Aalborg University
Semester Project
Signal Processing and Computing
group 870

Compressed Sensing Signal Reconstruction Determination using Phase Transitions

participant:
Konstantinos Voulgaris

supervisor:
Thomas Arildsen

28th of May 2015

AALBORG UNIVERSITET

STUDENTERRAPPORT

Department of Electronic systems
Fredrik Bajers Vej 7
9220 Aalborg E
Phone: 96 35 86 90
http://es.aau.dk

Title: Compressed Sensing Signal Reconstruction using Phase Transitions

Theme: Signal Processing and Computing

Project Period: Feb. 3rd - May. 28th 2015

Project Group: 15gr870

Participant:

Konstantinos Voulgaris

Supervisor:

Thomas Arildsen

Number of Copies:2
Pages:18
Number of Appendixes:1
Finished 28th of May 2015

Konstantinos Voulgaris

Preface

This study is produced and written by 2nd semester student from Signal Processing and Computing from group 870 at Aalborg University with Thomas Arildsen as supervisor. The paper is split up into worksheets each marked with a unique number, the purpose of the worksheets is to document different subjects which have been investigated during the project period.

- All figures and tables are provided with a caption and a unique number according to the worksheet.
- The respective number and caption are placed after the figure or table.
- All equations are referred to by number, but with no caption.
- Citations are presented by indices in square brackets, i.e. [*index*]. The bibliography is displayed after the last worksheet.

Abbreviations

CS	Compressed Sensing
SPMD	Single Progamme to Multiple Data
COP	Convex Optimization Problem
CPU	Central Processing Unit

Contents

1 Compressed Sensing

The conventional approach in the field of signal recovery, known as the *Nyquist–Shannon* sampling theorem, is based upon the theoretical foundation of *Shannon*, *Nyquist*, *Whittaker* and *Kotelnikov* [1]. The fundamental approach that relies upon the sampling theorem states the idea that a signal can be reconstructed correctly by taking twice of the highest frequency which is present in the signal(the so-called *Nyquist* rate). This particular approach though has a very significant drawback:in cases where the *Nyquist* rate is high , a significant amount of samples is required to recover the signal. In such case the application of *Nyquist–Shannon* sampling theorem is non appealing because a significant amount of data is needed to be stored or transmitted.

In the last few years a new framework has been developed in the field of signal acquisition: compressed sensing (CS)[2] aims to significantly reduce the number of samples required for a correct signal recovery and thus abate the storage requirements of such process. The basic principle upon the CS method is based, is the fact that natural images are sparse in the domain of the corresponding transform representation. The term sparse representation states the idea that a signal with a size of N can be described by a vector whose entries only have k non–zero values.

The recent work of *Donoho* and *Tanner* [1] showed that a signal with a sparse representation is possible to be recovered correctly by taking only a number of n (where $n < N$) measurements from the initial signal. These n measurements are obtained as a linear combination of the initial signal entries. In simple terms, CS introduces a new approach in the field of signal recovery where the number of samples which are required to reconstruct the signal is far below from those implied by the sampling theorem. This is a key difference between CS and the *sampling theorem* and this is the main reason why CS is more appealing compared to the conventional approach. Those two frameworks also differ on the procedures which are used in order to recover the signal. In CS the whole process is based on non–linear methods while on the other hand the *Nyquist–Shannon* approach aims to obtain the signal by making use of linear techniques which are more simple to compute and therefore easier to be interpreted [3] .

The set of the obtained n measurements can be represented in a vector form as y, where $y \in \mathbb{R}^n$. A crucial parameter when applying CS is to define the encoding strategy used to obtain y. Signals are transformed to a domain representation which takes the form of an orthogonal basis Ψ, the so called dictionaries. There are several situations where Ψ may not be practical for data compression or even unknown for the decoder. These cases indicate the need to design the encoding strategy with no regards to Ψ. Thus the

design of the $n \times N$ measurement matrix Φ, which is used to extract y from Ψ, must be incoherent with the structure of Ψ [2]. Hence the linear combination which is performed on x_0 in order to obtain y takes the form of the sensing matrix $A = \Phi\Psi$, where $A \in \mathbb{R}^{n \times N}$.

1.1 MATHEMATICAL FORMULATION

Once the main principles of CS are stated, the next step is to develop the mathematical pattern which performs the reconstruction of the initial signal x_0. A very common strategy in the field of signal recovery is to solve our problem as convex optimization problem(COP). The basic principle of CS is sparsity, thus the obtained solution ,which takes the form of a vector, needs to have as many zero entries as possible. Hence there is need for an objective function which fulfills this restriction.

The most sparse representation of the reconstructed signal $\hat{x}$ can be obtained by making use of the ℓ_0–norm which is a clear sparse indicator. There is one basic impracticality when attempting to minimize the ℓ_0–norm. The minimization of the ℓ_0–norm is not convex. In order to overcome this impracticality, convex relaxation is applied to the ℓ_0–norm. Therefore we make use of the ℓ_1–norm which is the best convex approximation of the ℓ_0–norm [4].

