NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

GRADUATE PROGRAM

MASTER THESIS

ROBUST PRINCIPAL COMPONENT ANALYSIS: THEORETICAL ASPECTS AND ALGORITHMIC COMPARATIVE EVALUATION FOR DIMENSIONALITY REDUCTION

Michail N. Giannopoulos

***Advisors*:** **Sergios Theodoridis,** Professor
Yannis Kopsinis, Post Doc Researcher

ATHENS

November 2016

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΥΡΩΣΤΗ ΑΝΑΛΥΣΗ ΚΥΡΙΩΝ ΣΥΝΙΣΤΩΣΩΝ: ΘΕΩΡΗΤΙΚΕΣ ΠΤΥΧΕΣ ΚΑΙ ΑΛΓΟΡΙΘΜΙΚΗ ΣΥΓΚΡΙΤΙΚΗ ΑΠΟΤΙΜΗΣΗ ΓΙΑ ΜΕΙΩΣΗ ΤΗΣ ΔΙΑΣΤΑΣΗΣ

Μιχαήλ Ν. Γιαννόπουλος

***Επιβλέποντες*:** **Σέργιος Θεοδωρίδης,** Καθηγητής
Ιωάννης Κοψίνης, Μεταδιδακτορικός Ερευνητής

ΑΘΗΝΑ

Νοέμβριος 2016

MASTER THESIS

Robust Principal Component Analysis: Theoretical Aspects and Algorithmic Comparative Evaluation for Dimensionality Reduction

Michail N. Giannopoulos

S.N.: M1272

Advisors: **Sergios Theodoridis,** Professor
Yannis Kopsinis, Post Doc Researcher

November 2016

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εύρωστη Ανάλυση Κύριων Συνιστωσών: Θεωρητικές Πτυχές και Αλγοριθμική Συγκριτική Αποτίμηση για Μείωση της Διάστασης

Μιχαήλ Ν. Γιαννόπουλος

Α.Μ.: Μ1272

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Σέργιος Θεοδωρίδης,** Καθηγητής
Ιωάννης Κοψίνης, Μεταδιδακτορικός Ερευνητής

Νοέμβριος 2016

ABSTRACT

In the present master thesis we examine the question of whether the PCA method for dimensionality reduction could become robust vis-à-vis gross errors, and if so which algorithmic scheme from the literature would be the best choice.

In the beginning, we present the classical PCA method, its main ideas, those key properties that have made it so popular, its advantages and its disadvantages.

Afterwards, we state the main theoretical results concerning the possibility of robustyfying the PCA method, as well as some interesting applications of real life in which a robust PCA method could prove extremely useful.

Subsequently, a detailed presentation of the most popular algorithmic schemes designed to tackle this problem takes place, followed by a respective comparative analysis among them based on widely used quality metrics used in this scientific field.

Finally, a case-study inspired by the field of image processing is examined, in order on the one hand to evaluate the performance of the algorithmic schemes studied in the present thesis under tougher experimental circumstances, as well as on the other hand to examine their practical use in realistic applications.

SUBJECT AREA: Signal Processing, Statistics

KEYWORDS: Principal Component Analysis, sparsity, low-rank matrices, convex optimization, image processing

ΠΕΡΙΛΗΨΗ

Στην παρούσα διπλωματική εργασία εξετάζεται το κατά πόσο η ευρέως γνωστή ανάλυση κύριων συνιστωσών ως μια μέθοδος μείωσης της διάστασης μπορεί να καταστεί εύρωστη απέναντι σε ακραίες τιμές / παρατηρήσεις, και αν κάτι τέτοιο είναι δυνατό ποιο αλγοριθμικό σχήμα από τη βιβλιογραφία αποτελεί την καλύτερη επιλογή.

Αρχικά, παρουσιάζεται η κλασική ανάλυση κύριων συνιστωσών, οι βασικές της ιδέες, εκείνες οι ιδιότητες-κλειδιά της οι οποίες την έχουν καταστήσει τόσο δημοφιλή, τα πλεονεκτήματά της καθώς και τα μειονεκτήματα αυτής.

Στη συνέχεια, γίνεται μνεία στα βασικά θεωρητικά αποτελέσματα που αφορούν στην πιθανότητα η ανάλυση κύριων συνιστωσών να καταστεί εύρωστη απέναντι σε ακραίες τιμές, καθώς επίσης και σε μερικές ενδιαφέρουσες εφαρμογές της πραγματικής ζωής όπου κάτι τέτοιο θα ήταν αρκετά χρήσιμο.

Ακολούθως, λαμβάνει χώρα μια αναλυτική παρουσίαση των πιο διάσημων αλγοριθμικών σχημάτων που σχεδιάστηκαν ώστε να αντιμετωπίσουν αυτό το πρόβλημα, ακολουθούμενη από μία συγκριτική ανάλυση μεταξύ τους η οποία εδράζεται σε ευρέως χρησιμοποιούμενες μετρικές ποιότητας σε αυτό το επιστημονικό πεδίο.

Τέλος, εξετάζεται μια μελέτη-περίπτωσης προερχόμενη από το πεδίο της επεξεργασίας εικόνας, ώστε από τη μία πλευρά να αποτιμηθεί η επίδοση των υπο μελέτη αλγορίθμων σε “δυσκολότερες” πειραματικές συνθήκες, από την άλλη δε πλευρά να διερευνηθεί η πρακτική χρησιμότητά τους σε ρεαλιστικές εφαρμογές.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Επεξεργασία Σήματος, Στατιστική

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Ανάλυση κύριων συνιστωσών, αραιότητα, χαμηλής τάξης-πίνακες, κυρτή βελτιστοποίηση, επεξεργασία εικόνας

To my family

ΕΥΧΑΡΙΣΤΙΕΣ/AKNOWLEDGEMENTS

For the completion of the present thesis I would like to thank my advisors Professor Sergios Theodoridis and Post Doc Researcher Yannis Kopsinis for their cooperation, their advices, and their invaluable educational directions.

CONTENTS

FIGURES LIST

IMAGES LIST

TABLES LIST

INTRODUCTION

The present thesis was elaborated in the context of completion of the Graduate Program of the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens.

1. INTRODUCTION

Living in the outer world, human perception receives multiple incentives as well as information every day. In many cases, the data we receive from nature (e.g. the different colors we see with our eyes) are so vast that we find it at least difficult to process them in the way we desire to. Our mental representation of the world is although a little "subtractive", in the sense that it is based on a relatively small number of perceptually relevant features. As a result, methods for removing what in science is called "redundant information" while at the same time the remaining information retains an intuitive and handy structure, are of utmost importance these days.

What scientists -and in many cases all humans- do when they want to examine and explain a certain problem / phenomenon is observation. In other words, they take measurements. A very important question that arises automatically concerns the number of measurements / data points needed in order to obtain statistically sound and reliable results. Unfortunately, the answer to the previous question is neither simple nor encouraging. The reason is that as we live in the era of *Big Data*, the measurements we take in most cases also live in very high dimensional spaces -often in hundreds or thousands of dimensions. The result is that the required amount of data we need to support our result grows dramatically fast as the dimensionality of the problem increases. In a more formal and maybe explicative way, we can say that the time and memory needed for an algorithm to solve a particular problem is exponential in the number of dimensions of the data. This well-known phenomenon that appears in many different domains of science (such as statistics, machine learning, data mining etc.) is what we call the *Curse of Dimensionality*, and the first one referred to it is Richard E. Bellman when considering problems in dynamic optimization ([7], [6]).

When facing the curse of dimensionality, a good solution can often be found by changing the algorithm, or by pre-processing the data into a lower dimensional form. For example, the notion of *Intrinsic Dimension* (ID) refers to the fact that any low-dimensional data space can trivially be turned into a higher-dimensional space by adding redundant (e.g. duplicate) or randomized dimensions, and in turn many high-dimensional data sets can be reduced to lower dimensional data without significant information loss. This is also reflected by the effectiveness of dimension reduction methods -such as *Principal Component Analysis* (PCA)- in many situations.

The key feature to all previous considerations as methods facing the curse of dimensionality is the choice of working with data in a much lower dimensional space than the real one. In other words, our goal is to transform our original high dimensional set of measurements into a much lower one that we can easily process, but at the same time the relevant transformation has to be done in such a way that it removes the information redundancies while retaining most of the useful information of the original set of measurements. The process described below is what is called *Dimensionality Reduction*, and is widely used in machine learning as well as in statistics. More precisely and formally, dimensionality reduction is the process of reducing the number of random variables under consideration, and can be divided into *Feature Selection* and *Feature Extraction*.

While feature selection approaches try to find a subset of the original variables (also called features or attributes), feature extraction methods transform the data in the high-dimensional space to a space of fewer dimensions. The data transformation may be linear, as in PCA, but many nonlinear dimensionality reduction techniques also exist and are widely used.

2. PRINCIPAL COMPONENT ANALYSIS

As mentioned in the Introduction, principal component analysis is just one of many techniques available in our palette for achieving dimensionality reduction. It is widely used for this purpose, and it is not exceeding to say that it is the most popular among them. So, it is of crucial importance to take a deeper look at it, in order to understand the method itself as well as those key elements contributing to its popularity.

2.1 Motivation and statement of the problem

We start the conversation about PCA by presenting the motivation which led to it: in our attempts to understand various phenomena around us we take measurements of different quantities of interest. Unfortunately, many of the collected data may be redundant, contaminated with noise, and in general difficult to process due to their complicated structure. The goal of PCA is to “discover” the real underlying structure and probable relationship among the data, in such a way that most useful information is preserved using an intuitive method for achieving it at the same time.

It is time mathematics enter the game, explaining what until this moment may seem at least general. Suppose we have a set of measurements $\boldsymbol{x} \in R^m$, whose structure is unclear as mentioned before. What the PCA method does is creating a new set of data

$$\boldsymbol{y} = \boldsymbol{A}^T \boldsymbol{x} \textbf{ (2.1)}$$

, where $\boldsymbol{A}$ is the *Transformation Matrix* which exploits the statistical information describing the data. In terms of linear algebra, “the goal of principal component analysis is to identify the most meaningful basis to re-express a data set. The hope is that this new basis will filter out the noise and reveal hidden structure”, as mentioned very explicatively in [55].

2.2 Linear Algebra highlights the crucial details

At this point, let us go back to the introduction of this thesis, and mention again that PCA is a linear method for dimensionality reduction. This may seem a surplus information when looking at the method for the first time, but in reality it is not at all. The reason is not only the obvious adjective that discriminates it from the relevant nonlinear methods addressing the same problem, but most importantly that this adjective refers specifically to the transformation matrix $\boldsymbol{A}$ and its “properties”. Taking this argument a little further, the crucial part is the choice of the transformation matrix $\boldsymbol{A}$, or, in linear algebra words, the selection of the proper basis. So, now the question PCA method is trying to answer becomes finding a linear combination of the original basis which highlights most of the information of the original data set.

Before answering the previous question, it is deemed necessary to give a more intuitive interpretation of the equation **(2.1)** which transforms our original data set to a new one according to the PCA method. What this equation depicts is a change of basis, which in linear algebra can have the following interpretations:

- $\boldsymbol{A}$ is the matrix which is responsible for the transformation of $\boldsymbol{x}$ into $\boldsymbol{y}$
- The rows of $\boldsymbol{A}$ constitute the new set of basis vectors
- Geometrically, $\boldsymbol{A}$ is a rotation (and probably a following scaling) of the original basis vectors

At this point, it is necessary to note that at the equation **(2.1)** $\boldsymbol{x}$ and $\boldsymbol{y}$ can also be matrices ($\boldsymbol{X}$ and $\boldsymbol{Y}$ respectively) instead of vectors, without loss of generality. The relative extension / generalization is quite obvious, and the process of achieving it will be skipped.

2.3 A Signal Processing perspective

As everything in real world is imperfect, so are the measurements used for experimental purposes. There is quite an evolved theory around noise nowadays, concerning methods from predicting its distribution over the data to others of removing it. In this section we are not going to quantify the noise level of the data itself, but on the contrary we will confront it with respect to the "pure" data -those data which are not infected with noise. In signal processing and communication systems' theory, this constitutes a short introduction for making the reader familiar with what is called *Signal-to-Noise Ratio* (SNR). From a mathematical point of view, SNR is defined as:

$$SNR = \frac{\sigma_{signal}^2}{\sigma_{noise}^2} \quad \textbf{(2.2)}$$

As it is obvious from the definition, we desire $SNR > 1$ (measurements with low noise), and surely not $SNR < 1$ (measurements with high noise).

What PCA method is trying to do is finding those directions (in general, those structures) towards which the SNR is maximized. In other words, those directions towards which the *Variance* of the signal / data becomes maximum. When referring to directions, and taking into mind what was discussed in the previous section, we realize that in fact we are talking about a rotation of the *Cartesian Axis System*. Let's take a look at the following image, for further explanations:

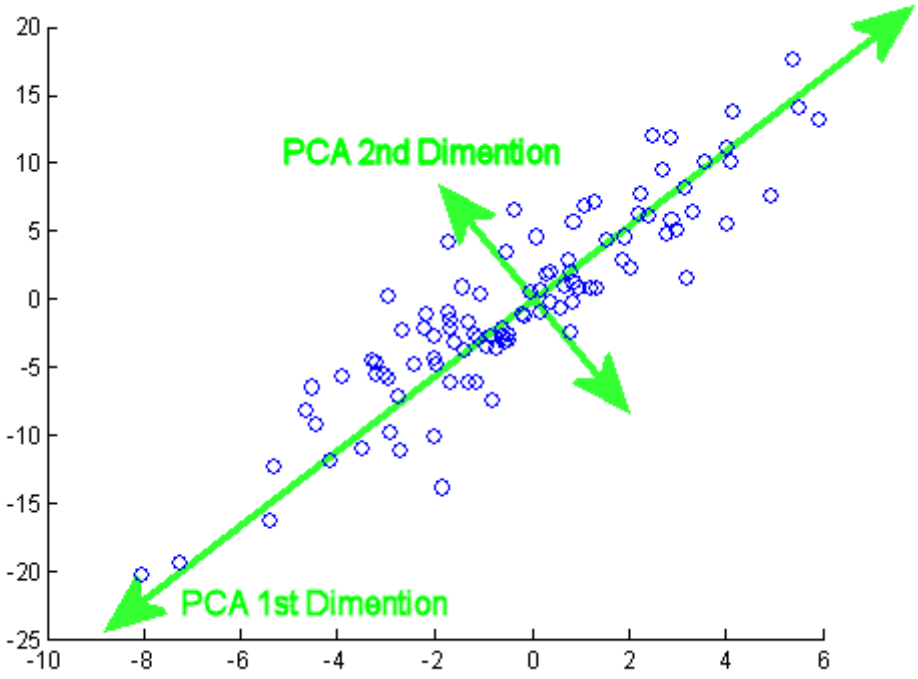

Image 1: Visualization of PCA method

The blue dots are our data, living in the two dimensional space. In other words, each data point is a vector with two coordinates (x, y). As stated before, the goal of the PCA method is to find along which direction(s) the greatest amount of variability of the data is found. As it is clear from Image 1, those directions are the green lines demonstrating exactly what we wanted. Most of our data are gathered around the "longer" green line (the first principal component), while the "shorter" green line (the second principal component) depicts the spread of the data around the first component. Trying to give an intuitive interpretation leads us to the conclusion that most of our data live exactly along the first principal component, while possible deviations from it implies that there is an amount of

noise at them. In effect, what happened is a rotation of the original axis system in the direction in which maximum variance of data is observed (firstly for the “pure” ones and secondly for the “infected”). Last but not least, we observe that the new axis (the principal components) are still orthogonal to each other, a fact that it is going to be explained later.

2.4 Statistics points out the “proper” matrix

As it is clear from equation **(2.1)**, what PCA method does is a transformation of the original data set to a new one, exploiting the statistical information hidden in the first. At this section we try to make this statistical asset a little clearer.

