# Compressed sensing and the basis pursuit algorithm

**Dias Da Cruz Steve**

Student project
Master in mathematics

Faculty of Sciences, Technology and Communication
University of Luxembourg
December 4, 2015

# 1 Introduction

In image processing, we want to reconstruct an image as good as possible from an as small as possible amount of measured data. When the information acquisition is linear, the problem is reduced to a linear system of equations. The measured information will be represented by a vector $x \in \mathbb{R}^m$, whereas the original information is represented by a vector $y \in \mathbb{R}^N$. The linear measurement is then represented by a matrix $A \in \mathbb{R}^{m \times N}$. The reconstruction of $x$ is then identical to solving the linear system

$$y = Ax \tag{1}$$

However, from linear algebra we know that if $m < N$ our system of equations will have infinitely many solutions (if there exists at least one solution). Thus, it will not be possible to obtain a unique solution when we want to reconstruct $x$. However, under certain assumptions it will be possible to reconstruct the initial source even though the number $m$ of available measurements will be smaller than the signal itself $N$. For this, we need to introduce sparsity:

**Definition 1.** A vector is called ***k-sparse*** if at most $k$ entries are non-zero:

$$\|x\|_0 = \#\{i : x_i \neq 0\} \leq k. \tag{2}$$

We say that our original image is sparse, if it is compressible. This means, that an image is well approximated by a sparse vector (JPEG only stores the largest wavelet coefficients [4]). For instance, if we want to compress an image, we would measure every single pixel, discard most of the measurements and keep the important ones. Conversely, if we want to reconstruct an image, we will not need to know every single pixel of the original image. It will be enough to measure the compressed version "directly". Nevertheless, the main problem is that we cannot know the positions of the zero entries of the original image in advance. If we want to reconstruct a sparse image, we need to specify two thinks:

1. Which matrices $A \in \mathbb{R}^{m \times N}$ are suitable to represent our linear measurements? We cannot choose arbitrary matrices, since not every matrix is fitted for compressed sensing[1][2].

2. Are there efficient algorithms to reconstruct $x$ from $y = Ax$?

[1] The identity matrix would mostly pick zero entries of $x$ and hence almost no information is available about the non-zero entries.

[2] The matrix should be designed for all x simultaneously.

A first algorithm, which one could propose, would be to minimize the number non-zero entries in $x$. We want to find the sparsest vector $z$, i.e. the vector $z$ with the minimal amount of non-zero entries, thus, the vector $z$ with the maximal amount of zero entries, such that it is consistent with the equation $y = Ax$. Thus, we want

$$\text{minimize } \|z\|_0 \qquad \text{subject to } Az = y, \tag{$P_0$}$$

which then implies that the vector $z$ obtained verifies $Az = Ax$. Unfortunately, for most cases, this equation is NP[3]-hard to solve [5]. However, by replacing $\|x\|_0$ by the $l_1$-norm[4] one can solve the following optimization much more efficiently

$$\text{minimize } \|z\|_1 \qquad \text{subject to } Az = y, \tag{$P_1$}$$

because the latter is a convex function, for which exist efficient convex optimization methods. Due to the shape of the $l_1$ ball, $l_1$ minimization will indeed favor sparsity[1].

Next, we will need to define suitable measurement matrices. However, it is still an open problem to construct explicit matrices, which would be optimal for a reconstruction. If we are allowed to choose the matrix freely (no additional conditions), the most promising technique and also the breakthrough of compressed sensing, is to choose matrices with random coefficients. The easiest promising examples are Gaussian matrices, which consists of independent random variables following a standard normal distribution $N(0, 1)$ or Bernoulli[5] matrices, which consists of independent random variables taking the values $+1$ and $-1$ with equal probability. An important result in compressed sensing is, by choosing such a $m \times N$ Gaussian or Bernoulli matrix, that you will be able to reconstruct with high probability all $k$-sparse vectors $x$ from $y = Ax$, provided that

$$m \geq Ck \ln\left(\frac{N}{k}\right), \tag{3}$$

where $C > 0$ is a constant independent of $k, m$ and $N$ and where the bound is in fact optimal. This actually means that the amount $m$ of data needed to reconstruct a $k$-sparse vector scales almost linearly in $k$, while the length $N$ only has a logarithmic influence.

[3] non-polynomial time complexity

[4] $\sum_i |x_i|$

[5] $P(X = 1) = 1 - P(X = -1) = 1 - q = p$

# 2 Basis pursuit

## 2.1 Null space property

This project is largely inspired on [2] and will focus on the basis pursuit ($l_1$-minimization) strategy, which consists on solving the convex optimization problem

$$\text{minimize } \|z\|_1 \qquad \text{subject to } Az = y, \tag{$P_1$}$$

where $z \in \mathbb{C}^N$. A solution to this minimization problem is called the ***minimizer***. First of all, we need to investigate a necessary condition on $A$ to ensure an exact reconstruction of sparse vectors when using basis pursuit. For the sake of simplicity, we introduce the notation $v_S$ for a vector $v \in \mathbb{C}^N$ and a set $S \subset [N] = \{1, 2, \ldots, N\}$, which then represents either the vector $v \in \mathbb{C}^S$ as a restriction of $v$ to the indices of $S$, or the vector $v \in \mathbb{C}^N$, which coincides with the vector $v$ on the indices of $S$.

**Definition 2.** The support of a vector $x \in \mathbb{C}^N$ is the index set of its nonzero entries, i.e.,

$$\operatorname{supp}(x) := \{j \in [N] : x_j \neq 0\}. \tag{4}$$

We also say that $x$ is supported on the set $S \in N$ if $\operatorname{supp}(x) = S$. Hence, all the coefficients of $x$ in $\overline{S}$ are 0.

