Numerical algorithms for the Mathematics of Information

Rodrigo Mendoza-Smith

St. Anne's College

University of Oxford

A thesis submitted for the examination of

Doctor of Philosophy

Trinity 2017

Statement of Originality

I hereby declare that this thesis was composed by myself and that the work contained therein is entirely my own, except where specific reference is made to the work of others. This dissertation contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Rodrigo Mendoza-Smith
September 2017

210882932276083329729217327639324961047155779175760663347052181148979 2

Acknowledgements

My deepest gratitude goes to my supervisor Jared Tanner for directing this research and for helping me find a place in the research community. Jared's mentorship and guidance were invaluable in a mathematical school of thought that is characterised by its interdisciplinarity.

I would also like to thank Dror Baron, Coralia Cartis, Pier Luigi Dragotti, Heather Harrington, Venkatesan Guruswami, Henry Pfister and other anonymous reviewers for their helpful comments and suggestions on the papers that compose this thesis.

Additionally, I would like to acknowledge Florain Wechsung with whom I worked on the Robust-ℓ_0 algorithm described in Chapter 4.

Thanks also to my examiners Robert Calderbank, Vidit Nanda and Andrew Thompson for the time they spent reading this thesis.

Finally, I would like to acknowledge the National Science and Technology Council in Mexico (CONACyT) for supporting my research through their PhD fellowship, the Mathematical Institute for their support during the first two years of my DPhil, and the Alan Turing Institute for their support during my fourth year.

Abstract

This thesis presents a series of algorithmic innovations in Combinatorial Compressed Sensing and Persistent Homology. The unifying strategy across our contributions is in translating structural patterns in the underlying data into specific algorithmic designs in order to achieve: better guarantees in computational complexity, the ability to operate on more complex data, highly efficient parallelisations, or any combination of these. Our main contributions can be broken down as follows.

1. Chapter 3 extends the work of [Berinde et al., 2008b, Jafarpour et al., 2009, Sarvotham et al., 2006b] and propose two algorithms *Parallel-ℓ_0* and *Serial-ℓ_0* for decoding a k-sparse signals $x \in \mathbb{R}^n$ from an underdetermined linear system $Ax = y \in \mathbb{R}^m$ where $A \in \mathbb{R}^{m \times n}$ is the adjacency matrix of a left d-regular expander graph. We guarantee recovery in $\mathcal{O}(n \log k)$ operations by assuming a *dissociated structure* in the non-zeros of the signal x or by imposing it in the columns of A. Our algorithms achieve phase transitions at $\rho \approx 0.3$, which are superior than those of ℓ_1-regularisation for small under-sampling ratios $m/n \in (0, 1)$.

2. Chapter 4 borrows from [Ma et al., 2013] to make *Parallel-ℓ_0* robust to additive noise. This extension is made by quantifying the uncertainty in the residual analytically via the Bayesian framework. These approximations hold for arbitrary noise and signal distributions, and only assume that the entries in η and x are sampled component-wise and are independent of each other. Computational results show that the algorithm is robust for signal-to-noise ratios of 10^{-3} and 10^{-2}.

3. Chapter 5 introduces persistent homology, a technique within *topological data analysis* with which the topological invariants of a manifold $\mathcal{M} \subset \mathbb{R}^n$ can be estimated from a finite dataset $S \subset \mathbb{R}^n$ sampled from $\mathcal{M}$. A crucial step in the persistent homology pipeline is the factorisation of a sparse upper-triangular matrix $\partial \in \mathbb{F}_2^{m \times m}$ over the Galois field of two elements. This factorisation can be made in time $\mathcal{O}(m^3)$ and [Morozov, 2005] gave an example for which this worst-case is achieved. In Section 5.5 we show that this worst-case example is

so structured that it can be trivially reduced in $\mathcal{O}(m)$ operations by a counting argument. This gives evidence that worst-case complexes are so structured that they do not represent the structure of *average* simplicial complexes that are found in nature.

4. In Chapter 6 we extend the work of [Bauer et al., 2014a, Chen and Kerber, 2011] and show that there is structure in ∂ that one can use to massively parallelise the factorisation of ∂ and that one can reliable estimate some important topological information in the data.

	Contribution	Submission	Collaborators
1	Chapter 3	[Mendoza-Smith and Tanner, 2017a]	Jared Tanner
2	Chapter 4	[Mendoza-Smith et al., 2017]	Jared Tanner and Florian Wechsung
3	Section 5.5	To be submitted for publication	Jared Tanner
4	Chapter 6	[Mendoza-Smith and Tanner, 2017b]	Jared Tanner

Table 1: Contributions

Contents

1 The Mathematics of Information **1**

1.1 Data complexity 1

1.2 Numerical algorithms in the Mathematics of Information 4

1.2.1 Compressed Sensing 5

1.2.2 Topological Data Analysis 6

1.3 A roadmap to this thesis 6

1.4 Notation 7

2 Combinatorial Compressed Sensing **9**

2.1 From Shannon's sampling theorem to Compressed Sensing 9

2.2 Iterative greedy algorithms 15

2.3 Measuring with sparse and structured operators 18

2.3.1 Expander sketching and ℓ_1-uncertainty principles 18

2.3.2 Graph theory and Expander graphs 20

2.3.3 Sparse Matching Pursuit (SMP) 24

2.3.4 Sequential Sparse Matching Pursuit (SSMP) 24

2.3.5 Left Degree Dependent Signal Recovery (LDDSR) 27

2.3.6 Expander Recovery (ER) 28

2.4 The dissociated model 29

2.4.1 Preservation of Entropy 30

2.4.2 Dissociated signals in previous art 31

2.4.3 LDDSR/ER and the dissociated condition 31

3 Expander ℓ_0-decoding **33**

3.1 Convergence of Expander ℓ_0-decoders 35

3.1.1 Shifted Parallel-ℓ_0 and Parallel-LDDSR 41

3.2 Numerical Experiments 42

3.2.1 Substantially higher phase transitions 42

3.2.2 Approximately flat phase transition 43
3.2.3 Fastest compressed sensing algorithm 45
3.2.4 Parallelisation brings important speedups: examples with $m \ll n$ 45
3.2.5 Convergence in $\mathcal{O}(\log k)$ iterations 47
3.2.6 Almost dissociated signals 48
3.2.7 d should be small, but not too small 50

4 Bayesian ℓ_0-decoding **53**
4.1 The Bayesian approach to ℓ_0-decoding 58
4.1.1 Explicit formulas for the centred Gaussian case 64
4.1.1.1 Estimating the tails in the Gaussian case 64
4.1.2 Comparison with empirical probabilities 65
4.1.3 Scaled probabilities $\breve{p}_z$ and $\breve{p}_e$ 67
4.1.4 Adaptive k 67
4.2 Numerical experiments 67
4.2.1 Stopping conditions 69
4.2.2 Selection of parameter c in Algorithm 15 69
4.2.3 Phase transitions and runtime 70
4.2.4 Dependence on noise variance, σ 74
4.2.5 Phase transitions for extreme subsampling, $\delta \ll 1$ 74

5 Persistent Homology **77**
5.1 Simplicial Homology 78
5.2 Persistent Homology 80
5.3 Boundary matrix reduction 82
5.4 Boundary reduction algorithms 85
5.5 Reducing Morozov's complex in $\mathcal{O}(m)$ operations 87

6 Parallel multi-scale reduction **93**
6.1 Theoretical preamble 94
6.2 Parallel multi-scale reduction 99
6.2.1 Phase 0: Initialising Pivots 99
6.2.2 Phase I: Finding local injections 100
6.2.3 Main iteration II: Parallel column reduction 100
6.2.4 Convergence 101
6.3 Clearing by compression 102
6.4 Essential estimation 103

6.5 Distributing the workload 104
6.5.1 Prioritise reduction of sets $\mathcal{N}(j)$ with large cardinality 104
6.5.2 Prioritise reduction of $\mathcal{N}(j) \cap \mathsf{Neg}$ 104
6.6 Numerical experiments 105
6.6.1 Operations are packed in fewer iterations 106
6.6.2 low^* is approximated at multiple scales 107
6.6.3 The set Ess can be reliably estimated after a few iterations . . 109

7 Conclusions and future work **113**
7.1 Future work and open questions 114

Bibliography **115**

List of Figures

1.1 A roadmap to this thesis 7

2.1 Schematic of expander graph 21

3.1 Phase Transitions for CCS algorithms 43
3.2 Mean time to exact convergence for Parallel-ℓ_0 44
3.3 Time (ms) of fastest algorithm 46
3.4 Algorithm selection map 46
3.5 Mean time to exact convergence for $\delta = 0.01$ and $n = 2^{20}$ 47
3.6 Mean time to exact convergence for $\delta = 0.1$ and $n = 2^{20}$ 48
3.7 Iterations vs ρ for sublinear algorithms. 49
3.8 Iterations vs ρ for linear algorithms. 49
3.9 Parallel-ℓ_0 with signal having a fixed proportion of identical nonzero elements in its support 50
3.10 Phase transitions of Parallel-ℓ_0 for several values of d 51

4.1 Generating model for noisy measurements 56
4.2 Comparison of empirical and analytical probabilities 66
4.3 Phase transitions, selection maps and timings for $n = 2^{18}$ 68
4.4 Phase transition widths for Robust-ℓ_0 variants. 72
4.5 Decrease in phase transition for varying σ. 73
4.6 Timing performance for low δ 75

5.1 Sparsity pattern of boundary matrices 84
5.2 Sparsity pattern for Morozov boundary matrices 88
5.3 ρ-curves for several complexes 91

6.1 Benchmarking on column operation overhead 108
6.2 Relative ℓ_1-error per iteration 109
6.3 Proportion of unreduced columns per iteration 110

6.4 Precision of Ess estimation 112

Chapter 1

The Mathematics of Information

This document reports a series of theoretical and computational results that concern problems in the *Mathematics of Information*. This term refers to the set of mathematical theories that study the methodologies to extract, measure, and represent the information present in data [Iserles and Schönlieb, 2013]. These theories have become increasingly relevant given the recent expansion of technologies for large-scale data processing and the shift of scientific and industrial research towards more heavily computerised data-based domains. Despite their clear purpose, the boundaries of the Mathematics of Information are not yet well-defined. However, they can be chartered with respect to its areas of application such as Data Science, Machine Learning, Signal Processing and Artificial Intelligence; or indeed, in terms of its most represented mathematical theories like Optimisation, Network Theory, Probability, Statistics, Functional Analysis, Approximation Theory and Numerical Analysis.

A defining characteristic of the theories in the Mathematics of Information is that they showcase the interplay between *data complexity* and *computational complexity*. Indeed, many problems in this area are intractable in the general case, so we seek to build mathematical models around the data in order to capture its structure and bring the problem's complexity down to the level of the efficiently computable. In the following sections we make these notions precise.

1.1 Data complexity

Almost all datasets of interest to humankind like audio signals [Oppenheim, 1999], networks [Newman, 2010], images and video[Forsyth and Ponce, 2011], documents [Manning et al., 1999] or genomic data [Cristianini and Hahn, 2006] can be represented geometrically either as a vector $v \in \mathbb{R}^n$, a matrix $A \in \mathbb{R}^{m \times n}$, or a tensor $T \in \mathbb{R}^{n_1 \times \cdots \times n_p}$.

Representing data as an abstract mathematical object is useful since it allows us to quantify its complexity and create mathematical models around it. The complexity of a dataset can be modelled or quantified in several ways.

I *Dimensionality:* The most basic way to measure the complexity of a dataset is through the dimensionality of the space it is embedded in. This is an important measure of complexity due to a phenomenon known as the *curse of dimensionality* [Bellman and Kalaba, 1959] which refers to the fact that the volumes in Euclidean space grow exponentially with the dimension of the space. The curse of dimensionality is an important shortcoming to Machine Learning [Friedman et al., 2001], Numerical Analysis [Trefethen, 2017] and other areas that aim to approximate functions from a limited number of datapoints.

However, arbitrarily large dimensionality is not always a burden and often becomes a rather useful assumption thanks to a dual phenomenon, which is often refered to as *concentration of measure* [Talagrand, 1996]. In the words of Tao [Tao, 2010] this phenomenon boils down to the observation that:

It is difficult for a large number of scalar random variables $\{X_1, \ldots, X_n\}$ *to work together to pull a general combination* $F(X_1, \ldots, X_n)$ *too far away from its mean.*

The concentration of measure phenomenon has given rise to some of the most exciting mathematical technologies in recent years as it provides new avenues to reduce the dimensionality of high-dimensional data without distorting its geometry. This is best exemplified by the Johnson-Lindenstrauss Lemma [Johnson and Lindenstrauss, 1984], which guarantees that if $\{x_1, \ldots, x_m\} \subset \mathbb{R}^n$ and $\varepsilon \in (0, 1)$, then there is a function $f : \mathbb{R}^n \to \mathbb{R}^{\mathcal{O}(\log(m)/\varepsilon^2)}$ such that $\forall\ i \neq j$

$$(1 - \varepsilon)\|x_i - x_j\| \leq \|f(x_i) - f(x_j)\| \leq (1 + \varepsilon)\|x_i - x_j\|.$$

II *Degrees of freedom:* The *manifold hypothesis* states that high-dimensional data tends to concentrate in the vicinity of a low-dimensional manifold [Fefferman et al., 2016]. Though there is no rigorous way to test the validity of the manifold hypothesis, there are a number of results that provide positive evidence on its favour, such as the concentration of measure phenomenon mentioned above or the recent achievements in *deep-learning* research [Krizhevsky et al., 2012].

One of the clearest models of data complexity in this scenario is given by the Grassmann manifold

$$\mathcal{G}_r^{m\times n} = \left\{M \in \mathbb{R}^{m\times n} : \operatorname{rank}(M) = r\right\}.$$

The Grassmann manifold has dimension $(m+n-r)r$ [Lee, 2003] and is a popular model of data complexity in the Mathematics of Information. This model of data complexity might be counter-intuitive as it is known that if $A \in \mathbb{R}^{m\times n}$ is sampled according to a continuous distribution then $\Pr[\operatorname{rank}(A) = n] = 1$ since any polynomial on such variables is non-zero with probability one. The apparent *abundance* of full-rank matrices is reconciled with the manifold hypothesis by assuming that many of the matrices we observe in nature are indeed low-rank matrices that have been perturbed to be full-rank. For example, [Candès et al., 2011] proposed a model in which matrices $A \in \mathbb{R}^{m\times n}$ are a superposition of a low-rank matrix L that captures the essential information in the matrix, and a sparse matrix S that acts as a perturbation to L.

III *Probabilistic:* A popular way to define structure in the data is by modelling the probability mechanisms that generate it. Examples of this include Probabilistic Graphical Models [Koller and Friedman, 2009] and Gaussian Processes [Rasmussen and Williams, 2006]. A much simpler example is given by linear regression, where parameters $x \in \mathbb{R}^n$ of a hyperplane $f(a, z) = z - \langle x, a\rangle$ are sought to be recovered from a set of corrupted observations $\{(a_1, y_1), \ldots, (a_m, y_m)\}$. This is modelled by the system

$$y = x^T A + \varepsilon \in \mathbb{R}^m \tag{1.1}$$

for $A \in \mathbb{R}^{n\times m}$ and a noise vector $\varepsilon \in \mathbb{R}^m$. When $m < n$ the problem is underdetermined so infinitely many solutions exist. In order to guarantee solvability, we need to introduce additional information in the problem which is a process known as *regularisation*. A principled way of regularising is by assuming that x is sampled according to a probability distribution F_x. Having F_x Gaussian corresponds to a penalisation on the ℓ_2-magnitude of x and is commonly known as *Tikhonov regularisation* [Tikhonov, 1963], while having F_x being Laplace corresponds to penalising its ℓ_1-norm of x and is generically called *Lasso regularisation* Tibshirani [1996].

IV *Topological complexity:* In the field of *Topological Data Analysis* it is assumed that datasets are sampled from a high-dimensional shape $\mathcal{M} \subset \mathbb{R}^n$ embedded in

$\mathbb{R}^n$. The field of algebraic topology classifies shapes according to their number of *high-dimensional holes*, which is a quantity that is invariant to continuous deformation of the shape $\mathcal{M}$. In this sense, the number of high-dimensional holes becomes a notion of data complexity for the dataset in general. For example, [Carlsson, 2009] performed an experiment in which he showed that a large set of images in $\mathbb{R}^{3\times 3}$ lies approximately on a Klein bottle. In this sense, an alternate measure of complexity in the context of a manifold is through *homology* or by estimating the number of high-dimensional holes in the manifold $\mathcal{M}$.

V *Other:* This list is of course not exhaustive and there are several other ways to quantify the complexity of a dataset or indeed, of points within a dataset. For example, for a matrix the *sparsity* measures its number of non-zero entries, while the *rigidity* [Valiant, 1977] encodes the minimum Hamming distance between the matrix and other matrices of rank at most r.

1.2 Numerical algorithms in the Mathematics of Information

In order to extract the essential information from the dataset under consideration it is necessary to perform numerical computations over the data. For instance classifying points into classes can be done via a *support vector machine* which calls for a quadratic programming problem [Boyd and Vandenberghe, 2004]. The directions of maximum variance in a dataset are computed through *Principal Component Analysis* [Wold et al., 1987], which involves eigenvalue algorithms like the power iteration method. The state-of-the-art optimisation methods for training neural network are based on stochastic gradient descent [Bottou, 2010].

This thesis is concerned with the numerical underpinnings of problems in the Mathematics of Information that can be framed as *inverse problems*. Namely, problems in which the causal factor $\theta \in \Omega$ of a mathematical model $M(\theta)$ is sought to be estimated from a dataset $\mathcal{D} = \{y_1, \ldots, y_m\}$ generated from $M(\theta)$. Inverse problems are challenging because often the number of data-points in $\mathcal{D}$ is not enough to single out an element of the parameter space Ω. In [Hadamard, 1902] Hadamard defined the mathematical properties of models that have enough structure go guarantee physical meaning of the solution.

Definition 1.1 (Well-posed problem [Hadamard, 1902])**.** *A problem* $\mathcal{D} \leftarrow M(\theta)$ *is* well-posed *if*

1. *A solution $\theta \in \Omega$ exists.*

2. *The solution $\theta \in \Omega$ is unique.*

3. *Small perturbations in θ cause small perturbations in $\mathcal{D}$.*

A common problem in the Mathematics of Information is dealing with underdetermined problems or problems in which there is not enough data to uniquely specify the nature of the underlying model. When models are indistinguishible, one seeks to introduce additional information in the system that is consistent to *prior* beliefs on the nature of the generating system. The Tikhonov and Lasso regularisation methods mentioned above are examples of these priors. The definition given by Hadamard was tailored to describe the *ill-posedness* of the physics and engineering problems that were relevant at his time. More than one hundred years later, it becomes clear that it is not enough to guaranteeing existence and uniqueness of solutions, but also that they are *computable*. Hence, a sensible appendix to Hadamard's definition of well-posed problem could be:

4. *The solution $\theta \subset \Omega$ is efficiently computable (with non-trivial probability).*

This thesis deals with two new types of inverse problems.

1.2.1 Compressed Sensing

For a vector $x \in \mathbb{R}^n$, let $\|x\|_0$ be the number of non-zeros in x. For $k < n$ let χ_k^n be the set of k-sparse, *i.e.*

$$\chi_k^n = \{x \in \mathbb{R}^n : \|x\|_0 \leq k\}. \tag{1.2}$$

Compressed sensing seeks to recover a sparse signal $x \in \chi_k^n$ from $m < n$ linear non-adaptive measurements $y = Ax \in \mathbb{R}^m$. This is an inverse problem since one seeks to find x from m pieces of data generated by inner products of x with m rows of a matrix $A \in \mathbb{R}^{m \times n}$. Introducing the prior information on the sparsity of x we arrive at the regularised problem

$$\min_x \|y - Ax\|_2 \quad \text{s.t.} \quad \|x\|_0 \leq k. \tag{1.3}$$

The solution of (1.3) is not efficiently computable in the general case (see Chapter 2). The compressed sensing paradigm imposes a model on the matrix $A \in \mathbb{R}^{m \times n}$ to show that in most cases the problem is well-posed according to our extended definition.

1.2.2 Topological Data Analysis

Topological data analysis (TDA) addresses the algebraic-topological analogue of inverse problems. In Topological Data Analysis one wishes to recover the topological invariants that characterise a deformable high-dimensional *shape* $\mathcal{M} \subset \mathbb{R}^n$ from a dataset $\mathcal{D} = \{x_1, \ldots, x_m\}$ sampled from $\mathcal{M}$. These *topological invariants* are those topological properties of $\mathcal{M}$ that are preserved through continuous deformations, like connectedness, number of loops, and number of high-dimensional voids [Edelsbrunner and Harer, 2010] and are encoded as a set of *Betti numbers*, see Chapter 5. The persistent homology paradigm which is of high relevance to our work provides a framework for estimating the underlying topological signature of the shape $\mathcal{M}$ from a $\mathcal{D}$.

1.3 A roadmap to this thesis

This thesis presents advances in two distinctively different areas of computational mathematics. Specifically,

This chapter provides an introduction to the Mathematics of Information and introduces the notational conventions of the thesis.

Chapter 2 introduces compressed-sensing, iterative greedy algorithms, and the graph-theoretic notions that are at the backbone of combinatorial compressed sensing. Additionally, it introduces the notion of dissociated data which is crucial for our developments in Chapters 3 and 4.

Chapter 3 gives our first contribution: Parallel-ℓ_0 and Serial-ℓ_0, which are the two fastest algorithms in compressed sensing when the data is dissociated and the adjacency matrix is an expander matrix. This chapter includes theoretical guarantees for these algorithms and a series of numerical experiments that benchmark them against the previous state-of-the-art.

Chapter 4 extends Parallel-ℓ_0 to scenarios where the signal is corrupted with additive noise. The extension is made by quantifying the uncertainty of the measurements via the Bayesian framework. This chapter includes a framework that work for arbitrary signals and noise distributions as long as they are continuous. Appropriate numerical experiments are also shown.

Chapter 5 shifts the theme to Computational Algebraic Topology and specifically to the computation of Persistent Homology for Topological Data Analysis. This

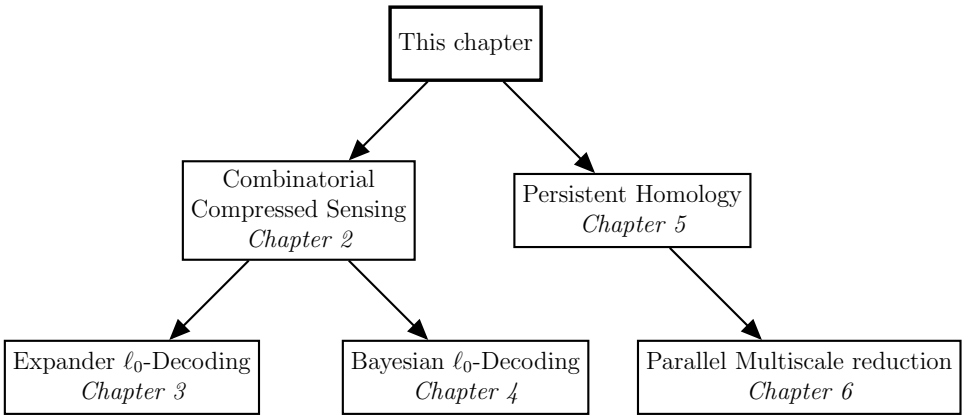


Figure 1.1: **A roadmap to this thesis.** A dependency diagram between each chapter is shown.

chapter introduces the necessary ideas from topology and algebra to build the Persistent Homology pipeline and introduces previous algorithms for the reduction of boundary matrices, which is an essential step when computing persistent homology. The chapter includes a proof-of-concept showing that the data on which the standard algorithms for computing persistent homology achieve the worst-case is extremely structured and not representative of most datasets.

Chapter 6 presents an algorithm to massively parallelise the computation of persistent homology in such a way that reduction can be done at all scales of the relevant filtration and such that the algorithm can be interrupted and yield reliable approximations to the persistent topology of the dataset.

Chapter 7 outlines our conclusions and summarises attractive directions for future work.

To facilitate the exploration of our work, we provide a roadmap to describe the dependencies between the chapters in our manuscript.

1.4 Notation

Here we define the basic notation that is relevant to our discussion. For a subset $S \subset \Omega$, we let $|S|$ be its cardinality, and $\Omega \setminus S$ denote its complement. We adopt notation from combinatorics and for $n \in \mathbb{N}$ use the shorthands $[n] := \{1, \dots, n\}$, $[n]^{(k)} = \{S \subset [n] : |S| = k\}$, and $[n]^{(\leq k)} = \{S \subset [n] : |S| \leq k\}$. As mentioned in the previous section, for $x \in \mathbb{R}^n$, we let $\operatorname{supp}(x) = \{i : x_i \neq 0\}$ be its support, and $\operatorname{argsupp}(x) = \{x_i : i \in \operatorname{supp}(x)\}$ be the set of nonzero values in x. With this, we let

$\|x\|_0 := |\operatorname{supp}(x)|$, and $\chi_k^n = \{x \in \mathbb{R}^n : \|x\|_0 \leq k\}$ be the set of k-sparse vectors in $\mathbb{R}^n$. A number of compressed sensing algorithms project vectors from $\mathbb{R}^n$ to χ_k^n via the *hard-thresholding* function $H_k : \mathbb{R}^n \to \chi_k^n$ which sets to zero all but the k largest elements in x in magnitude. We also let $\mathbb{D}(\mathbb{R})$ be the set of distributions supported on $\mathbb{R}$. If $\mu \in \mathbb{D}(\mathbb{R})$, we use the notation $z \sim \mu$ to denote that z was drawn according to the distribution μ. We also use the notation $v_i \overset{\text{i.i.d.}}{\sim} \mu$ to denote that each v_i is drawn independently at random from μ, and $U(S)$ to denote the uniform distribution over a set S. We reserve the notation $\mathbb{1}\{\cdot\}$ for indicator functions that return 1 if the argument is true and 0 otherwise.

For a field $\mathbb{F}$ and a matrix $A \in \mathbb{F}^{m \times n}$, we let $\operatorname{nnz}(A) = |\{(i,j) \in [m] \times [n] : A_{i,j} \neq 0\}|$ be the number of nonzero entries in A, $a_j \in \mathbb{F}_2^m$ be its j-th column and $a_{i,\cdot}$ be its i-th row.

Chapter 2

Combinatorial Compressed Sensing

2.1 From Shannon's sampling theorem to Compressed Sensing

Let χ_k^n be the set of n-dimensional k-sparse vectors as defined in (1.2). In a nutshell, compressed sensing states that if $x \in \chi_k^n$, then it can be sampled and efficiently reconstructed from $m = \mathcal{O}(k)$ non-adaptive linear measurements $y = Ax \in \mathbb{R}^m$ provided $k < m < n$ are sufficiently large. This result is the product of a large research trajectory that started in 1949 with the sampling theorem of Shannon.

Theorem 2.1 ([Shannon, 1949])**.** *If $f(t)$ is a continuous-time signal and contains no frequencies higher than W cycles per second, it is completely determined by giving its ordinates at a series of points spaced $\frac{W}{2}$ seconds apart.*

Theorem 2.1 describes the trade-off between a signal's *data complexity* as measured by the bandwidth of the signal in the frequency space and the *sampling rate* in the physical space needed to characterise it. The bandlimited *data model* assumed on $f(t)$ is equivalent to requiring $\hat{f}(\omega)$, the Fourier transform[1] of $f(t)$, to vanish outside a bounded region of its domain. The duality between the time and the frequency space is further described by Heisenberg's uncertainty principle [Heisenberg, 1927] for which Gabor's Harmonic Analysis version states that a signal and its Fourier transform can not both be highly concentrated [Gabor, 1946]. In other words, Gabor showed that if $f(t)$ is zero almost everywhere in an interval Δt and $\hat{f}(\omega)$ is zero almost everywhere outside an interval $\Delta\omega$, then[2]

$$\Delta t \cdot \Delta\omega \geq \frac{1}{4\pi}. \tag{2.1}$$

[1] Defined on the real line as $\hat{f}(\omega) = \int_{\mathbb{R}} f(t)e^{-2\pi i t\omega}dt$

[2] Technically speaking, it is assumed that $\Delta t := \|(t-t_0)f\|_{L_2(\mathbb{R})}$ and $\Delta\omega := \|(\omega-\omega_0)\hat{f}\|_{L_2(\mathbb{R})}$ for $t_0, \omega_0 \in \mathbb{R}$ and $\|f\|_{L_2(\mathbb{R})} = \|\hat{f}\|_{L_2(\mathbb{R})} = 1$ and $\|\cdot\|_{L_2(\mathbb{R})}$ being the norm in the $L_2(\mathbb{R})$ space.

The uncertainty principle of Gabor implies that the information in non-transient signals or functions that are smooth at low scales is most likely concentrated in an interval in the Fourier space. Though restricted to band-limited signal models, this was a deep and general result and laid the foundations of time-frequency resolution analysis which eventually led to the interest of wavelets [Hill, 2007]. However, this intervaled model was not flexible enough to fully describe the duality in *sparse* or *spiked* data models that was being studied in the context of subset selection [Garside, 1965], seismology [Levy and Fullagar, 1981] and medical ultrasound [Papoulis and Chamzas, 1979]. Indeed, sparse data models were being successfully implemented in engineering application by the geophysics community which sought to recover sparse seismic signals from an incomplete set of its Fourier components by ℓ_1-minimisation [Levy and Fullagar, 1981, Oldenburg et al., 1983, Santosa and Symes, 1986]. The success of these engineering applications made evident that sparse data can be found in nature, so a number of theoretical results soon followed. An important innovation describing the relation between sparsity and ℓ_1 minimisation was given by the uncertainty principles of Donoho and Stark [Donoho and Stark, 1989] which showed that the sets of concentration in Gabor's version of Heisenberg's uncertainty principle need not be intervals, but only measurable, so that if $f \in \mathbb{R}^n$ is a finite dimensional signal and $\hat{f} \in \mathbb{R}^n$ is its discrete Fourier transform, then

$$|\operatorname{supp}(f)| + |\operatorname{supp}(\hat{f})| \geq 2\sqrt{n}. \tag{2.2}$$

The results in [Donoho and Stark, 1989] provided a theoretical justification of a previous result presented [Santosa and Symes, 1986] which showed that a signal with k spikes can be recovered from m Fourier mesurements provided that $m \geq n(1-(2k)^{-1})$.

Though the results in [Donoho and Stark, 1989, Santosa and Symes, 1986] increased popularity in the Fourier-sampling/ℓ_1-minimisation framework, it was well-known that the Fourier transform alone was insufficient to sparsify data with local discontinuities or very sharp oscillations. One way to represent generic data with local transiencies is through a basis containing well-localised functions at the relevant scales. The first such basis was given in 1910 by Haar [Haar, 1910] in the form of a model piecewise constant function with bounded support whose dilations and translations generate an orthonormal basis. The work by Haar served as a starting point in an enormous research effort to find models of basis-generating wave-like oscillations, which pinnacled with the discovery of *Daubechies wavelets* [Daubechies, 1988] which have a support of minimum size for any given number of vanishing moments[3]. Over

[3]See [Mallat, 1999] for an introduction to the theory of wavelets

the years, numerous families of wavelets continued to appear [Candès and Donoho, 2000, Guo et al., 2006, Beylkin et al., 1991], which increased our power to compressively represent many more sources of data. As a result, [Mallat and Zhang, 1993] coined the concept of *dictionary* to refer to overcomplete sets of *atoms* over which signals can be decomposed after which the problems of non-uniqueness and overfitting became evident. To resolve these issues the problem had to be regularised so for an overcomplete dictionary $A \in \mathbb{R}^{m\times n}$ the *sparsest* representation $x \in \mathbb{R}^n$ of a signal $y = Ax \in \mathbb{R}^m$ was sought, giving the ℓ_0-minimisation problem

$$\min \|x\|_0 \qquad \text{s.t.} \qquad y = Ax, \tag{2.3}$$

which seeks for the sparsest solution that agrees with the measurements. The problem (2.3) is NP-hard as it can be efficienty reduced to the *minimum set cover problem* [Foucart and Rauhut, 2013], so sparsity-promoting heuristics started to emerge. The first such strategy was the Matching Pursuit algorithm [Mallat and Zhang, 1993] which first established the *iterative* and *greedy* algorithmic model that would become standard in compressed-sensing. Specifically, Matching Pursuit (Algorithm 1) progressively refines an estimate $\hat{x}$ of the solution by selecting atoms in the dictionary that best-correlate with the residual $r = y - A\hat{x}$ at each iteration. It was shown in [Jones, 1987] that Algorithm 1 converges for a general dictionary A.

Algorithm 1: Matching Pursuit [Mallat and Zhang, 1993]

Data: $A \in \mathbb{R}^{m\times n}$; $y \in \mathbb{R}^m$
Result: $\hat{x} \in \mathbb{R}^n$ s.t. $y \approx A\hat{x}$
$\hat{x} \leftarrow 0$, $r \leftarrow y$;
while *not converged* **do**
  Find $j \in [n]$ s.t. $\left(A^T r\right)_j = \|A^T r\|_\infty$;
  $\hat{x} \leftarrow \hat{x} + (a_j^T r)a_j$;
  $r \leftarrow y - A\hat{x}$;
end

A slightly more refined strategy was given by Orthogonal Matching Pursuit (OMP) [Davis et al., 1994, Pati et al., 1993, Chen et al., 1989] which rather than minimising $\|r - a_j z\|_2$ over $(j, z) \in [n] \times \mathbb{R}$, first constructs an estimate of $\text{supp}(\hat{x})$ and uses it to compute the best-fit $\arg\min_z \|y - A_{\text{supp}(\hat{x})} z\|_2$ restricted to this support, see Algorithm 2.

Algorithms 1 and 2 are not guaranteed to produce unique sparse decompositions, so [Chen et al., 2001] introduced the idea of an *optimal* decomposition as that with

Algorithm 2: Orthogonal Matching Pursuit [Davis et al., 1994, Pati et al., 1993, Chen et al., 1989]

Data: $A \in \mathbb{R}^{m\times n}$; $y \in \mathbb{R}^m$
Result: $\hat{x} \in \mathbb{R}^n$ s.t. $y \approx A\hat{x}$
$\hat{x} \leftarrow 0$, $r \leftarrow y$;
$\Lambda \leftarrow \emptyset$;
while *not converged* **do**
  Find $j \in [n]$ s.t. $\left(A^T r\right)_j = \|A^T r\|_\infty$;
  $\Lambda \leftarrow \Lambda \cup \{j\}$;
  $\hat{x} \leftarrow \arg\min_z \|y - Az\|_2 \,\text{s.t.}\, \text{supp}(z) \subset \Lambda$;
  $r \leftarrow y - A\hat{x}$;
end

minimum ℓ_1-energy that is consistent with the measurments, *i.e.*

$$\min \|x\|_1 \qquad \text{s.t.} \qquad y = Ax. \tag{2.4}$$

The program (2.4) is commonly known as *basis pursuit* and became an important object of study. A number of results concerning the sparsifying properties of the ℓ_1 regulariser soon started to appear. For example, [Donoho and Huo, 2001] showed that if a signal is highly sparse in a dictionary of spiked sequences and sinusoids then there is only one highly sparse representation of the signal in terms of these atoms and can be recovered by (2.4). A few years later, [Donoho and Elad, 2003] gave a result that guranteed decodability with (2.4) in the more general setting of non-orthogonal dictionaries, while [Gribonval and Nielsen, 2003] gave precise conditions to guarantee ℓ_0-ℓ_1 *equivalence*, namely equivalence between the programs (2.3) and (2.4). Simultaneously, [Gilbert et al., 2003b, Tropp, 2004] showed that a measurement rate of $m = \mathcal{O}(k^2)$ was sufficient to guarantee equivalence between Algorithm 2 and (2.4). These results were given in terms of a property of the measurement matrix called *coherence* or the maximum absolute value of the cross inner-products between the columns of a matrix.

The seminal work on compressed-sensing was given with the work of [Candès and Tao, 2006, Candès et al., 2006a, Donoho, 2006c] where it was shown that a k-sparse signal $x \in \chi_k^n$ can be efficiently recovered via (2.4) from $m = \mathcal{O}(k \log n)$ non-adaptive measurements $\{\langle a_1, x\rangle, \ldots, \langle a_m, x\rangle\}$. These groundbreaking results were trumpeted by a fundamental innovation in uncertainty principles [Candès and Romberg, 2006],

which showed that with *overwhelming probability* [4]

$$|\operatorname{supp}(f)| + |\operatorname{supp}(\hat{f})| > \frac{n}{\sqrt{(\beta+1)\log n}}. \tag{2.5}$$

The key argument behind (2.5) was that the uncertainty principle of Donoho and Stark in (2.2) was tight only for signals that were very special and not representative of *most* signals.

The $\mathcal{O}(k \log n)$ sampling rate shown initially is off from the optimal $\mathcal{O}(k)$ rate by a $\mathcal{O}(\log n)$-factor. This might sound reasonable if we think of the $\mathcal{O}(\log n)$-factor as the *price to pay* for not knowing in advance the location of $\operatorname{supp}(x)$. However, it was later shown in [Candès and Tao, 2005, Donoho, 2006b] that when measuring with the uniform spherical ensemble or with the sub-Gaussian[5] ensemble, this sampling rate could be further improved to

$$m = \mathcal{O}\left(k \log \frac{n}{k}\right). \tag{2.6}$$

The improvement of (2.6) over $\mathcal{O}(k \log n)$ is significant, since (2.6) reduces to the *optimal* rate of $\mathcal{O}(k)$ when the parameters (k, m, n) of the problem scale according to the linear growth asymptotic

$$\frac{m}{n} \to \delta \in (0,1) \text{ and } \frac{k}{m} \to \rho \in (0,1) \text{ as } k, m, n \to \infty. \tag{2.7}$$

The constant $\delta \in (0,1)$ is known as the *undersampling ratio* while ρ is known as the *sparsity ratio*. In proving this result, [Candès and Tao, 2005] introduced the *Restricted Isometry Constants* (RIC-2)

$$c_k := \min_{c \geq 0} c \quad \text{s.t.} \quad (1-c)\|x\|_2 \leq \|Ax\|_2 \leq (1+c)\|x\|_2 \quad \forall x \in \chi_k^n, \tag{2.8}$$

which for given $k \in [n]$ measure the extent for which a matrix $A \in \mathbb{R}^{m \times n}$ is locally almost isometric for all k-sparse signals. In a way, the RIC-2 constants c_k quantify the degree to which A locally satisfies the Pythagorean theorem. Indeed, if $\hat{a}_1, \ldots, \hat{a}_n \in \mathbb{R}^n$ are the columns of an orthonormal matrix $\hat{A} \in \mathbb{R}^{n \times n}$ and $x \in \mathbb{R}^n$, then $|\text{hypotenuse}|^2 = \|Ax\|^2 = \|x\|^2$ so Pythagoras' theorem guarantees that

$$(1-0)\|x\| \leq |\text{hypotenuse}| = \|\hat{A}x\| \leq (1+0)\|x\|.$$

The existence of the RIC-2 follows from concentration of measure inequalities in random matrix theory. In [Candès and Tao, 2005] it was shown that, with overwhelming

[4] With probability greater than or equal to $1 - \mathcal{O}(n^{-c})$ for every fixed $c > 0$ [Tao, 2012].
[5] $\mathbb{E}[\exp(\theta A_{i,j})] \leq \exp(c\theta^2)$ for all $\theta \in \mathbb{R}$ and all $i \in [m], j \in [n]$.

probability, Gaussian matrices with mean zero and variance $1/m$ have sufficiently good RIC-2 to guarantee ℓ_0-ℓ_1 equivalence. A more general result was given in [Candès and Tao, 2006] ensured that sub-Gaussian matrices satisfy $c_{2k} < 0.6246$ with overwhelming probability and thus guarantee ℓ_0-ℓ_1 equivalence. In [Candès et al., 2006b] these results were extended to deal with scenarios where the signal is corrupted by additive noise and when the underlying signal is not exactly sparse. Additional to these results was the observation that sub-Gaussian matrices enable *universality*, meaning that if $\Psi \in \mathbb{R}^{n\times n}$ is a fixed orthogonal matrix and $A \in \mathbb{R}^{m\times n}$ is sub-Gaussian with $m = \mathcal{O}(k\log(n/k))$, then $A\Psi$ will have a sufficiently small RIC-2 constants with overwhelming probability [Foucart and Rauhut, 2013]. This was of particular importance to imaging application where $s \in \mathbb{R}^n$ is not naturally sparse, but is known to be sparse in some basis $\Psi \in \mathbb{R}^{n\times n}$.

The limits of ℓ_1-minimisation were charted in [Donoho, 2006a] by proving that when $\delta = \frac{m}{n} \in (0,1)$ and A is sampled uniformly from the Grassmann manifold of m-dimensional orthoprojectors of $\mathbb{R}^n$, there are constants $\rho_S(\delta), \rho_W(\delta) \in (0,1)$ such that as $k, m, n \to \infty$ with respect to the linear growth asymptotic (2.7),

$$\frac{k}{m} < \rho_S(\delta) \text{ implies } \ell_0\text{-}\ell_1 \text{ equivalence with probability } 1 - o(1),$$
$$\frac{k}{m} < \rho_W(\delta) \text{ implies } \ell_0\text{-}\ell_1 \text{ equivalence with overwhelming probability.}$$

These results were based on deep results coming from combinatorial geometry regarding the expected number of k-faces in a cross-polytope $\{x \in \mathbb{R}^n : \|x\|_1 \leq t\}$ that survive a random orthogonal projection [Affentranger and Schneider, 1992, Böröczky Jr and Henk, 1999, Vershik and Sporyshev, 1986]. In [Donoho and Tanner, 2009b] it was formally shown that that these notions are *precise* in the sense that having $\frac{k}{m} > \rho_W(\delta)$ or $\frac{k}{m} > \rho_S(\delta)$ implied that ℓ_0-ℓ_1 equivalence failed to hold. An extension of this result to Gaussian random matrices was later given by [Baryshnikov and Vitale, 1994]. Therefore, the Gaussian ensemble exhibits an abrupt *phase transition* in which for a given $\delta \in (0,1)$, the probability of successfully recovering the signal changes dramatically from one to zero as $\frac{k}{m}$ grows beyond $\rho_W(\delta)$ or $\rho_S(\delta)$. Finally, [Donoho and Tanner, 2009a] posed the hypothesis that under the ℓ_1-minimisation framework many other families of random matrices match the quality of phase transitions of the Gaussian ensemble.

