve in the privacy parameter space $[0, \infty) \times [0, 1]$ that separates the values (ε, δ) above it for which differential privacy is satisfied by the mechanism M and the ones below it where no guarantee can be given.

Mechanisms that satisfy pure differential privacy also have a privacy profile as they are (ε, δ)-differentially private with $\delta = 0$ for some value of ε. This is illustrated in Figure 2.2 from [6] where the privacy profiles of the randomized response, Laplace and analytic Gaussian mechanisms are represented for some parameter setting that gives the same intercept $\delta_M(0)$ for each one.

Note that the privacy profile of the randomized response and Laplace mechanisms both reach a plateau $\delta = 0$ when the privacy budget meets the required value for pure differential privacy to be satisfied, respectively $\varepsilon \geq \ln \frac{\pi}{1-\pi}$ and $\varepsilon \geq \frac{\Delta_1 f}{b}$. For smaller privacy budgets, a larger value of δ given by the privacy profile is necessary to preserve the differential privacy guarantee.

2.1.3 Other relaxations of differential privacy

We now present two different relaxations of differential privacy which are Rényi differential privacy [95] and f-differential privacy [36]. These are based on alternative ways to measure the similarity between the two distributions $Q(\cdot|X)$ and $Q(\cdot|X')$.

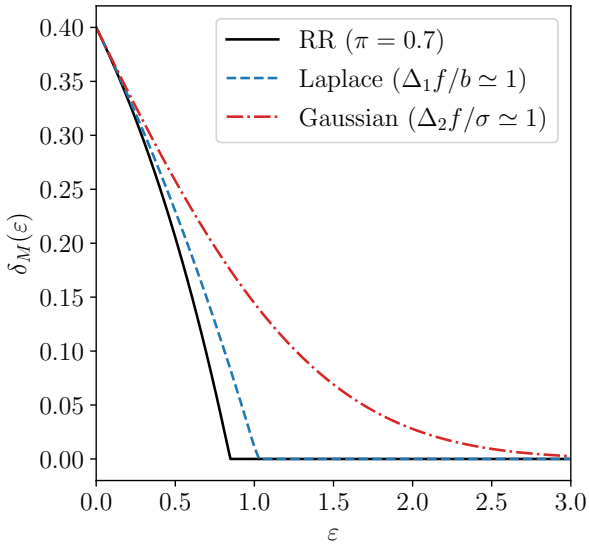


Figure 2.2 – Privacy profiles of the randomized response (RR), Laplace and Gaussian mechanisms. Parameters are chosen such that $\delta_M(\varepsilon) = 0.4$ at $\varepsilon = 0$.

Rényi differential privacy

Another relaxed definition of pure differential privacy proposed by [95] is *Rényi differential privacy*. In comparison to approximate differential privacy, it is a stronger privacy definition based on the Rényi divergence [109] to measure the similarity between $Q(\cdot|X)$ and $Q(\cdot|X')$. Let P_1 and P_2 be two distributions on a measurable space $(\mathcal{Z}, \mathcal{B})$ with respective densities p_1 and p_2 dominated by a common measure μ. The Rényi divergence of orders $1 < \alpha < \infty$ and $\alpha = \infty$ of P_1 from P_2 are respectively defined as

$$D_\alpha(P_1 \parallel P_2) = \frac{1}{\alpha - 1} \ln \left\{ \int \left(\frac{p_1(z)}{p_2(z)} \right)^\alpha p_2(z) d\mu(z) \right\},$$

$$D_\infty(P_1 \parallel P_2) = \ln \left\{ \sup_{B \in \mathcal{B}} \frac{P_1(B)}{P_2(B)} \right\}.$$

We observe that the Rényi divergence with $\alpha = \infty$ is closely related to the definition of pure differential privacy. Indeed, a privacy mechanism M is ε-differentially private if and only if its distribution $Q(\cdot|X)$ satisfies

$$D_\infty(Q(\cdot|x) \parallel Q(\cdot|x')) \leq \varepsilon, \quad \forall x, x' \in \mathcal{X}^m : \mathrm{d}(x, x') = 1.$$

Rényi differential privacy relaxes this definition by using the Rényi divergence of order

$1 < \alpha < \infty$. Concretely, a privacy mechanism M is said to satisfy (α, ε)-Rényi differential privacy with $\alpha > 1$ if

$$D_\alpha(Q(\cdot|x) \parallel Q(\cdot|x')) \leq \varepsilon, \quad \forall x, x' \in \mathcal{X}^m : \mathrm{d}(x, x') = 1.$$

This relaxed definition of privacy benefits from common properties shared with pure differential privacy. In particular, it is preserved under adaptive sequential composition and is immune to post-processing. In addition, if a privacy mechanism provides (α, ε)-Rényi differential privacy, it also satisfies $(\varepsilon + \frac{\ln 1/\delta}{\alpha - 1}, \delta)$-differential privacy for any $\delta \in (0, 1)$. Finally, the Gaussian mechanism as defined above satisfies $(\alpha, \alpha/2\sigma^2)$-Rényi differential privacy if $\Delta_2 f = 1$.

f-differential privacy

Whereas Rényi differential privacy is a divergence based relaxation of differential privacy, *f-differential privacy* introduced by [36] is inspired by the hypothesis testing formulation of privacy. This statistical point of view considers the problem of distinguishing between the datasets X and X' from the perspective of an attacker:

$$H_0 : P = P_0 \quad \text{versus} \quad H_1 : P = P_1,$$

where P_0 and P_1 are the distributions of the privacy mechanisms $M(X)$ and $M(X')$, respectively. The result of the mechanism is taken as input for the rejection rule $\phi \in [0, 1]$ and its type I and type II errors are respectively given by

$$\alpha_\phi = \mathbb{E}_{P_0}[\phi], \quad \beta_\phi = 1 - \mathbb{E}_{P_1}[\phi].$$

The authors characterize the trade-off between these two errors through the function $T(P_0, P_1) : [0, 1] \to [0, 1]$ defined as

$$T(P_0, P_1)(\alpha) = \inf_\phi \{\beta_\phi : \alpha_\phi \leq \alpha\},$$

where the infimum is taken over all rejection rules. The greater this function is, the harder it is to distinguish between the distributions P_0 and P_1.

Let $f : [0, 1] \to [0, 1]$ be a trade-off function equal to $T(P_0, P_1)$ for some distributions

P_0 and P_1, the privacy mechanism M is said to satisfy f-differential privacy if

$$T(Q(\cdot|X), Q(\cdot|X')) \geq f,$$

for all datasets X and X' such that $\mathrm{d}(X, X') = 1$. This generalization of differential privacy thus states that testing $H_0 : M(X) \sim Q(\cdot|X)$ against $H_1 : M(X) \sim Q(\cdot|X')$ is at least as difficult as distinguishing P_0 from P_1 at any level of type I error.

Contrary to the differential privacy definitions parameterized by real values (ε, δ, α) presented above, this approach is parameterized by a function f and offers a complete characterization of privacy comparable to privacy profiles. Moreover, this relaxed definition of pure differential privacy is a generalization of approximate differential privacy as well. Lastly, f-differential privacy benefits from immunity to post-processing and a tight composition theorem.

The authors of [36] also define *μ-Gaussian differential privacy* as the particular case of f-differential privacy where f is the trade-off function of two Gaussian distributions $\mathcal{N}(0, 1)$ and $\mathcal{N}(\mu, 1)$ with $\mu \geq 0$. Besides being fully described by a single parameter, this specific definition has several attractive properties. Indeed, it can be proven that all hypothesis testing based notions of privacy converge to Gaussian differential privacy under composition in the limit. Finally, a Gaussian mechanism with properly scaled variance $\sigma^2 = (\Delta_2 f/\mu)^2$ provides μ-Gaussian differential privacy.

2.2 Differential privacy applications

2.2.1 Application in industry

The strong privacy guarantees provided by differential privacy have been the subject of intensive academic research but have also sparked interest within the information technology industry. Notably, it has been adopted by some of the most dominant companies including Apple [123], Microsoft [35], Google [48, 1] and Meta [139]. We present here a few real-world applications developed by these organizations.

Collecting and publishing data on user activity

As differential privacy relates to sensitive data, a first straightforward application is to use it to collect and analyze information that belongs to the users of a service. Specifically,

the local variant of differential privacy lends itself well to this practice. In *local differential privacy* [42], each individual directly sanitizes its own piece of private data without the intervention of a trusted party. The locally sanitized data is then centrally collected for statistical analysis. As a result, data owners in this framework do not have to share their sensitive information and the data collector only has access to the sanitized output.

Apple has implemented local differential privacy in algorithms to allow app developers to collect usage and typing history. Their system architecture is documented in a patent application [124] and associated white paper [123]. In the latter, the authors combine different technical advances like the use of the Fourier transform to spread out signal information and sketching techniques to reduce the massive domain dimensionality. They also provide an analysis of the trade-offs among privacy, utility, server computation and device bandwidth. These algorithms have been deployed on hundreds of millions of devices for various purposes such as identifying popular emojis, popular health data types and media playback preferences in Apple's web browser.

One of the main challenges facing the industry to deploy differential privacy systems lies in the management of the privacy budget, with a tension between a valid level of privacy and a reasonable amount of noise for concrete applications. This difficulty is highlighted by several research works [122, 55] showing the strong privacy limitations in the early deployments of Apple's differentially private algorithms.

In addition to contributing to the development of differential privacy [46, 47], Microsoft has worked on data collection over time from devices with privacy guarantees [35]. The latter degrade as the same dataset is queried several times as we have seen above with the sequential composition property. The authors propose locally differentially private mechanisms to perform mean and histogram estimation on repeatedly collected data. Their solution has been deployed across millions of devices.

For its part, Google has developed a system called Randomized Aggregatable Privacy-Preserving Ordinal Response (RAPPOR) [48] which aims to collect and analyze web browsing behavior with local differential privacy and utility guarantees. It combines the randomized response mechanism with Bloom filters to obtain a compact representation of sensitive data. RAPPOR has been deployed on Google's web browser to collect data from millions of consenting users and identify popular web destinations. Following research [49] explains how to extract these destination identities without prior knowledge of their web addresses.

The French telecommunications company Orange is interested in privacy-preserving

data publication. It has proposed a solution [12] which combines co-clustering with synthetic data generation to sanitize datasets under differential privacy. Its research also focuses on the protection of mobility data [52], for which simple pseudonymization is not always enough to prevent a privacy breach [41]. In particular, Orange has developed an open-source Android application [101] which implements *geo-indistinguishability* [4], an adaptation of differential privacy to location data that can be satisfied through the addition of planar Laplacian noise.

Training machine learning models

Beyond classical statistics, another popular application is the training of machine learning models with differential privacy guarantees. In this context, the sensitive information is the training data of which the model can be seen as a function. Previous work has shown it is possible to extract parts of the training data only by interacting with the model through inputs and outputs [53]. In situations where the training mechanism and model parameters are shared, privacy guarantees are even more essential. To address this problem, Google has developed [1] new algorithmic techniques to train machine learning models such as neural networks under differential privacy along with analysis tools to compute the corresponding privacy parameters.

Training machine learning models consists in the minimization of a loss function $\mathcal{L}$ by tuning the model parameters $\boldsymbol{\theta} \in \mathbb{R}^p$. In practice, this process is usually done by the stochastic gradient descent (SGD) algorithm which estimates the gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$ on a batch of random training examples. The approach described in [1] intends to limit the impact of the training data during the training process. The authors propose a differentially private version of the SGD inspired by [119] and outlined in Algorithm 1.

Compared to the classical SGD, this algorithm includes two additional instructions at each training step: gradient clipping and noise addition. The first one controls the influence of each individual training example on the computed gradient by ensuring its ℓ_2 norm is always smaller than a clipping threshold C. Then, Gaussian noise is added over the average gradient on the group of examples with variance depending on both C and a parameter σ. Essentially, this corresponds to a Gaussian mechanism which protects the gradient whose ℓ_2-sensitivity is bounded beforehand.

As this privacy mechanism is applied at each training step, a relevant problem is the

Algorithm 1: Differentially private SGD

Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_i \mathcal{L}(\boldsymbol{\theta}, x_i)$. Parameters: learning rate η, batch size L, noise scale σ, gradient norm bound C.

Initialize $\boldsymbol{\theta}_0$ randomly

for $t \in [T]$ **do**

  Take a random sample L_t with sampling probability L/N

  Compute gradient: $\forall i \in L_t$, $\boldsymbol{g}_t(x_i) \leftarrow \nabla_{\boldsymbol{\theta}_t}\mathcal{L}(\boldsymbol{\theta}_t, x_i)$

  Clip gradient: $\bar{\boldsymbol{g}}_t(x_i) \leftarrow \boldsymbol{g}_t(x_i)/\max(1, \frac{\|\boldsymbol{g}_t(x_i)\|_2}{C})$

  Add noise: $\tilde{\boldsymbol{g}}_t \leftarrow \frac{1}{L}(\sum_i \bar{\boldsymbol{g}}_t(x_i) + \mathcal{N}(\mathbf{0}, \sigma^2 C^2 \mathbf{I}_p))$

  Descent: $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta\tilde{\boldsymbol{g}}_t$

end

Output: $\boldsymbol{\theta}_T$

computation of the overall privacy parameters (ε, δ) once the model is trained. Direct bounds for these are obtained from the sequential and advanced composition theorems presented in Section 2.1.2. A more refined solution introduced by [1] is the *moments accountant* procedure that keeps count of the privacy parameters during the training process through the composability of differential privacy. This accounting method yields tighter bounds by proving that the differentially private SGD satisfies $(\mathcal{O}(\frac{L}{N}\varepsilon\sqrt{T}), \delta)$-differential privacy with respect to the complete set of examples $\{x_1, \ldots, x_N\}$ for appropriate choices of the noise scale and clipping threshold parameters. More precisely, the moments accountant keeps track of the higher-order moments of the privacy loss random variable. The latter is defined as $\ln\left\{\frac{Q(B|x)}{Q(B|x')}\right\}$ at $B \in \mathcal{B}$ for a privacy mechanism with distribution $Q(\cdot|X)$ and for all adjacent datasets $x, x' \in \mathcal{X}^m$. This random variable is closely related to the Rényi differential privacy definition [95] presented in Section 2.1.3. This relaxation of differential privacy is therefore well suited for the privacy parameter computations of the moments accountant. Since this method, other works on privacy amplification by iteration [50, 8] have been conducted to further improve the bounds on the privacy parameters for learning algorithms.

The differentially private SGD and moments accountant have been implemented in the **TensorFlow Privacy** [59] Python library. It is based on TensorFlow [2], the software library for machine learning developed by Google. A similar implementation of the differentially private SGD named **Opacus** [139] has been proposed by Meta for their machine learning framework PyTorch. This Python library provides a privacy accounting method based on Rényi differential privacy as well.

2.2.2 Application to graphs

Our work aims to protect sensitive information contained in a signal defined on the vertices of a graph. While it is different from directly protecting a graph with differential privacy, we wish to mention several ways to achieve this in order to better distinguish between these two approaches. Here, we briefly present some privacy mechanisms that sanitize queries on graphs or entire graphs. More details and examples can be found in the recent survey from [84].

Graph query sanitization

Differential privacy was originally introduced to offer privacy guarantees to sensitive information contained in tabular datasets. Its definition has been adapted to the graph data structure in which vertices represent individuals and edges represent relationships between them. Statistical analysis on graphs studies the structure and relationships between the vertices and edges that compose them. Similarly to tabular data, this information can be obtained through a query function f applied to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. Some of the commonly used statistics for graphs are vertex degrees and their distribution, centrality and distance metrics as well as subgraph counts.