**Definition 3.** A matrix $A \in \mathbb{K}^{m \times N}$ is said to satisfy the ***null space property*** relative to a set $S \subset [N]$ if

$$\|v_S\|_1 < \|v_{\overline{S}}\|_1 \qquad \text{for all } v \in \ker A \setminus \{0\}, \tag{5}$$

where $\overline{S}$ is the complement of $S$. It is said to satisfy the null space property of order $k$ if it satisfies the null space property relative to any set $S \subset [N]$ with $\#S \leq k$.

**Remark 2.1.** *Since $S \cup \overline{S} = N$, we get when adding $\|v_S\|_1$ on both sides of the inequality 5 the following result*

$$2\|v_S\|_1 < \|v\|_1 \qquad \text{for all } v \in \ker A \setminus \{0\}. \tag{6}$$

**Theorem 2.1.** *Given a matrix $A \in \mathbb{K}^{m \times N}$, every vector $x \in \mathbb{K}^N$ supported on a set $S$ is the unique solution of ($P_1$) with $y = Ax$ if and only if $A$ satisfies the null space property relative to $S$.*

*Proof.* $\Longrightarrow$: Let us fix $S$ and suppose that every vector $x \in \mathbb{K}^N$ supported on $S$ is the unique minimizer of $\|z\|_1$ subject to $Az = Ax$. Consequently, every vector $v \in \ker A \setminus \{0\}$ implies that $v_S$ is the unique minimizer of $\|z\|_1$

subject to $Az = Av_S$ (since it is true for every vector of $\mathbb{K}^N$ supported on $S$). We have that $A(v_{\overline{S}} + v_S) = Av = 0$, since $v \in \ker A \setminus \{0\}$ and $v \neq 0$. This implies that $A(-v_{\overline{S}}) = Av_S$. Since $v_S$ is the unique minimizer, we get that $\|v_S\|_1 < \|v_{\overline{S}}\|_1$, which then implies the null space property relative to $S$.

$\Longleftarrow$: Suppose that the null space property relative to S holds. Take a vector $x \in \mathbb{K}^N$ supported on S. Next, take another vector $z \in \mathbb{K}^N$,$z \neq x$ satisfying $Az = Ax$. We need to prove that $x$ is a minimizer of $\|z\|_1$.$Az = Ax \implies Az - Ax = 0 \implies A(z-x) = 0 \implies z - x \in \ker A \setminus \{0\}$ (since $x \neq z$). Define $v := x - z \in \ker A \setminus \{0\}$. By the null space property we get

$$\begin{aligned} \|x\|_1 \leq \|x - z_S\|_1 + \|z_S\|_1 &= \|v_S\|_1 + \|z_S\|_1 \\ &\leq \|v_{\overline{S}}\|_1 + \|z_S\|_1 = \|-z_{\overline{S}}\|_1 + \|z_S\|_1 = \|z\|_1, \end{aligned}$$

where $\|v_S\|_1 \leq \|v_{\overline{S}}\|_1$ because of the null space property and $\|v_{\overline{S}}\|_1 = \|x_{\overline{S}}\|_1 - \|z_{\overline{S}}\|_1$ with $\|x_{\overline{S}}\|_1 = 0$, since $x$ is supported on $S$. □

By definition of the null space property and the related null space property of order $k$ and by letting vary the set $S$ we obtain as an immediate consequence of this the following theorem:

**Theorem 2.2.** *Given a matrix $A \in \mathbb{K}^{m \times N}$, every $k$-sparse vector $x \in \mathbb{K}^N$ is the unique solution of $(P_1)$ with $y = Ax$ if and only if $A$ satisfies the null space property of order $k$.*

Remember, that the goal of compressed sensing is to find the sparsest vector, i.e. the vector which is the minimizer for the equation $(P_0)$. For this, let us check the relation between the $l_1$-minimization and the $l_0$-minimization. Assume that the null space property of order $k$ is satisfied. The theorem then implies that every $k$-sparse vector $x$ is the unique solution of $(P_1)$ and it is recovered from $y = Ax$ and $l_1$-minimization. Suppose that $z$ is the minimizer of the $l_0$-minimization with $y = Ax$. The last assumption implies that $\|z\|_0 \leq \|x\|_0$ (since it is the minimizer). Since $x$ is $k$-sparse and by the definition of $\|\cdot\|_0$, this implies that $z$ is $k$-sparse. However, since every $k$-sparse vector is the unique $l_1$-minimizer, it follows that $x = z$.

## 2.2 Stability

In a realistic and applied scenario, a vector would usually not be a sparse vector, but only be close to a sparse vector. Thus, we need to recover a vector $x \in \mathbb{C}^N$ with an error controlled by its distance to $k$-sparse vectors. This property is called ***stability*** of the reconstruction scheme with respect to sparsity defect. The basis pursuit remains stable under a strengthened version of the null space property:

**Definition 4.** A matrix $A \in \mathbb{C}^{m\times N}$ is said to satisfy the ***stable null space property*** with constant $0 < \rho < 1$ relative to a set $S \subset [N]$ if

$$\|v_S\|_1 \leq \rho \|v_{\overline{S}}\|_1 \qquad \text{for all } v \in \ker A. \tag{7}$$

It is said to satisfy the stable null space property of order $k$ with constant $0 < \rho < 1$ if it satisfies the stable null space property with constant $0 < \rho < 1$ relative to any set $S \subset [N]$ with $\#S \leq k$.

