Compressed sensing for error correction on real-valued vectors
Abstract

Compressed sensing (CS) is a relatively new branch of mathematics with very interesting applications in signal processing, statistics and computer science. This thesis presents some theory of compressed sensing, which allows us to recover (high-dimensional) sparse vectors from (low-dimensional) compressed measurements by solving the $\ell_1$-minimization problem. A possible application of CS to the problem of error correction is also presented, where sparse vectors are that of arbitrary noise. Successful sparse recovery by $\ell_1$-minimization relies on certain properties of rectangular matrices. But these matrix properties are extremely subtle and difficult to numerically verify. Therefore, to get an idea of how sparse (or dense) errors can be, numerical simulation of error correction was done. These simulations show the performance of error correction with respect to various levels of error sparsity and matrix dimensions. It turns out that error correction degrades slower for low matrix dimensions than for high matrix dimensions, while for sufficiently sparse errors, high matrix dimensions offer a higher likelihood of guaranteed error correction.

Acknowledgements

I would like to thank my supervisor Patrik Wahlberg for suggesting the very interesting topic of compressed sensing for a bachelor’s thesis.
1 Introduction

Picture a scenario where a certain measurement device (a sensor) is to measure a signal made up of a sequence of \( n \) data points, and at the same time compress this data sequence into \( m \) data points, where \( m \) is much smaller than \( n \). Such a scenario is interesting if the number \( n \) of data points is far too much to handle, because then it is desirable to compress a signal into less storage space. The relatively new branch of mathematical research called compressed sensing deals with circumstances in which this scenario of sensing and compression is possible. In its most basic form, compressed sensing regards retrieval of high-dimensional vectors from compressed measurements of said vectors. These measurements are compressed (linearly mapped to a lower-dimensional space) at the same stage as they are measured (sensed), hence the name of the subject. Compressed sensing grew from the work of Candès, Romberg, Tao [2-6, 8] and Donoho [10] in the 2000s, where they showed that a signal having a sparse representation can be recovered exactly from a small set of linear measurements.

In the context of compressed sensing (CS), measurement and compression takes place at the same time. Of course, if one could easily handle the \( n \)-samples-long signal after having measured it, then CS could be ignored and other compression techniques could instead be used. Here, the act of simultaneous measurement and compression is referred to as sensing measurement, and it corresponds to multiplication of a vector \( x \in \mathbb{R}^n \) by a special sensing matrix \( A \in \mathbb{R}^{m \times n} \), resulting in a measurement vector \( y \in \mathbb{R}^m \). In other words, an input vector \( x \) is linearly mapped by \( A \) to a lower-dimensional vector \( y \), and \( x \) is somewhere among the infinitely many solutions to the system

\[ Ax = y. \]

We want this compression of \( x \) into \( y \) to be lossless (ie. no information about \( x \) is lost in \( y \)), which means that we would like it to be possible to retrieve \( x \) from knowledge of only \( A \) and \( y \). Since \( m \) is significantly smaller than \( n \), this linear system above is severely underdetermined. Therefore, projection of a vector in \( \mathbb{R}^n \) into \( \mathbb{R}^m \) in this fashion does not seem to achieve much, since isolating \( x \) from solutions to the system above would be like finding a needle in an infinite haystack. But a key observation underlying CS is that most signals of interest usually have a special type of structure to them, called sparsity, and that sparsity can enable successful retrieval of \( x \) (from \( A \) and \( y \)) by means of special algorithms.

Vectors are sparse when they are made up of only a handful of non-zero components. For example, the vector \( x \) itself could have only a few non-zero components. But worth mentioning is that sparsity is not limited to the canonical basis. A vector \( x \in \mathbb{R}^n \) can instead be sparse in some other basis represented by a basis transformation matrix \( \Phi \in \mathbb{R}^{n \times n} \). Given \( x = \Phi c \), the alternative vector \( c \) could be sparse. In that case, measurement is done using the matrix \( A' = A\Phi^{-1} \) so that \( A'x = (A\Phi^{-1})(\Phi c) = A(\Phi^{-1}\Phi)c = Ac = y \) corresponds to
a measurement of the sparse vector $c$. Of course, $\Phi$ and its inverse (and the matrix products denoted) may be extremely time-consuming to compute, but they can be done once beforehand. For simplicity, we will only consider sparsity in the canonical basis.

Given a vector $x \in \mathbb{R}^n$, a matrix $A$ and a vector $y = Ax \in \mathbb{R}^m$, is it possible to retrieve $x$ with knowledge of only $A$ and $y$? Indeed it is, as long as $x$ is sparse enough with respect to $A$. By constraining $x$ to be sparse, $x$ can be successfully retrieved from $A$ and $y$ by solving the $\ell_0$-minimization problem

$$\min_{x \in \mathbb{R}^n} \|z\|_0 \quad \text{s.t.} \quad Az = y.$$  \hspace{1cm} (P0)

(See the Preliminaries section for clarification on the notation used here.) By definition, any algorithm solving this problem prioritizes sparse solutions, which makes it reasonable to believe that this does in fact recover sparse vectors. By also constraining the sensing matrix $A$ to have specific properties, we can take a step further and completely replace (P0) with the $\ell_1$-minimization problem

$$\min_{x \in \mathbb{R}^n} \|z\|_1 \quad \text{s.t.} \quad Az = y$$ \hspace{1cm} (P1)

for recovering sparse vectors. (These two problems will be frequently referred to, so the reader is encouraged to keep them in mind.) This replacement might not seem necessary, but it turns out that several efficient numerical algorithms have been developed for solving (P1) [11, 12]. While (P0) is theoretically more straightforward for recovering sparse vectors, numerical algorithms for solving it are extremely cumbersome if not intractable, which is not the case for solving (P1). However, sparse recovery by (P1) is not advantageous for all matrices $A$, because it has the (not so slight) downside that it requires a special kind of null space for $A$. Thus, we are interested in constraints on $A$, under which sparse solutions to the theoretically simpler problem (P0) and the practically simpler problem (P1) coincide.

In this thesis, we present some compressed sensing theory which allows sparse vector recovery to be feasible. This is done by showing some properties that $A$ needs to have for sparse vectors to be retrievable by (P1). Building on this theory, we also present a possible application of CS to the problem of error correction. It turns out that by encoding a vector into a higher-dimensional redundant vector, it is possible to completely remove an error of the encoded vector provided that the error is sparse enough. Some numerical simulations of parts of such an error correcting procedure are also presented, which aims to test the limits of such error correction.
2 Preliminaries

For the thesis to make sense, some definitions and notation must first be made clear. Everything described in this section will be taken for granted further on. Unless stated otherwise, the three variables \( k \leq m \leq n \) shall be positive integers. The letter \( k \) will denote the sparsity of a vector (see definition 9).

**Definition 1.** A subset \( R \subseteq S \) is said to be a proper subset of \( S \) if \( R \) is neither the empty set nor the whole set \( S \). If \( R \) is a proper subset, then we denote this by \( R \subset S \).

**Definition 2.** Given a set \( S \) and a subset \( R \subseteq S \), the complement of \( R \) (with respect to \( S \)) is denoted \( R^C \). Unless stated otherwise, the expression \( R \subseteq \{1, \ldots, N\} \) implies that the complement \( R^C \) is taken with respect to the set \( \{1, \ldots, N\} \), where \( N > 0 \) is some integer.

**Definition 3.** The cardinality [14, p. 24] of a set \( S \) is denoted \(|S|\).

**Definition 4.** The floor of a real number \( r \), denoted \([r] \), is defined as the largest integer \( q \) such that \( q \leq r \).

**Definition 5.** Given a vector space \( V \) over the field of real numbers \( \mathbb{R} \), a so-called norm is a function \( \|\cdot\|: V \to [0, \infty) \subset \mathbb{R} \) with the following properties: for any \( v, w \in V \) and \( \alpha \in \mathbb{R} \)

- \( \|v + w\| \leq \|v\| + \|w\| \)
- \( \|\alpha v\| = |\alpha| \|v\| \)
- \( \|v\| > 0 \in \mathbb{R} \iff v \neq 0 \in V \)
- \( \|v\| = 0 \in \mathbb{R} \iff v = 0 \in V \).

**Definition 6.** The \( \ell_p \)-norm of the vector space \( \mathbb{R}^n \) is defined as

\[
\|v\|_p = \begin{cases} 
(\sum_{i=1}^{n} |v_i|^p)^{\frac{1}{p}} & 1 \leq p < \infty \\
\max_i |v_i| & p = \infty
\end{cases}
\]

for any \( v \in \mathbb{R}^n \) whose \( i \)th component is \( v_i \).

Components of a vector will most often be denoted as the vector itself subscripted by an index (an integer between 1 and \( n \)). Any deviation from this convention is explicitly mentioned.