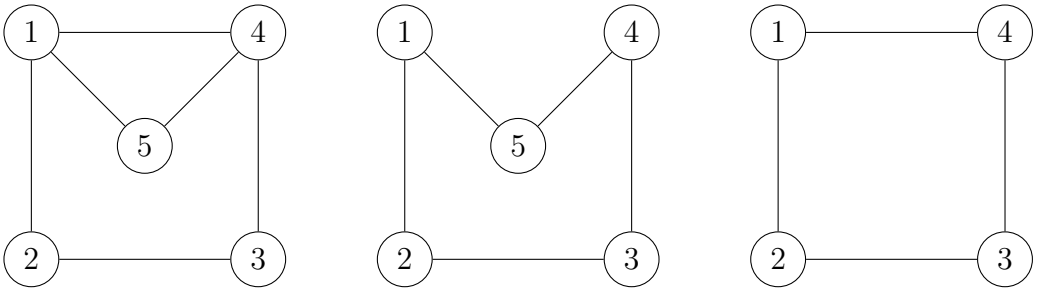
Mechanisms that aim to sanitize such graph queries usually compute the sensitive output $f(\mathcal{G})$ and then transform it to ensure privacy. A definition of differential privacy for graphs can be obtained by replacing datasets with graphs in Definition 2.1.4.

Definition 2.2.1 (Graph (ε, δ)-differential privacy)**.** Let $\varepsilon, \delta \geq 0$. The privacy mechanism M is said to satisfy (ε, δ)-differential privacy if, for all neighboring graphs $\mathcal{G}$ and $\mathcal{G}'$,

$$Q(B|\mathcal{G}) \leq e^{\varepsilon} Q(B|\mathcal{G}') + \delta, \quad \forall B \in \mathcal{B}.$$

In differential privacy for tabular data, the privacy mechanism distribution Q is conditioned on datasets x and x' that differ in one entry. Analogously, there are two common notions of *neighboring graphs* based on either their vertices or edges. Both use the symmetric difference between two sets given by $\mathcal{A} \triangle \mathcal{B} = (\mathcal{A} \cup \mathcal{B}) \setminus (\mathcal{A} \cap \mathcal{B})$.

Two graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ are *edge neighbors* if they share the same vertices and differ in one edge, that is, $\mathcal{V} = \mathcal{V}'$ and $|\mathcal{E} \triangle \mathcal{E}'| = 1$. This definition is illustrated by Figures 2.3a and 2.3b where the graphs only differ by the edge $\{1, 4\}$.



(a) $\mathcal{G}$ (b) $\mathcal{G}'$ (c) $\mathcal{G}''$

Figure 2.3 – Examples of neighboring graphs

Two graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ are *vertex neighbors* if they differ in one vertex and its associated edges, that is, $|\mathcal{V} \triangle \mathcal{V}'| = 1$ and $\mathcal{E} \triangle \mathcal{E}' = \{\{i, j\} \in \mathcal{E} \cup \mathcal{E}' \mid i \in \mathcal{V} \triangle \mathcal{V}'\}$. The graphs shown on Figures 2.3a and 2.3c satisfy this definition as they differ by the vertex 5 along with the edges $\{1, 5\}$ and $\{4, 5\}$.

These two notions of neighboring graphs produce two variants of graph differential privacy formalized by [67]: *edge-differential privacy* and *node-differential privacy*. The former protects a relationship between two individuals while the latter protects the presence of an individual and its relationships with others. Intuitively, it is easier to achieve edge-differential privacy than node-differential privacy in general.

The Laplace mechanism is one of the key components of edge-differentially private mechanisms. Identically to the tabular case, it adds noise to the graph query output with a scale computed from an appropriate sensitivity. For instance, the notion of smooth sensitivity is introduced by [112] to sanitize subgraph counting queries. These count how many times a certain subgraph, such as a triangle or a star, appears in a given graph. Smooth sensitivity has since been broadly used in subsequent research works to improve the utility of private mechanisms for both tabular and graph data.

Another common graph query is the degree sequence corresponding to a monotonic non-decreasing sequence of vertex degrees. This query benefits from a small ℓ_1-sensitivity as adding or removing an edge between two vertices only affects their respective degrees. An algorithm based on constraint inference has been proposed by [67] to improve the utility of edge-differentially private degree sequences.

As node-differential privacy depends on a different vertex and several different edges, the same graph queries tend to have a larger sensitivity than for edge-differential privacy and thus lose utility. As a result, there are less works within this variant of graph differential privacy. Two kinds of methods have been proposed to tackle this sensitivity problem:

top-down projections to graphs with lower degrees and bottom-up generalizations using Lipschitz extensions.

The top-down approach involves mapping the original graph to a new graph with a bounded maximum degree, and then answering queries on the new graph with noise that is proportional to the sensitivities of the query function and the projection map. The notion of restricted sensitivity was introduced by [14] to produce node-differentially private subgraph counts with this approach. A privacy mechanism which sanitizes a graph degree distribution with a projection eliminating all vertices whose degrees exceed a chosen threshold is described in [76].

The bottom-up approach initially answers queries on graphs with bounded maximum degrees and then extends the answers to arbitrary graphs using Lipschitz extensions. Next, noise is added to the extended answers with classical privacy mechanisms like the Laplace mechanism. This approach is employed in [76] and [108] to sanitize subgraph counts and degree sequences, respectively.

Graph sanitization

In addition to sanitizing graph queries, another common method to protect sensitive information in graph data is by creating synthetic graphs that resemble the original ones. One of the primary benefits of this technique is its independence from graph queries, making it suitable for answering any future query with minimal or no risk of privacy loss. We now mention some approaches proposed in the literature to sanitize graphs under differential privacy.

A first approach to synthesize a private graph is to build a generative model whose parameters are estimated in a manner that is differentially private. The Pygmalion model [110] follows this methodology by using a dK-graph model [90] to count the number of connected subgraphs of size k with varying degree combinations as dK-series. Then, the latter are sorted and partitioned into disjoint unions of close sub-series sanitized with the Laplace mechanism. After applying the constraint inference-based method from [67] to reduce noise, a synthetic graph is created from the series.

An alternative approach to generative graph models consists in approximating the original weight and Laplacian matrices through perturbation strategies. A first example introduced in [134] adds Laplacian noise to the largest k eigenvalues and eigenvectors of the weight matrix before turning them into a lower rank matrix. Another example of this approach is presented in [19] where weight matrices are sanitized for weighted directed

graphs. The Laplace mechanism is applied to the matrices depending on the sensitivity within blocks of vertices.

Graph signal sanitization

In the case where a graph signal $\boldsymbol{f} \in \mathbb{R}^n$ corresponds to the output of a query function $f : \mathcal{X}^m \to \mathbb{R}^n$ applied to a dataset x, sanitization can be achieved through classical privacy mechanisms for tabular data. For instance, if each individual of a dataset can be associated with a vertex of a graph, a query that counts the number of individuals for each vertex can be sanitized with simple Gaussian noise [7]. This is the chosen approach presented in Chapter 3.

Another interesting example of differential privacy for graph signals is found in the sanitized training of neural networks. First, a graph structure can be derived from the underlying network architecture. Then, a signal on this graph is given by the network parameter values and updated multiple times by querying batches of private data through the training mechanism. In this setting, differentially private SGD discussed in Section 2.2.1 corresponds to graph signal sanitization using the classical Gaussian mechanism. Although not developed here, this example seems to be both a natural ground for application of our denoising methodology and a promising direction for future research.

Chapter 3

Large graph signal denoising with application to differential privacy

Over the last decade, signal processing on graphs has become a very active area of research. Specifically, the number of applications, for instance in machine or deep learning, using frames built from graphs, such as wavelets on graphs, has increased significantly. We consider in particular the case of signal denoising on graphs via a data-driven wavelet tight frame methodology. This adaptive approach is based on a threshold calibrated using Stein's unbiased risk estimate adapted to a tight frame representation. We make it scalable to large graphs using Chebyshev-Jackson polynomial approximation, which allow fast computation of the wavelet coefficients, without the need to compute the Laplacian eigendecomposition. However, the overcomplete nature of the tight frame, transforms a white noise into a correlated one. As a result, the covariance of the transformed noise appears in the divergence term of the SURE, thus requiring the computation and storage of the frame, which leads to an impractical calculation for large graphs. To estimate such covariance, we develop and analyze a Monte Carlo strategy, based on the fast transformation of zero mean and unit variance random variables. This new data-driven denoising methodology finds a natural application in differential privacy. A comprehensive performance analysis is carried out on graphs of varying size, from real and simulated data.

3.1 Introduction

Data acquired from large-scale interactive systems, such as computer, ecological, social, financial or biological networks, become increasingly widespread and accessible. In modern machine learning, the effective representation, processing or analysis of these large-scale structured data with graphs or networks are some of the key issues [99, 16]. The emerging field of Graph Signal Processing (GSP) highlights connections between signal processing and spectral graph theory [115, 103], while building bridges to address

these challenges. Indeed, GSP has led to numerous applications in the field of machine learning: convolutional neural networks (CNN) on graphs [18], [68, 33], semi-supervised classification with graph CNN [77, 64] or community detection [127] to name just a few. We refer the reader to [38] for a recent review providing new perspectives on GSP for machine learning including, for instance, the important role it played in some of the early designs of graph neural networks (GNNs) architectures. Moreover, the recent study in [54] shows that popular GNNs designed from a spectral perspective, such as spectral graph convolutional networks or graph attention networks, are implicitly solving graph signal denoising problems.

In the past decades, sparse approximation with respect to a frame played a fundamental role in many areas such as signal compression and restoration, data analysis, and GSP in general. Indeed, overcomplete representations like wavelet frames have several advantages and offer more flexibility over orthonormal bases. One representative family of overcomplete systems derived form the orthonormal *Diffusion Wavelets* of [27] is the so-called *Spectral Graph Wavelet Transform* (SGWT) of [65] constructed from a general wavelet frame. In a denoising context, SGWT has recently been adapted by [58] to form a tight frame using the Littlewood-Paley decomposition inspired by [28]. Based on SGWT, [86] proposed an automatic calibration of the threshold parameter by adapting Stein's unbiased risk estimate (SURE) for a noisy signal defined on a graph and decomposed in a given wavelet tight frame. Even if this selection criterion produces efficient estimators of the unknown mean squared error (MSE), the main limitation is the need for a complete eigendecomposition of the Laplacian matrix, making it intractable for large-scale graphs.

We propose here to extend this methodology to large sparse graphs by avoiding this eigendecomposition, thus extending its range of application. Different strategies have been proposed in the context of GSP, one of the most popular is based on Chebyshev polynomial approximation [65]. However, even if Chebyshev expansions are a good choice in many scenarios, approximations of discontinuous or non-periodic functions suffer from the *Gibbs phenomenon*. A simple strategy commonly used in GSP [114] to reduce possible spurious oscillations without additional computational cost is the introduction of Jackson's *damping coefficients* [73, 34] which allows for higher orders of approximation.

As SURE can be evaluated in the wavelet domain, its calculation benefits directly from these efficient numerical approximations. In order to make it suitable for large sparse graphs, the only problematic step is the computation of the weights appearing in its expression. Indeed, since the SGWT is no longer orthogonal a white Gaussian noise in

the graph domain is transformed into a correlated one thus involving the covariance of the transformed noise in the resulting SURE divergence term. The latter requires the explicit computation and storage of the frame in order to be calculated. Inspired by the estimation of the correlation between wavelets centered at different vertices proposed in [127], our contribution is to take advantage of the interpretation of the SURE weights as the covariance between wavelet transforms of random signals in order to estimate them with Monte Carlo approximation. We then plug this weight estimator in the SURE formula to obtain an estimator of SURE that extends well to large graph signals. In addition, we provide expressions for the variance of our proposed estimators and show that drawing Monte Carlo samples from the centered Rademacher distribution gives a smaller variance compared to the standard Gaussian distribution. Our approach is in line with other methods [107, 138] that also use Monte Carlo strategy, but to estimate the entire divergence term involved in the calculation of SURE, in the case of uncorrelated noise.

Our proposed method can remove noise from any signal defined on a graph, this includes images [115] and 3D meshes [100] which can have a large number of vertices. Here, we focus on an interesting application in differential privacy [46] whose purpose is to protect sensitive data used by algorithms. Such privacy guarantees are usually achieved by adding white noise to the signal which inevitably reduces its statistical utility as the relevant information it contains is perturbed. This utility can be partially recovered through denoising on the condition that no information about the original signal is used. As our proposed data-driven methodology lends itself well to this usage for graph signals, we incorporate it in our numerical experiments. These give an evaluation of our Monte Carlo estimator of SURE and its weights, along with the overall denoising methodology on both small and large graphs. In summary, the contributions of this paper are as follows:

— We propose a Monte Carlo estimation of Stein's unbiased risk estimate (SURE) that extends to signals defined on large-scale graphs. This method avoids the computationally expensive eigendecomposition of the graph Laplacian matrix required to compute weights that appear in the SURE expression.
— Provided expressions for the variance of our estimators show that Monte Carlo samples drawn from a Rademacher distribution is more efficient than with a Gaussian distribution. This theoretical result is illustrated through numerical experiments that compare both distributions.
— A performance analysis of the proposed graph signal denoising methodology shows

its performance on real data protected with differential privacy and simulated large graph signals.

The paper is structured as follows. We introduce our notation of graph signals and briefly recall the SGWT definition of [65], its construction by [58] and polynomial approximation in Section 3.2. Our proposed Monte Carlo estimators of SURE and its weights along with their respective variance are presented in Section 3.3. In Section 3.4 we present the notion of differential privacy from [46] and two methods to achieve it in the context of graph signals. Finally, we numerically evaluate our estimators and compare our methodology to the DFS fused lasso introduced in [69] for small and large graphs in Section 3.5.

3.2 Graph signal denoising

Consider a signal $\boldsymbol{f} \in \mathbb{R}^{\mathcal{V}}$ defined on an undirected weighted graph $\mathcal{G}$, with set of vertices $\mathcal{V}$ of cardinality n, and weighted adjacency matrix $\mathbf{W}$ with entries $(w_{ij})_{i,j\in\mathcal{V}}$. The (unnormalized) graph Laplacian matrix $\mathcal{L} \in \mathbb{R}^{\mathcal{V}\times\mathcal{V}}$ associated with $\mathcal{G}$ is the symmetric matrix defined as $\mathcal{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is the diagonal matrix with diagonal coefficients $D_{ii} = \sum_{j\in\mathcal{V}} w_{ij}$. We present here our methodology with this particular Laplacian matrix but it can be easily adapted to its normalized and random walk counterparts like presented below in Section 3.2.3.

The noise corruption model can be written as

$$\tilde{\boldsymbol{f}} = \boldsymbol{f} + \boldsymbol{\xi},$$

where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_n)$. The purpose of denoising is to build an estimator $\hat{\boldsymbol{f}}$ of $\boldsymbol{f}$ that depends only on $\tilde{\boldsymbol{f}}$.