We know from statistics that for a given set of measurements $X = \{x_1, x_2, \dots, x_n\}$ with zero *Mean*, the variance is defined as:

$$\sigma_X^2 = \frac{\sum_{i=1}^{n} x_i^2}{n} \textbf{ (2.3)}$$

When referring to multivariate analysis, we introduce the notion of *Covariance*, which in turn measures the variance between two variables. In a more formal way, the covariance between two *Random Variables* $X = \{x_1, x_2, \dots, x_n\}$ and $Y = \{y_1, y_2, \dots, y_n\}$ is defined as:

$$\sigma_{XY}^2 = \frac{\sum_{i=1}^{n} x_i y_i}{n} \textbf{ (2.4)}$$

A first conclusion which comes from the above definition is that a high σ_{XY}^2 indicates highly correlated data, while a zero one indicates that our data are completely uncorrelated. That is exactly the key point of the PCA method from a statistical point of view: it creates new data which are (mutually) uncorrelated.

When working with vectors and generally matrices, we can easily define the respective quantities. The most important of them is the *Covariance Matrix*, which is defined as:

$$\boldsymbol{C}_X = \frac{\sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^T}{n} \textbf{ (2.5)}$$

, where $\boldsymbol{x}_i$ are the data vectors. Practically, $\boldsymbol{C}_X$ is an average of the data vectors $\boldsymbol{x}_i$. It is obvious from equation **(2.5)** and linear algebra basics ([56]) that $\boldsymbol{C}_X$ is a symmetric matrix and consequently its eigenvectors are mutually orthogonal. Concerning its elements, the diagonal terms of $\boldsymbol{C}_X$ constitute the variance of each data vector, while the off-diagonal ones constitute the covariance between them.

Given the above definitions and their practical meaning, we turn back to equation **(2.1)** in order to derive the covariance matrix of the new / transformed data. Supposed that $E[\boldsymbol{x}] = 0$, obviously $E[\boldsymbol{y}] = 0$. This may seem a tacit assumption at this point, but is not at all: in reality, when we examine zero-mean data the covariance matrix coincides with the *Correlation Matrix* (and as one entity we are going to treat them from now on in this Chapter, unless it is pointed out differently). Consequently, we have:

$$\boldsymbol{C}_Y = E[\boldsymbol{y}\boldsymbol{y}^T] = E[\boldsymbol{A}^T \boldsymbol{x}\boldsymbol{x}^T \boldsymbol{A}] = \boldsymbol{A}^T \boldsymbol{C}_X \boldsymbol{A} \textbf{ (2.6)}$$

At this point comes the crucial question: if we could manipulate matrix $\boldsymbol{A}$ as we want, what “form” do we desire matrix $\boldsymbol{C}_Y$ to have? Firstly, given the fact that our goal is to create uncorrelated data, $\boldsymbol{C}_Y$ must be diagonal (all of its off-diagonal elements should be zero). That’s a logical requirement, if we take in mind the fact that the off-diagonal terms of $\boldsymbol{C}_X$ in equation **(2.6)** depict the covariance between our measurements. Secondly, as we seek to find those directions / components in which the variance of the data becomes maximum one after another, the elements of $\boldsymbol{C}_Y$ must be sorted in descending order according to variance.

The easiest way of achieving the previous two goals in order to diagonalize matrix $\boldsymbol{C_Y}$ is to choose matrix $\boldsymbol{A}$ in such a way that its columns are the orthonormal *Eigenvectors* $\boldsymbol{a_i}$ of $\boldsymbol{C_X}$. The result is that the matrix $\boldsymbol{C_Y}$ will have the following form:

$$\boldsymbol{C_Y = A^T C_X A = \Lambda} \textbf{ (2.7)}$$

, where $\boldsymbol{\Lambda}$ is the diagonal matrix whose diagonal elements are the respective *Eigenvalues*, λ_i, of matrix $\boldsymbol{C_X}$. Given the fact that $\boldsymbol{C_X}$ is a positive-definite matrix (in reality, it is at least positive-semidefinite), it is known from linear algebra that its eigenvalues are all positive.

Given the above discussion, we finally give a more natural and intuitive meaning of the whole process: The eigenvectors $\boldsymbol{a_i}$ of matrix $\boldsymbol{C_X}$ are the principal components we seek to find, and each one of them corresponds to an eigenvalue λ_i. Those eigenvalues are all positive and sorted in descending order according to variance, a fact that depicts "how important" the respective principal component is for the "explanation" of as much information as possible. The orthogonality assumption of the principal components simplifies the solution due to handful linear algebra techniques available for achieving this goal, and indicates that from a linear algebra perspective what is really done is just a rotation of the Cartesian axis system.

At this point, we should mention that the PCA method was invented in 1901 by Karl Pearson ([49]), as an analogue of the *Principal Axis Theorem* in mechanics; it was later independently developed (and named) by Harold Hotelling ([37]) in the 1930s. It is also known in the field of signal processing as the *Karhunen-Loève Transform* (KLT, named after Kari Karhunen and Michel Loève), which in the theory of stochastic processes is a representation of a *Stochastic Process* as an infinite linear combination of orthogonal functions, analogous to a *Fourier Series* representation of a function on a bounded interval. In reality, PCA constitutes the discrete counterpart of the KLT, in the sense that it is the method followed by the latter when applied to a discrete and finite process. For more information about the KLT the reader is referred to [40].

2.5 The PCA Algorithm and some examples

The goal of this section is to present in a compact form the algorithm followed by the PCA method, and give a certain example that demonstrates its use.

The steps of the PCA algorithm are the following:

1. Estimation of the covariance matrix $\boldsymbol{C_X}$, usually by the equation **(2.5)**. If the data mean is not zero, we must subtract it first.
2. Computation of the eigenvalues, λ_i, and the corresponding eigenvectors, $\boldsymbol{a_i}$, of the covariance matrix $\boldsymbol{C_X}$.
3. Sorting of the eigenvalues λ_i in descending order (according to variance, as explained in the previous section).
4. Selection of the (for example) m largest eigenvalues, and in general of how many eigenvalues we think are important for not losing important part of information.
5. Utilization of the corresponding m eigenvectors as columns for the transformation matrix $\boldsymbol{A}$.
6. Transformation of every element / vector $\boldsymbol{x}$ from the original high dimensional space to the lower new one, via the equation **(2.1)**.

Below, it follows an example from [60] that depicts exactly how the PCA method works in practice:

We have a set $\boldsymbol{X}$ of N=500 two-dimensional vectors from a zero-mean *Gaussian Distribution* with covariance matrix

$$\boldsymbol{C}_X = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 1.0 \end{bmatrix}$$

By applying the PCA method as explained in detailed steps above, we are going to determine the eigenvalues λ_i and the corresponding eigenvectors $\boldsymbol{a}_i$ of $\boldsymbol{C}_X$ (in other words the principal components), as well as the percentage of the total variance "explained" by each one of them as the ratio $\frac{\lambda_i}{\lambda_0+\lambda_1}, i = 0, 1.$

After doing so (the corresponding Matlab code which forms and solves the example is available in [60]), the data points of the set $\boldsymbol{X}$ and the derived principal components are depicted below:

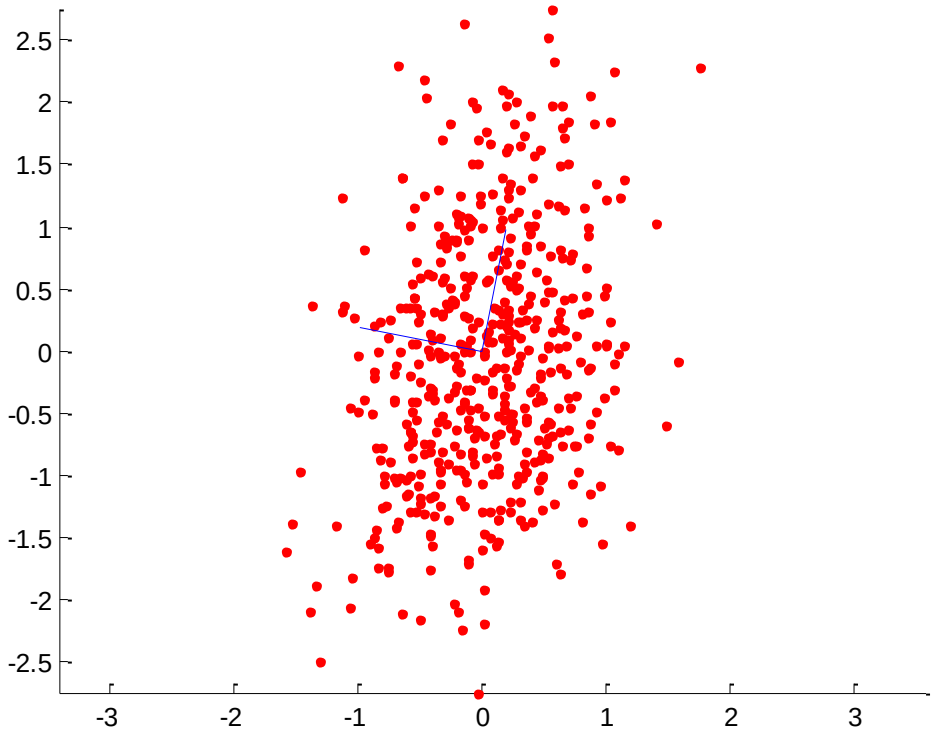

Figure 1: Data points and principal components of data set X

As it is obvious from Figure 1, the first principal component $\boldsymbol{\alpha}_0$ (the one that seems "more vertical" in the above figure) explains the most total variance of the data set $\boldsymbol{X}$. More precisely, after doing the required calculations, the percentage of the total variance explained by $\boldsymbol{\alpha}_0$ is 78.98% while that of $\boldsymbol{\alpha}_1$ is 21.02%. In a more intuitive interpretation of this result, we could say that if we project the data points of the data set $\boldsymbol{X}$ along $\boldsymbol{\alpha}_0$ we preserve 78.98% of their total variance -at the cost of losing at the same time the rest 21.02% associated with $\boldsymbol{\alpha}_1$.

2.6 Redundancy and Dimensionality Reduction

In section 2.3 a visualization of the manner in which the PCA method works was stated. More precisely, Image 1 depicted how the principal components are selected according to the maximization of variability of the data. This intuitive argument is lying in the heart of the dimensionality reduction process that takes place via the PCA method, a formidable

advantage that has made it so famous not only among statisticians but among the whole scientific community in general.

In order to be able to explain qualitatively how dimensionality reduction is reached, we first give a more quantitative argument based on what is called *Redundancy*. In a way, redundancy is related with the *Correlation* among the data. As we have seen so far, PCA is a method which generates uncorrelated data. From a statistician's point of view, uncorrelated data means data whose distribution is not "very strict", in the sense that are quite dispersed in the space they live in. In other words, redundancy is a regime in which our data have a high degree of correlation between them (the appearance of few data points helps us predict the rest of them). An example of data with high redundancy is seen in the following image:

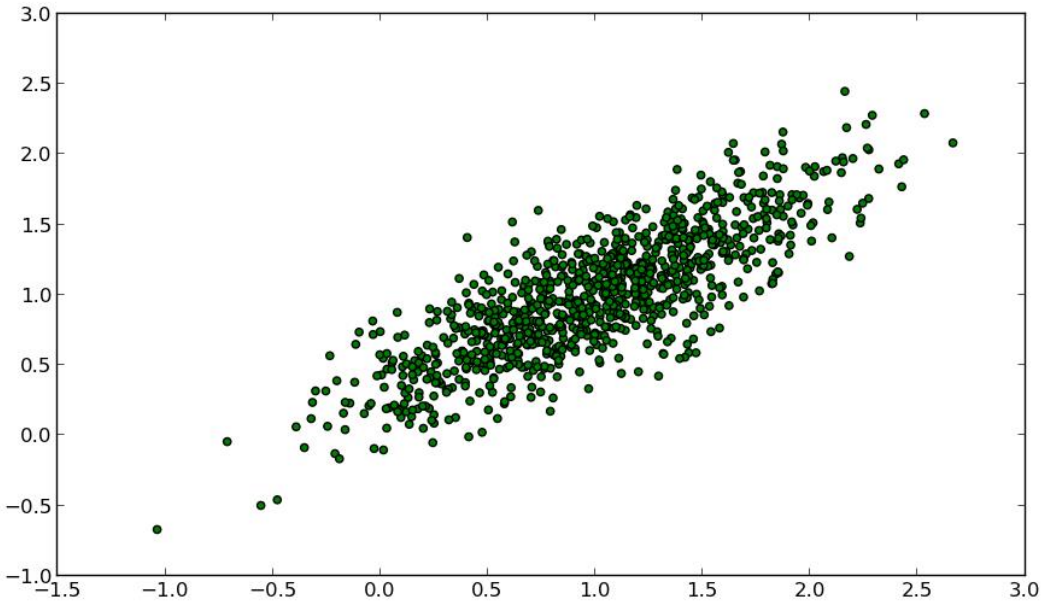

Image 2: Data with high redundancy

On the other hand, an example of data with low redundancy is depicted in the image below:

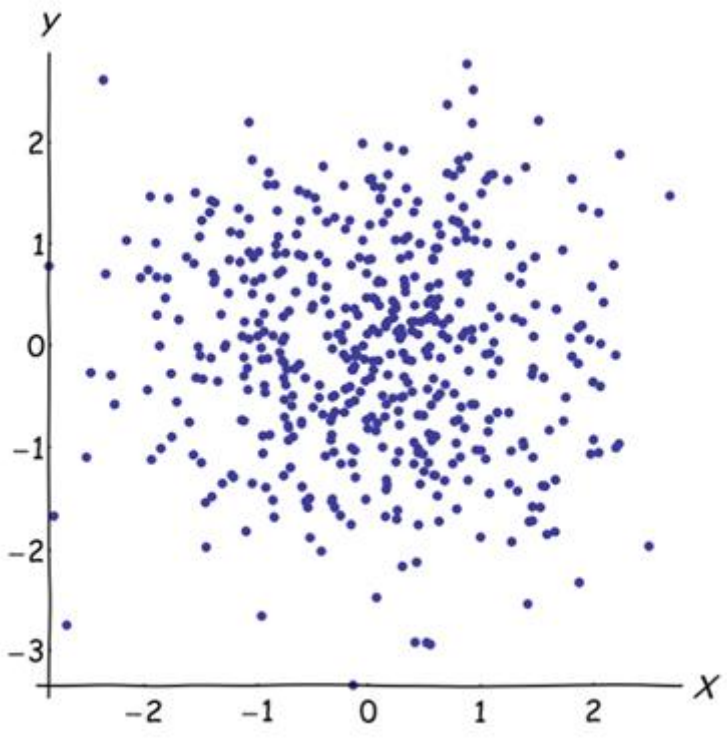

Image 3: Data with low redundancy

As it is clear from Image 3, our data points are quite uncorrelated, in the sense that one cannot predict $\boldsymbol{X}$ from $\boldsymbol{Y}$ (and vice versa). On the other hand, this is quite easier to be achieved with the data points of Image 2, using the best-fit line. In reality, due to the highly-correlated data points, only one dimension (a line) would be enough instead of two (a plane). This result is quite elegant, and at the same time is at perfect fit with the parsimonious nature of everything in our world, described very aptly by the famous *Occam's Razor Principle*: “Non sunt multiplicanda entia sine necessitate” (“Entities must not be multiplied beyond necessity”) as well as by the great Ancient Greek philosopher Aristotle himself: “Nature operates in the shortest way possible”.