2.2 Iterative greedy algorithms

Even though (2.4) can be posed as a linear program, using convex optimisation methods is not the best strategy to follow. Though the simplex algorithm [Dantzig and Thapa, 2006] is known to have cubic average average convergence [Borgwardt, 1987], this is too high to be useful in real-applications where the ambient dimensionality of the signal is potentially very large and is known to be exponential in the worst-case [Dantzig and Thapa, 2006]. A number of algorithms for solving the convex relaxation via the interior point method[6] were proposed in [Chen et al., 2001, Candès and Romberg, 2005, Kim et al., 2007], but can be costly when the matrices are dense. The best alternatives to convex optimisation from the computational point of view are iterative greedy algorithms. Iterative greedy algorithms for compressed sensing seek the sparsest solution to a large underdetermined linear system of equations $y = Ax$ and by iteratively refining an estimate $\hat{x}$ of the solution and shrinking the residual $r = y - A\hat{x}$.

The study of iterative greedy algorithms in compressed sensing started with the work of [Tropp and Gilbert, 2007] which showed that for $t \in (0, 0.36)$, Algorithm 2 with $\Omega(k \log(n/t))$ Gaussian measruements exactly recovers any k-sparse signal with a probability greater than $1-2t$. A more refined strategy was given shortly after with the developement of Compressive Sampling Matching Pursuit (CoSaMP) [Needell and Tropp, 2009]. CoSaMP can be understood as a three-step adaptation of Algorithm 2

Algorithm 3: Compressive Sampling Matching Pursuit [Needell and Tropp, 2009]

Data: $A \in \mathbb{R}^{m \times n}$; $y \in \mathbb{R}^m$
Result: $\hat{x} \in \mathbb{R}^n$ s.t. $y \approx A\hat{x}$
$\hat{x} \leftarrow 0$, $r \leftarrow y$;
while *not converged* **do**
  $\Lambda \leftarrow \mathrm{supp}(x) \cup \mathrm{supp}(H_{2k}(A^T r))$;
  $\hat{x}' \leftarrow \arg\min_z \|y - Az\|_2 \,\mathrm{s.t.}\, \mathrm{supp}(z) \subset \Lambda$;
  $\hat{x} \leftarrow H_k(\hat{x}')$;
  $r \leftarrow y - A\hat{x}$;
end

to compressed sensing. First, it pools the indices of the $2k$ largest entries in absolute value in $A^T r$ with those of the support of the current estimation to form a set Λ. Second, it finds the least-squares estimate of the signal constrained to be supported

[6] See [Boyd and Vandenberghe, 2004] for an introduction.

at Λ. Finally, it projects the estimate to χ_k^n by setting to zero all but its k largest entries in absolute value. A sufficient condition for recovery was also given in [Needell and Tropp, 2009] guaranteeing optimal recovery whenever the RIC-2 constant of the matrix is $c_{4k} < 0.1$.

Another important class of algorithms is the class of linesearch algorithms, which start with an estimate $\hat{x} \leftarrow 0$ and iteratively refine the estimation by moving in the direction of steepest descent and then projecting the solution back to χ_k^n. In Iterative Hard Thresholding (IHT) [Blumensath and Davies, 2009], the stepsize μ is set to one for all iterations, while in Normalized Iterative Hard Thresholding (NIHT) [Blumensath and Davies, 2010] the optimal stepsize is computed at each iteration, see Algorithm 4. In [Blumensath and Davies, 2009] the condition $c_{3k} < 1/\sqrt{8}$ is given as a sufficient condition for optimal recovery. The Hard Thresholding Pursuit [Foucart, 2011] shown in Algorithm 5 adds a least-squares minimisation step to Algorithm 4 by tracking the support of the largest k entries in absolute value of $\hat{x} + \mu A^T(y - A\hat{x})$ and then finding the least-squares estimate subject to this support. Finally,

Algorithm 4: (Normalized) Iterative Hard Thresholding [Blumensath and Davies, 2009, 2010]

Data: $A \in \mathbb{R}^{m\times n}$; $y \in \mathbb{R}^m$
Result: $\hat{x} \in \mathbb{R}^n$ s.t. $y \approx A\hat{x}$
$\hat{x} \leftarrow 0$, $r \leftarrow y$;
while *not converged* **do**
  Compute optimal step-size μ;
  $\hat{x} \leftarrow H_k(\hat{x} + \mu A^T r)$;
  $r \leftarrow y - A\hat{x}$;
end

Algorithm 5: Hard thresholding Pursuit [Foucart, 2011]

Data: $A \in \mathbb{R}^{m\times n}$; $y \in \mathbb{R}^m$
Result: $\hat{x} \in \mathbb{R}^n$ s.t. $y \approx A\hat{x}$
$\hat{x} \leftarrow 0$, $r \leftarrow y$;
while *not converged* **do**
  $\Lambda \leftarrow \mathrm{supp}(H_k(\hat{x} + A^T r))$;
  $\hat{x} \leftarrow \arg\min_z \|y - Az\|_2 \,\mathrm{s.t.}\, \mathrm{supp}(z) \subset \Lambda$;
  $r \leftarrow y - A\hat{x}$;
end

Subspace Pursuit (SP) [Dai and Milenkovic, 2009] splits the support-estimation step

of Algorithm 5 in two stages where in each stage an ℓ_2-minimisation step is performed. It was also shown in [Dai and Milenkovic, 2009] that Algorithm 6 has a recovery condition of $c_{3k} < 0.165$.

Algorithm 6: Subspace Pursuit [Foucart, 2011]

Data: $A \in \mathbb{R}^{m\times n}$; $y \in \mathbb{R}^m$
Result: $\hat{x} \in \mathbb{R}^n$ s.t. $y \approx A\hat{x}$
$\hat{x} \leftarrow 0$, $r \leftarrow y$;
while *not converged* **do**
  $\Lambda \leftarrow \Lambda \cup \mathrm{supp}(H_k(A^T r))$;
  $\hat{x} \leftarrow \arg\min_z \|y - Az\|_2 \,\mathrm{s.t.}\, \mathrm{supp}(z) \subset \Lambda$;
  $\Lambda \leftarrow \mathrm{supp}(\hat{x})$;
  $\hat{x} \leftarrow \arg\min_z \|y - Az\|_2 \,\mathrm{s.t.}\, \mathrm{supp}(z) \subset \Lambda$;
  $r \leftarrow y - A\hat{x}$;
end

Most compressed-sensing sampling and recovery guarantees hold in the asymptotic regime $k, m, n \to \infty$, so they most likely manifest for very large problem sizes. For this reason, it is particularly important to have low-latency algorithms with provable guarantees like Algorithms 4, 5 and 6 and in particular to take advantage of hardware innovations like parallel architectures. In [Blanchard and Tanner, 2013] parallel implementations of these and other algorithms were given. More recently, [Blanchard et al., 2015] proposed CG-IHT which is a version of 4 in which rather than using the search direction $p = A^T r$, it uses conjugate gradients[7] to compute the optimal step-size and search direction. Most of our contributions in this thesis are designed in such a way that they can profit from parallelisations.

Over the years, many other algorithms implementing different models have appeared. Among these are the message-passing algorithms [Donoho et al., 2009], algorithms based on belief-propagation [Sarvotham et al., 2006a], and more recently neural-based compressed sensing decoders [Mousavi and Baraniuk, 2017].

The computational complexity of the iterative greedy algorithms reviewed in this section is of $\Omega(\mathrm{nnz}(A))$ since all have to compute $A^T r$. An utopically optimal decoding algorithm would be one with a convergence guarantee of $\mathcal{O}(\mathrm{nnz}(A))$ or at the *data-reading* cost.

[7] See [Trefethen and Bau III, 1997] for an introduction

2.3 Measuring with sparse and structured operators

Though most of the theoretical guarantees are given in terms of the RIC-2 constants of sub-Gaussian matrices, these can be suboptimal in practice due to their high complexity for storage, generation, and computation since, in order to sample a sub-Gaussian matrix $A \in \mathbb{R}^{m\times n}$, one needs to sample mn numbers at a time and space cost of $\mathcal{O}(mn)$. Moreover, in the absence of any other structure the cost of computing the matrix vector products Ax and $A^T y$ is, respectively, $\mathcal{O}(km)$ and $\mathcal{O}(mn)$. Some alternatives like partial Fourier ensembles, which are generating by drawing m rows from the discrete Fourier transform, have much lower complexity due to their structure. Indeed, the DFT maps $\mathbb{C}^n$ onto itself via

$$y_k = \sum_{j=1}^{n} x_{j-1} e^{-i2\pi k(j-1)/n}, \tag{2.9}$$

so the mapping can be completely recontructed from m indices in $[n]$. The sparse fast Fourier transform (sparse-FFT) developed in [Hassanieh et al., 2012] guarantees computation of Ax at cost $\mathcal{O}(k \log n)$ and of $A^T y$ at cost $\mathcal{O}(n \log n)$. However, partial Fourier matrices are not known to allow sampling at the optimal rate of $\mathcal{O}(k \log(n/k))$ [Bassalygo and Pinsker, 1973, Alon and Capalbo, 2002], rather the best analysis ensures recovery for $m \sim k \log^2 k \log n$ [Haviv and Regev, 2015].

With this in mind, it be desirable to have other sensing operators that guarantee an optimal measurement rate while being sufficiently structure as to guarantee efficiency. One such class of operators was proposed by [Berinde et al., 2008a] and is given as a family of adjacency matrices of so-called expander graphs. These matrices are sparse, binary and have have $d \ll m$ ones per column, offering the possibility of storage and generation at cost $\mathcal{O}(dn)$ and of A and A^T being applied at cost of $\mathcal{O}(\frac{d}{m}k)$ and $\mathcal{O}(dn)$ respectively for an asymptotically optimal number of measurements $m = \mathcal{O}(k)$ [Bah and Tanner, 2013, Berinde et al., 2008a]. Hence, expander matrices compare favourably against sub-Gaussian and partial Fourier ensembles as shown in Table 2.1.

2.3.1 Expander sketching and ℓ_1-uncertainty principles

It was noted in [Berinde et al., 2008a] that the problem of sparse recovery with compressed sensing resembles the problem of *linear sketching*, in which the dimensionality of a vector $x \in \mathbb{R}^n$ is reduced by a linear mapping $A : \mathbb{R}^n \to \mathbb{R}^m$ in such a way that the essential geometric information of x is preserved with high probability. In an

	Storage	Generation	Ax	$A^T y$	m	$\frac{\text{nnz}(A)}{mn}$
sub-Gaussian	$\mathcal{O}(mn)$	$\mathcal{O}(mn)$	$\mathcal{O}(km)$	$\mathcal{O}(mn)$	$\mathcal{O}(k\log(n/k))$	$\Omega(1)$
Partial Fourier	$\mathcal{O}(m)$	$\mathcal{O}(m)$	$\mathcal{O}(k\log n)$	$\mathcal{O}(n\log n)$	$\mathcal{O}(k\log^2 k\log n)$	$\Omega(1)$
Expander	$\mathcal{O}(dn)$	$\mathcal{O}(dn)$	$\mathcal{O}(dk)$	$\mathcal{O}(dn)$	$\mathcal{O}(k\log(n/k))$	$o(1)$

Table 2.1: **Complexity of measurement operators.** A comparison between the complexity of measurement operators.

attempt to reconcile this area with the compressed-sensing paradigm, [Berinde et al., 2008a] proposed sketching $x \in \chi_k^n$ using an *expander matrix* or the adjacency matrix of a *left d-regular expander graph*[8]; this is known as *combinatorial compressed sensing*. Expander matrices are highly sparse and structured and, as seen in Table 2.1, this makes them highly efficient in terms of generation, storage and computation. It is because of this comparative advantage that expander matrices are ideal for *data-stream computing* [Alon et al., 1996, Indyk, 2007, Muthukrishnan et al., 2005] which studies algorithms that compute on *data-streams* or sequences of data items that can only be read once by the algorithm computing on them [Henzinger et al., 1998]. In this context a sketch $y = Ax$ of a vector x is sought to be maintained as a sequence of updates of the form

$$\omega_{j_1}, \omega_{j_2}, \ldots, \omega_{j_p}, \ldots$$

becomes available. In this case, rather than updating $x_{j_\ell} \leftarrow x_{j_\ell} + \omega_{j_\ell}$ in the ambient dimension, the sketch is efficiently updated as

$$Ax \leftarrow Ax + \omega_{j_\ell} a_{j_\ell}. \tag{2.10}$$

Data streaming has been used for data summarisation [Cormode and Muthukrishnan, 2005], estimating frequencies of objects [Charikar et al., 2002, Gilbert et al., 2002] compressing large datasets [Gilbert et al., 2003a], measuring network traffic [Estan and Varghese, 2003] and optimising queries [Muthukrishnan et al., 2005], which opens up applications to combinatorial compressed sensing. Apart from linear sketching and data-stream computing, adjacency matrices of expander graphs are useful in many other applications that include graph sketching, combinatorial group testing,

[8]See Section 2.3.2 for details.

network routing, error-correcting codes, fault-tolerance, and distributed storage [Capalbo et al., 2002]. Additionally, some applications like the single-pixel camera [Baraniuk, 2008, Takhar et al., 2006, Duarte et al., 2008] consider measurement devices with binary sensors that inherently correspond to binary and sparse inner products, and that unfortunately, fall outside the set of matrices for which the widely used restricted isometry techniques apply. More recently, expander matrix models have also been used in Artificial Intelligence research to explain the behaviour of deep neural networks [Arora et al., 2014].

The birth of combinatorial compressed-sensing was accompained with new uncertainty principle [Berinde et al., 2008a], which showed that if $A \in \{0,1\}^{m\times n}$ is an expander matrix and $x \in \mathrm{Ker}(A)$, then for any $S \in [n]^k$ it holds that

$$\|x_S\|_1 \leq \frac{2\varepsilon}{1-2\varepsilon}\|x\|_1, \tag{2.11}$$

for some parameter $\varepsilon \in (0,1)$ that measures the connectivity of the underlying expander graph, see Section 2.3.2. This uncertainty principle was analogous to a result given in [Kashin and Temlyakov, 2007] and thus guarateed recovery by ℓ_1-minimisation. Indeed, given the structure of these matrices and the previous advances in coding theory a series of algorithms designed specifically to work with expander matrices was soon presented in [Jafarpour et al., 2009, Berinde et al., 2008b, Berinde and Indyk, 2009, Xu and Hassibi, 2007], which we reveiw shortly.

2.3.2 Graph theory and Expander graphs

In this section we review the basic notions of graph theory that are relevant to our discussion. A *bipartite graph* is a 3-tuple $G = (U, V, E)$ such that $U \cap V = \emptyset$ and $E \subset U \times V$. Elements in $U \cup V$ are called *nodes* or the *vertices*, while tuples in E are called *edges*. Under the assumption that $|U| = n$ and $|V| = m$, we abuse notation and let $U = [n]$ be the set of *left-nodes*, and $V = [m]$ be the set of *right-nodes* noting that since $m < n$ there are nodes in the graph with the same label, but can nevertheless be uniquely identified from the context. A bipartite graph is said to be *left d-regular* if the number of edges emanating from each left node is identically d, and is said to be *unbalanced* if $m < n$, see Figure 2.1 for an illustration of a bipartite graph. For $S \subset U \cup V$ we define $\mathcal{N}(S) \subset U \cup V$ to be the *neighbourhood* of S, *i.e.* the set of nodes in $U \cup V$ that are connected to S through an element of E. We note that for bipartite graphs, $\mathcal{N}(S) \subset V$ only if $S \subset U$, and $\mathcal{N}(S) \subset U$ only if $S \subset V$. An *expander graph* is an unbalanced, left d-regular, bipartite graph that is *well-connected* in the sense of the following definition.

Definition 2.2 (Expander graph [Sipser and Spielman, 1996])**.** *An unbalanced, left d-regular, bipartite graph $G = ([n], [m], E)$ is a (k, ε, d)-expander if*

$$|\mathcal{N}(S)| > (1 - \varepsilon)d|S| \quad \forall \quad S \in [n]^{(\leq k)}. \tag{2.12}$$

We call $\varepsilon \in (0, 1)$ the expansion parameter *of the graph.*

A schematic of an expander graph is shown in Figure 2.1.

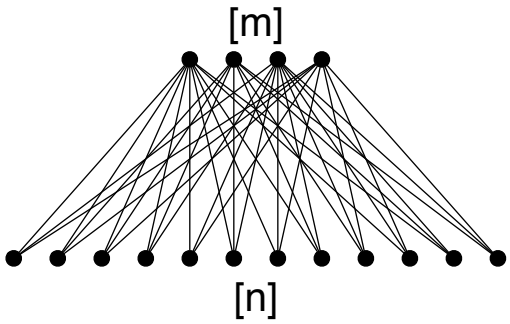


Figure 2.1: **Schematic of an expander graph with $d = 3$.** Every left d-regular bipartite graph is an expander for some k and ε.

Hence, the expander graphs that we consider can be thought of as tuples $G = ([n], [m], E)$ such that all subsets $S \in [n]^{(\leq k)}$ have at most $\varepsilon d|S|$ fewer neighbours than the number of edges emanating from S. It will be convenient to think of an expander in linear algebra terms, which can be done via its *adjacency matrix*.

Definition 2.3 (Expander matrix $\mathbb{E}_{k,\varepsilon,d}^{m\times n}$)**.** *The adjacency matrix of an unbalanced, left d-regular, bipartite graph $G = ([n], [m], E)$ is the binary sparse matrix $A \in \mathbb{R}^{m\times n}$ defined as*

$$A_{ij} = \begin{cases} 1 & i \in \mathcal{N}(j) \subset [m] \\ 0 & \text{otherwise.} \end{cases} \tag{2.13}$$

We let $\mathbb{E}_{k,\varepsilon,d}$ be a the set of adjacency matrices of (k, ε, d)-expander graphs.

By Definition 2.3 it holds that if $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$, then $\sum_{i=1}^{m} \mathbb{1}\{|A_{i,j}| > 0\} = d$ for all $j \in [n]$ and

$$\left|\left\{i \in [m] : \sum_{j\in S} \mathbb{1}\{|A_{i,j}| > 0\}\right\}\right| > (1 - \varepsilon)d|S|$$

for all $S \in [n]^{(\leq k)}$.

We note that $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$ is a sparse binary matrix with exactly d ones per column, and also that any left d-regular bipartite graph will satisfy (2.12) for some k and ε. As mentioned previously, [Berinde et al., 2008a] showed that these matrices possess a

bounded restricted isometry constant in the ℓ_1 norm (RIC-1), which analogously to (2.8) is defined as

$$c_k^1 := \min_{c\geq 0}(1-c)\|x\|_1 \leq \|Ax\|_1 \leq (1+c)\|x\|_1. \tag{2.14}$$

The existence of expander graphs with the optimal measurement rate of $m = \mathcal{O}(k\log(n/k))$, is addressed in the following theorem.

Theorem 2.4 (Existence of optimal expanders [Capalbo et al., 2002, Bassalygo and Pinsker, 1973])**.** *For any $n/2 \geq k \geq 1$ and $\varepsilon > 0$, there is a (k, ε, d)-expander with*

$$d = \mathcal{O}(\log(n/k)/\varepsilon), \quad \text{and} \quad m = \mathcal{O}(k\log(n/k)/\varepsilon^2). \tag{2.15}$$

Theorem 2.4 also implies that in the linear growth asymptotic of $k \sim m \sim n \to \infty$ and for a fixed $\varepsilon > 0$, it holds that $d = \mathcal{O}(1)$ and $m = \mathcal{O}(k)$; that is, the number of non-zeros per column does not increase with the problem size. The existence of optimal expander is complemented by the result of [Bah and Tanner, 2013] who showed that a matrix drawn from $\{0,1\}^{m\times n}$ with each column having support drawn uniformly from $[m]^d$ and independently from other columns, is an expander with high probability. Moreover, [Guruswami et al., 2010] gave an explicit deterministic construction of an expander matrix achieving $m \approx n/\operatorname{poly}(\log n)$ and ensuring recovery for $k = \mathcal{O}\left(n/((\log n)^{C\log\log\log n})\right)$.

The structure in expander matrices is not limited to their left d-regularity. As reviewed in the manuscript [Capalbo et al., 2002], expander matrices satisfy a number of useful properties. Among these is the so-called *unique neighbour property*, which says that for $x \in \chi_k^n$ at least $(1-2\varepsilon)kd$ entries of $y = Ax$ equal a nonzero value of x. This property is guaranteed by Lemma 2.5. For completeness, we give a proof of this Lemma. Similar proofs might have been presented previously in classic literature.

Lemma 2.5 (Unique neighbour property [Sipser and Spielman, 1996])**.** *Let $G = ([n], [m], E)$ be an unbalanced, left d-regular, bipartite graph, and $S \in [n]^{(\leq k)}$. Define,*

$$\mathcal{N}_1(S) = \{i \in \mathcal{N}(S) : |\mathcal{N}(i) \cap S| = 1\}, \tag{2.16}$$

and

$$\mathcal{N}_{>1}(S) = \mathcal{N}(S) \setminus \mathcal{N}_1(S). \tag{2.17}$$

If G is a (k, ε, d)-expander graph, then

$$|\mathcal{N}_1(S)| > (1-2\varepsilon)d|S| \ \ \forall \ \ S \in [n]^{(\leq k)}. \tag{2.18}$$

Proof. For any unbalanced, left d-regular, bipartite graph it holds that:

$$|\mathcal{N}_1(S)| + |\mathcal{N}_{>1}(S)| = |\mathcal{N}(S)|, \tag{2.19}$$

$$|\mathcal{N}_1(S)| + 2|\mathcal{N}_{>1}(S)| \leq d|S|. \tag{2.20}$$

Where (6.8) follows from the definition of $\mathcal{N}_{>1}(S)$, and (2.20) by double-counting the edges emanating from S to $\mathcal{N}(S)$. Now, to prove that (2.18) is necessary, assume that G is a (k, ε, d)-expander graph. Then, for $S \in [n]^{(\leq k)}$ we have that

$$|\mathcal{N}(S)| > (1 - \varepsilon)d|S|. \tag{2.21}$$

Combining (6.8), (2.20) and (2.21) we get the chain of inequalities

$$d|S| - |\mathcal{N}_{>1}(S)| \geq |\mathcal{N}(S)| > (1 - \varepsilon)d|S|, \tag{2.22}$$

which yield

$$|\mathcal{N}_{>1}(S)| < \varepsilon d|S|. \tag{2.23}$$

Plugging (2.23) into (6.8) and using (2.21) we obtain

$$|\mathcal{N}_1(S)| > (1 - 2\varepsilon)d|S|. \tag{2.24}$$

□

The unique neighbour property is widely used in the analysis of CCS, and is a central piece in the analysis of our algorithms as it implies the existence of the RIC-1 constants (2.14).

This expander property becomes particularly interesting in the realm of iterative greedy decoding algorithms. Algorithms for combinatorial compressed sensing differ from the *geometric* decoders shown in Algorithms 1, 2, 4, 5, 3, 6 in that they consider updating the j^{th} entry of the approximation, $\hat{x}_j$, based on a non-inner product score $s_j \in \mathbb{R}$ dependent on $r_{\mathcal{N}(j)}$; that is, on the residual restricted to the support of the j^{th} column of A. Even though the geometric decoders are still able to decode signals that were measured with expanders, they are not optimised to take advantage of the additional information

$$A \in \mathbb{E}_{k,\varepsilon,d}^{m \times n}.$$

In this sense, it is expected that combinatorial compressed sensing algorithms can outperform geometric algorithms in terms of speed or recovery region. In the remainder of this section we deconstruct past CCS algorithms into their essential components so as to give a high-level overview of their shared characteristics.

2.3.3 Sparse Matching Pursuit (SMP)

SMP was proposed in [Berinde et al., 2008b] to decode $\hat{x}$ from $y = Ax$ with a voting-like mechanism in the spirit of the *count-median* algorithm from data-stream computing (see [Cormode and Muthukrishnan, 2005] for details). SMP can also be viewed as an *expander* adaptation of the Iterative Hard Thresholding algorithm (IHT) [Blumensath and Davies, 2009], which uses the line-search $\hat{x} \leftarrow H_k[\hat{x} + p]$ to minimise $\|y - A\hat{x}\|_2^2$ over χ_k^n, indeed it was rediscovered from this perspective in [Foucart and Rauhut, 2013][pp. 452] where it is referred to as EIHT. Due to the structure of expander matrices, SMP chooses the direction $p = \mathcal{M}(y - A\hat{x})$ with $\mathcal{M} : \mathbb{R}^m \to \mathbb{R}^n$ defined as

$$[\mathcal{M}(r)]_j = \text{median}(r_{\mathcal{N}(j)}). \tag{2.25}$$

After thresholding, this choice yields the iteration,

$$\hat{x} \leftarrow H_k \left[\hat{x} + H_{2k}[\mathcal{M}(y - A\hat{x})]\right]. \tag{2.26}$$

SMP and its theoretical guarantees are stated in Algorithm 7 and Theorem 2.6.

Algorithm 7: SMP [Berinde et al., 2008b]

Data: $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$; $y \in \mathbb{R}^m$
Result: $\hat{x} \in \mathbb{R}^n$ s.t. $\|x - \hat{x}\|_1 = \mathcal{O}(\|y - Ax\|_1/d)$
$\hat{x} \leftarrow 0$, $r \leftarrow y$;
while *not converged* **do**
  $\hat{x} \leftarrow H_k\left[\hat{x} + H_{2k}[\mathcal{M}(r)]\right]$;
  $r \leftarrow y - A\hat{x}$;
end

Theorem 2.6 (SMP [Berinde et al., 2008b])**.** *Let $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$ and let $y = Ax + \eta$ for $x \in \chi_k^n$. Then, there exists an $\varepsilon \ll 1$ such that SMP recovers $\hat{x} \in \mathbb{R}^n$ such that $\|x - \hat{x}\|_1 = \mathcal{O}(\|\eta\|_1/d)$. The algorithm terminates in $\mathcal{O}(\log(d\|x\|_1/\|\eta\|_1))$ iterations with complexity $\mathcal{O}(nd + n\log n)$.*

2.3.4 Sequential Sparse Matching Pursuit (SSMP)

It was observed in [Berinde and Indyk, 2009] that SMP typically failed to converge to the sought sparsest solution when the problem parameters fall outside the region of theoretical guarantees. Though SMP updates each entry in x to individually reduce the ℓ_1 norm of the residual, by updating multiple values of x in parallel causes SMP to diverge even for moderately small ratios of k/m. To overcome these limitations, the

authors proposed SSMP, which updates $\hat{x}$ sequentially rather than in parallel. That is, at each iteration, SSMP will look for a single node $j \in [n]$ and an update $\omega \in \mathbb{R}$ that minimise $\|r - \omega a_j\|_1$, which can be found by computing $\arg\max_{j\in[n]} \mathcal{M}(r)$. This approach results in a strict decrease in $\|r\|_1$, but the sequential update results in an overall increase in computational complexity, see Table 2.2 for a comparison in the noiseless case. SSMP and its theoretical guarantees are stated in Algorithm 8 and Theorem 2.7.

Algorithm 8: SSMP [Berinde and Indyk, 2009]

Data: $A \in \mathbb{E}^{m\times n}_{k,\varepsilon,d}$; $y \in \mathbb{R}^m$; $c > 1$;
Result: $\hat{x} \in \mathbb{R}^n$ s.t. $\|x - \hat{x}\|_1 = \mathcal{O}(\|y - Ax\|_1/d)$
$\hat{x} \leftarrow 0$, $r \leftarrow y$;
while *not converged* **do**
  Find $(j, \omega) \in [n] \times \mathbb{R}$ s.t. $\|r - \omega a_j\|_1$ is minimized;
  $\hat{x}_j \leftarrow \hat{x}_j + \omega$;
  Perform $\hat{x} \leftarrow H_k[\hat{x}]$ every $(c-1)k$ iterations;
  $r \leftarrow y - A\hat{x}$;
end

Theorem 2.7 (SSMP [Berinde and Indyk, 2009])**.** *Let $A \in \mathbb{E}_{(c+1)k,\varepsilon,d}$ and let $y = Ax + \eta$ for $x \in \chi^n_k$. Then, there exists an $\varepsilon \ll 1$ such that SSMP with fixed $c > 1$ recovers $\hat{x} \in \mathbb{R}^n$ such that $\|x - \hat{x}\|_1 = \mathcal{O}(\|\eta\|_1)$. The algorithm terminates in $\mathcal{O}(k)$ iterations of complexity $\mathcal{O}\left(\frac{d^3 n}{m} + n + \left(\frac{n}{k}\log n\right)\log\|x\|_1\right)$.*

If $v \in \mathbb{R}^d$, let v^s a permutation of v satisfying $v^s_1 \leq \cdots \leq v^s_d$. The median of v is given by

$$\text{median}(v) = \begin{cases} \frac{1}{2}\left(v^s_{\frac{d}{2}} + v^s_{\frac{d}{2}+1}\right) & \text{If } d \text{ is even} \\ v^s_{\frac{(d+1)}{2}} & \text{if } d \text{ is odd} \end{cases}$$

The operation $\text{median}(r_{\mathcal{N}(j)})$ can be recast as the problem of finding the scalar $\omega \in \mathbb{R}$ that minimises $\|r - \omega a_j\|_1$ since the function

$$\|r - \omega a_j\|_1 = \sum_{i\in\mathcal{N}(j)} |r_i - \omega| + \text{constant} \tag{2.27}$$

is at a minimum when $|\{i \in \mathcal{N}(j) : r_i - \omega > 0\}| = |\{i \in \mathcal{N}(j) : r_i - \omega < 0\}|$. We elaborate on this claim in the following Lemma.

Lemma 2.8 (Optimality of the median)**.** *Let $r \in \mathbb{R}^m$ and A be the adjacency matrix of a left d-regular bipartite graph G. For $j \in [n]$, let a_j be the j-th column of A. Then,*

$$\arg\min_{\omega \in \mathbb{R}} \|r - \omega a_j\|_1 = \text{median}(r_{\mathcal{N}(j)}). \tag{2.28}$$

Proof. Let $r \in \mathbb{R}^m$ and $j \in [n]$. Define a mapping $f : \mathbb{R} \to \mathbb{R}$ by

$$f(\omega) = \|r - \omega a_j\|_1 \tag{2.29}$$

The function $f(\omega)$ is convex. To simplify notation let $\hat{r} := r_{\mathcal{N}(j)} \in \mathbb{R}^d$ and rewrite f as

$$\begin{aligned} f(\omega) &= \sum_i |r_i - \omega A_{i,j}| \\ &= \sum_{i \in \mathcal{N}(j)} |r_i - \omega| + \sum_{i \notin \mathcal{N}(j)} |r_i| \\ &= \sum_{i \in [d]} |\hat{r}_i - \omega| + \text{constant}. \end{aligned}$$

Let $\partial f(\omega)$ denote the subdifferential of f at $\omega \in \mathbb{R}$. It is a basic result in convex optimisation[9] that if $f(\omega)$ is convex, then ω^* is a global minimum of f if and only if $0 \in \partial f(\omega^*)$ [Nesterov, 2013, Theorem 3.1.15]. We prove (2.28) by showing that $0 \in \partial f(\omega^*)$.

For each $i \in [d]$, let $f_i(\omega) := |\hat{r}_i - \omega|$. Each $f_i(\omega)$ is closed and convex and has a subdifferential given by

$$\partial f_i(\omega) = \begin{cases} \{+1\} & \hat{r}_i - \omega < 0 \\ [-1, 1] & \hat{r}_i - \omega = 0 \\ \{-1\} & \hat{r}_i - \omega > 0 \end{cases} \tag{2.30}$$

It follows from [Nesterov, 2013, Lemma 3.1.9] that for $\omega \in \mathbb{R}$,

$$\partial f(\omega) = \sum_{i \in [d]} \partial f_i(\omega). \tag{2.31}$$

Let $\hat{r}^s$ be a permutation of $\hat{r}$ satisfying $\hat{r}^s_1 \leq \cdots \leq \hat{r}^s_d$. We consider two cases,

1. If d is even, let
$$L = \left\{1, \ldots, \frac{d}{2}\right\}, R = \left\{\frac{d}{2} + 1, \ldots, d\right\}.$$
By the definition of the median, $\hat{r}^s_i \leq \omega^*$ for $i \in L$, while $\omega^* \leq \hat{r}^s_i$ for $i \in R$. Hence, $+1 \in \partial f_i(\omega^*)$ for all $i \in L$ and $-1 \in \partial f_i(\omega^*)$ for all $i \in R$. Since $|L| = |R|$, it follows that $0 \in \partial f(\omega^*)$.

[9] We refer the reader to [Nesterov, 2013] for an introduction to convex optimisation.

2. If d is odd, let

$$L = \left\{1, \dots, \frac{d+1}{2} - 1\right\}, C = \left\{\frac{d+1}{2}\right\}, R = \left\{\frac{d+1}{2} + 1, \dots, d\right\}.$$

Again, by the definition of the median, $\hat{r}_i^s \leq \omega^*$ for $i \in L$, $\omega^* = r_i^s$ for $i \in C$, and that $\omega^* \leq \hat{r}_i^s$ for $i \in R$. Hence, $+1 \in \partial f_i(\omega^*)$ for all $i \in L$, $0 \in \partial f_i(\omega^*)$ for $i \in C$, and $-1 \in \partial f_i(\omega^*)$ for all $i \in R$. Since $|C| = 1$ and $|L| = |R|$, it follows that $0 \in \partial f(\omega^*)$.

□

We remark that Algorithm 8 needs to compute a score median$(r_{\mathcal{N}(j)})$ for each $j \in [n]$, which can be done at cost $\mathcal{O}(dn)$. It is important to note that they do not need to recompute all the scores at each iteration. A common strategy is to compute each of the scores once and store them with their corresponding node $j \in [n]$ in some data structure (like priority queues [Berinde and Indyk, 2009] or red-black trees [Jafarpour et al., 2009]). Then, at each iteration, we can efficiently request the node $j \in [n]$ that maximises the score (median, mode, etc.) and use it to update $\hat{x}_j$. This affects $d = |\mathcal{N}(j)|$ entries of the residual, so we only need to recompute the scores corresponding to $|\bigcup_{i \in \mathcal{N}(j)} \mathcal{N}(i)| = \mathcal{O}(d^2 n/m)$ right nodes. A similar observation holds for Algorithms 9 and 10 described below.

2.3.5 Left Degree Dependent Signal Recovery (LDDSR)

LDDSR was proposed in [Xu and Hassibi, 2007] and decodes by exploiting the unique neighbour property given in Lemma 2.5. The main insight is that one can lower bound the number of elements in $\{i \in [m] : y_i \in \text{argsupp}(x)\}$, and use the structure of A to find a $j \in [n]$ and a nonzero value $\omega \in \mathbb{R}$ that appears more than $d/2$ times in $r_{\mathcal{N}(j)} \in \mathbb{R}^d$. It is shown in [Xu and Hassibi, 2007] that updating $\hat{x}_j \leftarrow \hat{x}_j + \omega$ guarantees a decrease in $\|r\|_0$ when $\varepsilon = 1/4$. LDDSR and its theoretical guarantees are stated in Algorithm 9 and Theorem 2.9.

Theorem 2.9 (LDDSR [Xu and Hassibi, 2007]). *Let $A \in \mathbb{E}_{k,\varepsilon,d}^{m \times n}$ with $\varepsilon = 1/4$ and $x \in \chi_k^n$. Given $y = Ax$, LDDSR recovers x in at most $\mathcal{O}(dk)$ iterations with complexity $\mathcal{O}(\frac{d^3 n}{m} + n)$.*

Algorithm 9: LDDSR [Xu and Hassibi, 2007]

Data: $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$; $y \in \mathbb{R}^m$
Result: $\hat{x} \in \mathbb{R}^n$ s.t. $x = \hat{x}$
$\hat{x} \leftarrow 0$, $r \leftarrow y$;
while *not converged* **do**
  Find $(j,\omega) \in [n] \times \mathbb{R} \setminus \{0\}$ s.t. $|\{i \in \mathcal{N}(j) : r_i = \omega\}| > \frac{d}{2}$;
  $\hat{x}_j \leftarrow \hat{x}_j + \omega$;
  $r \leftarrow y - A\hat{x}$;
end

2.3.6 Expander Recovery (ER)

ER [Jafarpour et al., 2009] differs from LDDSR by considering $\varepsilon \leq 1/4$ and suitably adapting the set of indices from which an entry in $\hat{x}$ may be updated. This modification allows the number of iterations guaranteed to be improved, see Theorem 2.9. In particular, ER gurantees convergence in $\mathcal{O}(k)$ iterations of complexity $\mathcal{O}(nd)$. ER and its theoretical guarantees are stated in Algorithm 10 and Theorem 2.10.

Algorithm 10: ER [Jafarpour et al., 2009]

Data: $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$; $y \in \mathbb{R}^m$
Result: $\hat{x} \in \mathbb{R}^n$ s.t. $x = \hat{x}$
$\hat{x} \leftarrow 0$, $r \leftarrow y$;
while *not converged* **do**
  Find $(j,\omega) \in [n] \times \mathbb{R} \setminus \{0\}$ s.t. $|\{i \in \mathcal{N}(j) : r_i = \omega\}| \geq (1-2\varepsilon)d.$;
  $\hat{x}_j \leftarrow \hat{x}_j + \omega$;
  $r \leftarrow y - A\hat{x}$;
end

Theorem 2.10 (ER [Jafarpour et al., 2009])**.** *Let $A \in \mathbb{E}_{2k,\varepsilon,d}$ with $\varepsilon \leq 1/4$ and $m = \mathcal{O}(k\log(n/k))$. Then, for any $x \in \chi_k^n$, given $y = Ax$, ER recovers x in at most $\mathcal{O}(k)$ iterations of complexity $\mathcal{O}(\frac{d^3 n}{m} + n)$.*

Though ER seemingly requires knowledge of ε to implement, which is NP-hard to compute, knowledge of ε can be circumvented by selecting the node to update by

$$\arg\max_{j\in[n]} \left|\{i \in \mathcal{N}(j) : r_i = \mathrm{mode}(r_{\mathcal{N}(j)})\}\right|. \tag{2.32}$$

2.4 The dissociated model

Our contributions are made possible by assuming a *dissociated* signal model. The following definition makes this term precise for sparse signals.

Definition 2.11 (Dissociated signals)**.** *A signal $x \in \mathbb{R}^n$ is dissociated if*

$$\sum_{j \in T_1} x_j \neq \sum_{j \in T_2} x_j \qquad \forall\, T_1, T_2 \subset \operatorname{supp}(x) \text{ s.t. } T_1 \neq T_2. \tag{2.33}$$

The name *dissociated* comes from the field of additive combinatorics (see Definition 4.32 in [Tao and Vu, 2006]), where a set S is called *dissociated* if the set of all sums of distinct elements of S has maximal cardinality. The model (2.33) might seem restrictive. However, it need not be exactly fulfilled for our algorithms to work. In fact, it is fulfilled almost surely for isotropic signals, and more generally by any signal whose nonzeros can be modelled as being drawn from a continuous distribution. This dissociated signal model was previously used in the analysis of Sudocodes [Sarvotham et al., 2006b], see Section , an alternative algorithm which as here considers a compressed sensing model composed of a dissociated signal $x \in \chi_k^n$ and a sparse binary matrix $A \in \mathbb{E}_{k,\varepsilon,d}^{m \times n}$. Alternatively to uniform recovery guarantees for the dissociated signal model and binary matrices, one can consider probability-one guarantees for sparse signals and scaled binary matrices whose columns are scaled by random numbers drawn i.i.d. randomly from a continuous distribution independent of x. That is, by drawing a dissociated $z \in \mathbb{R}^n$ and considering the scaled matrix $\hat{A} \in \mathbb{R}^{m \times n}$ by drawing $A \in \{0,1\}^{m \times n}$ and defining

$$\hat{A} = \begin{bmatrix} | & | & & | \\ z_1 \cdot a_1 & z_2 \cdot a_2 & \dots & z_n \cdot a_n \\ | & | & & | \end{bmatrix} \tag{2.34}$$

For simplicity of exposition we present uniform results for the dissociated signals and binary matrices.

Dissociated signals have some useful properties. A key observation necessary in building our results is given in Lemma 2.12.

Lemma 2.12 (Properties of dissociated signals)**.** *Let $x \in \chi_k^n$ be dissociated. Then,*

1. $x_i \neq x_j \;\; \forall\, i, j \in \operatorname{supp}(x), \;\; i \neq j.$
2. $\sum_{j \in T} x_j \neq 0 \;\; \forall\, \emptyset \neq T \subset \operatorname{supp}(x).$

Proof. The result follows from (2.33). For (i) we set $T_1 = \{i\}$ and $T_2 = \{j\}$, and for (ii) we let $T_2 = \emptyset$. □

2.4.1 Preservation of Entropy

We now discuss the concept of dissociated signals under an Information Theory viewpoint. To do this, suppose that $(X_1, \ldots, X_k)$ is a vector of k random variables associated with $\{x_1, \ldots, x_k\} = \text{supp}(x)$ and that $(X_1, \ldots, X_k) \sim p$ for some distribution p supported on a finite set. Note that condition (iii) in Definition 2.11 implies that,

$$x_{i_1} + \cdots + x_{i_\ell} \neq x_{j_1} + \cdots + x_{j_\ell} \qquad \text{for} \qquad i_1 \neq j_1, \ldots, i_\ell \neq j_\ell. \tag{2.35}$$

Now, consider the following Shannon-entropy inequalities,

Lemma 2.13 (Entropy inequalities [Cover and Thomas, 2012] [Kelbert and Suhov, 2013])**.** *For a random variable $X \sim p$, let $H(\cdot)$ be its Shannon entropy. Now, let $X_1, \cdots, X_k$ be a set of random variables with joint distribution $(X_1, \ldots, X_k) \sim p$. Assume that the random variable X_i is supported on $\{(x_i)_1, \ldots, (x_i)_\ell\}$. Then,*

$$H(X_1 + \cdots + X_k) \leq H(X_1, \ldots, X_k) \leq H(X_1) + \cdots H(H_k) \tag{2.36}$$

With equality on the left if and only if $(x_1)_{i_1} + \cdots + (x_k)_{i_k} \neq (x_1)_{j_1} + \cdots + (x_k)_{j_k}$ for $i_l \neq j_l$, and equality on the right if and only if $X_i \perp X_j$ for $i \neq j$.