**Definition 7.** The inner product in \( \mathbb{R}^n \) is denoted

\[
\langle v, w \rangle = \sum_{i=1}^{n} v_i w_i .
\]

Recall the identity \( \langle v, v \rangle = \|v\|^2_2 \) for any \( v \in \mathbb{R}^n \).
Definition 8. The support of a vector $v \in \mathbb{R}^n$, denoted $\text{supp}(v)$, is the set of indices $i$ such that $v_i \neq 0$. Moreover, given a subset $S \subseteq \{1, \ldots, n\}$, the components $(v_S)_i$ of the vector $v_S$ are defined as

$$(v_S)_i = \begin{cases} v_i & i \in S \\ 0 & i \notin S \end{cases}.$$ 

The notation $\|v\|_0 = |\text{supp}(v)|$ is also used, but $\|\cdot\|_0$ is not a norm, since the second axiom of norms ($\|\alpha v\|_0 = |\alpha| \|v\|_0$) only holds for $\alpha = \pm 1$. Nonetheless, the notation $\|v\|_0$ is justified by the limit

$$\lim_{p \to 0^+} \|v\|^p_p = \lim_{p \to 0^+} \left( \sum_{i=1}^n |v_i|^p \right)^{\frac{1}{p}} = \lim_{p \to 0^+} \sum_{i=1}^n |v_i|^p = \sum_{v_i \neq 0} 1 = |\text{supp}(v)|.$$ 

Definition 9. The set $\Sigma_k \subseteq \mathbb{R}^n$ consists of all vectors $v \in \mathbb{R}^n$ that satisfy $\|v\|_0 \leq k \leq n$. These vectors are called $k$-sparse.

Note that $\Sigma_k$ is not closed under vector addition, but instead we have that $v \pm w \in \Sigma_{k+k'}$ whenever $v \in \Sigma_k$ and $w \in \Sigma_{k'}$ hold. Note also that $\Sigma_k \subset \Sigma_{k'}$ holds whenever $k' > k \geq 0$.

Definition 10. The rank of a matrix is the dimension of the space spanned by its column vectors [15, p. 488].

Definition 11. The spark of a matrix $A$, denoted $\text{spark}(A)$, is the smallest number of linearly dependent columns of $A$ [12, p. 17]. Stated differently,

$$\text{spark}(A) = \min_{v \neq 0} \|v\|_0 \quad \text{s.t.} \quad Av = 0.$$ 

For example, it is enough to state that $\text{spark}(A) \leq 3$ if there exist three columns of $A$ that are linearly dependent, in which case there is a 3-sparse vector in $\ker A$. Throughout the thesis, we want sparse vectors to stay away from the null space of the sensing matrix, so $\text{spark}(A)$ should be as high as possible.

The spark of a matrix does not exceed $m+1$ for an $m \times n$-matrix. This can be seen from the fact that any set of $m+1$ vectors in $\mathbb{R}^m$ are always linearly dependent. Thus, it is easy to see that for any $A \in \mathbb{R}^{m \times n}$ with at least as many columns as rows, $\text{spark}(A) \in [2, m+1]$ holds [12, p. 17]. If $m = n$ holds and all columns are linearly independent, then $\text{spark}(A) = m + 1$ is defined for the sake of consistency.

Definition 12. Given a nonempty subset $S \subseteq \{1, \ldots, n\}$ and a matrix $A \in \mathbb{R}^{m \times n}$, the matrix $A_S$ is the submatrix of $A$ consisting of all columns whose index is in $S$. The columns are in their respective order.

Definition 13. A matrix is called random Gaussian if all components are independent and identically distributed (iid) according to the normal distribution $\mathcal{N}(0, \sigma)$ for some variance $\sigma^2 > 0$. 

5
3 Retrieving vectors from sensing measurements

When speaking of solutions to (P0) or (P1), it is understood that the corresponding vector $y$ in either problem is that of a sensing measurement of a $k$-sparse vector, and that a solution of either problem is that of an attempt to retrieve this $k$-sparse vector. It will become apparent to the focused reader that sparse vectors are very special, in the regard that they are decompressible by solving (P0) or (P1).

The problem underlying the whole thesis is that $Ax = y \in \mathbb{R}^m$ is a sensing measurement of some unique (possibly unknown) vector $x \in \mathbb{R}^n$, and that we are trying to retrieve $x$ using only $A$ and $y$. Indeed, since $y = Ax$, $x$ is somewhere among the infinitely many solutions to the vastly underdetermined system

$$y = Az, \quad A \in \mathbb{R}^{m \times n}, \quad m \ll n.$$  

The symbol $\ll$ denotes a much less than relation. If we let $x \in \mathbb{R}^n$ be any vector, there is no hope for finding $x$ using only $A$ and $y$. But if $x$ is sparse, then it may be found among the solutions to (P0). As is stated, solutions to (P0) have minimal support, meaning they are as sparse as possible. It therefore makes sense to have (P0) as a default way of sparse recovery, since it clearly prioritizes sparsity in solutions. In fact, if we let a vector $x$ be sufficiently sparse, and let $Ax = y$, then a solution to (P0) will be unique, and the solution is guaranteed to be $x$. The following theorem establishes exactly how sparse $x$ needs to be, for a solution to be unique.

**Theorem 1.** For any $x \in \Sigma_k$ and $A \in \mathbb{R}^{m \times n}$, there is a one-to-one correspondence between $x$ and $y = Ax$ if and only if $2k < \text{spark}(A)$.

**Proof.** Note first that the product $Ac$ can be thought of as a linear combination of columns of $A$, with coefficients given by non-zero components of $c \in \mathbb{R}^n$. A set of columns of $A$ are linearly dependent if $Ac = 0$ for a corresponding $c \neq 0 \in \mathbb{R}^n$, in which case $\|c\|_0 \geq \text{spark}(A)$.

Now, suppose instead the correspondence is one-to-one while $2k \geq \text{spark}(A)$. Since $2k \geq \text{spark}(A)$ holds, a linear combination of columns of $A$ can be made nontrivial such that $Aw = 0$ for some non-zero $w \in \Sigma_{2k}$. Stated differently, $0 \neq w \in \ker A$. This vector can be split into two distinct $k$-sparse vectors $z$ and $z'$ (by choosing two disjoint supports of size $k$), ie. $w = z + z'$. We then have

$$A(z + z') = 0 \iff Az = A(-z').$$

But then $y = Az$ corresponds to two distinct vectors $z, -z' \in \Sigma_k$, which is a contradiction.

On the other hand, suppose the correspondence is not one-to-one while $2k < \text{spark}(A)$. Then $Ax = Az$ for two distinct $x, z \in \Sigma_k$. Especially we have $x - z \in \Sigma_{2k} \setminus \{0\}$. But since $2k < \text{spark}(A)$ holds, any set of $2k$ columns of $A$
are linearly independent, and thus \( A(x - z) = 0 \) implies that \( x - z = 0 \in \mathbb{R}^n \), which is yet another contradiction. Hence the theorem must hold.

When trying to solve (P0), a valid strategy is to go through all \( S \subset \{1, \ldots, n\} \) in the order of ascending cardinality (from \( |S| = 1 \) up to the largest \( |S| \) for which \( 2|S| < \text{spark}(A) \)), and check if any system

\[
A_S z = y, \quad A_S \in \mathbb{R}^{m \times |S|}, \quad z \in \mathbb{R}^{|S|}
\]

has a solution. As long as \( 2k < \text{spark}(A) \), Theorem 1 implies that there is only one such solution in \( \Sigma_k \) for a given \( y \), a solution whose retrieval is the essence of compressed sensing. One needs to find an \( S \) among

\[
\binom{n}{|S|} = \frac{n!}{(|S|)!(|n - |S|)!}
\]

other subsets of the same size. This binomial coefficient hints at an overwhelmingly large search space, making it extremely difficult to solve for large \( n \) and moderately large \( |S| \). Unless there exists an algorithm that can consistently, and efficiently ignore the vast majority of such subsets, solving (P0) is computationally intractable. But under some constraints on \( A \), (P0) can be replaced by (P1), which is a much more efficiently solvable problem [12, p. 38]. Efficient ways to solve (P1) already exist, and so far, several algorithms have been proposed for applications of compressed sensing, some of which rely on other properties of \( A \) not mentioned in this thesis [11, 13]. Especially, it can be shown that (P1) can be translated to a linear program (see Appendix section 5.1), which has been known since the 1950s [9].

### 3.1 The null space property

But under which constraints on \( A \) does (P1) give valid retrievals of compressively sensed sparse vectors? For a given matrix, this is not at all obvious, and it may even be hopeless to determine. Nonetheless, the matrix \( A \) could have a special kind of null space, which ensures that a solution to (P1) exists and is unique given that the measured vector is \( k \)-sparse, and as long as \( 2k < \text{spark}(A) \), this solution corresponds to that of (P0). If \( A \) has this kind of null space, then it has the property defined as follows.