The implementation of CS aims to reconstruct x_0 by taking only a few measurements into account which in our case take the form of the vector y. Derived on this, the minimization of the ℓ_1–norm must yield a solution which is consistent with y. Therefore the system equation introduced in 1.1 is added as a constraint to the minimization problem.

$$y = A\hat{x} \tag{1.1}$$

Derived on this, the minimization problem takes finally the form which is introduced in Equation 1.2.

$$\begin{aligned} \underset{\hat{x} \in \mathbb{R}^{N \times 1}}{\text{minimize}} \quad & \|\hat{x}\|_1 \\ \text{subject to} \quad & y = A\hat{x} \end{aligned} \tag{1.2}$$

The last step of this pattern is to make the distinction between a successful and a failed attempt to reconstruct x_0. The reconstructed signal $\hat{x}$ is considered to be correctly recovered with respect to the normalized squared modeling error criterion if

$$\left(\frac{||x-\hat{x}||_2}{||x||_2} \right)^2 < \left(\textit{tolerance} \right)^2 \tag{1.3}$$

for $0 < (\textit{tolerance})^2 < 1$ [5].

1.2 Phase Transitions

CS introduce a simple way to sample signals that are already compressed, the reconstruction of the original signal from the compressed samples is a more complicated matter. Several algorithms have been developed to perform such task. In order to evaluate the performance of these algorithms there is the need for an objective measure. For this purpose the phase transition measure is introduced [6].

Phase transitions evaluate the probability of successful reconstruction versus the undersampling ratio and the sparsity of the signal. The term undersampling ratio is used to define the amount of samples that is used to perform signal recovery and it is denoted as $\delta = n/N$. Sparsity represents the number of non–zero elements of the original signal denoted, in relation to the number of measurements, as $\rho = k/n$. δ, ρ are the so called undersampling/sparsity coordinates ([6]) where $\delta, \rho \in [0, 1]^2$.

The probability of successful reconstruction varies over the different pairs of (δ, ρ). For the different instances of ρ the probability of correct reconstruction ,for the evaluated algorithm, decreases while ρ increases since it gets more difficult for the algorithm to reconstruct a signal when there are more non–zero coefficients per measurement. On the other hand, the probability for a successful signal acquisition increases while δ follows an upward trend.The reason why that happens has to do with the fact that more measurements are available for the algorithm thus the task of estimating the coefficients gets easier. Therefore different pairs of (δ, ρ) produce different probabilities of successful reconstruction. There are two basic phases: in the first phase the probability of success is essentially one while in the second phase the probability of success is essentially zero. The so far obtained results indicate that the transition from the one phase to the other is fairly sharp ([6]). Based on this observation the proposed model to locate the phase transition phenomenon makes use of the logistic sigmoid function introduced in section 2.1.

The phase transition phenomenon of a CS algorithm is located at the pair of (δ, ρ) for which the probability of a success is equal to the probability of a failure. In order to get a better idea about the phase transition phenomenon we can consider the sparsity coordinate as a function of δ , or else $\rho(\delta)$. In such case the phase transition phenomenon occurs at the highest possible instance of ρ for which $\rho(\delta) = 0.5$. Beyond that particular instance of ρ it is more likely for the algorithm to produce a failed recovery than a successful recovery of x_0.

In order to obtain a visual inspection of the phase transition phenomenon a simulation of phase transition is introduced in Figure 1.1. The phase transition curve shows, as a function of δ, the instance of ρ for which the probability of successful reconstruction is equal to the probability of a failure when the algorithm attempts to reconstruct the signal.In other words ,for the corresponding instance of ρ, the phase transition phenomenon behaves as a 50% probability boundary between the two basic phases. The corresponding algorithm is more likely to succeed than fail below the curve whereas the opposite holds above the curve. Therefore the desired trend of the curve is to be as close to $\rho = 1$ as possible.

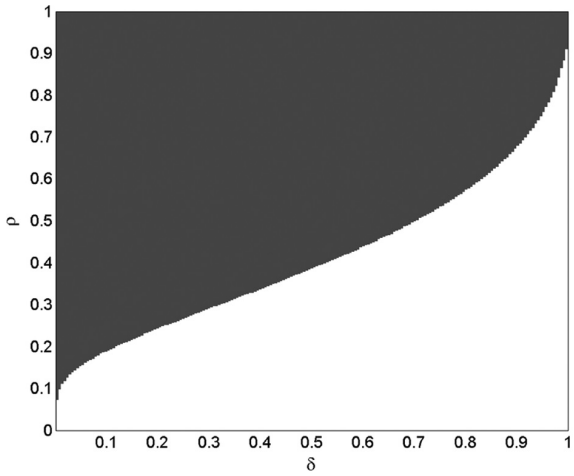


Figure 1.1. Simulation of a phase transition curve.