In the case of discretely supported distributions, a dissociated signal can be understood as one in which the entries on $\text{supp}(x)$ are drawn according to a distribution p fulfilling,

(i) $\Pr[X_i = \omega \mid X_j = \omega] = 0 \;\forall\; i \neq j \;\forall\; \omega \neq 0.$

(ii) $\Pr[\sum_{j \in T} X_j = 0] = 0 \;\forall\; T \subset [k]$

(iii) $\Pr[\sum_{j \in T_1} X_j = \sum_{j \in T_2} X_j] = 0 \;\forall\; T_1, T_2 \subset [k]$ with $T_1 \neq T_2$.

Property (iii) above, together with Lemma (2.13) say that probability distribution on the support of dissociated signals imply

$$H\left(\sum_{j \in T} X_j\right) = H(X_1, \ldots, X_k) \qquad \forall \qquad T \subset [k] \tag{2.37}$$

And since the value of each entry in $y = Ax$ is distributed according to $\sum_{j \in T} X_j$ for some $T \subset [k]$, this implies that linear transformations with expander matrices preserve the information in x.

2.4.2 Dissociated signals in previous art

The first algorithm to assume a dissociated strategy was Sudocodes [Sarvotham et al., 2006b]. Just as Algorithms 11 and 12, Sudocodes requires x to be dissociated, but exploits this property by focusing on the zeros in the measurement vector. Precisely, in that if the nonzeros in the signal are drawn from a continuous distribution, then $\sum_{j\in T} x_j \neq 0$ for all $T \subset \text{supp}(x)$ which implies that $x_j = 0$ whenever $y_i = 0$ and $A_{i,j} = 1$. Sudocodes consists of two phases. The first phase uses this observation to exclude possible locations of the nonzeros in x. This effectively reduces the number of columns in the sensing matrix and the length of x and y. The resulting reduced problem is then passed to a traditional compressed sensing algorithm in order to determine the final locations and values of the nonzeros in x, for complete details see [Sarvotham et al., 2006b]. Preliminary tests of applying the decoding algorithms presented in Chapter 3 to the reduced system resulting from the first phase of Sudocodes was observed to have a greater computational complexity than applying the algorithms directly to the original linear system of equations. More recently, [Ma et al., 2013] developed a noise-robust version of Sudocodes.

2.4.3 LDDSR/ER and the dissociated condition

A form of Lemma 2.5 is used in [Jafarpour et al., 2009, Lemma 3] to show that there is always a $j \in \text{supp}(x)$ and a nonzero value $\omega \in \mathbb{R}$ such that at least $(1-2\varepsilon)d$ entries of $r_{\mathcal{N}(j)} \in \mathbb{R}^d$ are identical to ω. This implies that at most $2\varepsilon d$ entries in $r_{\mathcal{N}(j)}$ are zero [Jafarpour et al., 2009, Lemma 4], which they use to show a contraction rate of

$$\begin{aligned} \|y - A(\hat{x} + \omega e_j)\|_0 &= \|(y - A\hat{x}) - \omega a_j\|_0 \\ &< \|y - A\hat{x}\|_0 - (1-2\varepsilon)d + (2\varepsilon d) \\ &= \|y - A\hat{x}\|_0 - (1-4\varepsilon)d. \end{aligned} \tag{2.38}$$

When assuming a dissociated model, there are no zeros in $(y - A\hat{x})_{\mathcal{N}(j)}$ so the contraction rate can be refined to

$$\|y - A(\hat{x} + \omega e_j)\|_0 < \|y - A\hat{x}\|_0 - (1-2\varepsilon)d. \tag{2.39}$$

The inequality (2.39) also applies to Algorithms 9 and 10 when the signal is dissociated.

		Objective	Score	Signal	$H_k(\cdot)$	Concurrency	Iterations	Iteration cost
Prior art	SMP [Berinde et al., 2008b]	ℓ_1	median	any	Yes	parallel	$\mathcal{O}(\log \|x\|_1)$	$\mathcal{O}(nd + n \log n)$
	SSMP [Berinde and Indyk, 2009]	ℓ_1	median	any	Yes	serial	$\mathcal{O}(k)$	$\mathcal{O}(\frac{d^3 n}{m} + n + \frac{n \log n}{k} \log \|x\|_1)$
	LDDSR [Xu and Hassibi, 2007]	ℓ_0	mode	any	No	serial	$\mathcal{O}(dk)$	$\mathcal{O}(\frac{d^3 n}{m} + n)$
	parallel-LDDSR	ℓ_0	mode	dissociated	No	parallel	$\mathcal{O}(\log k)$	$\mathcal{O}(nd)$
	ER [Jafarpour et al., 2009]	ℓ_0	mode	any	No	serial	$\mathcal{O}(k)$	$\mathcal{O}(\frac{d^3 n}{m} + n)$
Contributions	serial-ℓ_0	ℓ_0	$\|\cdot\|_0$	dissociated	No	serial	$\mathcal{O}(n \log k)$	$\mathcal{O}(d)$
	parallel-ℓ_0	ℓ_0	$\|\cdot\|_0$	dissociated	No	parallel	$\mathcal{O}(\log k)$	$\mathcal{O}(nd)$

Table 2.2: **Iterative greedy CCS algorithms in the noisless case.** A comparision between iterative greedy CCS algorithms in the noiseless case. Our contributions, Serial-ℓ_0 and Parallel-ℓ_0 greatly improve the worst-case computational complexity and when the strategy in LDDSR is parallelised, it inherits the same complexity guarantee when assuming a dissociated signal.

Chapter 3

Expander ℓ_0-decoding

Our first contribution is in the nexus of a series of papers [Jafarpour et al., 2009, Berinde et al., 2008b, Berinde and Indyk, 2009, Xu and Hassibi, 2007] proposing iterative greedy algorithms for combinatorial compressed sensing. As mentioned in Chapter 2, these algorithms seek to recover the sparsest solution of a large underdetermined linear system of equations $y = Ax + \eta$ by iteratively refining an estimation $\hat{x}$ using information about the residual $r = y - A\hat{x}$. As summarised in Section 2.3, the design of these algorithms has been optimised to best tradeoff speed, robustness, and recovery region. For instance, at each iteration, SMP [Berinde et al., 2008b] updates several entries of $\hat{x}$ in parallel, allowing it to provably recover an arbitrary $x \in \chi_k^n$ in $\mathcal{O}(\log \|x\|_1)$ iterations of complexity $\mathcal{O}(dn + n \log n)$. However, SMP is only able to recover the sparsest solution when the fraction of nonzeros in the signal is substantially less than other compressed sensing algorithms. On the other hand, at each iteration, LDDSR [Xu and Hassibi, 2007] and ER [Jafarpour et al., 2009] update a single entry of $\hat{x}$ in such a way that a contraction of $\|y - A\hat{x}\|_0$ is guaranteed. This reduction in the residual's sparsity is achieved by exploiting the *unique-neighbour* presented in Lemma 2.5. Essentially, Lemma 2.5 guarantees that most of the entries from x will appear repeatedly as entries in $y = Ax$ or that for most $i \in [m]$, we will have $y_i \in \{x_j : j \in \text{supp}(x)\}$. In [Xu and Hassibi, 2007] and [Jafarpour et al., 2009], this property is used to give sufficient conditions for decrease of $\|y - A\hat{x}\|_0$ under the regime of single updating of $\hat{x}$. However, this regime of single updating in LDDSR and ER typically requires greater computational time than existing compressed-sensing algorithms. In this chapter we present the design of an algorithmic model that successfully combines the coordinate selection strategy for

⋆ *The contributions of this chapter are joint work with Jared Tanner.*

updating $\hat{x}$ of LDDSR and ER with the parallel updating scheme of SMP and the signal structure assumed in Sudocodes. This synthesis is made possible by assuming a dissociated model in the signal or the matrix which aids in the identification of the columns associated with the nonzeros in the measured signal. In its simplest form, we can model this with a *dissociated* signal model, see Definition 2.11. We report in our numerical experiments that even though this condition can always be enforced, it need not be exactly fulfilled for our algorithms to work since we observe that our algorithms' phase transitions decrease gracefully as the dissociated property is lost by having a fraction of the nonzeros in x be equal, see Figure 3.9.

With this assumption, our contributions are a form of *model-based compressed sensing* [Baraniuk et al., 2010] in which apart from assuming $x \in \chi_k^n$, one also assumes special dependencies between the values of its nonzeros with the goal to improve the algorithm's speed or recovery ability. Our contributions are Serial-ℓ_0 and Parallel-ℓ_0, shown respectively in Algorithms 11, 12, and with convergence guarantees summarised in Theorem 3.1.

Algorithm 11: Serial-ℓ_0

Data: $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$; $y = Ax \in \mathbb{R}^m$ for $x \in \chi_k^n$; $\alpha \in (1, d]$
Result: $\hat{x} \in \chi_k^n$ s.t. $x = \hat{x}$
$\hat{x} \leftarrow 0$, $r \leftarrow y$;
while *not converged* **do**
  for $j \in [n]$ **do**
    $T \in \{\omega_j \in \mathbb{R} : \|r\|_0 - \|r - \omega_j a_j\|_0 \geq \alpha\}$;
    for $\omega_j \in T$ **do**
      $\hat{x}_j \leftarrow \hat{x}_j + \omega_j$;
    end
    $r \leftarrow y - A\hat{x}$;
  end
end

Theorem 3.1 (Convergence of Expander ℓ_0-Decoders)**.** *Let $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$, $\varepsilon \leq 1/4$ and $x \in \chi_k^n$ be a dissociated signal. Then, Serial-ℓ_0 and Parallel-ℓ_0 with $\alpha = (1-2\varepsilon)d$ can recover x from $y = Ax \in \mathbb{R}^m$ in $\mathcal{O}(dn \log k)$ operations.*

Serial-ℓ_0 is very similar to the *peeling-decoder* introduced by [Luby et al., 2001] for the Binary Erasure Channel, but differs from it in its ability to revisit nodes $j \in [n]$ once a nonzero value has been assigned. Having charted the development of the previous art in Chapter 2, we focus on proving Theorem 3.1. In doing so, we

Algorithm 12: Parallel-ℓ_0

Data: $A \in \mathbb{E}^{m\times n}_{k,\varepsilon,d}$; $y = Ax \in \mathbb{R}^m$ for $x \in \chi^n_k$; $\alpha \in (1, d]$
Result: $\hat{x} \in \chi^n_k$ s.t. $x = \hat{x}$
$\hat{x} \leftarrow 0$, $r \leftarrow y$;
while *not converged* **do**
  $T \leftarrow \{(j, \omega_j) \in [n] \times \mathbb{R} : \|r\|_0 - \|r - \omega_j a_j\|_0 \geq \alpha\}$;
  for $(j, \omega_j) \in T$ **do**
    $\hat{x}_j \leftarrow \hat{x}_j + \omega_j$;
  end
  $r \leftarrow y - A\hat{x}$;
end

contrast our algorithms to the state-of-the-art algorithms for compressed-sensing and show that when the signal is dissociated and $\eta = 0$, Serial-ℓ_0 and Parallel-ℓ_0 are the fastest algorithms available when implemented, respectively, in a serial or a parallel architecture.

3.1 Convergence of Expander ℓ_0-decoders

In this section we describe our first two contributions: Serial-ℓ_0 and Parallel-ℓ_0. These algorithms advance combinatorial compressed sensing by having comparatively high phase transitions while retaining the low computational complexity of SMP and the parallel implementation of LDDSR. In particular, Parallel-ℓ_0 is observed to typically recover the sparsest solution of underdetermined systems of equations in less time than any other compressed sensing algorithm when the signal is dissociated and the sensing matrix is an expander graph. Serial-ℓ_0 and Parallel-ℓ_0 look for a solution by identifying nodes which if updated sequentially would strictly reduce the $\|r\|_0$ by at least α. That is, they will choose a coordinate j of x, and an update value ω such that,

$$(j, \omega) \in [n] \times \mathbb{R} \quad \text{s.t.} \quad \|r\|_0 - \|r - \omega a_j\|_0 \geq \alpha, \tag{3.1}$$

for some $\alpha \in (1, d]$. By selecting a pair (j, ω) satisfying (3.1), Serial-ℓ_0 yields a decrease in $\|r\|_0$ at every update, and is guaranteed to converge in $\mathcal{O}(n \log k)$ iterations of computational complexity $\mathcal{O}(d)$ if the signal is dissociated. Parallel-ℓ_0 is designed similarly, but adapted to be able to take full advantage of modern massively parallel computational resources. Indeed, Parallel-ℓ_0 selects and updates all pairs (j, ω) satisfying (3.1) and updates these values in x in parallel. Under this updating scheme,

a strict contraction in $\|r\|_0$ is guaranteed at every iteration when the signal is dissociated and $\alpha = (1-2\varepsilon)d$ with $\varepsilon \leq 1/4$, though we show in Section 3.2 that one can fix $\alpha = 2$ and get high phase transitions and exceptional speed.

We begin by presenting the key technical lemmas that explain the behaviour of an iteration of Serial-ℓ_0 and Parallel-ℓ_0. In particular, technical lemmas are stated to show how often values in Ax appear when $x \in \chi_k^n$ and $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$, and that when a value in Ax appears sufficiently often it must be a value from x at a specified location. This property ensures the algorithm updates its approximation $\hat{x}$ with values x_j in the j^{th} entry, that is with the exact values from x at the correct locations.

Lemma 3.2 (Bounded frequency of values in expander measurements of dissociated signals)**.** *Let $x \in \chi_k^n$ be dissociated, $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$, and ω a nonzero value in Ax. Then, there is a unique set $T \subset \text{supp}(x)$ such that $\omega = \sum_{j\in T} x_j$ and the value ω occurs in y at most d times,*

$$|\{i \in [m] : y_i = \omega\}| \leq d \quad \forall\, \omega \neq 0. \tag{3.2}$$

Proof. The uniqueness of the set $T \subset \text{supp}(x)$ such that $\omega = \sum_{j\in T} x_j$ follows by the definition of dissociated. Since $|\mathcal{N}(j)| = d$ for all $j \in [n]$, we have that,

$$|\{i \in [m] : y_i = \omega\}| = \left|\bigcap_{j\in T} \mathcal{N}(j)\right| \leq |\mathcal{N}(j_0)| = d \tag{3.3}$$

for any $j_0 \in T$. □

Lemma 3.3 (Pairwise column overlap)**.** *Let $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$. If $\varepsilon \leq 1/4$, every pair of columns of A intersect in less than $(1-2\varepsilon)d$ rows, that is, for all $j_1, j_2 \in [n]$ with $j_1 \neq j_2$*

$$\left|\mathcal{N}(j_1)\bigcap\mathcal{N}(j_2)\right| < (1-2\varepsilon)d. \tag{3.4}$$

Proof. Let $S \subset [n]$ be such that $|S| = 2$ then

$$|\mathcal{N}(S)| > 2(1-\varepsilon)d \geq 2d - (1-2\varepsilon)d, \tag{3.5}$$

where the first inequality is Definition 2.2 and the second inequality follows from $\varepsilon \leq 1/4$. However, $|\mathcal{N}(S)|$ can be rewritten as

$$|\mathcal{N}(S)| = |\mathcal{N}(j_1)| + |\mathcal{N}(j_2)| - \left|\mathcal{N}(j_1)\bigcap\mathcal{N}(j_2)\right|, \tag{3.6}$$

for some $j_1, j_2 \in [n]$. Coupling (3.6) with (3.5) gives (3.4). □

Lemma 3.4 (Progress)**.** *Let $y = Ax$ for dissociated $x \in \chi_k^n$ and $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$ with $\varepsilon \leq 1/4$. There is a pair $(j,\omega) \in [n] \times \mathbb{R}$ such that*

$$|\{i \in \mathcal{N}(j) : y_i = \omega\}| \geq (1-2\varepsilon)d. \tag{3.7}$$

Proof. Let $S = \text{supp}(x)$, then by the unique neighbour property (2.18) it holds that $|\mathcal{N}_1(S)| > (1-2\varepsilon)d|S|$, where $\mathcal{N}_1(S)$ is defined in (2.16), or alternatively, by $\mathcal{N}_1(S) = \{i \in [m] : y_i = x_j, j \in S\}$ in the context of dissociated signals. Given the lower bound in $|\mathcal{N}_1(S)| > (1-2\varepsilon)d|S|$, if $|S| \neq 0$, at least one $j \in S$ must have at least $(1-2\varepsilon)d$ neighbours in y with identical nonzero entries. Letting ω take the value of such repeated nonzeros in y gives the required pair $(j,\omega) \in [n] \times \mathbb{R}$. □

Lemma 3.5 (Support identification)**.** *Let $y = Ax$ for dissociated $x \in \chi_k^n$ and $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$ with $\varepsilon \leq 1/4$. Let $\omega \neq 0$ be such that* (3.7) *holds. Then, $\omega = x_j$.*

Proof. Our claim is that for any ω which is a nonzero value from y, if the cardinality condition (3.7) is satisfied then the value $\omega = \sum_{j\in T} x_j$ occurs for the set T being a singleton, $|T| = 1$. Lemma 3.2 states that T is unique and that

$$|\{i \in \mathcal{N}(j) : y_i = \omega\}| = \left|\bigcap_{j\in T} \mathcal{N}(j)\right|. \tag{3.8}$$

If $|T| > 1$ then the above is not more than the cardinality of the intersection of any two of the sets $\mathcal{N}(j_1)$ and $\mathcal{N}(j_2)$, and by (3.4) in Lemma 3.3 that is less than $(1-2\varepsilon)d$ which contradicts the cardinality condition (3.7) and consequently $|T| \leq 1$. However, Lemma 3.4 guarantees that $|T| > 0$, so $|T| = 1$ and $\omega = x_j$. □

Equipped with Lemmas 2.12 - 3.5 we prove Theorem 3.1 considering Serial-ℓ_0 and Parallel-ℓ_0 separately, beginning with the later. Note that since $x \in \chi_k^n$ and the algorithm only sets entries in $\hat{x}$ to the correct values of x, then $x - \hat{x} \in \chi_k^n$, and Lemmas 3.4 and 3.5 hold with y replaced by $r = y - A(x - \hat{x})$.

Theorem 3.6 (Convergence of Parallel-ℓ_0)**.** *Let $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$ and let $\varepsilon \leq 1/4$, and $x \in \chi_k^n$ be dissociated. Then, Parallel-ℓ_0 with $\alpha = (1-2\varepsilon)d$ can recover x from $y = Ax \in \mathbb{R}^m$ in $\mathcal{O}(\log k)$ iterations of complexity $\mathcal{O}(dn)$.*

Proof. Let $\hat{x} = 0$ be our initial approximation to $x \in \chi_k^n$. During the ℓ^{th} iteration of Parallel-ℓ_0, let $S_\ell = \text{supp}(x - \hat{x})$ and include a subscript on the identification set $T = T_\ell \subset [n]$. As $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$ and $\varepsilon \leq 1/4$, by Lemma 3.5 and the required entry-wise reduction in the residual by at least $\alpha = (1-2\varepsilon)d$, it follows that Parallel-ℓ_0 only sets

entries in $\hat{x}$ to the correct values of x and as a result $\|x - \hat{x}\|_0 \leq \|x\|_0 = k$ for every iteration. This is consistent with the peeling algorithm in [Luby et al., 2001, Zhang and Pfister, 2012]. Moreover, by Lemma 3.4, the set $T_\ell \neq \emptyset$ as long as $x \neq \hat{x}$, so the algorithm eventually converges.

In fact, we show that the rate of reduction of $\|x - \hat{x}\|_0$ per iteration is by at least a fixed fraction $\frac{2\varepsilon d}{1+\lfloor 2\varepsilon d \rfloor}$. As $A \in \mathbb{E}^{m\times n}_{k,\varepsilon,d}$ has d nonzeros per column, the reduction in the cardinality of the residual, say $\|r^\ell\|_0 - \|r^{\ell+1}\|_0$, can be at most $d|T_\ell|$. That is,

$$\|r^\ell\|_0 - \|r^{\ell+1}\|_0 \leq d|T_\ell|. \tag{3.9}$$

To establish a fractional decrease in $|S_{\ell+1}|$ we develop a lower bound on $\|r^\ell\|_0 - \|r^{\ell+1}\|_0$. For $Q \subset S_\ell$ define the set $\mathcal{N}_1^{S_\ell}(Q)$ to be the set of nodes in $\mathcal{N}_1(S_\ell)$ and such that $i \in \mathcal{N}(j)$ for some $j \in Q$, *i.e.*

$$\mathcal{N}_1^{S_\ell}(Q) = \{i \in \mathcal{N}_1(S_\ell) : i \in \mathcal{N}(j), j \in Q\}. \tag{3.10}$$

Consider the partition $S_\ell = T_\ell \cup (S_\ell \setminus T_\ell)$ and rewrite $\mathcal{N}_1(S_\ell)$ as the disjoint union

$$\mathcal{N}_1(S_\ell) = \mathcal{N}_1^{S_\ell}(T_\ell) \cup \mathcal{N}_1^{S_\ell}(S_\ell \setminus T_\ell). \tag{3.11}$$

Note that $\mathcal{N}_1^{S_\ell}(T_\ell) \neq \mathcal{N}_1(T_\ell)$, and that by (3.10) and the dissociated signal model, $\mathcal{N}_1^{S_\ell}(T_\ell) \subset [m]$ is the set of indices in r^ℓ that are identical to a nonzero in x and that have a frequency of at least $\alpha = (1 - 2\varepsilon)d$, so

$$\|r^\ell\|_0 - \|r^{\ell+1}\|_0 \geq |\mathcal{N}_1^{S_\ell}(T_\ell)|. \tag{3.12}$$

At iteration ℓ, if $T_\ell = S_\ell$, the full support of x is correctly identified, so $x = \hat{x}$ after updating $\hat{x}$. Otherwise, $T_\ell \neq S_\ell$ and the set $S_\ell \setminus T_\ell$ is not identified by the algorithm at this iteration. We derive a lower bound on $|\mathcal{N}_1^{S_\ell}(T_\ell)|$ by considering two cases: $\alpha \in \mathbb{N}$ and $\alpha \notin \mathbb{N}$.

If $\alpha \in \mathbb{N}$, then each node in $S_\ell \setminus T_\ell$ has at most $\alpha - 1$ duplicates in r^ℓ, so

$$\left|\mathcal{N}_1^{S_\ell}(S_\ell \setminus T_\ell)\right| \leq (\alpha - 1)\,|S_\ell \setminus T_\ell|. \tag{3.13}$$

Using the the unique neighbour property (2.18) and the identity given in (3.10) it follows that

$$\begin{aligned}
&|\mathcal{N}_1^{S_\ell}(T_\ell)| + |\mathcal{N}_1^{S_\ell}(S_\ell \setminus T_\ell)| \\
&> (1 - 2\varepsilon)d\,(|T_\ell| + |S_\ell \setminus T_\ell|) \\
&= (1 - 2\varepsilon)d|T_\ell| + |S_\ell \setminus T_\ell| + (\alpha - 1)|S_\ell \setminus T_\ell|.
\end{aligned} \tag{3.14}$$

Now, using (3.13) to lower bound (3.14), and solving for $|\mathcal{N}_1^{S_\ell}(T_\ell)|$ gives

$$|\mathcal{N}_1^{S_\ell}(T_\ell)| \geq (1-2\varepsilon)d|T_\ell| + |S_\ell \setminus T_\ell|. \tag{3.15}$$

By coupling (3.15), (3.12), and (3.9) into a chain of inequalities it is seen that

$$(1-2\varepsilon)d|T_\ell| + (|S_\ell| - |T_\ell|) \leq d|T_\ell|, \tag{3.16}$$

which simplifies to

$$|T_\ell| \geq \frac{1}{1+2\varepsilon d}|S_\ell|. \tag{3.17}$$

If $\alpha \notin \mathbb{N}$, then each node in $S_\ell \setminus T_\ell$ has at most $\lfloor \alpha \rfloor$ duplicates in r^ℓ, so

$$\left|\mathcal{N}_1^{S_\ell}(S_\ell \setminus T_\ell)\right| \leq \lfloor \alpha \rfloor |S_\ell \setminus T_\ell|. \tag{3.18}$$

Similarly as in the former case, using (3.10) and the the unique neighbour property (2.16), we obtain

$$\begin{aligned} &|\mathcal{N}_1^{S_\ell}(T_\ell)| + |\mathcal{N}_1^{S_\ell}(S_\ell \setminus T_\ell)| \\ &> (1-2\varepsilon)d\left(|T_\ell| + |S_\ell \setminus T_\ell|\right) \\ &= (1-2\varepsilon)d|T_\ell| + (\alpha - \lfloor \alpha \rfloor)|S_\ell \setminus T_\ell| + \lfloor \alpha \rfloor |S_\ell \setminus T_\ell|. \end{aligned} \tag{3.19}$$

Just as in the previous case, (3.19) is bounded from below using (3.18), and the resulting inequality is used to get

$$|\mathcal{N}_1^{S_\ell}(T_\ell)| \geq (1-2\varepsilon)d|T_\ell| + (\alpha - \lfloor \alpha \rfloor)|S_\ell \setminus T_\ell|. \tag{3.20}$$

Inequalities (3.20), (3.12), and (3.9) are then used to derive

$$\alpha|T_\ell| + (\alpha - \lfloor \alpha \rfloor)(|S_\ell| - |T_\ell|) \leq d|T_\ell|. \tag{3.21}$$

It follows from $\alpha = (1-2\varepsilon)d \notin \mathbb{N}$ and the properties of step functions that

$$d - \lfloor (1-2\varepsilon)d \rfloor = -\lfloor -2\varepsilon d \rfloor = \lceil 2\varepsilon d \rceil = 1 + \lfloor 2\varepsilon d \rfloor, \tag{3.22}$$

and

$$(1-2\varepsilon)d - \lfloor (1-2\varepsilon)d \rfloor = 1 - 2\varepsilon d - \lfloor 2\varepsilon d \rfloor, \tag{3.23}$$

so (3.21) simplifies to

$$|T_\ell| \geq \frac{1 - 2\varepsilon d + \lfloor 2\varepsilon d \rfloor}{1 + \lfloor 2\varepsilon d \rfloor}|S_\ell|. \tag{3.24}$$

Finally, note that (3.24) reduces to (3.17) when $\alpha \in \mathbb{N}$, so using $S_{\ell+1} = S_\ell \setminus T_\ell$ and (3.24), we conclude that

$$|S_{\ell+1}| \leq \frac{2\varepsilon d}{1 + \lfloor 2\varepsilon d \rfloor} |S_\ell|. \tag{3.25}$$

Since $|S_0| = k$ it follows that Parallel-ℓ_0 will have converged after ℓ^* iterations when $k(2\varepsilon d/(1 + \lfloor 2\varepsilon d \rfloor))^{\ell^*} < 1$, which is achieved for

$$\ell^* \geq \left(\log^{-1} \left(\frac{1 + \lfloor 2\varepsilon d \rfloor}{2\varepsilon d} \right) \right) \log k. \tag{3.26}$$

Each iteration of Parallel-ℓ_0 involves computing (3.7) for each $j \in [n]$, which is equivalent to n instances of finding the mode of a vector of length d which can be solved in $\mathcal{O}(d)$ complexity provided $\alpha > \lfloor d/2 \rfloor$[Boyer and Moore, 1991]. □

Theorem 3.7 (Convergence of Serial-ℓ_0)**.** *Let $A \in \mathbb{E}^{m \times n}_{k,\varepsilon,d}$ and let $\varepsilon \leq 1/4$, and $x \in \chi^n_k$ be a dissociated signal. Then, Serial-ℓ_0 with $\alpha = (1 - 2\varepsilon)d$ can recover x from $y = Ax \in \mathbb{R}^m$ in $\mathcal{O}(n \log k)$ iterations with complexity $\mathcal{O}(d)$.*

Proof. The loop over $j \in [n]$ for Serial-ℓ_0 identifies singletons T to update values in $\hat{x}$ in serial. The union of the singletons for $j \in [n]$ includes the set of all nodes for which the residual would be reduced by at least α if one were to forgo the serial update in $\hat{x}$. For $\alpha = (1 - 2\varepsilon)d$, the proof of convergence for Theorem 3.6 establishes that this results in a reduction of the cardinality of $\operatorname{supp}(x - \hat{x})$ by at least a fraction $2\varepsilon d/(1 + \lfloor 2\varepsilon d \rfloor)$. That is, for p an integer, Serial-ℓ_0 satisfies

$$|\operatorname{supp}(x - \hat{x})| \leq k \left(\frac{2\varepsilon d}{1 + \lfloor 2\varepsilon d \rfloor} \right)^p \tag{3.27}$$

after $\ell = pn$ iterations, and converges to $\hat{x} = x$ after at most $p^* > \log(k)/\log((1 + \lfloor 2\varepsilon d \rfloor)/(2\varepsilon d))$ for convergence after

$$\ell^* \geq n \left(\log^{-1} \left(\frac{1 + \lfloor 2\varepsilon d \rfloor}{2\varepsilon d} \right) \right) \log k. \tag{3.28}$$

iterations. Each iteration of Serial-ℓ_0 involves computing the mode of a vector of length d and updating d entries in the residual. Since we are interested in knowing the mode of $r_{\mathcal{N}(j)}$ only when the most frequent element occurs more than $d/2$ times, this value can be found at cost $\mathcal{O}(d)$ [Boyer and Moore, 1991]. □

3.1.1 Shifted Parallel-ℓ_0 and Parallel-LDDSR

Evaluating (3.7) for a given column $j \in \text{supp}(x)$ is equivalent to finding the mode of $r_{\mathcal{N}(j)}$. This can be done at cost $\mathcal{O}(d)$ using the Boyer-Moore Majority vote algorithm [Boyer and Moore, 1991]. However, this algorithm requires that an element of the array occurs more than $\lfloor d/2 \rfloor$ times, so it might fail when we set $\alpha \in [\lfloor d/2 \rfloor]$. Our numerical experiments (Section 3.2) show that best recovery regions are obtained for $\alpha = 2$, so we prefer to have an algorithm with $\mathcal{O}(d)$ per-iteration cost for all $\alpha \in [d]$.

Our approach is presented in Algorithm 13. Instead of looking for an $\omega \in \mathbb{R}$ satisfying (3.7) for each $j \in [n]$, at the ℓ^{th} iteration we consider the reduction caused by ω_j, defined as the $\ell \bmod d$-th element in $r_{\mathcal{N}(j)}$. When using this shifting strategy we compromise the final number of iterations, but we also keep a fixed cost of d complexity per iteration for any $\alpha \in [d]$. The convergence guarantees of our algorithms when using this shifting strategy are presented in Theorem 3.8.

Algorithm 13: Computation of score for serial-ℓ_0 and parallel-ℓ_0.

Data: $j \in [n]$; $r \in \mathbb{R}^m$; $\omega \in \mathbb{N}$
Result: $s_j \leftarrow |\{i \in \mathcal{N}(j) : r_i = \omega\}|$

Theorem 3.8 (Convergence of Shifted Parallel-ℓ_0)**.** *Let $A \in \mathbb{E}^{m\times n}_{k,\varepsilon,d}$ with $\varepsilon \leq 1/4$, and $x \in \chi^n_k$ be dissociated. Then, the shifted versions of Serial-ℓ_0 and Parallel-ℓ_0 with $\alpha = (1-2\varepsilon)d$ can recover x from $y = Ax \in \mathbb{R}^m$ in an average of $\mathcal{O}(dn \log k)$ operations.*

Proof. Let $\hat{x} = 0$ be the initial approximation to $x \in \chi^n_k$, and $A \in \mathbb{E}^{m\times n}_{k,\varepsilon,d}$ with $\varepsilon \leq 1/4$. At ℓ^{th} iteration, let $T = T_\ell$ be the set satisfying (3.7), that is, the one that Parallel-ℓ_0 has marked for update. For $j \in T$, let ω_j be the most frequent element in $r_{\mathcal{N}(j)}$. In shifted-parallel-ℓ_0, ω_j is not directly computed. Instead, at iteration ℓ, the frequency of the $\ell \bmod d$-th value in $r_{\mathcal{N}(j)}$ is computed using Algorithm 13 and tested against the imposed threshold α. In the worst case, this increases the number of iterations by a factor $\mathcal{O}(d)$. However, on average, this is not the case, and convergence in $\mathcal{O}(\log k)$ iterations is guaranteed.

To see this, let $j \in T$ and let ω be drawn at random from $r_{\mathcal{N}(j)}$. Then, $\Pr(\omega = \omega_j) \geq 1 - 2\varepsilon$, so on average at iteration ℓ we will identify $|T_\ell|(1-2\varepsilon)$ correct entries in $\text{supp}(x - \hat{x})$. Given the bound for $|T_\ell|$ (3.24) in the proof of parallel-ℓ_0, we have that at each iteration we identify at least $(1-2\varepsilon)\frac{1-2\varepsilon d+\lfloor 2\varepsilon d \rfloor}{1+\lfloor 2\varepsilon d \rfloor}|S_\ell|$. Therefore

$$|S_{\ell+1}| \leq \left(\frac{(1-2\varepsilon)(1-2\varepsilon d + \lfloor 2\varepsilon d \rfloor)}{1 + \lfloor 2\varepsilon d \rfloor} \right) |S_\ell|. \tag{3.29}$$

□

When $\varepsilon = 1/4$, we have that $(1 - 2\varepsilon)d = d/2$, so we recover a parallel version of LDDSR (Algorithm 9) for dissociated signals. We call this algorithm Parallel-LDDSR, and we test its performance in Section 3.2.

3.2 Numerical Experiments

In this section we perform a series of numerical experiments[1] to compare Parallel-ℓ_0 and Serial-ℓ_0 with state-of-the-art compressed sensing algorithms. These comparisons are done by adding Parallel-ℓ_0 and Serial-ℓ_0 to the GAGA software package [Blanchard and Tanner, 2013] which includes `CUDA-C` implementations of a number of compressed sensing algorithms as well as a testing environment to rapidly generate synthetic problem instances. This approach allows us to solve hundreds of thousands of randomly generated problems and to solve problems with n in the millions.

Unless otherwise stated, all tests were performed with the nonzeros of x drawn from a standard normal distribution $\mathcal{N}(0, 1)$ and the parameter α in Serial-ℓ_0 and Parallel-ℓ_0 was set to 2.

3.2.1 Substantially higher phase transitions

As explained in Chapter 2, the phase transition of a compressed-sensing algorithm [Donoho and Tanner, 2010b] is the largest value k/m, which we denote $\rho^*(m/n)$ noting its dependence on m/n, for which the algorithm is typically (say greater than half of the instances) able recovery all k sparse vectors with $k < m\rho^*(m/n)$.

The value $\rho^*(m/n)$ often converges to a fixed value as n is increased with m/n being a fixed fraction. Figure 3.1 shows the phase transition curve for each of the CCS algorithms surveyed in Section 2.3, as well as Parallel-ℓ_0 and Serial-ℓ_0. To facilitate comparison with nonCCS algorithms, Figure 3.1 also includes the theoretical phase transition curve for ℓ_1-regularization for A drawn Gaussian [Donoho, 2006a, 2005], which is observed to be consistent [Donoho and Tanner, 2009a] with ℓ_1-regularization for $A \in \mathbb{E}^{m\times n}_{k,\varepsilon,d}$. The curves were computed by setting $n = 2^{18}$, $d = 7$, and a tolerance of 10^{-6}. The testing is done at $m = \delta_p n$ for

$$\delta_p \in \{0.02p : p \in [4]\} \cup \left\{0.1 + \frac{89}{1900}(p-1) : p \in [20]\right\}.$$

[1] Figures 3.1-3.8 were computed using a Linux machine with Intel Xeon E5-2643 CPUs @ 3.30 GHz, NVIDIA Tesla K10 GPUs, and executed from Matlab R2015a. Figures 4.6-3.10 were computed using a Linux machine with Intel Xeon E5-2667 v2 CPUs @ 3.30GHz, NVIDIA Tesla K40 GPUs, and executed from Matlab R2015a.

For each δ_p, we set $\rho = 0.01$ and generate 10 synthetic problems to be applied to the algorithms, with x having independent and identically distributed normal Gaussian entries. With this restrictions, our signals are dissociated. If at least one such problem was recovered successfully, we increase ρ by 0.01 and repeat the experiment. The recovery data is then fitted using a logistic function in the spirit of [Blanchard and Tanner, 2015] and the 50% recovery transition of the logistic function is computed and shown in Figure 3.1.

Note the low phase-transition curve of SMP and the substantially higher phase-transition curve of Parallel-ℓ_0 and Serial-ℓ_0. As mentioned previously, the multiple updating mechanism of SMP gives it sublinear convergence guarantees, but greatly compromises its region of recovery. We emphasise that the phase transition curves for Serial-ℓ_0 and Parallel-ℓ_0 are higher than those for SMP, SSMP, ER, and parallel-LDDSR. In particular, they are even higher than ℓ_1-regularization for $\delta \lessapprox 0.4$.

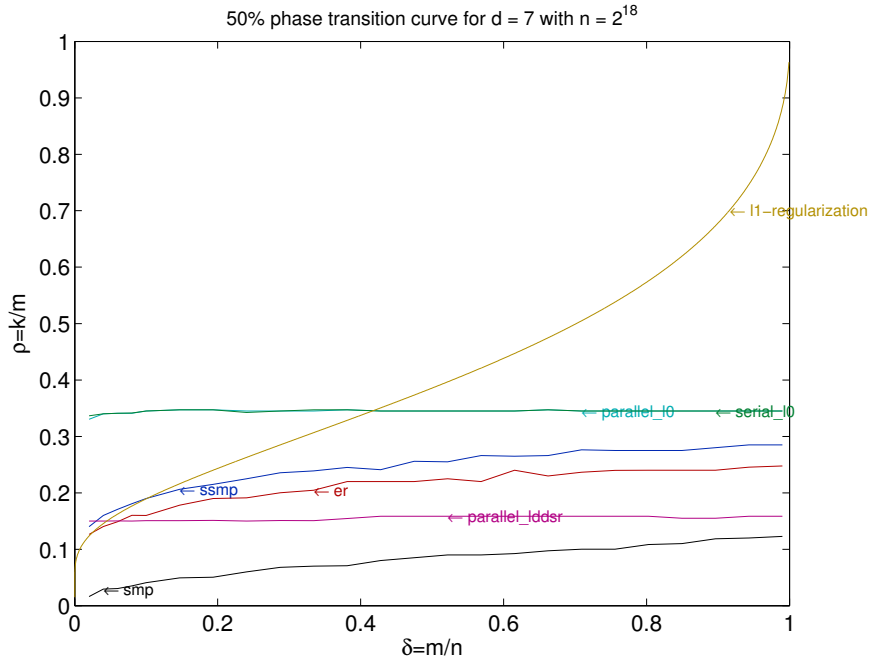


Figure 3.1: **Phase Transitions for CCS algorithms** 50% recovery probability logistic regression curves for $\mathbb{E}_{\varepsilon,7,k}$ and $n = 2^{18}$. The curve for ℓ_1-regularization is the theoretical curve for dense Gaussian ensembles, and is shown for reference. The expander-ℓ_0 algorithms achieve the highest phase transition for small δ.

3.2.2 Approximately flat phase transition

It is shown in Figure 3.1 that Serial-ℓ_0 and Parallel-ℓ_0 have a nearly constant phase transition of just over 0.3 even for very small values of δ. We hypothesise that this

flat phase transition persists for any fixed $\delta \in (0, 1)$ provided n is sufficiently large. Such a behaviour is reminiscent of Prony's method [de Prony, 1795], which has phase transition of 1/2 and 1/4 for Fourier and DCT matrices respectively [Dragotti and Lu, 2014]. Even though there are some adaptations of Prony's method for decoding k-sparse signals [Dragotti and Lu, 2014], they require finding the roots of a polynomial of degree n, which yields a complexity of $\mathcal{O}(n^3)$. This nearly constant phase transition of ℓ_0-decoding can be partially explained through the scaling analysis of LDPC codes presented in [Zhang and Pfister, 2012]. We provide numerical support of this claim in Figure 4.6, where for fixed $\delta = 10^{-3}$ and $d = 7$, we have plotted the average time to convergence for Parallel-ℓ_0 as ρ increases. The experiment was repeated for each $n \in \{2^{22}, 2^{24}, 2^{26}\}$, by initialising $\rho = 0.01$ and generating 30 problems at each ρ. If at least 50% of the problems converge we average out the time to convergence for successful cases, and perform the update $\rho \leftarrow \rho + 0.01$; otherwise, we stop. Our results in Figure 4.6 show that for $\delta = 10^{-3}$, the phase transition of the algorithm increases with n to just over 0.3.

Finally, in Table 3.1 we show the average timing depicted in Figure 4.6 for $\rho = 0.05$ which shows the approximate increase in the average computation time being proportional to n.

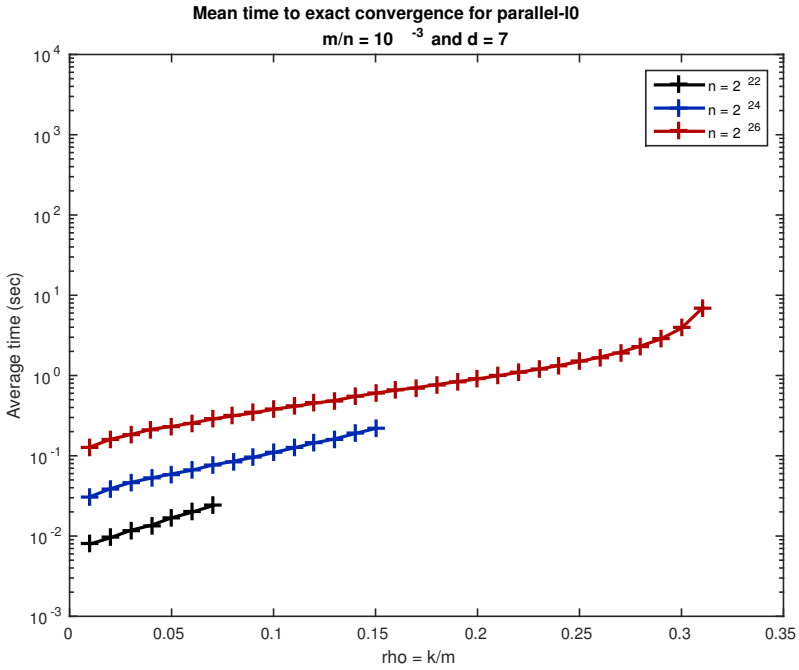


Figure 3.2: **Mean time to exact convergence for Parallel-ℓ_0.** Average recovery time (sec) for Parallel-ℓ_0, with dependence on ρ for $\delta = 0.001$ and $\mathbb{E}_{k,\varepsilon,7}$ with $n \in \{2^{22}, 2^{24}, 2^{26}\}$. The phase transition approaches ≈ 0.3 as $n \to \infty$.

n	time t_n	ratio t_{4n}/t_n
2^{22}	0.0167	3.338
2^{24}	0.0557	4.163
2^{26}	0.2319	-

Table 3.1: **Average time scaling for Parallel-ℓ_0.** Average recovery time (sec) for Parallel-ℓ_0 at $\rho = 0.05$ and $\delta = 10^{-3}$ for $n \in \{2^{22}, 2^{24}, 2^{26}\}$.