Turning back now to the quantitative arguments of achieving dimensionality reduction via the PCA method, we just have to bring to our memory what the method really does: it’s a linear transformation of a high-dimensional space into a lower one, whose components are uncorrelated. In other words, the original data vector $\boldsymbol{x}$ lies in the n-dimensional space whereas the transformed one, $\boldsymbol{y}$, lies into an m-dimensional subspace of n (obviously $m < n$). That means that the intrinsic dimensionality of our data set is $m < n$, or more formally, that our data set can be described by m *Free Parameters*. In that case, there exist $n - m$ zero eigenvalues, and in practice we should ignore those eigenvalues with small values in order to get an approximation of the ID. An intuitive interpretation of the above inequality is that the intrinsic dimension is smaller than the “nominal” one exactly due to the correlation that exists among our original data set -which in turn leads us to the conclusion that geometrically they are concentrated throughout a hyperplane. That fact exploits the PCA method in order to reveal the dimension of that hyperplane across which our data are spread. In other words, PCA method achieves dimensionality reduction by revealing the number of free parameters that are responsible for the variability of a signal, namely the real information “coded” by the data.

2.7 Properties of PCA and general comments

In the final section of this Chapter, we have chosen to mention some well-known advantages of the PCA method, indicative of its popularity. Technical details are avoided, but for the diligent reader they are all available in [59].

First of all, we should mention the reason why the adjective “principal” is of crucial importance for the method. The reason lies in the *Mean Square Error* (MSE) approximation of the PCA method. In particular, our original data vector in the n-dimensional space is defined as:

$$\boldsymbol{x} = \sum_{i=0}^{n-1} y(i)\boldsymbol{a}_i \textbf{ (2.8)}$$

, while its approximation which takes into account only m ($m < n$) of the basis vectors (principal components) is:

$$\hat{\boldsymbol{x}} = \sum_{i=0}^{m-1} y(i)\boldsymbol{a}_i \textbf{ (2.9)}$$

If we try to approximate $\boldsymbol{x}$ by $\hat{\boldsymbol{x}}$, the resulting MSE is given by the following equation:

$$E[\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2] = E\left[\left\|\sum_{i=m}^{n-1} y(i)\boldsymbol{a}_i\right\|^2\right] \textbf{ (2.10)}$$

Obviously, our goal is to make the above MSE as minimum as possible. The key to achieve this is the proper selection of the eigenvectors. Taking into mind the definition of the eigenvectors as well as their orthonormal property, as it is explained in details in [59], we finally get:

$$E[\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2] = \sum_{i=m}^{n-1} \boldsymbol{a}_i^T \lambda_\iota \boldsymbol{\alpha}_\iota = \sum_{i=m}^{n-1} \lambda_i \textbf{ (2.11)}$$

As a result, if we choose in equation **(2.9)** those eigenvectors corresponding to the m largest eigenvalues of the covariance matrix, then the MSE in equation **(2.11)** becomes minimum, and more precisely is equal to the sum of the $n - m$ smallest eigenvalues. Furthermore, as it stated also in [59], this MSE is the minimum MSE regarding any other approximation of $\boldsymbol{x}$ by an m-dimensional vector. Taken all this into mind, the contribution of the adjective “principal” to the method now becomes a little clearer.

Last but not least, one should not forget to mention the property of the PCA method concerning the preservation of the maximum total variance of the original data. More precisely, given the equation **(2.1)**, we have:

$$\sigma_{y(i)}^2 = E[y(i)^2] = \lambda_i \textbf{ (2.12)}$$

In other words, the eigenvalues of the input covariance matrix are equal to the variances of the transformed data. Thus, the selection of those transformed data corresponding to the m largest eigenvalues, concludes to the maximization of their sum variance $\sum_i \lambda_i$. As a result, the m selected transformed data preserve most of the total variance of the original data. Indeed, as it is known from linear algebra basics ([56]), that variance is equal to the *Trace* of the covariance matrix $\boldsymbol{C}_X$. Finally, as it is clearly explained in [59], we should highlight the fact that the above property is more general, in the sense that it can be proven that among all possible data sets of m-dimensional vectors obtained via any orthogonal linear transformation of input data $\boldsymbol{x}$, the ones resulting from the PCA method exhibit the largest sum variance.

3. ROBUST PRINCIPAL COMPONENT ANALYSIS

In Chapter 2 a quite detailed presentation of the PCA method took place. As it became clear, PCA is arguably the most widely used -statistical- method nowadays in order to achieve dimensionality reduction. Most of the arguments that contribute to this effect were stated above, as well as a plethora of the benefits arising from its usage. But, as everything in real life, there has to be some disadvantages of the method as well. Nothing around us is perfect, and reasonably PCA couldn't be an exception to the rule. The question that immediately arises to the diligent reader is "*When does PCA fail?*"

3.1 Constructional limits of PCA

Although it may seem a bit weird, the main disadvantage of PCA occurs from a property it exhibits, which in many cases is desirable for any algorithmic scheme aiming to solve a particular problem. More precisely, we refer to the fact that PCA is a non-parametric algorithm: we feed it with data -of arbitrarily magnitude or structure- and the result that comes out has nothing to do with any tweaking of "cheating" parameters. Obviously, this can be considered as positive feature of the algorithm, in the sense that its results are objective and do not depend on the relative user. We can compare it in a way, with a "black box" which operates on an input in order to generate a certain output, and whose internal elements cannot be touched by the user who provides it the input (just as in *Linear Transform Invariant* (LTI) *Systems*' theory, except that in our case the topping of the box has faded a little as we are aware of its internal structure).

On the other hand, the feature described above can also be seen as a weakness, and in reality is the major one of the PCA method. To be more specific, we should be aware of what happens in a situation in which our data have been generated (on purpose or not) in such a way that puts the main argument / goal of PCA in jeopardy: the decorrelation of the data. In other words, given the fact that the goal of PCA is to de-correlate the data by tossing away second order dependencies among them, what happens when the distribution of the data indicates that in fact there exist higher-order dependencies among them? Unfortunately, the news are not good: the classical PCA method, by definition, will fail to reveal the real structure of the data. Nevertheless, many methods have been derived in order to alleviate and overcome such problems among the data distribution, such as *Kernel PCA* (KPCA) and *Independent Component Analysis* (ICA), with great success in many occasions where PCA fails.

3.2 PCA's "fatal" enemy: Outliers

In the previous section we saw that the PCA method faces an important difficulty when there is high-order dependency among the data to be processed, a difficulty however which can be surpassed by using some modifications of the classical PCA method. Given the fact that PCA is the most widely talked subject in applied multivariate statistics, all its' above benefits and drawbacks are more or less taught in every department or course of statistics around the world. Every undergraduate student probably thinks of the PCA method in a more general matrix frame, in such a way that our available data points constitute a data matrix $\boldsymbol{D}$, which is the superposition of a low-rank matrix $\boldsymbol{A_0}$ and a perturbation matrix $\boldsymbol{P_0}$:

$$\boldsymbol{D = A_0 + P_0} \quad \textbf{(3.1)}$$

When plotting these data points they could probably seem to live in a high dimensional space, but in reality they may be very well localized around a low-dimensional structure (the low-rank matrix $\boldsymbol{A_0}$ indicates exactly that there exists some kind of correlation among the data). In the general case, they may not be distributed exactly around a low-

dimensional structure but around an approximately one, that's why we should take into account a -small- perturbation (the perturbation matrix $\boldsymbol{P_0}$). So what is done when applying the PCA method in practice is that given a data matrix $\boldsymbol{D}$ whose columns are data points, we seek to find the best rank-k approximation matrix $\boldsymbol{A}$ (in an l^2 sense) to the given data matrix $\boldsymbol{D}$. Computationally speaking, we seek to solve the following optimization problem:

$$\min_{s.t.\ rank(A)\leq k} \|\boldsymbol{D}-\boldsymbol{A}\| \quad \textbf{(3.2)}$$

, where $\|\boldsymbol{D}\|$ denotes the l_2-norm, i.e. the largest singular value of $\boldsymbol{D}$. The solution to this problem is of course given via the well-known *Eckart-Young-Mirsky Theorem* ([27], [46]):

$$\boldsymbol{D}=\boldsymbol{U\Sigma V}^*=\sum_i \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^* \text{ --> } \boldsymbol{A}=\sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^* \ \textbf{(3.3)}$$

, where k is the rank of matrix $\boldsymbol{A}$, $\sigma_1, \sigma_2, \dots, \sigma_k$ are the positive *Singular Values* of matrix $\boldsymbol{A}$, and $\boldsymbol{U}=[\boldsymbol{u}_1, \boldsymbol{u}_2, \dots \boldsymbol{u}_k]$ and $\boldsymbol{V}=[\boldsymbol{v}_1, \boldsymbol{v}_2, \dots \boldsymbol{v}_k]$ are the matrices of left and right *Singular Vectors* respectively. Practically, what is done is a truncation of the *Singular Value Decomposition* (SVD) of the data matrix $\boldsymbol{D}$ by throwing away the smallest singular values, in such a way that the resulted approximation is as close as possible to the data matrix $\boldsymbol{D}$ (obviously achieving dimensionality reduction, via the trimming of the SVD).

What in contrary is not taught so much or is given less importance than deserved, is that PCA has an important drawback: it is extremely sensitive to what we call outliers (in statistics, an *Outlier* is an observation point that is distant from other observations -either due to variability in the measurement or because an experimental error occurred).

For example, let's suppose we have points in the plane lying around a one dimensional space as in the following image:

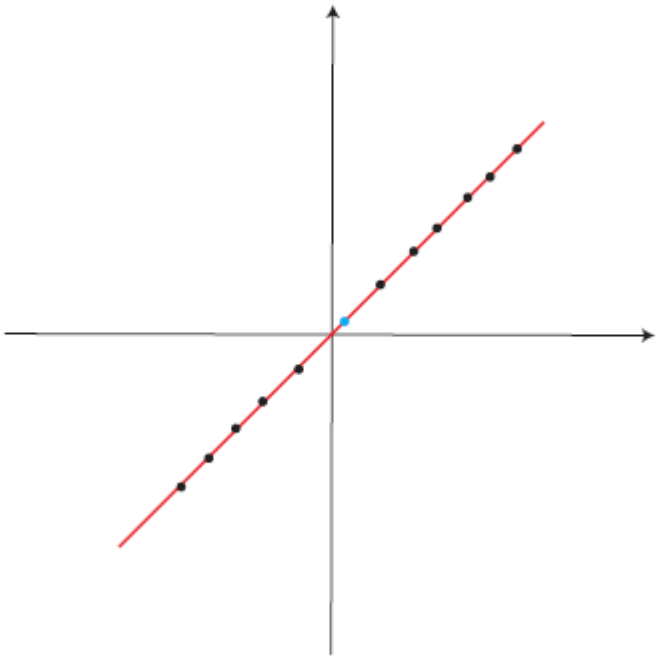

Image 4: PCA success

If we apply the PCA method, we are going to find that in reality our data points live around the red line in Image 4, which was successfully indicated by the PCA method.

Then, let's suppose that during the data collection there was one data point that was incorrectly recorded, for example due to a sensor failure, and our new dataset is like below:

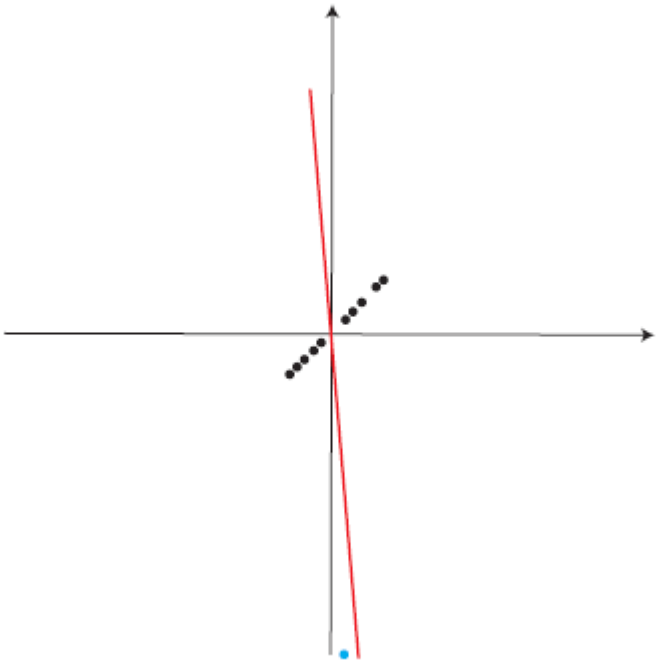

Image 5: PCA failure

In fact, what really happened in contrast to the previous situation, is that there is a significant change in the y-value of the blue data point. This data point for some reason is quite distant from all the rest, and can be treated as we saw previously as an outlier. The problem now is that if we apply the PCA method again as before, we are going to find the red line in Image 5, which has nothing to do with the line of first variation.

What is clear from this example is that one single -badly- corrupted entry of the data matrix is more than enough for things to completely break down. To make things even worse, we just mention that in daily life “incidents” like the one mentioned above are not the exception to the rule but they occur all the time. And the reason is that in this big data world we live in, with vast amounts of data being collected for various application purposes, we cannot assume that they are “clean”. On the contrary, arbitrarily corrupted measurements can come from an “innocent” sensor failure to human-driven malicious tampering. It is straightforward of paramount importance to overcome this fundamental obstacle, and make the PCA method robust vis-à-vis grossly corrupted observations.

3.3 Motivation and statement of the problem

In the previous section we became familiar with the Achilles’ heel of the PCA method: its sensitivity to the existence of outliers. So, it is clear that the motivation for dealing with this problem is that we want to be able to apply PCA in practice and large scale, but we cannot ignore the fact that a fraction of the entries in our data matrix may have been corrupted. To make things a little clearer, the problem is mathematically stated as below:

Suppose there is available a huge data matrix $\boldsymbol{D}$, which is a superposition of a low-rank matrix $\boldsymbol{A_0}$ and a sparse matrix $\boldsymbol{E_0}$ -both of arbitrary magnitude- as in the following equation:

$$\boldsymbol{D = A_0 + E_0} \quad \textbf{(3.4)}$$

We do not know neither the column or row space of $\boldsymbol{A_0}$ nor the location or the cardinality of the non-zero elements of $\boldsymbol{E_0}$. Is there any way of recovering the matrices $\boldsymbol{A_0}$ and $\boldsymbol{E_0}$?

If we would like to visualize the mathematical notations, we would have a blind deconvolution / separation problem as that in the following image:

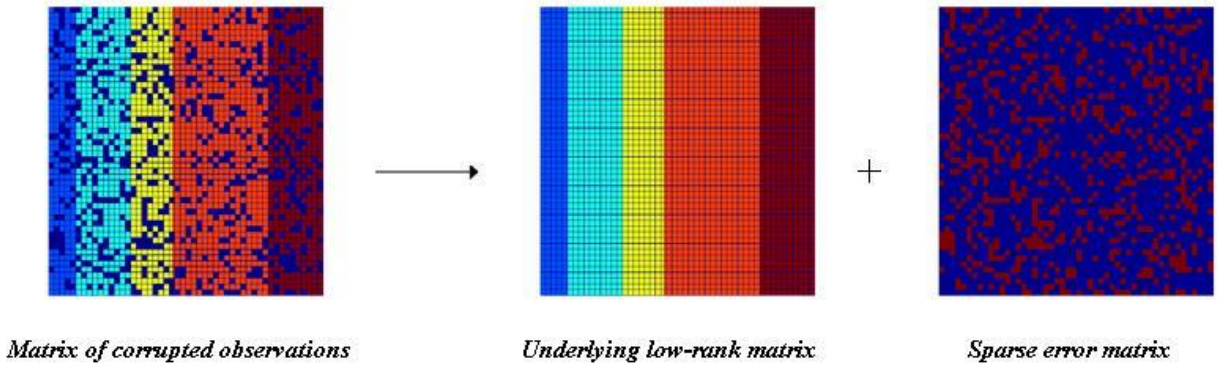

Image 6: The separation problem

The matrix on the left side of Image 6 is available to the statistician / machine learner, and it's a sum of a low-rank matrix and a sparse matrix. In other words, we have a low-rank matrix which is not observed because some of its entries are corrupted, and the corruption is exactly carried by the sparse matrix. The only thing we know is the sum of these two matrices, and our goal is to recover them accurately. Of course this may seem not to say impossible, but at least challenging.