A simple way to construct an effective non-linear estimator is obtained by thresholding the SGWT coefficients of $\boldsymbol{f}$ on a frame (see [65] for details about the SGWT). Given the Laplacian and a given frame, denoising in this framework can be summarized as follows:

1. Analysis: Compute the SGWT transform $\mathcal{W}\tilde{\boldsymbol{f}}$
2. Thresholding: Apply a given thresholding operator (*e.g., soft* or *hard*) to the coefficients $\mathcal{W}\tilde{\boldsymbol{f}}$
3. Synthesis: Compute the inverse SGWT transform to obtain an estimate $\hat{\boldsymbol{f}}$ of the original signal

This procedure can be viewed as an extension of the wavelet denoising methodology from Donoho and Johnstone [39] to the SGWT.

3.2.1 Spectral graph wavelet transform

The SGWT decomposes a signal into a frame $\mathfrak{F} = \{\boldsymbol{r}_i\}_{i\in I}$ of vectors of $\mathbb{R}^{\mathcal{V}}$ with frame bounds $A, B \in [0, \infty)$ satisfying for all $\boldsymbol{f} \in \mathbb{R}^{\mathcal{V}}$

$$A\|\boldsymbol{f}\|_2^2 \leq \sum_{i\in I} |\langle \boldsymbol{f}, \boldsymbol{r}_i\rangle|^2 \leq B\|\boldsymbol{f}\|_2^2.$$

When $A = B = 1$, the above inequality becomes Parseval's identity and such a frame is said to be *tight*.

As $\mathcal{L}$ is a symmetric matrix, its spectral decomposition is given by $\mathcal{L} = \sum_{\ell=1}^n \lambda_\ell \langle \boldsymbol{v}_\ell, \cdot\rangle \boldsymbol{v}_\ell$, where $0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_n$ are the (ordered) eigenvalues of $\mathcal{L}$ and $(\boldsymbol{v}_\ell)_{1\leq\ell\leq n}$ are the associated eigenvectors. Then for any function $\rho : \mathrm{sp}(\mathcal{L}) \to \mathbb{R}$ defined on the spectrum of $\mathcal{L}$, we have the functional calculus formula $\rho(\mathcal{L}) = \sum_{\ell=1}^n \rho(\lambda_\ell)\langle \boldsymbol{v}_\ell, \cdot\rangle \boldsymbol{v}_\ell$.

We build a tight frame following [80, 58] with a finite partition of unity $(\phi_j)_{j=0,\ldots,J}$ on the compact $[0, \lambda_n]$ defined as follows: let $\omega : \mathbb{R}^+ \to [0, 1]$ be some continuous function with support in $[0, 1]$, satisfying $\omega \equiv 1$ on $[0, b^{-1}]$, for some $b > 1$, and set

$$\phi_0(x) = \omega(x), \quad \phi_j(x) = \omega(b^{-j}x) - \omega(b^{-j+1}x),$$

for $j = 1, \ldots, J$, where $J = \lfloor \ln \lambda_n / \ln b \rfloor + 2$. In our numerical experiments, we use the following piecewise linear function ω:

$$\omega(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq b^{-1} \\ \frac{b}{1-b}x + \frac{b}{b-1} & \text{if } b^{-1} < x \leq 1 \\ 0 & \text{if } x > 1. \end{cases}$$

with parameter $b = 2$. An alternative ω function is the $\mathcal{C}^\infty$ function $h_c(x) = g_c(x+1)g_c(1-x)$ where $g_c(x) = f(x)/(f(x)+f(c-x))$ and $f(x) = e^{-1/x}\mathbf{1}_{\{x>0\}}$. Another choice is to take a $\mathcal{C}^3$ piecewise polynomial plateau function like the authors of [58]. Figure 3.1 illustrates the ω and h_c functions with parameters $b = 2$ and $c = 1$, respectively.

The partition of unity $(\phi_j)_{j=0,\ldots,4}$ on $[0, \lambda_n]$ obtained with the graph presented in the first experiment from Section 3.5 is shown in Figure 3.2.

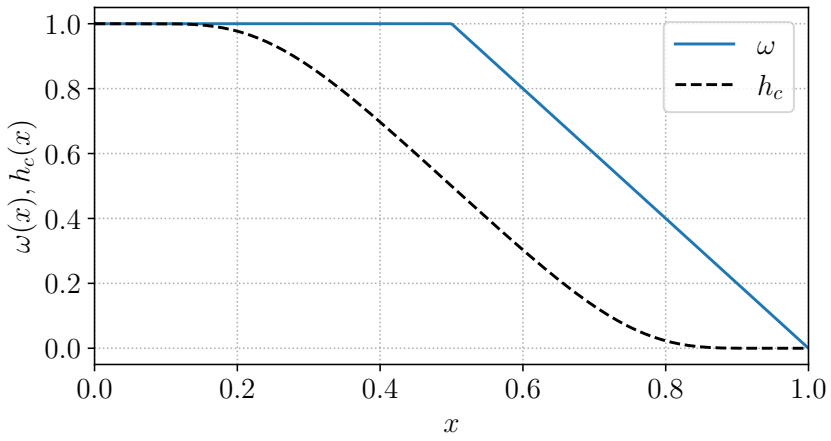


Figure 3.1 – ω and h_c functions on $[0, 1]$

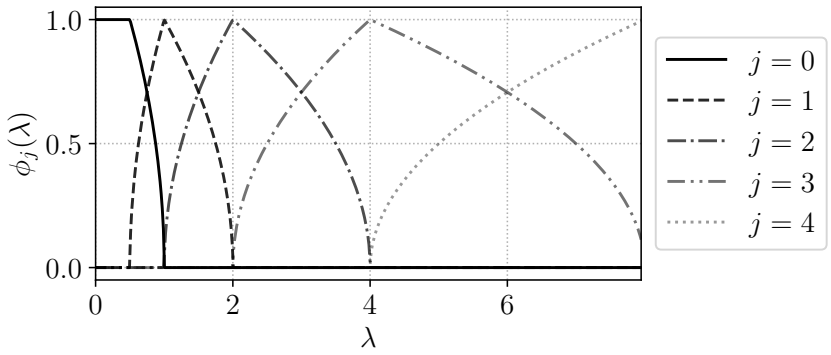


Figure 3.2 – Finite partition of unity on $[0, \lambda_n]$

Using Parseval's identity, we can show that the following set is a tight frame:

$$\mathfrak{F} = \left\{ \sqrt{\phi_j}(\mathcal{L})\boldsymbol{\delta}_i, j = 0, \dots, J, i \in \mathcal{V} \right\}.$$

Decomposing a signal $\boldsymbol{f} \in \mathbb{R}^{\mathcal{V}}$ into this frame results in its SGWT along the $(J+1)$ scales:

$$\mathcal{W}\boldsymbol{f} = \left(\sqrt{\phi_0}(\mathcal{L})\boldsymbol{f}^\top, \dots, \sqrt{\phi_J}(\mathcal{L})\boldsymbol{f}^\top \right)^\top \in \mathbb{R}^{n(J+1)}.$$

With the tightness property of the frame, the inverse transform is directly given by the application of the adjoint matrix to the wavelet coefficients:

$$\mathcal{W}^* \left(\boldsymbol{\eta}_0^\top, \boldsymbol{\eta}_1^\top, \dots, \boldsymbol{\eta}_J^\top \right)^\top = \sum_{j \geq 0} \sqrt{\phi_j}(\mathcal{L})\boldsymbol{\eta}_j.$$

3.2.2 SGWT polynomial approximation

Direct computation of the SGWT entails functional calculus on the graph Laplacian matrix $\mathcal{L}$ and thus the computation of its eigenvectors and eigenvalues. This limits applications to reasonably sized graphs that have less than a few thousand vertices. For larger ones, the computationally expensive eigendecomposition can be avoided through a fast transform based on Chebyshev polynomial approximation [65].

The Chebyshev polynomials of the first kind $T_k(x)$ are obtained from the recurrence relation $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0(x) = 1$ and $T_1(x) = x$. They form an orthogonal basis of the Hilbert space $L^2([-1,1], dx/\sqrt{1-x^2})$. Any filter ρ can be approximated with the truncated Chebyshev expansion of degree K

$$\rho_K(\mathcal{L}) = \sum_{k=0}^{K} \theta_k(\tilde{\rho}) T_k(\widetilde{\mathcal{L}}),$$

where $\theta_k(\tilde{\rho})$ is the k^{th} coefficient of the Chebyshev expansion of function $\tilde{\rho}(x) = \rho(\frac{\lambda_n}{2}(x+1))$ and $T_k(\widetilde{\mathcal{L}})$ is the k-degree Chebyshev polynomial computed for $\widetilde{\mathcal{L}} = \frac{2}{\lambda_n}\mathcal{L} - \mathbf{I}_n$. This transformation of $\mathcal{L}$ extends the expansion to any Laplacian matrix by mapping $[0, \lambda_n]$ into $[-1, 1]$. According to [65], for all filter ρ defined on $\mathrm{sp}(\mathcal{L})$ and all signal $\boldsymbol{f}$, the approximation $\rho_K(\mathcal{L})\boldsymbol{f}$ is close to $\rho(\mathcal{L})\boldsymbol{f}$.

While this first approximation is more practical than the direct SGWT, it is subjected to the *Gibbs phenomenon*. A solution is to include Jackson coefficients g_k^K as damping multipliers in the Chebyshev expansion:

$$\rho_K(\mathcal{L}) = \sum_{k=0}^{K} g_k^K \theta_k(\tilde{\rho}) T_k(\widetilde{\mathcal{L}}).$$

An expression of these damping factors can be found in [73], a shorter form proposed in [34] is given by

$$g_k^K = \frac{\sin(k+1)\alpha_K}{(K+2)\sin(\alpha_K)} + \left(1 - \frac{k+1}{K+2}\right)\cos(k\alpha_K),$$

where $\alpha_K = \pi/(K+2)$. This Chebyshev-Jackson polynomial approximation reduces Gibbs oscillations resulting in a better convergence as the degree K increases.

3.2.3 Extension to other Laplacian matrices

As previously mentioned, this methodology also adapts well to the normalized and random walk (or asymmetric) Laplacian matrices, respectively defined as $\mathcal{L}^{\text{norm}} = \mathbf{D}^{-\frac{1}{2}}\mathcal{L}\mathbf{D}^{-\frac{1}{2}}$ and $\mathcal{L}^{\text{rw}} = \mathbf{D}^{-1}\mathcal{L}$. These have been used as an alternative to the unnormalized graph Laplacian $\mathcal{L}$ in other related methods such as the graph Fourier transform [57].

The normalized Laplacian matrix is real symmetric like $\mathcal{L}$ which means it is diagonalizable and therefore suited for our approach. As its spectrum $\text{sp}(\mathcal{L}^{\text{norm}}) = \{\mu_1, \dots, \mu_n\}$ is always contained in the interval $[0, 2]$, its maximum eigenvalue μ_n is bounded by 2. This represents a special case of the construction described above and requires a few modifications. First, the formula $J = \lfloor \ln \mu_n / \ln b \rfloor + 2$ that determines the number of scales in the wavelet decomposition restricts the choice of parameter b to the interval $(1, 2]$ in order to get more than $J + 1 = 3$ scales. Then, the polynomial approximation consists of the truncated Chebyshev expansion of function $\tilde{\rho}(x) = \rho(x + 1)$ with the appropriate transformation $\widetilde{\mathcal{L}}^{\text{norm}} = \mathcal{L}^{\text{norm}} - \mathbf{I}_n$.

On the other hand, the random walk Laplacian matrix is not symmetric but is diagonalizable nonetheless as it is similar to the normalized Laplacian: $\mathcal{L}^{\text{norm}} = \mathbf{D}^{\frac{1}{2}}\mathcal{L}^{\text{rw}}\mathbf{D}^{-\frac{1}{2}}$. Its eigenvalues and eigenvectors are easily obtained from the eigendecomposition of $\mathcal{L}^{\text{norm}}$:

$$\begin{aligned} \mathcal{L}^{\text{norm}}\boldsymbol{u}_\ell &= \mu_\ell \boldsymbol{u}_\ell \\ \mathbf{D}^{\frac{1}{2}}\mathcal{L}^{\text{rw}}\mathbf{D}^{-\frac{1}{2}}\boldsymbol{u}_\ell &= \mu_\ell \boldsymbol{u}_\ell \\ \mathcal{L}^{\text{rw}}(\mathbf{D}^{-\frac{1}{2}}\boldsymbol{u}_\ell) &= \mu_\ell (\mathbf{D}^{-\frac{1}{2}}\boldsymbol{u}_\ell), \end{aligned}$$

where $\boldsymbol{u}_\ell$ is the eigenvector of $\mathcal{L}^{\text{norm}}$ associated with μ_ℓ. We see that $\mathcal{L}^{\text{rw}}$ has exactly the same spectrum as $\mathcal{L}^{\text{norm}}$ and its set of eigenvectors is given by $\{\mathbf{D}^{-\frac{1}{2}}\boldsymbol{u}_\ell\}_{\ell=1,\dots,n}$. Since these form an orthonormal basis for $\mathbb{R}^n$ with the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathbf{D}} = \boldsymbol{x}^\top \mathbf{D} \boldsymbol{y}$, we have the spectral decomposition $\mathcal{L}^{\text{rw}} = \sum_{\ell=1}^n \mu_\ell \langle \mathbf{D}^{-\frac{1}{2}}\boldsymbol{u}_\ell, \cdot \rangle_{\mathbf{D}} \mathbf{D}^{-\frac{1}{2}}\boldsymbol{u}_\ell$. The functional calculus formula for any function ρ defined on the spectrum of $\mathcal{L}^{\text{rw}}$ is thus $\rho(\mathcal{L}^{\text{rw}}) = \sum_{\ell=1}^n \rho(\mu_\ell) \langle \mathbf{D}^{-\frac{1}{2}}\boldsymbol{u}_\ell, \cdot \rangle_{\mathbf{D}} \mathbf{D}^{-\frac{1}{2}}\boldsymbol{u}_\ell$. Chebyshev polynomial approximation can then be applied in the same way as for the normalized Laplacian matrix.

3.3 SURE weights Monte Carlo estimation

From [86], SURE for a general thresholding process $h : \mathbb{R}^{n(J+1)} \to \mathbb{R}^{n(J+1)}$ is given by the following identity

$$\mathbf{SURE}(h) = -n\sigma^2 + \|\widetilde{\boldsymbol{F}} - h(\widetilde{\boldsymbol{F}})\|_2^2 + 2\sigma^2 \sum_{i,j=1}^{n(J+1)} \gamma_{ij}\partial_j h_i(\widetilde{\boldsymbol{F}}),$$

where $\widetilde{\boldsymbol{F}} = \mathcal{W}\tilde{\boldsymbol{f}}$ is the wavelet transform of the noisy signal $\tilde{\boldsymbol{f}}$. In [86], the weights $\gamma_{ij} = (\mathcal{W}\mathcal{W}^*)_{ij}$, $i, j = 1, \ldots, n(J+1)$, are computed from the full reduction of the Laplacian matrix which is no longer tractable for large graphs. However, as shown in [65], the SGWT can be efficiently approximated by using Chebyshev polynomials. Besides, it is clear from the probabilistic interpretation given in [86, Theorem 1] that

$$\forall i, j = 1, \ldots, n(J+1), \quad \gamma_{ij} = \mathbb{E}[(\mathcal{W}\boldsymbol{\varepsilon})_i(\mathcal{W}\boldsymbol{\varepsilon})_j],$$

where $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)$ are *i.i.d.* random variables with zero mean and variance one. Thus, taking advantage of this identity, we propose to estimate the weights with Monte Carlo approximation as follows:

1. Generate $(\varepsilon_{ik})_{i=1,\ldots,n,k=1,\ldots,N}$ of *i.i.d.* random variables such that $\mathbb{E}[\varepsilon_{ik}] = 0$ and $\mathbb{V}[\varepsilon_{ik}] = 1$

2. Compute
$$\hat{\gamma}_{ij} = \frac{1}{N}\sum_{k=1}^{N}(\mathcal{W}\boldsymbol{\varepsilon}_k)_i(\mathcal{W}\boldsymbol{\varepsilon}_k)_j,$$
where $\boldsymbol{\varepsilon}_k = (\varepsilon_{ik})_{i=1,\ldots,n}$ are random signals

Generally speaking, whereas Monte Carlo are simple methods to implement and can be easily parallelized, they suffer from their slow rate of convergence. In practice, a well-chosen distribution for the random variables ε_{ik} can result in a lower variance of the estimator $\mathbb{V}[\hat{\gamma}_{ij}]$ whose expression is given below. In fact, it is even more interesting to compute the variance of SURE when the estimator $\hat{\gamma}_{ij}$ is plugged in place of the weights γ_{ij} in the SURE expression.