3.2.3 Fastest compressed sensing algorithm

When the signal is dissociated, Parallel-ℓ_0 is generally the fastest algorithm for matrices $A \in \mathbb{E}^{m\times n}_{k,\varepsilon,d}$. We show this numerically by computing the phase transitions of

Serial-ℓ_0, Parallel-ℓ_0, parallel-LDDSR, ALPS, CGIHT, CSMPSP, ER, FIHT, HTP, NIHT, SMP, SSMP;

and comparing their average time to convergence at each point of (δ, ρ). The phase transitions are computed similarly to those in Figure 3.1, with problem parameters of $n = 2^{18}$ and $d = 7$. In particular, Parallel-ℓ_0 is also used with $\alpha = 2$. The results are shown in Figures 3.3 and 3.4. Specifically, Figure 3.3 shows the time in milliseconds that the fastest algorithm takes to converge when the problem parameters are located at (δ, ρ). The fastest algorithm is in turn identified in Figure 3.4, where we can see that Parallel-ℓ_0 is consistently the fastest algorithm within its phase transition, except for $\rho \ll 1$ where parallel-LDDSR takes less time. However, we note that the convergence guarantees of parallel-LDDSR come as a byproduct of our analysis the domain in which it is faster than Parallel-ℓ_0 is the region of least importance for applications as it indicates more than three fold more measurements were taken than would have been necessary if Parallel-ℓ_0 were used.

3.2.4 Parallelisation brings important speedups: examples with $m \ll n$

The speed of Algorithms 7-10 can be improved if their non inner-product *scores* s_j and *updates* u_j are computed in parallel for each $j \in [n]$. However, implementing this parallelisation is not enough to cut down an algorithm's complexity to that of

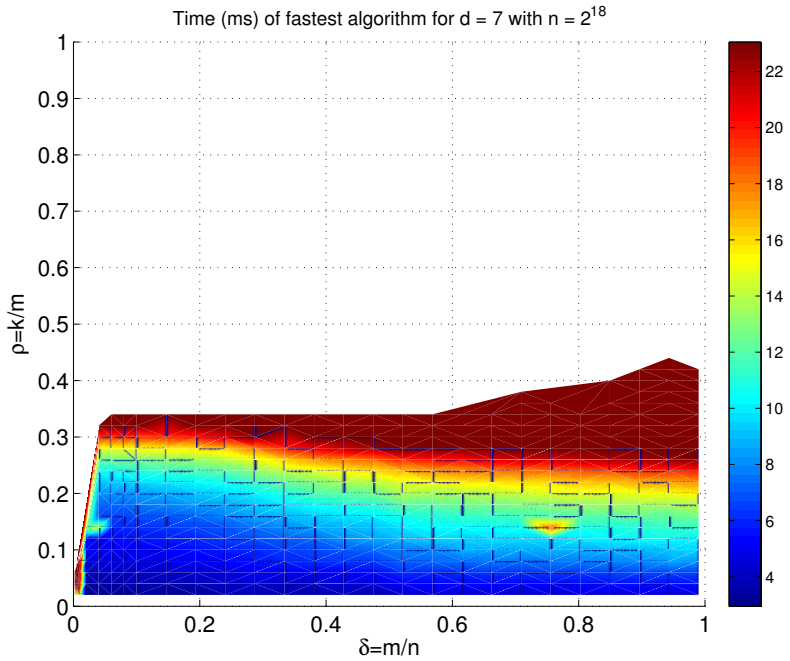


Figure 3.3: **Time (ms) of fastest algorithm.** Average recovery time (ms) of the fastest algorithm at each (δ, ρ) for $\mathbb{E}_{k,\varepsilon,7}$ and $n = 2^{18}$.

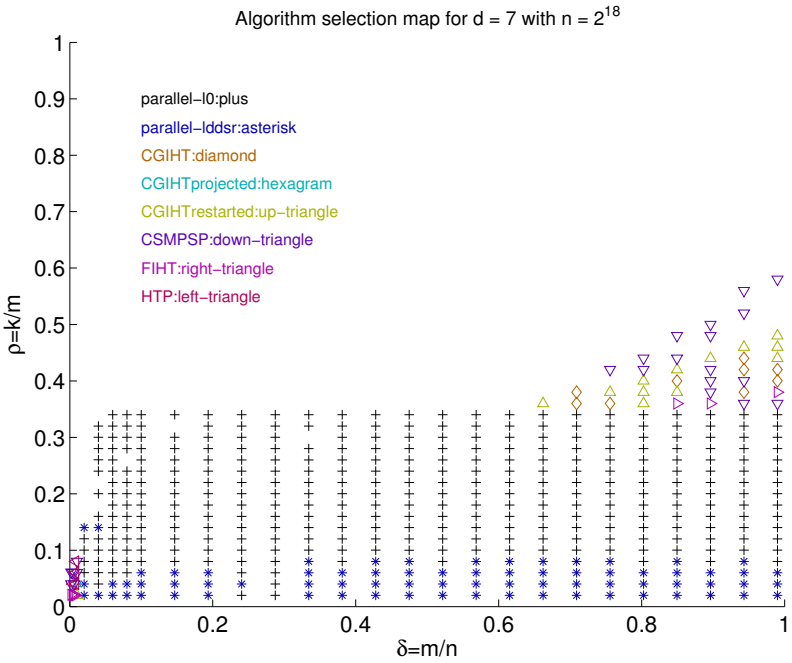


Figure 3.4: **Algorithm selection map.** Selection map of the fastest algorithm at each (δ, ρ) for $\mathbb{E}_{k,\varepsilon,7}$ and $n = 2^{18}$. The expander-ℓ_0 algorithms dominate the selection map except on the regions where they do not converge, $\delta \approx 1$.

the state-of-the-art's. Figures 3.5-3.6 show the average time to exact convergence for each of the combinatorial compressed sensing algorithms. It can be seen in addition to Serial-ℓ_0 and Parallel-ℓ_0 having higher phase transition than ER and SSMP, they are also substantially faster to converge to the true solution for $n = 2^{20}$ and either $\delta = 0.01$ or $\delta = 0.1$. It is interesting to note that for this problem size Serial-ℓ_0 is substantially faster than ER and SSMP, even when the two latter are implemented in parallel and run on a modern high performance computing GPU.

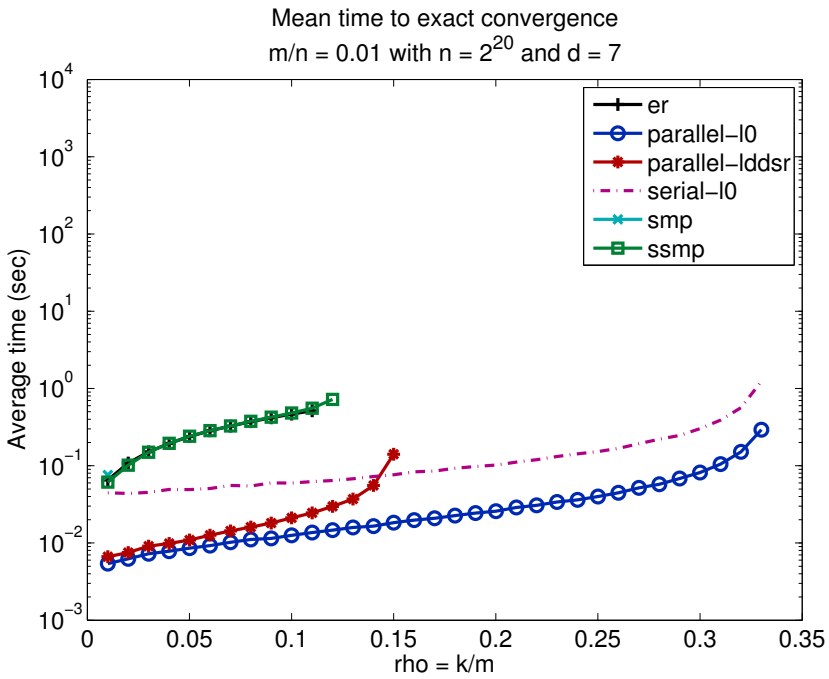


Figure 3.5: **Mean time to exact convergence for $\delta = 0.01$ and $n = 2^{20}$.** Average recovery time (sec) with dependence on ρ for $\delta = 0.01$ and $\mathbb{E}_{k,\varepsilon,7}$ with $n = 2^{20}$. The expander-ℓ_0 algorithms preserve their high phase transition under the extreme undersampling scenario $\delta = 0.01$.

3.2.5 Convergence in $\mathcal{O}(\log k)$ iterations

The theoretical guarantees of Serial-ℓ_0 and Parallel-ℓ_0 state that convergence can be achieved in $\mathcal{O}(nd \log k)$ operations. The number of operations per iteration can be verified simply by counting operations in the algorithm, which is $\mathcal{O}(d)$ for Serial-ℓ_0 and $\mathcal{O}(nd)$ for Parallel-ℓ_0 and recording the number of iterations. Figure 3.7 shows that the number of iterations to convergence for Serial-ℓ_0, Parallel-ℓ_0, and parallel-LDDSR. The tests were performed by fixing $n = 2^{20}$, $\delta = 0.1$, and $d = 7$, and considering signals with sparsity ranging from $\rho = 0.05$ to $\rho = 0.1$. It can be seen

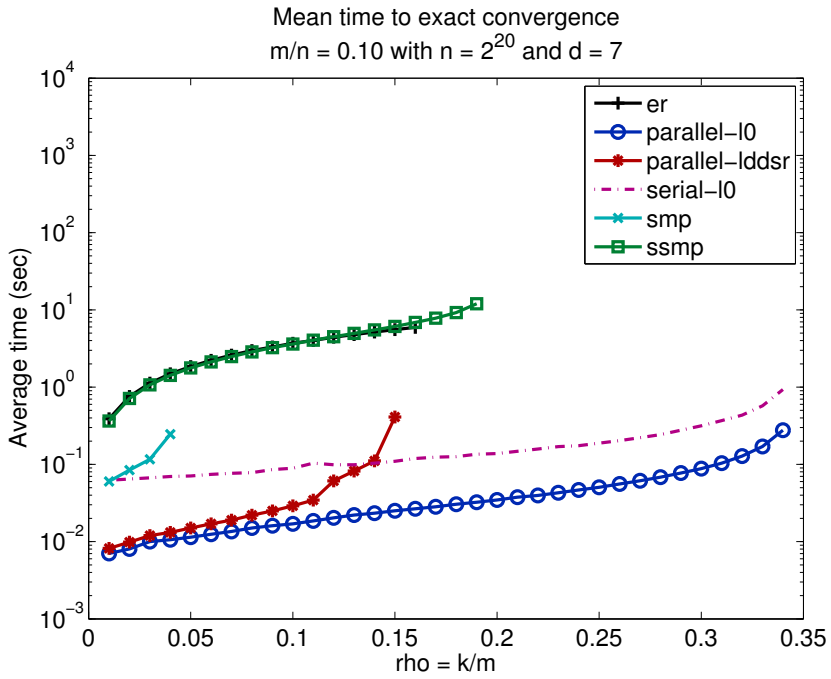


Figure 3.6: **Mean time to exact convergence for $\delta = 0.001$ and $n = 2^{20}$.** Average recovery time (sec) with dependence on ρ for $\delta = 0.1$ and $\mathbb{E}_{k,\varepsilon,7}$ with $n = 2^{20}$. The expander-ℓ_0 algorithms preserve their high phase transition under the extreme undersampling scenario fo $\delta = 0.001$.

in Figure 3.7 that the number of iterations to convergence is bounded by the curve $f(k) = \log k$, thus verifying our claims. We also make clear that by our notion of *iteration*, Serial-ℓ_0 is shown to converge in $\mathcal{O}(n \log k)$ iterations, but for the sake of this experiment, we normalise the final number of iterations for Serial-ℓ_0 by a factor of n. Note the lower number of iteration by Serial-ℓ_0 due to its serial implementation with residual updates revealing more entries that satisfy the reduction of the residual by α. Now, to give a point of comparison, we also compute the number of iterations for ER and SSMP, which take $\mathcal{O}(k)$ iterations to converge. The results are shown in Figure 3.8, where the same parameters as in Figure 3.7 have been used. In particular, we can see that for a problem with $k/m = 0.8$, Parallel-ℓ_0 takes 5 iterations, while ER and SSMP take about 8000 iterations to solve the same problem.

3.2.6 Almost dissociated signals

The analysis of Parallel-ℓ_0 and Serial-ℓ_0 relied on the model of dissociated signals (2.33). We explore the effect on recovery ability of Parallel-ℓ_0 and Serial-ℓ_0 as the signal model is no longer dissociated, with a fixed fraction of the values in x being

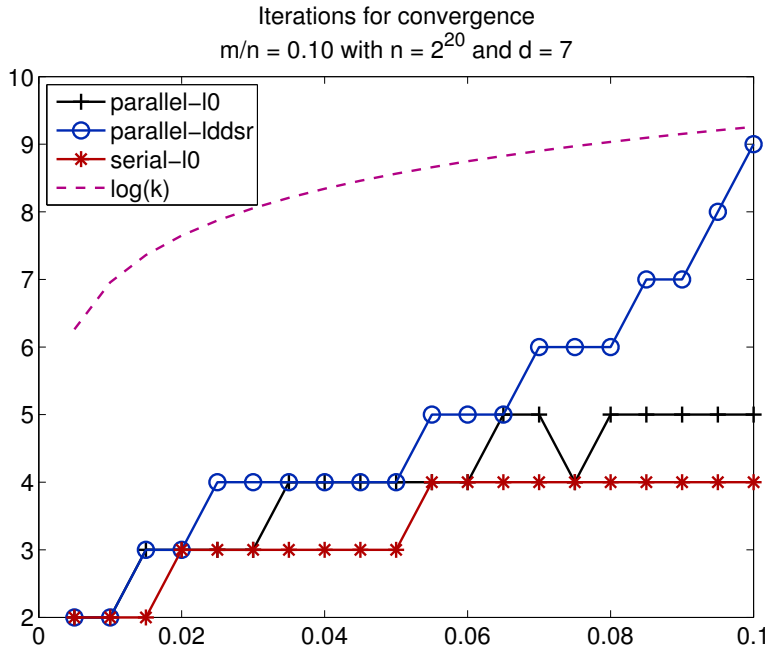


Figure 3.7: **Iterations vs ρ for sublinear algorithms.** Number of iterations to convergence for Parallel-ℓ_0, Serial-ℓ_0, and parallel-LDDSR at $\delta = 0.1$ with $\mathbb{E}_{k,\varepsilon,7}$ and $n = 2^{20}$. The number of iterations of Serial-ℓ_0 has been normalised by n to showcase its $\mathcal{O}(n \log k)$ guarantee in the number of iterations.

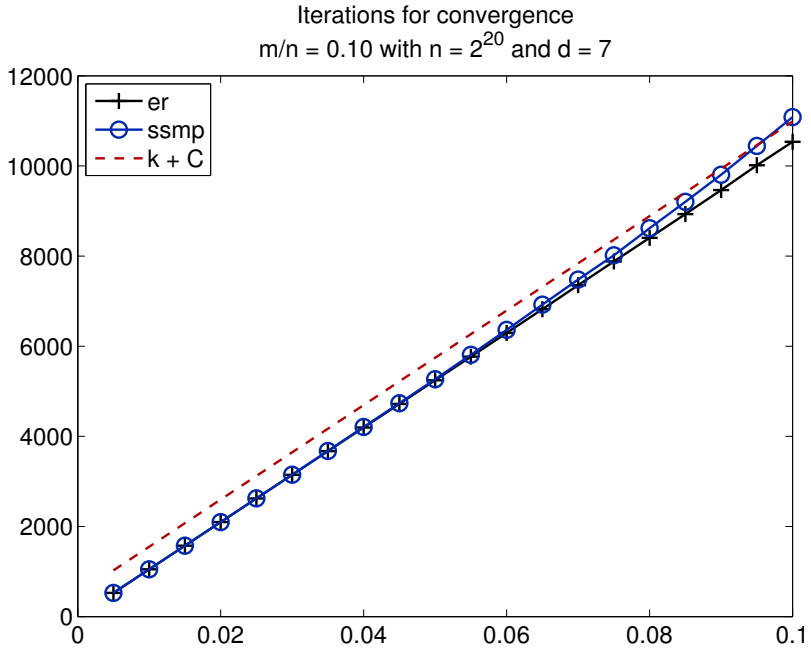


Figure 3.8: **Iterations vs ρ for linear algorithms.** Number of iterations to convergence for ER and SSMP at $\delta = 0.1$ with $\mathbb{E}_{k,\varepsilon,7}$ and $n = 2^{20}$.

equal. To do this, we consider signals $x \in \chi_k^n$ with nonzero values composed of two bands: one in which *all* entries are equal to a fixed value drawn at random from a standard normal distribution $\mathcal{N}(0,1)$, and another one in which *each* entry is drawn independently of each other from $\mathcal{N}(0,1)$. Our results are shown in Figure 3.9, where we can see that as the fraction of values which are equal increases (shown in the figure by the parameter *band*), the phase transitions gracefully decrease from the flat shape observed for perfectly dissociated signals to an increasing log-shaped curve when $band = 0.9$. Note that the overall phase transition decreases, with the greatest decrease for $\delta \ll 1$.

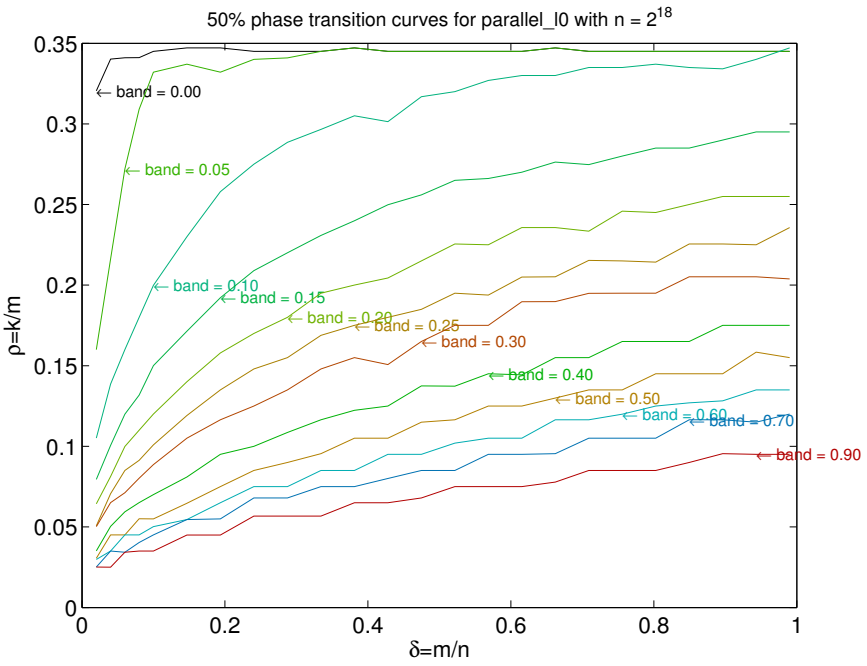


Figure 3.9: **Parallel-ℓ_0 with signal having a fixed proportion of identical nonzero elements in its support.** 50% recovery probability logistic regression curves for Parallel-ℓ_0 with $\mathbb{E}_{k,\varepsilon,7}$ and $n = 2^{18}$, with signals having a fixed proportion, *band*, of identical nonzero elements in its support. The phase transition of Parallel-ℓ_0 decrease gracefully as the dissociated condition fades in the signal.

3.2.7 d should be small, but not too small

Selection of the number of nonzeros per column, d, has not been adressed. In our numerical experiments we have consistently chosen $d = 7$ as the left-degree of our expander. Our choice of $d = 7$ for our problem size's order of magnitude is justified by Figure 3.10, where we have computed the phase transitions for Parallel-ℓ_0 for all odd values of d between 5 and 19. For $d = 5$, the phase transition of the algorithm

is very low, thus signalling expanders of bad quality. For $d = 7$ the phase transition is substantially greater than when $d = 5$, and gradually decreases for values of d greater than seven. Note that the expander condition implies $(1 - \varepsilon)dk < m$ which encourages small values of d in order that m/k can be as large as possible.

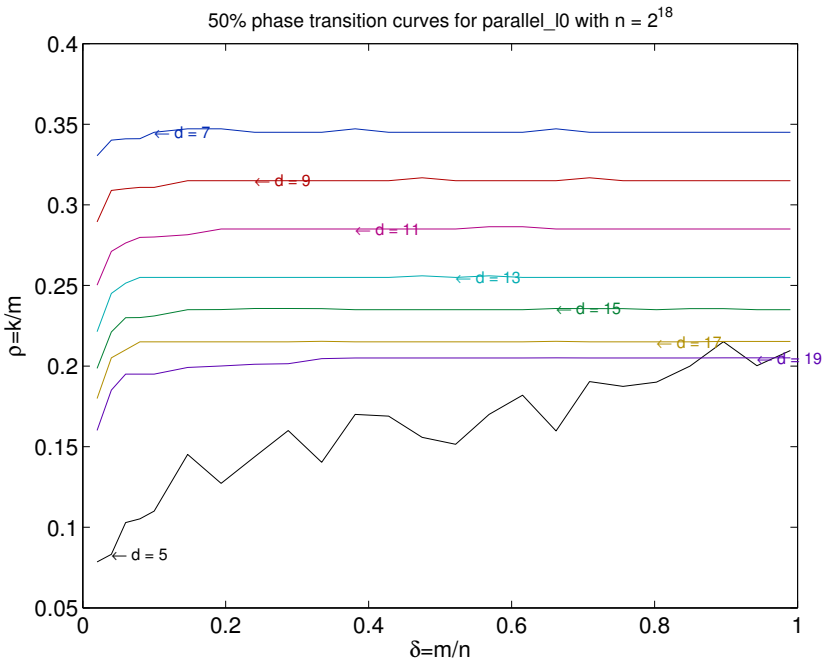


Figure 3.10: **Phase transitions of Parallel-ℓ_0 for several values of** d 50% recovery probability logistic regression curves for Parallel-ℓ_0 with $\mathbb{E}_{k,\varepsilon,d}$ and $n = 2^{18}$ for $d \in \{5, 7, 9, 11, 13, 15, 17, 19\}$. For the problem size under consideration, $d = 7$ yields the highest phase transition.

Chapter 4

Bayesian ℓ_0-decoding

Let $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$ and $\mathrm{nnz}(A)$ be the number of non-zeros in A. It was shown in Chapter 3 that a vector $x \in \mathbb{R}^n$ with at most $k < n$ nonzeros can be recovered from an expander sketch Ax in $\mathcal{O}(\mathrm{nnz}(A)\log k)$ operations via Parallel-ℓ_0 decoding algorithm. In this chapter we present the Robust-ℓ_0 decoding algorithm, which robustifies Parallel-ℓ_0 when the sketch Ax is corrupted by additive noise. This robustness is achieved by approximating the asymptotic posterior distribution of values in the sketch given its corrupted measurements via the Bayesian framework. We provide analytic expressions that approximate these posteriors under the assumptions that the nonzero entries in the signal and the noise are drawn from continuous distributions. Numerical experiments show that Robust-ℓ_0 is superior to existing greedy and combinatorial compressed sensing algorithms in the presence of small to moderate signal-to-noise ratios in the setting of Gaussian signals and Gaussian additive noise. It is shown in Chapter 3 that if $y = Ax$ is an expander sketch and $x \in \chi_k^n$ is dissociated, then there exists a subset $T \subset [n]$ such that, for each $j \in T$, $|\{i \in [m] : y_i = x_j\}|$ is bounded below by a positive constant depending on d and ε. Letting $a_j \in \mathbb{R}^m$ be the j-th column in A, this guarantees that if $|\{i \in \mathrm{supp}(a_j) : y_i = y_\ell\}| > d/2$ then $x_j = y_\ell$. Parallel-ℓ_0 (Algorithm 12) implements this observation by letting $\hat{x} = 0$ and estimating the decrease in $\|y\|_0$ when performing the update $\hat{x}_j \leftarrow \hat{x}_j + y_\ell$. We denote for $j \in [n]$ its neighbours by $\mathcal{N}(j) := \{i \in [m] : |A_{ij}| > 0\}$; to estimate the decrease in $\|y\|_0$, Parallel-ℓ_0 computes

$$n_e \leftarrow |\{\ell \in \mathcal{N}(j) : y_i = y_\ell\}|, \tag{4.1}$$

$$n_z \leftarrow |\{\ell \in \mathcal{N}(j) : y_\ell = 0\}|, \tag{4.2}$$

for each $j \in [n]$. Therefore, Parallel-ℓ_0 can be rewritten as Algorithm 14.

⋆ *The contributions of this chapter are joint work with Jared Tanner and Florian Wechsung.*

Algorithm 14: Parallel-ℓ_0 (rewritten)

Data: $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$; $y = Ax \in \mathbb{R}^m$ for $x \in \chi_k^n$; $\alpha \in (1, d]$
Result: $\hat{x} \in \mathbb{R}^n$ s.t. $\hat{x} = x$
$\hat{x} \leftarrow 0$, $r \leftarrow y$;
while *not converged* **do**
  for $j \in [n]$ **do**
    $u \leftarrow 0$;
    for $i \in \mathcal{N}(j)$ **do**
      if $r_i \neq 0$ **then**
        $n_e \leftarrow |\{\ell \in \mathcal{N}(j) : r_i = r_\ell\}|$;
        $n_z \leftarrow |\{\ell \in \mathcal{N}(j) : r_\ell = 0\}|$;
        if $n_e - n_z \geq \alpha$ **then**
          $u_j \leftarrow r_i$;
        end
      end
    end
  end
  $\hat{x} \leftarrow \hat{x} + u$;
  $r \leftarrow r - A\hat{x}$;
end

We robustify Parallel-ℓ_0 to the additive noise signal model of $\hat{y} = y+\eta$ with $y = Ax$ and $\eta \in \mathbb{R}^m$ by replacing (4.1)-(4.2) with scores that estimate the distribution of n_e and n_z in (4.8)-(4.9). That is, we follow a Bayesian approach to the computation of these scores and estimate:

1. The probability of $y_i = 0$ given that we observe $\hat{y}_i$.

$$p_z(\omega) := \mathbb{P}(y_i = 0 \mid \hat{y}_i = \omega). \tag{4.3}$$

2. The probability of $y_{i_1} = y_{i_2}$ given that we observe $\hat{y}_{i_1} - \hat{y}_{i_2}$.

$$p_e(\omega) := \mathbb{P}(y_{i_1} = y_{i_2} \mid \hat{y}_{i_1} - \hat{y}_{i_2} = \omega) \tag{4.4}$$

Among the contributions in this chapter are a series approximations for (4.3)-(4.4) when the signals and measurements are generated according to the generating model given in Definition 4.1 and illustrated in Figure 4.1. In what follows, we let $\mathbb{D}(\mathbb{R})$ be the set of distributions supported on $\mathbb{R}$. If $\mu \in \mathbb{D}(\mathbb{R})$, we use the notation $z \sim \mu$ to denote that z was drawn according to the distribution μ. We also use the notation $v_i \overset{\text{i.i.d.}}{\sim} \mu$ to denote that each v_i is drawn independently at random from μ. Finally, we use $U(S)$ to denote the uniform distribution over a set S.

Definition 4.1 (Generating model $\mathrm{GM}(n, m, k, d, \mu, \nu)$)**.** *Let $n, m, k, d \in \mathbb{N}$ be such that $k < m < n$ and $d \ll m$. Let $\mu, \nu \in \mathbb{D}(\mathbb{R})$. Then, the problem $(A, \hat{y})$ is drawn from the model $\mathrm{GM}(n, m, k, d, \mu, \nu)$ if $A \in \{0, 1\}^{m \times n}$ and $\hat{y} \in \mathbb{R}^m$ are such that*

1. *each column of A has a support drawn uniformly at random from $[m]^{(d)}$;*
2. *$\mathrm{supp}(x)$ is drawn uniformly at random from $[n]^{(k)}$;*
3. *$x_j \overset{i.i.d.}{\sim} \mu$ for each $j \in \mathrm{supp}(x)$;*
4. *$\eta_i \overset{i.i.d.}{\sim} \nu$ for each $i \in [m]$;*
5. *η_i is independent of x_j for all $i \in [m], j \in \mathrm{supp}(x)$;*
6. *$y = Ax$ and $\hat{y} = y + \eta$.*

We write $(A, \hat{y}) \sim \mathrm{GM}(n, m, k, d, \mu, \nu)$ to denote problem instances drawn from this signal model.

Remark 4.2. *It is important to note that a matrix $A \in \mathbb{R}^{m \times n}$ generated under the model presented in Definition 4.1 and Figure 4.1, is a (k, ε, d)-expander matrix with high probability, see [Bah and Tanner, 2013] and [Foucart and Rauhut, 2013, Theorem 13.6].*

The generating model in Definition 4.1 allows us to define robust estimates for (4.3)-(4.4) for general noise and signal distributions and to any degree of accuracy under the assumption μ and ν are available. While other similar generating models can be considered using the techniques presented here, for ease of exposition and clarity, we restrict our description to this model. From there we can define noisy analogues to the values n_e and n_z in (4.1)-(4.2) used in Parallel-ℓ_0 but which are robust to additive noise. The contributions of this chapter are

1. to present principled ways to compute (4.3)-(4.4) in the case where the nonzeros in η and x are drawn from continuous probability distributions,
2. to provide a variation of Parallel-ℓ_0 that is robust to noise.

We include Definition 4.3 in order to define some actions on measures used in our contributions.

Definition 4.3 (Measures [Klenke, 2013])**.** *Let $\mathcal{B}(\mathbb{R})$ denote the Borel σ-algebra over $\mathbb{R}$. If $E \in \mathcal{B}(\mathbb{R})$ let $-E := \{-x : x \in E\}$. Let $\mu \in \mathbb{D}(\mathbb{R})$, we define the following measures.*

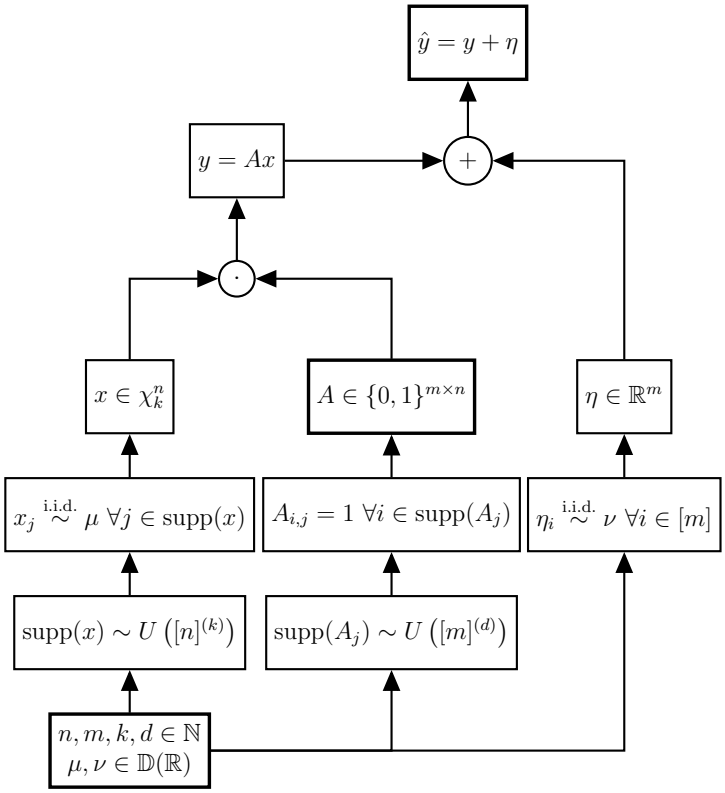


Figure 4.1: **Generating model for noisy measurements.** The generating model $\text{GM}(n, m, k, d, \mu, \nu)$ starts with a specification of the problem parameters and the noise and signal distributions and generates x, η, A independently.

1. *The q-convolution,*

$$\begin{aligned}\mu_0(E) = \delta_0(E) &= \begin{cases} 1 & 0 \in E \\ 0 & 0 \notin E \end{cases}, \forall E \in \mathcal{B}(\mathbb{R}) \\ \mu_1(E) &= \mu(E), \forall E \in \mathcal{B}(\mathbb{R}) \\ \mu_{q+1}(E) &= (\mu_q * \mu)(E), \forall E \in \mathcal{B}(\mathbb{R}), q \in \mathbb{N}\end{aligned}$$

2. *The negative measure,*

$$\mu^-(E) = \mu(-E), \quad \forall E \in \mathcal{B}(\mathbb{R})$$

3. *The symmetrized measure,*

$$\bar{\mu}(E) = \frac{\mu(E) + \mu(-E)}{2}, \quad \forall E \in \mathcal{B}(\mathbb{R})$$

4. *The measure associated with the difference of two random variables,*

$$\tilde{\mu}(E) = (\mu * \mu^-)(E), \quad \forall E \in \mathcal{B}(\mathbb{R}).$$

Among our main contributions of this chapter is Theorem 4.4.

Theorem 4.4 (Probabilities for general signal and noise distributions)**.** *Let $\delta, \rho \in (0,1)$. For each $n > 1$, let $m = \delta n$, $k = \rho m$ and $d < m$. If $\mu, \nu \in \mathbb{D}(\mathbb{R})$ and $(A, \hat{y}) \sim \mathrm{GM}(n, m, k, d, \mu, \nu)$. Then as $n \to \infty$,*

$$p_z(\omega) \to \frac{\nu(\omega)}{\sum_{q \geq 0} \frac{(d\rho)^q}{q!} (\nu * \mu_q)(\omega)}, \tag{4.5}$$

$$p_e(\omega) \to \frac{\tilde{\nu}(\omega)}{\sum_{q \geq 0} \frac{(2d\rho)^q}{q!} (\tilde{\nu} * \bar{\mu}_q)(\omega)}. \tag{4.6}$$

Where $\mu_q, \bar{\mu}_q, \tilde{\nu}$ are probability measures constructed as in Definition 4.3.

Equations (4.5) and (4.6) allows us to quantify the uncertainty associated with computing the score for Parallel-ℓ_0 under the presence of additive noise. Note that equations (4.5) and (4.6) can be easily adapted to alternative generative models, such as where the expected density of nonzeros per row varies, but for expository clarity we restrict our discussion to this somewhat generic model. It will be discussed in Section 4.1.3 that equations (4.5) and (4.6) should not be used directly, but instead be scaled by considering *normalised* functions $\breve{p}_e$ and $\breve{p}_z$ defined as,

$$\breve{p}_e(\omega) = \frac{p_e(\omega)}{\max_s p_e(s)}, \quad \breve{p}_z(\omega) = \frac{p_z(\omega)}{\max_s p_z(s)}. \tag{4.7}$$

In the most general case n_e and n_z can be written as the sum of individual scores q_e and q_z as follows

$$n_e \leftarrow \sum_{\ell \in \mathcal{N}(j)} q_e \left(r_{i_1} - r_{i_2} \mid t\right) \tag{4.8}$$

$$n_z \leftarrow \sum_{\ell \in \mathcal{N}(j)} q_z \left(r_i \mid t\right) \tag{4.9}$$

A confidence threshold $t > 0$ needs to be given for some variants of our algorithm, so to simplify the exposition we include this parameter in the scores $q_e(\cdot \mid t)$ and $q_z(\cdot \mid t)$ regardless of whether it is used or not. A summary of the score functions are given in Table 4.1.

	Robust-ℓ_0 Continuous	Robust-ℓ_0 Quantised	Parallel-ℓ_0
$q_e(r_{i_1} - r_{i_2} \mid t)$	$\breve{p}_e(r_{i_1} - r_{i_2})$	$\mathbb{1}\{\breve{p}_e(r_{i_1} - r_{i_2}) \geq t\}$	$\mathbb{1}\{r_{i_1} = r_{i_2}\}$
$q_z(r_i \mid t)$	$\breve{p}_z(r_i)$	$\mathbb{1}\{\breve{p}_z(r_i) \geq 1 - t\}$	$\mathbb{1}\{r_i = 0\}$

Table 4.1: **Comparison of scores for Expander ℓ_0-decoders.** Comparison of scores used in Expander ℓ_0-decoding vs Robust-ℓ_0 to identify candidate updates to the sparse signal $\hat{x}$.

To put this result into context and show its applicability, we recall the remark after Definition 4.1, stating that random matrices as considered in this work are indeed expander matrices with high probability.

We furthermore emphasize the fact that the algorithm is designed in a way that allows for massively parallel implementations.

4.1 The Bayesian approach to ℓ_0-decoding

We now consider the case where the measurements y are subject to additive noise, i.e. instead of $y = Ax$, we measure $\hat{y} = y + \eta$ where η is a realization of a random variable with $\eta_i \sim \nu$.

Parallel-ℓ_0 is not able to cope with additive noise, as it needs to make decisions whether a value in $\hat{y}$ is zero and whether two values in $\hat{y}$ are equal to each other.

Algorithm 15: Robust-ℓ_0

Data: $A \in \mathbb{E}^{m\times n}_{k,\varepsilon,d}$; $y = Ax \in \mathbb{R}^m$ for $x \in \chi^n_k$; $\alpha \in (1, d]$; $\mu, \nu \in \mathbb{D}(\mathbb{R})$; $c \in (0, 1)$
Result: $\hat{x} \in \mathbb{R}^n$ s.t. $\hat{x} \approx x$
Estimate $\breve{p}_z = \breve{p}_z(d, k, m, n, \mu, \nu)$ as in (4.7);
Estimate $\breve{p}_e = \breve{p}_e(d, k, m, n, \mu, \nu)$ as in (4.7);
if `quantised` **then**
  Use *quantised* scores given in Table 4.1.
else
  Use *continuous* scores given in Table 4.1.
end
$\hat{x} \leftarrow 0$;
$\hat{r} \leftarrow y$;
$t \leftarrow 1$;
while *not converged and* $t > 0$ **do**
  $x' \leftarrow \hat{x}$;
  $r \leftarrow \hat{r}$;
  for $j \in [n]$ **do**
    for $i \in \mathcal{N}(j)$ **do**
      if $1 - \breve{p}_z(r_i) \geq t$ **then**
        $n_e \leftarrow \sum_{\ell\in\mathcal{N}(j)} q_e(r_i - r_\ell \mid t)$;
        $n_z \leftarrow \sum_{\ell\in\mathcal{N}(j)} q_z(r_\ell \mid t)$;
        $\omega \leftarrow \frac{1}{n_e}\sum_{\ell\in\mathcal{N}(j)} r_\ell q_e(r_i - r_\ell \mid t)$;
        if $\|r - \omega e_i\|_1 \leq \|r\|_1$ *and* $n_e - n_z \geq \alpha$ **then**
          $x'_j \leftarrow x'_j + \omega$;
        end
      end
    end
  end
  $x' \leftarrow \mathcal{H}_k(x')$;
  $r \leftarrow y - Ax'$;
  $t \leftarrow t - c$;
  if $\|r\|_1 < \|\hat{r}\|_1$ **then**
    $\hat{x} \leftarrow x'$;
    $\hat{r} \leftarrow r$;
    if `adaptive_k` **then**
      $k_0 \leftarrow k - \sum_{j\in[n]} \breve{p}_z(\hat{x}_j)$;
      $k_0 \leftarrow \max(k_0, \lfloor \frac{m}{100} \rfloor)$;
      Recompute $\breve{p}_z = \breve{p}_z(k_0, m, n, \mu, \nu)$;
      Recompute $\breve{p}_e = \breve{p}_e(k_0, m, n, \mu, \nu)$;
    end
  end
end

While for very small noise levels we could consider two values as equal if they are within a certain number of standard deviations, for larger noise levels the decision becomes more challenging. As discussed in the previous section, we need to know $p_z(\hat{y}_i)$ and $p_e(\hat{y}_{i_1} - \hat{y}_{i_2})$ which correspond, respectively, to the probability of $y_i = 0$ given that we observe $\hat{y}_i$ and the probability of $y_{i_1} = y_{i_2}$ given that we observe $\hat{y}_{i_1} - \hat{y}_{i_2}$. The functions p_z and p_e depend on the parameters fed into the generative model in Definition 4.1 and in particular on the distribution of y and $\hat{y}$. Hence, since y is a vector of sparse inner products we need to understand the limiting behaviour of sparse sums.