3.4 Theoretical aspects of Robust Principal Component Analysis

Of course, the problem of robustifying PCA is not new. We just have to bring to mind from Chapter 1 that the PCA method was "invented" in the beginning of the previous century, to be persuaded that for sure many scientists would have faced it decades before. In order to give a clear and up-to-date description of the theory underlying this problem, we are going to highlight the results of the pioneering paper of Emmanuel J. Candès et al. in 2011 ([19]), due to two main reasons: the first one is that they consider an idealized version of the *Robust Principal Component Analysis* (RPCA) problem (which aims to recover a low-rank matrix $\boldsymbol{A_0}$ from highly corrupted measurements which satisfy the equation **(3.4)**), which can be seen as the most "universal and objective" of all the others; the second reason is that, as it is clearly stated in [19], none of the other existing approaches of solving the RPCA problem achieves that in polynomial time with guaranteed performance under broad conditions (a more detailed explanation is available in [19]).

3.4.1 Choosing the algorithm

In fact, the consideration of the RPCA problem in [19] is not only the most universal of all the others, but mainly the most intuitive one. The method used for the decomposition of the data matrix $\boldsymbol{D}$ is actually a tractable convex optimization technique, which is quoted below: Among all possible decompositions, we seek for a fit $\boldsymbol{A}$ and a fit $\boldsymbol{E}$ such as their sum is of course the observation matrix $\boldsymbol{D}$, but at the same time we desire that the final choice guarantees minimum complexity of the whole process. In other words, we seek for a matrix $\boldsymbol{A}$ which is as low-rank as possible and a matrix $\boldsymbol{E}$ which is as sparse as possible, whose sum gives the observation matrix $\boldsymbol{D}$. In mathematical words, we can formulate this intuitive idea as the following optimization problem:

$$\min_{s.t.\boldsymbol{A}+\boldsymbol{E}=\boldsymbol{D}} rank(\boldsymbol{A}) + \lambda\|\boldsymbol{E}\|_0, \lambda > 0 \quad \textbf{(3.5)}$$

, where $\|\boldsymbol{E}\|_0$ is the l_0-norm of matrix $\boldsymbol{E}$ -which is also known as the "counting norm" because in reality it represents the number of the non-zero elements of matrix $\boldsymbol{E}$.

What depicts equation **(3.5)** is what seems logical to do in order to obtain the best-fitting solution possible. But in reality, there is a huge computational problem concerning equation **(3.5)**, as minimizing the rank function under equality constraints is known to be a NP-hard problem -as well as that of minimizing the l_0-norm (for more details see Appendices I and II). What practically that means is that we can be able to solve a problem of this formulation when the dimension of the problem is for example 10 (i.e. matrices 10×10), but we cannot when it equals 11. The reason for that is that the best algorithms known to do that, have complexity double exponential in the dimension n, as explained in [15].

So what is suggested in [19], and is done into similar situations, is what is called convex relaxation. In other words, we choose to minimize a proxy for the objective function in **(3.5)**, and finally -as it is clearly explained in Appendix II- we choose the nuclear norm instead of the rank function and the l_1-norm instead of the l_0-norm. Taken this modification in mind, the equation **(3.5)** now becomes:

$$\min_{s.t.\boldsymbol{A}+\boldsymbol{E}=\boldsymbol{D}} \|\boldsymbol{A}\|_* + \lambda\|\boldsymbol{E}\|_1, \lambda > 0 \quad \textbf{(3.6)}$$

, where $\|\boldsymbol{A}\|_*$ is the nuclear norm of matrix $\boldsymbol{A}$ (the sum of its singular values) and $\|\boldsymbol{E}\|_1$ is the l_1-norm of matrix $\boldsymbol{E}$ (the sum of its absolute values, supposed we treat it as a gigantic vector).

The surprising as well as bizarre thing about equation **(3.6)**, which is also known as *Principal Component Pursuit* (PCP), is that it finds exactly the desired solution under broad conditions via algorithmic schemes whose complexity is not much higher than that of the classical PCA method.

3.4.2 "Appropriate" separations

As we noted before, equation **(3.6)** gives us the exact solution to our separation problem, but it does not achieve that always and under any circumstances. On the contrary, there has to be some restrictions. These restrictions obviously concern the observed data matrix $\boldsymbol{D}$, and more precisely its form.

Let's suppose for instance that we observe a data matrix $\boldsymbol{D}$ of the following form:

$$\boldsymbol{D} = \boldsymbol{e}_1\boldsymbol{e}_n^* = \begin{bmatrix} 0 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \quad \textbf{(3.7)}$$

If we are given such an observation matrix, anyone could argue that the low-rank component could be zero and the sparse one could be the matrix $\boldsymbol{D}$ itself, or vice-versa. Or that the low-rank component could be the "half" of matrix $\boldsymbol{D}$ and the sparse component the other "half", and of course they are all good solutions. So, in order for the problem to make sense, it is quite meaningful to demand that the low-rank component is not sparse. Making use of the incoherence parameter μ introduced in [15], we can quantify this demand as follows:

Given the SVD of the low-rank component:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^*, \quad \textbf{(3.8)}$$

, where r is the rank of matrix $\boldsymbol{A}$, $\sigma_1, \sigma_2, \dots, \sigma_r$ are the positive singular values of matrix $\boldsymbol{A}$, and $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \dots \boldsymbol{u}_r]$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \dots \boldsymbol{v}_r]$ are the matrices of left and right singular vectors respectively, we desire exactly those singular vectors (principal components) not to be sparse, otherwise we are in big trouble as it became clear above. The incoherence condition in [15] states that there exists a "coherence" parameter $\mu \geq 1$, which represents the degree of sparsity of the singular vectors, such that:

$$\begin{cases} \max_i \|\boldsymbol{U}^* \boldsymbol{e}_i\|^2 \leq \dfrac{\mu r}{n_1} \quad (\mathbf{3.9}) \\ \max_i \|\boldsymbol{V}^* \boldsymbol{e}_i\|^2 \leq \dfrac{\mu r}{n_2} \quad (\mathbf{3.10}) \\ \|\boldsymbol{U}\boldsymbol{V}^*\|_\infty \leq \sqrt{\dfrac{\mu r}{n_1 n_2}} \quad (\mathbf{3.11}) \end{cases}$$

, where $\|\boldsymbol{D}\|_\infty = \max_{i,j} |\boldsymbol{D}_{ij}|$ is the l_∞-norm of matrix $\boldsymbol{D}$ treated as a gigantic vector.

Although equations **(3.9)**-**(3.11)** may seem at least daunting, if we see their geometrical interpretation things surely become clearer:

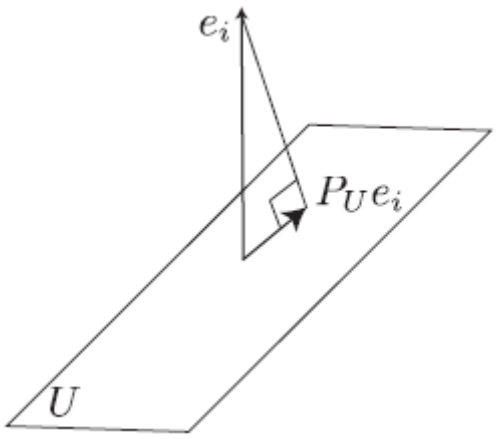

Image 7: Geometrical interpretation of the Incoherence Condition

Image 7 shows graphically the degree of correlation between the *Basis Vectors* and the *Column* / *Row Space* of our matrix. What is done intuitively is a projection of the basis vector $\boldsymbol{e}_i = (0,0,\dots 1,0,0,\dots 0)$ onto the column space U of matrix $\boldsymbol{A}$, followed by the calculation of the norm of this vector. If the column space is orthogonal to $\boldsymbol{e}_i$ then of course the norm is zero, whereas if the column space contains $\boldsymbol{e}_i$ the norm is one. The parameter μ exactly quantifies the degree with which the column space is aligned with the basis vector, being essentially the maximum norm of this projection normalized by the dimension of the space we are projecting onto (i.e. the rank) divided by the ambient dimension. The conclusion of equations **(3.9)**-**(3.10)** is that if μ is small then the column space and the row space of our matrix are not well-aligned with the coordinate axes. On the other hand, if μ is large then the column space and the row space of our matrix are well-aligned with the coordinate axes, and we face situations like equation **(3.7)** where μ is maximum as the column space contains $\boldsymbol{e}_n$. Conclusively, what we demand in reality is a small coherence parameter μ, which indicates that the singular vectors of the low-rank matrix $\boldsymbol{A}$ are not sparse.

Even if the above restriction seems enough for our problem to make sense and be solvable, in fact it is not at all. And the reason for that is quite simple: what we have managed to achieve so far is to have available a “nice” low-rank matrix whose singular vectors are not sparse, and what is left to do is just to corrupt it -in order to create the observed data matrix $\boldsymbol{D}$. The bad news is that it is possible to corrupt it in such a devilish way that the low-rank component cannot be recovered, and quite easily indeed: we just have to corrupt one entire column / row. For example, let’s suppose there is available a nice low-rank matrix $\boldsymbol{A}$ constructed in such a way to obey the above incoherence restriction, and we decide to create the sparse component in such a way as its first column

is the opposite of that of $\boldsymbol{A}$ and every other of its columns is zero. What is going to happen practically when we add them up in order to generate the data matrix $\boldsymbol{D}$, is seen below:

$$\boldsymbol{E} = \begin{bmatrix} * & 0 \cdots & 0 \\ \vdots & \ddots & \vdots \\ * & 0 \cdots & 0 \end{bmatrix} \rightarrow \boldsymbol{D} = \boldsymbol{A} + \boldsymbol{E} = \begin{bmatrix} 0 & * \cdots & * \\ \vdots & \ddots & \vdots \\ 0 & * \cdots & * \end{bmatrix} \textbf{(3.12)}$$

In doing such a corruption we did not increase the rank, in fact we decreased it, and the result is that there is no way of recovering $\boldsymbol{A}$ and $\boldsymbol{E}$ from $\boldsymbol{D}$ -since $\boldsymbol{D}$ has a column space included in that of $\boldsymbol{A}$. To avoid such situations, we demand that the entries to be corrupted are actually selected uniformly at random, in order that an entire corrupted column / row is met with very low probability.

Concluding, tying up together the two restrictions, we demand the low-rank component not to be sparse and the sparse component not to be low-rank and selected uniformly at random. If these two broad conditions are met in practice, then the PCP has a very elegant solution which we are going to expose right below.

3.4.3 Main results

In this section we just present the main result of [19], which is a theorem that states that the solution of PCP recovers exactly the low-rank component of $\boldsymbol{D}$ as well as the sparse one. We are not going to extend to technical details or proofs, because on the one hand they are all available in [19], and on the other hand such a thing would be beyond the scope of this thesis. Taken all this into mind, the well-known Theorem 1.1 of [19] states the following:

Theorem: Suppose $\boldsymbol{A}_0$ is $n \times n$, obeys **(3.9)-(3.10)**, and that the support set of $\boldsymbol{E}_0$ is uniformly distributed among all sets of cardinality m. Then there is a numerical constant c such that with probability at least $1 - cn^{-10}$ (over the choice of support of $\boldsymbol{E}_0$), PCP with $\lambda = \frac{1}{\sqrt{n}}$ is exact, i.e. $\boldsymbol{A} = \boldsymbol{A}_0$ and $\boldsymbol{E} = \boldsymbol{E}_0$, provided that:

$$\begin{cases} rank(\boldsymbol{A}_0) \leq \dfrac{\rho_r n}{\mu(\log n)^2} \ (\mathbf{3.13}) \\ m \leq \rho_s n^2 \ (\mathbf{3.14}) \end{cases}$$

, where ρ_r and ρ_s are numerical constants. In the general rectangular case where $\boldsymbol{A}_0$ is $n_1 \times n_2$, PCP with $\lambda = \frac{1}{\sqrt{\max\limits_{n_1,n_2}(n_1,n_2)}}$ succeeds with probability at least $1 - c \max\limits_{n_1,n_2}(n_1,n_2)^{-10}$, provided that:

$$\begin{cases} rank(\boldsymbol{A}_0) \leq \dfrac{\rho_r \min\limits_{n_1,n_2}(n_1,n_2)}{\mu\left(\log \max\limits_{n_1,n_2}(n_1,n_2)\right)^2} \\ m \leq \rho_s n_1 n_2 \end{cases}$$

The remarkable thing about this result is that it works under very broad conditions with very high probability. Of course it is a stochastic result, but the only probabilistic assumption in it concerns the locations of the non-zero elements of $\boldsymbol{E}_0$ (in such a way as to approach a “fair” regime for their selection).

Another feature of the main result in [19] that must be highlighted is that there is no tuning parameter in the algorithmic scheme. Such a thing may seem at least weird, in the sense that in similar situations usually takes place a cross-validation process in order to identify the best scalar λ to balance the two terms of the *Objective Function*. But, as explained in

details in [19], this particular choice of λ is universal and in fact the simplest one among a range of correct values emerging from the proof of the Theorem.

3.5 Applications

As we have seen so far, RPCA is a quite nice theoretical extension of the classical PCA in order to be able to work through situations in which gross errors (i.e. outliers) occur. But it would be a pity if all these elegant theoretical results wouldn't be applicable in real life. As expected although, obviously they have enormous impact in everyday applications, a couple of whom are indicatively listed below in this last section of the present Chapter:

Ranking and Collaborative Filtering: It concerns of course situations that arise from incomplete questionnaires, and the goal of course is to complete the missing entries. A classic example of this category of applications is the well-known *Netflix Prize* for movie ranking. In a short, Netflix is a company that rents movies in the United States of America. As every company of its size, it has a huge database of movies and users who have rent any of them. What Netflix did is sending an email to every user who rented a movie from their database, asking him to rate the movie he had just seen. And so users would go out and sparsely enter entries in huge database Netflix was assembling, which had as columns the company's movies (about 18.000 of them at the time) and as rows its clients (about 500.000 at the time). Of course this was a huge data matrix which was extraordinarily sparsely sampled because users on average rated about 30 to 50 movies, and so instead of having 18.000 ratings per row we had only 30 to 50. What simply Netflix wanted is to complete the database matrix, in other words the company was seeking for an algorithm that completes the missing users' ratings, and launched a prize of 1.000.000 dollars to whomever could be able to come up with a prediction algorithm that was besting their own algorithm by 10%. The reason is quite obvious, if we take in mind that the prediction of missing ratings implies an efficient recommender system, which in turn leads to happy customers and as a result a lot of income is generated. In fact, this problem is a typical *Matrix Completion* (MC) problem, but in reality we cannot overlook the fact that in this matrix there could be lots of bogus ratings (for some reason, people enter ratings which have nothing to do with their own preferences). So, in reality, we must separate the good ratings from the bogus ones, in order to be able to construct an efficient recommender system. Obviously, the good ratings will be held at a low-rank matrix and the bogus ones at a sparse one, making it clear that there is affinity with the RPCA problem we studied above.

Video Surveillance: In this type of application, there is available a sequence of video frames and the goal is to separate foreground from background. What is done practically is a storage of the images as columns of a huge data matrix, which we want to separate as before to a low-rank and a sparse component. The separation here makes sense in the way that the background is going to be an extremely low-rank matrix, as it does not change from frame to frame because it is highly correlated, while at the same time the foreground is going to be picked up by the sparse component.

Of course these were just two of many similar applications of the RPCA problem. What must be noticed is that any problem that requires a decomposition of a matrix to a low-rank and a sparse component can be recasted as a RPCA problem.

4. ALGORITHMIC METHODS FOR SOLVING THE RPCA PROBLEM

In the previous Chapter, a brief as well as comprehensive description of the main aspects (theoretical and practical) of the RPCA problem took place. Most of the analysis made was dominated by only one of the several approaches that have been developed to address the RPCA problem via low-rank plus sparse matrix decomposition, and more precisely the RPCA-PCP method ([19]). However, there are numerous other approaches as well, such as the following ones:

- RPCA via Outlier Pursuit ([70])
- RPCA via Iteratively Reweighted Least Squares ([33], [34], [35])
- Bayesian RPCA ([23])
- Variational RPCA ([1])
- Approximated RPCA ([73])

Although the aforementioned approaches are quite interesting, a detailed description of each one of them is out of the scope of this thesis. Following the reasoning of the previous Chapter, we chose to refer to algorithmic schemes designed to address the RPCA-PCP method as it was the leading one in this field.