3.3.1 Variance of the SURE weight estimator

A straightforward computation gives the expectation of $\hat{\gamma}_{ij}$

$$\mathbb{E}[\hat{\gamma}_{ij}] = \mathbb{E}\left[\left(\sum_{p=1}^{n} \mathcal{W}_{ip}\varepsilon_{p1}\right)\left(\sum_{p=1}^{n} \mathcal{W}_{jp}\varepsilon_{p1}\right)\right] = \sum_{p,q=1}^{n} \mathcal{W}_{ip}\mathcal{W}_{jq}\mathbb{E}[\varepsilon_{p1}\varepsilon_{q1}] = \sum_{p=1}^{n} \mathcal{W}_{ip}\mathcal{W}_{jp} = \gamma_{ij}.$$

The variance of $\hat{\gamma}_{ij}$ is given by the following result and its computation is derived underneath.

Proposition 3.3.1.

$$\mathbb{V}[\hat{\gamma}_{ij}] = \frac{1}{N}\left\{\mathbb{V}[\varepsilon_{11}^2]\sum_{p=1}^{n} \mathcal{W}_{ip}^2\mathcal{W}_{jp}^2 + 2\mathbb{E}[\varepsilon_{11}^2]^2 \sum_{\substack{p,q=1,\\ p\neq q}}^{n} \mathcal{W}_{ip}\mathcal{W}_{iq}\mathcal{W}_{jp}\mathcal{W}_{jq}\right\}.$$

Proof. The formula of the weights $\hat{\gamma}_{ij}$ is given by

$$\hat{\gamma}_{ij} = \frac{1}{N}\sum_{k=1}^{N}\left(\sum_{p=1}^{n} \mathcal{W}_{ip}\varepsilon_{pk}\right)\left(\sum_{p=1}^{n} \mathcal{W}_{jp}\varepsilon_{pk}\right).$$

The variance of $\hat{\gamma}_{ij}$ reads

$$\mathbb{V}[\hat{\gamma}_{ij}] = \frac{1}{N}\mathbb{V}\left[\left(\sum_{p=1}^{n} \mathcal{W}_{ip}\varepsilon_{p1}\right)\left(\sum_{p=1}^{n} \mathcal{W}_{jp}\varepsilon_{p1}\right)\right].$$

Then, on the one hand

$$\begin{aligned}\mathbb{E}\left[\left(\sum_{p=1}^{n} \mathcal{W}_{ip}\varepsilon_{p1}\right)^2\left(\sum_{p=1}^{n} \mathcal{W}_{jp}\varepsilon_{p1}\right)^2\right] &= \sum_{p,q,r,s=1}^{n} \mathcal{W}_{ip}\mathcal{W}_{iq}\mathcal{W}_{jr}\mathcal{W}_{js}\mathbb{E}[\varepsilon_{p1}\varepsilon_{q1}\varepsilon_{r1}\varepsilon_{s1}] \\ &= \mathbb{E}[\varepsilon_{11}^4]\sum_{p=1}^{n} \mathcal{W}_{ip}^2\mathcal{W}_{jp}^2 + \mathbb{E}[\varepsilon_{11}^2]^2 \sum_{\substack{p,r=1,\\ p\neq r}}^{n} \mathcal{W}_{ip}^2\mathcal{W}_{jr}^2 + 2\mathbb{E}[\varepsilon_{11}^2]^2 \sum_{\substack{p,q=1,\\ p\neq q}}^{n} \mathcal{W}_{ip}\mathcal{W}_{iq}\mathcal{W}_{jp}\mathcal{W}_{jq}.\end{aligned}$$

On the other hand,

$$\mathbb{E}\left[\left(\sum_{p=1}^{n} \mathcal{W}_{ip}\varepsilon_{p1}\right)^2\right] = \sum_{p,q=1}^{n} \mathcal{W}_{ip}\mathcal{W}_{iq}\mathbb{E}[\varepsilon_{p1}\varepsilon_{q1}] = \mathbb{E}[\varepsilon_{11}^2]\sum_{p=1}^{n} \mathcal{W}_{ip}^2.$$

Finally,

$$\begin{aligned}\mathbb{V}\left[\left(\sum_{p=1}^{n}\mathcal{W}_{ip}\varepsilon_{p1}\right)\left(\sum_{p=1}^{n}\mathcal{W}_{jp}\varepsilon_{p1}\right)\right] &= \mathbb{E}[\varepsilon_{11}^4]\sum_{p=1}^{n}\mathcal{W}_{ip}^2\mathcal{W}_{jp}^2 + \mathbb{E}[\varepsilon_{11}^2]^2\sum_{\substack{p,r=1,\\ p\neq r}}^{n}\mathcal{W}_{ip}^2\mathcal{W}_{jr}^2 \\ &\quad + 2\mathbb{E}[\varepsilon_{11}^2]^2\sum_{\substack{p,q=1,\\ p\neq q}}^{n}\mathcal{W}_{ip}\mathcal{W}_{iq}\mathcal{W}_{jp}\mathcal{W}_{jq} - \mathbb{E}[\varepsilon_{11}^2]^2\sum_{p,q=1}^{n}\mathcal{W}_{ip}^2\mathcal{W}_{jq}^2 \\ &= \mathbb{V}[\varepsilon_{11}^2]\sum_{p=1}^{n}\mathcal{W}_{ip}^2\mathcal{W}_{jp}^2 + 2\mathbb{E}[\varepsilon_{11}^2]^2\sum_{\substack{p,q=1,\\ p\neq q}}^{n}\mathcal{W}_{ip}\mathcal{W}_{iq}\mathcal{W}_{jp}\mathcal{W}_{jq}.\end{aligned}$$

□

Note the usual rate of convergence $\sqrt{N}$ from Monte Carlo estimation. In many papers in the literature ε_{11} is chosen to be distributed as a standard Gaussian random variable so that $\mathbb{V}[\varepsilon_{11}^2] = 2$. However, if ε_{11} has a centered Rademacher distribution with probability mass function $\frac{1}{2}\delta_{-1} + \frac{1}{2}\delta_1$, then ε_{11}^2 is deterministic and $\mathbb{V}[\varepsilon_{11}^2] = 0$. With such a choice, the variance of $\hat{\gamma}_{ij}$ is then reduced to

$$\mathbb{V}[\hat{\gamma}_{ij}] = \frac{2}{N}\sum_{\substack{p,q=1,\\ p\neq q}}^{n}\mathcal{W}_{ip}\mathcal{W}_{iq}\mathcal{W}_{jp}\mathcal{W}_{jq}.$$

This trick is actually well known in the literature [71]. This computation somehow provides arguments in favor of the Rademacher distribution.

Another way to further reduce the variance of $\hat{\gamma}_{ij}$ is to take advantage of the SGWT localization property. Let us denote by $\lfloor x \rfloor$ the integer part of a real $x \in \mathbb{R}$. Then, for any $i \in \{1, \ldots, n(J+1)\}$ and any $p \in \{1, \ldots, n\}$

$$|\mathcal{W}_{ip}| = \left|\left\langle \sqrt{\phi_{\lfloor i/n \rfloor}(\mathcal{L})}\boldsymbol{\delta}_{i-\lfloor i/n \rfloor}, \boldsymbol{\delta}_p \right\rangle\right| \leq \|\phi_{\lfloor i/n \rfloor}(\mathcal{L})\|_2^2 \leq 1.$$

Since the SGWT is localized both in the space and the frequency domain, $\mathcal{W}_{ip}$ vanishes as the geodesic distance between $i - \lfloor i/n \rfloor$ and p grows. Thus, the performance of the Monte Carlo estimation could be improved by a suitable calibration of the partition of unity. As a consequence, most terms in the expression of $\mathbb{V}[\hat{\gamma}_{ij}]$ are small thanks to the localization properties of SGWT.

3.3.2 Variance of the SURE estimator

The SURE plug-in estimator is obtained by replacing the weights γ_{ij} with their Monte Carlo estimators $\hat{\gamma}_{ij}$:

$$\widehat{\mathbf{SURE}}(h) = -n\sigma^2 + \|\widetilde{\boldsymbol{F}} - h(\widetilde{\boldsymbol{F}})\|_2^2 + 2\sigma^2 \sum_{i,j=1}^{n(J+1)} \hat{\gamma}_{ij}\partial_j h_i(\widetilde{\boldsymbol{F}}).$$

Given the observed wavelet coefficients $\widetilde{\boldsymbol{F}}$, this estimator of SURE has no bias as it is a linear function of the unbiased weight estimators $\hat{\gamma}_{ij}$. The following proposition presents its conditional variance.

Proposition 3.3.2.

$$\begin{aligned}\mathbb{V}[\widehat{\mathbf{SURE}}(h)|\widetilde{\boldsymbol{F}}] = \frac{4\sigma^4}{N} \sum_{i,j,k,\ell=1}^{n(J+1)} \partial_j h_i(\widetilde{\boldsymbol{F}})\partial_k h_\ell(\widetilde{\boldsymbol{F}}) \Bigg\{ &\mathbb{V}[\varepsilon_{11}^2] \sum_{p=1}^{n} \mathcal{W}_{ip}\mathcal{W}_{jp}\mathcal{W}_{kp}\mathcal{W}_{\ell p} \\ &+ \mathbb{E}[\varepsilon_{11}^2]^2 \sum_{\substack{p,q=1,\\ p\neq q}}^{n} \mathcal{W}_{ip}\mathcal{W}_{jq}\mathcal{W}_{kp}\mathcal{W}_{\ell q} + \mathbb{E}[\varepsilon_{11}^2]^2 \sum_{\substack{p,q=1,\\ p\neq q}}^{n} \mathcal{W}_{ip}\mathcal{W}_{jq}\mathcal{W}_{kq}\mathcal{W}_{\ell p} \Bigg\}.\end{aligned}$$

Here again, the Rademacher distribution reduces the variance compared to the Gaussian distribution.

Proof. From the SURE expression, it follows that

$$\begin{aligned}\mathbb{V}[\widehat{\mathbf{SURE}}(h)|\widetilde{\boldsymbol{F}}] &= 4\sigma^4 \mathbb{E}\left[\left(\sum_{i,j=1}^{n(J+1)} (\hat{\gamma}_{ij} - \gamma_{ij})\partial_j h_i(\widetilde{\boldsymbol{F}}) \right)^2 \right] \\ &= 4\sigma^4 \sum_{i,j,k,\ell=1}^{n(J+1)} \partial_j h_i(\widetilde{\boldsymbol{F}})\partial_k h_\ell(\widetilde{\boldsymbol{F}}) \mathbb{E}[(\hat{\gamma}_{ij} - \gamma_{ij})(\hat{\gamma}_{k\ell} - \gamma_{k\ell})] \\ &= 4\sigma^4 \sum_{i,j,k,\ell=1}^{n(J+1)} \partial_j h_i(\widetilde{\boldsymbol{F}})\partial_k h_\ell(\widetilde{\boldsymbol{F}}) \left[\mathbb{E}[\hat{\gamma}_{ij}\hat{\gamma}_{k\ell}] - \gamma_{ij}\gamma_{k\ell}\right].\end{aligned}$$

Then,

$$N^2\mathbb{E}[\hat{\gamma}_{ij}\hat{\gamma}_{k\ell}] = \sum_{a,b=1}^{N} \mathbb{E}\left[\left(\sum_{p=1}^{n} \mathcal{W}_{ip}\varepsilon_{pa} \right) \left(\sum_{p=1}^{n} \mathcal{W}_{jp}\varepsilon_{pa} \right) \left(\sum_{p=1}^{n} \mathcal{W}_{kp}\varepsilon_{pb} \right) \left(\sum_{p=1}^{n} \mathcal{W}_{\ell p}\varepsilon_{pb} \right) \right].$$

Developing each term of the sum above, it follows

$$N^2\mathbb{E}[\hat{\gamma}_{ij}\hat{\gamma}_{k\ell}] = \sum_{a,b=1}^{N}\sum_{p,q,r,s=1}^{n}\mathcal{W}_{ip}\mathcal{W}_{jq}\mathcal{W}_{kr}\mathcal{W}_{\ell s}\mathbb{E}[\varepsilon_{pa}\varepsilon_{qa}\varepsilon_{rb}\varepsilon_{sb}].$$

Thus, on the one hand

$$\begin{aligned}N^2\mathbb{E}[\hat{\gamma}_{ij}\hat{\gamma}_{k\ell}] &= N\mathbb{E}[\varepsilon_{11}^4]\sum_{p=1}^{n}\mathcal{W}_{ip}\mathcal{W}_{jp}\mathcal{W}_{kp}\mathcal{W}_{\ell p} + N\mathbb{E}[\varepsilon_{11}^2]^2\sum_{\substack{p,q=1,\\ p\neq q}}^{n}\mathcal{W}_{ip}\mathcal{W}_{jp}\mathcal{W}_{kq}\mathcal{W}_{\ell q}\\ &+ N\mathbb{E}[\varepsilon_{11}^2]^2\sum_{\substack{p,q=1,\\ p\neq q}}^{n}\mathcal{W}_{ip}\mathcal{W}_{jq}\mathcal{W}_{kp}\mathcal{W}_{\ell q} + N\mathbb{E}[\varepsilon_{11}^2]^2\sum_{\substack{p,q=1,\\ p\neq q}}^{n}\mathcal{W}_{ip}\mathcal{W}_{jq}\mathcal{W}_{kq}\mathcal{W}_{\ell p}\\ &+ N(N-1)\mathbb{E}[\varepsilon_{11}^2]^2\sum_{p,q=1}^{n}\mathcal{W}_{ip}\mathcal{W}_{jp}\mathcal{W}_{kq}\mathcal{W}_{\ell q}.\end{aligned}$$

On the other hand,

$$\gamma_{ij}\gamma_{k\ell} = \mathbb{E}[\varepsilon_{11}^2]^2\left(\sum_{p=1}^{n}\mathcal{W}_{ip}\mathcal{W}_{jp}\right)\left(\sum_{p=1}^{n}\mathcal{W}_{kp}\mathcal{W}_{\ell p}\right) = \mathbb{E}[\varepsilon_{11}^2]^2\sum_{p,q=1}^{n}\mathcal{W}_{ip}\mathcal{W}_{jp}\mathcal{W}_{kq}\mathcal{W}_{\ell q}.$$