1.3 PROBLEM STATEMENT

Phase transitions are an objective measure which provide us the opportunity to evaluate the performance of a CS algorithm. Although the basic idea behind this standard measure is simple, the determination of phase transitions is a task which can be hardly performed by taking a theoretical approach. In order to fulfill such task there is the need of obtaining an empirical approach to derive the phase transitions of an algorithm. Therefore an extensive amount of computational experiments needs to be conducted. The main task of this project is to develop and implement a systematic framework to conduct the experiments.

There are several approaches which could be made to develop such framework. This particular framework though focus on designing various number of coefficients,sensing matrices and measurements. More specifically:

- **The coefficient vector**. Signals of different sizes are taking into account. For each of the corresponding vectors x the number of non-zero elements,κ, is assumed to be known and the entries of the coefficients are stacked to the support set Ω. The values for each x are randomly chosen from an ensemble of distributions, such as the Gaussian distribution, with respect to the location of the coefficients.
- **The sensing matrix**. There is a need for an encoding strategy in order to fulfill the incoherence principle, therefore the sensing matrix is not needed to be designed with respect to a particular dictionary. For this purpose each signal is acquired through an ensemble of sensing matrices.
- **The measurements**. The probability of success varies regarding the number of measurements which are taken into account. For this purpose the performance of the algorithms is measured for different pairs of (δ, ρ), both in the interval of $(0, 1]$ so that the phase transitions can be determined. For each instance of (δ, ρ), signals with the same level of sparsity but different coefficient values are tested.

1.4 PROJECT OVERVIEW

In accordance to the problem statement the goal of this project is to develop a systematic framework in order to determine the phase transitions of a CS algorithm. This effort aims to fulfill this task by making use of parallel computing in order to evaluate the performance of the given algorithm for individual pairs of (δ, ρ) since each point can be evaluated differently compared to the other. Finally the results will plotted in a 2D-plane to represent the performance of each algorithm.

2 Sigmoid fit on Logistic Regression model

In previous sections we have described the mathematical pattern (1.1) upon the CS algorithm attempt to recover the signal and also phase transitions (1.2) which is a standard measure to evaluate the performance of a CS algorithm. The current section aims to introduce a logistic regression approach that models the location where the phase transition phenomenon occurs.

The current approach introduced by *Donoho* and *Tanner* locates the phase transition phenomenon by utilizing algorithms from the field of combinatorial geometry the so–called regular polytopes [7]. For a signal with an infinite size of N, the corresponding phase transition curve, introduced in Figure 2.1 behaves as an upper bound for the particular approach.

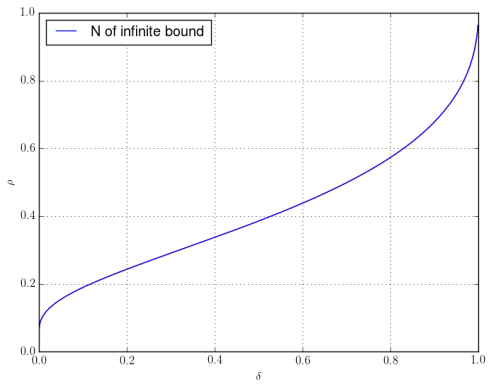


Figure 2.1. Phase transition of the cross–polytope for N of infinite size.

For a signal with a finite size of N the approach locates the point where the phase transition occurs by making use of an empirically derived formula which calculates the least amount of measurements needed to obtain the desired result. This formula is introduced in Equation 2.1, where $\rho(\delta, Q)$ is the function of the corresponding polytope. In our case the regular polytope that utilizes the same mathematical pattern to reconstruct the signal like the one introduced in 1.1 is the so–called cross–polytope [7]. The ϵ parameter corresponds to the fraction of success thus in our case $\epsilon = 0.5$.

$$n > \kappa/\rho(\delta, Q) \cdot (1 - R(\epsilon, n, N))^{-1}. \tag{2.1}$$

where $R(\epsilon, n, N) := 2[n^{-1} \log(4(N+2)^6)/\epsilon]^{1/2}$.

In our project we aim to introduce a modeled approach to conduct the analysis on the collected statistical data. The probability p for a correct reconstruction of the signal varies over the different pairs of (δ, ρ). In such case the collected data can be delivered as a three –dimensional mesh, (δ, ρ, p), which illustrates the relation between the different parameters of the reconstruction and the corresponding probability. The main focus of the model is to locate ,for each instance of δ, the corresponding instance of ρ where the phase transition phenomenon arises. Thus the reconstruction algorithm fulfills a given property for the particular pair of (δ, ρ): $p = 0.5$,where p can be consider as the threshold value.

A very common approach to model threshold phenomena in the field of statistical data analysis is to utilize logistic functions. A graphical representation of such function is introduced in Figure 2.2. Based on this approach a logistic regression model is introduced by *Donoho* and *Tsaig* ([8]) in order to locate the area where the phase transition phenomenon arises. The conducted analysis which was made on the obtained results showed that the phase transition tends to have a sigmoid fit over the different instances of the problem. For a small instance of δ the algorithm makes use of a small amount of measurements. Therefore when the number of non–zero coefficients increases the probability of success drops rapidly. As a result the corresponding curve shows a very sharp tendency. The opposite case holds while δ increases when more measurements are available thus the algorithm can reconstruct the signal with a higher probability. In such case the corresponding curve is less sharp.