**Definition 14.** A matrix \( A \in \mathbb{R}^{m \times n} \) is said to have the null space property of order \( k \) (NSP of order \( k \)), if for any \( v \in \ker A \setminus \{0\} \) and any set \( S \subseteq \{1, \ldots, n\} \) satisfying \( |S| \leq k \leq n \), the following inequality holds

\[
\|v_S\|_1 < \|v_{S^c}\|_1.
\]

Note that since the cardinality of \( S \subseteq \{1, \ldots, n\} \) can be chosen arbitrarily (as long as it does not exceed the maximal NSP order), NSP of order \( k \) directly implies NSP of order \( k - 1 \). By induction, NSP of order \( k \) entails NSP of all lower orders.
Note also that if $A$ has the NSP of order $k$, then $k$-sparse vectors are never in $\ker A$, since $v \in \Sigma_k$ always satisfies
\[
\|v_{\text{supp}(v)}\|_1 \geq \|v_{\text{supp}(v)^c}\|_1 = 0.
\]
Another purpose of the NSP is to have existence and uniqueness of solutions to (P1). If $y = Ax$, where $x$ is $k$-sparse, then a solution to (P1) (given this $y$) is guaranteed to be $x$ if solutions are unique. This is established by the following theorem.

**Theorem 2.** A vector $x \in \Sigma_k$ is the unique solution to (P1) given $y = Ax$, if and only if $A$ has the null space property of order $k$.

**Proof.** Given an arbitrary $v \in \ker A \setminus \{0\}$ and $S$ such that $|S| \leq k$, $v_S$ is necessarily $k$-sparse. Assume first that $x \in \Sigma_k$ is the unique solution to (P1) given $y = Ax$. Then $v_S$ is the unique solution to (P1) given $y = Av_S$. (Think of this as an assignment to the corresponding $y$ in the problem (P1), so that if $y = Av_S$ is given, (P1) will be solved with this $A$ and $y$.) Since $0 = Av = A(v_S + v_{S^c})$ implies $Av_S = A(-v_{S^c})$, $\|v_S\|_1 < \|v_{S^c}\|_1$ must hold by assumption. Since $v$ is chosen arbitrarily, this inequality holds for any $v \in \ker A \setminus \{0\}$ and thus $A$ has the NSP of order $k$.

Now assume on the other hand that $A$ has the NSP of order $k$. We want to show that $x \in \Sigma_k$ is the unique solution to (P1) given $y = Ax$. So suppose $x$ is $k$-sparse with support $S$. Then any other vector $w \neq x$ satisfying $Ax = Aw = y$ is such that $(x - w) \in \ker A \setminus \{0\}$, because $0 = Ax - Aw = A(x - w)$. We also have that $(x - w)_S = x - w_S$, and $x_{S^c} = 0 \in \mathbb{R}^n$, because of $S$ being the support of $x$. Then,
\[
\|x\|_1 = \|x - w_S + w_S\|_1
\]
\[
\leq \|x - w_S\|_1 + \|w_S\|_1
\]
\[
= \|(x - w)_S\|_1 + \|w_S\|_1
\]
\[
< \|(x - w)_{S^c}\|_1 + \|w_S\|_1 \quad \text{(NSP)}
\]
\[
= \|(-w)_{S^c}\|_1 + \|w_S\|_1
\]
\[
= \|w\|_1
\]

i.e. $\|x\|_1 < \|w\|_1$ for any $w$ satisfying $Ax = Aw$. This implies that $x$ is strictly the smallest vector in $\ell_1$ satisfying $Ax = y$, and hence $x$ is the unique solution to (P1).
3.2 The restricted isometry property

The null space property of order \( k \) is both necessary and sufficient for sparse recovery by (P1), but one might wonder if there are any matrices that have the null space property of order \( k \) at all. Verifying NSP for a given matrix and order is nontrivial, because it suffers from the same drawback as methods for solving (P0). To see this, observe that verification of the inequality

\[
\|v_S\|_1 < \|v_{S^c}\|_1
\]

may (in the worst case) force us to test all subsets \( S \subseteq \{1, \ldots, n\} \) of at most a given cardinality \( k \). The amount of such subsets is \( \binom{n}{|S|} \) for a fixed cardinality \( |S| \), which is massive for moderately large parameters. Candès and Tao presented in [7] an alternative property for matrices that can be sufficient for the NSP to be present. This alternative property, which relies on an approximately length-preserving behavior of a matrix when operating on vectors in \( \Sigma_k \), is defined as follows.

**Definition 15.** A matrix \( A \in \mathbb{R}^{m \times n} \) is said to have the **restricted isometry property of order \( k \)** (RIP of order \( k \)) if there exists a minimal constant \( \delta_k \in [0, 1) \) such that for any \( x \in \Sigma_k \),

\[
(1 - \delta_k) \|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta_k) \|x\|_2^2 .
\]

Unless stated otherwise, \( \delta_k \) denotes this smallest constant for a given matrix.

Note especially that if \( k' > k > 0 \), then \( \delta_{k'} \geq \delta_k \geq 0 \), which must hold because \( \Sigma_k \subset \Sigma_{k'} \). This also means that \( A \) has the RIP of order \( k - 1 \) if it has the RIP of order \( k \). By induction, RIP of order \( k \) entails RIP of all lower orders. Moreover, \( k \) must be less than \( \text{spark}(A) \), since otherwise some non-zero \( x \in \Sigma_k \) may be in \( \ker A \), in which case

\[
0 < (1 - \delta_k) \|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta_k) \|x\|_2^2
\]

does not hold. Thus, \( k < \text{spark}(A) \) is necessary for RIP of order \( k \). The next theorem shows a remarkable connection between the restricted isometry property and the null space property.

**Lemma 1.** Given two vectors \( v, w \in \mathbb{R}^n \), the vector \( z \) whose \( i \)-th component is \( z_i = v_i w_i \) satisfies the inequality

\[
\|z\|_1 \leq \|v\|_2 \|w\|_2 .
\]

**Proof.** Recall the Cauchy-Schwarz inequality, which states

\[
|\langle v, w \rangle| \leq \|v\|_2 \|w\|_2 , \quad v, w \in \mathbb{R}^n .
\]

Since this holds for arbitrary \( v, w \in \mathbb{R}^n \), it should also hold for the vectors \( v', w' \) whose components are

\[
v'_i = |v_i| , \quad w'_i = |w_i| , \quad i = 1, \ldots n .
\]

The inner product between \( v' \) and \( w' \) gives
\[ \langle v', w' \rangle = \sum_{i=1}^{n} |v_i| |w_i| = \sum_{i=1}^{n} |v_i w_i| = \sum_{i=1}^{n} |z_i| = \|z\|_1. \]

Observe also that \( \|v\|_2 = \|v'\|_2 \) and \( \|w\|_2 = \|w'\|_2 \) (by definition), from which the lemma statement now follows.

**Lemma 2.** For any two vectors \( v, w \in \mathbb{R}^n \), the following equality holds

\[ \langle v, w \rangle = \frac{1}{4} \left( \|v + w\|_2^2 - \|v - w\|_2^2 \right). \]

**Proof.** With the definition of the inner product, we have the chain of equalities

\[
\|v + w\|_2^2 - \|v - w\|_2^2 = \sum_{i=1}^{n} (v_i + w_i)^2 - \sum_{i=1}^{n} (v_i - w_i)^2
\]

\[
= \sum_{i=1}^{n} (v_i^2 + 2v_i w_i + w_i^2) - \sum_{i=1}^{n} (v_i^2 - 2v_i w_i + w_i^2)
\]

\[
= \|v\|_2^2 + \left( \sum_{i=1}^{n} 2v_i w_i \right) + \|w\|_2^2 - \left( \|v\|_2^2 - \left( \sum_{i=1}^{n} 2v_i w_i \right) + \|w\|_2^2 \right)
\]

\[
= \left( \sum_{i=1}^{n} 2v_i w_i \right) + \left( \sum_{i=1}^{n} 2v_i w_i \right)
\]

\[
= \sum_{i=1}^{n} 4v_i w_i = 4 \langle v, w \rangle.
\]

Division by 4 all throughout, proves the lemma statement.

**Theorem 3.** Let \( k \) be such that \( 1 \leq k \leq \lfloor n/2 \rfloor \). If \( A \in \mathbb{R}^{m \times n} \) has the RIP of order \( 2k \) with \( \delta_{2k} < 1/3 \), then \( A \) has the NSP of order \( k \).