Before presenting specific algorithms, it should be wise to mention the gap that they came to fulfil, in other words some weaknesses / limitations of the original RPCA-PCP method that became clear. More precisely, we should take into mind that the original approach described in [19] and [69] employs convex optimization techniques to address the PRCA problem, which under minimal assumptions recovers the low-rank matrix as well as the sparse one perfectly. Although this is a quite encouraging result, we should not forget that in fact it is a batch method which has consequently several limitations dealing with real-time applications. Furthermore, as each frame is treated as a separate column vector, any potential spatial or temporal features are lost. Finally, one of the main assumptions of [19] referring to the demand of the low-rank component to be exactly low-rank and the sparse one to be exactly sparse is quite often violated in many applications such as video surveillance, as noise affects every entry of the data matrix (for other minor / major limitations of the RPCA-PCP method concerning foreground detection in video surveillance the diligent reader is referred to [10]).

Taking into account the comments above, many methods have been developed to overcome the drawbacks arising from the nature of classical RPCA-PCP method itself as well as other ones which appear commonly in practice such as:

- Presence of noise
- Quantization of the pixels
- Spatial constraints of the foreground pixels
- Local variations in the background

The most well-known methods achieving those goals are presented in [74], [5], [58] and [68] respectively. The forthcoming table (presented in [10]) shows in details the different versions of the RPCA-PCP problem as well as the basic “technical concepts” of each one of them:

Table 1: Versions of the RPCA-PCP problem

Methods	Decomposition	Minimization	Constraints	Convexity

PCP[19]	$\boldsymbol{D = A + E}$	$\min_{\boldsymbol{A,E}} \|\boldsymbol{A}\|_* + \lambda\|\boldsymbol{E}\|_1$	$\boldsymbol{D - A - E = 0}$	Yes
SPCP[74]	$\boldsymbol{D = A + E + N}$	$\min_{\boldsymbol{A,E}} \|\boldsymbol{A}\|_* + \lambda\|\boldsymbol{E}\|_1$	$\|\boldsymbol{D - A - E}\|_F < \delta$	Yes
QPCP[5]	$\boldsymbol{D = A + E}$	$\min_{\boldsymbol{A,E}} \|\boldsymbol{A}\|_* + \lambda\|\boldsymbol{E}\|_1$	$\|\boldsymbol{D - A - E}\|_\infty < 0.5$	Yes
BPCP[58]	$\boldsymbol{D = A + E}$	$\min_{\boldsymbol{A,E}} \|\boldsymbol{A}\|_* + \kappa(1-\lambda)\|\boldsymbol{A}\|_{2,1} + \kappa\lambda\|\boldsymbol{E}\|_{2,1}$	$\boldsymbol{D - A - E = 0}$	Yes
LPCP[68]	$\boldsymbol{D = DU + E}$	$\min_{\boldsymbol{U,E}} \alpha\|\boldsymbol{U}\|_* + \beta\|\boldsymbol{U}\|_{2,1} + \beta\|\boldsymbol{E}\|_1$	$\boldsymbol{D - DU - E = 0}$	Yes

As mentioned before, in the present thesis we cope with the classical RPCA-PCP problem (theoretically as well as concerning algorithmic schemes). In this direction, just before presenting explicitly each one of the most basic algorithms addressing this problem, it is deemed appropriate to collect them together as to have a general and compact view of each one of them. This purpose is achieved via the following table (presented in [10]):

Table 2: Basic Algorithms for solving the RPCA-PCP problem

Solvers	Complexity
Singular Value Thresholding Algorithm (SVT) [12]	$O(mnmin(m,n))$
Accelerated Proximal Gradient Algorithm (APG) [43]	$O(mnmin(m,n))$ Full SVD
Dual Method (DM) [43]	$O(rmn)$ Partial SVD
Augmented Lagrange Multiplier Method (ALM) [42]	$O(mnmin(m,n))$
Alternating Direction Method (ADM) [72]	$O(mnmin(m,n))$

All the above algorithmic methods have one thing in common: they try to solve iteratively the same optimization problem described in general by the following form:

$$\min_{\boldsymbol{A,E}} \eta\|\boldsymbol{A}\|_* + \lambda\|\boldsymbol{E}\|_F^2 \quad \textbf{(4.1)}$$

, in the case that it has a closed-form solution-of course by employing different techniques. In such a scenario, the RPCA-PCP problem can be recasted as a *Semi-Definite Programming* (SDP) problem, and therefore be solved using off-the-shelf *Interior-Point Methods* and packages. The main problem with this consideration is that it exhibits prohibitive complexity as the scale of the problem increases. In fact, interior-point methods require computing the step-direction, whose complexity for a $m \times n$ matrix $\boldsymbol{D}$ is $O\left((mnmin(m,n))^2\right)$. In order to understand intuitively what that means we just have to consider the (simple) case of a square $n \times n$ matrix $\boldsymbol{D}$ which leads to a computational complexity of $O(n^6)$. Obviously, any interior-point method will be brought to its knees for even a relatively small number of data, i.e. n=100, leaving no hope about any thoughts of tens / hundreds of thousands or millions of data -which however is mostly the case in real applications.

To overcome (mainly) this scalability issue, the algorithmic schemes presented in Table 2 use different approaches deriving quite interesting results. In the following sections, each one of them is presented in brief focusing mainly on those key-ideas, implementation details, parameter tuning and -of course- the algorithms themselves that make them so extraordinary as well as worth-mentioning.

4.1 Singular Value Thresholding Algorithm

As became clear before, scalability is of utmost importance in real applications. Bearing that into mind, as well as that interior-point methods choke for even small number of data, it would be wise to use first-order information. In such a direction was the novel iterative algorithm introduced in [12], developed to minimize the nuclear norm for the MC problem. This paper was addressing the problem posed in [15], in which it was solved via the usage of an advanced SDP solver named SDPT3 ([62]). Given the fact that this solver uses interior-point methods, it suffers from the same scalability issues mentioned before -as for the computation of the Newton direction huge systems of linear equations needed to be solved. To make things worse, even iterative solvers (i.e. the *Method of Conjugate Gradients*) for the Newton-step do not save the day, since the *Condition Number* of the Newton system increases as we approach the solution.

4.1.1 Algorithm Outline

As mentioned earlier, the *Singular Value Thresholding* (SVT) *Algorithm* is a fist-order method designed to solve nuclear norm minimization problems of the general form:

$$\min_{s.t.\boldsymbol{A}(\boldsymbol{X})=\boldsymbol{b}} \|\boldsymbol{X}\|_* \quad \textbf{(4.2)}$$

, where $\boldsymbol{A}$ is a linear operator acting on the space of $n_1 \times n_2$ matrices and $\boldsymbol{b} \in R^m$.

The above optimization problem can be reformulated as follows:

$$\min_{s.t.P_{\Omega}(\boldsymbol{X})=P_{\Omega}(\boldsymbol{M})} \|\boldsymbol{X}\|_* \quad \textbf{(4.3)}$$

, where P_{Ω} is the orthogonal projector onto the span of matrices which vanish outside of Ω. In other words:

$$P_{\Omega}(\boldsymbol{X})_{ij} = \begin{cases} X_{ij,(i,j)\in\Omega} \\ 0, otherwise \end{cases} \quad \textbf{(4.4)}$$

The key-idea of the SVT algorithm is summarized in the following two operations, which take place until a specified stopping criterion is reached:

$$\begin{cases} \boldsymbol{X}^k = shrink(\boldsymbol{Y}^{k-1}, \tau) \\ \boldsymbol{Y}^k = \boldsymbol{Y}^{k-1} + \delta_k P_{\Omega}(\boldsymbol{M} - \boldsymbol{X}^k) \end{cases} \quad \textbf{(4.5)}$$

, where $\tau > 0$ and $\{\delta_k\}_{k\geq 1}$ is a sequence of scalar *Step-Size Parameters*.

In the first equation of **(4.5)**, $shrink(\boldsymbol{Y}^k, \tau)$ is a non-linear function which behaves as a soft-thresholding operator at level τ to the singular values of the input matrix. As it is expected from **(4.5)**, it constitutes the key building block of the whole algorithm, so it is deemed appropriate to explain exactly how it works according to [12]. More precisely, for a given matrix $\boldsymbol{X} \in R^{n_1 \times n_2}$ and its SVD $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the $n_1 \times r$ and $n_2 \times r$ matrices with orthonormal columns containing the left and right singular values of $\boldsymbol{X}$ respectively and $\boldsymbol{\Sigma}$ is the matrix which contains the singular values $\{\sigma_i\}_{1\leq i\leq r}$ at its diagonal, the soft-thresholding operator D_τ is defined as follows:

$$\begin{cases} D_\tau(\boldsymbol{X}) \coloneqq \boldsymbol{U} D_\tau(\boldsymbol{\Sigma}) \boldsymbol{V}^* \\ D_\tau(\boldsymbol{\Sigma}) = diag(\{\sigma_i - \tau\}_+), \tau \geq 0 \\ t_+ = max(0, t) \end{cases} \textbf{(4.6)}$$

As it is obvious from the above definition, the operator D_τ applies a soft-thresholding process to the singular values of $\boldsymbol{X}$, making it responsible for the name of the whole algorithm itself.

What should be highlighted here is on the one hand the fact that as the value of the parameter τ increases the sequence $\{\boldsymbol{X}^k\}$ converges to a solution similar to that of **(4.3)**, and on the other hand that in the case where many of the singular values of $\boldsymbol{X}$ are below the threshold τ then by construction the rank of $D_\tau(\boldsymbol{X})$ will be lower than that of $\boldsymbol{X}$. As a result, at each iteration step the only computational-thirsty operation to be done is a single one computation of an SVD -which in turn leads to proportionate computational as well as storage savings.

4.1.2 SVD Computation

As explained before, the SVT algorithm requires the computation of a SVD due to the soft-thresholding operation that takes place. In reality however, not all the singular values of the input matrix are needed but only that fraction of them which is above the threshold τ. Since the SVT algorithm was designed to cope with large matrices and the computation of their singular values and respective singular vectors, numerical linear algebra methods and relative already implemented packages could become extremely useful. For that reason as well as for some additional ones explained it details in [12], the inventors of the SVT algorithm chose to use PROPACK ([41]).

Concerning the SVD packages, as for most of them it is possible to specify the number of singular values to compute this is not the case with PROPACK at all. On the contrary, PROPACK cannot compute only those singular values which are above the threshold τ, a job that has to be done -intuitively- by the user. More precisely, what has to be done is the specification of the number s of the desired singular values in order that the software package computes the s-largest singular values and the corresponding singular vectors. As this process takes place in the k-th iteration of the algorithm, the respective number s_k of singular values of $\boldsymbol{Y}^{k-1}$ needs to be determined ahead of time. What is proposed in [12] as a choice for s_k is the following rule:

$$s_k = r_{k-1} + 1 \textbf{ (4.7)}$$

, where r_{k-1} is the number of the non-zero singular values of $\boldsymbol{X}^{k-1}$ at the previous iteration, i.e. in mathematical terms:

$$r_{k-1} = rank(\boldsymbol{X}^{k-1}) \textbf{ (4.8)}$$

The rationale of the choice in **(4.7)** is justified in [12] via the argument that in the case that some of the computed singular values are below τ then the choice in **(4.7)** is a good one, while if this not the case an extra increment of s_k has to take place until we fall into the first case. Such an increment may have either additive form:

$$s_{k+1} = s_k + l \textbf{ (4.9)}$$

, or even a multiplicative one:

$$s_{k+1} = l s_k \textbf{ (4.10)}$$

, where l is a predefined integer.

In [12], the additive form of increment is chosen with $l = 5$, a choice that seems to work pretty well in practice.

4.1.3 Step-Size Parameters

Another crucial part of the SVT algorithm that has to be clarified is that concerning the choice of the step-size parameter $\{\delta_k\}_{k\geq 1}$ involved in the second equation of **(4.5)**. For simplicity reasons, $\{\delta_k\}_{k\geq 1}$ is chosen to be constant and invariant of the iteration count:

$$\delta_k = \delta, k = 1,2, \dots \textbf{(4.11)}$$

As explained in details in [12], a wise choice would dictate to pick a step-size δ living in the interval $0 < \delta < 2$. However, for reasons of faster convergence, in [12] the step-size δ is eventually selected as follows:

$$\delta = 1.2\frac{n_1 n_2}{m} \textbf{(4.12)}$$

A heuristic reasoning for that choice is available in [12], but is at the same time out of the scope of this thesis and therefore will be skipped.

4.1.4 Initialization Steps

As it will become clear shortly afterwards, the SVT algorithm starts with $\boldsymbol{Y}^0 = \boldsymbol{0}$. Of course, our desire is to choose the threshold τ large enough to ensure that the solution provided by the algorithm is close enough to that we desire. In order to evaluate the second equation of **(4.5)**, as it is explained in [12], we define k_0 as that integer obeying the following property:

$$\frac{\tau}{\delta\|P_\Omega(\boldsymbol{M})\|_2} \in (k_0 - 1, k_0] \textbf{(4.13)}$$

Given the fact that $\boldsymbol{Y}^0 = \boldsymbol{0}$, it follows straightaway that:

$$\begin{cases} \boldsymbol{X}^k = \boldsymbol{0} \\ \boldsymbol{Y}^k = k\delta P_\Omega(\boldsymbol{M}), k = 1,2, \dots, k_0 \end{cases} \textbf{(4.14)}$$

Further computational burden can be removed adopting smart strategies from the *Compressed Sensing* / *Compressive Sampling* (CS) literature. For additional details, a more detailed sight is available in [12].

4.1.5 Stopping Criteria

Before introducing the SVT algorithm itself, we should strictly define its stopping criteria. In [12], two types of them are proposed:

- The first one comes from the *Karush-Kuhn-Tucker* (KKT) *Conditions* / *First-Order Optimality Conditions*
- The second one is motivated by duality theory

Here we only cope with the first one just for brevity reasons. More precisely, the proposed stopping criterion related to the KKT conditions is the above one:

$$\frac{\|P_\Omega(\boldsymbol{X}^k - \boldsymbol{M})\|_F}{\|P_\Omega(\boldsymbol{M})\|_F} \leq \varepsilon \textbf{(4.15)}$$

, where ε is a fixed tolerance (i.e. 10^{-4}). As it is justified in details in [12], the stopping criterion in **(4.15)** somehow "equivalent" to the following one:

$$\frac{\|\boldsymbol{X}^k - \boldsymbol{M}\|_F}{\|\boldsymbol{M}\|_F} \leq \varepsilon \textbf{(4.16)}$$

In other words, we control the *Relative Reconstruction Error* by controlling the relative error on the set of the sampled data.

4.1.6 SVT Algorithm

Finally, we conclude this section by presenting the SVT algorithm itself exactly as it was derived in [12]. We should not forget to mention here that this algorithm was designed to tackle the MC problem, not the RPCA one. Nevertheless, with minor modifications it can also handle the latter as well, with the main ideas remaining the same.

Before presenting the algorithm we should make clear that for each non-negative integer $s \leq min(n_1, n_2)$ the triplet $[\boldsymbol{U}^k, \boldsymbol{\Sigma}^k, \boldsymbol{V}^k]_s$ is composed as follows:

$$\begin{cases} \boldsymbol{U}^k = [\boldsymbol{u}_1^k, \boldsymbol{u}_2^k, \dots, \boldsymbol{u}_s^k] \\ \boldsymbol{V}^k = [\boldsymbol{v}_1^k, \boldsymbol{v}_2^k, \dots, \boldsymbol{v}_s^k] \\ \boldsymbol{\Sigma}^k = diag\left(\sigma_1^k, \sigma_2^k, \dots, \sigma_s^k\right) \end{cases} \quad \textbf{(4.17)}$$

In other words, it represents the first s left and right singular vectors as well as the first s singular values respectively.