Before we come to the main statement of this section, we will need a lemma to prove the upcoming theorem:

**Lemma 2.1.** *Given a set $S \subset [N]$ and vectors $x, z \in \mathbb{C}^N$. Then,*

$$\|(x-z)_{\overline{S}}\|_1 \leq \|z\|_1 - \|x\|_1 + \|(x-z)_S\|_1 + 2\|x_{\overline{S}}\|_1 \tag{8}$$

*Proof.* First, it is clear that we have these two inequalities:

$$\begin{aligned} \|x\|_1 = \|x_{\overline{S}}\|_1 + \|x_S\|_1 &\leq \|x_{\overline{S}}\|_1 + \|(x-z)_S\|_1 + \|z_S\|_1 \\ \|(x-z)_{\overline{S}}\|_1 &\leq \|x_{\overline{S}}\|_1 + \|z_{\overline{S}}\|_1. \end{aligned}$$

If we sum up both inequalities, we will get:

$$\|x\|_1 + \|(x-z)_{\overline{S}}\|_1 \leq 2\|x_{\overline{S}}\|_1 + \|(x-z)_S\|_1 + \|z\|_1,$$

since $\|z_{\overline{S}}\|_1 + \|z_S\|_1 = \|z\|_1$ □

**Theorem 2.3.** *The matrix $A \in \mathbb{C}^{m\times N}$ satisfies the stable null space property with constant $0 < \rho < 1$ relative to $S$ if and only if*

$$\|z - x\|_1 \leq \frac{1+\rho}{1-\rho}(\|z\|_1 - \|x\|_1 + 2\|x_{\overline{S}}\|_1) \tag{9}$$

*for all vectors $x, z \in \mathbb{C}^N$ with $Az = Ax$. This actually implies that the difference between the vector $z$ and $x$ is controlled by the difference between their norms.*

*Proof.* $\Longleftarrow$: For this implication, let us first assume that $A$ satisfies (9) for all vectors $x, z \in \mathbb{C}^N$ with $Az = Ax$. We then need to prove that $A$ satisfies (7) for all $v \in \ker A$. Take $v \in \ker A$. This implies that $Av = 0$ and since we know that $v = v_S + v_{\overline{S}}$, we can write $Av_{\overline{S}} = A(-v_S)$. Now, we can apply (9) with $x = -v_S$ and $z = v_{\overline{S}}$, which implies

$$\|v_{\overline{S}} - (-v_S)\|_1 \leq \frac{1+\rho}{1-\rho}(\|v_{\overline{S}}\|_1 - \|-v_S\|_1 + 2\|-v_{\overline{S}}\|_1),$$

which then implies

$$\|v\|_1 \leq \frac{1+\rho}{1-\rho}(\|v_{\overline{S}}\|_1 - \|v_S\|_1).$$

Multiplying both sides by $(1-\rho)$ and decomposing $\|v\|_1$ yields

$$(1-\rho)(\|v_S\|_1 + \|v_{\overline{S}}\|_1) \leq (1+\rho)(\|v_{\overline{S}}\|_1 - \|v_S\|_1),$$

which finally leads to

$$\|v_S\|_1 \leq \rho\|v_{\overline{S}}\|_1.$$

The last inequality is precisely the condition for $A$ to satisfy the stable null space property with constant $0 < \rho < 1$ relative to $S$.

$\Longrightarrow$: Now, for this implication, let's assume that $A$ satisfies the stable null space property with constant $0 < \rho < 1$ relative to $S$. Thus, we need to prove the inequality (9) for all vectors $x, z \in \mathbb{C}^N$ with $Az = Ax$. First, take $x, z \in \mathbb{C}^N$ such that $Az = Ax$. Then $Az = Ax \implies A(z-x) = 0 \implies z - x \in \ker A$. Define $y = z - x \in \ker A$. The stable null space property then implies that

$$\|v_S\|_1 \leq \rho\|v_{\overline{S}}\|_1. \tag{10}$$

The previous Lemma 2.1 states that

$$\|(x-z)_{\overline{S}}\|_1 \leq \|z\|_1 - \|x\|_1 + \|(x-z)_S\|_1 + 2\|x_{\overline{S}}\|_1,$$

which in our situation yields

$$\|v_{\overline{S}}\|_1 \leq \|z\|_1 - \|x\|_1 + \|v_S\|_1 + 2\|x_{\overline{S}}\|_1. \tag{11}$$

Replacing (10) into (11) implies

$$\|v_{\overline{S}}\|_1 \leq \|z\|_1 - \|x\|_1 + \rho\|v_{\overline{S}}\|_1 + 2\|x_{\overline{S}}\|_1.$$

This can be rewritten as

$$\|v_{\overline{S}}\|_1 - \rho\|v_{\overline{S}}\|_1 \leq \|z\|_1 - \|x\|_1 + 2\|x_{\overline{S}}\|_1.$$

Factoring out $\|v_{\overline{S}}\|_1$ and dividing by $(1-\rho)$ (since $0 < \rho < 1$) gives

$$\|v_{\overline{S}}\|_1 \leq \frac{1}{1-\rho}(\|z\|_1 - \|x\|_1 + 2\|x_{\overline{S}}\|_1).$$

Once again, we apply the inequality (10) and we get

$$\|v\|_1 = \|v_{\overline{S}}\|_1 + \|v_S\|_1 \leq \|v_{\overline{S}}\|_1 + \rho\|v_{\overline{S}}\|_1 \leq \frac{1+\rho}{1-\rho}(\|z\|_1 - \|x\|_1 + 2\|x_{\overline{S}}\|_1),$$

which is the desired inequality. $\square$

This theorem implies a weaker theorem, for which we first need to introduce a new notation.