Finally, the difference is given by

$$\begin{aligned}&N^2\mathbb{E}[\hat{\gamma}_{ij}\hat{\gamma}_{k\ell}] - N^2\gamma_{ij}\gamma_{k\ell}\\ &\quad= N\mathbb{E}[\varepsilon_{11}^4]\sum_{p=1}^{n}\mathcal{W}_{ip}\mathcal{W}_{jp}\mathcal{W}_{kp}\mathcal{W}_{\ell p} + N\mathbb{E}[\varepsilon_{11}^2]^2\sum_{\substack{p,q=1,\\ p\neq q}}^{n}\mathcal{W}_{ip}\mathcal{W}_{jp}\mathcal{W}_{kq}\mathcal{W}_{\ell q}\\ &\quad+ N\mathbb{E}[\varepsilon_{11}^2]^2\sum_{\substack{p,q=1,\\ p\neq q}}^{n}\mathcal{W}_{ip}\mathcal{W}_{jq}\mathcal{W}_{kp}\mathcal{W}_{\ell q} + N\mathbb{E}[\varepsilon_{11}^2]^2\sum_{\substack{p,q=1,\\ p\neq q}}^{n}\mathcal{W}_{ip}\mathcal{W}_{jq}\mathcal{W}_{kq}\mathcal{W}_{\ell p}\\ &\quad+ N(N-1)\mathbb{E}[\varepsilon_{11}^2]^2\sum_{p,q=1}^{n}\mathcal{W}_{ip}\mathcal{W}_{jp}\mathcal{W}_{kq}\mathcal{W}_{\ell q} - N^2\mathbb{E}[\varepsilon_{11}^2]^2\sum_{p,q=1}^{n}\mathcal{W}_{ip}\mathcal{W}_{jp}\mathcal{W}_{kq}\mathcal{W}_{\ell q}.\end{aligned}$$

Hence,

$$N^2\mathbb{E}[\hat{\gamma}_{ij}\hat{\gamma}_{k\ell}] - N^2\gamma_{ij}\gamma_{k\ell} = N\mathbb{V}[\varepsilon_{11}^2]\sum_{p=1}^{n}\mathcal{W}_{ip}\mathcal{W}_{jp}\mathcal{W}_{kp}\mathcal{W}_{\ell p} + N\mathbb{E}[\varepsilon_{11}^2]^2\sum_{\substack{p,q=1,\\ p\neq q}}^{n}\mathcal{W}_{ip}\mathcal{W}_{jq}\mathcal{W}_{kp}\mathcal{W}_{\ell q}$$
$$+ N\mathbb{E}[\varepsilon_{11}^2]^2\sum_{\substack{p,q=1,\\ p\neq q}}^{n}\mathcal{W}_{ip}\mathcal{W}_{jq}\mathcal{W}_{kq}\mathcal{W}_{\ell p}.$$

Summarizing,

$$\mathbb{V}[\widehat{\mathbf{SURE}}(h)|\widetilde{\boldsymbol{F}}] = \frac{4\sigma^4}{N}\sum_{i,j,k,\ell=1}^{n(J+1)}\partial_j h_i(\widetilde{\boldsymbol{F}})\partial_k h_\ell(\widetilde{\boldsymbol{F}})\Bigg\{\mathbb{V}[\varepsilon_{11}^2]\sum_{p=1}^{n}\mathcal{W}_{ip}\mathcal{W}_{jp}\mathcal{W}_{kp}\mathcal{W}_{\ell p}$$
$$+ \mathbb{E}[\varepsilon_{11}^2]^2\sum_{\substack{p,q=1,\\ p\neq q}}^{n}\mathcal{W}_{ip}\mathcal{W}_{jq}\mathcal{W}_{kp}\mathcal{W}_{\ell q} + \mathbb{E}[\varepsilon_{11}^2]^2\sum_{\substack{p,q=1,\\ p\neq q}}^{n}\mathcal{W}_{ip}\mathcal{W}_{jq}\mathcal{W}_{kq}\mathcal{W}_{\ell p}\Bigg\}.$$

□

3.3.3 Computational complexity

The polynomial approximation of all random signal wavelet transforms $\mathcal{W}\boldsymbol{\varepsilon}_k$ is of order $\mathcal{O}(N(mK + n(J+1)K))$, where m is the number of edges in the graph [65]. Then, computing every $(\hat{\gamma}_{ij})_{i,j=1,\dots,n(J+1)}$ term requires $\mathcal{O}(n^2(J+1)^2(2N-1))$ operations. The computation of all the weights is useful when performing block thresholding on the wavelet coefficients $\mathcal{W}\tilde{\boldsymbol{f}}$ [86] which shows good denoising performance but is relatively computationally expensive. Alternatively, a coordinatewise thresholding process only needs the diagonal weights $(\hat{\gamma}_{ii})_{i=1,\dots,n(J+1)}$ whose computation is reduced to $\mathcal{O}(n(J+1)N)$ operations.

After this initial weight estimation, the computational complexity for the approximated wavelet transform of the noisy signal $\tilde{\boldsymbol{f}}$ is $\mathcal{O}(mK+n(J+1)K)$. The coordinatewise thresholding step has an average cost of $\mathcal{O}(n(J+1)\ln(n(J+1)))$ according to [39]. Finally, the approximated inverse transform has the same complexity as the forward transform.

3.4 Differential privacy and Gaussian mechanism

We now give a definition of differential privacy [46] which constitutes a strong standard for privacy guarantees about algorithms that use sensitive data. Let $(X_1, \dots, X_m)$ be a random vector containing the private data of m individuals we wish to protect with a *privacy mechanism*. This information is collected in a dataset $X = (X_1, \dots, X_m)$ that serves as an input to the privacy mechanism M which returns a *sanitized* output $Z = (Z_1, \dots, Z_k) = M(X)$ that preserves the privacy of each individual. Let $(\mathcal{X}^m, \mathcal{A}^m)$ and $(\mathcal{Z}, \mathcal{B})$ be the measurable spaces where X and Z respectively take values. The privacy mechanism distribution $Q(\cdot|X)$ corresponds to the conditional distribution of Z given X, that is $Q(B|x) = \mathbb{P}(Z \in B|X = x)$, where $Q(\cdot|\cdot) : \mathcal{B} \times \mathcal{X}^m \to [0, 1]$ is a Markov kernel.

Let $\varepsilon \geq 0$ be the privacy budget and $\delta \geq 0$ another privacy parameter. The privacy mechanism M is said to satisfy (ε, δ)-differential privacy if for any two datasets $x, x' \in \mathcal{X}^m$ that differ in a single entry and for any subset of outputs $B \in \mathcal{B}$, we have

$$Q(B|x) \leq e^{\varepsilon} Q(B|x') + \delta.$$

As it appears from this definition, smaller privacy parameters lead to closer output distributions and hence a better privacy preserving mechanism. Intuitively, differential privacy protects individuals by ensuring the inclusion or removal of their information from the input dataset does not affect much the output distribution.

In this paper, we are interested in functions $f : \mathcal{X}^m \to \mathbb{R}^n$ that map a dataset X to a graph signal $\boldsymbol{f} \in \mathbb{R}^n$. In order to achieve differential privacy, a common method is to introduce just enough uncertainty in the function response to hide the participation of any single individual. The Gaussian mechanism does so by adding white Gaussian noise $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$ to the response where the standard deviation σ is calibrated to the privacy parameters and a third quantity $\Delta_2 f$ called the ℓ_2-sensitivity. It is defined by $\Delta_2 f = \max \|f(x) - f(x')\|_2$ which corresponds to the maximum impact a single individual's information can have on the signal $\boldsymbol{f}$. This method yields a sanitized output $Z = f(X) + \boldsymbol{\xi}$ that can be interpreted as a noisy signal $\tilde{\boldsymbol{f}} = \boldsymbol{f} + \boldsymbol{\xi}$ with our noise corruption model given in Section 3.2.

We present two Gaussian mechanisms that use different variance formulas to sanitize a function $f : \mathcal{X}^m \to \mathbb{R}^n$ with ℓ_2-sensitivity $\Delta_2 f$. First, the classical Gaussian mechanism proposed by [46] preserves (ε, δ)-differential privacy for any $\varepsilon, \delta \in (0, 1)$ if $\sigma \geq \Delta_2 f \sqrt{2 \ln(1.25/\delta)}/\varepsilon$. The authors of [7] have shown that this formula is not optimal

and can be further improved to reduce the amount of noise needed to achieve the same degree of privacy. Whereas the classical Gaussian mechanism uses a Gaussian tail approximation to obtain a bound for the standard deviation, their proposed approach uses numerical evaluations of the cumulative Gaussian distribution function $\Phi(x) = \mathbb{P}(\mathcal{N}(0,1) \leq x)$ to determine an optimal variance. Their analytic Gaussian mechanism preserves (ε, δ)-differential privacy for any $\varepsilon \geq 0$ and $\delta \in [0,1]$ if and only if

$$\Phi\left(\frac{\Delta_2 f}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_2 f}\right) - e^{\varepsilon}\Phi\left(-\frac{\Delta_2 f}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_2 f}\right) \leq \delta.$$

The Gaussian mechanism offers a solution to sanitize a signal at the expense of its utility as it is perturbed by the introduced noise. Indeed, there is a trade-off between privacy and utility: very small values of ε and δ ensure a strong degree of privacy but can be detrimental to the utility of the sanitized data, and vice versa. A valuable aspect of differential privacy is its immunity to post-processing [44, Prop. 2.1] as long as no knowledge about the original signal is used. Formally, the composition of any data-independent function with an (ε, δ)-differentially private mechanism is also (ε, δ)-differentially private. Therefore, a data-driven denoising method such as ours can improve the utility of a sanitized signal with no loss of privacy. Indeed, the SGWT, SURE and their respective approximations only require information contained in the known Laplacian matrix $\mathcal{L}$ and observed noisy signal $\tilde{\boldsymbol{f}}$.

3.5 Numerical experiments

In this section, we present an experimental evaluation of our Monte Carlo estimator of the SURE weights described in Section 3.3 and graph signal denoising methodology. Specifically, we are interested in the sanitization and denoising of density maps of located events over a given period. First, we consider signals built from real datasets gathering positions of taxis in the cities of New York and San Francisco. The relatively small size of their corresponding graphs enables us to diagonalize their respective Laplacian matrices and directly compute the SGWT and SURE in a reasonable amount of time, thus allowing the comparison with our approximation method. Then, we generate signals on a large graph for which the eigendecomposition of the Laplacian matrix is not tractable. Hereafter, wavelet transforms are performed with the piecewise linear ω function presented in Section 3.2.1 and each polynomial approximation is of degree $K = 100$.

3.5.1 Monte Carlo estimator of the SURE weights

This experiment makes use of the New York City (NYC) yellow taxi trip records publicly released each year by the Taxi and Limousine Commission (TLC). These datasets contain in particular the pickup and drop-off locations and times of each trip whose distribution has been the subject of different GSP applications [103, 7, 24]. In the past, a bad pseudonymization of the taxi ID led to a privacy breach [41] for the drivers and their passengers about where they might reside and the places they frequent.

Since the yellow taxis mostly operate in the borough of Manhattan, we focus our experiment on Manhattan Island and build an associated graph with **OSMnx** [15]. This Python package automatically downloads urban networks from the OpenStreetMap database, converts them to graph objects of the **NetworkX** [62] package and offers a variety of analysis tools. The resulting graph consists of 4513 vertices and 9743 edges representing street intersections and segments, respectively.

With this first graph, we evaluate the SURE weights Monte Carlo estimators when their samples are either drawn from a centered Rademacher or standard Gaussian distribution. Their SGWT are computed with the Chebyshev polynomial approximation and we estimate the weights for different Monte Carlo sample sizes N. We focus on the diagonal weights $\hat{\gamma}_{ii}$, $i = 1, \dots, n(J+1)$ as only these are needed in the coordinatewise thresholding process and compare them to the weights obtained from the direct transform by averaging the MSE along the n vertices and $(J+1)$ scales over 50 repetitions: $\mathbf{MSE}((\gamma_{ii})_i, (\hat{\gamma}_{ii})_i) = \frac{1}{n(J+1)} \sum_{i=1}^{n(J+1)} (\gamma_{ii} - \hat{\gamma}_{ii})^2$.

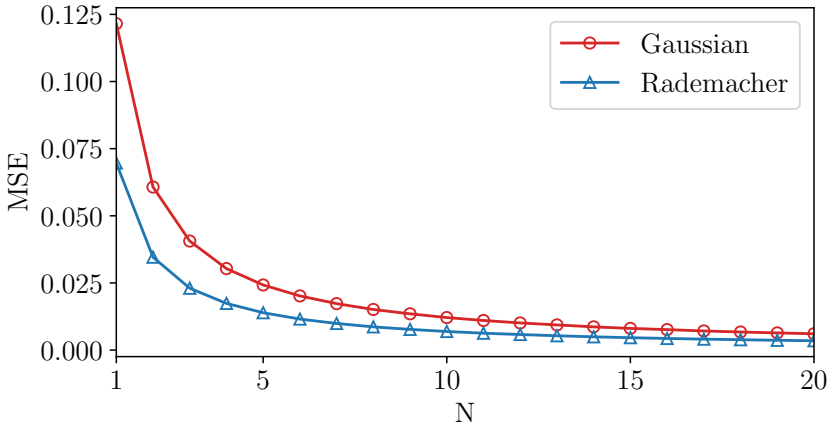


Figure 3.3 – Average MSE between the SURE weights γ_{ii} and their Monte Carlo estimators $\hat{\gamma}_{ii}$ over 50 repetitions.

Results are presented in Figure 3.3 where we see that drawing samples from the centered Rademacher distribution gives estimates closer to the real weights in terms of MSE for any sample size compared to the standard Gaussian distribution. This illustrates the gain in variance achieved with the former distribution as mentioned in Section 3.3.1.

3.5.2 Monte Carlo estimator of the SURE

We now present how SURE behaves when the estimated weights $\hat{\gamma}_{ii}$ are plugged in. On the same graph, we build a signal $\boldsymbol{f}$ by counting the number of taxi pickups projected to the nearest intersection over a period of one hour. Here, we consider the time interval between 00:00 and 01:00 on Sept 24, 2014, as chosen by [7] to compare our results in a similar configuration. We add some white Gaussian noise with standard deviation $\sigma = 1$ to obtain a noisy signal and compute its SGWT coefficients with the Chebyshev-Jackson approximation. The denoising is done by applying the James-Stein thresholding function $\tau(x, t) = x \max\{1 - t^2|x|^{-2}, 0\}$ to the coefficients with the threshold that minimizes $\mathbf{MSE}(\boldsymbol{f}, \hat{\boldsymbol{f}})$. Finally, for this optimal threshold we estimate SURE between the estimate $\hat{\boldsymbol{f}}$ and the original signal with known σ for different Monte Carlo sample sizes and both the centered Rademacher and standard Gaussian distributions.