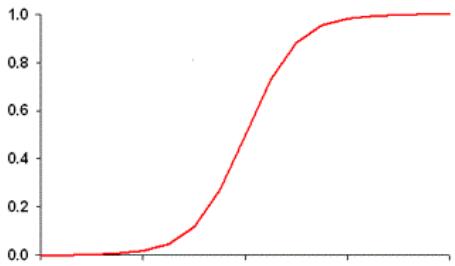


Figure 2.2. Graphical representation of a logistic function

2.1 Regression

In the previous section we provided the main reason why the tracking of the location where the phase transition phenomenon arises takes the form of a logistic regression model. The current section aims to provide the reader a detailed description about the structure of the corresponding model. A general tendency for the different instances of δ is demonstrated in figure 2.3. The main task of the model is to find the highest instance of ρ for which $\rho(\delta) = 0.5$. As soon as this task is fulfilled for all the instances of δ which are utilized by the reconstruction algorithm, we can derive the phase transition curve which provides an overall view about the performance of the algorithm.

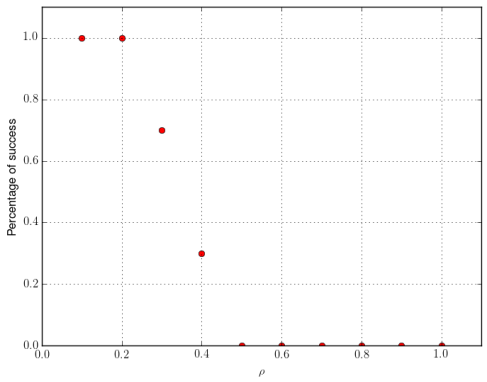


Figure 2.3. Tendency representation obtained by a random test of the framework for $N = 100$, $\delta = 0.3$.

The main task, when designing the phase transition curve, is to find for each instance of δ the corresponding instance of ρ where the probability of success is equal to the probability of failure. In other words we need to find the instance x_c of ρ which corresponds to the borderline between a successful attempt to reconstruct the signal and a failed attempt. In order to obtain the desired solution x_c there is the need to find the function which fits in our data. In the previous section 2 it was explained that the corresponding function takes a sigmoid form. Derived on this we are seeking for a function with the same tendency. Thus in Equation 2.2 a function is introduced which aims to fulfill that task. The basic parameter of this function is variable x_c which corresponds to the value of ρ where the phase transition phenomenon is located. The conducted results for each instance of δ vary over the different instances of ρ thus each δ has a unique threshold value x_c. s is a parameter which is related to the slope of the given function and has an impact to the sharpness of the phase transition curve. The variables x correspond to the instances of ρ which are taken into account when conducting the experiments.

$$Sigmoid(x, x_c, s) = \frac{1}{1 + \exp(-(x - x_c) \cdot s)} \tag{2.2}$$

As soon as the sigmoid function is defined, the next step in our model is to fit the given function to the obtained data. In such case there is the need to evaluate the performance of a possible solution for (x_c, s). Thus an error function is introduced in Equation 2.3 which returns the residual between the actual values and the predicted by the model set of values (x_c, s). The actual results are represented in a vector form by the variable y. The value of the error function varies over the different instances of ρ depending on the residual between the actual results y and the corresponding instance of ρ.

$$Error(y, x, x_c, s) = y - Sigmoid(x, x_c, s) \tag{2.3}$$

So far we have defined the form of the logistic function which is utilized by the corresponding model and also the way to evaluate if the model fits well the function to the collected data. The next step is to define the way which yields the optimal solution

for the model with respect to all the instances of ρ which are taken into account. For this purpose we follow the least squares approximation introduced in Equation 2.4 which aims to minimize the error function. The solution of the minimization problem yields the value x_c where the phase transition phenomenon occurs and also the corresponding value s.

$$\underset{x_c,s}{\text{minimize}} \sum_{i=1}^{l} (y_i - Sigmoid(x_i, x_c, s))^2 \tag{2.4}$$

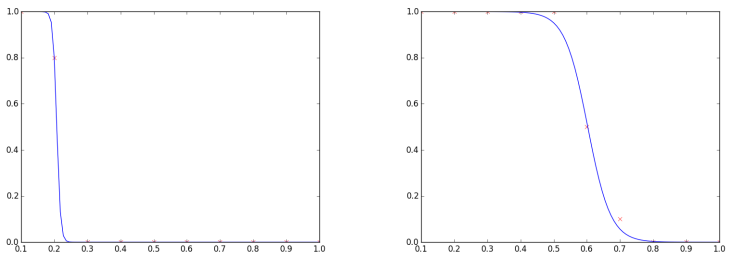
where l corresponds to the different samples of ρ per instance of δ.