**Definition 5.** For $p > 0$, the ***$l_p$-error of best $k$-term approximation*** to a vector $x \in \mathbb{C}^N$ is defined by

$$\sigma_k(x)_p := \inf\{\|x - z\|_p, z \in \mathbb{C}^N \text{ is } k\text{-sparse}\}. \tag{12}$$

The infimum is achieved for a $k$-sparse vector $z \in \mathbb{C}^N$, whose nonzero entries equal the $k$ largest absolute entries of $x$. The infimum is achieved independently of $p > 0$

**Theorem 2.4.** *Suppose that a matrix $A \in \mathbb{C}^{m \times N}$ satisfies the stable null space property of order $k$ with constant $0 < \rho < 1$. Then, for any $x \in \mathbb{C}^N$, a solution $x^\#$ of $(P_1)$ with $y = Ax$ approximates the vector $x$ with $l_1$-error*

$$\|x - x^\#\|_1 \leq \frac{2(1+\rho)}{(1-\rho)} \sigma_k(x)_1. \tag{13}$$

*This theorem cannot guarantee the uniqueness of the $l_1$-minimizer. Nevertheless, even when the $l_1$-minimizer is not unique, the theorem states that every solution $x^\#$ of $(P_1)$ with $y = Ax$ satisfies (13).*

*Proof.* Theorem 2.3 implies this theorem. Take $S$ to be the set of $k$ largest absolute coefficients of $x$, so that $\|x_{\overline{S}}\|_1 = \sigma_k(x)_1$ (by the previous definition). If $x^\#$ is a minimizer of $(P_1)$, then $\|x^\#\|_1 \leq \|x\|_1$ and $Ax^\# = Ax$. The right-hand side of the inequality (9) with $z = x^\#$ can therefore be estimated by the righ-hand side of the inequality (13). □

## 2.3 Robustness

The next problem is that, in a realistic environment, it will be impossible to measure a signal $x \in \mathbb{C}^N$ with an infinite precision. Hence, the measurement vector $y \in \mathbb{C}^m$ is only an approximation of the vector $Ax \in \mathbb{C}^m$. Thus, we have

$$\|Ax - y\| \leq \eta,$$

for some $\eta \geq 0$ and for some norm $\|\cdot\|$ on $\mathbb{C}^m$ (usually the $l_2$-norm or the $l_1$-norm). Our algorithm should output a vector, whose distance to the original vector is controlled by the measurement error $\eta \geq 0$. This property is called ***robustness*** of the reconstruction scheme with respect to measurement error. Take $z \in \mathbb{C}^N$ and replace the problem $(P_1)$ by the convex optimization problem

$$\text{minimize } \|z\|_1 \qquad \text{subject to } \|Az - y\| \leq \eta. \tag{$P_{1,\eta}$}$$

The robustness of the basis pursuit algorithm is then guaranteed by another strengthening of the null space property. Namely, by the following

**Definition 6.** A matrix $A \in \mathbb{C}^{m\times N}$ is said to satisfy the ***robust null space property*** (with respect to $\|\cdot\|$) with constant $0 < \rho < 1$ and $\tau > 0$ relative to a set $S \subset [N]$ if

$$\|v_S\|_1 \le \rho\|v_{\overline{S}}\|_1 + \tau\|Av\| \qquad \text{for all } v \in \mathbb{C}^N. \tag{14}$$

It is said to satisfy the robust null space property of order $k$ with constant $0 < \rho < 1$ and $\tau > 0$ if it satisfies the robust null space property with constant $\rho$ and $\tau$ relative to any set $S \subset [N]$ with $\#S \le k$.

**Remark 2.2.** *If $v \in \ker A$, then $Av = 0$ and hence $\|Av\| = 0$. In that case, the robust null space property would imply the stable null space property.*

As before, we are going to prove a stronger theorem, which will then imply a weaker one, which will immediately be related to the $(P_{1,\eta})$ problem.

**Theorem 2.5.** *The matrix $A \in \mathbb{C}^{m\times N}$ satisfies the robust null space property with constant $0 < \rho < 1$ and $\tau > 0$ relative to $S$ if and only if*

$$\|z - x\|_1 \le \frac{1+\rho}{1-\rho}(\|z\|_1 - \|x\|_1 + 2\|x_{\overline{S}}\|_1) + \frac{2\tau}{1-\rho}\|A(z-x)\| \tag{15}$$

*for all vectors $x, z \in \mathbb{C}^N$ with $Az = Ax$.*

*Proof.* $\Longleftarrow$: First let assume that $A$ satisfies (15) for all vectors $x, z \in \mathbb{C}^N$ with $Az = Ax$. Take $v \in \mathbb{C}^N$ and decompose it into $v = v_{\overline{S}} + v_S$. Then we can take $x = -v_S$ and $z = v_{\overline{S}}$. By applying inequality (15), we obtain

$$\|v\|_1 = \|v_{\overline{S}} + v_S\|_1 \le \frac{1+\rho}{1-\rho}(\|v_{\overline{S}}\|_1 - \|v_S\|_1 + 2\|0\|_1) + \frac{2\tau}{1-\rho}\|A(v_{\overline{S}} + v_S)\|,$$

which is equal to

$$\|v\|_1 \le \frac{1+\rho}{1-\rho}(\|v_{\overline{S}}\|_1 - \|v_S\|_1) + \frac{2\tau}{1-\rho}\|Av\|.$$

Multiplying both side by $(1-\rho)$ and decomposing $v = v_{\overline{S}} + v_S$ leads to

$$(1-\rho)\|v_{\overline{S}} + v_S\|_1 \le (1+\rho)(\|v_{\overline{S}}\|_1 - \|v_S\|_1) + 2\tau\|Av\|.$$

Rearranging the terms then leads to

$$2\|v_S\|_1 \le 2\rho\|v_{\overline{S}}\|_1 + 2\tau\|Av\|,$$

which is finally equal to

$$\|v_S\|_1 \leq \rho \|v_{\overline{S}}\|_1 + \tau \|Av\|$$

and thus $A$ fulfills the conditions to verify the robust null space property with constant $0 < \rho < 1$ and $\tau > 0$ relative to $S$.