After that note, the SVT algorithm is depicted in the following image:

Input: sampled set Ω and sampled entries $\mathcal{P}_\Omega(\boldsymbol{M})$, step size δ, tolerance ϵ, parameter τ, increment ℓ, and maximum iteration count $k_{\max}$
Output: $\boldsymbol{X}^{\text{opt}}$
Description: Recover a low-rank matrix $\boldsymbol{M}$ from a subset of sampled entries

Set $\boldsymbol{Y}^0 = k_0 \delta \mathcal{P}_\Omega(\boldsymbol{M})$ (k_0 is defined in (5.3))
Set $r_0 = 0$
for $k = 1$ **to** $k_{\max}$
  Set $s_k = r_{k-1} + 1$
  repeat
    Compute $[\boldsymbol{U}^{k-1}, \boldsymbol{\Sigma}^{k-1}, \boldsymbol{V}^{k-1}]_{s_k}$
    Set $s_k = s_k + \ell$
  until $\sigma_{s_k - \ell}^{k-1} \leq \tau$
  Set $r_k = \max\{j : \sigma_j^{k-1} > \tau\}$
  Set $\boldsymbol{X}^k = \sum_{j=1}^{r_k} (\sigma_j^{k-1} - \tau) \boldsymbol{u}_j^{k-1} \boldsymbol{v}_j^{k-1}$
  if $\|\mathcal{P}_\Omega(\boldsymbol{X}^k - \boldsymbol{M})\|_F / \|\mathcal{P}_\Omega \boldsymbol{M}\|_F \leq \epsilon$ **then break**
  Set $Y_{ij}^k = \begin{cases} 0 & \text{if } (i,j) \notin \Omega, \\ Y_{ij}^{k-1} + \delta(M_{ij} - X_{ij}^k) & \text{if } (i,j) \in \Omega \end{cases}$
end *for* k
Set $\boldsymbol{X}^{\text{opt}} = \boldsymbol{X}^k$

Image 8: The Singular Value Thresholding Algorithm

4.2 Accelerated Proximal Gradient Algorithm

As it became quite clear from the discussion above, new solvers other than the interior-point ones should be invented in order to tackle problems arising in many applications which involve matrices with dimensions tens of thousands. The main reason for that is that in reality interior-point methods rely on second-order information of the objective function. To overcome the scalability issue that arises from that choice, a good alternative is to use only first-order information -just as was the case with the *Iterative Thresholding* (IT) SVT algorithm analyzed in the previous section.

Iterative thresholding algorithms' usage does not come from the sky when referring to scalability issues arising in convex optimization problems. In fact, it is widely used in the field of CS as well as that of MC. Taken that into mind, an IT scheme was proposed in [69] exhibiting quite good results concerning scalability issues. The main problem with that algorithmic scheme however was that its convergence rate is extremely slow, typically requiring about 10000 iterations to converge (with each one of them having the

same cost as one SVD). As a result, even for matrix sizes of less than 1000×1000 several hours are required for the algorithm to converge.

To alleviate such an inconvenience, two new algorithms were proposed in [43] for solving the RPCA problem. The first one, which is presented here, is an *Accelerated Proximal Gradient* (APG) *Algorithm* based on the FISTA framework demonstrated in [4] -coupled with a fast continuation technique. The second one, which is presented in the upcoming section, is a *Gradient Ascend Algorithm* applied to the *Dual Problem* -a technique to which the whole algorithm owes its name.

4.2.1 General Formulation

The RPCA optimization problem can be recasted as an optimization problem of the general form:

$$\min_{s.t. \boldsymbol{AX}=\boldsymbol{b}} g(\boldsymbol{X}) \quad \textbf{(4.18)}$$

, where $g(.)$ is a continuous convex function, $\boldsymbol{b}$ is a vector of observations and $\boldsymbol{A}$ is a linear map. As it is explained in Appendix II, a commonly used tactic in practice when facing optimization problems is to relax them, and if we do so with **(4.18)** we get:

$$\min_{x \in H} F(\boldsymbol{X}) = \mu g(\boldsymbol{X}) + f(\boldsymbol{X}) \quad \textbf{(4.19)}$$

, where H is a real *Hilbert Space* equipped with a norm $\|.\|$ and

$$\begin{cases} \mu > 0 \\ f(\boldsymbol{X}) = \frac{\|\boldsymbol{AX}-\boldsymbol{b}\|^2}{2} \end{cases} \quad \textbf{(4.20)}$$

In other words, $f(\boldsymbol{X})$ depicts a kind of penalty for violations of the equality constraints while μ is a relaxation parameter which when approaching 0 any solution of **(4.19)** also approaches the solution set of **(4.18)**.

One of the main gains of the formulation **(4.19)** is that it can be optimized efficiently by *Proximal Gradient* (PG) *Algorithms* ([4], [65]) as it is explained in [43]. The reason for that is the "nature" of the penalty function $f(\boldsymbol{X})$, which is convex, smooth and has a Lipschitz continuous *Gradient*:

$$\|\nabla f(X_1) - \nabla f(X_2)\| \leq L_f \|X_1 - X_2\| \quad \textbf{(4.21)}$$

To achieve such an efficient performance, PG algorithms adopt an alternative objective function to be minimized instead of $F(\boldsymbol{X})$, whose definition is given below:

$$Q(\boldsymbol{X}, \boldsymbol{Y}) = f(\boldsymbol{Y}) + \langle \nabla f(\boldsymbol{Y}), \boldsymbol{X} - \boldsymbol{Y} \rangle + \frac{L_f}{2} \|\boldsymbol{X} - \boldsymbol{Y}\|^2 + \mu g(\boldsymbol{X}) \quad \textbf{(4.22)}$$

In reality, $Q(\boldsymbol{X}, \boldsymbol{Y})$ is a sequence of separable quadratic approximations to $F(\boldsymbol{X})$ which form upper bounds $F(\boldsymbol{X})$ for any $\boldsymbol{Y}$. Further discussion about the choice of the specific points $\boldsymbol{Y}$ is out of the scope of this thesis, but we should highlight the fact that the whole process is inspired by the famous paper of Y. Nesterov ([48]) and refer to [43] for further information. Taken all that into mind, the general form of the PG Algorithm is depicted in the following image ([43]):

Algorithm 1 (General Proximal Gradient Algorithm)

1: while not converged do
2: $Y_k \leftarrow X_k + \frac{t_{k-1}-1}{t_k}(X_k - X_{k-1})$.
3: $G_k \leftarrow Y_k - \frac{1}{L_f}\nabla f(Y_k)$.
4: $X_{k+1} \leftarrow \arg\min_X \left\{\mu g(X) + \frac{L_f}{2}\|X - G_k\|^2\right\}$.
5: $t_{k+1} \leftarrow \frac{1+\sqrt{4t_k^2+1}}{2}$, $k \leftarrow k+1$.
6: end while

Image 9: The Proximal Gradient Algorithm

4.2.2 Algorithm Outline

As explained in details above, the minimization of a sequence of separable quadratic approximations of the objective function seems a good idea. In fact, the reason for that to happen is that in many cases, it is possible to find a simple expression for the minimizer $\boldsymbol{X}_{k+1}$ mentioned above. Taken into mind previous works on soft-thresholding techniques employed in the field of CS ([4], [13]) as well as in that of MC ([12], [63]), enriching them at the same time with continuation techniques, the algorithm proposed in [43] seems quite promising. More precisely, the iterates $\boldsymbol{X}_k$ are ordered pairs $(\boldsymbol{A}_k, \boldsymbol{E}_k) \in R^{m\times n} \times R^{m\times n}$ and $g(\boldsymbol{X}_k) = \|\boldsymbol{A}_k\|_* + \lambda\|\boldsymbol{E}_k\|_1$. Then the optimization problem **(4.19)** takes the following form:

$$\min_{\boldsymbol{A},\boldsymbol{E}} F(\boldsymbol{X}) = \mu\|\boldsymbol{A}\|_* + \mu\lambda\|\boldsymbol{E}\|_1 + \frac{1}{2}\|\boldsymbol{D} - \boldsymbol{A} - \boldsymbol{E}\|_F^2 \quad \textbf{(4.23)}$$

As it is explained in details in [43], using soft-thresholding techniques to compute the iterates $\boldsymbol{X}_{k+1}$ is a reasonable choice made in [69]. However, as it is mentioned above, the number of iterations required for the algorithm to converge is quite large. Nevertheless, if we make use of the PG framework presented above in cooperation with smooth techniques proposed in [48], we could speed up the convergence rate of the algorithm. The most important feature although of the present algorithm is that it makes use of continuation techniques, in the sense that it does not apply the PG algorithm directly to **(4.23)** with a fixed relaxation parameter μ, but on the contrary μ varies from a large initial value μ_0 to a floor $\bar{\mu}$ following a geometrical decrease at each iteration step. As a result, the number of the required iterations for convergence reduces significantly (for more details about convergence theorems the diligent reader is referred to [43]). Having clarified those two crucial details, the APG algorithm for the RPCA problem is depicted in the following image:

Algorithm 2 (Robust PCA via Accelerated Proximal Gradient)

Input: Observation matrix $D \in \mathbb{R}^{m \times n}$, λ.

$A_0, A_{-1} \leftarrow 0$; $E_0, E_{-1} \leftarrow 0$; $t_0, t_{-1} \leftarrow 1$; $\bar{\mu} \leftarrow \delta \mu_0$.

while not converged do

$Y_k^A \leftarrow A_k + \frac{t_{k-1}-1}{t_k}(A_k - A_{k-1})$, $Y_k^E \leftarrow E_k + \frac{t_{k-1}-1}{t_k}(E_k - E_{k-1})$.

$G_k^A \leftarrow Y_k^A - \frac{1}{2}\left(Y_k^A + Y_k^E - D\right)$.

$(U, S, V) \leftarrow \mathrm{svd}(G_k^A)$, $A_{k+1} = U\mathcal{S}_{\frac{\mu_k}{2}}[S]V^T$.

$G_k^E \leftarrow Y_k^E - \frac{1}{2}\left(Y_k^A + Y_k^E - D\right)$.

$E_{k+1} = \mathcal{S}_{\frac{\lambda\mu_k}{2}}[G_k^E]$.

$t_{k+1} \leftarrow \frac{1+\sqrt{4t_k^2+1}}{2}$.

$\mu_{k+1} \leftarrow \max(\eta\mu_k, \bar{\mu})$.

$k \leftarrow k + 1$.

end while

Output: $A \leftarrow A_k$, $E \leftarrow E_k$.

Image 10: The Accelerated Proximal Gradient Algorithm

4.2.3 Stopping Criteria

The stopping criterion of the APG Algorithm bears a strong resemblance to that proposed in [63]. Before stating it, just as in [43], we define:

$$\begin{cases} S_{k+1}^A = 2(\boldsymbol{Y}_k^A - \boldsymbol{A}_{k+1}) + (\boldsymbol{A}_{k+1} + \boldsymbol{E}_{k+1} - \boldsymbol{Y}_k^A - \boldsymbol{Y}_k^E) \\ S_{k+1}^E = 2(\boldsymbol{Y}_k^E - \boldsymbol{E}_{k+1}) + (\boldsymbol{A}_{k+1} + \boldsymbol{E}_{k+1} - \boldsymbol{Y}_k^A - \boldsymbol{Y}_k^E) \end{cases} \textbf{(4.24)}$$

, where:

$$\begin{cases} S_{k+1} = (S_{k+1}^A, S_{k+1}^E) \\ \|S_{k+1}\|^2 = \left\|S_{k+1}^A\right\|_F^2 + \|S_{k+1}^E\|_F^2 \end{cases} \textbf{(4.25)}$$

The iteration loop is terminated when $\|S_{k+1}\|$ is lower than a defined tolerance. In other words, the distance between the origin and the set of *Sub-Gradients* of the *Cost Function* in **(4.23)** at $(\boldsymbol{A}_{k+1}, \boldsymbol{E}_{k+1})$ remains upper bounded by $\|S_{k+1}\|$.

4.2.4 Step-Size Parameters

As stated in [43], as the relaxation parameter μ decreases the closer is the solution gained by the APG algorithm to that of the RPCA problem. A good choice suggested in [43] from empirical results is:

$$\begin{cases} \mu_0 = 0.99\|\boldsymbol{D}\|_2 \\ \delta \leq 10^{-5} \end{cases} \textbf{(4.26)}$$

We should not forget to mention here that in the worst case scenario the iteration complexity of the APG algorithm with decreasing sequence of relaxation parameters μ_k is no better than that with constant ones ($\mu_k = \bar{\mu}$ for all k). However, the usage of a decreasing sequence of relaxation parameters μ_k leads to a significant reduce to the number of iterations required for the algorithm to converge for most of the practical applications.

4.2.5 SVD Computation

Finally, as far as the SVD computation at each iteration is concerned, there is no need to compute the full SVD at each iteration, but only a partial one. The reason for that is that

the soft-thresholding operation will evaporate those singular values corresponding to large values of μ_k. Towards this logic, packages for the computation of partial SVDs (like PROPACK, [41]) would be extremely handful.

4.3 Dual Method

As it has become quite clear so far, the computational bottleneck of the most algorithms developed to tackle the RPCA problem is the computation of the SVD. Although some methods and useful packages may accelerate the whole process, the overall complexity continues to strongly depend on that computation.

In order to avoid such a dependence, an interesting algorithmic scheme was developed in [43]. The authors tried to solve firstly the dual problem (from which the novel method inherited its name as the *Dual Method* (DM)), and subsequently compute the solution of the *Primal Problem*.

4.3.1 Algorithm Outline

As it is explained in details in Appendix I, the spectral norm $\|.\|_2$ is the dual norm of the nuclear norm $\|.\|_*$ of a matrix. From a pure computational point of view, the spectral norm of a matrix is much easily computed than its nuclear norm, as in reality it is its largest singular value (and in general can be computed without the SVD). Inspired by this idea, the authors in [43] suggested to solve the dual of the RPCA problem, which has the following form:

$$\max_{s.t. J(\boldsymbol{Y})\leq 1} \langle \boldsymbol{D}, \boldsymbol{Y} \rangle \quad \textbf{(4.27)}$$

, where:

$$\begin{cases} \langle \boldsymbol{A}, \boldsymbol{B} \rangle = tr(\boldsymbol{A}^T \boldsymbol{B}) \\ J(\boldsymbol{Y}) = max\left(\|\boldsymbol{Y}\|_2, \frac{1}{\lambda} \|\boldsymbol{Y}\|_\infty \right) \end{cases} \quad \textbf{(4.28)}$$

From **(4.27)**-**(4.28)** it is clear that on the one hand the objective function is a linear one while on the other hand $J(\boldsymbol{Y})$ is positive and homogenous (see Appendix I for more information about positivity-homogeneity). As a result, the optimal solution of **(4.27)** must lie on the manifold:

$$S = \{\boldsymbol{Y} | J(\boldsymbol{Y}) = 1\} \quad \textbf{(4.29)}$$

, and therefore the inequality constraint can be replaced by an equality one. This process will lead to an optimization problem on a nonlinear non-smooth manifold, which can then be solved by *Steepest Ascend* techniques.

More precisely, at each iteration step k, we need to compute the steepest ascend direction $\boldsymbol{W}_k$ at the estimate of $\boldsymbol{Y}$ at this iteration, $\boldsymbol{Y}_k$. This can be achieved via the projection of the gradient $\boldsymbol{D}$ of the objective function of **(4.27)** onto the tangent cone of S. Afterwards, line search can take place along $\boldsymbol{W}_k$ in order to determine the step-size parameter δ_k, by solving the following problem:

$$\delta_k = arg \max_{\delta \geq 0} \left\langle \boldsymbol{D}, \frac{\boldsymbol{Y}_k + \delta \boldsymbol{W}_k}{J(\boldsymbol{Y}_k + \delta \boldsymbol{W}_k)} \right\rangle \quad \textbf{(4.30)}$$

, and then the estimation of $\boldsymbol{Y}$ at iteration $k+1$ can be computed as follows:

$$\boldsymbol{Y}_{k+1} = \frac{\boldsymbol{Y}_k + \delta_k \boldsymbol{W}_k}{J(\boldsymbol{Y}_k + \delta_k \boldsymbol{W}_k)} \quad \textbf{(4.31)}$$

As it is stated as well as proved in [43], the subsequent algorithm which implements the above ideas finds the optimal solution of the dual problem of the RPCA problem.