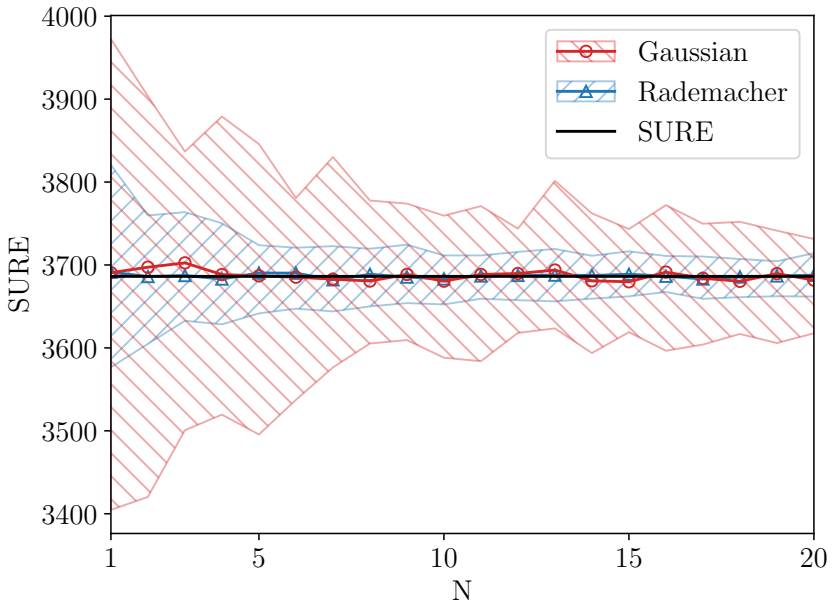


Figure 3.4 – Average SURE Monte Carlo estimate and 95% CI over 50 repetitions.

Figure 3.4 shows the average SURE estimate over 50 repetitions along with a 95%

empirical confidence interval. We visualize the SURE plug-in estimator unbiasedness as it is centered on the real SURE value. Additionally, we observe the smaller variance of the Monte Carlo estimator when samples are drawn from the centered Rademacher distribution compared to the Gaussian distribution, as previously mentioned in Section 3.3.2. As a result, we estimate SURE in the following experiments with 10 samples from this distribution.

3.5.3 Denoising of differentially private graph signals

We illustrate denoising performance with the SURE Monte Carlo estimator on two relatively small graphs. This allows for the explicit eigendecomposition of their associated Laplacian matrices and computation of the SGWT and SURE with which we compare our proposed method.

New York City taxis

Considering the number of taxi pickups at each intersection from the last experiment as our signal, we now apply the differential privacy mechanisms presented in Section 3.4 to sanitize it. Both the classical and analytic Gaussian mechanisms are used for different values of the privacy budget ε and therefore noise levels σ, while the other privacy parameter is set to $\delta = 10^{-6}$ as in [7]. Note that to satisfy the constraint associated with the maximum value taken by the ε parameter, the resulting noise levels are particularly high. We aim to protect the taxi passengers and assume they only take a taxi once within an hour. This gives an upper bound of their individual contribution to the signal and thus we have an ℓ_2-sensitivity of $\Delta_2 f = 1$.

We compare different denoising methods using signal-to-noise ratio $\mathbf{SNR}(\boldsymbol{f}, \hat{\boldsymbol{f}}) = 20\log_{10}(\|\boldsymbol{f}\|_2/\|\boldsymbol{f} - \hat{\boldsymbol{f}}\|_2)$ as a performance measure. We also compute it between the original and noisy signals to get a baseline of the amount of input noise after sanitization: $\mathrm{SNR}_{\mathrm{in}} = \mathbf{SNR}(\boldsymbol{f}, \tilde{\boldsymbol{f}})$. Three denoising methods based on the application of a level-dependent James-Stein thresholding function to wavelet coefficients are considered, each of them uses a different criterion to select the optimal thresholds: (1) an oracle estimator that directly computes the SGWT and minimizes the real MSE; (2) a second estimator that instead minimizes SURE; and (3) our proposed estimator that approximates the SGWT with Chebyshev-Jackson polynomials and estimates SURE with Monte Carlo. As shown by [39], for a coordinatewise thresholding process such as James-Stein,

SURE reaches its minimum for some threshold t chosen among the absolute values of the noisy wavelet coefficients $\{|\widetilde{F}(i)|, i = 1, \ldots, n(J+1)\}$. We further reduce this set to its percentiles to find a compromise between the range and number of candidate threshold values.

These estimators are compared to the DFS fused lasso, a regularization method introduced in [69]. It first performs a standard depth-first search (DFS) traversal algorithm to reduce the initial graph to a chain graph. Then, it runs a 1-dimensional fused lasso [125], a special case of graph trend filtering [132], over this simpler graph. In doing so, this method avoids the prohibitive computational cost of standard graph trend filtering over an arbitrary graph at the expense of less statistical accuracy. Here, the comparison is made on unweighted graphs as the DFS fused lasso is limited to them, whereas the SGWT can be applied to graphs with edge weights. In the experiments, the DFS and fused lasso are respectively conducted with the **igraph** [30] and **glmgen** [5] R packages.

Table 3.1 – Average SNR performance over 10 realizations of high to low privacy budget sanitization on the NYC graph.

	Classical Gaussian mechanism				Analytic Gaussian mechanism			
ε	0.2	0.3	0.5	1	0.2	0.3	0.5	1
σ	26.49	17.66	10.60	5.30	18.99	12.99	8.06	4.22
SNR_{in}	-12.48 ± 0.11	-8.96 ± 0.11	-4.52 ± 0.11	1.5 ± 0.11	-9.59 ± 0.11	-6.29 ± 0.11	-2.14 ± 0.11	3.47 ± 0.11
SGWT_{MSE}	0.23 ± 0.32	1.61 ± 0.26	3.55 ± 0.23	7.0 ± 0.16	1.36 ± 0.27	2.73 ± 0.23	4.79 ± 0.18	8.29 ± 0.15
$\text{SGWT}_{\text{SURE}}$	0.1 ± 0.24	1.5 ± 0.31	3.52 ± 0.2	6.94 ± 0.17	1.25 ± 0.38	2.71 ± 0.25	4.75 ± 0.2	8.24 ± 0.18
$\text{SGWT}^{\text{CJ}}_{\text{SURE, MC}}$	0.1 ± 0.28	1.51 ± 0.34	3.52 ± 0.24	6.92 ± 0.15	1.3 ± 0.34	2.71 ± 0.25	4.74 ± 0.22	8.24 ± 0.17
DFS_{MSE}	0.88 ± 0.02	1.18 ± 0.09	2.33 ± 0.18	5.58 ± 0.15	1.09 ± 0.09	1.72 ± 0.12	3.43 ± 0.17	6.96 ± 0.16
DFS_{SURE}	0.85 ± 0.03	1.11 ± 0.08	2.26 ± 0.16	5.55 ± 0.14	1.02 ± 0.07	1.65 ± 0.18	3.4 ± 0.15	6.95 ± 0.16

Table 3.1 summarizes the results of this experiment over 10 sanitization realizations. We observe that the wavelet transform oracle estimator (SGWT_{MSE}) performs better than the oracle DFS fused lasso (DFS_{MSE}) for most values of privacy budget. When considering stronger degrees of privacy with the classical Gaussian mechanism which requires the most amount of input noise, the oracle DFS fused lasso presents better results. Our approach combining Chebyshev-Jackson polynomial approximation with SURE Monte Carlo estimation ($\text{SGWT}^{\text{CJ}}_{\text{SURE, MC}}$) gives slightly lower SNR values than its oracle counterpart but nevertheless shows better denoising performance compared to the regularization method except for the case where the noise is very high.

San Francisco taxis

We check these initial results with a second dataset that contains the GPS coordinates of 536 taxis collected over a month in the San Francisco Bay Area [105]. Each entry

consists of the taxi location and whether it currently has passengers at a given time with approximately one minute between updates. In a similar fashion as for the previous experiment, we concentrate on the city of San Francisco and get the associated graph of the street network from **OSMnx**. It is about twice as large with 9,573 vertices and 15,716 edges, causing a longer but still practicable computation of the SGWT.

Pickup locations are inferred by keeping the entries whose occupancy status goes from "free" to "occupied", giving an approximation close to the minute. We build a signal by counting these pickups projected to the nearest intersection on the day of May 25, 2008. Sanitization is then applied with the analytic Gaussian mechanism and parameter values $\delta = 10^{-6}$ and $\Delta_2 f = 2$. The latter is chosen by assuming the individual passengers do a maximum of four taxi trips within a day, all starting from distinct places.

Table 3.2 – Average SNR performance over 10 realizations of high to low privacy budget sanitization on the San Francisco graph.

ε	0.20	0.50	1
σ	37.98	16.12	8.45
$\mathrm{SNR}_{\mathrm{in}}$	-11.95 ± 0.05	-4.51 ± 0.05	1.1 ± 0.05
$\mathrm{SGWT}_{\mathrm{MSE}}$	0.80 ± 0.27	4.34 ± 0.11	8.24 ± 0.07
$\mathrm{SGWT}_{\mathrm{SURE}}$	0.77 ± 0.31	4.32 ± 0.09	8.22 ± 0.06
$\mathrm{SGWT}^{\mathrm{CJ}}_{\mathrm{SURE,\ MC}}$	0.76 ± 0.32	4.34 ± 0.12	8.22 ± 0.08
$\mathrm{DFS}_{\mathrm{MSE}}$	0.22 ± 0.02	3.02 ± 0.1	6.77 ± 0.1
$\mathrm{DFS}_{\mathrm{SURE}}$	0.17 ± 0.07	2.99 ± 0.12	6.76 ± 0.1

Results presented in Table 3.2 are in line with those obtained above. The wavelet transform oracle estimator gives the best overall results and our method performs again better than the oracle DFS fused lasso for the considered privacy budget values.

3.5.4 Denoising of large graph signals

In this experiment, we apply our method on a large graph whose scale prevents us from decomposing the Laplacian matrix due to the prohibitive computational cost. The road network of Pennsylvania from [82] is such a graph consisting of 1,088,092 vertices and 1,541,898 edges. Synthetic signals are generated on this graph following the methodology proposed in [9]: with two parameters $p \in (0, 1)$ and $k \in \mathbb{N}$, we produce a signal $\boldsymbol{f}_{p,k} = \mathbf{W}^k \boldsymbol{x}_p$ where $\boldsymbol{x}_p$ is an i.i.d. realization of n Bernoulli random variables of parameter p. As this data is entirely simulated and not related to the information of real individuals,

the added noise does not depend on some privacy budget ε value and is instead directly chosen. Here, a noisy signal $\tilde{\boldsymbol{f}} = \boldsymbol{f}_{0.001,4} + \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$ is generated for different values of σ.

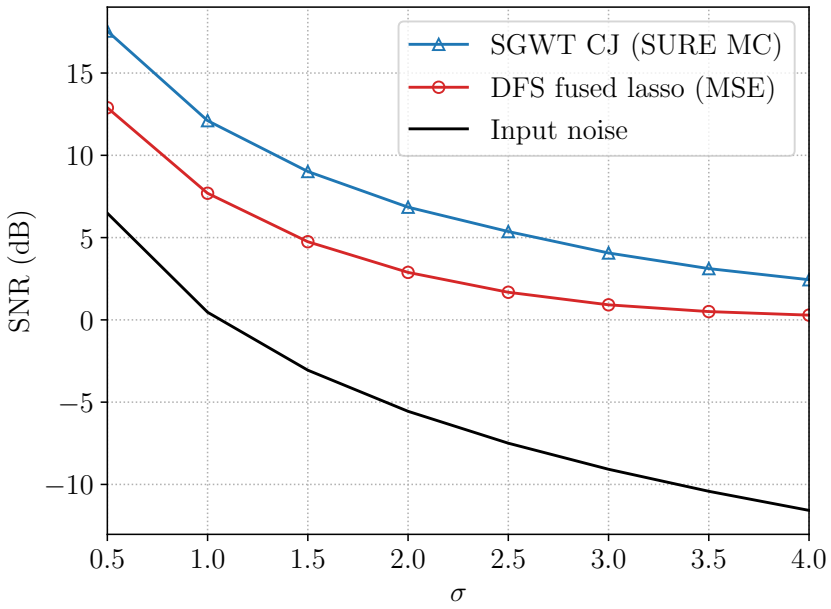


Figure 3.5 – Average SNR performance over 5 realizations of each noise level setting on the Pennsylvania graph

Figure 3.5 presents the average SNR values over 5 noise realizations for our proposed estimator and the oracle DFS fused lasso. We see that the results observed on small graph signals extend well to the large-scale setting, with better performance for the Chebyshev-Jackson polynomial approximation with the SURE Monte Carlo estimator on a range of input SNR similar to the previous experiments.

Regarding computing time, the DFS fused lasso is, however, more efficient than the approximated SGWT and estimated SURE for this application. On a standard laptop (Intel Core i5@1.70GHz-16Go DDR4@2400MHz), each of the realizations is denoised in 4 seconds by the regularization method while it takes less about 3 minutes for the latter after an initial estimation of the SURE weights done in 1 minute. We do not observe a significant difference between drawing Monte Carlo samples from the Rademacher or the Gaussian distributions, both take the same amount of time. In our procedure, the most time-consuming step is the threshold optimization (2m30s), followed by the inverse wavelet transform (25s) and the forward transform (7s).

CONCLUSION

In this dissertation, we propose an extension of SURE to large-scale graphs in the context of signal denoising with thresholding of SGWT coefficients. In particular, we use Monte Carlo and Chebyshev-Jackson polynomial approximation to build an estimator of its weights in order to avoid the computationally expensive eigendecomposition of the graph Laplacian matrix. Provided expressions for the variance of both weights and SURE estimators show that the Rademacher distribution is better suited than the Gaussian one for this method. We evaluate our data-driven approach through numerical experiments with an application in differential privacy to improve the utility of sanitized graph signals. Results show the MSE can be efficiently estimated with our extended SURE on small and large graphs. Additionally, this methodology shows better performance than the DFS fused lasso.

There is room for improvement in this approach to remove noise with better precision. For instance, the thresholding function we used can be generalized to $\tau_\beta(x,t) = x\max\{1 - t^\beta|x|^{-\beta}, 0\}$ with $\beta \geq 1$. Common choices for β include soft thresholding ($\beta = 1$) and hard thresholding ($\beta = \infty$) but an optimization algorithm for this parameter would be more beneficial to further improve performance. An additional thresholding strategy worth considering is block thresholding which partitions wavelet coefficients within each scale to identify localized features in the signal. Depending on the regularity of the original signal, this can help to remove noise with more accuracy as a different threshold value is selected for each block. An expression of SURE provided for block thresholding processes with SGWT coefficients by [86] could be extended to large graphs with our approach.

Another direction for future research is to adapt our methodology to be run on a distributed system in order to further reduce computing time. First, sanitization with the Gaussian mechanism only consists of the addition of independent Gaussian noise to each vertex of the graph. Differential privacy in this case can thus be achieved in a distributed manner over subgraphs of the initial graph. Threshold selection by SURE optimization can also be computed in a distributed manner thanks to the additive nature of the SURE formula. But the Laplacian matrix and the SGWT cannot be directly decomposed over separated groups of graph vertices. This yields at least two important challenges in order

to distribute 1) the computation of the weights in the SURE formula, and 2) the SGWT thresholding procedure. For specific graph structures (e.g. relatively distinct subgraphs), localization properties of the SGWT would certainly help finding accurate approximations for these distributed computations.

Regarding potential applications combining large graph signal denoising and differential privacy, we can mention the method from [1] which trains a neural network with differential privacy guarantees by adding Gaussian noise to the gradient at each training step. An interesting avenue of research would be to develop an appropriate graph structure on which the trained model parameters are defined as a noisy signal. Our denoising methodology may then be used throughout or after the training process in order to restore some of the performance lost with the sanitization. As neural networks can contain a large number of parameters, our approach would be well adapted to this application.