$\Longrightarrow$: For this implication, assume that $A$ satisfies the robust null space property with constant $0 < \rho < 1$ and $\tau > 0$ relative to $S$. Now, take $x, z \in \mathbb{C}^N$ and define $v := z - x$ ($v$ does not necessarily lie in $\ker A$!). Then, the robust null space property implies that

$$\|v_S\|_1 \leq \rho \|v_{\overline{S}}\|_1 + \tau \|Av\|.$$

Lemma 2.1 implies

$$\|v_{\overline{S}}\|_1 \leq \|z\|_1 - \|x\|_1 + \|v_S\|_1 + 2\|x_{\overline{S}}\|_1,$$

since $v = z - x$. Replacing the first inequality into the second one yields:

$$\|v_{\overline{S}}\|_1 \leq \|z\|_1 - \|x\|_1 + \rho \|v_{\overline{S}}\|_1 + \tau \|Av\| + 2\|x_{\overline{S}}\|_1,$$

which then implies the following inequality

$$\|v_{\overline{S}}\|_1 \leq \frac{1}{1-\rho}(\|z\|_1 - \|x\|_1 + \tau \|Av\| + 2\|x_{\overline{S}}\|_1).$$

Appliying this inequality and the robust null space property to the decomposition of $v$ finally yields

$$\begin{aligned}
\|v\|_1 = \|v_S\|_1 + \|v_{\overline{S}}\|_1 &\leq \rho \|v_{\overline{S}}\|_1 + \tau \|Ax\| + \|v_{\overline{S}}\|_1 \\
&= (1+\rho)\|v_{\overline{S}}\|_1 + \tau \|Ax\| \\
&\leq \frac{1+\rho}{1-\rho}(\|z\|_1 - \|x\|_1 + \tau \|Av\| + 2\|x_{\overline{S}}\|_1) + \tau \|Ax\| \\
&= \frac{1+\rho}{1-\rho}(\|z\|_1 - \|x\|_1 + 2\|x_{\overline{S}}\|_1) + \frac{2\tau}{1-\rho}\|Ax\|,
\end{aligned}$$

which is the desired inequality. □

This theorem then implies an extension of theorem 2.4. For the case of $\eta = 0$ both theorem actually coincide.

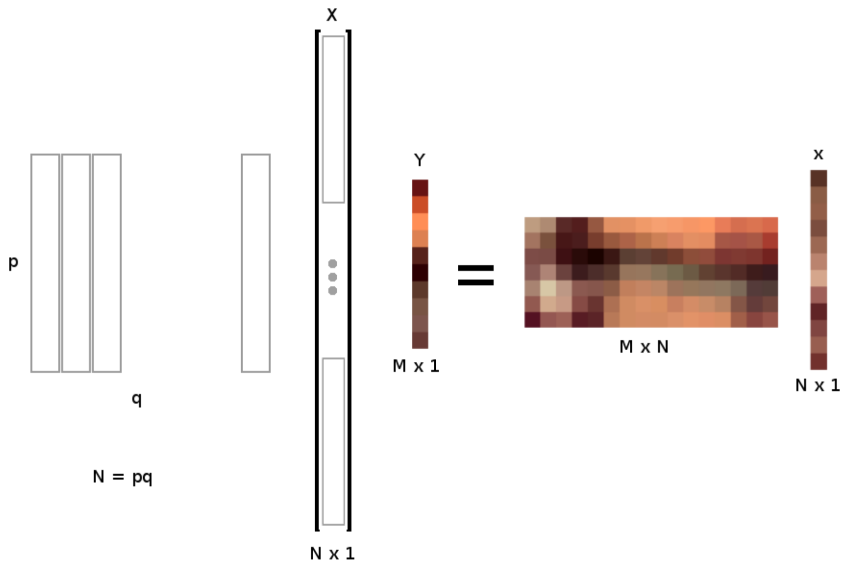
**Theorem 2.6.** *Suppose that a matrix $A \in \mathbb{C}^{m \times N}$ satisfies the robust null space property of order $k$ with constant $0 < \rho < 1$ and $\rho > 0$. Then, for any $x \in \mathbb{C}^N$, a solution $x^\#$ of $(P_{1,\eta})$ with $y = Ax + \mathrm{e}$ and $\|\mathrm{e}\| \leq \eta$ approximates the vector $x$ with $l_1$-error*

$$\|x - x^\#\|_1 \leq \frac{2(1+\rho)}{(1-\rho)}\sigma_k(x)_1 + \frac{4\tau}{1-\rho}\eta. \tag{16}$$

# 3 Implementation

## 3.1 Introduction

An image is usually represented by a matrix $A$, where the entry $a_{ij}$ represents the color of the pixel at position $ij$. Since the basis pursuit algorithm reconstructs and works for vectors, we need to represent the image as a vector. For this, the representation will need to be as in the next picture:




This means, that the different columns of the matrix will be stacked, one after each other, to one big vector. Then, we can apply the basis pursuit algorithm, since we have a standard problem to solve. The result will be a another vector, which will need to be re-transformed into a matrix in order to get an image as a result. Providing the example, our vector $x$ is constructed by decomposing the matrix $I \in \mathbb{R}^{p \times q}$ by taking $q$ columns of height $p$ and putting them, one after each other, to one big vector of size $N \times 1$, where $N = pq$.

However, there are even more constraints. There are different possibilities on how an image can be represented in a matrix. The color of a pixel is defined by its RGB value. This means, that every pixel is defined by its red, green and blue value. Each characteristic can take a value from 0 to 255. Another possibility would be to represent the color as a hex-value. However, for simplicity of this project, I am going to consider only images in gray-scale. This has the advantage, that the image can be represented by a usual 2-dimensional matrix, where the entries take values from 0 to 255 indicating the gray-scale value.

Last but not least, I need to mention the ***discrete cosine transform***. The discrete cosine transform represents an image as a sum of sinusoids of varying magnitudes and frequencies. The discrete cosine transform has the property that most of the visually significant information about an image is concentrated in just a few coefficients of the discrete cosine transform. For instance, the JPEG compression is using the discrete cosine transform to compress images efficiently. The inverse of the discrete cosine transform reconstructs a sequence from its discrete cosine transform coefficients. In our algorithm, we are reconstructing an image using the inverse discrete cosine transform in order to get a better and more efficient result, since we are interested in the visually significant information about an image.