In order to find the steepest ascend direction $\boldsymbol{W}_k$, the suggestion made in [43] is the following one:

$$\boldsymbol{W}_k = \boldsymbol{D} - \boldsymbol{D}_k \textbf{ (4.32)}$$

, where $\boldsymbol{D}_k$ is the projection of $\boldsymbol{D}$ onto the normal cone $N(\boldsymbol{Y}_k)$ of S. As it is stated in [43], the *Normal Cone* is defined as:

$$N(\boldsymbol{Y}_k) = \{\alpha \boldsymbol{X} : \alpha \geq 0, \boldsymbol{X} \in \partial J(\boldsymbol{Y}_k)\} \textbf{ (4.33)}$$

Then, the sub-gradient of J can be computed as follows:

$$\partial J(\boldsymbol{Y}_k) = \begin{cases} \partial \|\boldsymbol{Y}_k\|_2, \ \partial \|\boldsymbol{Y}_k\|_2 > \frac{1}{\lambda}\partial \|\boldsymbol{Y}_k\|_\infty \\ \partial\left(\frac{1}{\lambda}\|\boldsymbol{Y}_k\|_\infty\right), \ \partial \|\boldsymbol{Y}_k\|_2 < \frac{1}{\lambda}\partial \|\boldsymbol{Y}_k\|_\infty \\ ch\left\{\partial \|\boldsymbol{Y}_k\|_2, \partial\left(\frac{1}{\lambda}\|\boldsymbol{Y}_k\|_\infty\right)\right\}, \ \partial \|\boldsymbol{Y}_k\|_2 = \frac{1}{\lambda}\partial \|\boldsymbol{Y}_k\|_\infty \end{cases} \textbf{ (4.34)}$$

, as $J(\boldsymbol{Y})$ is the maximum of two convex functions ("ch" denotes the convex hull -see Appendix II).

As a consequence, in the case where $\partial \|\boldsymbol{Y}_k\|_2$ and $\frac{1}{\lambda}\partial \|\boldsymbol{Y}_k\|_\infty$ are not equal, all that has to be done is the computation of the projection of $\boldsymbol{D}$, $\pi_{2/\infty}(\boldsymbol{D})$, onto the cone $N_{2/\infty}(\boldsymbol{Y}_k)$ generated by the respective sub-gradient $\partial \|.\|_{2/\infty}$ at $\boldsymbol{Y}_k$, according to which one of them is the larger one. However, if the aforementioned sub-gradients are equal to one another, then the normal cone is composed as follows:

$$N(\boldsymbol{Y}_k) = N_2(\boldsymbol{Y}_k) + N_\infty(\boldsymbol{Y}_k) \textbf{ (4.35)}$$

In such a case the computation of the projection of $\boldsymbol{D}$ onto $N(\boldsymbol{Y}_k)$ can be accomplished by alternating between projections onto $N_2(\boldsymbol{Y}_k)$ and $N_\infty(\boldsymbol{Y}_k)$. In other words, an *Alternating Projection* (AP) *Scheme* can be implemented obeying the following idea:

$$\begin{cases} \boldsymbol{A}_{i+1} = \pi_2(\boldsymbol{D} - \boldsymbol{E}_i) \\ \boldsymbol{E}_{i+1} = \pi_\infty(D - \boldsymbol{A}_{i+1}) \\ i = i + 1 \end{cases} \textbf{ (4.36)}$$

, where $\boldsymbol{E}_0 = \boldsymbol{0}$ and the counter i is initialized at zero. The hopeful news are that, as it is proved in details in [43], the above algorithmic scheme will eventually derive the desired projection $\boldsymbol{D}_k$.

As the dual problem can be solved as explained above, there remain two KKT conditions to be fulfilled for the primal one (the RPCA problem) to have a solution as well:

$$\begin{cases} \widehat{\boldsymbol{Y}} \in \partial \|\boldsymbol{A}\|_* \\ \frac{1}{\lambda}\widehat{\boldsymbol{Y}} \in \partial \|\boldsymbol{E}\|_1 \end{cases} \textbf{ (4.37)}$$

, where $\widehat{\boldsymbol{Y}}$ is the solution obtained for the dual problem. But from the definition of these two sub-gradients, there are three different cases arising according to the value of the respective norms:

- If $\left\|\widehat{\boldsymbol{Y}}\right\|_2 < 1$, the solution of the primal problem will be $\begin{cases} \boldsymbol{A} = \boldsymbol{0} \\ \boldsymbol{E} = \boldsymbol{D} \end{cases}$
- If $\frac{1}{\lambda}\left\|\widehat{\boldsymbol{Y}}\right\|_\infty < 1$, the solution of the primal problem will be $\begin{cases} \boldsymbol{A} = \boldsymbol{D} \\ \boldsymbol{E} = \boldsymbol{0} \end{cases}$
- If $\left\|\widehat{\boldsymbol{Y}}\right\|_2 = \frac{1}{\lambda}\left\|\widehat{\boldsymbol{Y}}\right\|_\infty = 1$, then (as it is stated in [43] in the form of Theorem 3.4) any pair of accumulation points $\widehat{\boldsymbol{A}}$, $\widehat{\boldsymbol{E}}$ generated by projecting $\boldsymbol{D}$ onto $N\left(\widehat{\boldsymbol{Y}}\right)$ via the AP algorithm **(4.36)** solves the primal RPCA problem

As a result, by solving the dual problem of the RPCA problem, we obtain the solution to the primal one. Having explained those key ideas behind the interpretation of the duality theory as well as its usage in order to solve the primal optimization problem, the DM algorithm for the RPCA problem is depicted in the following image:

Algorithm 3 (Robust PCA via the Dual)

Input: Observation matrix $D \in \mathbb{R}^{m\times n}$, λ.

$Y_0 = \text{sgn}(D)/J(\text{sgn}(D))$; $k \leftarrow 0$.
while not converged do
  Compute the projection D_k of D onto $N(Y_k)$:
  if $\|Y_k\|_2 > \lambda^{-1}|Y_k|_\infty$ **then**
    $D_k \leftarrow \pi_2(D)$, $A \leftarrow D$, $E \leftarrow 0$.
  else if $\lambda^{-1}|Y_k|_\infty > \|Y_k\|_2$ **then**
    $D_k \leftarrow \pi_\infty(D)$, $A \leftarrow 0$, $E \leftarrow D$.
  else
    $A \leftarrow 0$, $E \leftarrow 0$.
    while not converged do
      $A \leftarrow \pi_2(D-E)$, $E \leftarrow \pi_\infty(D-A)$.
    end while
    $D_k \leftarrow A + E$.
  end if
  Do line search to determine a step size δ_k.
  $Y_{k+1} \leftarrow \frac{Y_k+\delta_k(D-D_k)}{J(Y_k+\delta_k(D-D_k))}$ and $k \leftarrow k+1$.
end while

Output: (A, E).

Image 11: The Dual Method Algorithm

4.3.2 Norm Computation

As it was mentioned above, $J(.)$ is the maximum of two norms: the spectral norm and the max-row-sum norm. As it is obvious from their respective definitions in Appendix II, the most computational thirsty among them is the spectral norm. As a result, efficient methods for computing the spectral norm of a matrix should be used in order to reduce computational complexity. As it is explained in [43], for such a computation the PROPACK package ([41]) is chosen because on the one hand it is way faster than the classical *Power Method* (PM) and other hand it can compute solely the largest singular value of a matrix without computing firstly its (computational "ponderous") SVD.

Furthermore, as far as the computation of the projection $\boldsymbol{D}_k$ of $\boldsymbol{D}$ onto the normal cone $N(\boldsymbol{Y}_k)$ of S is concerned, an "equality check" between $\|\boldsymbol{Y}_k\|_2$ and $\frac{1}{\lambda}\|\boldsymbol{Y}_k\|_\infty$ has to take place. For this purpose, what is suggested in [43] is to check whether the discrepancy between them is larger than a predefined tolerance ε (i.e. $\varepsilon = 10^{-4}$).

4.3.3 SVD Computation

As far as the computation of the *Principal Singular Spaces* is concerned, the PROPACK package ([41]) is chosen once more. Nevertheless, it should be mentioned here that possibly there are faster methods to do this computation, as it is explained in [43], but the authors finally selected PROPACK as at the time [43] was published the relative search was at an exploring stage.

4.3.4 Step-Size Parameters

As it is known from optimization theory, when employing steepest ascend / descend methods, there is always the possibility of zig-zagging around the solution. Therefore for the determination of the step-size parameter δ_k an exact line search has to take place. The method chosen to do so is based on *Armijo's Rule*, as it is explained in [43]. We should not forget to mention at this point the fact that the projections onto the normal cone described in this algorithm are performed inexactly, for speeding up the whole process (for further details see [43]).

4.3.5 Stopping Criteria

The stopping criteria of the DM algorithm obviously concern the two different methods that take place as the algorithm goes on: the steepest ascend method (for the computation of the direction $\boldsymbol{W}_k$) and the alternating projection method (for the computation of the projection of $\boldsymbol{D}$ onto $N(\boldsymbol{Y}_k)$).

For the first one, the respective stopping criterion is suggested ([43]) to be the following one:

$$\frac{\|\boldsymbol{D}-\boldsymbol{D}_k\|_F}{\|\boldsymbol{D}\|_F} < \varepsilon \quad \textbf{(4.38)}$$

, or in other words we demand the reconstruction error to be lower than a predefined tolerance (i.e. $\varepsilon = 2 \times 10^{-5}$).

Following a similar way of thinking, the stopping criterion corresponding to the second one is suggested ([43]) to be the following one:

$$\begin{cases} \frac{\|\boldsymbol{A_i}-\boldsymbol{A}_{i-1}\|_F}{\|\boldsymbol{D}\|_F} < 10^{-8} \\ \frac{\|\boldsymbol{E_i}-\boldsymbol{E}_{i-1}\|_F}{\|\boldsymbol{D}\|_F} < 10^{-8} \end{cases} \quad \textbf{(4.39)}$$

4.4 Augmented Lagrange Multiplier Method

Although the aforementioned APG method constitutes a good choice for coping with the RPCA problem, it has been proven in theory that its convergence speed is only sub-linear. Considering that and exploring for further ameliorations, the authors in [42] adopted an *Augmented Lagrange Multiplier* (ALM) *Method* / approach lent by constrained optimization theory.

More precisely, ALM methods are a certain class of algorithms for solving constrained optimization problems. They have similarities to *Penalty Methods* in that they replace a constrained optimization problem by a series of unconstrained problems and add a penalty term to the objective; the difference is that the ALM method adds yet another term, designed to mimic a *Lagrange Multiplier*. It should be mentioned here that the ALM method is not the same as the *Method of Lagrange Multipliers*. Nevertheless, the two methods are somehow related in the sense that in the ALM method the unconstrained objective function is the *Lagrangian Function* of the constrained problem, with an additional penalty term (the augmentation).

The method was originally known as the *Method of Multipliers*, and was studied much in the 1970s and 1980s as a good alternative to penalty methods. It was first discussed by Hestenes ([36]) and by Powell ([52]) in 1969. The method was also studied by Bertsekas, notably in his 1982 book ([8]) for solving constrained optimization problems of the following kind:

$$\begin{cases} \min\limits_{s.t.h(\boldsymbol{X})=\boldsymbol{0}} f(\boldsymbol{X}) \\ f: R^n \to R \\ h: R^n \to R^m \end{cases} \textbf{(4.40)}$$

This problem can be solved as a series of unconstrained minimization problems. For example, the penalty method approach uses as the unconstrained objective function the following one:

$$P(\boldsymbol{X}, \boldsymbol{Y}, \mu) = f(\boldsymbol{X}) + \mu \|h(\boldsymbol{X})\|_F^2 \textbf{ (4.41)}$$

, and then solves the optimization problem iteratively. Then, at the next iteration step, it re-solves the problem using a larger value of the step-size parameter μ, while at the same time the old solution is used as an initial guess or a "warm-start".

On the other hand, the ALM method approach uses as the unconstrained objective function the *Augmented Lagrangian Function*, which is defined as follows:

$$L(\boldsymbol{X}, \boldsymbol{Y}, \mu) = f(\boldsymbol{X}) + \langle \boldsymbol{Y}, h(\boldsymbol{X}) \rangle + \frac{\mu}{2} \|h(\boldsymbol{X})\|_F^2 \textbf{ (4.42)}$$

, where $\mu > 0$ and $\langle . \rangle$ denotes the standard *Trace Inner Product*. Consequently, the optimization problem can then be solved iteratively via the algorithmic scheme ([9]) described in the following image:

Algorithm 3 (General Method of Augmented Lagrange Multiplier)

1: $\rho \geq 1$.
2: **while** not converged **do**
3: Solve $X_{k+1} = \arg\min_X L(X, Y_k, \mu_k)$.
4: $Y_{k+1} = Y_k + \mu_k h(X_{k+1})$;
5: Update μ_k to μ_{k+1}.
6: **end while**
Output: X_k.

Image 12: The Augmented Lagrange Multiplier Method Algorithm

The reason why the above algorithmic scheme is so attractive is that, as it has been proven in [8], the Lagrange multipliers $\boldsymbol{Y}_k$ produced by it converge to the optimal solution:

- Q-linearly, when $\{\mu_k\}$ is a bounded increasing sequence, and f as well as g are continuously differentiable functions
- Super-Q-linearly, when $\{\mu_k\}$ is an unbounded increasing sequence, and f as well as g are continuously differentiable functions

Another advantage of the ALM method is that the parameter tuning is much easier than that of IT algorithmic schemes, as the optimal step-size parameter for the update of $\boldsymbol{Y}_k$ is proven to be the chosen penalty term μ_k. Nevertheless, the major merit of the ALM method is that, unlike penalty methods, it is not necessary to require $\mu_k \to \infty$ in order to solve the original constrained optimization problem. Instead, because of the presence of the (estimated) Lagrange multiplier term $\boldsymbol{Y}_k$ whose accuracy improves at every iteration step, μ_k can stay much smaller. Furthermore, as it is suggested in [39], the ALM method is generally preferred to the quadratic penalty method -since the extra computational cost for its evaluation is insignificant compared to the potential of ill-conditioning that may be caused due to the requirement of the step-size parameter μ to go to infinity.

4.4.1 Exact ALM Method

As it is stated in [42], the ALM method can be applied to the RPCA problem, if we set:

$$\begin{cases} \boldsymbol{X} = (\boldsymbol{A}, \boldsymbol{E}) \\ f(\boldsymbol{X}) = \|\boldsymbol{A}\|_* + \lambda\|\boldsymbol{E}\|_1 \\ h(\boldsymbol{X}) = \boldsymbol{D} - \boldsymbol{A} - \boldsymbol{E} \end{cases} \textbf{(4.43)}$$

Obviously, the Lagrangian function will then be:

$$L(\boldsymbol{A}, \boldsymbol{E}, \boldsymbol{Y}, \mu) = \|\boldsymbol{A}\|_* + \lambda\|\boldsymbol{E}\|_1 + \langle \boldsymbol{Y}, \boldsymbol{D} - \boldsymbol{A} - \boldsymbol{E}\rangle + \frac{\mu}{2}\|\boldsymbol{D} - \boldsymbol{A} - \boldsymbol{E}\|_F^2 \textbf{ (4.44)}$$

Afterwards, there are two alternatives concerning our choice of solving the following problem exactly or not:

$$(\boldsymbol{A}_{k+1}^*, \boldsymbol{E}_{k+1}^*) = arg\min_{\boldsymbol{A},\boldsymbol{E}} L(\boldsymbol{A}, \boldsymbol{E}, \boldsymbol{Y}_k^*, \mu) \textbf{ (4.45)}$$

If we choose to solve the above sub-problem exactly, then the respective algorithm derived in [42] is called the *Exact Augmented Lagrange Multiplier* (EALM) *Method* and is depicted in the following image:

Algorithm 4 (RPCA via the Exact ALM Method)

Input: Observation matrix $D \in \mathbb{R}^{m\times n}$, λ.