Lemma 4.5 (Limiting distribution for sparse sums of random variables)**.** *Let $p \in (0,1)$, let $\mu \in \mathbb{D}(\mathbb{R})$ and let $\mu_q \in \mathbb{D}(\mathbb{R})$ be its the q-fold convolution. For each $n \geq 1$, let*

$$s_n := \sum_{j=1}^{n} b_j x_j \tag{4.10}$$

be such that,

1. *$x_j \overset{i.i.d.}{\sim} \mu$ for each $j \in [n]$,*
2. *$b_j \overset{i.i.d.}{\sim} \mathrm{Ber}(\frac{p_n}{n})$ for each $j \in [n]$ with $p_n \to p \in \mathbb{R}$ as $n \to \infty$.*

Then, as $n \to \infty$ it holds that $s_n \overset{(d)}{\to} s$ where

$$s \sim \exp(-p) \sum_{q \geq 0} \frac{p^q}{q!} \mu_q. \tag{4.11}$$

Proof. Let $\psi_{s_n}(t)$ be the characteristic function of s_n. Let $x \sim \mu$, then

$$\begin{aligned}
\psi_{s_n}(t) &= \mathbb{E}\left[\exp(its_n)\right] \\
&= \prod_{j=1}^{n} \mathbb{E}\left[\exp(itb_j x_j)\right] \\
&= \left(\left(1 - \frac{p_n}{n}\right) + \left(\frac{p_n}{n}\right)\psi_x(t)\right)^n \\
&= \left(1 + \frac{p_n(\psi_x(t) - 1)}{n}\right)^n.
\end{aligned}$$

Taking the limit $n \to \infty$ we see that

$$\lim_{n \to \infty} \psi_{s_n}(t) = \exp(-p)\exp(p\psi_x(t)). \tag{4.12}$$

Letting $w_q = \sum_{j=1}^{q} x_j$ it holds by the independence of $\{x_1, \ldots, x_q\}$ that $w_q \sim \mu_q$ and

$$[\psi_x(t)]^q = \psi_{w_q}(t). \tag{4.13}$$

Now, consider a random variable z distributed according to

$$z \sim \exp(-p) \sum_{q \geq 0} \frac{p^q}{q!} \mu_q. \tag{4.14}$$

The characteristic function of z is given by

$$\begin{aligned}
\psi_z(t) &= \mathbb{E}\left[\exp(itz)\right] \\
&= \exp(-p) \sum_{q \geq 0} \frac{p^q}{q!} \psi_{w_q}(t) \\
&= \exp(-p) \sum_{q \geq 0} \frac{p^q}{q!} \left(\psi_x(t)\right)^q \\
&= \exp(-p) \sum_{q \geq 0} \frac{\left(p\psi_x(t)\right)^q}{q!} \\
&= \exp(-p) \exp(p\psi_x(t)). \qquad (4.15)
\end{aligned}$$

Therefore (4.15) equals (4.12). By Lévy's continuity Theroem pointwise convergence of the characteristic functions implies weak convergence of the random variables (cf. [Klenke, 2013, Theorem 15.23]) and hence the statement follows. □

Theorem 4.6 (Distribution of $\hat{y}_i$ and $\hat{y}_{i_1} - \hat{y}_{i_2}$)**.** *Fix $\delta, \rho \in (0,1)$ and for $n \in \mathbb{N}$ let $m = \delta n$, $k = \rho m$ and $d \ll m$. Furthermore, let μ and ν be measures and assume that $\hat{y} = y + \eta = Ax + \eta$ is drawn from the model $\mathrm{GM}(n, m, k, d, \mu, \nu)$. Then as $n \to \infty$*

$$\hat{y}_i \xrightarrow{(d)} \hat{y}_i^* \quad \text{and} \quad \hat{y}_{i_1} - \hat{y}_{i_2} \xrightarrow{(d)} \hat{g}^*$$

where

$$\hat{y}_i^* \sim \exp(-d\rho) \sum_{q \geq 0} \frac{(d\rho)^q}{q!} \nu * \mu_q, \tag{4.16}$$

$$\hat{g}^* \sim \exp(-2d\rho) \sum_{q \geq 0} \frac{(2d\rho)^q}{q!} \tilde{\nu} * \bar{\mu}_q. \tag{4.17}$$

Proof. To show (4.16), let $i \in [m]$ and

$$y_i = \sum_{j=1}^{n} A_{i,j} x_j.$$

By our assumptions on A and x,

$$\begin{aligned}
\mathbb{P}\left(A_{i,j} x_j \neq 0\right) &= \mathbb{P}\left(A_{i,j} \neq 0 \wedge x_j \neq 0\right) \\
&= \mathbb{P}\left(A_{i,j} \neq 0\right) \mathbb{P}\left(x_j \neq 0\right) \\
&= \frac{d}{m} \frac{k}{n} \\
&= \frac{d\rho}{n}
\end{aligned}$$

where for two events E_1 and E_2 we let $E_1 \wedge E_2$ be the conjunction of the events. Note that if $A_{i,j}x_j \neq 0$, then $j \in \mathrm{supp}(x)$ so $A_{i,j}x_j = x_j \sim \mu$. Hence, letting $b_j \sim \mathrm{Ber}\left(\frac{d\rho}{n}\right)$,

$$y_i \overset{(d)}{=} \sum_{j=1}^{n} b_j x_j.$$

Invoking Lemma 4.5 with $p_n = p = d\rho$ we obtain that $y_i \to y_i^*$ as $n \to \infty$ where

$$y_i^* \sim \exp(-d\rho) \sum_{q \geq 0} \frac{(d\rho)^q}{q!} \mu_q.$$

By the independence of y_i^* and η_i, since $\hat{y}_i = y_i + \eta_i$, the distribution of $\hat{y}_i^*$ is given by

$$\hat{y}_i^* \sim \left(\exp(-d\rho) \sum_{q \geq 0} \frac{(d\rho)^q}{q!} \mu_q \right) * \nu. \tag{4.18}$$

Equation (4.16) follows from (4.18) and the distributivity of the convolution operator.

To show (4.17), let $i_1, i_2 \in [m]$ be such that $i_1 \neq i_2$ and let

$$y_{i_1} - y_{i_2} = \sum_{j=1}^{n} \left(A_{i_1,j} - A_{i_2,j}\right) x_j.$$

Similarly to the previous case, we compute

$$\begin{aligned}
\mathbb{P}\left(\left(A_{i_1,j} - A_{i_2,j}\right) x_j \neq 0\right) &= \mathbb{P}\left(A_{i_1,j} - A_{i_2,j} \neq 0 \wedge x_j \neq 0\right) \\
&= \mathbb{P}\left(A_{i_1,j} - A_{i_2,j} \neq 0\right) \mathbb{P}\left(x_j \neq 0\right) \\
&= 2\frac{d}{m} \frac{(m-1) - (d-1)}{m-1} \frac{k}{n} \\
&= \frac{2d\rho}{n}\left(1 - o(1)\right)
\end{aligned}$$

Note that if $(A_{i_1,j} - A_{i_2,j})x_j \neq 0$, then $j \in \mathrm{supp}(x)$ and $(A_{i_1,j} - A_{i_2,j})$ is either $+1$ with probability $\frac{1}{2}$ or -1 with probability $\frac{1}{2}$. Then, Hence,

$$(A_{i_1,j} - A_{i_2,j})x_j \sim \begin{cases} \mu & \text{with probability } \frac{1}{2}, \\ \mu^- & \text{with probability } \frac{1}{2}, \end{cases}$$

then,

$$(A_{i_1,j} - A_{i_2,j})x_j \sim \bar{\mu}.$$

Letting $b'_j \sim \mathrm{Ber}\left(\frac{2d\rho}{n}(1 - o(1))\right)$,

$$y_{i_1} - y_{i_2} \overset{(d)}{=} \sum_{j=1}^{n} b'_j x_j.$$

Again, invoking Lemma 4.5 with $p_n = 2d\rho(1 - o(1))$ we obtain that $p = 2d\rho$ and also that as $n \to \infty$, $y_{i_1} - y_{i_2} \to g^*$ with

$$y^*_{i_1} - y^*_{i_2} \sim \exp(-2d\rho) \sum_{q \geq 0} \frac{(2d\rho)^q}{q!} \bar{\mu}_q. \tag{4.19}$$

Therefore, Given that $\eta_{i_1} - \eta_{i_2} \sim \nu * \nu^-$ and that

$$\hat{y}_{i_1} - \hat{y}_{i_2} = (y_{i_1} - y_{i_2}) + (\eta_{i_1} - \eta_{i_2}),$$

we convolve (4.19) with $\nu * \nu^-$ to recover (4.17). □

We are now ready to prove Theorem 4.4,

Proof. Theorem 4.4. Using Bayes rule we write

$$\mathbb{P}(y_i = 0 | \hat{y}_i = \omega) = \frac{\mathbb{P}(\hat{y}_i = \omega \wedge y_i = 0)}{\mathbb{P}(\hat{y}_i = \omega)}. \tag{4.20}$$

From (4.16) we can deduce that

$$\mathbb{P}(\hat{y}_i = \omega \wedge y_i = 0) = \exp(-d\rho)\nu(\omega) \tag{4.21}$$

and using equation (4.16) from Theorem 4.6 we obtain that as $n \to \infty$,

$$\mathbb{P}(\hat{y}_i = \omega) \to \exp(-d\rho) \sum_{q \geq 0} \frac{(d\rho)^q}{q!} (\nu * \mu_q)(\omega). \tag{4.22}$$

Coupling (4.20), (4.22) and (4.21) yields (4.5).

Again, by Bayes rule,

$$\mathbb{P}\left(y_{i_1} = y_{i_2} | \hat{y}_{i_1} - \hat{y}_{i_2} = \omega\right) = \frac{\mathbb{P}\left(y_{i_1} = y_{i_2} \wedge \hat{y}_{i_1} - \hat{y}_{i_2} = \omega\right)}{\mathbb{P}\left(\hat{y}_{i_1} - \hat{y}_{i_2} = \omega\right)}. \tag{4.23}$$

Noting that $\hat{y}_{i_1} - \hat{y}_{i_2} = (y_{i_1} - y_{i_2}) + (\eta_{i_1} - \eta_{i_2})$,

$$\mathbb{P}\left(y_{i_1} = y_{i_2} \wedge \hat{y}_{i_1} - \hat{y}_{i_2} = \omega\right) = \tilde{\nu}(\omega) \tag{4.24}$$

By (4.16) from Theorem 4.6 we obtain that as $n \to \infty$,

$$\mathbb{P}(\hat{y}_{i_1} - \hat{y}_{i_2} = \omega) \to \exp(-2d\rho) \sum_{q \geq 0} \frac{(2d\rho)^q}{q!} \tilde{\nu} * \bar{\mu}_q(\omega) \tag{4.25}$$

Coupling (4.23), (4.24), and (4.25) yields (4.6).

□

4.1.1 Explicit formulas for the centred Gaussian case

We further elucidate (4.5)-(4.6) from Theorem 4.4 in the case when μ and ν are Gaussian with mean zero, which are the distributions considered in Section 4.2. To this end, let $\mu = \mathcal{N}(0, \sigma_s^2)$ and $\nu = \mathcal{N}(0, \sigma_n^2)$. We observe that in this case $\bar{\mu} = \mu$, $\mu_q = \bar{\mu}_q = \mathcal{N}(0, q\sigma_s^2)$ and $\tilde{\nu} = \mathcal{N}(0, 2\sigma_n^2)$. Hence, denoting by $\varphi(\cdot \mid \sigma^2)$ the probability density function of a centred Gaussian random variable with variance σ^2, we obtain

$$p_z(\omega) \to \frac{\varphi(\omega \mid \sigma_n^2)}{\sum_{q\geq 0} \frac{(d\rho)^q}{q!}\varphi(\omega \mid q\sigma_s^2 + \sigma_n^2)} \tag{4.26}$$

$$p_e(\omega) \to \frac{\varphi(\omega \mid 2\sigma_n^2)}{\sum_{q\geq 0} \frac{(2d\rho)^q}{q!}\varphi(\omega \mid q\sigma_s^2 + 2\sigma_n^2)}. \tag{4.27}$$

4.1.1.1 Estimating the tails in the Gaussian case

We approximate the infinite sum in in the denominators of (4.26) and (4.27) by doing an approximation to the tail of this summation.

Lemma 4.7 (Sums of centred density functions)**.** *Let μ_i be the density function of a random varible with mean zero and variance σ_i^2 and let $\alpha_i > 0$ be such that $\sum_i \alpha_i = 1$. Then, $\mu = \sum_i \alpha_i \mu_i$ is the density function of a random variable with mean zero and variance $\sum_i \alpha_i \sigma_i^2$.*

Proof. Let $x \sim \mu$ and $x_i \sim \mu_i$ be such that $\mathbb{E}[x_i] = 0$ and $\mathbb{V}ar[x_i] = \sigma_i^2$.

$$\begin{aligned}
\mathbb{V}ar[x] &= \mathbb{E}[x^2] \\
&= \int \omega^2 \mu(\omega) d\omega \\
&= \int \omega^2 \left(\sum_i \alpha_i \mu_i(\omega) \right) d\omega \\
&= \sum_i \alpha_i \int \omega^2 \mu_i(\omega) d\omega \\
&= \sum_i \alpha_i \sigma_i^2
\end{aligned}$$

□

To simplify notation let $\sigma_q^2 = q\sigma_s^2 + \sigma_n$ for $q \in \mathbb{N} \cup \{0\}$. Consider the series in the denominator of (4.22)

$$S(\omega) := \sum_{q=0}^{\ell} \frac{(d\rho)^q}{q!}\varphi(\omega \mid \sigma_q^2) + \sum_{q=\ell+1}^{\infty} \frac{(d\rho)^q}{q!}\varphi(\omega \mid \sigma_q^2).$$

Let,

$$R_z(\ell) := \exp(d\rho) - \sum_{q=0}^{\ell} \frac{(d\rho)^q}{q!}$$

and write

$$S_a(\omega) := \sum_{q=0}^{\ell} \frac{(d\rho)^q}{q!} \varphi(\omega \mid \sigma_q^2),$$

$$S_b(\omega) := \sum_{q=\ell+1}^{\infty} \frac{(d\rho)^q}{q!} \varphi(\omega \mid \sigma_q^2).$$

Note that S_b/R_z satisfies the conditions of Lemma 4.7 so it corresponds to the density function of a centred random variable with variance

$$\begin{aligned}\sigma_z^2 &= \frac{1}{R_z} \sum_{q=\ell+1}^{\ell} \frac{(d\rho)^q}{q!} (q\sigma_s^2 + \sigma_n^2) \\ &= \frac{\sigma_s^2 (d\rho) R_z(\ell-1) + \sigma_n^2 R_z(\ell)}{R_z(\ell)}\end{aligned}$$

Therefore,

$$p_z(\omega) \approx \frac{\varphi(\omega \mid \sigma_n^2)}{\sum_{q=0}^{\ell} \frac{(d\rho)^q}{q!} \varphi(\omega \mid \sigma_q^2) + R_z(\ell) \varphi(\omega \mid \sigma_z^2)}. \tag{4.28}$$

A similar argument shows that

$$p_e(\omega) \approx \frac{\varphi(\omega \mid 2\sigma_n^2)}{\sum_{q=0}^{\ell} \frac{(2d\rho)^q}{q!} \varphi(\omega \mid \sigma_{q,e}^2) + R_e(\ell) \varphi(\omega \mid \sigma_e^2)}. \tag{4.29}$$

Where $\sigma_{q,e}^2 = q\sigma_s^2 + \sigma_n^2$, and

$$\begin{aligned}R_e(\ell) &= \exp(2d\rho) - \sum_{q=0}^{\ell} \frac{(d\rho)^q}{q!}, \\ \sigma_e^2 &= \frac{\sigma_s^2 (2d\rho) R_e(\ell-1) + 2\sigma_n^2 R_e(\ell)}{R_e(\ell)}.\end{aligned}$$

4.1.2 Comparison with empirical probabilities

We test the approximations given in (4.28) and (4.29) by randomly generating $\hat{y}$ and y according the generating models $\mathrm{GM}(n, \delta n, \rho\delta n, 7, \mathcal{N}(0,1), \mathcal{N}(0,\sigma^2))$, for $\delta = 0.3$, $\rho \in \{0.1, 0.3\}$ and $\sigma \in \{10^{-3}, 10^{-2}\}$. The results can be seen in Figure 4.2.

Overall the analytical expressions fit the empirical probabilities very well, indicating that the approximations we made in the calculations above are justified. However, we observe that as ρ and especially σ increase, both functions drop significantly. This means that for large values of these parameters, the noise eventually dominates and it is difficult to decided whether values are zero or equal.

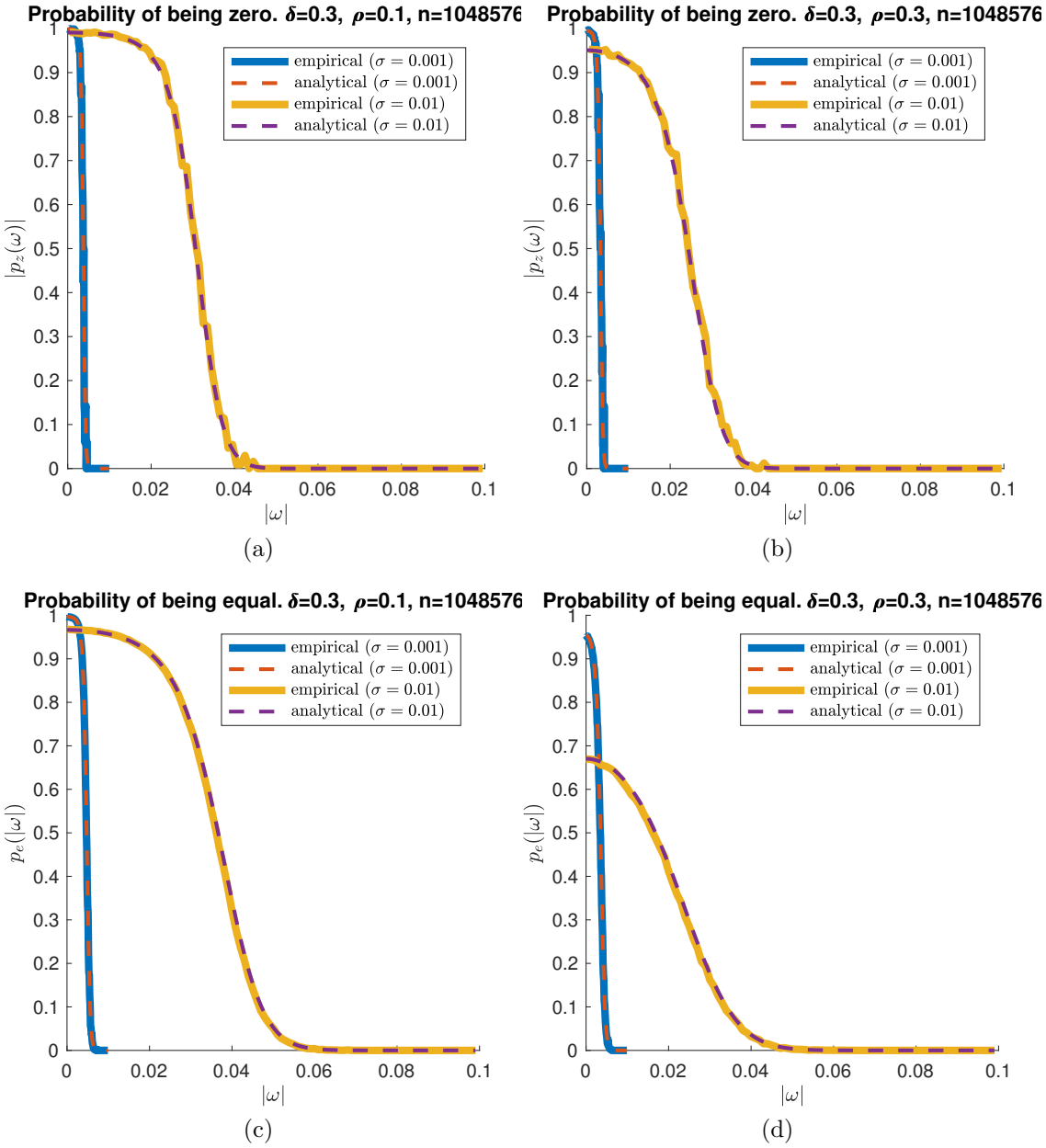


Figure 4.2: **Comparison of empirical and analytical probabilities.** Comparison of analytical and empirical probabilities of a value in the residual being zero or two values in the residual being equal for $\rho \in \{0.1, 0.3\}$ and $\sigma_n \in \{10^{-3}, 10^{-2}\}$.

4.1.3 Scaled probabilities $\breve{p}_z$ and $\breve{p}_e$

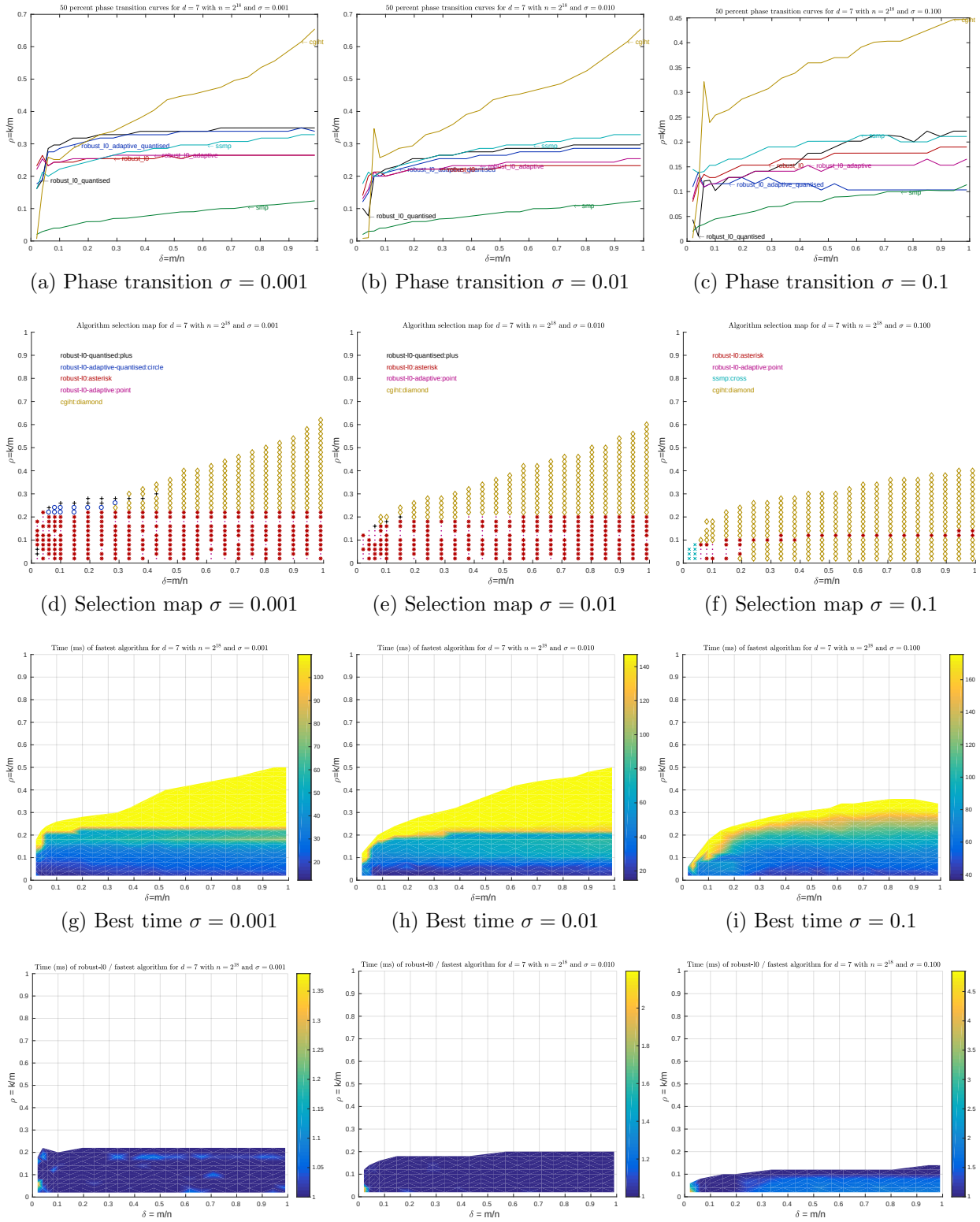
We mentioned in previous sections, our algorithms do not implement the functions p_z and p_e exactly, but a scaled version of these. As observed in Figure 4.2, the value of $\max_s p_e(s)$ and $\max_s p_z(s)$ varies significantly as σ and ρ change. Algorithm 15 evaluates whether a given score is large or not by implementing a sweeping parameter t that is set to one at the beginning of the algorithm and decreased by a constant c after every iteration. In order to use a fixed initial t we consider the scaled probabilities $\breve{p}_e$ and $\breve{p}_z$ in (4.7); otherwise the initial value of t would depend on σ and ρ.

4.1.4 Adaptive k

Algorithm 15 can optionally account for the sparsity of the current estimate via the flag `adaptive_k`. In the noiseless model of $r = A\hat{x}$, an update of the form $\hat{x} \leftarrow \hat{x} + \omega e_j$ with Parallel-ℓ_0 guarantees that $j \in \text{supp}(x)$ so that at the next iteration the problem with residual $r - a_j\omega$ and $(k-1)$-sparse signal is considered. The `adaptive_k` flag updates the sparsity prior in $\hat{x}$ after every update in hope of having more reliable estimates of p_e and p_z. We will see in the numerical experiments that under the data generating model that we tested this strategy does not bring substantial benefits. We don't rule out the posssibility that there are other signal and noise distributions for which this flag becomes especially useful, but we leave that as future work.

4.2 Numerical experiments

In this section we present numerical experiments which validate the efficacy of Robust-ℓ_0 decoding. In particular, we contrast Robust-ℓ_0 with other state-of-the-art greedy algorithms for compressed sensing in terms of their ability to recover the measured signal for varying problem sizes (k, m, n) as well as their computational complexity. To facilitate reproducibility we begin by describing the stopping conditions and measures used to denote successful recovery in the presence of noise in Section 4.2.1, along with how the parameter c is varied in Section 4.2.2. We then present the main numerical results in Section 4.2.3 where the algorithms phase transitions and runtime are presented, along with Sections 4.2.4 and 4.2.5 which show further details on Robust-ℓ_0 decoding's performance as a function of noise variance and for extreme subsampling respectively.



(a) Phase transition $\sigma = 0.001$ (b) Phase transition $\sigma = 0.01$ (c) Phase transition $\sigma = 0.1$

(d) Selection map $\sigma = 0.001$ (e) Selection map $\sigma = 0.01$ (f) Selection map $\sigma = 0.1$

(g) Best time $\sigma = 0.001$ (h) Best time $\sigma = 0.01$ (i) Best time $\sigma = 0.1$

(j) Timing ratio: Robust-ℓ_0, σ = 0.001 (k) Timing ratio: Robust-ℓ_0, σ = 0.001 (l) Timing ratio: Robust-ℓ_0, σ = 0.001

Figure 4.3: **Phase transitions, selection maps and timings for** $n = 2^{18}$. Robust-ℓ_0 has a comparable performance to Parallel-ℓ_0 for noise levels of $\sigma = 10^{-3}$ and its performance decreases as the noise level increases.

4.2.1 Stopping conditions

We are interested in the signal model $y = Ax + \eta$ where $\eta \sim \mathcal{N}(0, \sigma^2 I_{m\times m})$. If $\hat{x}$ is an approximation to x the residual is $r = y - A\hat{x}$. Note that if $\hat{x} = x$, then

$$\begin{aligned}\|r\|_1 &= \|y - A\hat{x}\|_1 \\ &= \|y - Ax\|_1 \\ &= \|\eta\|_1\end{aligned}$$

and we should not seek reductions in the residual below $\|\eta\|_1$ which would result in fitting to the additive noise. We further account for the variance of $\|\eta\|_1$ and denote the algorithm to have successfully recovered x if $\hat{x}$ satisfies

$$\frac{\|x - \hat{x}\|_1}{\|x\|_1} \leq \frac{\mathbb{E}\left[\|\eta\|_1\right] + C_1\sqrt{\mathrm{Var}\left[\|\eta\|_1\right]}}{\|x\|_1} \tag{4.30}$$

for some $C_1 \geq 0$. We should be aware that the right hand side of (4.30) might be greater than 1 for some choices of k, m and σ. When this happens, the stopping condition (4.30) becomes invalid since we expect $\hat{x}$ to have captured a proportion of the ℓ_1-energy of $\|x\|_1$. Hence, if the right hand side of (4.30) is greater than $\frac{1}{10}$ we clip the upper bound at this value and use the stopping condition

$$\frac{\|x - \hat{x}\|_1}{\|x\|_1} \leq \min\left(\frac{\mathbb{E}\left[\|\eta\|_1\right] + C_1\sqrt{\mathrm{Var}\left[\|\eta\|_1\right]}}{\|x\|_1}, \frac{1}{10}\right). \tag{4.31}$$

For the numerical experiments conducted in this section we consider nonzero entries in x drawn as $x_i \sim \mathcal{N}(0, 1)$, and noise $\eta_i \sim \mathcal{N}(0, \sigma^2)$ for which $\mathbb{E}\left[\|x\|_1\right] = k\sqrt{\frac{2}{\pi}}$ and $\mathbb{E}\left[\|\eta\|_1\right] = m\sigma\sqrt{\frac{2}{\pi}}$ and moreover $\mathrm{Var}\left[\|\eta\|_1\right] = m\sigma^2\left(1 - \frac{2}{\pi}\right)$, see e.g. [Leone et al., 1961].

4.2.2 Selection of parameter c in Algorithm 15

The sweeping parameter t in Algorithm 15 is initialised at 1 and updated by decreasing it by a constant value c. We observed in our experiments that the quality of the phase transitions are sensitive on the parameter c especially for low δ and ρ. We do not provide a way to fine-tune c, but we run our phase transitions with the following choices:

1. If the algorithm is `quantised`,

$$c = \begin{cases} 0.01 & \delta \leq 0.05 \\ 0.05 & \delta > 0.05 \text{ and } \rho \in (0, 0.1] \\ 0.075 & \delta > 0.05 \text{ and } \rho \in (0.1, 0.2] \\ 0.1 & \delta > 0.05 \text{ and } \rho \in (0.2, 1) \end{cases}. \tag{4.32}$$

Algorithm	`adaptive_k`	`quantised`
Robust-ℓ_0	No	No
Robust-ℓ_0-adaptive	Yes	No
Robust-ℓ_0-quantised	No	Yes
Robust-ℓ_0-adaptive-quantised	Yes	Yes

Table 4.2: **Variants of Robust-ℓ_0.** An index of the Robust-ℓ_0 variants with their respective defining characteristics is shown.

2. If the algorithm is `continuous`,

$$c = \begin{cases} 0.01 & \delta \leq 0.05 \\ 0.025 & \delta > 0.05 \end{cases} . \tag{4.33}$$

The values in (4.32) and (4.33) were chosen heuristically for ν and μ Gaussian.

4.2.3 Phase transitions and runtime

We benchmark the variants of Robust-ℓ_0 against other greedy algorithms via their *phase-transitions and runtime*. The user can supply two binary flags, `adaptive_k` and `quantised` which yield four different variants of Robust-ℓ_0. We assigned a unique label to each of these variants as described in Table 4.2.

The phase transition of a compressed-sensing algorithm [Donoho and Tanner, 2010b] is the largest value of k/m for which the algorithm is typically able to recover all k sparse vectors with sparsity less than k for a fixed m/n. Hence, for a fixed value of $\delta = m/n$ the phase transition of an algorithm is the largest value $\rho^*(\delta)$ for which the algorithm converges for all $\rho(\delta) < \rho^*(\delta)$. The value $\rho^*(m/n)$ often converges to a fixed value as $n \to \infty$, so phase transitions often partition the $\delta \times \rho$ space into two regions: One in which the algorithm converges with high probability and another in which the algorithm doesn't converge with high probability. We benchmark Robust-ℓ_0 against the combinatorial compressed sensing algorithms. Specifically, our tests include the following algorithms,

$$\{\text{Robust-}\ell_0, \text{Robust-}\ell_0\text{-adaptive, Robust-}\ell_0\text{-quantised, Robust-}\ell_0\text{-adaptive-quantised,}$$
$$\text{SSMP, SMP, CGIHT}\}.$$

In the numerical experiments Parallel-ℓ_0 was compared against the same set of algorithms; out of those we have selected SSMP and SMP as these perform best and are similar in nature to Robust-ℓ_0. We also compare with CGIHT, as this algorithm was shown to be the fastest among the greedy algorithms compared in [Blanchard

et al., 2015, Blanchard and Tanner, 2013]. Figures 4.3a, 4.3b and 4.3c show the phase transition curves for these algorithms with $\sigma = 10^{-3},\ 10^{-2},\ 10^{-1}$ respectively. The curves were computed by setting $n = 2^{18}$, $d = 7$, and using the stopping condition $\|r\|_1 \leq \mathbb{E}\left[\|\eta\|_1\right] = m\sigma\sqrt{\frac{2}{\pi}}$ and a success condition (4.30) with $C_1 = 1$. The testing is done at $m = \delta_p n$ for

$$\delta_p \in \{0.02p : p \in [4]\} \cup \left\{0.1 + \frac{89}{1900}(p-1) : p \in [20]\right\}.$$

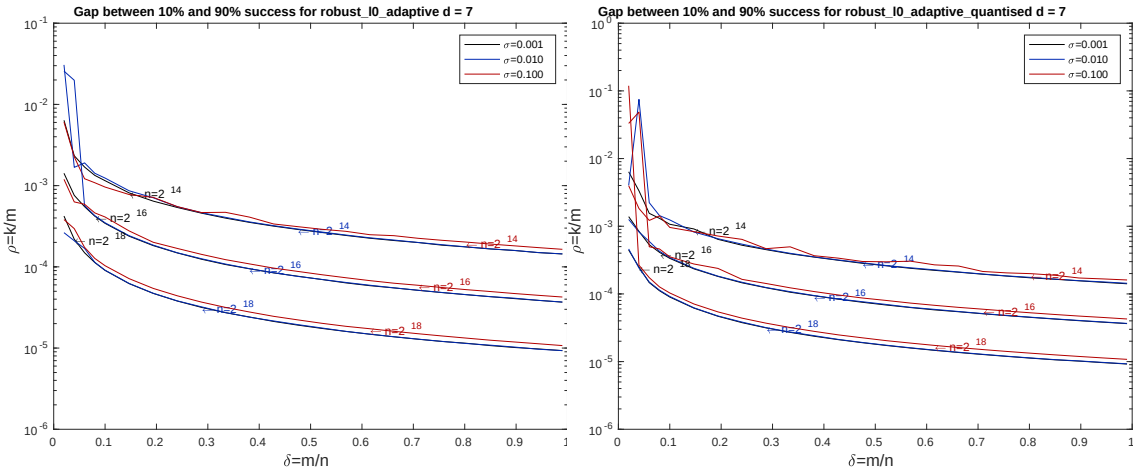
For each δ_p, we set $\rho = 0.01$ and generate 10 synthetic problems to apply the algorithms to, with a problem generated as in GM with the given parameters and μ and ν being normal Gaussian. If at least one such problem was recovered successfully, the sparsity ratio ρ is increased by 0.01 and the experiment is repeated. Following the testing framework in [Blanchard and Tanner, 2015], the recovery data is fitted using a logistic function and finally the 50% recovery transition function is computed and presented in the phase transition plots contained herein. Figures 4.3d, 4.3e and 4.3f show a selection map for these algorithms. Namely, these plots indicate which algorithm requires the least computational time[1] at each point in the $\delta \times \rho$ space where the algorithm converges. Finally, Figures 4.3g, 4.3h and 4.3i show the total time for convergence in milliseconds for the fastest algorithm at each point in the $\delta \times \rho$ space. The ratio of the time for the second fastest algorithm over the time for the fastest algorithm is given in Figures 4.3j, 4.3k, and 4.3l.

We can see from Figure 4.3 that CG-IHT dominates the upper region of the phase transition space, while the Robust-ℓ_0 algorithms only converge for $\rho \lessapprox 0.3$ which is consistent with the observed phase transitions for Parallel-ℓ_0 presented in Chapter 3. In terms of speed, Robust-ℓ_0 seems to be the most competitive for $\sigma \in \{10^{-3}, 10^{-2}\}$ and $\rho \lessapprox 0.2$. However, for large noise, $\sigma = 10^{-1}$, CGIHT becomes the fastest algorithm of all. We remark that, while CGIHT performs very well in our numerical tests, the current theory developed for it does not hold in this setting, as it requires a measurement matrix A with zero-mean columns.

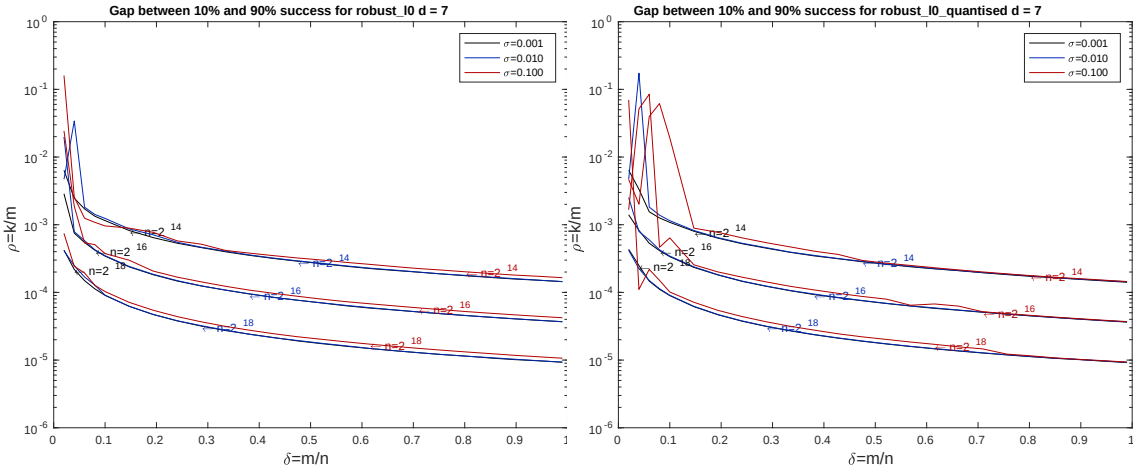
Figure 4.4 shows the widths for the Robust-ℓ_0 algorithms. The widths measure how sharp the phase transition of an algorithm is; namely, how thin is the boundary between the region of recovery with high-probability and the region of recovery where combinatorial search is needed. It has been shown that the widths of a compressed

[1] All the numerical results presented here were performed using a Linux machine with Intel Xeon E5-2643 CPUs 3.30 GHz, NVIDIA Tesla K10 GPUs and executed from Matlab R2016b. The code was added to the GAGA library available at http://www.gaga4cs.org/, and described in [Blanchard and Tanner, 2013], so as to facilitate large scale benchmarking against the other algorithms presented here which are also contained in the aforementioned library.

sensing algorithm tend to zero as $n \to \infty$ when decoding with linear programming [Donoho and Tanner, 2010a], and we usually expect the same behaviour for other algorithms [Blanchard et al., 2015]. Figure 4.4 show that the widths for the Robust-ℓ_0 algorithms indeed decrease with n and with δ. The observed smoothness of the phase transition widths signal also suggest that the stopping conditions of the algorithm are consistent for the problem under consideration.

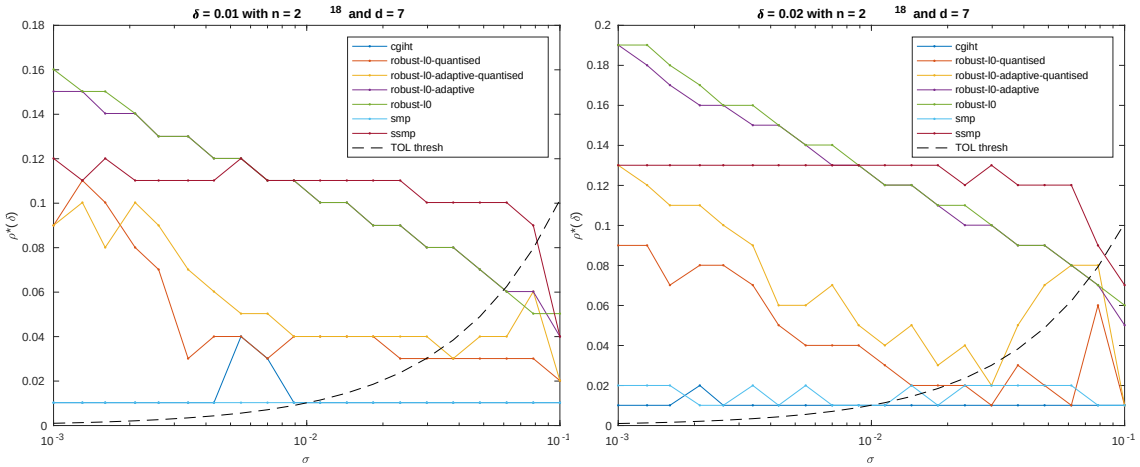


(a) robust-ℓ_0-adaptive (b) robust-ℓ_0-adaptive-quantised



(c) robust-ℓ_0 (d) robust-ℓ_0-quantised

Figure 4.4: **Phase transition widths for Robust-ℓ_0 variants.** The phase transition widths of each of the Robust-ℓ_0 variants decrease as $n \to \infty$ as is expected for a compressed sensing decoder. Interference is high at low values of δ.



(a) $\delta = 0.01$ (b) $\delta = 0.02$

(c) $\delta = 0.10$ (d) $\delta = 0.20$

Figure 4.5: **Decrease in phase transition for varying σ.** The figures show the decrease in phase transitions as a function of the noise level for fixed n and δ.

4.2.4 Dependence on noise variance, σ

We now investigate the extent to which the phase transitions of the algorithm decrease as we increase the noise level σ. To do this, we consider $\delta \in \{0.01, 0.02, 0.1, 0.2\}$ and for each value of δ we compute we define the grid

$$\sigma \in \{10^{-3+\frac{i}{10}} : i \in \{0\} \cup [20]\}.$$

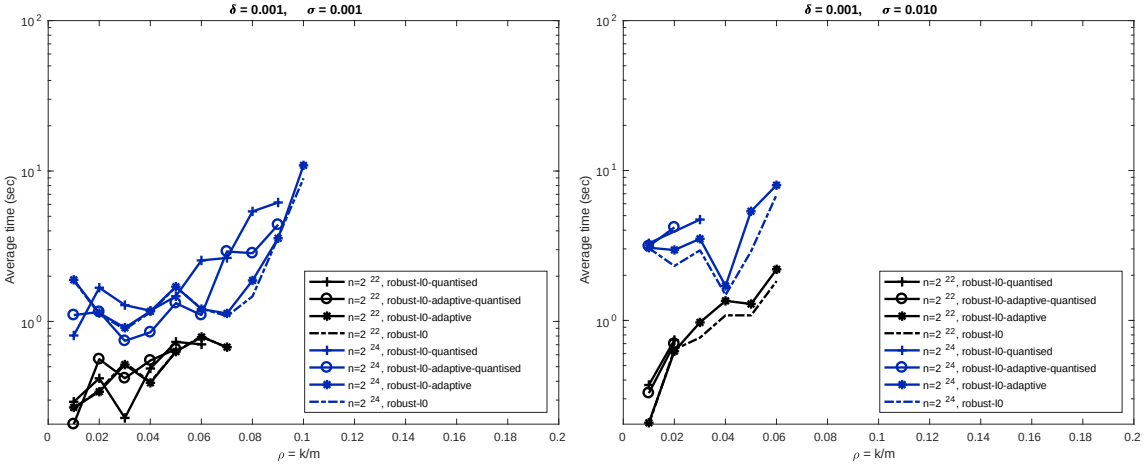
Then at each value of σ we let $\rho = 0.01$ and draw ten problem instances from GM with signal distribution $\mathcal{N}(0, 1)$ and noise distribution $\mathcal{N}(0, \sigma^2)$. If at least one of the problems was recovered successfully, then we set $\rho \leftarrow \rho + 0.01$ and repeat the experiment. We do this procedure for each of the algorithms considered and record the largest ρ having at least 50% success rate. The results are shown in Figure 4.5. In order to show where the clipping in (4.31) becomes active, the figures also show a `TOL`-curve which partitions the space into the region where the right hand side of (4.31) equals $\frac{1}{10}$ (bottom region) and the region where it equals the right hand side of (4.30) (top region). We can appreciate from Figure 4.5 that for small δ both Robust-ℓ_0 and SSMP have the best recovery capabilities, with Robust-ℓ_0 being preferable for noise levels $\sigma \lessapprox 10^{-2}$ and SSMP being better suited for larger noise levels. For larger δ, CGIHT is preferable except for very low noise levels.