I am going to present an algorithm, which will be implemented in MATLAB. Moreover, to perform the basis pursuit algorithm, I am going to use the $l_1$-magic[6] package.

I will shortly mention how one could solve the problem $(P_1)$ in practice. If you want to implement a solution to this problem, it is much easier and convenient to solve the equivalent problem to $(P_1)$, which is called a linear program. So, instead of solving

$$\text{minimize } \|z\|_1 \qquad \text{subject to } Az = y, \tag{$P_1$}$$

we are going to solve

$$\text{minimize} \sum_{j=1}^{N} c_j \qquad \text{subject to } Az = y \text{ and } -c_j \leq z_j \leq c_j, \tag{$P_1'$}$$

where $c_j \in \mathbb{R}^N$ and $z = (z_1, z_2, \ldots, z_N) \in \mathbb{R}^N$. This linear program is exactly what the $l_1$-magic file will perform, when called during the following implementation.

## 3.2 Implementation

I will put enough comments in the code in order to explain why the different steps are necessary. This implementation is an adaption of a script by Stuart Gibson[7]. I chose to use gaussian matrices, since they satisfy the null space property with a high probability [2].

```
%___DEPENDENCIES___
% Requires l_1-MAGIC: Recovery of Sparse Signals via Convex
```

[6] statweb.stanford.edu/~candes/l1magic

[7] www.mathworks.com/matlabcentral/fileexchange/41792-simple-compressed-sensing-example

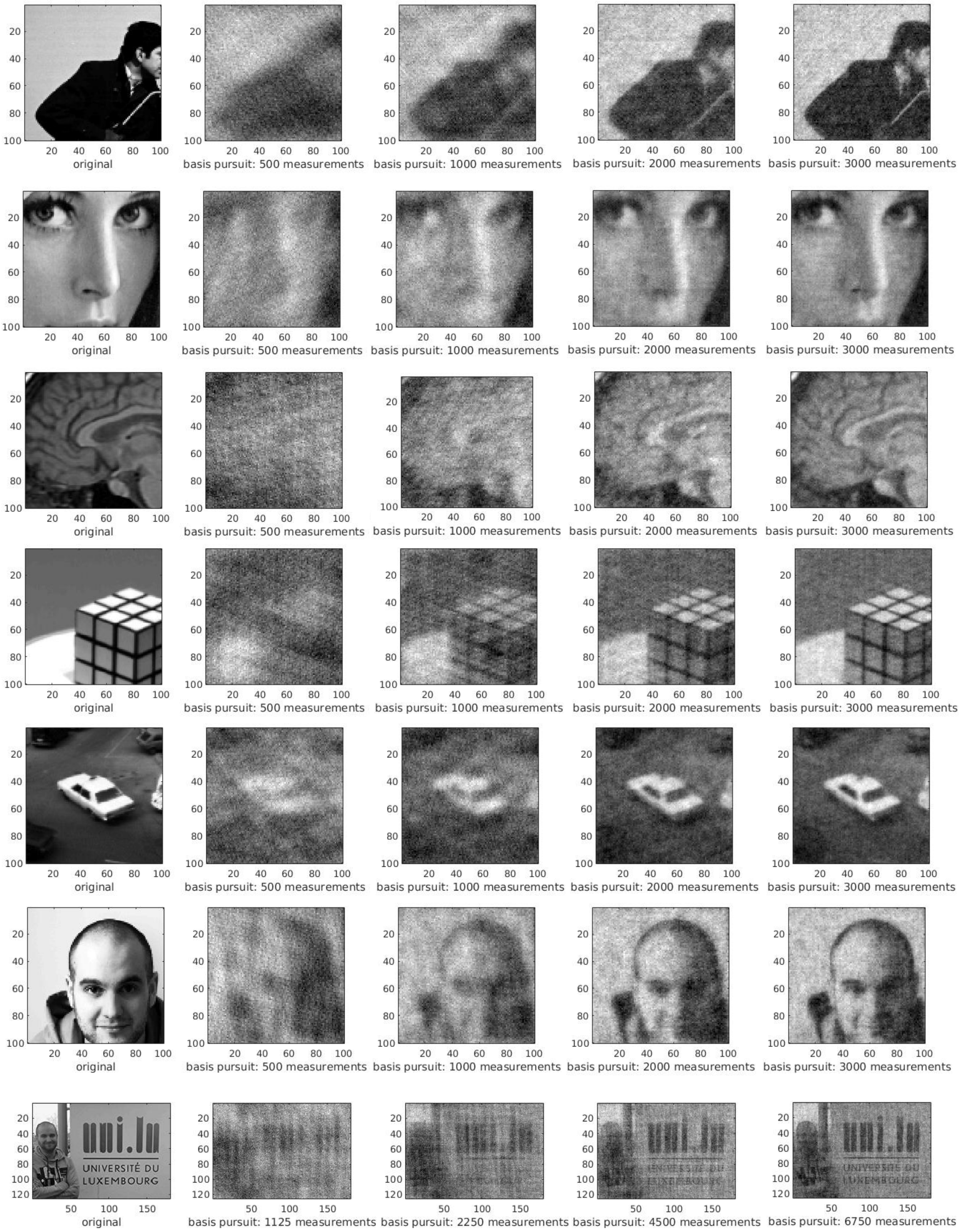
```
% Programming v1.11 by J. Candes and J. Romberg, Caltech, 2005.
%
%___VARIABLES___
% x = original signal (nx1)
% z = discrete cosine transformation of original signal
% y = compressed signal (mx1)
% Phi = measurement
% s = sparse coefficient vector (to be determined) (nx1)
%
%___PROBLEM___
% Invert the matrix equation y = Theta * s and therefore recover hat_x as
% k-sparse linear combination of basis functions contained in Phi. Note
% also that y = Phi * x.
%
%___SOLUTION___
% Let Phi be a matrix of i.i.d. Gaussian variables. Solve matrix inversion
% problem using basis pursuit (BP).

%include the l1-magic function we need
path(path, './Optimization');

%___INPUT IMAGE____________________________________________________
clear, close all, clc

% import image (important, use greyscale image)
A = imread('taxi.bmp');

% crop the image, will speed up algorithm instead of checking the whole
% image (the algo just takes too long for big pictures)
A = A([51:150],[51:150]);

% transform the matrix into a vector by taking one column after each other.
% cast values to double, necessary for matrix multiplication
x = double(A(:));

% apply the discrete cosine transformation to the vector
z = dct(x);

% get the length of the new vector
n = length(x);

%___MEASUREMENT MATRIX______________________________________________
% m defines the number of measurements to make
m = 3000;

% create a random matrix (gaussian)
Phi = randn(m,n);

%___COMPRESSION____________________________________________________
```

```
y = Phi*z;

%___BP SOLUTION__________________________________________________________
% L1-magic toolbox
% s2 = initial point
% Phi = matrix
% y = vector of observations
% 5e-3 = tolerance
% 20 = max number of iterations

% Solves
% min_s1 ||s1||_1  s.t.  Phi*s1 = y

s2 = Phi'*inv(Phi*Phi')*y;
s1 = l1eq_pd(s2,Phi,Phi',y,5e-3,20);

%___IMAGE RECONSTRUCTIONS_________________________________________________
% apply the inverse discrete cosine transformation to reconstruct image
x1 = idct(s1);

%___OUTPUT RESULT_________________________________________________________
% reshape transforms the vector back to a matrix
figure('name','Compressive sensing image reconstructions')
subplot(1,2,1), imagesc(reshape(x,100,100)), xlabel('original'), axis image
subplot(1,2,2), imagesc(reshape(x1,100,100)), xlabel(['basis pursuit: ' num2str(m)
colormap gray
```