**Proof.** Let two vectors \( z, w \in \Sigma_k \) with \( \|z\|_2 = \|w\|_2 = 1 \) have mutually disjoint supports. Then \( \|z \pm w\|_2^2 = 2 \) necessarily holds. We assume that \( A \) has the RIP of order \( 2k \) (and hence of order \( k \) as well), which means

\[
2(1 - \delta_{2k}) \leq \|A(z \pm w)\|_2^2 \leq 2(1 + \delta_{2k}). \tag{1}
\]

A sign change of the above inequality gives

\[
-2(1 + \delta_{2k}) \leq -\|A(z \mp w)\|_2^2 \leq -2(1 - \delta_{2k}). \tag{2}
\]

Summation of each side of (1) and (2) gives

\[
-4\delta_{2k} \leq \|A z \pm Aw\|_2^2 - \|A z \mp Aw\|_2^2 \leq 4\delta_{2k}
\]

10
from which we can see, together with Lemma 2, that

\[ |\langle Az, A(\pm w) \rangle| = \frac{1}{4} \left\| Az \pm Aw \right\|^2 - \left\| Az \mp Aw \right\|^2 \leq \delta_2 k. \]

To ease notation, we drop the ± and ⊕ signs. Letting \( ||z||_2 > 0 \) and \( ||w||_2 > 0 \) be something other than 1 gives

\[ |\langle Az, Aw \rangle| \leq \delta_2 k ||z||_2 ||w||_2 \]  

as long as \( z \) and \( w \) are \( k \)-sparse with disjoint supports, while \( A \) has the RIP of order \( 2k \).

Now choose an arbitrary \( v \in \ker A \setminus \{0\} \). Let \( S_1 \) contain the indices of the \( k \) largest components of \( v \) in absolute value (if such a set is not unique, then simply choose one of them). Then, define other sets \( S_i, i \geq 2 \) such that \( v_{S_i} \) contains the \( k \) largest components of \( v_{R_i} \) in absolute value, where \( R_i \) is defined as

\[ R_i = \bigcup_{j=1}^{i-1} S_j \]

There will be a total of \( M \) such sets, and they satisfy

\[ S_i \cap S_j = \emptyset \iff i \neq j \]

and

\[ \bigcup_{i=1}^{M} S_i = \{1, \ldots, n\} \]

This can be done, because subscripting \( v \) by \( R_i \) eliminates \( R_i \) from the process of finding largest components of \( v_{R_i} \) (in absolute value). The exact value of \( M \) is not important (in fact, since \( k \leq n/2 \) holds, \( M \) is at least 2), but we shall keep in mind some properties that these sets induce.

First of all, any non-zero component of \( v_{S_i} \) is greater than or equal to any component of \( v_{S_{i+1}} \) (in absolute value). Also, since the sets \( S_i \) are mutually disjoint, the previous arguments above can be used. By letting them be mutually disjoint, we have

\[ 0 = Av = A \left( \sum_{i=1}^{M} v_{S_i} \right) \]

\[ \iff Av_{S_i} = \sum_{i=2}^{M} A(-v_{S_i}) \]

\[ \implies \left\| Av_{S_i} \right\|^2_2 = \langle Av_{S_i}, Av_{S_i} \rangle \]
\[
\left\langle Av_{S_1}, \sum_{i=2}^{M} A(-v_{s_i}) \right\rangle.
\]

From the RIP of order \( k \), we have

\[
(1 - \delta_k) \|v_{S_1}\|_2^2 \leq \|Av_{S_1}\|_2^2 \implies \|v_{S_1}\|_2^2 \leq \frac{1}{(1 - \delta_k)} \|Av_{S_1}\|_2^2
\]

\[
= \frac{1}{(1 - \delta_k)} \left\langle Av_{S_1}, \sum_{i=2}^{M} A(-v_{s_i}) \right\rangle
\]

\[
= \frac{1}{(1 - \delta_k)} \sum_{i=2}^{M} \langle Av_{S_1}, A(-v_{s_i}) \rangle \tag{4}
\]

\[
\leq \frac{1}{(1 - \delta_k)} \sum_{i=2}^{M} |\langle Av_{S_1}, A(-v_{s_i}) \rangle|\]

\[
= \frac{1}{(1 - \delta_k)} \sum_{i=2}^{M} |\langle Av_{S_1}, A(v_{s_i}) \rangle|
\]

\[
\leq \frac{1}{(1 - \delta_k)} \sum_{i=2}^{M} \delta_{2k} \|v_{S_i}\|_2 \|v_{S_i}\|_2 . \tag{5}
\]

Equation (4) comes from linearity of the inner product, and equation (5) comes from the inequality in (3). Dividing by \( \|v_{S_i}\|_2 > 0 \) all throughout, we get

\[
\|v_{S_i}\|_2 \leq \frac{\delta_{2k}}{1 - \delta_k} \sum_{i=2}^{M} \|v_{S_i}\|_2 . \tag{6}
\]

The sum above satisfies the inequality

\[
\sum_{i=2}^{M} \|v_{S_i}\|_2 \leq \sum_{i=2}^{M} \sqrt{k} \|v_{S_i}\|_\infty
\]

\[
= \sum_{i=2}^{M} \sqrt{k} \left( \max_{j \in S_{i}} |v_{j}| \right) .
\]

The ordering (in absolute value) implicitly defined by the sets \( S_i \) gives

\[
\max_{j \in S_i} |v_{j}| \leq \min_{j \in S_{i-1}} |v_{j}|
\]

which when applied above gives

\[
\sum_{i=2}^{M} \sqrt{k} \left( \max_{j \in S_i} |v_{j}| \right) \leq \sum_{i=2}^{M} \sqrt{k} \left( \min_{j \in S_{i-1}} |v_{j}| \right) . \tag{7}
\]
Observe that each set $S_i$ except for the last one has cardinality $k$. This means that for all $S_i$ except the last one, the minimum value of $|v_j| (j \in S_i)$ satisfies the inequality
\[ \min_{j \in S_i} |v_j| \leq \frac{1}{k} \sum_{j \in S_i} |v_j| . \]

Now since $i$ goes from 2 to $M$, $S_{i-1}$ goes from $S_1$ to $S_{M-1}$. With this in mind, we get the following inequality starting from equation (7)
\[ \sum_{i=2}^{M} \sqrt{k} \left( \min_{j \in S_{i-1}} |v_j| \right) \leq \sum_{i=2}^{M} \sqrt{k} \left( \frac{1}{k} \sum_{j \in S_{i-1}} |v_j| \right) = \sum_{i=1}^{M-1} \frac{1}{\sqrt{k}} \left( \sum_{j \in S_i} |v_j| \right) \leq \sum_{i=1}^{M} \frac{1}{\sqrt{k}} \left( \sum_{j \in S_i} |v_j| \right) = \frac{1}{\sqrt{k}} \| v \|_1 . \]

From equation (6), this implies
\[ \| v_{S_1} \|_2 \leq \frac{\delta_{2k}}{(1 - \delta_k) \sqrt{k}} \| v \|_1 \]
or equivalently
\[ \sqrt{k} \| v_{S_1} \|_2 \leq \frac{\delta_{2k}}{(1 - \delta_k)} \| v \|_1 . \]

Define the vector $e \in \mathbb{R}^n$ with components
\[ e_i = \begin{cases} 1 & i \in S_1 \\ 0 & i \notin S_1 \end{cases} \]
for which Lemma 1 gives the first inequality below
\[ \| v_{S_1} \|_1 \leq \| e \|_2 \| v_{S_1} \|_2 = \sqrt{k} \| v_{S_1} \|_2 \leq \frac{\delta_{2k}}{1 - \delta_k} \| v \|_1 < \frac{1}{2} \| v \|_1 . \]

The last inequality above holds because $0 \leq \delta_k \leq \delta_{2k} < \frac{1}{3}$ gives
\[ \frac{\delta_{2k}}{1 - \delta_k} < \frac{1}{2} \iff 2\delta_{2k} < 1 - \delta_k \iff 2\delta_{2k} + \delta_k < \frac{2}{3} + \frac{1}{3} = 1 . \]

Observe finally that the inequality
\[ \| v_{S_1} \|_1 < \frac{1}{2} \| v \|_1 \]
is equivalent to

13
\[2 \|v_{S_1}\|_1 < \|v\|_1\]
\[\iff \|v_{S_1}\|_1 + \|v_{S_1}\|_1 < \|v\|_1\]
\[\iff \|v_{S_1}\|_1 < \left\| v_{(S_1)\complement} \right\|_1. \tag{8}\]

Note the similarity of (8) to the inequality given in the definition of NSP. However, NSP of order \(k\) requires that \(S_1\) is chosen arbitrarily, which it is obviously not. But this is fine, because by definition of \(S_1\), we have the following two inequalities

\[\|v_{S_1}\|_1 \geq \|v_T\|_1\]
\[\left\| v_{(S_1)\complement} \right\|_1 \leq \|v_T\|_1\]

for any other \(T\) of cardinality \(k\). With inequality (8), this gives

\[\|v_T\|_1 < \left\| v_{T\complement} \right\|_1\]

for any \(T\) of cardinality \(k\). Since \(k \leq \lfloor n/2 \rfloor\) and \(v \in \ker A \setminus \{0\}\) were given, this holds for any such \(v, T\) and \(k\). Hence, \(A\) has the NSP of order \(k\).

\[\square\]

There are other RIP conditions on \(A\) that also guarantee \(A\) to have the NSP of order \(k\), for example that \(\delta_{2k} < \sqrt{2} - 1\) should hold [12, p. 22], which is less strict than \(\delta_{2k} < 1/3\) since \(\sqrt{2} - 1 > 1/3\). An RIP constant close to zero essentially guarantees that no matter what sparse vector is chosen, the matrix roughly preserves the vector’s distance. Clearly, a linear space \(V \subset \mathbb{R}^n \setminus \{0\}\) for which a linear transformation preserves distance, is far enough from the kernel (which is desirable).