4.2.5 Phase transitions for extreme subsampling, $\delta \ll 1$

The numerical experiments of Parallel-ℓ_0 in Chapter 3 showed flat phase transitions; that is, it was observed that $\rho^*(\delta)$ remained approximately 0.3 as $\delta \to 0$ provided n was sufficiently large. While Robust-ℓ_0 does not exhibit precisely the same behaviour in the presence of noise, we do observe that $\rho^*(\delta)$ remains nontrivial even for δ as small as 10^{-3}, again provided n is sufficiently large. We provide numerical evidence for this in Figure 4.6. For each $(\delta, \sigma) \in \{0.001, 0.01\} \times \{0.001, 0.01\}$ we let $\rho = 0.01$ and solve ten problems drawn from GM with nonzero distribution $\mathcal{N}(0, 1)$ and noise distribution $\mathcal{N}(0, \sigma^2)$. If at least one problem instance converges, we average the run-time of the problems that converged and repeat the process with $\rho \leftarrow \rho + 0.01$. We plot the timing at each ρ for parameter values for which at which at least 50% of the problems were successfully recovered under the criteria (4.31).

It can be seen in Figure 4.6a-4.6d that the phase transition either remains nearly unchanged or increases as n increases from 2^{22} to 2^{24}. In particular, Figure 4.6a shows that for $\sigma = 0.001$ and $\delta = 0.001$, the phase transitions for all variants of the Robust-ℓ_0 algorithms in fact increase from $\rho \approx 0.08$ to $\rho \approx 0.1$ as n increases from 2^{22}

to 2^{24}. Additionally, contrasting Figures 4.6a and 4.6b or 4.6c and 4.6d shows that a ten-fold increase in σ has the expected effect of reducing the phase transition and increasing the computational time. Figures 4.6a and 4.6b show the phase transition for Robust-ℓ_0 and Robust-ℓ_0-adaptive remain significant even for δ as small as 10^{-3}. Figures 4.6c and 4.6d show results for the same set of experiments, but for $\delta = 0.01$ which corresponds to a ten-fold increase in δ over the value used in Figures 4.6a and 4.6b. For $\sigma = 0.001$ the phase transition of Robust-ℓ_0 reaches $\rho \approx 0.17$, while for $\sigma = 0.01$ the phase transitions drop to $\rho \approx 0.12$ and there is an increase in the computational time.



(a) $\delta = 0.001$, $\sigma = 0.001$

(b) $\delta = 0.001$, $\sigma = 0.01$

(c) $\delta = 0.01$, $\sigma = 0.001$

(d) $\delta = 0.01$, $\sigma = 0.01$

Figure 4.6: **Timing performance for low δ.** Average recovery time at each ρ for fixed δ and σ and varying n for each variant of Robust-ℓ_0.

Chapter 5

Persistent Homology

The Topological Data Analysis pipeline starts with a dataset $S \subset \mathbb{R}^n$ sampled from a *shape* $\mathcal{M}$ embedded in $\mathbb{R}^n$. Its inference problem is to estimate the topological invariants of $\mathcal{M}$ at multiple *scales*, which are encoded as a set of *Betti numbers*

$$\text{Betti}(\mathcal{M}) := \{b_{p,r} : 0 \leq p \leq d, r \in [0, \infty)\}. \tag{5.1}$$

which geometrically represent the number of p-dimensional holes at scale $r \in [0, \infty)$ in a simplicial complex triangulation of $\mathcal{M}$. The notion of *scale* is important because even tough the data is embedded in a metric space that provides a notion of *distance*, there is no predefined notion of *closeness*. In the *persistent homology* paradigm [Edelsbrunner et al., 2002, Ghrist, 2014], the set Betti($\mathcal{M}$) is approximated at multiple scales $\{r_1, \ldots, r_T\} \subset [0, \infty)$ through a two-step procedure. First, a scale-indexed filtration of simplicial complexes is constructed yielding a simplicial complex K with vertex set S and m simplices. The resulting simplicial complex is represented as an $m \times m$ matrix ∂. Though persistence can be computed over any field, in this thesis we restrict to the Galois field of two elements $\mathbb{F}_2$, so we assume that $\partial \in \mathbb{F}_2^{m \times m}$. The homologies in the data are revealed by computing a factorisation of ∂, which is a process called *reduction*. Knowledge of the homologies persistent in a dataset can aid interpretation of the data [Carlsson, 2009] and to recognise patterns in it [Carlsson, 2014]. For example, [Carlsson et al., 2008] use qualitative topological analysis to study the behaviour of natural images to develop a compression algorithm; [Chan et al., 2013] applies the topological data analysis framework to the study of phylogenetic trees; [Lum et al., 2013] applies topological methods to gene expression of breast tumors, voting data from the U.S. House of Representatives and player performance of professional basketball players; [Taylor et al., 2015] uses this to study the spread of contagions on networks; and [Nicolau et al., 2011] to analyse microarray

data for gene expression analysis. The persistent homology pipeline is powered by a technology called *simplicial homology* which we now describe.

5.1 Simplicial Homology

In what follows, we use the shorthand 2^S to denote the power set of a finite set S and $|S|$ to denote its cardinality. Additionally, we use $\mathbb{Z}_{m+1}$ as a shorthand for $[m] \cup \{0\}$. The structure at the centre of simplicial homology is that of a *simplicial complex*, which we now define.

Definition 5.1 (Simplex, simplicial complex, and p-simplices)**.** *Let S be a finite set.*

1. *$\emptyset \neq \sigma \subset S$ is a* simplex*.*
2. *The dimension of the simplex σ is defined as $\dim(\sigma) = |\sigma| - 1$.*
3. *If $\alpha \subset \sigma \subset S$, then α is a* face *of σ.*
4. *$K \subset 2^S$ is a* simplicial complex *if,*
$$\sigma \in K \text{ and } \alpha \subset \sigma \Rightarrow \alpha \in K$$
5. *$K_p = \{\sigma \in K : \dim(\sigma) = p\}$ is the set of p-simplices in K.*

By Definition 5.1, any subset σ of S is a simplex and any non-empty subset α of σ is a face of σ. Note that the notion of *simplex* in Definition 5.1 is given for any finite set, so it is abstract. If $S = \{v_1, \ldots, v_p\} \subset \mathbb{R}^n$ contains $p+1$ affinely independent points, we can realise a S geometrically as a p-simplex by considering its *convex hull*

$$\text{conv}(S) := \left\{ \sum_{i \leq p} \lambda_i v_i : 0 \leq \lambda_i \in \mathbb{R}, \sum_i \lambda_i = 1 \right\}. \tag{5.2}$$

A face of the convex hull $\text{conv}(S)$ is any $\alpha = \text{conv}(S')$ where $S' \subset S$, so $\text{conv}(S)$ has $2^{p+1} - 1$ faces. For a given simplicial complex K, a *p-chain* is any $c \in 2^{K_p}$. If $c_1, c_2 \in 2^{K_p}$ are two p-chains, we can define the addition operation $+$ via

$$c_1 + c_2 = (c_1 \cup c_2) - (c_1 \cap c_2).$$

When equipped with this operation, the set of p-chains forms an abelian group with neutral element $\emptyset$. This group is denoted by $(\mathsf{C}_p, +)$ and is defined over $\mathbb{F}_2$ as

$$\mathsf{C}_p(K) = \left\{ \sum_{c \in K_p} \alpha_c c : \alpha_c \in \mathbb{F}_2 \right\}. \tag{5.3}$$

Even though $\mathsf{C}_p(K)$ depends on an underlying simplicial complex K, we will write $\mathsf{C}_p = \mathsf{C}_p(K)$ when there is no risk of confusion. It follows from (5.3) and Definition 5.1 that $K_p \in \mathsf{C}_p(K)$. There is one such group for each $p \in \mathbb{Z}$ and C_p can be related to C_{p-1} via *boundary maps*. These also allow us to study the p-chains with Linear Algebra techniques.

Definition 5.2 (Boundary map)**.** *Let K be a simplicial complex on S. The boundary of a p-simplex $\sigma \in K_p$ is defined as,*

$$\mathrm{bd}(\sigma) = \{\alpha \subset \sigma : \dim(\alpha) = \dim(\sigma) - 1\} \in \mathsf{C}_{p-1}. \tag{5.4}$$

The boundary of a p-chain $c \in \mathsf{C}_p$ is defined as,

$$d_p(c) = \sum_{\sigma \in c} \mathrm{bd}(\sigma) \in \mathsf{C}_{p-1}. \tag{5.5}$$

Let $(\mathbb{F}_2^m, \oplus)$ be the additive group of m-dimensional vectors over $\mathbb{F}_2$. Define an enumeration on the simplices in $K_{p-1} = \{\alpha_1, \ldots, \alpha_m\}$, and let $\psi_p : K_p \to \mathbb{F}_2^m$ be defined by the mapping,

$$\psi_p(\sigma) = \begin{bmatrix} \mathbb{1}\{\alpha_1 \in \mathrm{bd}(\sigma)\} \\ \vdots \\ \mathbb{1}\{\alpha_m \in \mathrm{bd}(\sigma)\} \end{bmatrix} \in \mathbb{F}_2^m \tag{5.6}$$

By the simplicial complex structure of K, each $\sigma \in K_p$ is uniquely defined by an element in $\mathbb{F}_2^m$ via ψ_p and the boundary map d_p. This is because if $\mathrm{bd}(\sigma_1) = \mathrm{bd}(\sigma_2)$, then both σ_1 and σ_2 have the same $(p-1)$-dimensional faces, and since each of these faces belongs to the simplicial complex, both σ_1 and σ_2 must have the same components at each dimension $q \leq p$. This naturally induces an injection between the p-chains and $\mathbb{F}_2^m$ via the function $\varphi : (\mathsf{C}_p, +) \to (\mathbb{F}_2^m, \oplus)$ given by

$$\varphi(c) = \sum_{\sigma \in c} \psi_p(\sigma) \in \mathbb{F}_2^m \tag{5.7}$$

By the group structure, $\varphi(\cdot)$ satisfies $\varphi(c + d) = \varphi(c) \oplus \varphi(d)$ for all $c, d \in \mathsf{C}_p$, so it is an isomorphism. Hence, the boundary operator allows us to speak about the boundary of a chain and to define a mapping between them. In this respect, C_p can be thought of as a vector space generated by $\{\varphi(c) : c \in \mathsf{C}_p\}$. The collection of boundary operators $\{\mathsf{C}_p\}_{p \in \mathbb{N}}$ and the collection of boundary maps $\{d_p : \mathsf{C}_p \to \mathsf{C}_{p-1}\}_{p \in \mathbb{N}}$ form a *chain complex* which is represented pictorially as,

$$\cdots \to \mathsf{C}_p \xrightarrow{d_p} \mathsf{C}_{p-1} \xrightarrow{d_{p-1}} \cdots \xrightarrow{d_1} \mathsf{C}_0 \xrightarrow{d_0} \emptyset.$$

If K is a simplicial complex, the kernel and image of C_p are geometrically meaningful. Let,

$$\mathsf{Z}_p := \operatorname{Ker} d_p = \{c \in \mathsf{C}_p : d_p(c) = \emptyset\}, \tag{5.8}$$

$$\mathsf{B}_p := \operatorname{Im} d_{p+1} = \{c \in \mathsf{C}_p : \exists a \in \mathsf{C}_{p+1} \text{ s.t. } c = d_{p+1}(a)\}. \tag{5.9}$$

Elements in $\mathsf{Z}_p = \operatorname{Ker} \partial_p$ are called *p-cycles*, while elements in $\mathsf{B}_p = \operatorname{Im} \partial_{p+1}$ are called *p-boundaries*. The following lemma guarantees that both Z_p and B_p are subgroups of C_p.

Lemma 5.3 ($\mathsf{B}_p \subset \mathsf{Z}_p \subset \mathsf{C}_p$ [Edelsbrunner and Harer, 2010])**.** *The following holds for all* $p \in \mathbb{N}$*,*

$$d_{p-1} \circ d_p(c) = \emptyset \quad \forall \ \ c \in \mathsf{C}_p \tag{5.10}$$

This implies that $\mathsf{B}_p \subset \mathsf{Z}_p \subset \mathsf{C}_p$*.*

Given that B_p is a subspace of Z_p we can define the quotient of both spaces. This is called the *p-th homology group* and is written as,

$$\mathsf{H}_p = \mathsf{Z}_p / \mathsf{B}_p. \tag{5.11}$$

The elements of H_p are called *homology classes* and are of the form $c + \mathsf{B}_p \ \forall \ c \in \mathsf{Z}_p$. The p-th *Betti number* is defined as

$$b_p := \operatorname{rank} \mathsf{H}_p = \operatorname{rank} \mathsf{Z}_p - \operatorname{rank} \mathsf{B}_p, \tag{5.12}$$

and counts the number of p-dimensional holes of the simplicial complex K. The Betti numbers are invariant to continuous deformations of the geometric realisation of K and are one of our main objects of interest. Having surveyed the essential components of simplicial homology, we move on to describe persistent homology.

5.2 Persistent Homology

In persistent homology, a filtered simplicial complex K is built from a point-cloud $S \subset \mathbb{R}^d$ sampled from a data manifold $\mathcal{M} \subset \mathbb{R}^d$. As mentioned earlier, the goal of persistent homology is to estimate the relevant homological features of $\mathcal{M}$ at all scales $r \in [0, \infty)$ given the dataset S.

The geometry of $\mathcal{M}$ is captured with a simplicial complex that approximates it with a *triangulation* at scale t. Formally, we say that $\mathcal{M}$ is *triangulable* if there is a simplicial complex $\mathbf{M}$ together with a homeomorphism between $\mathcal{M}$ and the geometric

realisation of $\mathbf{M}$ [Edelsbrunner and Harer, 2010]. At each scale $r \in \{r_1, \ldots, r_T\} \subset [0, \infty)$ a simplicial complex triangulation $\mathbf{M}_r$ is computed from S by letting $K = \emptyset$ and adding a simplex $\sigma \subset S$ to K whenever the points in σ are sufficiently close to each other. The notion of closeness is implied by the relevant scale parameter, and is assessed for r via a monotonic function[1] $f_r : 2^S \to \{0, 1\}$ inducing a filtration

$$\mathbf{M}_0 \subset \mathbf{M}_1 \subset \cdots \subset \mathbf{M}_T = K \tag{5.13}$$

of simplicial complexes $\mathbf{M}_i = \left\{\sigma \in 2^S : f_{r_i}(\sigma) = 1\right\}$. The resulting simplicial complex depends on the choice of the function. If $\mathbb{R}^d$ is the d-dimensional Euclidean space with p-norm $\|\cdot\|_p$ and $S \subset \mathbb{R}^d$, we let . For example, if $\mathcal{B}_r(c) := \left\{x \in \mathbb{R}^d : \|x - c\| \leq r\right\}$ is the d-dimensional ball of radius $r \geq 0$ centred at $c \in \mathbb{R}^d$, then the function

$$f_r(\sigma) = \mathbb{1}\left\{\bigcap_{x_0 \in \sigma} \mathcal{B}_r(x_0)\right\},$$

generates the *Čech complex* filtration. And if $\mathrm{diam}(S) := \max_{x,y \in S} \|x - y\|$ is the *diameter* of $S \subset \mathbb{R}^d$, the function

$$f_r(\sigma) = \mathbb{1}\left\{\mathrm{diam}(\sigma) \leq 2r\right\}$$

generates the *Vietoris-Rips complex* filtration. Hence, given a grid $\{r_1, \ldots, r_T\}$ of scales, the sequence $\{\mathbf{M}_1, \ldots, \mathbf{M}_T\}$ can be constructed incrementally by setting $\mathbf{M}_0 \leftarrow \emptyset$ and letting

$$\mathbf{M}_i \leftarrow \mathbf{M}_{i-1} \cup \left\{\sigma \subset S : f_{r_i}(\sigma) = 1\right\} \quad \forall\, i \in [T]. \tag{5.14}$$

We shall assume that simplicial complex K in (5.13) has m elements and is given by

$$K = \{\sigma_1, \ldots, \sigma_m\}.$$

It is further assumed that the simplices in K are indexed according to a *compatible ordering*, meaning that the simplices in $\mathbf{M}_\ell$ always precede the ones in $K \setminus \mathbf{M}_\ell$, and that the faces of any given simplex always precede the simplex.

Persistent homology tracks how the homology of the filtration changes at each scale r_t or, equivalently, as new simplices are added to the filtration. Indeed, when adding simplex σ_i at scale r_t, the homology of $\mathbf{M}_t$ can change in one of two possible ways [Edelsbrunner and Harer, 2010].

1. A class of dimension $\dim(\sigma_i)$ is created. In this case, σ_i is a *positive* simplex.

[1] The elements of 2^S form a poset when ordered by inclusion.

2. A class of dimension $\dim(\sigma_i) - 1$ is destroyed. In this case, σ_i is a *negative* simplex.

If σ_j is a negative simplex, then it destroys the class created by a positive simplex σ_i with $i < j$, see Lemma 5.6. This observation induces a natural pairing (σ_i, σ_j) between a negative simplex σ_j and the positive simplex σ_i it destroys. This is because if σ_j is a negative simplex, then it destroyed a class that was created by a simplex σ_i with $i < j$. The pair (σ_i, σ_j), for which we will often use the shorthand (i, j), is called a *persistence pair* and has an *index persistence* of $j - i$.

When r_T is sufficiently small, we might find that some simplices are never destroyed; these simplices represent the homology classes that are *persistent* in the filtration up to scale r_T, and we called them *essential*. The persistence pairs are computed by representing the filtration (5.13) as a boundary matrix $\partial \in \mathbb{F}_2^{m\times m}$ and applying the *reduction algorithm* [Edelsbrunner et al., 2002] to it. We discuss this process in the next section.

5.3 Boundary matrix reduction

A filtration can be represented in terms of a binary matrix that encodes its simplices via a representation in the style of (5.6). This matrix is called the *boundary matrix* and is defined as follows,

Definition 5.4 (Boundary matrix ∂)**.** *Given a simplicial complex $K = \{\sigma_1, \ldots, \sigma_m\}$, the boundary matrix $\partial \in \mathbb{F}_2^{m\times m}$ of K is defined as*

$$\partial_{i,j} = \begin{cases} 1 & \sigma_i \in \mathrm{bd}(\sigma_j) \\ 0 & \textit{otherwise} \end{cases}$$

For the rest of this thesis, we implicitly assume that if $\partial \in \mathbb{F}_2^{m\times m}$, then ∂ is the boundary matrix of some simplicial complex K. Boundary matrices are sparse, binary, and upper-triangular by the compatible order assumption on the filtration. If ∂ is a boundary matrix, it has a function $\mathrm{low}_\partial : [m] \to \mathbb{Z}_{m+1}$ associated to it that tracks the location of the largest index in the support of each column. This idea central in the reduction process of the matrix, so we state it in a definition.

Definition 5.5 ($\mathrm{low}_\partial(\cdot)$)**.** *Let $\partial \in \mathbb{F}_2^{m\times m}$ be a boundary matrix and $\partial_j \in \mathbb{F}_2^m$ be the j-th column of ∂. The function $\mathrm{low}_\partial : [m] \to [m]$ is defined as,*

$$\mathrm{low}_\partial(j) = \begin{cases} \max\{i \in [m] : \partial_{i,j} = 1\} & \textit{if } \partial_j \neq 0. \\ 0 & \textit{if } \partial_j = 0. \end{cases}$$

The matrix ∂ will be said to be *reduced* when $\text{low}_\partial \in \mathbb{Z}_{m+1}^m$ is an injection over its support, *i.e.* when

$$\text{low}_\partial(j_1) = \text{low}_\partial(j_2) > 0 \Rightarrow j_1 = j_2.$$

An illustration of several boundary matrices with their respective $\text{low}_\partial(\cdot)$ function is shown in Figure 5.1. We use the notation ∂^* to denote a matrix ∂ with injective low_∂ and remark that even though ∂ can have several reductions, the injection is a property of the complex K and does not depend on any particular reduction ∂^*, see [Edelsbrunner and Harer, 2010, p.183]. Moreover, when there is no risk of confusion, we let low^* be the injection of the boundary matrix under consideration. Given the injection property, $\text{low}^*(\cdot)$ partitions the simplices in K as,

$$\begin{aligned}
\mathsf{Pos} &= \{j \in [m] : \text{low}^*(j) = 0\} \\
\mathsf{Neg} &= \{j \in [m] : \text{low}^*(j) > 0\} \\
\mathsf{Ess} &= \{j \in \mathsf{Pos} : \nexists\, k \text{ s.t. } \text{low}^*(k) = j\}
\end{aligned}$$

so $\mathsf{Pos} \cap \mathsf{Neg} = \emptyset$ and $\mathsf{Ess} \subset \mathsf{Pos}$. The vector low^* reveals information about the pairings by virtue of the following Lemma.

Lemma 5.6 (Pairing [Edelsbrunner et al., 2002])**.** *If σ_j is a negative simplex, then $\sigma_{\text{low}^*(j)}$ is a positive simplex.*

It will also be convenient to define the set of Pairs as

$$\mathsf{Pairs} = \left\{(i, j) \in [m] \times [m] : i = \text{low}^*(j) > 0\right\} \tag{5.15}$$

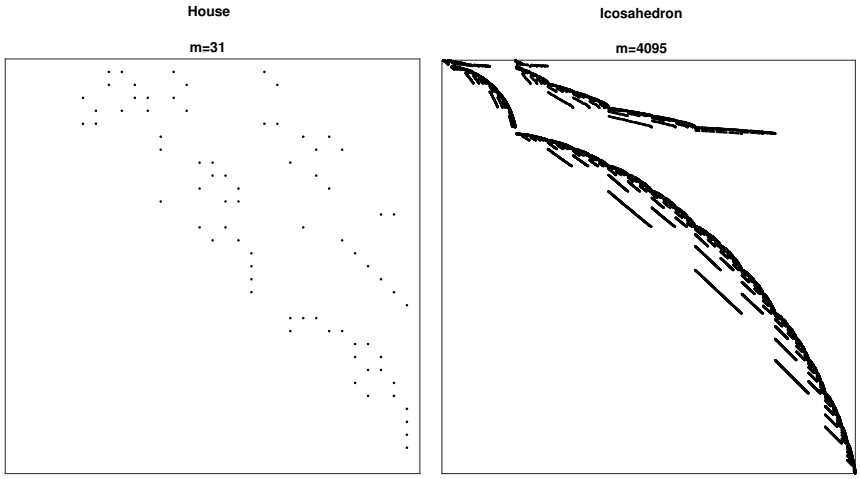
Additionally, the set of Paired simplices is the union of the set of negative and associated positive simplices

$$\mathsf{Paired} = \left\{j \in [m] : \text{low}^*(j) \in [m]\right\} \cup \left\{\text{low}^*(j) \in [m] : j \in [m]\right\} \tag{5.16}$$

We use the terms *positive*, *negative* and *essential* interchangeably between simplices and columns. These relations can be illustrated diagramatically with the following flowchart.

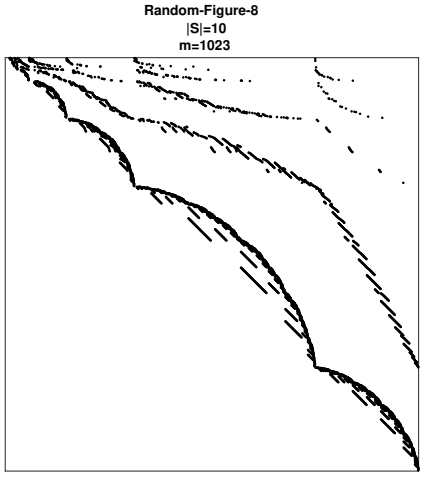
$$\sigma_j \in K \Rightarrow \begin{cases} \text{low}^*(j) > 0 & \Rightarrow j \in \mathsf{Neg},\ \text{low}^*(j) \in \mathsf{Pos} \Rightarrow (\text{low}^*(j), j) \in \mathsf{Pairs} \\ \text{low}^*(j) = 0 & \Rightarrow j \in \mathsf{Pos} \Rightarrow \begin{cases} \exists k : j = \text{low}^*(k) \Rightarrow (j, k) \in \mathsf{Pairs} \\ \nexists k : j = \text{low}^*(k) \Rightarrow j \in \mathsf{Ess} \end{cases} \end{cases}$$

The technology used to reduce ∂ is known as the *reduction algorithm* (Alg. 16) and was first presented in [Edelsbrunner et al., 2002]. Its convergence guarantees are given in Theorem 5.7.

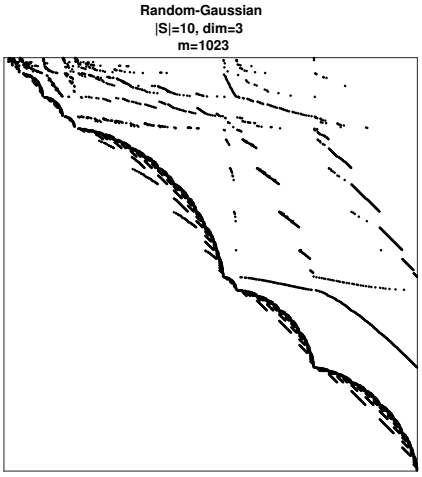


(a) House figure

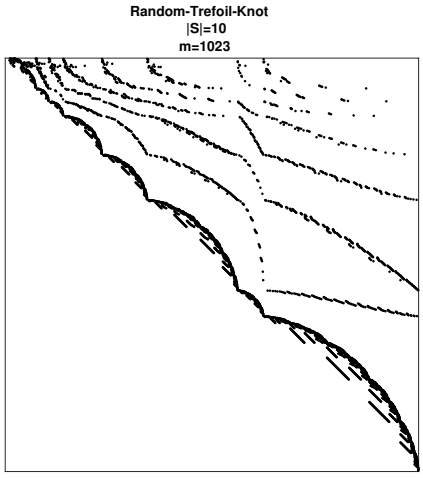
(b) Icosahedron



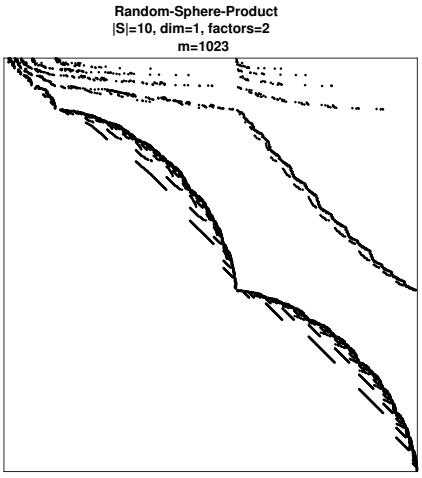
(c) Random figure 8



(d) Random Gaussian



(e) Random trefoil knot



(f) Sphere product

Figure 5.1: **Sparsity pattern of boundary matrices.** The sparsity a boundary matrix is shown for a selection of topological shapes.

Algorithm 16: Standard reduction [Edelsbrunner et al., 2002]

Data: $\partial \in \mathbb{F}_2^{m\times m}$
Result: $\text{low}^* \in \mathbb{Z}_{m+1}^m$
for $j \in [m]$ **do**
  while $\exists\ j_0 < j\ :\ \text{low}_\partial(j_0) = \text{low}_\partial(j)$ **do**
    $\partial_j \leftarrow \partial_j + \partial_{j_0}$;
  end
end
$\text{low}^* \leftarrow \text{low}_\partial$;

Theorem 5.7 (Convergence of Alg. 16 [Edelsbrunner et al., 2002])**.** *Alg. 16 converges for any boundary matrix* $\partial \in \mathbb{F}_2^{m\times m}$ *in at most* $\mathcal{O}(m^3)$ *operations.*

This concludes the final ingredient of the TDA pipeline, which we graphically depict below.

$$\boxed{S \subset \mathbb{R}^n} \rightarrow \boxed{K = \{\sigma_1, \ldots, \sigma_m\}} \rightarrow \boxed{\partial \in \mathbb{F}_2^{m\times m}} \rightarrow \boxed{\text{low}^* \in \mathbb{Z}_{m+1}^m} \rightarrow \boxed{\text{Barcodes}}$$

It was shown in [Morozov, 2005] that Algorithm 16 has a worst-case complexity of $\mathcal{O}(m^3)$ where m is the number of simplices in the filtration, which imposes an important bottleneck in the pipeline as $m \to \infty$. However, it has also been observed that for real world datasets Algorithm 16 normally converges in less that $\mathcal{O}(m^3)$ operations. Additionally, some strategies have attempted to make reduction more efficient by using the structure of the boundary matrix or parallelising the workload. We review some of these strategies in the next section.

5.4 Boundary reduction algorithms

Let K be the simplicial complex corresponding to a filtration over a grid $\{r_1, \ldots, r_T\}$ and point-cloud $S \subset \mathbb{R}^d$. The number of simplices in K is of order $\Omega(2^{|S|})$ as $r_T \to \text{diam}(S)$, which turns the process unfeasible even for moderately large point-clouds S and scales r_T. Hence, though Alg. 16 terminates in $\mathcal{O}(m^3)$ steps, we may find that $m = \mathcal{O}(2^{|S|})$ making the reduction unfeasible. The main computational overhead when reducing a boundary matrix is in performing left-to-right column operations, so many reduction algorithms have implemented strategies that cut the number of column operations required to fully reduce the matrix. One such strategy was given in [Chen and Kerber, 2011, Bauer et al., 2014a] by reducing columns in blocks of

decreasing dimension and observing that, by Lemma 5.6, column $\text{low}^*(j)$ can be set to zero whenever column j is reduced. As $\dim(\sigma_{\text{low}^*(j)})$ is of one lower dimension that $\dim(\sigma_j)$, clearing results in all positive non-essential columns of dimension less than $\dim(K)$ being set to zero without zeroing them through column additions. Setting a positive column to zero when its corresponding negative pair has been found is often called *clearing*, so we adopt this terminology. Alg. 18 heavily relies on this clearing strategy so we sketch the pseudocode of [Chen and Kerber, 2011] in Alg. 17.

Algorithm 17: Standard reduction with a twist [Chen and Kerber, 2011]

Data: $\partial \in \mathbb{F}_2^{m\times m}$
Result: $\text{low}^* \in \mathbb{Z}_m^{m+1}$
for $d \in \{\dim K, \dim K - 1, \ldots, 1\}$ **do**
  for $j \in K_d$ **do**
    while $\exists j_0 < j : \text{low}_\partial(j_0) = \text{low}_\partial(j)$ **do**
      $\partial_j \leftarrow \partial_j + \partial_{j_0}$;
    end
    if $\partial_j \neq 0$ **then**
      $\partial_{\text{low}_\partial(j)} \leftarrow 0$;
    end
  end
end
$\text{low}^* \leftarrow \text{low}_\partial$;

Another strategy to reduce the complexity of Alg. 16 is called *compression* [Bauer et al., 2014a] and consists in deriving analytical guarantees to nullify nonzeros in the boundary matrix without affecting the pairing of the simplices. This has the objective of saving arithmetic operations and hence reducing the flop count. However, for very large values of m, which are typical for large filtration values where m grows exponentially with $|S|$, it is necessary to also parallelise these approaches to facilitate scalability. The idea of distributing the workload of the reduction algorithm has already been explored in [Bauer et al., 2014a,b] where the matrix is partitioned into b blocks of contiguous columns to be independently reduced in a shared-memory or distributed system. The numerical simulations in [Bauer et al., 2014b] show that these strategies can indeed bring substantial speed-ups when implemented on a cluster, but also requires the provision of the parameter b as well as a number of design choices for a practical implementation.

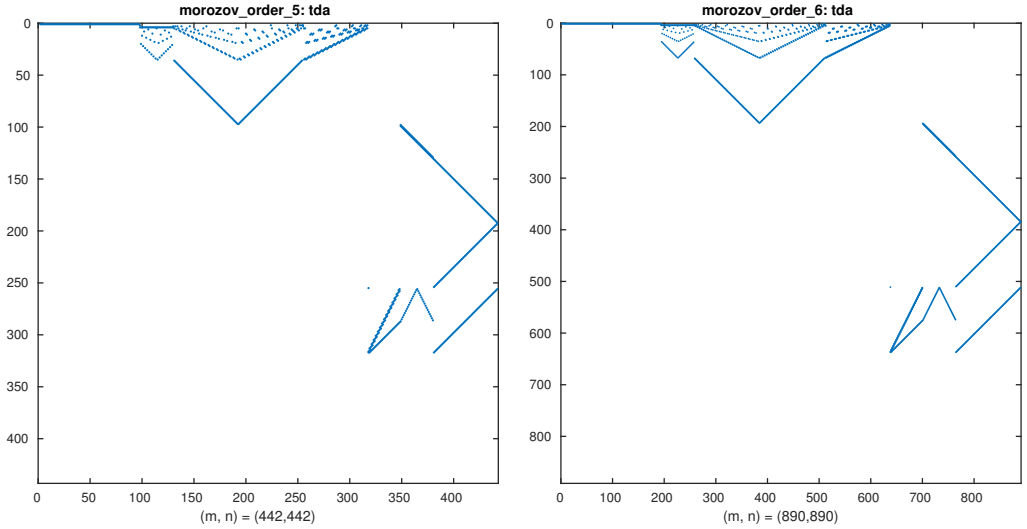
Apart from parallelisation, a number of other approaches have been proposed. For instance, [Milosavljević et al., 2011] adapted the Coppersmith-Winograd algorithm

[Coppersmith and Winograd, 1990] to guarantee reduction in time $\mathcal{O}(m^{2.3755})$. A set of sequential algorithms that exploits duality of vector spaces was given in [De Silva et al., 2011a,b], but as pointed out in [Otter et al., 2017] is only known to give speed-ups when applied to Vietoris-Rips complexes. Finally, [Sheehy, 2014] projects ∂ to a low-dimensional space while controlling the error between the resulting barcodes, but doing so in practice can be as costly as reducing the matrix in its ambient space. Despite these advances, the complex given in [Morozov, 2005] can still be adapted to yield a worst-case complexity of $\Omega(m^3)$. Our first result is to show that this example is so structured that it can be reduced with a counting argument.

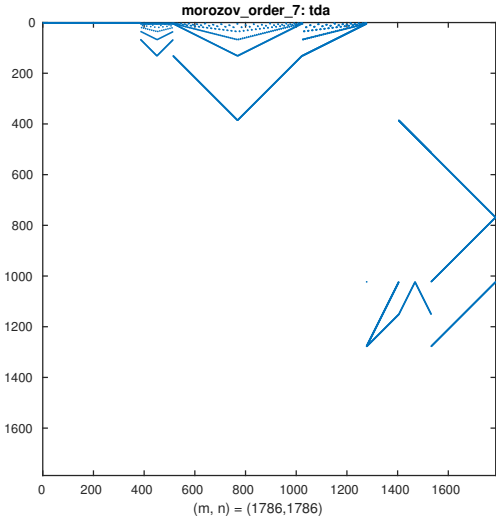
5.5 Reducing Morozov's complex in $\mathcal{O}(m)$ operations

In this section we consider reducing *Morozov's complex*, namely, the simplicial complex constructed in [Morozov, 2005] for which Algorithm 16 converges in $\Omega(m^3)$ operations. We do not review the construction of Morozov's complex, but note that there is an implicit notion of *order of complexity* $\omega \in \mathbb{N}$ in the construction of this complex and that $m = \Omega(2^\omega)$ as $\omega \to \infty$. We have implemented a function[2] that generates Morozov's complex up to order $\omega \in \mathbb{N}$ and generate some examples for $\omega \in \{5, 6, 7\}$ in Figure 5.2. The exponential growth in the dimension of the boundary matrix is also evident from this figure. It can be seen from Figure 5.2 that these boundary matrices have a special kind of structure. The first thing we note is that each of these matrices can be partitioned into blocks, where each block corresponds to the boundary matrix of simplices of dimension $p \in \{0, 1, 2\}$. Simplices of dimension 0 correspond to trivial vertex representations which do not need to be reduced. Now, looking at the sparsity pattern of the block corresponding to simplices of dimension 1, it is possible to note that the $\text{low}_\partial \in \mathbb{Z}_{m+1}^m$ forms a reversed triangle $\bigtriangledown$. It is evident that the sequence of simplices in the left edge of this triangle is increasing with the row index $i \in [m]$ of the matrix. Moreover, for these simplices, $\text{left}(\text{low}_\partial(j)) = j$ so their pivots can be immediately read as low^*'s. Moreover, these simplices can also be labelled as negative. It is less trivial to read off the category of the simplices in the right hand edge of the triangle and those in the block corresponding to simplices of dimension 2. The remainder of this section deals with a strategy that allows us to reduce Morozov boundary matrices in time $\mathcal{O}(m)$, and that has the potential to speed up reduction of other boundary matrices.

[2] In a fork of the library `Javaplex` [Tausz et al., 2014] in `https://github.com/rodrgo/javaplex`



(a) Morozov $\omega = 5$ (b) Morozov $\omega = 6$



(c) Morozov $\omega = 7$

Figure 5.2: **Sparsity pattern for Morozov boundary matrices.** The sparsity pattern of Morozov boundary matrices is shown for several orders of complexity.

We start with the following definition.

Definition 5.8 (left(·))**.** *Let $\partial \in \mathbb{F}_2^{m\times m}$ be a boundary matrix. The* left *function is defined as*

$$\text{left}(i) = \begin{cases} \min\{j \in [m] : \partial_{i,j} = 1\} & \partial_{i,\cdot} \neq 0 \\ 0 & \partial_{i,\cdot} = 0 \end{cases} \tag{5.17}$$

That is, for each row $i \in [m]$, the function left maps the index of the first simplex in the filtration that has σ_i as a face. If $\partial \in \mathbb{F}_2^{m\times m}$ is a boundary matrix, then $j \in \mathsf{Neg}$ or $j \in \mathsf{Pos}$, but not both. We have argued in the preliminary section that if $j \in \mathsf{Neg}$ then it is paired with a column $\text{low}^*(j) \in \mathsf{Pos}$, and we also have $\text{low}^*(j) < j$. Note that $j \in \mathsf{Neg}$ whenever there is an $i \in [m]$ such that $\text{left}(i) = j$ and that, by Lemma 5.6, every $j \in \mathsf{Neg}$ must be paired with some $\ell \in \mathsf{Pos} \cap [j-1]$. The sets $\mathsf{Pos} \cap [j]$ and $\mathsf{Neg} \cap [j]$ can be partitioned as follows.

$$\mathsf{Neg}\cap[j] \begin{cases} A_j := \{\ell \in \mathsf{Neg}\cap[j] : \ell \notin \text{Im left}\} \\ B_j := \{\ell \in \mathsf{Neg}\cap[j] : \ell \in \text{Im left}\} \end{cases}$$

$$\mathsf{Pos}\cap[j] \begin{cases} C_j := \{\ell \in \mathsf{Pos}\cap[j] : \exists k \in \mathsf{Neg}\cap[j] : (\ell,k) \in \mathsf{Pairs}\} \\ D_j := \{\ell \in \mathsf{Pos}\cap[j] : \exists k \in \mathsf{Neg}\cap([m]\setminus[j]) : (\ell,k) \in \mathsf{Pairs}\} \\ E_j := \{\ell \in \mathsf{Pos}\cap[j] : \ell \in \mathsf{Ess}\} \end{cases}$$

We note that by the pairing restriction we must have

$$|A_j| + |B_j| = |C_j|. \tag{5.18}$$

We now derive a result that allows us to profit from this observation.

Lemma 5.9 (Clearing by counting)**.** *If for $j \in [m]$ it holds that $|A_j| = j/2$, then*

$$|B_j| = |D_j| = |E_j| = 0$$

and

$$\{\ell \in [j] : \ell \notin A_j\} = C_j \subset \mathsf{Pos} \tag{5.19}$$

Proof. Suppose that for $j \in [m]$ it holds that $|A_j| = j/2$. Then, for all $k \in A_j$, there exists $\ell \in [j-1] \cap \mathsf{Pos}$ such that $(\ell, k) \in \mathsf{Pairs}$. Hence, $\{\ell \in [j] : \ell \notin A_j\} = C_j$. Since $|A_j| = |C_j|$ and $A_j \cap C_j = \emptyset$, we have that $A_j \cup C_j = [j]$. Hence, $|B_j| = |D_j| = |E_j| = 0$. □

Hence, we define the following curve,

Definition 5.10 (ρ-curve)**.** *Let $\partial \in \mathbb{F}_2^{m\times m}$. The ρ curve is initialised by setting $\rho_0 = 0$ and letting*

$$\rho_j = \begin{cases} \rho_{j-1} + 1 & j \in \operatorname{Im}\mathsf{left} \\ \rho_{j-1} - 1 & j \notin \operatorname{Im}\mathsf{left} \end{cases} \tag{5.20}$$

Lemma 5.9 can be rewritten in the following way:

Lemma 5.11 (Clearing with ρ-curve)**.** *Let $\partial \in \mathbb{F}_2^{m\times m}$ be a boundary matrix and let ρ be defined as in* (5.20)*. If $\rho_j = 0$, then*

$$\{\ell \in [j-1] : \ell \notin \operatorname{Im}\mathsf{left}\} \subset \mathsf{Pos}\,.$$

Proof. By the definition of ρ_j in (5.20),

$$\rho_j = |A_j| - |B_j \cup C_j \cup D_j \cup E_j|.$$

If $\rho_j = 0$, then $|A_j| = |B_j \cup C_j \cup D_j \cup E_j|$. Since all of these sets are mutually exclusive, we obtain that $|A_j| = |B_j| + |C_j| + |D_j| + |E_j|$. Coupling this with (5.18), we arrive at the system,

$$\begin{cases} |A_j| = |B_j| + |C_j| + |D_j| + |E_j| \\ |C_j| = |A_j| + |B_j| \end{cases} \tag{5.21}$$

which yields the result

$$|B_j| = |D_j| = |E_j| = 0 \;\text{ and }\; |A_j| = |C_j| \tag{5.22}$$

Hence $|A_j| = |C_j| = j/2$ and by Lemma 5.9 the result follows. □

Lemma 5.11 says that if we can establish that there are enough positive and negative columns to guarantee pairing up to a given $j \in [m]$, then all columns $\ell \in [j-1]$ for which $\ell \notin \operatorname{Im}\mathsf{left}$ must be positive columns and can immediately be set to zero.