BIBLIOGRAPHY

[1] Martin Abadi et al., « Deep Learning with Differential Privacy », *in*: CCS '16 (2016), pp. 308–318, DOI: 10.1145/2976749.2978318, URL: https://doi.org/10.1145/2976749.2978318.

[2] Martın Abadi et al., « TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow. org (2015) », *in*: *URL https://www. tensorflow. org* (2015).

[3] Claudio Altafini, « Consensus problems on networks with antagonistic interactions », *in*: *IEEE transactions on automatic control* 58.*4* (2012), pp. 935–946.

[4] Miguel E Andrés et al., « Geo-indistinguishability: Differential privacy for location-based systems », *in*: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 901–914.

[5] Taylor Arnold, Veeranjaneyulu Sadhanala, and Ryan Tibshirani, « glmgen: Fast algorithms for generalized lasso problems », *in*: (2014), R package version 0.0.3.

[6] Borja Balle, Gilles Barthe, and Marco Gaboardi, « Privacy Profiles and Amplification by Subsampling », *in*: *Journal of Privacy and Confidentiality* 10.*1* (Jan. 2020), pp. 1–32, ISSN: 2575-8527, DOI: 10.29012/jpc.726, URL: https://journalprivacyconfidentiality.org/index.php/jpc/article/view/726.

[7] Borja Balle and Yu-Xiang Wang, « Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising », *in*: *Proceedings of the 35th International Conference on Machine Learning*, ed. by Jennifer Dy and Andreas Krause, PMLR, 2018, pp. 394–403, URL: http://proceedings.mlr.press/v80/balle18a.html.

[8] Borja Balle et al., « Privacy amplification by mixing and diffusion mechanisms », *in*: *Advances in neural information processing systems* 32 (2019).

[9] Hamid Behjat et al., « Signal-Adapted Tight Frames on Graphs », *in*: *IEEE Transactions on Signal Processing* 64.*22* (Nov. 2016), pp. 6017–6029, ISSN: 1053587X, DOI: 10.1109/TSP.2016.2591513.

[10] Mikhail Belkin and Partha Niyogi, « Towards a theoretical foundation for Laplacian-based manifold methods », *in*: *Journal of Computer and System Sciences* 74.*8* (2008), Learning Theory 2005, pp. 1289–1308, ISSN: 0022-0000, DOI: https://doi.org/10.1016/j.jcss.2007.08.006, URL: https://www.sciencedirect.com/science/article/pii/S0022000007001274.

[11] John J Benedetto and Matthew Fickus, « Finite normalized tight frames », *in*: *Advances in Computational Mathematics* 18.*2* (2003), pp. 357–385.

[12] Tarek Benkhelif et al., « Co-clustering for differentially private synthetic data generation », *in*: *Personal Analytics and Privacy. An Individual and Collective Perspective: First International Workshop, PAP 2017, Held in Conjunction with ECML PKDD 2017, Skopje, Macedonia, September 18, 2017, Revised Selected Papers 1*, Springer, 2017, pp. 36–47.

[13] Simeon Berman, *Sojourns and extremes of stochastic processes*, CRC Press, 1992.

[14] Jeremiah Blocki et al., « Differentially private data analysis of social networks via restricted sensitivity », *in*: *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, 2013, pp. 87–96.

[15] Geoff Boeing, « OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks », *in*: *Computers, Environment and Urban Systems* 65 (Sept. 2017), pp. 126–139, ISSN: 0198-9715, DOI: 10.1016/J.COMPENVURBSYS.2017.05.004.

[16] Michael M Bronstein et al., « Geometric deep learning: going beyond euclidean data », *in*: *IEEE Signal Processing Magazine* 34.*4* (2017), pp. 18–42.

[17] Andries E Brouwer and Willem H Haemers, *Spectra of graphs*, Springer Science & Business Media, 2011.

[18] Joan Bruna et al., « Spectral networks and locally connected networks on graphs », *in*: *ICLR*, 2014.

[19] Solenn Brunet et al., « Edge-calibrated noise for differentially private mechanisms on graphs », *in*: *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, IEEE, 2016, pp. 42–49.

[20] T Tony Cai, « Adaptive wavelet estimation: a block thresholding and oracle inequality approach », *in*: *The Annals of statistics* 27.*3* (1999), pp. 898–924.

[21] T Tony Cai, Yichen Wang, and Linjun Zhang, « The cost of privacy: Optimal rates of convergence for parameter estimation with differential privacy », *in*: *The Annals of Statistics* 49.*5* (2021), pp. 2825–2850.

[22] Konstantinos Chatzikokolakis et al., « Broadening the Scope of Differential Privacy Using Metrics », *in*: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7981 LNCS (2013), pp. 82–102, ISSN: 03029743, DOI: 10.1007/978-3-642-39077-7_5, URL: https://link.springer.com/chapter/10.1007/978-3-642-39077-7_5.

[23] Elie Chedemail et al., « Large Graph Signal Denoising with Application to Differential Privacy », *in*: *IEEE Transactions on Signal and Information Processing over Networks* 8 (2022), pp. 788–798.

[24] Siheng Chen, Aarti Singh, and Jelena Kovačević, *Multiresolution Representations for Piecewise-Smooth Signals on Graphs*, 2018, arXiv: 1803.02944 [eess.SP].

[25] F. R. K. Chung, *Spectral Graph Theory*, American Mathematical Society, 1997.

[26] Ronald R Coifman and Mauro Maggioni, « Diffusion wavelets », *in*: *Applied and computational harmonic analysis* 21.*1* (2006), pp. 53–94.

[27] Ronald R. Coifman and Mauro Maggioni, « Diffusion wavelets », *in*: *Applied and Computational Harmonic Analysis* 21.*1* (2006), Special Issue: Diffusion Maps and Wavelets, pp. 53–94, ISSN: 1063-5203, DOI: https://doi.org/10.1016/j.acha.2006.04.004, URL: http://www.sciencedirect.com/science/article/pii/S106352030600056X.

[28] Thierry Coulhon, Gerard Kerkyacharian, and Pencho Petrushev, « Heat kernel generated frames in the setting of Dirichlet spaces », *in*: *Journal of Fourier Analysis and Applications* 18 (2012), pp. 995–1066.

[29] Mark Crovella and Eric Kolaczyk, « Graph wavelets for spatial traffic analysis », *in*: *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, vol. 3, IEEE, 2003, pp. 1848–1857.

[30] Gabor Csardi and Tamas Nepusz, « The igraph software package for complex network research », *in*: *InterJournal* Complex Systems (2006), p. 1695, URL: https://igraph.org.

[31] Kinkar Ch. Das, « Extremal graph characterization from the bounds of the spectral radius of weighted graphs », *in*: *Applied Mathematics and Computation* 217.*18* (2011), pp. 7420–7426, ISSN: 0096-3003, DOI: https://doi.org/10.1016/j.amc.2011.02.033, URL: https://www.sciencedirect.com/science/article/pii/S0096300311002232.

[32] Ingrid Daubechies, *Ten lectures on wavelets*, SIAM, 1992.

[33] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst, « Convolutional neural networks on graphs with fast localized spectral filtering », *in*: *NIPS*, 2016, pp. 3844–3852.

[34] Edoardo Di Napoli, Eric Polizzi, and Yousef Saad, « Efficient estimation of eigenvalue counts in an interval », *in*: *Numerical Linear Algebra with Applications* 23.*4* (2016), pp. 674–692.

[35] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin, « Collecting telemetry data privately », *in*: *Advances in Neural Information Processing Systems* 30 (2017).

[36] Jinshuo Dong, Aaron Roth, and Weijie J. Su, *Gaussian Differential Privacy*, 2019, DOI: 10.48550/ARXIV.1905.02383, URL: https://arxiv.org/abs/1905.02383.

[37] Wei Dong, Charikar Moses, and Kai Li, « Efficient k-nearest neighbor graph construction for generic similarity measures », *in*: *Proceedings of the 20th international conference on World wide web*, 2011, pp. 577–586.

[38] Xiaowen Dong et al., « Graph signal processing for machine learning: A review and new perspectives », *in*: *IEEE Signal Processing Magazine* 37.*6* (2020), pp. 117–127.

[39] David L Donoho and Iain M Johnstone, « Adapting to unknown smoothness via wavelet shrinkage », *in*: *Journal of the american statistical association* 90.*432* (1995), pp. 1200–1224.

[40] David L Donoho and Iain M Johnstone, « Ideal spatial adaptation by wavelet shrinkage », *in*: *biometrika* 81.*3* (1994), pp. 425–455.

[41] Marie Douriez et al., « Anonymizing NYC Taxi Data: Does It Matter? », *in*: *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2016, pp. 140–148, DOI: 10.1109/DSAA.2016.21.

[42] John C Duchi, Michael I Jordan, and Martin J Wainwright, « Local privacy and statistical minimax rates », *in*: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, IEEE, 2013, pp. 429–438.

[43] Cynthia Dwork, « Differential Privacy », *in*: *Automata, Languages and Programming*, ed. by Michele Bugliesi et al., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12, ISBN: 978-3-540-35908-1.

[44] Cynthia Dwork and Aaron Roth, « The Algorithmic Foundations of Differential Privacy », *in*: *Foundations and Trends® in Theoretical Computer Science* 9.*3–4* (2014), pp. 211–407, ISSN: 1551-305X, DOI: 10.1561/0400000042, URL: http://dx.doi.org/10.1561/0400000042.

[45] Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan, « Boosting and differential privacy », *in*: *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS* (2010), pp. 51–60, ISSN: 02725428, DOI: 10.1109/FOCS.2010.12.

[46] Cynthia Dwork et al., « Calibrating Noise to Sensitivity in Private Data Analysis », *in*: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 3876 LNCS (2006), pp. 265–284, ISSN: 03029743, DOI: 10.1007/11681878_14, URL: https://link.springer.com/chapter/10.1007/11681878_14.

[47] Cynthia Dwork et al., « Our Data, Ourselves: Privacy Via Distributed Noise Generation », *in*: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 4004 LNCS (2006), pp. 486–503, ISSN: 16113349, DOI: 10.1007/11761679_29, URL: https://link.springer.com/chapter/10.1007/11761679_29.

[48] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova, « Rappor: Randomized aggregatable privacy-preserving ordinal response », *in*: *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, 2014, pp. 1054–1067.

[49] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson, « Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries », *in*: *Proceedings on Privacy Enhancing Technologies* 3 (2016), pp. 41–61.

[50] Vitaly Feldman et al., « Privacy amplification by iteration », *in*: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE, 2018, pp. 521–532.

[51] Miroslav Fiedler, « Algebraic connectivity of graphs », *in*: *Czechoslovak mathematical journal* 23.*2* (1973), pp. 298–305.

[52] Marco Fiore et al., « Privacy in trajectory micro-data publishing: a survey », *in*: *Transactions on Data Privacy* 13 (2020), pp. 91–149.

[53] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart, « Model inversion attacks that exploit confidence information and basic countermeasures », *in*: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[54] Guoji Fu et al., « Understanding Graph Neural Networks from Graph Signal Denoising Perspectives », *in*: *arXiv preprint arXiv:2006.04386* (2020).

[55] Andrea Gadotti et al., « Pool Inference Attacks on Local Differential Privacy: Quantifying the Privacy Guarantees of Apple's Count Mean Sketch in Practice », *in*: *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 501–518.

[56] Matan Gavish, Boaz Nadler, and Ronald R Coifman, « Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning », *in*: *ICML*, 2010.

[57] Benjamin Girault, Antonio Ortega, and Shrikanth S. Narayanan, « Irregularity-Aware Graph Fourier Transforms », *in*: *IEEE Transactions on Signal Processing* 66.*21* (2018), pp. 5746–5761, DOI: 10.1109/TSP.2018.2870386.

[58] Franziska Göbel, Gilles Blanchard, and Ulrike von Luxburg, « Construction of tight frames on graphs and application to denoising », *in*: *Handbook of big data analytics*, Springer, 2018, chap. 20, pp. 503–522.

[59] Google, *TensorFlow Privacy*, https://github.com/tensorflow/privacy.

[60] Leo J Grady and Jonathan R Polimeni, *Discrete calculus: Applied analysis on graphs for computational science*, vol. 3, Springer, 2010.

[61] Robert M Gray et al., « Toeplitz and circulant matrices: A review », *in*: *Foundations and Trends® in Communications and Information Theory* 2.*3* (2006), pp. 155–239.

[62] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart, « Exploring Network Structure, Dynamics, and Function using NetworkX », *in*: *Proceedings of the 7th Python in Science Conference*, ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman, Pasadena, CA USA, 2008, pp. 11–15.

[63] Rob Hall, Alessandro Rinaldo, and Larry Wasserman, « Differential privacy for functions and functional data », *in*: *The Journal of Machine Learning Research* 14.*1* (2013), pp. 703–727.

[64] Will Hamilton, Zhitao Ying, and Jure Leskovec, « Inductive representation learning on large graphs », *in*: *Advances in neural information processing systems*, 2017, pp. 1024–1034.

[65] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval, « Wavelets on graphs via spectral graph theory », *in*: *Applied and Computational Harmonic Analysis* 30.*2* (2011), pp. 129–150.

[66] Wolfgang Härdle et al., *Wavelets, approximation, and statistical applications*, vol. 129, Springer Science & Business Media, 2012.

[67] Michael Hay et al., « Accurate estimation of the degree distribution of private networks », *in*: *2009 Ninth IEEE International Conference on Data Mining*, IEEE, 2009, pp. 169–178.

[68] Mikael Henaff, Joan Bruna, and Yann LeCun, « Deep convolutional networks on graph-structured data », *in*: *NIPS*, 2015.

[69] Oscar Hernan Madrid Padilla et al., « The DFS Fused Lasso: Linear-Time Denoising over General Graphs », *in*: *Journal of Machine Learning Research* 18.*176* (2018), pp. 1–36.

[70] Roger A Horn and Charles R Johnson, *Matrix analysis*, Cambridge university press, 2012.

[71] M. F. Hutchinson, « A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines », *in*: *Communications in Statistics - Simulation and Computation* 19.*2* (1990), pp. 433–450, ISSN: 15324141, DOI: 10.1080/03610919008812866.

[72] Elvin Isufi et al., « Graph Filters for Signal Processing and Machine Learning on Graphs », *in*: *arXiv preprint arXiv:2211.08854* (2022).

[73] Laurent O. Jay et al., « Electronic structure calculations for plane-wave codes without diagonalization », *in*: *Computer Physics Communications* 118.*1* (1999), pp. 21–30, ISSN: 0010-4655, DOI: https://doi.org/10.1016/S0010-4655(98)00192-1, URL: https://www.sciencedirect.com/science/article/pii/S0010465598001921.

[74] Sadeep Jayasumana et al., « Kernel Methods on Riemannian Manifolds with Gaussian RBF Kernels », *in*: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.*12* (2015), pp. 2464–2477, DOI: 10.1109/TPAMI.2015.2414422.

[75] Tony Jebara, Jun Wang, and Shih-Fu Chang, « Graph Construction and B-Matching for Semi-Supervised Learning », *in*: *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, Montreal, Quebec, Canada: Association for Computing Machinery, 2009, pp. 441–448, ISBN: 9781605585161, DOI: 10.1145/1553374.1553432, URL: https://doi.org/10.1145/1553374.1553432.

[76] Shiva Prasad Kasiviswanathan et al., « Analyzing graphs with node differential privacy », *in*: *Theory of Cryptography: 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, Springer, 2013, pp. 457–476.