### 3.3 Finding a good sensing matrix

So far, we have presented two properties of \(A\) which allow us to replace (P0) with (P1). But both the NSP and RIP of a given order is very hard to determine for a given matrix, so at first glance their introduction (especially that of RIP) seems impractical. The difficulty of verifying RIP is essentially the same as that of NSP (possibly forcing us to test for all subsets of \(\{1, \ldots, n\}\) of at most some cardinality). This is where the next theorem comes in. This next theorem implies that a random Gaussian matrix can have the RIP of order \(k\) with overwhelming probability. If \(A \in \mathbb{R}^{m \times n}\) is of the form

\[
\frac{1}{\sqrt{m}} \begin{bmatrix}
\omega_{11} & \cdots & \omega_{1n} \\
\vdots & \ddots & \vdots \\
\omega_{m1} & \cdots & \omega_{mn}
\end{bmatrix}
\]
where $\omega_{ij}$ are independent and identically distributed standard normal random variables, then for some $k \leq m \leq n$, the matrix $A$ has the RIP of order $k$ with extremely high probability. Thus, the problem of finding an appropriate sensing matrix can be made probabilistic, but practically effortless. Vershynin et. al. proved in [13, p. 210-255] an asymptotic bound for $k, m, n$ that ensures that $A$ has the RIP of order $k$ with a least probability. This bound is given by the following theorem, stated without proof.

**Theorem 4.** Let $k \leq m \leq n$ and $\varepsilon \in (0, 1]$ be fixed, and let $A \in \mathbb{R}^{m \times n}$ be a random Gaussian matrix. If $m$ satisfies

$$m \geq C\varepsilon^{-2}k \log(en/k)$$

then

$$\mathbb{P} [\delta_k \leq \varepsilon] \geq 1 - 2e^{-c\varepsilon^2m}$$

for some positive constants $c$ and $C$ independent of $k, m, n$.

Looking solely at Theorem 3, the RIP of order $2k$ may be seen as a convoluted way of having the NSP of half the order, but RIP clearly serves a purpose here. One can view the RIP and Theorem 2–4 as a bridge between having requirements on a sensing matrix for sparse recovery, and actually finding a matrix with these requirements. Quite amazingly, a good candidate for a sensing matrix can be chosen completely at random. While the null space property is a very straightforward requirement on $A$, the practicality of actually finding an appropriate $A$ is made clear by these theorems together with the restricted isometry property.

Another nice feature of random Gaussian matrices is that their spark is always maximal. Then by Theorem 1, any $x \in \Sigma_k$ satisfying $2k \leq m$ is retrievable from (P0) given $y = Ax$. This maximal spark property follows from the next theorem.

**Theorem 5.** Let $\Psi_i \in \mathbb{R}^m, i \in \{1, \ldots, m\}$ be (column) vectors whose components are iid. standard normal random variables. Then these vectors are linearly dependent with probability zero.

**Proof.** Let $\Psi \in \mathbb{R}^{m \times m}$ be the matrix containing the columns vectors $\Psi_i, i \in \{1, \ldots, m\}$ with components $\Psi_{j,i}, j \in \{1, \ldots, m\}$. We are interested in determining whether there is a subset $C \subset \mathbb{R}^m \setminus \{0\}$ such that for each $c \in C$

$$\mathbb{P} [\Psi c = 0] > 0.$$

If such a subset $C$ exists, then the vectors $\Psi_i$ are linearly dependent with non-zero probability. The theorem statement holds if there exists no such subset.

Assume by contradiction that such a set $C$ exists. Then for some $c \in C$ with at least one non-zero component among $c_i, i \in \{1, \ldots, m\}$, we have
\[ \Psi c = \sum_{i=1}^{m} c_i \Psi i = 0 \]

with non-zero probability. In terms of probability theory, \( \Psi c = 0 \) is an event, which is equivalent to the limit of

\[ 0 \leq \|\Psi c\|_{\infty} \leq \varepsilon \]

as \( \varepsilon > 0 \) tends to zero. This other event above is the event in which \( \Psi c \) falls within the arbitrarily small hypercube \([-\varepsilon, \varepsilon]^m\). The theorem is proved if the probability of such an event tends to zero with \( \varepsilon \), regardless of \( c \in C \). Now assume instead that for some \( c \in C \), it does not tend to zero with \( \varepsilon \), ie.

\[
\lim_{\varepsilon \to 0^+} \mathbb{P} \left[ -\varepsilon / \sigma \leq \omega \leq \varepsilon / \sigma \right] = \lim_{\varepsilon \to 0^+} \int_{-\varepsilon / \sigma}^{\varepsilon / \sigma} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt > 0.
\]

But this is clearly false given the boundedness and continuity of \( t \mapsto \frac{1}{\sqrt{2\pi}} e^{-t^2/2} \). Hence, the non-emptiness of \( C \) is contradictory.
Theorem 5 also implies that any random Gaussian matrix has full rank. Since \( m \leq n \), the rank of \( A \) is \( m \) and the spark is \( m + 1 \), given that \( A \in \mathbb{R}^{m \times n} \) is random Gaussian. With this fact and the previously stated theorems in mind, we can conclude below that (P0) and (P1) are equivalent with a certain least probability.

**Corollary 1.** Let \( 2k \leq m \) hold, let \( A \in \mathbb{R}^{m \times n} \) be a random Gaussian matrix, and let \( \varepsilon < 1/3 \) be fixed. Let also \( y \) in the problems (P0) and (P1) be that of a sensing measurement of a \( k \)-sparse vector (ie. \( y = Ax \) for some \( x \in \Sigma_k \)). If

\[
m \geq C'\varepsilon^{-2}k \log(en/(2k))
\]

then (P0) and (P1) have the same solutions with probability at least \( 1 - 2e^{-c\varepsilon^2m} \), for some constants \( c \) and \( C' \) independent of \( k, m, n \).

**Proof.** Since \( \varepsilon < 1/3 \), Theorem 3 and 4 imply that \( A \) has the NSP of order \( k \) with probability at least \( 1 - 2e^{-c\varepsilon^2m} \) (recall that the NSP of order \( k \) follows if \( \delta_{2k} < 1/3 \)). Theorem 2 then imply that \( k \)-sparse vectors are retrievable by solving (P1). Meanwhile, since \( 2k \leq m < \text{spark}(A) \) holds according to Theorem 5, solving (P0) retrieves \( k \)-sparse vectors (regardless of the NSP of \( A \)), and thus (P0) and (P1) have the same solutions with the given least probability. \( \square \)

### 3.4 Conclusion

Quite remarkably, with the given least probability in Corollary 1, it is actually possible to retrieve vectors in \( \mathbb{R}^n \) with a sparsity as much as half the number of components of the sensing measurement. If \( m \) is also much smaller than \( n \), this entails a massive decrease in storage space for sensing measurements. Although, this is under the assumption that we are lucky to have generated a good random Gaussian sensing matrix.

As can be seen from the lower bound on \( m \) given by this corollary, an increasing \( n \) means an increasing right hand side, which implies an increasing lower bound for \( m \). If this lower bound is to be fixed for increasing \( n \), then \( \varepsilon \) must be increased above and beyond 1/3, and thus the probability that \( A \) is a good sensing matrix becomes less obvious. This shows a tradeoff between how sure we can be that a random sensing matrix works, and how much we want to compress incoming data from \( \mathbb{R}^n \). Such a tradeoff becomes clear with a concrete example where \( m = 4 \) and \( n \gg 4 \) is extremely large. In this example, a 2-sparse vector in \( \mathbb{R}^n \) could be measured with only four real numbers. Intuition tells us that this should not be very feasible if \( n \) is extremely large, since a significant amount of information from \( \mathbb{R}^n \) would have to be packed into four real numbers.
4 Compressed sensing for error correction

We have established some conditions on $A \in \mathbb{R}^{m \times n}$ which are sufficient for feasible recovery of arbitrary $k$-sparse vectors from sensing measurements. The optimal scenario is where $2k \leq m < \text{spark}(A)$ while $A$ has the NSP of order $k$, since then the correspondence between $x \in \Sigma_k$ and $y = Ax \in \mathbb{R}^m$ is one-to-one, and (P0) can be replaced with (P1) for sparse recovery. A quite remarkable application of compressed sensing is for error correction, where one can single out some arbitrary $k$-sparse noise with a sensing measurement and from there on reconstruct the actual noise (see the problem description below for details on this).