The least squares problem introduced in 2.4 is solved in Python progamming environment by making use of the scipy.optimize.leastsq function. The main task of this function is to find the optimal solution (x_c,s) which minimizes the respective instance of δ. In general the minimum value of a function occurs when the first order derivative yields to zero. In our case though the optimal solution is approximated with respect to a set of parameters, thus the derivative of the *Error* function take the form of the Jacobian matrix introduced in 2.5. The Jacobian matrix is utilized by the algorithm upon the scipy.optimize.leastsq function is based in order to obtain the optimal solution of the minimization problem introduced in Equation 2.4.

$$\begin{bmatrix} \frac{s \cdot exp(-(x_1-x_c)\cdot s)}{(1+exp(-(x_1-x_c)\cdot s))^2} & \frac{(-x_1+x_c)exp(-(x_1-x_c)\cdot s)}{(1+exp(-(x_1-x_c))^2} \\ \vdots & \vdots \\ \frac{s \cdot exp(-(x_l-x_c)\cdot s)}{(1+exp(-(x_l-x_c)\cdot s))^2} & \frac{(-x_l+x_c)exp(-(x_l-x_c)\cdot s)}{(1+exp(-(x_l-x_c))^2} \end{bmatrix} = \begin{bmatrix} \frac{\partial Error(x_1,x_c,s)}{\partial x_c} & \frac{\partial Error(x_1,x_c,s)}{\partial s} \\ \vdots & \vdots \\ \frac{\partial Error(x_l,x_c,s)}{\partial x_c} & \frac{\partial Error(x_l,x_c,s)}{\partial s} \end{bmatrix} \tag{2.5}$$

2.2 Discussion

At this point the structure of the logistic regression model is complete. The next step is to verify that the particular design fulfills our expectations. For this purpose we conduct a simple experiment with respect to the reconstruction pattern introduced in 1.1, for a fixed size of $N = 100$. For a small instance of δ the obtained solution is expected to correspond to a function with a very sharp phase transition while the corresponding phase transition for an instance od δ which is close to 1 is expected to be less sharp. The corresponding results are illustrated in Figure 2.2 and they are consistent with our expectations.



(a) $\delta = 0.3$ (b) $\delta = 0.9$

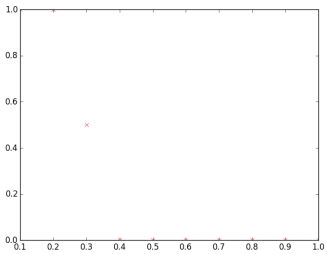
Figure 2.4. The red marks illustrate the percentage of success for each pair of (δ, ρ) with respect to the collected data.

2.2.1 Exceptional Case

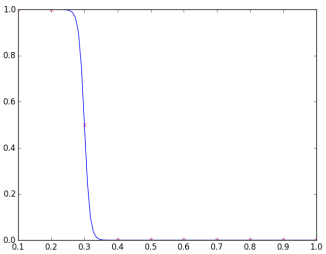
During the different simulations on the developed Logistic Regression Model it has been noticed that there is one case where the leastsq function cannot obtain a solution for δ. More particular when the probability of correct reconstruction of δ over the different instances of ρ drops from 1 to 0.5 and then to 0, like the one introduced in a row vector form in 2.6, the leastsq function yields *nan* (*not a number*). As a result the model cannot find a function that fits to the data. Such an example is illustrated in Figure 2.5(a).

$$A = [1, 1, 0.5, 0, 0, 0, 0, 0, 0, 0] \tag{2.6}$$

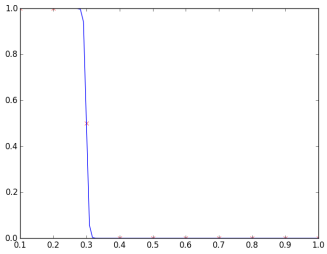
In order to investigate the reasons why that particular problem occurs we make some testings. First we make slight changes around the area of 0.5, the obtained results and the corresponding values are introduced in Figures 2.5(c) and 2.5(d). For the next testing instead of making changes in the area of 0.5, the changes are made around the area of the previous entry. The obtained result and the associated value are introduced in Figure 2.5(b). For all the different cases the Logistic Regression Model manages to find the optimal solution, and thus the corresponding function, successfully. Based on this observation we can claim that the Logistic Regression Model can successfully fit a function to the data. Therefore we claim that the problem is related with the solver and not with the model or the particular flow of results.



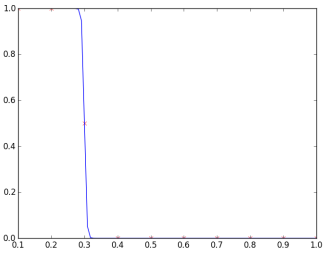
(a) entry $A(1,3) = 0.5$. The problem occurs. The marked points correspond to the percentage of success for each pair of (δ, ρ).