Figure 5.3 shows the ρ curve for a number of different complexes. The horizontal axis represents the simplex index $j \in [n]$ while the vertical axis represents the value of ρ_j. We can apply the result of Lemma 5.11 whenever $\rho_j = 0$, so we mark these values with an asterisk symbol $*$. It is seen in Figure 5.3 that Morozov's complex is so structured that the ρ curve follows a clear well-defined pattern and touches zero when evaluated at the last simplex of each p-dimensional block. The ρ-curves for the other simplices are not nearly as informative, showing that they are not structured enough to trigger an invokation of Lemma 5.11. Lemma 5.11 can be implemented as

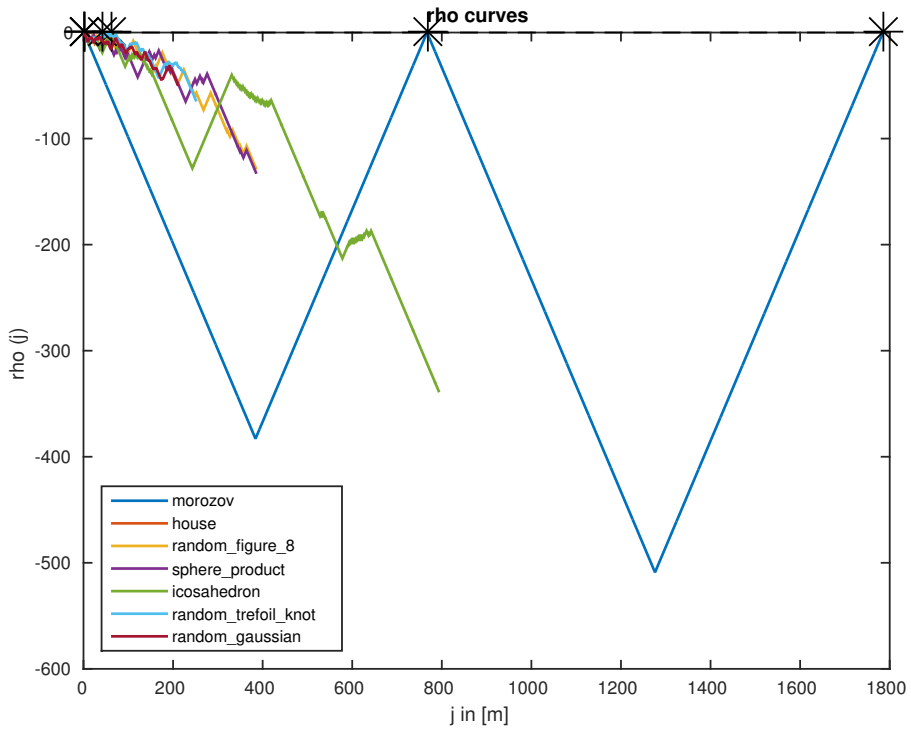


Figure 5.3: **ρ-curves for several complexes.** The ρ curve for a selection of complexes and for Morozov's complex of order 5 is shown. The ρ curve cuts the $\rho = 0$ level at five points in $[m]$.

a preprocessing step in the reduction algorithm.

The results in this section show that Morozov complex is so structured that it can be trivially reduced with a counting argument like the one given in Lemma 5.11. Though we do not expect this clearing strategy to generalise to other complexes, it gives evidence that worst-case complexes are very special and are not representative of the simplicial complexes that can be found in nature. This idea should be contrasted with the uncertainty principle (2.5) of [Candès and Romberg, 2006].

Chapter 6

Parallel multi-scale reduction

Algorithm 16 performs the reduction by sequentially minimising $\text{low}_\partial(j)$ by adding columns $j_0 < j$ for which $\text{low}_\partial(j_0) = \text{low}_\partial(j)$ until either there is no such column j_0 or column j has been set to zero. The computational overhead in Algorithm 16 is in computing the left-to-right column operations, and has a worst-case complexity of $\mathcal{O}(m^3)$ which is achieved by an example simplicial complex in [Morozov, 2005]. However, as we have argued in Section 5.5, Morozov's complex is a worst-case complex that is so structured that is reduction can be done trivially by a counting argument. It is expected that most of these worst-case complexes are edge-cases that are not observed in nature. Indeed, previous advances in reduction algorithms primarily focus on decreasing the computational cost by exploiting structure in ∂ to reveal some entries in ∂ can be set to zero [Bauer et al., 2014a, Chen and Kerber, 2011] or by parallelising the reduction by dividing the boundary matrix into blocks and partially reducing each block [Bauer et al., 2014a,b]. In this chapter, we propose a parallelistaion of the standard reduction algorithm that couples the speedups suggested in [Bauer et al., 2014a]. Our main contribution is to provide bounds on the location of $\text{low}^*(j)$, which are empirically observed to typically identify a large fraction of the $\text{low}^*(j)$, and suggest a framework by which the known $\text{low}^*(j)$ can be used to to reduce ∂ in a highly parallel fashion. Our innovations are based on an extension of the function $\text{left}(\cdot)$ presented in Definition 5.8. Specifically, they make use of a vector $\beta \in \mathbb{Z}_{m+1}^m$ that is computed with information derived from $\text{left}(\cdot)$ as suggested by the following definition.

Definition 6.1 (β_j)**.** *Let $\partial \in \mathbb{F}_2^{m\times m}$ be a boundary matrix and*

$$\mathcal{L}_j = \{i \in [m] : \text{left}(i) = j\}. \tag{6.1}$$

⋆ *The contributions of this chapter are joint work with Jared Tanner.*

Then, for $j \in [m]$ we let

$$\beta_j = \begin{cases} \max \mathcal{L}_j & \mathcal{L}_j \neq \emptyset \\ 0 & \mathcal{L}_j = \emptyset \end{cases} \tag{6.2}$$

In the rest of this chapter we discuss how the information in β can be used to:

1. identify columns that are already reduced,
2. design a parallelisation of Algorithm 16 and 17,
3. suggest priorities in reducing columns,
4. estimate topological information before the reduction terminates.

We now present Algorithm 18 and explain its particulars in the rest of this chapter.

6.1 Theoretical preamble

In this section we describe the theory behind the parallelisation strategy of Algorithm 18. The main algorithmic innovation is a strategy to efficiently distribute the workload of Algorithm 16 over $\mathcal{O}(m)$ processors to progressively entrywise minimize $\mathrm{low}_\partial \in \mathbb{Z}_{m+1}^m$. Additionally, Algorithm 18 is designed to take advantage of some structural patterns in the boundary matrix in order to minimise the total number of left-to-right column operations. The core observation is that, by simple inspection, the rows of the boundary matrix reveal a subset of nonzero entries in ∂ which cannot be modified in the reduction process, and consequently identify nonzero lower bounds on $\mathrm{low}^*(j)$ for a subset of $j \in [m]$.

The vector $\beta \in \mathbb{Z}_{m+1}^m$ carries a great deal of information about the nature of each column in the boundary matrix. Some of its properties are explored in Theorem 6.2.

Theorem 6.2 (Properties of β_j)**.** *Let $\partial \in \mathbb{F}_2^{m\times m}$ be a boundary matrix and β defined as in* (6.2)*. Then, the following statements hold.*

P.1 $\beta_j = 0 \Leftrightarrow \nexists i \in [m]\, s.t.\, \mathrm{left}(i) = j$

P.2 *β_j is invariant to left-to-right column operations.*

P.3 $\{j \in [m] : \beta_j > 0\} \subset \mathsf{Neg}$.

P.4 $\mathsf{Pos} \subset \{j \in [m] : \beta_j = 0\}$.

P.5 $\beta_j \leq \mathrm{low}^*(j) \leq \mathrm{low}_\partial(j)$ *for all* $j \in [m]$.

Algorithm 18: Parallel multi-scale reduction

Data: $\partial \in \mathbb{F}_2^{m\times m}$; `MAX_ITER` $\in [m]$ (for early stopping)
Result: $\mathrm{low}^* \in \mathbb{Z}_{m+1}^m$
`// Phase 0: Initialisation`
Build β_j according to (6.2);
Pivots $\leftarrow \{j \in [m] : \beta_j = \mathrm{low}_\partial(j) > 0\}$;
for $j \in$ Pivots **do**
  $\partial_{\mathrm{low}_\partial(j)} \leftarrow 0$;
end
iter $\leftarrow 0$;
while $\mathrm{low}_\partial \neq \mathrm{low}^*$ *or* iter $\leq$ `MAX_ITER` **do**
  `// Phase I: Local injections`
  for $d \in [\dim(K)]$ **do**
    maxCollision $\leftarrow 0$;
    for $j \in \{\ell \in K_d : \mathrm{low}_\partial(\ell) > 0\} \setminus$ Pivots **do**
      if $\mathrm{low}_\partial(j) >$ maxCollision **then**
        if $\mathrm{low}_\partial(j) \notin \mathrm{low}_\partial([j-1])$ **then**
          Pivots $\leftarrow$ Pivots $\cup \{j\}$;
        else
          maxCollision $\leftarrow \mathrm{low}_\partial(j)$;
        end
      end
    end
  end
  `// Phase II: Column reduction`
  for $j_0 \in$ Pivots **do**
    $\mathcal{N}(j_0) \leftarrow \{\ell \in [m] \setminus [j_0] : \mathrm{low}_\partial(\ell) = \mathrm{low}_\partial(j_0)\}$;
    for $j \in \mathcal{N}(j_0)$ **do**
      $\partial_j \leftarrow \partial_j + \partial_{j_0}$;
      if $\mathrm{low}_\partial(j) = \beta_j$ **then**
        Pivots $\leftarrow$ Pivots $\cup \{j\}$;
        $\partial_{\mathrm{low}_\partial(j)} \leftarrow 0$;
      end
    end
  end
  `// Increase iteration`
  iter $\leftarrow$ iter $+1$;
end
$\mathrm{low}^* \leftarrow \mathrm{low}_\partial$;

P.6 β can be computed in time $\mathcal{O}(\mathrm{nnz}(\partial))$.

Proof.

P.1 This follows directly from the definition of β_j.

P.2 Let $i \in [m]$ be a row of ∂ and $j = \mathrm{left}(i)$. Reduction of ∂_j is achieved by adding to ∂_j columns ∂_ℓ for $\ell < j$ By Definition 6.1 $\partial_{i,\ell} = 0$ for $\ell < j$ and hence $\partial_{i,j} = 1$ throughout the reduction. Consequently the set $\mathrm{left}(i)$ and values β_j are fixed in the reduction procedure.

P.3 If $\beta_j > 0$, then exists $i \in [m]$ such that $\mathrm{left}(i) = j > 0$. Then, $\partial_{i,j} = 1$ and there does not exist ∂_ℓ with $\ell < j$ such that $\partial_{i,\ell} = 1$. Hence $\partial_{i,j}$ can not be set to zero in the reduction, so $j \in \mathsf{Neg}$.

P.4 This result follows by taking complements on the result P.3.

P.5 By P.2 the set $\mathcal{L}_j$ in 6.1 are nonzero entries in ∂_j which cannot be modified in the reduction procedure and consequently $\mathrm{low}^*(j) \geq \beta_j$.

$$\mathrm{low}^*(j) \geq \max\{i \in \mathrm{supp}(\partial_j) : \mathrm{left}(i) = j\} = \beta_j$$

P.6 Note that when creating the boundary matrix, β can be constructed with the following procedure,

Algorithm 19: Building β

```
Data: Simplicial complex K = {σ_1, ..., σ_m}
Result: β_j for j ∈ [m]
for σ_j ∈ K do
    β_j ← 0;
    for σ_i ∈ bd(σ_j) do
        if σ_i has not been visited then
            β_j ← max(β_j, i);
            Mark σ_i as visited;
        end
    end
end
```

Algorithm 19 finishes in time proportional to $\sum_j |\,\mathrm{bd}(\sigma_j)|$, so it has complexity $\mathcal{O}\left(\mathrm{nnz}(\partial)\right)$.

□

The value of Theorem 6.2 is most immediately apparent in P.4 and in particular when $\mathrm{low}_\partial(j) = \beta_j$ in which case $\mathrm{low}^*(j)$ is determined. Our numerical experiments in Section 6.6 show empirically that a large number of columns can usually be identified as already being reduced without need of any column operations. Moreover, having access to a large number of reduced columns allows a massive parallelisation as described in Section 6.2 and illustrated in Section 6.6. While the calculation of β adds an initial computation to Algorithm 18, it can be computed at the matrix-reading cost of $\mathcal{O}(\mathrm{nnz}(\partial))$, so it can often be obtained as a by-product of creating the filtration without penalising the asymptotic computational complexity of constructing the matrix.

Additionally, the vector β gives a sufficient condition for a column to be in Neg. Knowledge that an unreduced column is necessarily negative can be used in numerous ways. For example, as discussed in sections 6.3, 6.4, and 6.5 this is useful to produce more reliable estimations of low^* in the case of early-stopping. Additionally, knowledge that a column is necessarily negative enhances a notion of clearing as discussed in Section 6.3.

Apart from Theorem 6.2, Algorithm 18 makes use of the observation that since low_∂ converges to an injection low^*, one can often inspect the sparsity pattern of ∂ to identify regions of $[m]$ where low_∂ is locally an injection. This local injection property is the basis of the following Theorems 6.3 and 6.4, which give sufficient conditions to identify sets $T \subset [m]$ such that $\mathrm{low}_\partial(j) = \mathrm{low}^*(j)$ for all $j \in T$. For a function $f : A \to B$, we let $f(A) = \{f(a) \in B : a \in A\}$ and $f(\emptyset) = \emptyset$. Hence, for a set $T \subset [m]$, we let $\mathrm{low}_\partial(T) = \{\mathrm{low}_\partial(j) : j \in T\}$ and $|\,\mathrm{low}_\partial(T)|$ be its cardinality. In what follows, we abuse notation and let K_p be the set of indices in $[m]$ suchthat $\dim(\sigma_j) = p$, that is,

$$K_p = \{j \in [|K|] : \dim(\sigma_j) = p\}\,. \tag{6.3}$$

Informally, the argument in Theorem 6.3 is that, for each dimension d, any subset of contiguous columns $T \subset K_d \setminus \{j \in K_d : \mathrm{low}_\partial(j) = 0\}$ with $\min T = \min K_d$ is already reduced if $\mathrm{low}_\partial : T \to [m]$ is an injection or, equivalently, if $|\,\mathrm{low}_\partial(T)| = |T|$.

Theorem 6.3 (Local injection I)**.** *Let $\partial \in \mathbb{F}_2^{m\times m}$ be the boundary matrix of a simplicial complex K and let $d \in [\dim(K)]$ and K_d defined as in* (6.3)*. Let*

$$T := K_d \setminus \{j \in K_d : \mathrm{low}_\partial(j) = 0\} \neq \emptyset$$

be such that $T = \{j_1, \ldots, j_{|T|}\}$ *for* $j_1 < \cdots < j_{|T|}$ *and for* $\ell \leq |T|$, *let* $T^\ell = \{j_1, \ldots, j_\ell\}$. *If*

$$\left|\mathrm{low}_\partial\left(T^\ell\right)\right| = \ell, \tag{6.4}$$

then $\mathrm{low}_\partial(j) = \mathrm{low}^*(j)$ *for all* $j \in T^\ell$.

Proof. Let T be defined as in Theorem 6.3. If $\ell = 1$, then there ∂_{j_1} is the first column of dimension d that appears in the filtration, so there is no column $j \in [j_1]$ that can reduce it. Hence, suppose that $\ell > 1$ and let $j_t \in T^\ell$ be such that $\mathrm{low}_\partial(j_t) \neq \mathrm{low}^*(j_t)$. Then there is $j < j_t$ such that $\mathrm{low}_\partial(j) = \mathrm{low}_\partial(j_t)$. This implies that $j \in K_d \cap [j_t]$ and that $\mathrm{low}_\partial(j) > 0$, so $j \in T_\ell$. Hence, $|T_\ell| = \ell$, but $|\,\mathrm{low}_\partial(T_\ell)| < \ell$, which contradicts (6.4). □

Theorem 6.4 addresses a generalisation of Theorem 6.3 and argues that for each dimension d, any subset of contiguous columns $T \subset K_d \setminus \{j \in K_d : \mathrm{low}_\partial(j) = 0\}$ is already reduced if $|\,\mathrm{low}_\partial(T)| = |T|$ and there is no column $j \in K_d \setminus T$ with $j < \min T$ such that $\mathrm{low}_\partial(j) \in \mathrm{low}_\partial(T)$.

Theorem 6.4 (Local injection II)**.** *Let* $\partial \in \mathbb{F}_2^{m\times m}$ *be the boundary matrix of a simplicial complex* K *and let* $d \in [\dim(K)]$. *Define* T *and* T^ℓ *as in Theorem 6.3. Assume that for* $k > 1$ *and* $k \leq \ell$ *it holds that*

$$\min \mathrm{low}_\partial(T^\ell \setminus T^{k-1}) > \max \mathrm{low}_\partial(T^{k-1}) \tag{6.5}$$

and

$$|\mathrm{low}_\partial\left(\{j_k, \ldots, j_\ell\}\right)| = \ell - k + 1. \tag{6.6}$$

Then, $\mathrm{low}_\partial(j) = \mathrm{low}^*(j)$ *for all* $j \in \{j_k, \ldots, j_\ell\}$.

Proof. Let T and T^ℓ be as in Theorem 6.4 so that $T^\ell \setminus T^{k-1} = \{j_k, \ldots, j_\ell\}$. Since $\ell > 0$ and $k > 1$, then $\ell - k + 1 > 0$. If $k = 2$, then $j_2 \in \mathrm{low}_\partial(T^\ell \setminus T^1)$ and invoking (6.5),

$$\mathrm{low}_\partial(j_2) \geq \min \mathrm{low}_\partial(T^\ell \setminus T^1) > \max \mathrm{low}_\partial(T^1) = \mathrm{low}_\partial(j_1),$$

so the result follows by invoking Theorem 6.3 with $\ell = 2$. Arguing as in Theorem 6.3, suppose that $k > 2$ and let $j_t \in \{j_k, \ldots, j_\ell\}$ be such that $\mathrm{low}_\partial(j_t) \neq \mathrm{low}^*(j_t)$. Then, there is $j < j_t$ such that $\mathrm{low}_\partial(j) = \mathrm{low}_\partial(j_t)$. Given that $j_k \leq j < j_t$ and by (6.5) then $j \in T^\ell \setminus T^{k-1}$. Hence, $|\,\mathrm{low}_\partial(T_\ell \setminus T^{k-1})| < \ell - k + 1$, which contradicts (6.6). □

We now describe how the results presented so far in this section interact in Algorithm 18.

6.2 Parallel multi-scale reduction

So far, Theorems 6.2 -6.4 identify a subset of the columns j in $\partial \in \mathbb{F}_2^{m\times m}$ for which $\text{low}^*(j)$ is known without need for performing column additions. In order to complete the reduction of ∂, that is identify low^*, it is necessary to perform column additions analogous to those in Algorithm 16; however,unlike the sequential ordering in Algorithm 16, we will propose massive parallelisation of the column additions. Towards this we define the set of columns $j \in [m]$ for which their $\text{low}^*(j)$ is known at any given iteration of a reduction algorithm as the Pivots:

$$\mathsf{Pivots} = \left\{ j \in [m] : \text{low}^*(j) \in [m] \text{ has been identified. } \right\}$$

Moreover, we define the set of column with which a pivot could be added to in order to reduce their low_∂ as the *neighbours* of column j:

$$\mathcal{N}(j) = \left\{ \ell > j : \text{low}_\partial(\ell) = \text{low}^*(j) \right\}. \tag{6.7}$$

Note that as low^* are an injection, if $j_1, j_2 \in \mathsf{Pivots}$ and $j_1 \neq j_2$, then $\mathcal{N}(j_1) \cap \mathcal{N}(j_2) = \emptyset$ so

$$\left(\bigcup_{j\in\mathsf{Pivots}} \mathcal{N}(j) \right) \bigcup \mathsf{Pivots} \subset [m] \tag{6.8}$$

is a partition and each column in $\mathcal{N}(j)$ for $j \in \mathsf{Pivots}$ can have their low reduced independently by adding column j to columns $\ell \in \mathcal{N}(j)$. In sequential algorithms like Algorithms 16 and 17, the set $\mathcal{N}(j)$ is constructed only after the block $[j-1] \cap K_{\dim(\sigma_j)}$ has been fully reduced. In contrast, Algorithm 18 starts by inspecting the sparsity pattern of ∂ to identify a large set of *seed* Pivots that are then used to build (6.7) for each $j \in \mathsf{Pivots}$. Given that (6.8) is a partition, each set $\mathcal{N}(j)$ is partially reduced independently, typically in parallel and new columns are appended to Pivots whenever their update is such that their $\text{low}^*(\ell)$ is identified by either by Theorems 6.2 - 6.4. We elaborate on these stages of Algorithm 18, and its convergence, in the remainder of this section.

6.2.1 Phase 0: Initialising Pivots

The first step of Algorithm 18 is to compute the vector β, which can be done at cost $\mathcal{O}(\text{nnz}(\partial))$ by Theorem 6.2. This vector is then coupled with low_∂ to build the set of initial pivots

$$\mathsf{Pivots} \leftarrow \{ j \in [m] : \beta_j = \text{low}_\partial(j) \},$$

which by Theorem 6.2, contains indices for columns that are already reduced, e.g. $\text{low}^*(j)$ is known for $j \in \mathsf{Pivots}$. The clearing strategy from Lemma 5.6 is applied to each of these columns to zero out their corresponding positive pairs. The resulting set of Pivots is then passed on to the main iteration.

6.2.2 Phase I: Finding local injections

In the first phase of the main iteration, the following Algorithm 20 is applied to each dimension $d \in [\dim(K)]$.

Algorithm 20: Finding local injections

```
Data: ∂ ∈ 𝔽₂^{m×m}; Pivots ⊂ Neg
Result: Pivots′ ⊂ Neg s.t. Pivots ⊂ Pivots′
maxCollision ← 0;
for j ∈ {ℓ ∈ K_d : low_∂(ℓ) > 0} \ Pivots do
    if low_∂(j) > maxCollision then
        if low_∂(j) ∉ low_∂([j − 1]) then
            Pivots ← Pivots ∪ {j};
        else
            maxCollision ← low_∂(j);
        end
    end
end
```

Algorithm 20 starts by initialising maxCollision $\leftarrow 0$ and walking through the non-zero columns in K_d that have not been identified as pivots. The idea is to identify regions of K_d where Theorems 6.3 and 6.4 can be applied to guarantee that a local injection exists. At each j, the variable maxCollision tracks the largest $i \in \text{low}_\partial([j-1]) \cap \text{low}_\partial(K_d)$ such that $i = \text{low}_\partial(j_1) = \text{low}_\partial(j_2)$ for $j_1 \neq j_2$. In terms of Theorem 6.4, maxCollision plays the rôle of $\max \text{low}_\partial(T^{j-1})$ in (6.5), since if $\text{low}_\partial(j) >$ maxCollision and $\text{low}_\partial(j) \notin \text{low}_\partial([j-1])$, then there is no column $\ell < j$ that can have $\text{low}_\partial(\ell) = \text{low}_\partial(j)$ and consequently $\text{low}_\partial(j)$ cannot be further reduced and therefore its $\text{low}^*(j)$ is known.

6.2.3 Main iteration II: Parallel column reduction

Once the first phase has been completed, the sets of $\mathcal{N}(j)$ are computed for each $j \in \mathsf{Pivots}$. Each column in $\bigcap_{j \in \mathsf{Pivots}} \mathcal{N}(j)$ has their associated pivot added to them; that is, $\partial_\ell \leftarrow \partial_\ell + \partial_j$ is carried out for each $\ell \in \mathcal{N}(j)$ for $j \in \mathsf{Pivots}$. Algorithm 18 then marks the column ℓ as reduced if $\text{low}_\partial(\ell) = \beta_\ell$, ℓ is added to the set of Pivots,

and the clearing strategy is called to set the associated positive column $\partial_{\mathrm{low}^*(\ell)}$ to zero.

6.2.4 Convergence

The convergence of Algorithm 18 is a consequence of the interplay between Phases I and II of the main iteration. We describe this mechanism in the following theorem.

Theorem 6.5 (Convergence of Algorithm 18)**.** *Let $\partial \in \mathbb{F}_2^{m\times m}$ be the boundary matrix of a simplicial complex K. Then, Algorithm 18 converges to* low^* *in at most m iterations.*

Proof. Let $\mathsf{Pivots}^\ell \subset [m]$ be the set of indices of columns that are known to be already reduced at the start of iteration ℓ, and let $\mathrm{low}_\partial{}^\ell$ be the corresponding estimate of low^*. We show that Algorithm 16 converges by showing that if $\mathrm{low}^* \neq \mathrm{low}_\partial{}^\ell$, then either

$$|\,\mathsf{Pivots}^{\ell+1}\,| \geq |\,\mathsf{Pivots}^{\ell}\,| + 1 \tag{6.9}$$

or,

$$\|\,\mathrm{low}_\partial{}^{\ell+1} - \mathrm{low}^*\,\| < \|\,\mathrm{low}_\partial{}^{\ell} - \mathrm{low}^*\,\|. \tag{6.10}$$

Note that $|\,\mathsf{Pivots}^0\,| \geq \dim(K) \geq 1$ since at Phase 0 at least the first column of each dimension must be identified as a pivot.

Suppose that at the start of iteration $\ell \geq 0$ the boundary matrix ∂ is still not reduced so that $\mathrm{low}_\partial{}^\ell \neq \mathrm{low}^*$. If a column is identified as reduced at the end of Phase I, then it is marked as a pivot so $|\,\mathsf{Pivots}^{\ell+1}\,| \geq |\,\mathsf{Pivots}^{\ell}\,| + 1$ and we are done. Hence, assume that no new pivots are identified. In this case, for each $j \in \mathsf{Pivots}^\ell$ we construct the set $\mathcal{N}(j)$ as in (6.7).

If $\exists j_0 \in \mathsf{Pivots}^\ell$ such that $|\mathcal{N}(j_0)| \geq 1$, then Algorithm 18 enters Phase II to reduce columns in $\mathcal{N}(j_0)$. In this case, (6.10) holds and we are done. Otherwise, suppose $|\mathcal{N}(j_0)| = 0$ for all $j \in \mathsf{Pivots}^\ell$. In this case, since the matrix is not reduced, the set

$$\mathcal{C} = \{j \in [m] : |\mathcal{N}(j)| > 0\}$$

is not empty. Let,

$$j_{\min} = \min \mathcal{C} \tag{6.11}$$

and let $d = \dim(\sigma_{j_{\min}})$. There are two possible cases,

1. $j_{\min} = \min K_d$: In this case, the column must have been identified as a pivot at Phase 0, which is a case previously excluded.

2. $j_{\min} > \min K_d$: Let

$$T = K_d \setminus \{j \in K_d : \mathrm{low}_\partial(j) = 0\}.$$

In this case, it must hold that $|\mathrm{low}_\partial([j_{\min}] \cap T)| = |[j_{\min}] \cap T|$ since otherwise, there would exist an $\ell \in [j_{\min} - 1] \cap T$ such that $\mathrm{low}_\partial(\ell) = \mathrm{low}_\partial(j_{\min})$ implying that $|\mathcal{N}(\ell)| > 0$ and thus contradicting the minimality of $j_{\min}$. Hence, by Theorem 6.3, $\mathrm{low}_\partial([j_{\min}] \cap T) = |[j_{\min}] \cap T|$, so $j_{\min}$ must have been identified at Phase I of the iteration. This is again, a contradiction.

Therefore, it is impossible to have $|\mathcal{N}(j_0)| = 0$ for all $j_0 \in \mathsf{Pivots}^\ell$ if no pivots were identified at Phase I of the iteration. Hence, as long as $\mathrm{low}_\partial{}^\ell \neq \mathrm{low}^*$ either (6.9) or (6.10) hold, so at each iteration Algorithm 18 will increase the number of pivots, decrease some entries in low_∂, or both. □

The multi-scale nature of Algorithm 18 is advantageous to implement a number of strategies that allow us progressively refine our knowledge of low^*. We describe some of these strategies in the next subsections.

6.3 Clearing by compression

In this section we describe a clearing strategy based on a column compression arguments inspired by the previous compression arguments presented in [Bauer et al., 2014a] to reduce operation flop-count. This strategy, in its simplest form presented in Corollary 6.7, can be straightforwardly implemented in a parallel architecture in Algorithm 18, but is not explicitly listed in Algorithm 18 for conciseness. Here, we present a compression result that can in some cases help clear columns in the matrix and in some other cases can give improved bounds on the location of low^*. *In doing so, it will be convenient to define the set* $\mathsf{Paired}^\ell \subset \mathsf{Paired}$*, as the sets of elements in* (5.16) *that have been identified at the* ℓ*-th iteration by Algorithm 18.*

Note that Paired^ℓ at iteration ℓ includes the set of unreduced columns that have been identified as negative. In particular, Theorem 6.2 guarantees that for any ℓ,

$$\{j \in [m] : \beta_j > 0\} \subset \mathsf{Neg} \subset \mathsf{Paired}^\ell.$$

Lemma 6.6 (Clearing by local compression)**.** *Let* $\partial \in \mathbb{F}_2^{m \times m}$ *be a boundary matrix of a simplicial complex* K*. Let* $j \in [m]$ *and* $d = \dim(\sigma_j) \in [\dim(K)]$*. Then at iteration* ℓ *of Algorithm 18,*

$$\mathrm{low}^*(j) \in L_j := \{0\} \cup (K_{d-1} \cap [\beta_j, \mathrm{low}_\partial(j)]) \setminus \mathsf{Paired}^\ell. \tag{6.12}$$

Proof. If $j \in [m]$, and $d = \dim(\sigma_j)$, then $\text{low}^*(j) \in K_{d-1}$ by the definition of ∂; moreover, $\text{low}^*(j) \in [\beta_j, \text{low}_\partial(j)]$ by Theorem 6.2. The injection condition on the support of low^* implies that if $\text{low}^*(j)$ can not be equal to any of the observed positive low^*'s. Finally, $\text{low}^*(j)$ can not be equal to the index of any of the identified negative columns as $\partial_{\text{low}^*(j)}$ is necessarily positive, and can not be equal to any of the indices of paired positive columns by the injection condition and Lemma 5.6. Hence, $\text{low}^*(j) \notin \mathsf{Paired}^\ell$. □

Lemma 6.6 implies the following corollary,

Corollary 6.7. *Let $\partial \in \mathbb{F}_2^{m\times m}$ and let L_j be defined as in* (6.12) *and such that $|L_j| = 1$. Let $i \in L_j$. If $i = 0$, then $\text{low}^*(j) = 0$. Else, $i = \text{low}^*(j)$ and consequently column j is negative and fully reduced and ∂_i is positive.*

6.4 Essential estimation

Another way to reuse the partial knowledge of negative and associated positive columns, as given by $\mathsf{Paired}^\ell \subset \mathsf{Paired}$ be the set of columns that have been identified as paired at the ℓ-th iteration of Algorithm 18, is to estimate the essential simplices

Lemma 6.8 (Essential estimation)**.** *Let $\partial \in \mathbb{F}_2^{m\times m}$ be a boundary matrix and $j \in [m]$, then at iteration ℓ*

$$\mathsf{Ess} \subset \left\{[m] \setminus \mathsf{Paired}^\ell\right\} \setminus \{\text{low}_\partial(j) : j \in [m]\} \tag{6.13}$$

Proof. Clearly, $\mathsf{Ess} \cap \mathsf{Paired}^\ell = \emptyset$ for all ℓ by the partition on the columns. We show that $\mathsf{Ess} \cap \{\text{low}_\partial(j) : j \in [m]\} = \emptyset$. To do this, we follow a proof by contradiction and suppose that $\text{low}_\partial(j) \in \mathsf{Ess}$, which implies

$$\nexists p \,\text{s.t.}\, \text{low}^*(p) = \text{low}_\partial(j) \tag{6.14}$$

as otherwise p is the index of a negative column and $\text{low}^*(p)$ the index of an associated positive non-essential column. We show that column j can not be either positive or negative, which leads to a contradiction.

1. $j \in \mathsf{Pos}$: As we are determining $\mathsf{Ess} \cap \{\text{low}_\partial(j) : j \in [m]\}$, we consider only the j such that $\text{low}_\partial(j) > 0$, but since $j \in \mathsf{Pos}$ we must have $\text{low}^*(j) = 0$. By the definition of j being positive there must exist $q < j$ such that $\text{low}^*(q) = \text{low}_\partial(j)$ in order that j can be reduced to the zero column; however, this contradicts (6.14) by setting $p = q$.

2. $j \in \mathsf{Neg}$: By Theorem 6.2 $\text{low}_\partial(j) \geq \text{low}^*(j)$. If $\text{low}_\partial(j) = \text{low}^*(j)$, then (6.14) is contradicted with $p = j$. Otherwise, if $\text{low}_\partial(j) > \text{low}^*(j)$, then by the definition of $\text{low}^*(j)$ there must exist $q < j$ such that $\text{low}^*(q) = \text{low}_\partial(j)$ in order to reduce column j. This again, contradicts (6.14)

□

Theorem 6.8 implies that the set of essential columns can be iteratively estimated by a set $E^\ell \subset [m]$ initialised at $E^0 = [m]$ and iteratively updated as,

$$E^{\ell+1} \leftarrow \left(E^\ell \setminus \mathsf{Paired}^\ell\right) \setminus \{\text{low}_\partial(j) : j \in [m]\}\,. \tag{6.15}$$

6.5 Distributing the workload

Central to the efficacy of Algorithm 18 is the typically large number of pivots available to reduce subsequent columns coupled with the ability of all column additions in its Phase 2 to be performed in parallel. We do not explicitly describe parallelisation strategies for Phase 2 due to architecture specificity, but make a few remarks regarding communication minimisation and early termination in Secs. 6.5.1 and 6.5.2 respectively.

6.5.1 Prioritise reduction of sets $\mathcal{N}(j)$ with large cardinality

Let,

$$\mathcal{C} = \{j \in \mathsf{Pivots} : |\mathcal{N}(j)| > 0\}\,, \tag{6.16}$$

be the set of columns that can be used in a given iteration to decrease low_∂.

If only $p < |\mathcal{C}|$ processors are available, and communication of these active pivots is the dominant cost, then it may be desirable to implement only a portion of the possible column additions available in Phase 2 duration an iteration. In such a setting, a priority queue can be used to order the sets $\mathcal{N}(j)$ based on their cardinality and act on multiples of p sets of active columns per iteration of Phase 2.

6.5.2 Prioritise reduction of $\mathcal{N}(j) \cap \mathsf{Neg}$

In settings where low^* cannot be fully determined due to computational or time constraints, the reduction ordering can be prioritised to minimize various objectives. One such objective is to minimize $\| \text{low}_\partial - \text{low}^* \|$ in a suitable norm as rapidly as possible. Recall, Theorem 6.2 gives bounds on the value of low^* and low_∂ is available by inspection. Due to the the identification of fully reduced negative column, say column

j allows clearing of column $\text{low}^*(j)$ there is benefit in prioritising the reduction of columns which are known to be negative, e.g. $\{j \in [m] : \beta_j > 0\} \subset \mathsf{Neg}$. Alternative prioritisations would be application specific, but can include such information as prioritising values for which the persistence is greater or improving the estimation of essential columns as described in Section 6.4.

6.6 Numerical experiments

In this section we evaluate the efficacy of our parallel multi-scale reduction Algorithm 18 empirically by comparing its performance against the standard reduction Algorithm 16 and standard reduction with a twist Algorithm 17. Our evaluation is on a set of synthetic simplicial complexes from point-clouds sampled from a set of a predefined set of ensembles. The point-clouds and their corresponding Rips-Vietoris simplicial complexes are generated with the `Javaplex` [Tausz et al., 2014] library. When building the simplicial complex, we supply the following parameters,

1. Number of points (N): The number of points in the point-cloud to be generated.
2. Maximum dimension ($\dim K$): The maximum dimension of the resulting simplicial complex K.
3. Maximum filtration value (r_T): The maximum radius in a predefined grid of scales $\{r_1, \ldots, r_T\} \subset [0, \infty)$ used to build the filtration.
4. Number of divisions (h): The grid of scales is uniform with spacing h, that is $h := r_{i+1} - r_i$ for $i \in [T-1]$.

Ensemble	N	$\dim K$	r_T	h
Gaussian Points in $\mathbb{R}^3$	15	5	5	10
Figure-8	15	5	5	10
Trefoil Knot	15	5	5	10
$\mathbb{S}^2 \times \mathbb{S}^2$	15	5	5	10

Central to our numerical evaluations in this manuscript is the number of iterations required. The reduction in Algorithms 16 and 17 is an *horizontal* procedure in the sense that no column in a given dimension is reduced before the preceeding columns in this dimension have been reduced. In contrast, Algorithm 18 is a *diagonal* procedure as it implements an horizontal type of reduction in Phases I and II, but rather than completely reducing the columns in order, it progressively prunes the matrix *vertically* by doing left-to-right column operations every time there is enough information to

guarantee that this is possible. Given this crucial difference in design, it becomes difficult to find appropriate benchmarking scenarios and, more fundamentally, to provide a notion of *iteration* that is agnostic to the algorithm's architecture. Hence, in our numerical experiments, we let an iteration be the set of operations that are performed at each cycle of the outer-most loop of the algorithm. This is a reasonable convention as it is consistent with each of the algorithm's concurrency and is also natural for the high-level pseudo-code description of their design. However, we note that this notion of iteration can be optimistic or pessimistic depending on the unit of computational overhead that is being measured. In our experiments, *left-to-right* column operations are chosen as the main unit of computational overhead and, indeed, our algorithm has been designed to minimise this kind of operations. Given the embarrassingly parallel nature of Phase II of Algorithm 18, this model is generous with our algorithm as it does not consider possible limitations to the number of processors available and it does not account for the communication overhead between processors. Finally, as our algorithm's practical performance is greatly limited by the hardware architecture, we point out that our numerical experiments should serve as a proof-of-concept of a design that can yield a powerful production implementation rather than a trustworthy comparison of the run-time or flop-count in the general case.

The numerical results presented in this section simulate a parallel implementation[1] A truly parallel implementation is left as future work, see 7.

6.6.1 Operations are packed in fewer iterations

For each ensemble we sample three point-clouds and reduce their corresponding boundary matrices using each of Algorithms 16 - 17. Figure 6.1 illustrates the computational complexity, as the number of column additions performed at each iteration and also the number of nontrivial *xor* operations when adding columns in the matrix; that is, the number of scalar operations of the form

$$\{1 \oplus 1, 1 \oplus 0, 0 \oplus 1\}\,.$$

Scalar operations are relevant because in some storage models like the one given in [Chen and Kerber, 2011] the computational cost of updating the matrix depends super-linearly on the number of non-zeros in the columns being added. Figures 6.1a, 6.1g and 6.1j show the number of column additions per iteration of each algorithm

[1] The code is available at https://github.com/rodrgo/tda.

Sample	Gaussian		Figure-8		Trefoil-knot		$\mathbb{S}^2 \times \mathbb{S}^2$	
	std	twist	std	twist	std	twist	std	twist
1	0.52	1.00	0.49	1.00	0.46	1.00	0.50	1.00
2	0.52	1.00	0.51	1.00	0.40	1.00	0.49	1.01
3	0.50	1.00	0.50	1.00	0.44	1.00	0.47	1.00

Table 6.1: **Ratio of total column additions.** For each ensemble, three point clouds are sampled and the corresponding simplicial complexes are reduced with each algorithm. The *ratio of total column additions* between Algorithm 18 and both Algorithms 16 and 17 is reported.

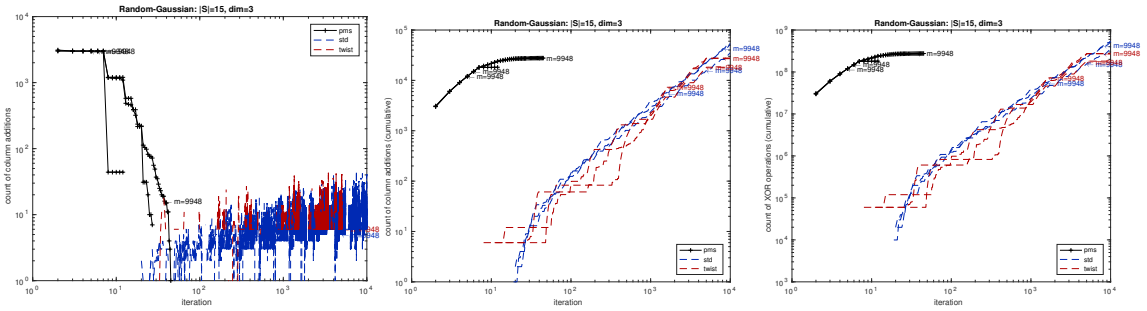
performs to reduce each of the simplicial complexes under consideration. Specifically, they illustrate how that Algorithm 18 allocates most of the column additions at the earliest iterations due to its highly parallel nature. On the other hand, Figures 6.1b, 6.1h, and 6.1k as well as Tab. 6.1 show that while Algorithm 18 reduces the number of iterations, it has a total number of column additions indistingushable to Algorithm 17 and about half that of Algorithm 16. This shows that Algorithm 18 is successful in packing the number of left-to-right column operations into considerably few independent iterations.