Successful recovery of $x \in \Sigma_k$ below a threshold for $k$ is quite a remarkable guarantee to have, but as is apparent from the nature of choosing the sensing matrix randomly, we do not actually know how large $k$ can be for a given sensing matrix. And even if we know this, it could occur that sparsity of noise goes beyond the order of NSP of $A$. Say, for example, that the $k$-sparse vectors in question are those of some unwanted noise $e \in \Sigma_k$ whose non-zero components are normally distributed with mean zero. If $k$ is too large, then successful mitigation of errors can not be guaranteed. But does vector reconstruction drastically fail if $k$ is too large?

As we will see, it turns out that even if $k$ is too large for the given sensing matrix $A$ (in the sense of guaranteed reconstruction), error correction may still be successful with high probability, provided that the matrix and vector dimensions are relatively small. Section 4 regards cases where $k$ goes beyond recovery guarantees offered by CS. Moreover, we will compare such a success rate for various parameters, to see if there is any benefit to choosing particular vector dimensions. In any case, we will assume that $k$-sparse vectors have Gaussian components. Such an assumption is not contrived, because thermal noise in nature is often modeled as Gaussian [17, p. 71-72].

4.1 Problem description

Bear in mind that the use of the letters $x$ and $y$ will be a bit different here, compared to previous sections. The rest of the thesis concerns itself with the problem of sending a vector of data $x \in \mathbb{R}^p$ through a noisy channel. The vector $x$ does not need to be sparse. This is done by first encoding $x$ into a so-called code vector $y \in \mathbb{R}^n$, which is done by multiplying $x$ by a special coding matrix $E$. Again, $y$ does not need to be sparse either. The code vector

$$y = Ex$$

is sent through the noisy channel, where both ends of the channel have beforehand agreed upon two matrices $E \in \mathbb{R}^{n \times p}$ and $A \in \mathbb{R}^{m \times n}$. We will assume that $p < n$. The matrix $E$ has full rank, and $y$ represents $x$ in a $p$-dimensional subset of $\mathbb{R}^n$ (roughly speaking, $y$ is a redundant form of $x$). The matrix $A$ is a
sensing matrix for $k$-sparse vectors, and it is such that the columns of $E$ are in $\ker A$. Now, suppose that the other end of the channel receives the corrupted code vector

$$y' = y + e$$

where $e$ is $k$-sparse noise whose non-zero components are Gaussian, and $\text{supp}(e)$ is completely arbitrary. The receiver of $y'$ is then interested in correcting the given code vector to obtain $y$. Once the receiver has $y$, then $x$ (the data of interest) can be retrieved since $Ex = y$ is (by definition) an overdetermined system with a unique solution. The first step in the error correction procedure is computing the product

$$Ay' = A(y + e) = A(Ex + e) = (AE)x + Ae = 0x + Ae = Ae = f \in \mathbb{R}^m.$$ 

The sensing matrix $A$ simultaneously annihilates the code vector $y$ and does a sensing measurement of $e$. Note that at this stage, the vector $y$ is still unknown to the receiver, but if $e$ can be recovered from $f$, then $y$ can be recovered simply by computing $y' - e$. Indeed, if we assume that $A$ has the null space property of order $k < \text{spark}(A)/2$ while $e \in \Sigma_k$, then it is possible to reconstruct $e$ from

$$\min_{z \in \mathbb{R}^n} \|z\|_1 \quad \text{s.t.} \quad Az = f$$

which is just (P1) for when Gaussian noise is measured. The theory of compressed sensing guarantees that $e$ can be reconstructed, given $f = Ae$, $e \in \Sigma_k$, $2k < \text{spark}(A)$ and that $A$ has the NSP of order $k$. The problem of reconstructing the noise $e$ is the crucial step of this error correction procedure. Since we practically do not know whether $A$ has the NSP of order $k$, our main interest is to get an idea of how crucial the knowledge of NSP is for sparse vector reconstruction.

### 4.2 Choice of sensing matrix

The sensing matrix $A \in \mathbb{R}^{m \times n}$ is random Gaussian as before. Construction of the coding matrix $E$ would be done by taking elements from $\ker A$ such that $E$ has full rank, but $E$ is not even considered in the numerical simulations. The choice of parameters $(m, n)$ is illustrated by the next figure, and is exclusive to the problem of error correction. Here, we choose $A$ with $n$ columns and $n - p$ rows, and when $p$ is fixed, we are only free to choose $n > p$. This is very different from choosing the rows and columns independently, so bear in mind that results from numerical simulations presented here may not translate well to other applications of compressed sensing.
Figure 1: Illustration of how the dimensions of $E \in \mathbb{R}^{n \times p}$, $A \in \mathbb{R}^{m \times n}$ and the relevant vectors may vary. The number of columns of $A$ is chosen to be $m = n - p$. This choice of $m$ is explained below.

Since we are dealing with a particular application where the amount $p$ of data (to be transferred) is fixed, the most obvious degree of freedom is $n$ (which is chosen according to how redundant we want the code to be). Since all $p$ columns of the full rank matrix $E$ are to be annihilated, $\dim(\ker A) \geq p$ should hold. Furthermore, Theorem 5 implies that the rank of $A$ is the number of rows (since $A$ is random Gaussian). Recognizing that too high of a nullity of $A$ is unnecessary, we can settle for a maximum of $n - p$ rows, in which case $\dim(\ker A) = p$ is given by the rank-nullity formula for matrices [1, p. 239][18, p. 48-49].

Of course, one may also be free to choose $p$, but the relationship between the number $n$ of rows and the number $n - p$ of columns is not like other applications of CS where the number of rows is variable and the number of columns is given.

Figure 2: Choice of rows and columns of the sensing matrix $A$ is done according to the leftmost picture, where $p$ is chosen first and we are only free to choose either $n$ or $m$ ($= n - p$). This is very different from other possible applications of compressed sensing, where the number of rows or columns are fixed (as illustrated by the other two pictures).

We will only consider the simple cases where $n$ is either twice as large as $p$, or where $n \in [1.5p, 2.5p]$.
4.3 Numerical simulations

What remains to answer is what happens when $k$ becomes too large. Section 4.3 accounts for this, which is beyond the guaranteed reconstruction that has been dealt with so far. We present some numerical simulations of the problem of reconstructing a noise vector $e \in \Sigma_k$. Reconstruction is done by solving (P1) as a corresponding linear program (see Appendix section 5.1 for details on this). The simulations in question are set up as follows.

1. Choose some set of values for $k, m, n, p$ for which $k \leq m < n$, $m = n - p$ and $0 < p < n$ hold.
2. Generate a random Gaussian sensing matrix $A \in \mathbb{R}^{m \times n}$.
3. Measure the success rate of reconstructing various noise vectors $e \in \Sigma_k$ from $f = Ae$. The support of $e$ is arbitrary and the non-zero components of $e$ are distributed according to $\mathcal{N}(0, 1)$. These vectors mimic unwanted noise.

For each sensing matrix $A \in \mathbb{R}^{(n-p) \times n}$ and sparsity level $k$, 1000 noise vectors $e \in \Sigma_k$ were tested for sensing and reconstruction. The data of interest is the success rate of reconstructing $e$. Success and failure of reconstructing $e \in \Sigma_k$ shall be taken as a Bernoulli random variable $X_i$, having the value 1 with unknown probability $q$ in case of success and 0 with unknown probability $1 - q$ in case of failure. By the nature of floating point precision in computers, a reconstruction of $e$ is deemed successful if its $\ell_2$-distance from $e$ is at most $10^{-12}$. This is accurate enough for our purposes. Other thresholds for success were tested, but they gave mostly the same results. In any case, we are interested in estimating this unknown probability $q$ by the sum

$$\hat{q} = \frac{1}{1000} \sum_{i=1}^{1000} X_i$$

since this estimates how likely it is for sparse reconstruction to succeed. We can say with at least 95% confidence that the actual value of $q$ is within the interval $[\hat{q} - 0.031, \hat{q} + 0.031]$ (see Appendix section 5.2 for details on this). This interval is illustrated with error bars in graphs.