(b) entry $A(1,2) = 0.99999$. Optimal solution found.



(c) entry $A(1,3) = 0.5000001$. Optimal solution found.



(d) entry $A(1,3) = 0.49999999$. Optimal solution found.

Figure 2.5.

3 Software Implementation

In a previous section (1.3) we introduced the main task of this project: the development of systematic framework for determination of phase transitions for CS algorithms. In order to perform this task there is also the need of a software pattern to develop the framework. The structure of the corresponding software design is separated in two main stages:

1. Initial stage.
2. Stage of testing.

The first step when designing the framework is to define the size of the problem and the number of realizations which are performed in order to obtain the results. Based on these, the initial stage of the corresponding algorithm consists of two simple steps:

1. Fixing the size N of the coefficient vector x_0.
2. Definition of the number T of realizations.

Once the initial parameters of the framework are defined then the main part of the structure follows, where the tests are operated. The testing stage is separated into two parts:

1. Define the instance of (δ, ρ). The main task is to perform the realizations for all the possible combinations of δ, ρ by conducting tests across the different pairs of (δ, ρ). First the corresponding instance of δ is defined and then the corresponding instances of ρ are selected sequentially. As soon as the test design is terminated for the last instance of ρ the algorithm proceeds with the next instance of δ and the whole design is repeated until tests are performed for the last instance of (δ, ρ).
2. Execute the test design introduced in 3.1 with respect to (δ, ρ). At this part of the software design the algorithmic code follows the modern trend in supercomputing which aims to utilize multiple cores [9]. For this purpose the corresponding software design follow a computational aspect which introduces parallel computing.

3.1 Test Design

As soon as the instance of δ is defined, each core performs the same task. The corresponding algorithmic code is executed in a sequential pattern where the number of k elements varies in the range from 1 to n and in each step is updated with a step size equal to 1. In each execution of the algorithm the corresponding steps are performed with respect to δ and k. The basic instructions executed in each step of this design are:

1. Create the support set Ω by taking into account the corresponding number of non–zero elements k.

2. Generate a random vector x_0 with respect to Ω and the predefined size N.
3. Derive a random $N \times N$ orthogonal matrix Ψ.
4. Derive the $n \times N$ measurement matrix Φ.
5. Obtain the $n \times N$ sensing matrix A.
6. Compute y.
7. Solve the minimization problem.
8. Compare the original signal x_0 versus the reconstructed $\hat{x}$.

3.2 The multiprocessing part

The implementation of multiprocessing provides us the opportunity to fully exploit the potentials of the corresponding CPU. By taking into account that the algorithm performs a specified amount of T realizations for each pair of (δ, ρ), the next step of the algorithm aims to distribute the workload such that the whole process is simplified and the required time is reduced. For this purpose each of the realizations is assigned to a corresponding core.

The computational complexity of such task though varies regarding the number of measurements which are taken into account in order to reconstruct the signal. Each core requires different amount of time to execute the task for different instances of δ. In such case the cores are not busy for the same time and the hardware resources are not utilized effectively. Therefore each core is assigned with the same instance of δ so that they all work in a approximately synchronized pattern and the required time is similar. For each realization a new set of the parameters,x_0,Ω,A,Ψ,Φ,y is generated in order to reconstruct the initial signal. The generated parameters are then utilized to solve the minimization problem (1.2) and obtain the solution $\hat{x}$. As soon as the execution process is terminated for each core the results are called back to the main algorithm and they are gathered in lists. Based on these, it is clear that each core is assigned to execute the same instructions and the main difference is the data used in each test. In other words the whole process works as a Single Programme to Multiple Data case(SPMD) from the corresponding computer architecture classification([10]).

The algorithmic code is executed for T different instances of δ in a predefined range from the interval $[0, 1]$ with predefined updating step. Each time the process is terminated for all the cores, the results are called back to the main algorithm and they are gathered in lists. At the last stage of the main algorithm the collected data are evaluated by the second part of the framework in oder to determine the phase transitions of the CS algorithm.

3.3 Validation

The previous sections provide a detailed analysis upon the developed software. The next step is to validate that the corresponding algorithmic design is consistent with the requirements of the given problem. In the particular case the main task of the developed structure is to evaluate the performance of a CS algorithm, thus the obtained results

correspond to a fraction of success, where the number of successfully reconstructed signals $\hat{x}$ is compared with the total amount of signals x_0 that the CS algorithm attempts to reconstruct.

The performance of the CS algorithm varies over the different instances of the problem which in the current case corresponds to different pairs of (δ, ρ). The obtained results follow an upward trend while δ gets closer to 1 while the opposite scenario takes place while ρ converges to 1. For $\delta = 1$ and low values of ρ it is expected that the performance of the algorithm must be very high since the amount of measurements utilized by the CS algorithm is the highest possible and the amount of non –zero coefficients low, thus the reconstruction process is easier. On the other hand the corresponding fraction of success follows a decaying tendency while ρ is getting closer to 1 since the number of non –zero coefficients increases. Based on this expectation a validation test takes place on the developed algorithmic structure. The size of the given problem is $N = 100$, the undersampling ratio $\delta = 1$ and the performance of the algorithm is tested for 10 different instances of ρ in the range $[0, 1]$. The acquired results introduced in table 3.1 meet the expectations for the particular instance of the problem which is an indication of correct behavior for the instance of the problem that was tested.