## 3.3 Examples

Here are some examples, using a different amount of measurements. The images are cropped and have all a size of $100 \times 100$ pixels. Thus, the total amount of information is 10000 pixels, on which we take 500,1000,2000 and 3000 measurements respectively, which provide an available information of 5% up to 30%. Only for the last example, the image has a size of $125 \times 180$ pixels and thus the amount of measurements had to be adapted. I want to remark, that at each run, the matrix is reinitialized with new random variables and hence is not the same for a given pair of images. The images were taken from the general toolbox of compressed sensing of images [8]

[8] www.numerical-tours.com/matlab/sparsity_2_cs_images


original
basis pursuit: 500 measurements
basis pursuit: 1000 measurements
basis pursuit: 2000 measurements
basis pursuit: 3000 measurements
original
basis pursuit: 500 measurements
basis pursuit: 1000 measurements
basis pursuit: 2000 measurements
basis pursuit: 3000 measurements
original
basis pursuit: 500 measurements
basis pursuit: 1000 measurements
basis pursuit: 2000 measurements
basis pursuit: 3000 measurements
original
basis pursuit: 500 measurements
basis pursuit: 1000 measurements
basis pursuit: 2000 measurements
basis pursuit: 3000 measurements
original
basis pursuit: 500 measurements
basis pursuit: 1000 measurements
basis pursuit: 2000 measurements
basis pursuit: 3000 measurements
original
basis pursuit: 500 measurements
basis pursuit: 1000 measurements
basis pursuit: 2000 measurements
basis pursuit: 3000 measurements
uni.lu
UNIVERSITÉ DU
LUXEMBOURG
original
basis pursuit: 1125 measurements
basis pursuit: 2250 measurements
basis pursuit: 4500 measurements
basis pursuit: 6750 measurements