4.3.1 Performance degradation with increasing noise density

The term noise density here is essentially the same as sparsity of $e \in \Sigma_k$. Informally speaking, $e$ is sparse if $k$ or $k/n$ is low, and $e$ is dense if $k$ or $k/n$ is high. Figure 3 shows one and the same set of $k$-sparse noise vectors being tested against four different sensing matrices. As can be seen from the significant overlap of these graphs, it is safe to assume that one realization of $A$ is good enough.
Figure 3: Estimated success rate of reconstructing $k$-sparse vectors. Four randomly chosen Gaussian sensing matrices were tested against the same data, as illustrated by four graphs.

Results presented in figure 4 show (like in figure 3) how the performance of reconstructing $e \in \Sigma_k$ degrades as the percentage of non-zero components of $e$ increases. Nine values of $n$ were chosen uniformly spaced in the interval $[1.5p, 2.5p]$. Darker shades correspond to lower values of $n$, so the darkest graph is for $n = 1.5p$. What is interesting here is seeing how big the fraction of corrupted components can be before successful error correction becomes unlikely. Most of the nine graphs have the value 1 around $k/n \approx 0.05$, which means that error correction is very likely to be successful when 5% of components are corrupted. At $k/n \approx 0.4$, all nine instances of $n$ gave poor results, which means error correction is practically impossible when 40% of components are corrupted.
Figure 4: Estimated success rate of reconstructing $k$-sparse vectors. Nine different values of $n$ in the interval $[1.5p, 2.5p]$ were tested across several values of $k/n$. Darker graph means smaller $n$. Each $n$ has a corresponding sensing matrix.

There is a clear pattern here for the darker graphs, namely that a given (fixed) noise density becomes more likely to be corrected as $n$ (and $m = n - p$) increases. This is evidence for the fact that $n$ (and $m$) should be as high as possible, if a higher likelihood of error correction is desired.
### 4.3.2 Choice of dimensions given fixed noise density

Suppose that $k$ is a given percentage of $n$, and that this percentage is too large for guaranteed reconstruction. Then, accepting some probability of success instead, one might wonder whether it is better to choose a large or small value for $p$. Figure 5 shows results from simulations aiming to make this clear.

![Figure 5: Estimated success rate of reconstructing $k$-sparse vectors. Four different sparsity percentages $k/n$ from the set $\{0.0625, 0.125, 0.1875, 0.25\}$ were tested across several values of $p$.](image)

It seems to be better to choose low values of $p$, if one desires high likelihood of error correction. This is taken from the downward trend that all four graphs exhibit for the given noise densities between 6.25 % and 25 %. To illustrate this point more clearly, two values of $p$ from the set $\{16, 64\}$ were further investigated. Figure 6 shows a comparison of the two instances $p = 16$ and $p = 64$. 

---

24
4.4 Conclusion

It is important to state the limitations of these numerical simulations. Many more values of $n$ and $p$ can be tested, and they might show slightly different results than what is presented here. This should be seen as an exploration of CS for error correction, where we compare various instances of relatively small matrix and vector dimensions. The issue of numerical precision and stability has mostly been ignored, and a further assumption regarding this issue is that the underlying routine for solving (P1) is well behaved. Thus, the results presented are subject to the same flaws as the Matlab routine for solving linear programs. Regarding the choice $(m, n) = (p, 2p)$ of parameters, we suspect that it may be the most common choice, given the simple nature of twice as large.
A higher value of $n/p$ seems to yield a higher success rate of reconstructing sparse vectors. For small values of $n/p$, figure 4 shows strong evidence for this. Figure 5 and 6 show evidence for a quite remarkable hypothesis, namely that dense noise for large $p$ is less likely to be successfully corrected than for equally dense noise for small $p$. Of course, this goes beyond what compressed sensing is about. Most of section 4 is about extreme levels of sparsity where compressed sensing falls apart. Still, a reasonable conclusion from these simulations is that CS falls apart rather smoothly if the sparse signals in question are Gaussian (which is rather realistic in the case of noise in nature).

But a larger $p$ is not necessarily bad. Revisiting Corollary 1 in section 3, we see that if $k$ is seen as a fraction of $n$, say $k = n/Q$, the right hand side becomes

$$m \geq C\varepsilon^{-2}(n/Q) \log(en/(2n/Q))$$

$$= C\varepsilon^{-2}n \log(eQ/2)/Q = C\varepsilon^{-2}n$$

given the substitution $C_Q = C\log(eQ/2)/Q$. Rewriting this as

$$\frac{m}{n} = \frac{p}{2p} = \frac{1}{2} \geq C\varepsilon^{-2}$$

we see that this inequality is independent of $p$. Since $\log(eQ/2)/Q$ decreases asymptotically with increasing $Q$, the value of $C_Q$ can be made sufficiently small so that the inequality above actually holds. This means that for at least some level of sparsity, errors can be corrected 100% of the time. Meanwhile, the corresponding least probability $1 - 2e^{-c\varepsilon^2m}$ (from Corollary 1) increases with $p$ since $m = p$. This means that it is actually more likely that $A$ is a good sensing matrix for higher values of $p$. We can thus conclude as follows.

Given that noise is not too dense, if one desires guaranteed error correction (guaranteed by the theory of compressed sensing), then one should choose a large $p$, whereas if error correction is occasionally allowed to fail, then one can choose a small $p$. 

26
5 Appendix

5.1 Equivalence of (P1) and linear programming

This section aims to establish the equivalence between the $\ell_1$-minimization problem and linear programming. Namely, given a matrix $A \in \mathbb{R}^{m \times n}$ which satisfies the null space property of order $k$, solutions to (P1) are equivalent to solutions of the linear program

$$\min \ |u - v|_1 \quad \text{s.t.} \quad [A \quad -A] \begin{bmatrix} u \\ v \end{bmatrix} = y , \quad \begin{bmatrix} u \\ v \end{bmatrix} \succeq 0$$

where $\succeq$ denotes the comparison $\geq$ componentwise. Let us first state that without regard for numerical accuracy in computers, optimal (minimal) solutions to linear programs are always global optima [16, p. 29] which is necessary for linear programs to be equivalent to (P1). Define the two vectors $u, v$ from the components $u_i = \max(z_i, 0)$, $v_i = \max(-z_i, 0)$, $i = 1, \ldots, n$.

Observe that these components are always non-negative. The vectors $u, v$ also have disjoint supports, since a component cannot simultaneously be positive and negative. Then $z = u - v$, and (P1) can be reformulated as

$$\min \ u_i, v_i \sum_i \ |u_i - v_i| \quad \text{s.t.} \quad A(u - v) = Au - Av = y.$$ 

This can be further reformulated using block matrix notation,

$$\min \ u_i, v_i \sum_i \ |u_i - v_i| \quad \text{s.t.} \quad [A \quad -A] \begin{bmatrix} u \\ v \end{bmatrix} = y . \quad (10)$$

But since $u_i$ and $v_i$ are never simultaneously non-zero for any $i = 1, \ldots, n$, we can separate the terms in the absolute value expression, which gives

$$\sum_i |u_i - v_i| = \sum_i (|u_i| + |v_i|) = \sum_i (|u_i| + |v_i|).$$

By definition, $u_i$ and $v_i$ are non-negative, which means (10) is equivalent to

$$\min \ u_i, v_i \sum_i (u_i + v_i) \quad \text{s.t.} \quad [A \quad -A] \begin{bmatrix} u \\ v \end{bmatrix} = y .$$

Observe also the apparent equality

$$\sum_i (u_i + v_i) = \left( \sum_i u_i \right) + \left( \sum_j v_j \right) = [1, 1, 1, \ldots, 1][u_1, \ldots, u_n, v_1, \ldots, v_n]^T.$$
Now, the implicit constraint here is that all entries of the vector $[u \ v]^T$ are non-negative, which allows (P1) to finally be reformulated as the linear program

$$\begin{align*}
\min & \quad [1, 1, \ldots, 1] \begin{bmatrix} u \\ v \end{bmatrix} \\
\text{s.t.} & \quad [A \ -A] \begin{bmatrix} u \\ v \end{bmatrix} = y, \quad \begin{bmatrix} u \\ v \end{bmatrix} \succeq 0
\end{align*}$$

where $\succeq$ denotes the comparison $\geq$ componentwise. As already stated, solutions to linear programs are always global minima, so the corresponding solution to (P1) will be the vector $x = u - v$. For this thesis, the function `linprog` in Matlab was used for solving (P1). In the making of the numerical simulations in section 4, the default algorithm in Matlab for `linprog` was the dual simplex algorithm (details on this algorithm can be found in [16, p. 158]).

### 5.2 Upper bound of error margins

Section 4.3 deals with several cases of estimating an unknown probability $q$. In numerical simulations, this probability is estimated by the fraction

$$\hat{q} = \frac{1}{N} \sum_{i=1}^{N} X_i$$

where $N = 1000$ and $X_i$ are Bernoulli random variables of value 1 (with probability $q$) in case of success and 0 (with probability $1 - q$) in case of failure. It is very important to have an idea of how precise such an estimate of $q$ is, so that we know whether to trust a given estimate. This section is dedicated to showing an upper bound to confidence intervals of $\hat{q}$, within which we can be at least 95 % sure that the actual value $q$ resides.