1	1	1	1	1	1	0.9	0.6	0.6	0.3

Table 3.1. Results from a validation test for $N = 100$ and $\delta = 1$.

The basic part of the developed design that has to do with testing is executed in parallel. In such case there is always the scenario where the different processes generate precisely the same data with each other. Thus there is the need for a validation test to make sure that this case does not hold for the particular design. For this purpose a simple version of the original code is introduced for $N = 100$ and $T = 10$, where the pair of (δ, ρ) is predefined as $(0.1, 0.1)$. For each realization the algorithmic code generates a coefficient vector x_0 with only one non–zero value ($k = 1$). The obtained results can be seen in table 3.2. The first row of the table contains the value of each k while the second row corresponds to the entry of x_0 where k is located for each realization. Derived from the obtained results we can sum up that the are not dependent with each other. Thus the developed code is indicated to be functional for the particular instance of the problem.

-0.239	1.172	-0.313	1.942	-1.936	1.312	0.638	0.207	-0.934	-0.215
46	17	37	24	84	7	71	53	85	70

Table 3.2. N=100,δ=0.1,ρ=0.1. The first row of the table corresponds to the value of k while the second row of the table corresponds to the location of k in x_0.

4 Results

The previous chapters described the main methods and models used in the current project to extract the phase transitions of a CS algorithm. The current chapter aims to provide to the reader a demonstration of these methods. For this purpose a large scaled experiment is conducted in the environment of python programming language. For the simulation, we fix the size of the signal $N = 1000$. Then the developed algorithm attempts to reconstruct 400 points on the grid (δ, ρ). The grid is specified by $(\delta, \rho) \in \{0.05,0.1,...,1\}^2$. For each point in the grid, 50 different realizations take place with respect to the corresponding algorithm. The conducted experiment takes place into two different stages:

1. Obtain the probability of a successful reconstruction for each pair of (δ, ρ).
2. Extract the phase transitions of the reconstruction algorithm with respect to the obtained results.

At the initial stage of the experiments the algorithm ,with respect to the software structure introduced in chapter 3, attempts to reconstruct the signal. The obtained results are demonstrated in Table 4.1. The rows of the table correspond to the different instances of δ while the columns of the table correspond to the different instances of ρ.

0.64	0.46	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.86	0.9	0.22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.96	1	0.74	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.96	0.96	0.94	0.18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.96	1	1	0.66	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0.92	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0.62	0.02	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0.86	0.02	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0.94	0.08	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0.32	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0.92	0.22	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0.44	0.02	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0.96	0.2	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0.82	0.06	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0.48	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0.98	0.46	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	0.98	0.44	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	0.9	0.14	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1.	0.14	0	0	0	0

Table 4.1. Experimental results

As soon as the algorithm terminates with the assigned task the next step is to map the collected results to the Logistic Regression Model introduced in chapter 2.1. In order to evaluate the performance of the Logistic Regression Model, the extracted phase transitions of the algorithm are illustrated in comparison with the upper and lower bound for the given problem, stated by the work of *Donoho* and *Tanner*. A graphical representation of the comparison is given in Figure 4.1. The phase transitions for the lower bound have been extracted with respect to the given size of the problem.

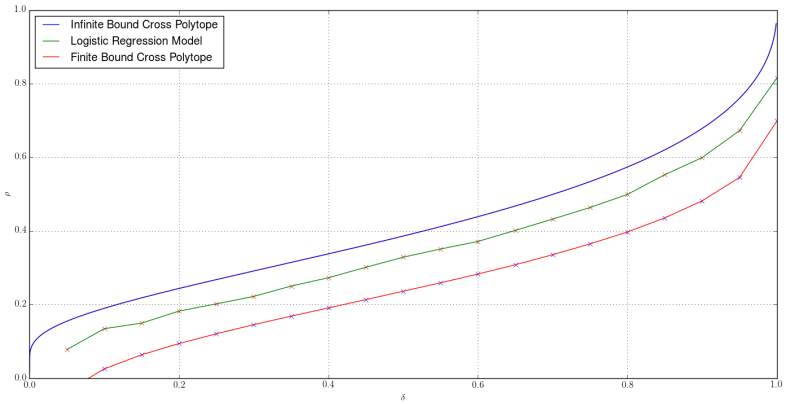


Figure 4.1. Graphical evaluation of the Logistic Regression Model. The marks on the phase transition curve correspond to the tested points.

The source for the tabulated values used to extract the phase transitions for the lower bound of the Cross–polytope is available at:
http://people.maths.ox.ac.uk/tanner/polytopes.shtml.