[77] Thomas N Kipf and Max Welling, « Semi-supervised classification with graph convolutional networks », *in*: *ICLR*, 2017.

[78] Luc Le Magoarou, Rémi Gribonval, and Nicolas Tremblay, « Approximate fast graph fourier transforms via multilayer sparse approximations », *in*: *IEEE transactions on Signal and Information Processing over Networks* 4.*2* (2017), pp. 407–420.

[79] Jaewoo Lee and Chris Clifton, « How much is enough? choosing ε for differential privacy », *in*: *International Conference on Information Security*, Springer, 2011, pp. 325–340, ISBN: 978-3-642-24861-0.

[80] Nora Leonardi and Dimitri Van De Ville, « Tight Wavelet Frames on Multislice Graphs », *in*: *IEEE Transactions on Signal Processing* 61.*13* (2013), pp. 3357–3367, DOI: 10.1109/TSP.2013.2259825.

[81] Nora Leonardi and Dimitri Van De Ville, « Tight wavelet frames on multislice graphs », *in*: *IEEE Transactions on Signal Processing* 61.*13* (2013), pp. 3357–3367.

[82] Jure Leskovec et al., « Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters », *in*: *Internet Mathematics* 6.*1* (2009), pp. 29–123, ISSN: 15427951, DOI: 10.1080/15427951.2009.10129177, arXiv: 0810.1355.

[83] Xiaoye Li et al., « Differential privacy for edge weights in social networks », *in*: *Security and Communication Networks* 2017 (2017).

[84] Yang Li et al., « Private graph data release: A survey », *in*: *ACM Computing Surveys* 55.*11* (2023), pp. 1–39.

[85] Xianming Liu et al., « Random walk graph Laplacian-based smoothness prior for soft decoding of JPEG images », *in*: *IEEE Transactions on Image Processing* 26.*2* (2016), pp. 509–524, DOI: 10.1109/TIP.2016.2627807.

[86] Basile de Loynes, Fabien Navarro, and Baptiste Olivier, « Data-driven thresholding in denoising with spectral graph wavelet transform », *in*: *Journal of Computational and Applied Mathematics* 389 (2021), p. 113319.

[87] Basile de Loynes, Fabien Navarro, and Baptiste Olivier, « Localized Fourier analysis for graph signal processing », *in*: *Applied and Computational Harmonic Analysis* 57 (2022), pp. 1–26.

[88] Keng-Shih Lu and Antonio Ortega, « Fast graph Fourier transforms based on graph symmetry and bipartition », *in*: *IEEE Transactions on Signal Processing* 67.*18* (2019), pp. 4855–4869.

[89] Mauro Maggioni et al., « Biorthogonal diffusion wavelets for multiscale representation on manifolds and graphs », *in*: *Wavelets XI*, vol. 5914, SPIE, 2005, pp. 543–555.

[90] Priya Mahadevan et al., « Systematic topology analysis and generation using degree correlations », *in*: *ACM SIGCOMM Computer Communication Review* 36.*4* (2006), pp. 135–146.

[91] Stéphane Mallat, *A wavelet tour of signal processing*, Elsevier, 1999.

[92] Stephane G Mallat, « A theory for multiresolution signal decomposition: the wavelet representation », *in*: *IEEE transactions on pattern analysis and machine intelligence* 11.*7* (1989), pp. 674–693.

[93] Frank McSherry, *How many secrets do you have?*, URL: https://github.com/frankmcsherry/blog/blob/master/posts/2017-02-08.md.

[94] Frank McSherry, « Privacy integrated queries: An extensible platform for privacy-preserving data analysis », *in*: *SIGMOD-PODS'09 - Proceedings of the International Conference on Management of Data and 28th Symposium on Principles of Database Systems* (2009), pp. 19–30, DOI: 10.1145/1559845.1559850.

[95] Ilya Mironov, « Rényi Differential Privacy », *in*: *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, 2017, pp. 263–275, DOI: 10.1109/CSF.2017.11.

[96] Sunil K Narang and Antonio Ortega, « Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs », *in*: *IEEE transactions on signal processing* 61.*19* (2013), pp. 4673–4685.

[97] Sunil K Narang and Antonio Ortega, « Lifting based wavelet transforms on graphs », *in*: *Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference*, Asia-Pacific Signal and Information Processing Association, 2009 Annual . . ., 2009, pp. 441–444.

[98] Sunil K Narang and Antonio Ortega, « Perfect reconstruction two-channel wavelet filter banks for graph structured data », *in*: *IEEE Transactions on Signal Processing* 60.*6* (2012), pp. 2786–2799.

[99] Maximilian Nickel et al., « A review of relational machine learning for knowledge graphs », *in*: *Proceedings of the IEEE* 104.*1* (2015), pp. 11–33.

[100] Masaki Onuki et al., « Graph Signal Denoising via Trilateral Filter on Graph Spectral Domain », *in*: *IEEE Transactions on Signal and Information Processing over Networks* 2.*2* (2016), pp. 137–148, DOI: 10.1109/TSIPN.2016.2532464.

[101] Orange, *GPSGhoster*, https://github.com/Orange-OpenSource/GPSGhoster.

[102] Antonio Ortega et al., « Graph Signal Processing: Overview, Challenges, and Applications », *in*: *Proceedings of the IEEE* 106.*5* (2018), pp. 808–828, DOI: 10.1109/JPROC.2018.2820126.

[103] Antonio Ortega et al., « Graph signal processing: Overview, challenges, and applications », *in*: *Proceedings of the IEEE* 106.*5* (2018), pp. 808–828.

[104] George M Phillips, *Interpolation and approximation by polynomials*, vol. 14, Springer Science & Business Media, 2003.

[105] Michal Piórkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser, *CRAWDAD dataset epfl/mobility (v. 2009-02-24)*, Downloaded from https://crawdad.org/epfl/mobility/20090224, Feb. 2009, DOI: 10.15783/C7J010.

[106] Markus Puschel and José MF Moura, « Algebraic signal processing theory: Foundation and 1-D time », *in*: *IEEE Transactions on Signal Processing* 56.*8* (2008), pp. 3572–3585.

[107] Sathish Ramani, Thierry Blu, and Michael Unser, « Monte-Carlo Sure: A Black-Box Optimization of Regularization Parameters for General Denoising Algorithms », *in*: *IEEE Transactions on Image Processing* 17.*9* (2008), pp. 1540–1554, DOI: 10.1109/TIP.2008.2001404.

[108] Sofya Raskhodnikova and Adam Smith, « Lipschitz extensions for node-private graph statistics and the generalized exponential mechanism », *in*: *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE, 2016, pp. 495–504.

[109] Alfréd Rényi et al., « On measures of entropy and information », *in*: *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, vol. 1, 547-561, Berkeley, California, USA, 1961.

[110] Alessandra Sala et al., « Sharing graphs using differentially private graph models », *in*: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, 2011, pp. 81–98.

[111] Aliaksei Sandryhaila and José MF Moura, « Discrete signal processing on graphs », *in*: *IEEE transactions on signal processing* 61.*7* (2013), pp. 1644–1656.

[112] Aliaksei Sandryhaila and Jose MF Moura, « Discrete signal processing on graphs: Frequency analysis », *in*: *IEEE Transactions on Signal Processing* 62.*12* (2014), pp. 3042–3054.

[113] Godwin Shen and Antonio Ortega, « Transform-based distributed data gathering », *in*: *IEEE Transactions on Signal Processing* 58.*7* (2010), pp. 3802–3815.

[114] David I Shuman, « Localized spectral graph filter frames: A unifying framework, survey of design considerations, and numerical comparison », *in*: *IEEE Signal Processing Magazine* 37.*6* (2020), pp. 43–63.

[115] David I Shuman et al., « The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains », *in*: *IEEE Signal Processing Magazine* 30.*3* (2013), pp. 83–98, DOI: 10.1109/MSP.2012.2235192.

[116] Amit Singer, « From graph to manifold Laplacian: The convergence rate », *in*: *Applied and Computational Harmonic Analysis* 21.*1* (2006), Special Issue: Diffusion Maps and Wavelets, pp. 128–134, ISSN: 1063-5203, DOI: https://doi.org/10.

1016/j.acha.2006.03.004, URL: https://www.sciencedirect.com/science/article/pii/S1063520306000510.

[117] Adam Smith, « Efficient, differentially private point estimators », *in*: *arXiv preprint arXiv:0809.4794* (2008).

[118] Adam Smith, « Privacy-preserving statistical estimation with optimal convergence rates », *in*: *Proceedings of the forty-third annual ACM symposium on Theory of computing*, 2011, pp. 813–822.

[119] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate, « Stochastic gradient descent with differentially private updates », *in*: *2013 IEEE global conference on signal and information processing*, IEEE, 2013, pp. 245–248.

[120] Daniel Spielman, « Spectral Graph Theory », *in*: *Combinatorial Scientific Computing*, Chapman and Hall / CRC Press, 2012, chap. 16.

[121] Charles M Stein, « Estimation of the mean of a multivariate normal distribution », *in*: *The annals of Statistics* (1981), pp. 1135–1151.

[122] Jun Tang et al., « Privacy loss in apple's implementation of differential privacy on macos 10.12 », *in*: *arXiv preprint arXiv:1709.02753* (2017).

[123] Differential Privacy Team, *Learning with privacy at scale*, tech. rep., Apple, 2017.

[124] Abhradeep Guha Thakurta et al., « Learning new words », *in*: *Granted US Patents* 9594741 (2017), p. 2.

[125] Robert Tibshirani et al., « Sparsity and smoothness via the fused lasso », *in*: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.*1* (2005), pp. 91–108, DOI: https://doi.org/10.1111/j.1467-9868.2005.00490.x, eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2005.00490.x, URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2005.00490.x.

[126] Daniel Ting, Ling Huang, and Michael I. Jordan, « An Analysis of the Convergence of Graph Laplacians », *in*: *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, Haifa, Israel: Omnipress, 2010, pp. 1079–1086, ISBN: 9781605589077.

[127] Nicolas Tremblay and Pierre Borgnat, « Graph wavelets for multiscale community mining », *in*: *IEEE Transactions on Signal Processing* 62.*20* (2014), pp. 5227–5239.

[128] Martin Vetterli and Cormac Herley, « Wavelets and filter banks: Theory and design », *in*: *IEEE transactions on signal processing* 40.*ARTICLE* (1992), pp. 2207–2232.

[129] Ulrike Von Luxburg, « A tutorial on spectral clustering », *in*: *Statistics and computing* 17.*4* (2007), pp. 395–416.

[130] John Von Neumann, « Distribution of the ratio of the mean square successive difference to the variance », *in*: *The Annals of Mathematical Statistics* 12.*4* (1941), pp. 367–395.

[131] Wei Wang and Kannan Ramchandran, « Random multiresolution representations for arbitrary sensor network graphs », *in*: *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 4, IEEE, 2006, pp. IV–IV.

[132] Yu-Xiang Wang et al., « Trend Filtering on Graphs », *in*: *Journal of Machine Learning Research* 17 (2016), pp. 1–41.

[133] Yue Wang, Xintao Wu, and Donghui Hu, « Using Randomized Response for Differential Privacy Preserving Data Collection. », *in*: *EDBT/ICDT Workshops*, vol. 1558, 2016, pp. 0090–6778.

[134] Yue Wang, Xintao Wu, and Leting Wu, « Differential privacy preserving spectral graph analysis », *in*: *Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part II 17*, Springer, 2013, pp. 329–340.

[135] Stanley L. Warner, « Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias », *in*: *Journal of the American Statistical Association* 60.*309* (1965), pp. 63–69, ISSN: 1537274X, DOI: 10.1080/01621459.1965.10480775.

[136] Atsushi Waseda and Ryo Nojima, « Analyzing Randomized Response Mechanisms Under Differential Privacy », *in*: *Information Security*, ed. by Matt Bishop and Anderson C A Nascimento, Cham: Springer International Publishing, 2016, pp. 271–282, ISBN: 978-3-319-45871-7.

[137] Larry Wasserman and Shuheng Zhou, « A statistical framework for differential privacy », *in*: *Journal of the American Statistical Association* 105.*489* (2010), pp. 375–389.

[138] Daniel S. Weller et al., « Monte Carlo SURE-based parameter selection for parallel magnetic resonance imaging reconstruction », *in*: *Magnetic Resonance in Medicine* 71.*5* (2014), pp. 1760–1770, DOI: https://doi.org/10.1002/mrm.24840, eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mrm.24840, URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.24840.

[139] Ashkan Yousefpour et al., « Opacus: User-friendly differential privacy library in PyTorch », *in*: *arXiv preprint arXiv:2109.12298* (2021).

[140] Xiaofan Zhu and Michael Rabbat, « Approximating signals supported on graphs », *in*: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2012, pp. 3921–3924.

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

ENSAI École nationale de la statistique et de l'analyse de l'information

Titre : Débruitage de signaux définis sur des graphes de grande taille avec application à la confidentialité différentielle

Mots clés : Ondelettes sur graphe, réduction du bruit, confidentialité différentielle

Résumé : Au cours de la dernière décennie, le traitement du signal sur graphe est devenu un domaine de recherche très actif. Plus précisément, le nombre d'applications utilisant des repères construits à partir de graphes, tels que les ondelettes sur graphe, a augmenté de manière significative. Nous considérons en particulier le débruitage de signaux sur graphes au moyen d'une décomposition dans un repère ajusté d'ondelettes. Cette approche est basée sur le seuillage des coefficients d'ondelettes à l'aide de l'estimateur sans biais du risque de Stein (SURE). Nous étendons cette méthodologie aux graphes de grande taille en utilisant l'approximation par polynômes de Chebyshev qui permet d'éviter la décomposition de la matrice laplacienne du graphe. La principale difficulté est le calcul de poids dans l'expression du SURE faisant apparaître un terme de covariance en raison de la nature surcomplète du repère d'ondelettes. Le calcul et le stockage de celui-ci est donc nécessaire et rédhibitoire à grande échelle. Pour estimer cette covariance, nous développons et analysons un estimateur de Monte-Carlo reposant sur la transformation rapide de signaux aléatoires. Cette nouvelle méthode de débruitage trouve une application naturelle en confidentialité différentielle dont l'objectif est de protéger les données sensibles utilisées par les algorithmes. Une évaluation expérimentale de ses performances est réalisée sur des graphes de taille variable à partir de données réelles et simulées.

Title: Large graph signal denoising with application to differential privacy

Keywords: Graph wavelets, noise reduction, differential privacy

Abstract: Over the last decade, signal processing on graphs has become a very active area of research. Specifically, the number of applications using frames built from graphs, such as wavelets on graphs, has increased significantly. We consider in particular signal denoising on graphs via a wavelet tight frame decomposition. This approach is based on the thresholding of the wavelet coefficients using Stein's unbiased risk estimate (SURE). We extend this methodology to large graphs using Chebyshev polynomial approximation, which avoids the decomposition of the graph Laplacian matrix. The main limitation is the computation of weights in the SURE expression, which includes a covariance term due to the overcomplete nature of the wavelet frame. The computation and storage of the latter is therefore necessary and impractical for large graphs. To estimate such covariance, we develop and analyze a Monte Carlo estimator based on the fast transform of random signals. This new denoising methodology finds a natural application in differential privacy whose purpose is to protect sensitive data used by algorithms. An experimental evaluation of its performance is carried out on graphs of varying size, using real and simulated data.