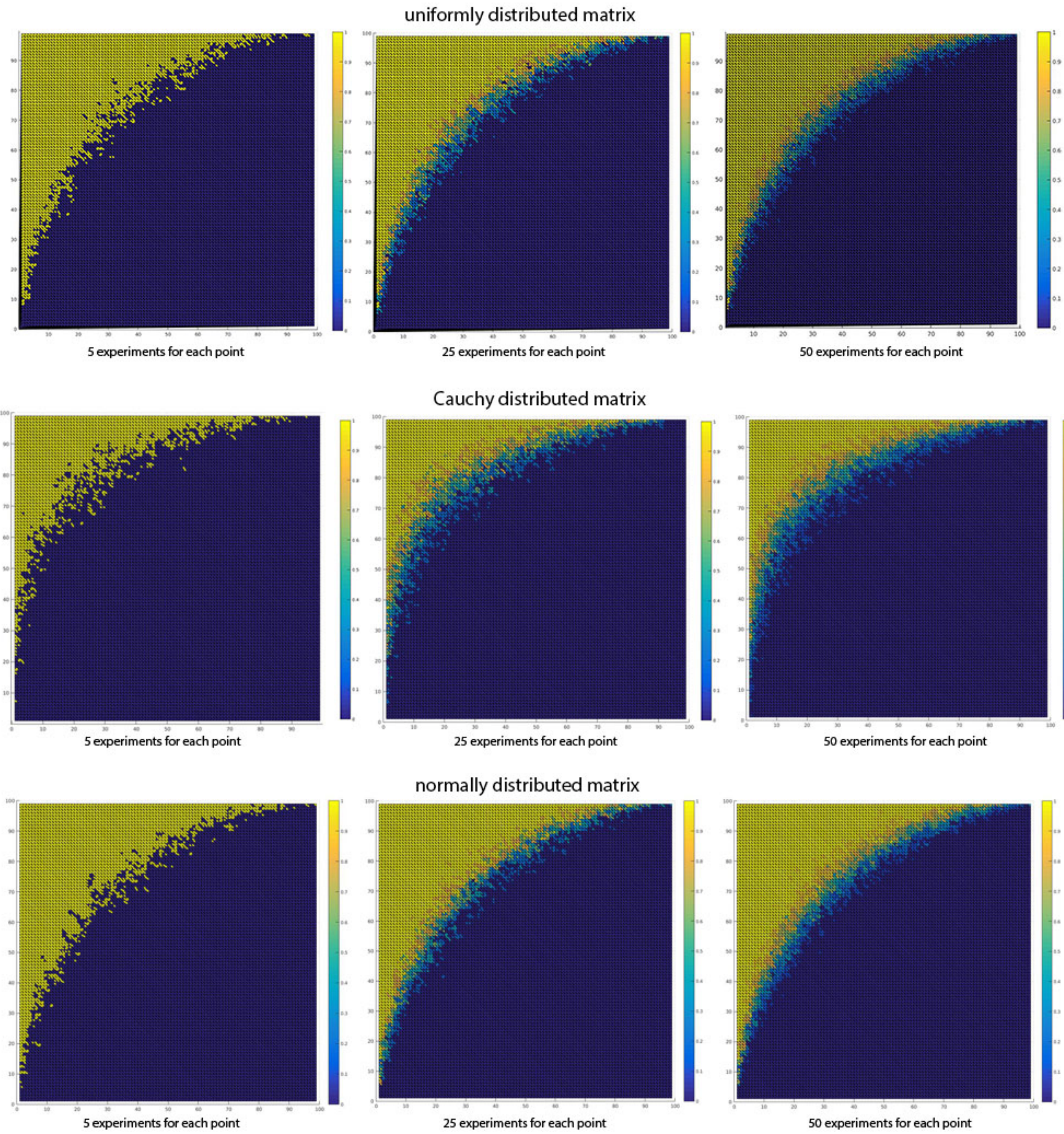
## 3.4 Implementation - Phase transition diagram

In this last section, I shortly introduce an algorithm on how to construct a diagram to illustrate the phase transition phenomenon in compressed sensing. For this, we create for each $k \in [1, N-1]$ a random vector, which is $k$-sparse. For each such vector, we try to reconstruct it after having applied a random matrix to it. This reconstruction experiment then is repeated a fixed number of times. The diagram then represents the probability of success to reconstruct a $k$-sparse vector by applying the $l_1$-minimization algorithm to it. You can find three examples after the implementation, where I used an uniformly, Cauchy and normally distributed matrix respectively.

```
%include the l1-magic function we need
path(path, './Optimization');

% preparation
clear, close all, clc

% number of experiments
% this number is multiplied by the number of tries of iterations of
% l1-magic
N = 1;
iterations = 5;

%dimension of ambient space
dim = 100;

% probability of success vector
% stores success rate to highlight in the diagram
Z=[];
lastPosition = 1;
for sparsity=1:(dim-1),
    for k=1:(dim-1),
        % count the number of successes for this sparsity level
        success=0;

        % number of measurements for each sparsity level
        for i=1:N

            % counter to get the right sparsity
            counter=1;
            % create vector
            randomVector = zeros(dim,1);

                % create a random vector with adapted sparsity to this
                % level
                while counter<sparsity+1,
```

```
                    % get random element of vector
                    x= randi(dim);
                    % if it is zero, then insert random -1 or 1
                    if randomVector(x)==0,
                        randomVector(x)=2*binornd(1,1/2)-1;
                        counter=counter+1;
                    end,
                end

            % random measurement matrix and result
            % uniformly distributed matrix
            %Phi = rand(k,dim);
            % gaussian matrix
            %Phi = randn(k,dim);
            % Cauchy matrix
            Phi = trnd(1,k,dim);

            b = Phi*randomVector;

            % initial point, necessary for l1-magic tool
            x0 = Phi'*inv(Phi*Phi')*b;

            % l1-optimization
            solution = l1eq_pd(x0,Phi,Phi',b,10^(-6),iterations);

            % check whether the optimization was "good"
            diffVector = solution-randomVector;
            diffNorm = norm(diffVector);

            % if l1-optimization was a success (arbitrary boundary)
            if diffNorm < 10^(-1),
                success = success+1;
            end
        end

        % store probability of success for this sparsity level
        Z(lastPosition) = (success/(N));
        lastPosition = lastPosition + 1;
    end
end


%  first coordinate matrix
X=[];
lastPosition = 1;
for i=1:(dim-1),
    for j=1:(dim-1),
        X(lastPosition) = i;
        lastPosition = lastPosition + 1;
```

```
    end
end

% second coordinate matrix
Y=[];
lastPosition = 1;
for i=1:(dim-1),
    for j=1:(dim-1),
        Y(lastPosition) = j;
        lastPosition = lastPosition + 1;
    end
end

% plot result
tri = delaunay(X,Y);
trisurf(tri,X,Y,Z);
```


uniformly distributed matrix
5 experiments for each point
25 experiments for each point
50 experiments for each point
Cauchy distributed matrix
5 experiments for each point
25 experiments for each point
50 experiments for each point
normally distributed matrix
5 experiments for each point
25 experiments for each point
50 experiments for each point

# References

[1] M. A. Davenport M. F. Duarte Y. C. Eldar and G. Kutyniok. Introduction to compressed sensing. *Cambridge University Press*, Compressed Sensing: Theory and Applications, 2012.

[2] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Birkhäuser, 2013.

[3] Gitta Kutyniok. Theory and applications of compressed sensing. 2012.

[4] Jin Li. Image compression: The mathematics of jpeg 2000. *MSRI Publications*, 46, 2003.

[5] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2), 1995.