4.1 Discussion

The illustration of the three different phase transitions in Figure 4.1 places the one which corresponds to the Logistic Regression Model in between the upper and lower bound ,for the given problem, introduced by *Donoho* and *Tanner*. This particular observation fulfills the expectations we had in the beginning of the current project.

The main task of this project was to develop a model for extracting the phase transitions of a reconstruction algorithm with respect to the so far obtained results on the current field. The results indicate that the developed model provides a fairly good way to fulfill such task for a signal with a finite size.

The phase transitions obtained from the implementation of the Logistic Regression Model are placed above the lower bound for a signal with a finite size. In the same time the phase transitions of the Logistic Regression are placed below the theoretical upper bound for the given problem. Based on these observations we can claim that the Modeled approach achieves an improved phase transition of a CS algorithm thus the current project introduces an effective way to accomplish the assigned task.

5 Conclusion

Throughout the current project we developed a systematic framework to extract the phase transitions of a CA algorithm. The structure of the particular framework, consists of two main stages:

1. Signal reconstruction.
2. Phase transitions acquisition.

At the first stage of the structure we use as a basis to reconstruct the original signal x_0 l_1–norm minimization problem introduced in Equation 1.2. The second stage of the framework introduces a Logistic Regression Model as the approach to acquire the phase transitions of a CS algorithm which makes use of the l_1–norm minimization problem. The work of *Donoho* and *Tanner* introduced an upper and lower bound for the phase transitions of a CS algorithm which is based on the particular COP. Based on the results which are demonstrated in chapter 4, we can claim that we achieved an improved phase transition for an algorithm which reconstruct a signal with a finite size. Thus the main task of the current work is fulfilled.

In the field of signal acquisition there are several approaches which attempt to reconstruct a signal. A very common approach is to make use of a COP with respect to the parameters of the given problem. In our work we tested the developed Logistic Regression Model on one of the existed COP's in the field of CS. The work of *Donoho* and *Tanner* introduced an upper and a lower bound for two other COP approaches that exist in the field of CS [11]. It is expected that for those two cases the phase transition obtained by the Logistic Regression Model will be also placed in between those boundaries.

There can also be approaches where the signal is reconstructed based on different strategy rather than COP. In that particular case, there might not be an upper and lower bound to evaluate the phase transitions of the current Logistic Regression Model. It is still expected though, that the implementation of current approach will still give useful information about the performance of the corresponding CS algorithm.

The reason why we claim that the Logistic Regression Model is a good approach to evaluate the performance of a CS algorithm even if the reconstruction strategy is different that the one we follow, has to do with the fundamentals of the Logistic Regression Model. The current approach is structured with respect to particular properties of the phase transitions regardless the method of reconstruction.

Bibliography

[1] Donoho, David L. and Jared Tanner (2010), Precise Undersampling Theorems.In: *Proceedings of the IEEE 98.6*, pp. 913.

[2] Candés, Emmanuel J. and Michael B. Wakin (2008). An Introduction To Compressive Sampling, In: *IEEE Signal Processing Magazine* 25.2, pp. 21–30.

[3] Mark A. Davenport, Marco F. Duarte, Yonina C. Eldar and Gitta Kutyniok, Introduction to Compressed Sensing, pp. 3-4, Cambridge University Press, 2011.

[4] Carlos Ramirez, Vladik Kreinovich, Miguel Argaiz, Why l –1 is a good approximation to l –0: A geometrix Explanation., University of Texas El Paso, downloaded from: http://digitalcommons.utep.edu/gi/viewcontent.cgi?rticle=1754&context=cs_techrep.

[5] Bob L. Sturm, WHEN EXACT RECOVERY IS EXACT RECOVERY IN COMPRESSED SAMPLING .

[6] Donoho, David L. and Jared Tanner (2010), Precise Undersampling Theorems.In: *Proceedings of the IEEE 98.6*, pp. 915–916.

[7] Donoho, David L. and Jared Tanner (2010), Precise Undersampling Theorems.In: *Proceedings of the IEEE 98.6*, pp. 917–919.

[8] David L. Donoho and Yaakov Tsaig.Fast Solution of ℓ_1–norm Minimization Problems When the solution May Be Sparse,In: In: *IEEE TRANSACTIONS ON INFORMATION THEORY, VOL.54,NO.11NOVEMBER 2008* , pp. 4799–4800.

[9] Torben Larsen, Thomas Arildsen, Tobias Lindstrøm Jensen *Behavioral Simulation and Computing for Signal Processing Systems*,Aalborg University, pp 11 –12.

[10] Torben Larsen, Thomas Arildsen, Tobias Lindstrøm Jensen *Behavioral Simulation and Computing for Signal Processing Systems*,Aalborg University, pp 163 –164.

[11] Donoho, David L. and Jared Tanner (2010), Precise Undersampling Theorems.In: *Proceedings of the IEEE 98.6*, pp. 916–917.

A CD

The content on the CD includes:

1. The worksheets
2. Python scripts used during the project
3. HDF5 files containing the results