The sum in equation (11) follows the binomial distribution $B(N, q)$. Let $Y$ denote this sum, so that $\frac{1}{N} Y$ is the estimate $\hat{q}$ of $q$. Then the mean and variance of $Y$ is $Nq$ and $Nq(1 - q)$ respectively. The standard deviation of $Y$ is the square root of the variance. Recall the following property of the variance

$$\text{Var} \left( \frac{1}{N} Y \right) = \frac{1}{N^2} \text{Var}(Y).$$

The standard deviation of $\hat{q}$ is then

$$\sqrt{\frac{1}{N^2} Nq(1 - q)} = \sqrt{\frac{q(1 - q)}{\sqrt{N}}}$$

which means that the precision of $\hat{q}$ increases with the square root of $N$. Constructing a confidence interval from this can be done by invoking the Central Limit Theorem [19, p. 233-234]. If $q$ is somewhat close to 1/2, then the (continuous) normal distribution $N(q, \sqrt{q(1 - q)/\sqrt{N}})$ closely approximates the (discrete) distribution of $Y/N$ for large values of $N$ [19, p. 421]. Recalling that the 2.5 % percentile is approximately 1.96, we can say with about 95 % confidence that the actual value $q$ lies within the interval
This interval is of the form \([\hat{\mu} - c\sigma, \hat{\mu} + c\sigma]\) where \(\hat{\mu}\) approximately follows \(\mathcal{N}(\mu, \sigma)\) and \(c\) is determined by some level of confidence. Of course, determining the standard deviation \(\sigma = \sqrt{q(1 - q)}/\sqrt{N}\) requires knowledge of \(q\) which is what we attempt to estimate in the first place, so we cannot stop here. One could estimate the (unknown) variance by the sample variance

\[
S^2 = \frac{1}{N - 1} \sum_{i=1}^{N} (X_i - \hat{q})^2 \approx q(1 - q)/N
\]

from which the standard deviation can be approximated by \(S\) (the question of whether \(S\) is a biased estimator is unimportant for large \(N\)). But this could yield \(S = 0\) if \(q\) is very close to 0 or 1, which means that \(\hat{q}\) is estimated to be infinitely precise (which is quite unrealistic). A problem related to this anomaly is that for fixed \(N\), a normal approximation of \(Y/N\) is worse when \(q\) is closer to 0 or 1 [19, p. 187-192]. Thus, the confidence interval in (12) loses its meaning as \(q\) tends further away from 1/2. To avoid this problem, we have chosen to only consider an upper bound of error margins, by replacing the confidence interval in (12) with the alternative interval

\[
\left[ \hat{q} - 1.96 \frac{\sqrt{N}}{\sqrt{q(1 - q)}}\sqrt{\frac{1}{2} \left( 1 - \frac{1}{2} \right)}, \ \hat{q} + 1.96 \frac{\sqrt{N}}{\sqrt{q(1 - q)}}\sqrt{\frac{1}{2} \left( 1 - \frac{1}{2} \right)} \right].
\]

This comes from the fact that \(\sqrt{q(1 - q)}\) has the largest value at \(q = 1/2\). This interval is rather a superset of the corresponding interval in (12), and it can even grossly overestimate the interval in (12) for \(q\) close to 0 or 1, but it avoids the problem of a precision overestimate from poor normal approximation. In the worst case, we only understate our confidence, so we can instead speak of a confidence at least 95 %. Evaluating for \(N = 1000\) and \(q = 1/2\) gives

\[
\frac{1.96}{\sqrt{N}} \sqrt{q(1 - q)} \approx 0.03099 < 0.031
\]

and we can then say with at least 95 % confidence that \(q \in [\hat{q} - 0.031, \hat{q} + 0.031]\) holds. This is of course not an exact statement, since a binomial distribution can only approximate the (continuous) normal distribution, but in approximate terms, the statement is valid.
5.3 Matlab code

5.3.1 Functions

function e_rec = reconstructed_vector(A, f)
    [~, n] = size(A);
    lb = zeros(2*n, 1);
    ub = Inf(2*n, 1);
    c = ones(2*n, 1);

    % This is just to prevent linprog from printing stuff.
    options = optimset('linprog');
    options.Display = 'off';

    % This is what Appendix section 5.1 is about.
    uv = linprog(c, -speye(2*n), lb, [A, -A], f, lb, ub, options);
    u = uv(1:n);
    v = uv((n+1):(2*n));
    e_rec = (u - v)';
end

function s = ec_success_rate(A, k, N, rng_state)
    % N is just how precise we want it to be.
    [~, n] = size(A);
    e_vecs = zeros(n, N);
    rng(rng_state);
    for i = 1:N
        e_vecs(:, i) = random_sparse_vector(n, k);
    end
    f_vecs = A * e_vecs;

    c = 0;
    for i = 1:N
        e = e_vecs(:, i);
        f = f_vecs(:, i);
        e_rec = reconstructed_vector(A, f);

        % If true, the reconstruction was successful.
        % 1e-12 is close enough for success.
        if norm(e - e_rec) <= 1e-12
            c = c + 1;
        end
    end
    s = c / N;
end
function A = random_gaussian_matrix(m, n)
    A = randn(m, n);
end

function e = random_sparse_vector(n, k)
    e = zeros(n, 1);
    r = randn(k, 1);
    e(1:k, 1) = r;
    e = e(randperm(n));
end

5.3.2 Figure 3 data generation
rng_state = rng(12345, 'twister');
rng_state2 = rng(23456, 'twister');
p = 32;
n = 2*p;
m = n - p;
krange = 1:24;

rng(rng_state);
A1 = random_gaussian_matrix(m, n);
A2 = random_gaussian_matrix(m, n);
A3 = random_gaussian_matrix(m, n);
A4 = random_gaussian_matrix(m, n);

for k = 1:24
    r(k, 1) = ec_success_rate(A1, k, 1000, rng_state2);
end

for k = 1:24
    r(k, 2) = ec_success_rate(A2, k, 1000, rng_state2);
end

for k = 1:24
    r(k, 3) = ec_success_rate(A3, k, 1000, rng_state2);
end

for k = 1:24
    r(k, 4) = ec_success_rate(A4, k, 1000, rng_state2);
end
5.3.3 Figure 4 data generation

```matlab
p = 32;
rng_state = rng(12345, 'twister');
rng_state2 = rng(23456, 'twister');

% These factors are how many times larger n is than p.
% multiplying these with p must yield even integers.
encoding_factors = (-4:4)/8 + 2;

n_values = p * encoding_factors;
m_values = n_values - p;
kmax_values = n_values / 2;
success_rate_graphs = {};

for i = 1:9
    success_rate_graphs{i} = zeros(kmax_values(i), 1);
end

for i = (10-(1:9))
    m = m_values(i);
    n = n_values(i);
    rng(rng_state);
    A = random_gaussian_matrix(m, n);
    for k = 1:(kmax_values(i))
        result = ec_success_rate(A, k, 1000, rng_state2);
        success_rate_graphs{i}(k, 1) = result;
    end
end
```

5.3.4 Figure 5 data generation

```matlab
rng_state = rng(12345, 'twister');
rng_state2 = rng(23456, 'twister');

% These are various values for k/n.
sparsity_percentages = (1:4)/16;

p_values = (1:8)*16;
n_values = 2 * p_values;
m_values = n_values - p_values;
k_values = zeros(8, 4);
```
for i = 1:4
    for j = 1:8
        k_values(j, i) = sparsity_percentages(i) * n_values(j);
    end
end

success_rate_graphs = zeros(8, 4);

for i = 1:4
    for j = (9-(1:8))
        m = m_values(j);
        n = n_values(j);
        rng(rng_state);
        A = random_gaussian_matrix(m, n);
        k = k_values(j, i);
        result = ec_success_rate(A, k, 1000, rng_state2);
        success_rate_graphs(j, i) = result;
    end
end

5.3.5 Figure 6 data generation

rng_state = rng(12345, 'twister');
rng_state2 = rng(23456, 'twister');

p1 = 16;
p2 = 64;

n1 = 2*p1; n2 = 2*p2;
m1 = n1-p1; m2 = n2-p2;

r1 = zeros(m1, 1);
r2 = zeros(m2, 1);

rng(rng_state);
A1 = random_gaussian_matrix(m1, n1);
A2 = random_gaussian_matrix(m2, n2);

for k = 1:m2
    r2(k, 1) = ec_success_rate(A2, k, 1000, rng_state2);
end

for k = 1:m1
    r1(k, 1) = ec_success_rate(A1, k, 1000, rng_state2);
end
References