6.6.2 low^* is approximated at multiple scales

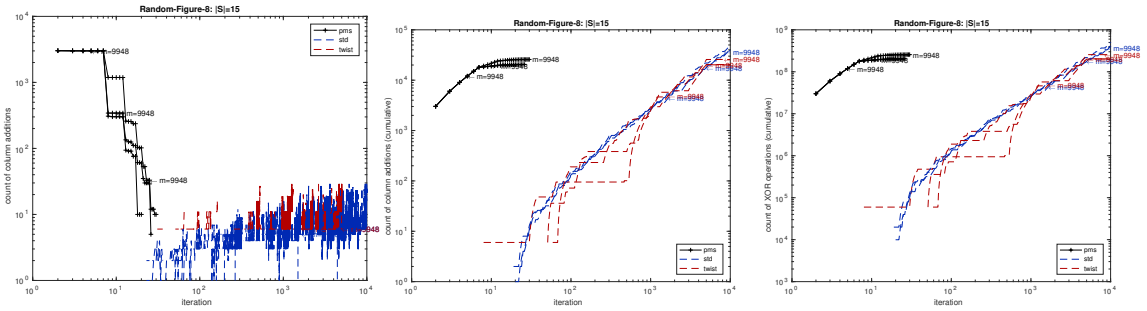
Let $\mathrm{low}_\partial{}^\ell$ be the estimate of low^* at the ℓ-th iteration of an algorithm. We evaluate the quality of the approximation by computing the relative ℓ_1-error as

$$\mathrm{error}^\ell = \frac{\| \mathrm{low}_\partial{}^\ell - \mathrm{low}^* \|_1}{\| \mathrm{low}^* \|_1}. \tag{6.17}$$

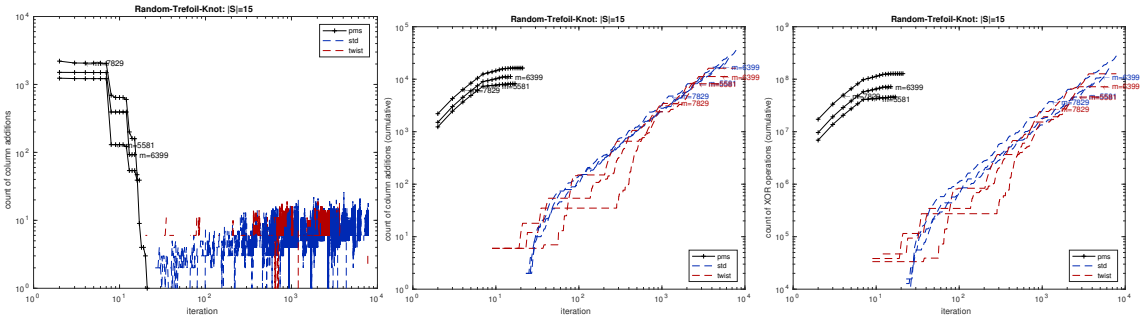
Figure 6.2 illustrates an improved rate of reduction in error^k per iteration as each of the algorithms progresses. This is achieved without resorting to further prioritisation of this reduction as described in Sec. 6.5.2 as we are simulating a number of processors in excess of the number of column additions needed per iteration. Tab. 6.2 shows the precise number of iterations each of Algorithms 16 - 17 need to achieve (6.17) less than 10^{-k} for $k = 1, 2, 3, 4$ and complete reduction. Tab. 6.2 shows the rapid reduction of (6.17) by Algorithm 18 compared to only appreciable reduction by Algorithms 16 and 17 near their complete reductions; e.g. wheras a relative reduction of (6.17) to 1% is achieved by Algorithm 18 in less than 20 iterations, Algorithms 16 and 17 take approximately ten and nine thousand iterations respectively for the same reduction. Fig. 6.3 and Tab. 6.3 show a similarly early decrease in the fraction of the number



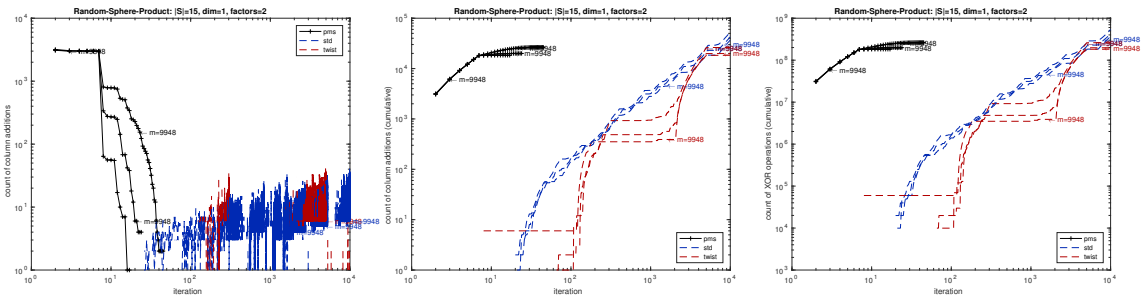
(a) Gaussian: Count col. adds (b) Gaussian: Cumsum col. adds (c) Gaussian: Count of `XOR` ops.



(d) Figure-8: Count of col. adds (e) Figure-8: Cumsum col. adds (f) Figure-8: Count of `XOR` ops.



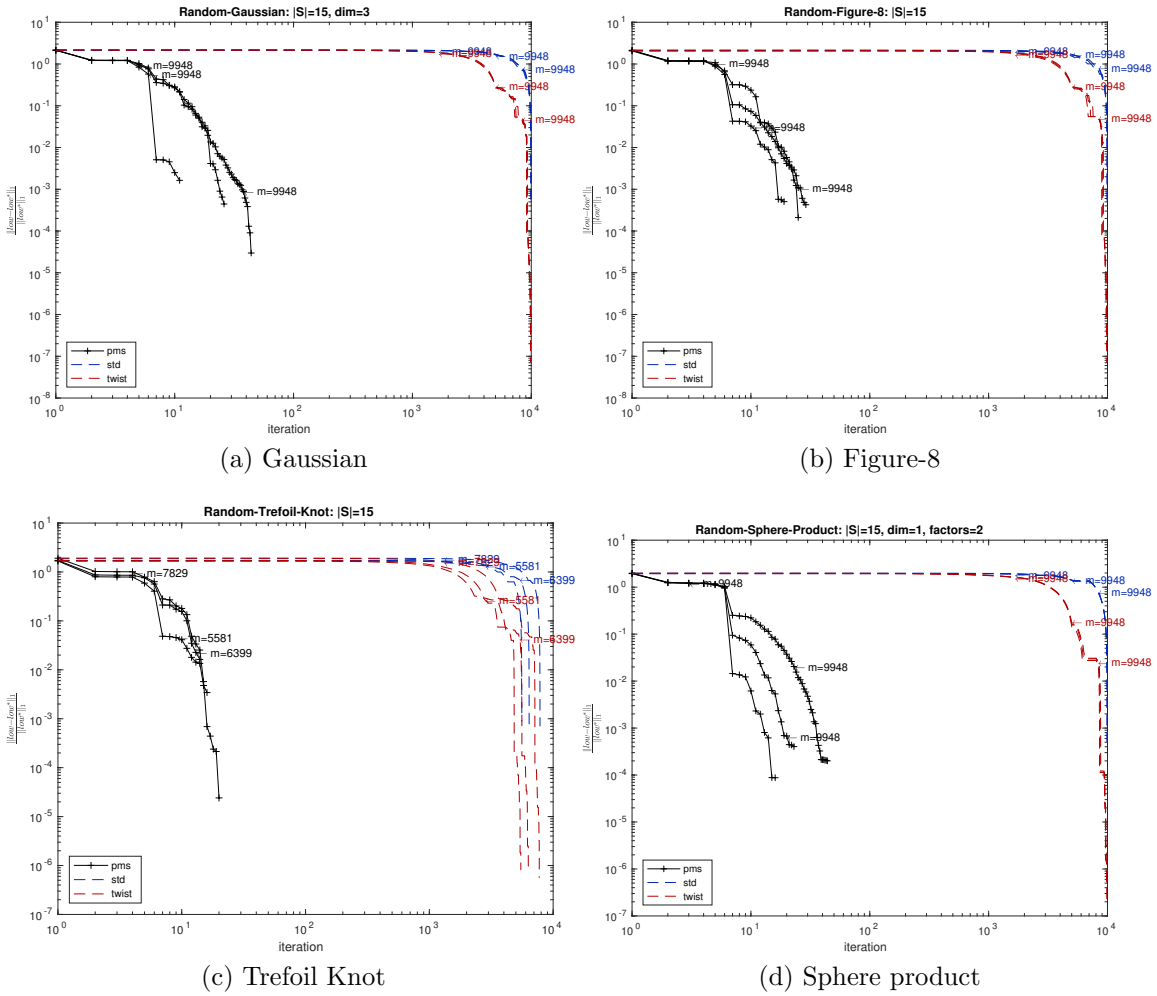
(g) Trefoil-K: Count col. adds (h) Trefoil-K: Cumsum col. adds (i) Trefoil-K: Count of `XOR` ops.



(j) $\mathbb{S}^2 \times \mathbb{S}^2$: Count col. adds (k) $\mathbb{S}^2 \times \mathbb{S}^2$: Cumsum col. adds (l) $\mathbb{S}^2 \times \mathbb{S}^2$: Count of `XOR` ops.

Figure 6.1: **Benchmarking on column operation overhead.** For each ensemble, three point clouds are sampled and their corresponding simplicial complexes are reduced with Algorithms 16 - 17. Their performance is benchmarked in three different ways: through the number of column additions done *at each* iteration (Figures 6.1a, 6.1d, 6.1g, 6.1j); through the cumulative number of column additions *up to* a given iteration (Figures 6.1b, 6.1e, 6.1h, 6.1k); and through the cumulative number of `XOR` operations done *up to* a given iteration (Figures 6.1c, 6.1f, 6.1i, 6.1l).

of columns which are fully reduced. Remarkably, Algorithm 18 has reduced at least 50% of the columns in only two iterations, and all but 1% within no more than 20 iterations; this is in contrast with between approximately two and a half and ten thousand iterations for compreable reductions by Algorithms 16 and 17.



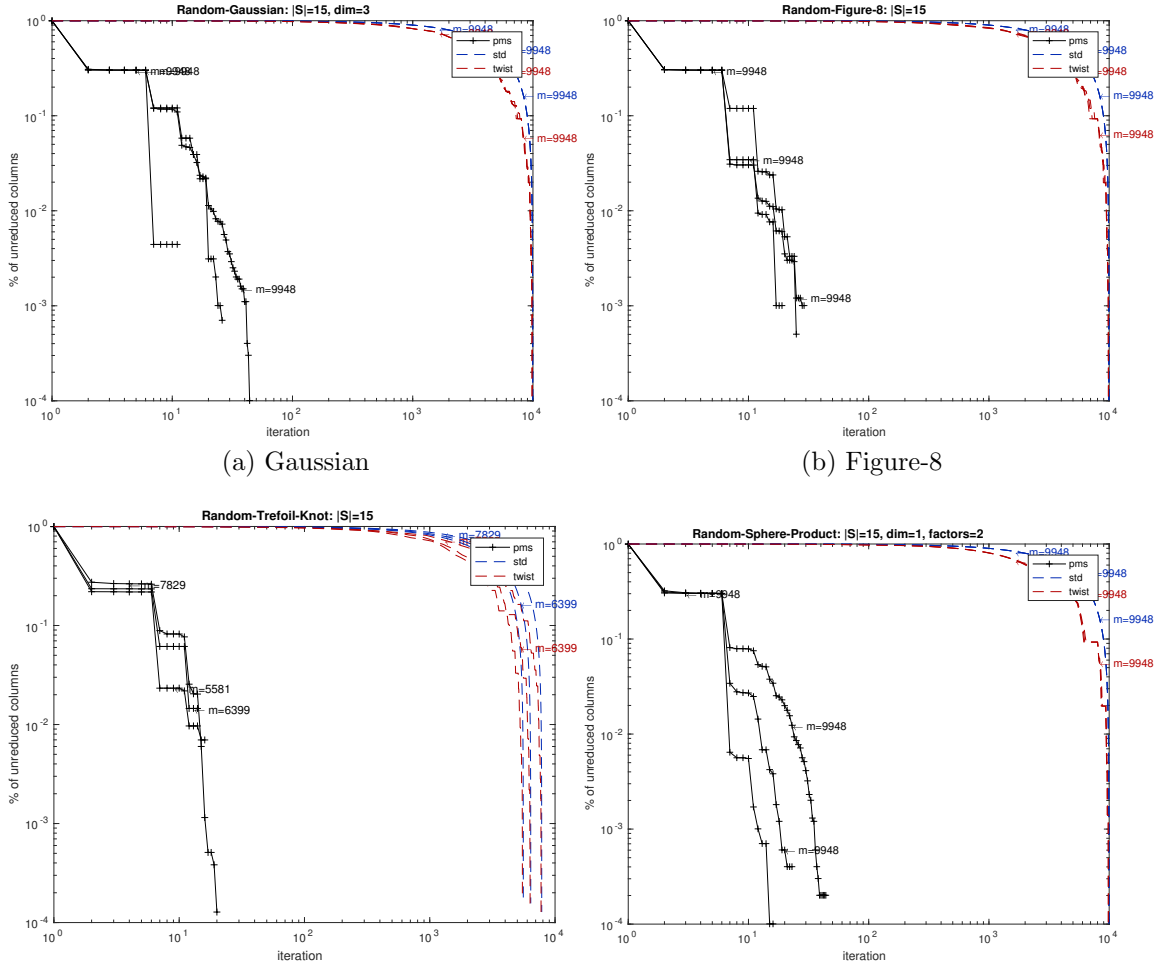
(a) Gaussian (b) Figure-8

(c) Trefoil Knot (d) Sphere product

Figure 6.2: **Relative ℓ_1-error per iteration.** For each ensemble, three point clouds are sampled and the corresponding simplicial complexes are reduced with each of the algorithms. The relative ℓ_1-error between low_∂ and low^*, as given in (6.17), is tracked.

6.6.3 The set Ess can be reliably estimated after a few iterations

Finally, we show how the efficacy of Lemma 6.8 in estimating the the set of essential columns. Let E^ℓ be defined as in (6.15), by Lemma 6.8, $\mathsf{Ess} \subset E^\ell$ for every ℓ. If E^ℓ is our estimation at iteration ℓ, then the number of true positives, false positives and



(a) Gaussian (b) Figure-8

(c) Trefoil Knot (d) Sphere product

Figure 6.3: **Proportion of unreduced columns per iteration.** For each ensemble, three point clouds are sampled and the corresponding simplicial complexes are reduced with each of Algorithms 16 - 17. The proportion of unreduced columns is presented at each iteration.

$\frac{\|\,\mathrm{low}_\partial - \mathrm{low}^*\,\|_1}{\|\,\mathrm{low}^*\,\|_1}$	Gaussian			Figure-8			Trefoil-knot			$\mathbb{S}^2 \times \mathbb{S}^2$		
	pms	std	twist	pms	std	twist	pms	std	twist	pms	std	twist
0.1	14	9786	7366	9	9756	6759	12	7687	5519	7	9751	5589
0.01	20	9933	9180	17	9930	8916	15	7816	7087	10	9930	8485
0.001	24	9948	9197	25	9948	8942	16	7829	7122	13	9948	8564
0.0001	27	9949	9198	26	9949	8944	20	7830	7309	15	9949	9388
0	27	9949	9858	26	9949	9869	21	7830	7732	17	9949	9840

Table 6.2: **Iterations to relative ℓ_1-error.** For each ensemble, one point cloud is sampled and the corresponding simplicial complex is reduced with each algorithm. The number of iterations to achieve a given *relative ℓ_1-error level* between low_∂ and low^*, as given by (6.17), is reported for of Algorithms 16 - 17.

Proportion	Gaussian			Figure-8			Trefoil-knot			$\mathbb{S}^2 \times \mathbb{S}^2$		
	pms	std	twist	pms	std	twist	pms	std	twist	pms	std	twist
0.90	2	996	528	2	996	499	2	784	392	2	996	519
0.50	2	4975	3405	2	4975	2974	2	3916	2536	2	4975	2958
0.10	12	8955	7424	7	8955	6849	7	7048	6140	7	8955	6115
0.05	15	9452	8432	7	9452	8618	12	7439	6738	7	9452	8404
0.01	20	9850	9471	17	9850	9494	15	7752	7427	7	9850	9461

Table 6.3: **Iterations to unreduced percentage.** For each ensemble, one point cloud is sampled and the corresponding simplicial complex is reduced with each of Algorithms 16 - 17. The number of iterations to achieve a given *proportion of unreduced columns* is presented for each algorithm.

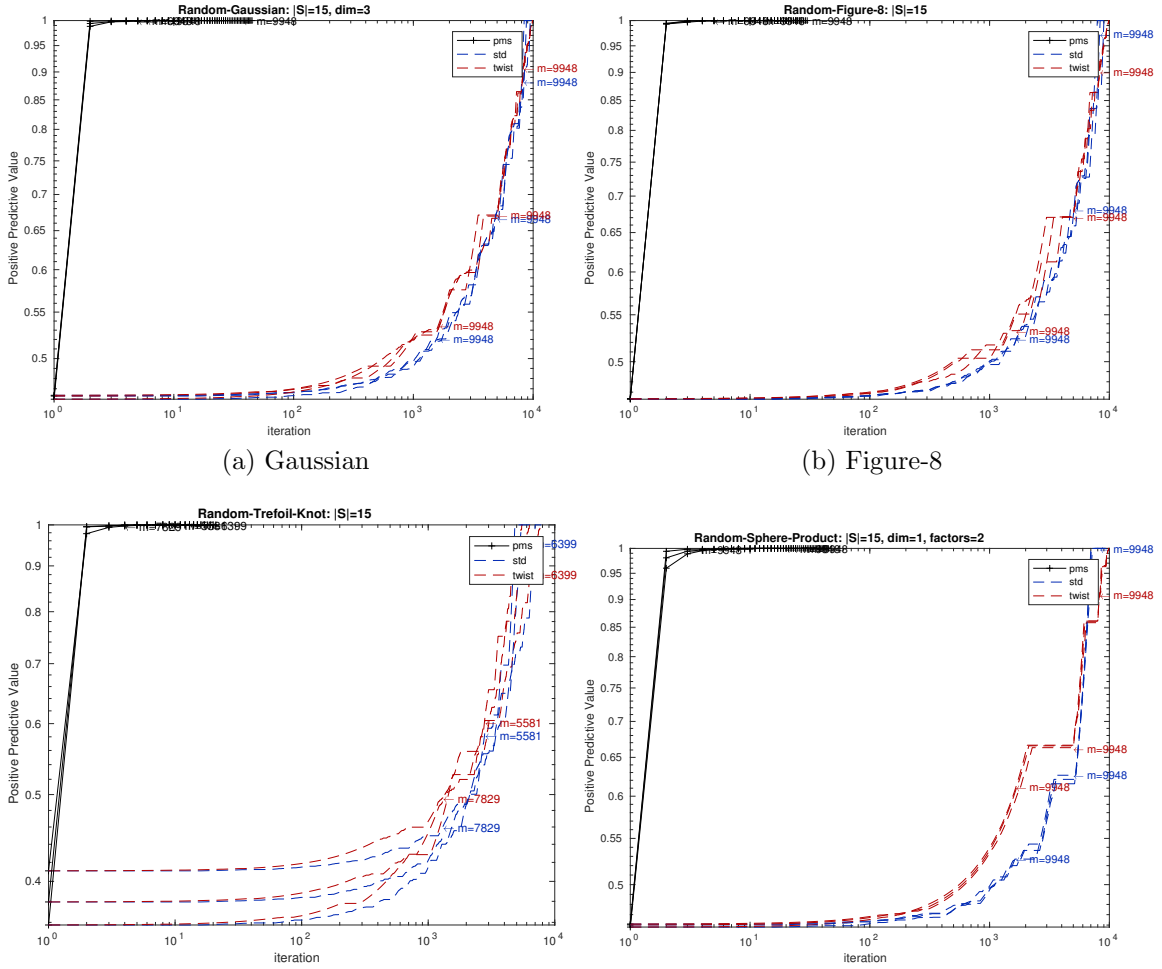
false negatives in the estimation is, respectively,

$$\begin{aligned} \mathrm{TP} &= |\,\mathsf{Ess} \cap E^\ell| = |\,\mathsf{Ess}\,|, \\ \mathrm{FP} &= |E^\ell \setminus \mathsf{Ess}\,| = |E^\ell| - |\,\mathsf{Ess}\,|, \\ \mathrm{FN} &= |\emptyset| = 0. \end{aligned}$$

To evaluate the quality of the estimation, we compute the *precision* as defined by

$$\mathrm{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \frac{|\,\mathsf{Ess}\,|}{|E^\ell|}$$

and note that the *recall* $= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$ is equal to 1 due to the estimate giving no false negatives. Figure 6.4 and Tab. 6.4 show the remarkably few iterations needed by Algorithm 18 to achieve precision near one while Algorithms 16 and 17 show increase in the precision to one only in the later iterations. Specifically, Algorithm 18 achieves precision of 95% with two iterations and complete precision within 8 iterations whereas Algorithms 16 and 17 require between seven and ten thousand iterations for similar precisions.



(a) Gaussian (b) Figure-8

(c) Trefoil Knot (d) $\mathbb{S}^2 \times \mathbb{S}^2$

Figure 6.4: **Precision of Ess estimation.** For each ensemble, three point clouds are sampled and the corresponding simplicial complexes are reduced with each algorithm. At each iteration, Ess is estimated and its precision is tracked.

Precision	Gaussian			Figure-8			Trefoil-knot			$\mathbb{S}^2 \times \mathbb{S}^2$		
	pms	std	twist	pms	std	twist	pms	std	twist	pms	std	twist
0.10	1	1	1	1	1	1	1	1	1	1	1	1
0.50	2	1033	559	2	1231	493	2	2132	1374	2	1016	557
0.90	2	8461	8244	2	7599	8238	2	6691	6445	2	6675	8222
0.95	2	8636	8938	2	7774	8892	2	6871	7123	2	6850	8519
1.00	8	8794	9858	5	7932	9869	8	7004	7732	7	7008	9866

Table 6.4: **Iterations to essential-estimation precision.** For each ensemble, one point cloud is sampled and the corresponding simplicial complex is reduced with each algorithm. The number of iterations to achieve a given *precision* of the set Ess is given for each algorithm.

Chapter 7

Conclusions and future work

In this thesis we have reported advances for two areas in the Mathematics of Information: combinatorial compressed sensing and computational algebraic topology.

Noiseless combinatorial compressed sensing: We have proposed two algorithms Parallel-ℓ_0 (Algorithm 12) and Serial-ℓ_0 (Algorithm 11) to decode a signal $x \in \chi_k^n$ from Ax when it is known that $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$ and that a dissociated structure on the problem can be assumed to exist. We argue that most signals have a dissociated structure and that if not, then the condition can be imposed on the matrix. Our theoretical guarantees ensure that both algorithms provably converge in $\mathcal{O}(dn \log k)$ operations and that they achieve very high phase transitions for dissociated data. In particular, Parallel-ℓ_0 is observed to be empirically the fastest compressed sensing algorithm in this scenario.

Noisy combinatorial compressed sensing: We showed in Chapter 4 that the algorithmic framework of Parallel-ℓ_0 and Serial-ℓ_0 can be adapted to account for measurements corrupted by additive noise. This is done by computing the asymptotic posterior distribution of an entry in the residual being zero or being equal to another entry in the residual given the corrupted measurements. This Bayesian approach to decoding was implemented in Robust-ℓ_0 (Algorithm 15) and its four variants (Table 4.1). We showed that the resulting algorithms inherit some desirable properties from Parallel-ℓ_0 like high phase transitions for $\delta \to 0$ and $n \to \infty$ as well as low computational latency. Finally, we perform numerical experiments to show that Robust-ℓ_0 is the best algorithm in cases of moderate signal-to-noise ratio and $\rho \lessapprox 0.3$.

Computation of persistent homology: We have extended the foundational algorithms of [Bauer et al., 2014a, Chen and Kerber, 2011, Edelsbrunner et al., 2002] to a

massively parallel algorithm for the reduction of boundary matrices. Our numerical experiments show that Algorithm 18 is able to pack more operations into few iterations, and to approximate low* simultaneously at all scales of the simplicial complex filtration. Our numerical experiments show that Algorithm 18, as compared with Algorithms 16 and 17, is able to pack more operations into few iterations, approximates low* simultaneously at all scales of the simplicial complex filtration, and determines the essential columns in remarkably few iterations. This massively parallel algorithm suggests the reduction of dramatically larger boundary matrices will now be possible, and moreover allows early termination with accurate results when computational constraints are reached.

7.1 Future work and open questions

The contributions of this thesis have opened up several avenues of future work which we are currently exploring.

1. *Expander dictionary learning*: In Chapter 3 we showed how the mapping

$$(Ax, A) \mapsto x$$

can be computed efficiently for $A \in \mathbb{E}_{k,\varepsilon,d}^{m\times n}$ and $x \in \chi_k^n$. We conjecture that there is enough information in the problem to efficiently and accurately compute the mapping

$$Ax \mapsto (A, x).$$

2. *Expander sketching for optimisation problems*: A common strategy in high-dimensional optimisation problems and in Bayesian optimisation is to use an sketching operator to project the data into a lower-dimensional subspace. The RIC-1 property in expander matrices could lead to better provable guarantees for particular optimisation methods.

3. *Convergence guarantees for Algorithm 18*: Algorithm 18 currently has no convergence guarantees. We conjecture that a convergence guarantee of $\mathcal{O}(\log m)$ iterations is possible.

4. *Production implementation of Algorithm 18*: A truly parallel implementation of Algorithm 18 is being currently developed.

Bibliography

Fernando Affentranger and Rolf Schneider. Random projections of regular simplices. *Discrete & Computational Geometry*, 7(1):219–226, 1992.

Noga Alon and Michael Capalbo. Explicit unique-neighbor expanders. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 73–79. IEEE, 2002.

Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29. ACM, 1996.

Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. In *ICML*, pages 584–592, 2014.

Bubacarr Bah and Jared Tanner. Vanishingly sparse matrices and expander graphs, with application to compressed sensing. *IEEE transactions on information theory*, 59(11):7491–7508, 2013.

Richard G Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 2008.

Richard G Baraniuk, Volkan Cevher, Marco F Duarte, and Chinmay Hegde. Model-based compressive sensing. *Information Theory, IEEE Transactions on*, 56(4): 1982–2001, 2010.

Yuliy M Baryshnikov and Richard A Vitale. Regular simplices and gaussian samples. *Discrete & Computational Geometry*, 11(2):141–147, 1994.

Leonid Alexandrovich Bassalygo and Mark Semenovich Pinsker. Complexity of an optimum nonblocking switching network without reconnections. *Problemy Peredachi Informatsii*, 9(1):84–87, 1973.

Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Clear and compress: Computing persistent homology in chunks. In *Topological Methods in Data Analysis and Visualization III*, pages 103–117. Springer, 2014a.

Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Distributed computation of persistent homology. In *2014 Proceedings of the Sixteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 31–38. SIAM, 2014b.

Richard Bellman and Robert Kalaba. On adaptive control processes. *IRE Transactions on Automatic Control*, 4(2):1–9, 1959.

Radu Berinde and Piotr Indyk. Sequential sparse matching pursuit. In *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pages 36–43. IEEE, 2009.

Radu Berinde, Anna C Gilbert, Piotr Indyk, H Karloff, and Martin J Strauss. Combining geometry and combinatorics: A unified approach to sparse signal recovery. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 798–805. IEEE, 2008a.

Radu Berinde, Piotr Indyk, and Milan Ruzic. Practical near-optimal sparse recovery in the l1 norm. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 198–205. IEEE, 2008b.

Gregory Beylkin, Ronald Coifman, and Vladimir Rokhlin. Fast wavelet transforms and numerical algorithms i. *Communications on pure and applied mathematics*, 44 (2):141–183, 1991.

Jeffrey D Blanchard and Jared Tanner. Gpu accelerated greedy algorithms for compressed sensing. *Mathematical Programming Computation*, 5(3):267–304, 2013.

Jeffrey D Blanchard and Jared Tanner. Performance comparisons of greedy algorithms in compressed sensing. *Numerical Linear Algebra with Applications*, 22(2):254–282, 2015.

Jeffrey D Blanchard, Jared Tanner, and Ke Wei. Cgiht: conjugate gradient iterative hard thresholding for compressed sensing and matrix completion. *Information and Inference: A Journal of the IMA*, 4(4):289–327, 2015.

Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009.

Thomas Blumensath and Mike E Davies. Normalized iterative hard thresholding: Guaranteed stability and performance. *Selected Topics in Signal Processing, IEEE Journal of*, 4(2):298–309, 2010.

Karl-Heinz Borgwardt. The simplex algorithm: A probabilistic analysis, algorithms and combinatorics, 1987.

Károly Böröczky Jr and Martin Henk. Random projections of regular polytopes. *Archiv der Mathematik*, 73(6):465–473, 1999.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

Stephen Poythress Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Robert S Boyer and J Strother Moore. *MJRTY—a fast majority vote algorithm*. Springer, 1991.

Emmanuel J Candès and David L Donoho. Curvelets: A surprisingly effective non-adaptive representation for objects with edges. Technical report, Stanford Univ Ca Dept of Statistics, 2000.

Emmanuel J Candès and Justin Romberg. l1-magic: Recovery of sparse signals via convex programming. *www.acm.caltech.edu/l1magic/downloads/l1magic.pdf*, 4:14, 2005.

Emmanuel J Candès and Justin Romberg. Quantitative robust uncertainty principles and optimally sparse decompositions. *Foundations of Computational Mathematics*, 6(2):227–254, 2006.

Emmanuel J Candès and Terence Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215, 2005.

Emmanuel J Candès and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *Information Theory, IEEE Transactions on*, 52(12):5406–5425, 2006.

Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, 2006a.

Emmanuel J Candès, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006b.

Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

Michael Capalbo, Omer Reingold, Salil Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 659–668. ACM, 2002.

Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

Gunnar Carlsson. Topological pattern recognition for point cloud data. *Acta Numerica*, 23:289, 2014.

Gunnar Carlsson, Tigran Ishkhanov, Vin De Silva, and Afra Zomorodian. On the local behavior of spaces of natural images. *International journal of computer vision*, 76(1):1–12, 2008.

Joseph Minhow Chan, Gunnar Carlsson, and Raul Rabadan. Topology of viral evolution. *Proceedings of the National Academy of Sciences*, 110(46):18566–18571, 2013.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Automata, languages and programming*, pages 784–784, 2002.

Chao Chen and Michael Kerber. Persistent homology computation with a twist. In *Proceedings 27th European Workshop on Computational Geometry*, volume 11, 2011.

Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

Sheng Chen, Stephen A Billings, and Wan Luo. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of control*, 50(5):1873–1896, 1989.

Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of symbolic computation*, 9(3):251–280, 1990.

Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

Nello Cristianini and Matthew W Hahn. *Introduction to computational genomics: a case studies approach*. Cambridge University Press, 2006.

Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55(5):2230–2249, 2009.

George B Dantzig and Mukund N Thapa. *Linear programming 1: introduction*. Springer Science & Business Media, 2006.

Ingrid Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 41(7):909–996, 1988.

Geoffrey Davis, Stéphane Mallat, and Zhifeng Zhang. Adaptive time-frequency decompositions. *OPTICAL ENGINEERING-BELLINGHAM-INTERNATIONAL SOCIETY FOR OPTICAL ENGINEERING-*, 33:2183–2183, 1994.

Baron de Prony. Gaspard-clair-françois-marie riche. essai expérimental et analytique sur les lois de la dilatabilité des fluides élastique et sur celles de la force expansive de la vapeur de l'eau et de la vapeur de l'alkool,a différentes températures. *J. de l'Ecole Polytechnique*, 1:24–76, 1795.

Vin De Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Dualities in persistent (co) homology. *Inverse Problems*, 27(12):124003, 2011a.

Vin De Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Persistent cohomology and circular coordinates. *Discrete & Computational Geometry*, 45(4):737–759, 2011b.

David Donoho and Jared Tanner. Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing. *Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci.*, 367 (1906):4273–4293, 2009a. ISSN 1364-503X. doi: 10.1098/rsta.2009.0152. URL

http://dx.doi.org/10.1098/rsta.2009.0152. With electronic supplementary materials available online.

David Donoho and Jared Tanner. Counting faces of randomly projected polytopes when the projection radically lowers dimension. *Journal of the American Mathematical Society*, 22(1):1–53, 2009b.

David L Donoho. High-dimensional centrally symmetric polytopes with neighborliness proportional to dimension. *Discrete & Computational Geometry*, 35(4):617–652, 2006a.

David L Donoho. For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution. *Communications on pure and applied mathematics*, 59(6):797–829, 2006b.

David L Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

David L Donoho and Xiaoming Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, 47(7):2845–2862, 2001.

David L Donoho and Philip B Stark. Uncertainty principles and signal recovery. *SIAM Journal on Applied Mathematics*, 49(3):906–931, 1989.

David L Donoho and Jared Tanner. Exponential bounds implying construction of compressed sensing matrices, error-correcting codes, and neighborly polytopes by random sampling. *IEEE Transactions on Information Theory*, 56(4):2002–2016, 2010a.

David L Donoho and Jared Tanner. Precise undersampling theorems. *Proceedings of the IEEE*, 98(6):913–924, 2010b.

David L Donoho, Arian Maleki, and Andrea Montanari. Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences*, 106(45): 18914–18919, 2009.

David Leigh Donoho. Neighborly polytopes and sparse solution of underdetermined linear equations. 2005.

David Leigh Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006c.

Pier Luigi Dragotti and Yue M Lu. On sparse representation in fourier and local bases. *IEEE Transactions on Information Theory*, 60(12):7888–7899, 2014.

Marco F Duarte, Mark A Davenport, Dharmpal Takbar, Jason N Laska, Ting Sun, Kevin F Kelly, and Richard G Baraniuk. Single-pixel imaging via compressive sampling. *IEEE signal processing magazine*, 25(2):83–91, 2008.

Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28(4):511–533, 2002.

Cristian Estan and George Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems (TOCS)*, 21(3):270–313, 2003.

Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 2016.

David Forsyth and Jean Ponce. *Computer vision: a modern approach*. Upper Saddle River, NJ; London: Prentice Hall, 2011.

Simon Foucart. Hard thresholding pursuit: an algorithm for compressive sensing. *SIAM Journal on Numerical Analysis*, 49(6):2543–2563, 2011.

Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*. Springer, 2013.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

Dennis Gabor. Theory of communication. part 1: The analysis of information. *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, 93(26):429–441, 1946.

MJ Garside. The best sub-set in multiple regression analysis. *Applied Statistics*, pages 196–200, 1965.

Robert Ghrist. Elementary applied topology. *Book in preperation*, 2014.

Anna C Gilbert, Sudipto Guha, Piotr Indyk, Yannis Kotidis, Sivaramakrishnan Muthukrishnan, and Martin J Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 389–398. ACM, 2002.

Anna C. Gilbert, Yannis Kotidis, S Muthukrishnan, and Martin J Strauss. One-pass wavelet decompositions of data streams. *IEEE Transactions on knowledge and data engineering*, 15(3):541–554, 2003a.

Anna C Gilbert, S Muthukrishnan, and Martin J Strauss. Approximation of functions over redundant dictionaries using coherence. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 243–252. Society for Industrial and Applied Mathematics, 2003b.

Rémi Gribonval and Morten Nielsen. Sparse representations in unions of bases. *IEEE transactions on Information theory*, 49(12):3320–3325, 2003.

Kanghui Guo, Gitta Kutyniok, and Demetrio Labate. Sparse multidimensional representations using anisotropic dilation and shear operators, 2006.

Venkatesan Guruswami, James R Lee, and Alexander Razborov. Almost euclidean subspaces of $\ell 1$n via expander codes. *Combinatorica*, 30(1):47–68, 2010.

Alfred Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3):331–371, 1910.

Jacques Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton university bulletin*, pages 49–52, 1902.

Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Simple and practical algorithm for sparse fourier transform. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1183–1194. Society for Industrial and Applied Mathematics, 2012.

Ishay Haviv and Oded Regev. The restricted isometry property of subsampled fourier matrices. *arXiv preprint arXiv:1507.01768*, 2015.

Werner Heisenberg. Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. *Zeitschrift für Physik*, 43:172–198, 1927.

Monika Rauch Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. Computing on data streams. *External memory algorithms*, 50:107–118, 1998.

Peter CJ Hill. Dennis gabor-contributions to communication theory & signal processing. In *EUROCON, 2007. The International Conference on" Computer as a Tool"*, pages 2632–2637. IEEE, 2007.

Piotr Indyk. Sketching, streaming and sublinear-space algorithms. *Graduate course notes, available at*, 2007.

Arieh Iserles and Carola-Bibiane Schönlieb. Mathematics of information. 2013.

Sina Jafarpour, Weiyu Xu, Babak Hassibi, and Robert Calderbank. Efficient and robust compressed sensing using optimized expander graphs. *Information Theory, IEEE Transactions on*, 55(9):4299–4308, 2009.

William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Lee K Jones. On a conjecture of huber concerning the convergence of projection pursuit regression. *The annals of statistics*, pages 880–882, 1987.

Boris S Kashin and Vladimir N Temlyakov. A remark on compressed sensing. *Mathematical notes*, 82(5):748–755, 2007.

Mark Kelbert and Yu M Suhov. *Information theory and coding by example*. Cambridge University Press, 2013.

Seung-Jean Kim, Kwangmoo Koh, Michael Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior-point method for large-scale l1-regularized least squares. *IEEE journal of selected topics in signal processing*, 1(4):606–617, 2007.

Achim Klenke. *Probability theory: a comprehensive course*. Springer Science & Business Media, 2013.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

John M Lee. *Smooth manifolds*. Springer, 2003.

FC Leone, LS Nelson, and RB Nottingham. The folded normal distribution. *Technometrics*, 3(4):543–550, 1961.

Shlomo Levy and Peter K Fullagar. Reconstruction of a sparse spike train from a portion of its spectrum and application to high-resolution deconvolution. *Geophysics*, 46(9):1235–1243, 1981.

Michael G Luby, Michael Mitzenmacher, Mohammad Amin Shokrollahi, and Daniel A Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569–584, 2001.

PY Lum, G Singh, A Lehman, T Ishkanov, Mikael Vejdemo-Johansson, M Alagappan, J Carlsson, and G Carlsson. Extracting insights from the shape of complex data using topology. *Scientific reports*, 3, 2013.

Yanting Ma, Dror Baron, and Deanna Needell. Two-part reconstruction in compressed sensing. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 1041–1044. IEEE, 2013.

Stéphane Mallat. *A wavelet tour of signal processing*. Academic press, 1999.

Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

Christopher D Manning, Hinrich Schütze, et al. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.

Rodrigo Mendoza-Smith and Jared Tanner. Expander ℓ_0-decoding. *Applied and Computational Harmonic Analysis*, 2017a.

Rodrigo Mendoza-Smith and Jared Tanner. Parallel multi-scale reduction of persistent homology filtrations. *arXiv preprint arXiv:1708.04710*, 2017b.

Rodrigo Mendoza-Smith, Jared Tanner, and Florian Wechsung. A robust parallel algorithm for combinatorial compressed sensing. *arXiv preprint arXiv:1704.09012*, 2017.

Nikola Milosavljević, Dmitriy Morozov, and Primoz Skraba. Zigzag persistent homology in matrix multiplication time. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pages 216–225. ACM, 2011.

Dmitriy Morozov. Persistence algorithm takes cubic time in worst case. *BioGeometry News, Dept. Comput. Sci., Duke Univ*, 2, 2005.

Ali Mousavi and Richard G Baraniuk. Learning to invert: Signal recovery via deep convolutional networks. *arXiv preprint arXiv:1701.03891*, 2017.

Shanmugavelayutham Muthukrishnan et al. Data streams: Algorithms and applications. *Foundations and Trends® in Theoretical Computer Science*, 1(2):117–236, 2005.

Deanna Needell and Joel A Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3): 301–321, 2009.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Mark Newman. *Networks: an introduction*. Oxford university press, 2010.

Monica Nicolau, Arnold J Levine, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, 2011.

DW Oldenburg, T Scheuer, and S Levy. Recovery of the acoustic impedance from reflection seismograms. *Geophysics*, 48(10):1318–1337, 1983.

Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6(1):17, 2017.

Athanasios Papoulis and Christodoulos Chamzas. Improvement of range resolution by spectral extrapolation. *Ultrasonic Imaging*, 1(2):121–135, 1979.

Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44. IEEE, 1993.

Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

Fadil Santosa and William W Symes. Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1307–1330, 1986.

Shriram Sarvotham, Dror Baron, and Richard G Baraniuk. Compressed sensing reconstruction via belief propagation. *preprint*, pages 1–24, 2006a.

Shriram Sarvotham, Dror Baron, and Richard G Baraniuk. Sudocodes - fast measurement and reconstruction of sparse signals. In *2006 IEEE International Symposium on Information Theory*, pages 2804–2808. IEEE, 2006b.

Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

Donald R Sheehy. The persistent homology of distance functions under random projection. In *Proceedings of the thirtieth annual symposium on Computational geometry*, page 328. ACM, 2014.

Michael Sipser and Daniel A Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, 1996.

Dharmpal Takhar, Jason Laska, Michael B Wakin, Marco F Duarte, Dror Baron, Shriram Sarvotham, Kevin Kelly, and Richard G Baraniuk. A new compressive imaging camera architecture using optical-domain compression. *IS&T/SPIE Computational Imaging IV*, 6065(606509):1, 2006.

Michel Talagrand. A new look at independence. *The Annals of probability*, pages 1–34, 1996.

Terence Tao. 254a, notes 1: Concentration of measure. *What's new (Blog)*, 2010.

Terence Tao. *Topics in random matrix theory*, volume 132. AMS Bookstore, 2012.

Terence Tao and Van H Vu. *Additive combinatorics*, volume 105. Cambridge University Press, 2006.

Andrew Tausz, Mikael Vejdemo-Johansson, and Henry Adams. JavaPlex: A research software package for persistent (co)homology. In Han Hong and Chee Yap, editors, *Proceedings of ICMS 2014*, Lecture Notes in Computer Science 8592, pages 129–136, 2014. Software available at http://appliedtopology.github.io/javaplex/.

Dane Taylor, Florian Klimm, Heather A Harrington, Miroslav Kramár, Konstantin Mischaikow, Mason A Porter, and Peter J Mucha. Topological data analysis of contagion maps for examining spreading processes on networks. *Nature communications*, 6, 2015.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Andrey Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Meth. Dokl.*, 4:1035–1038, 1963.

Lloyd N Trefethen. Cubature, approximation, and isotropy in the hypercube. *SIAM Review*, 59(3):469–491, 2017.

Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

Joel A Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information theory*, 50(10):2231–2242, 2004.

Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on information theory*, 53(12): 4655–4666, 2007.

Leslie G Valiant. Graph-theoretic arguments in low-level complexity. In *International Symposium on Mathematical Foundations of Computer Science*, pages 162–176. Springer, 1977.

Anatoly M Vershik and Piotr V Sporyshev. An asymptotic estimate of the average number of steps of the parametric simplex method. *USSR Computational Mathematics and Mathematical Physics*, 26(3):104–113, 1986.

Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

Weiyu Xu and Babak Hassibi. Efficient compressive sensing with deterministic guarantees using expander graphs. In *Information Theory Workshop, 2007. ITW'07. IEEE*, pages 414–419. IEEE, 2007.

Fan Zhang and Henry D Pfister. Verification decoding of high-rate ldpc codes with applications in compressed sensing. *IEEE Transactions on Information Theory*, 58 (8):5042–5058, 2012.