$Y_0^* = \text{sgn}(D)/J(\text{sgn}(D))$; $\mu_0 > 0$; $\rho > 1$; $k = 0$.
while not converged do
  // Lines 4-12 solve $(A_{k+1}^*, E_{k+1}^*) = \arg\min_{A,E} L(A, E, Y_k^*, \mu_k)$.
  $A_{k+1}^0 = A_k^*$, $E_{k+1}^0 = E_k^*$, $j = 0$;
  while not converged do
    // Lines 7-8 solve $A_{k+1}^{j+1} = \arg\min_A L(A, E_{k+1}^j, Y_k^*, \mu_k)$.
    $(U, S, V) = \text{svd}(D - E_{k+1}^j + \mu_k^{-1}Y_k^*)$;
    $A_{k+1}^{j+1} = US_{\mu_k^{-1}}[S]V^T$;
    // Line 10 solves $E_{k+1}^{j+1} = \arg\min_E L(A_{k+1}^{j+1}, E, Y_k^*, \mu_k)$.
    $E_{k+1}^{j+1} = \mathcal{S}_{\lambda\mu_k^{-1}}[D - A_{k+1}^{j+1} + \mu_k^{-1}Y_k^*]$;
    $j \leftarrow j + 1$.
  end while
  $Y_{k+1}^* = Y_k^* + \mu_k(D - A_{k+1}^* - E_{k+1}^*)$.
  Update μ_k to μ_{k+1}.
  $k \leftarrow k + 1$.
end while

Output: (A_k^*, E_k^*).

Image 13: The Exact Augmented Lagrange Multiplier Method Algorithm

The initialization of the above algorithm is inspired by the DM, developed in the above section, as it is clearly stated in [42]. The most crucial part although is that concerning the convergence rate of the EALM method, which is proven in [42] (in the form of a Theorem) to be at least $O(\mu_k^{-1})$. In other words, if on the one hand the step-size parameter μ_k grows geometrically then the EALM method will converge Q-linearly. On the other hand, if the step-size parameter μ_k grows faster so does the convergence rate of the EALM method. However, one should be very cautious with the choice of $\{\mu_k\}$, as larger values of it (in order to achieve higher convergence rates) will probably conclude to slower convergence rate of the IT solution of the sub-problem **(4.45)**, as it is explained in details in [42].

4.4.2 Inexact ALM Method

If we choose to solve the sub-problem **(4.45)** inexactly, then the respective algorithm derived in [42] is called the *Inexact Augmented Lagrange Multiplier* (IALM) *Method* and is depicted in the following image:

Algorithm 5 (RPCA via the Inexact ALM Method)

Input: Observation matrix $D \in \mathbb{R}^{m \times n}$, λ.

1: $Y_0 = D/J(D)$; $E_0 = 0$; $\mu_0 > 0$; $\rho > 1$; $k = 0$.
2: **while not converged do**
3: // Lines 4-5 solve $A_{k+1} = \arg\min_A L(A, E_k, Y_k, \mu_k)$.
4: $(U, S, V) = \text{svd}(D - E_k + \mu_k^{-1} Y_k)$;
5: $A_{k+1} = U\mathcal{S}_{\mu_k^{-1}}[S]V^T$.
6: // Line 7 solves $E_{k+1} = \arg\min_E L(A_{k+1}, E, Y_k, \mu_k)$.
7: $E_{k+1} = \mathcal{S}_{\lambda\mu_k^{-1}}[D - A_{k+1} + \mu_k^{-1} Y_k]$.
8: $Y_{k+1} = Y_k + \mu_k(D - A_{k+1} - E_{k+1})$.
9: Update μ_k to μ_{k+1}.
10: $k \leftarrow k + 1$.
11: **end while**

Output: (A_k, E_k).

Image 14: The Inexact Augmented Lagrange Multiplier Method Algorithm

The main idea behind this approach is that $\boldsymbol{A}_k$ as well as $\boldsymbol{E}_k$ will still converge to the optimal solution of the RPCA problem, even if their respective updates take place only once the sub-problem **(4.45)** is solved.

The validity as well as the optimality of the above approach is guaranteed via two Theorems stated in [42] concerning the choice of the step-size parameters $\{\mu_k\}$. Nevertheless, it should be mentioned that the respective convergence rate is not specified via those two Theorems, as it was the case with the EALM method mentioned above.

4.4.3 SVD Computation

As it is apparent from the previous images describing the EALM and IALM methods, the SVD computation is required for both of them. Borrowing the ideas developed in the previous sections of this Chapter, there is no need to compute the full SVD but only those singular values that exceed a specific threshold. Once more, such a goal is achieved via the help of the PROPACK package ([41]) for the SVD computation. As it was made clear before, the dimension of the principal singular space whose singular values exceed the specified threshold has to be determined ahead of time. For such a prediction, the respective criterion proposed in [42] for the IALM method is the following one:

$$sv_{k+1} = \begin{cases} svp_k + 1, & svp_k < sv_k \\ min\left(svp_k + round\left(0.05min(m,n), min(m,n)\right)\right), & svp_k = sv_k \end{cases} \quad \textbf{(4.46)}$$

, where sv_k is the predicted dimension, svp_k is the number of singular values in sv_k that exceed μ_k^{-1} and $sv_0 = 10$.

As for the EALM method, the above criterion is used for the inner loop which computes the solution of **(4.45)**, while for the outer one it takes the following form:

$$sv_{k+1} = min\left(svp_k + round\left(0.1min(m,n), min(m,n)\right)\right) \quad \textbf{(4.47)}$$

4.4.4 Order of Updating A and E

Although it may seem meaningless, the order of updating $\boldsymbol{A}$ and $\boldsymbol{E}$ does count in practice. More precisely, as it is explained in details in [42], after quite a lot numerical tests, it is proven that it is wiser to update first $\boldsymbol{E}$ and then $\boldsymbol{A}$ in the EALM method as well as in the IALM one. This is due to fact that following such an approach leads to a slightly lower number of iterations needed to achieve the same accuracy than that occurring from the

adoption of the opposite scenario. It shouldn't be forgotten here the fact that the update of $\boldsymbol{E}$ precedes that of $\boldsymbol{A}$ will probably lead to less computational burden concerning the partial SVD, as the rank of $\boldsymbol{A}_k$ will increase monotonically -as stated in [42].

4.4.5 Stopping Criteria

The stopping criteria of the EALM and IALM methods are directly connected to the KKT conditions of the RPCA problem ([42]):

$$\begin{cases} \boldsymbol{D} - \boldsymbol{A}^* - \boldsymbol{E}^* = \boldsymbol{0} \\ \boldsymbol{Y}^* \in \partial \|\boldsymbol{A}^*\|_* \\ \boldsymbol{Y}^* \in \partial(\|\lambda \boldsymbol{E}^*\|_1) \end{cases} \textbf{(4.48)}$$

For the last two conditions of **(4.48)** to hold, it is required that:

$$\partial \|\boldsymbol{A}^*\|_* \cap \partial(\|\lambda \boldsymbol{E}^*\|_1) = \emptyset \textbf{ (4.49)}$$

As a consequence, the stopping criteria proposed in [42] for the EALM and IALM methods are the summarized below:

$$\begin{cases} \frac{\|\boldsymbol{D}-\boldsymbol{A}_k-\boldsymbol{E}_k\|_F}{\|\boldsymbol{D}\|_F} < \varepsilon_1 \\ \frac{dist\left(\partial \|\boldsymbol{A}^*\|_*, \partial(\|\lambda \boldsymbol{E}^*\|_1)\right)}{\|\boldsymbol{D}\|_F} < \varepsilon_2 \end{cases} \textbf{(4.50)}$$

, where the $dist(.)$ operation in the second condition of **(4.50)** is defined as follows:

$$dist(\boldsymbol{X}, \boldsymbol{Y}) = min(\|x - y\|_F \mid x \in \boldsymbol{X}, y \in \boldsymbol{Y}) \textbf{ (4.51)}$$

As far as the EALM method is concerned, the second condition of **(4.50)** holds always true, due to the inner loop of the algorithm, so its check is superfluous. On the other hand, this is not the case with the IALM method, and to make things even worse, this check is computationally expensive -as explained in [42]. To avoid such inconvenient situations, what is proposed in [42] is the replacement of the second condition of **(4.50)** with the following one:

$$\left\|\widehat{\boldsymbol{Y}_k} - \boldsymbol{Y}_k\right\|_F < \mu_{k-1} \|\boldsymbol{E}_k - \boldsymbol{E}_{k-1}\|_F \textbf{ (4.52)}$$

The key idea behind such a choice is that it constitutes a good estimate of $dist\left(\partial \|\boldsymbol{A}^*\|_*, \partial(\|\lambda \boldsymbol{E}^*\|_1)\right)$ as the following two conditions hold true:

$$\begin{cases} \widehat{\boldsymbol{Y}_k} \in \partial \|\boldsymbol{A}^*\|_* \\ \boldsymbol{Y}_k \in \partial(\|\lambda \boldsymbol{E}^*\|_1) \end{cases} \textbf{(4.53)}$$

4.4.6 Step-Size Parameters

As far as the step-size parameter μ_k is concerned, what is proposed in [42] is the following adaptive update:

$$\mu_{k+1} = \begin{cases} \rho \mu_k, \frac{\mu_k \|\boldsymbol{E}_{k+1}-\boldsymbol{E}_k\|_F}{\|\boldsymbol{D}\|_F} < \varepsilon_2 \\ \mu_k, otherwise \end{cases} \textbf{(4.54)}$$

The adaptive nature of the above choice is preferred to a constant sequence $\{\mu_k\}$, as it constitutes a special case of the more general adaptive scenario, for which **(4.54)** is proven to also hold true in [42].

4.4.7 Initialization Steps

Finally, the determination of the specific values of the parameters involved in the above criteria has to take place.

According to [42], as far as the EALM method is concerned, we have:

$$\begin{cases} \mu_0 = \frac{0.5}{\|sgn(\boldsymbol{D})\|_2} \\ \rho = 6 \end{cases} \textbf{(4.55)}$$

For the inner loop, the stopping criteria are the following ones:

$$\begin{cases} \frac{\left\|\boldsymbol{A}_k^{j+1} - \boldsymbol{A}_k^j\right\|_F}{\|\boldsymbol{D}\|_F} < 10^{-6} \\ \frac{\left\|\boldsymbol{E}_k^{j+1} - \boldsymbol{E}_k^j\right\|_F}{\|\boldsymbol{D}\|_F} < 10^{-6} \end{cases} \textbf{(4.56)}$$

, while for the outer loop the respective criterion is:

$$\frac{\left\|\boldsymbol{D} - \boldsymbol{A}_k^* - \boldsymbol{E}_k^*\right\|_F}{\|\boldsymbol{D}\|_F} < 10^{-7} \textbf{(4.57)}$$

In what has to do with the IALM method, the respective initial setting to **(4.55)** takes the form:

$$\begin{cases} \mu_0 = \frac{1.25}{\|\boldsymbol{D}\|_2} \\ \rho = 1.6 \end{cases} \textbf{(4.58)}$$

As for the parameters involved in the stopping criteria, the following choice is proposed ([42]):

$$\begin{cases} \varepsilon_1 = 10^{-7} \\ \varepsilon_2 = 10^{-5} \end{cases} \textbf{(4.59)}$$

4.5 Alternating Direction Method

The ALM method analyzed above is very elegant strategy for treating the RPCA problem, with great theoretical as well as numerical results. However, if we examine a bit closer the RPCA problem **(3.6)**, we could notice that is a “well-structured” one -in the sense that it is separable both in the objective function as well as in the constraints. With that thought as a starting point, it was proposed in [72] to adopt the *Alternating Direction Method* (ADM, also known as the *Alternating Direction Method of Multipliers* (ADMM)) for exploiting exactly this favorable structure of the problem.

In general, the ADM method consists a variant of the standard ALM method that uses partial updates (similar to the *Gauss-Seidel Method* for solving linear equations) for the *Dual Variables*. More precisely, it tackles convex optimization problems with linear constraints -exploiting their separable nature- of the form:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + g(\boldsymbol{x}) \textbf{(4.60)}$$

The above optimization problem is equivalent to the constrained one:

$$\min_{s.t. \boldsymbol{x}=\boldsymbol{y}} f(\boldsymbol{x}) + g(\boldsymbol{y}) \textbf{(4.61)}$$

Though this change may seem trivial, the problem can now be treated using methods of constrained optimization (in particular, the ALM method), and the objective function is separable in $\boldsymbol{x}$ as well as in $\boldsymbol{y}$. The dual update requires solving a *Proximity Function* in $\boldsymbol{x}$ and $\boldsymbol{y}$ at the same time; the ADM method allows this problem to be solved approximately by first solving for $\boldsymbol{x}$ with $\boldsymbol{y}$ fixed, and then solving for $\boldsymbol{y}$ with $\boldsymbol{x}$ fixed -exploiting in that way the separable nature of the problem.

Rather than iterate until convergence (like the *Jacobi Method*), the algorithm proceeds directly to updating the dual variable and then repeating the process. This is not

equivalent to the exact minimization, but surprisingly, it can still be shown that this method converges to the right answer (under some assumptions). Because of this approximation, the algorithm is distinct from the pure ALM method.

4.5.1 Algorithm Outline

As it was made clear in the above section, when referring to the RPCA problem:

$$\min_{s.t.\boldsymbol{B}+\boldsymbol{A}=\boldsymbol{C}} \|\boldsymbol{B}\|_* + \gamma\|\boldsymbol{A}\|_1 \textbf{ (4.62)}$$

, where $\gamma > 0$, the augmented Lagrangian function defined in **(4.42)** has the following form:

$$L(\boldsymbol{A},\boldsymbol{B},\boldsymbol{Z}) = \|\boldsymbol{B}\|_* + \gamma\|\boldsymbol{A}\|_1 + \langle \boldsymbol{Z}, \boldsymbol{A}+\boldsymbol{B}-\boldsymbol{C}\rangle + \frac{\beta}{2}\|\boldsymbol{A}+\boldsymbol{B}-\boldsymbol{C}\|_F^2 \textbf{ (4.63)}$$

, where $\beta > 0$ represents a penalty parameter for violating the linear constraints and $\boldsymbol{Z} \in R^{m\times n}$ is the Lagrange multiplier of the linear constraints. After applying the ALM method, the solution to the problem is computed iteratively throughout the following scheme:

$$\begin{cases} (\boldsymbol{A}^{k+1},\boldsymbol{B}^{k+1}) \in arg\min_{s.t.\boldsymbol{A},\boldsymbol{B}\in R^{m\times n}} \{L(\boldsymbol{A},\boldsymbol{B},\boldsymbol{Z}^k)\} \\ \boldsymbol{Z}^{k+1} = \boldsymbol{Z}^k - \beta(\boldsymbol{A}^{k+1}+\boldsymbol{B}^{k+1}-\boldsymbol{C}) \end{cases} \textbf{ (4.64)}$$

As it is obvious from **(4.64)**, the low-rank component as well as the sparse one are minimized simultaneously. Nevertheless, if we take advantage of the separable flavor of the objective function as well as of that of the linear constraints, the above minimization can take place separately leading to the ADM approach proposed in [72]:

$$\begin{cases} \boldsymbol{A}^{k+1} \in arg\min_{s.t.\boldsymbol{A}\in R^{m\times n}} \{L(\boldsymbol{A},\boldsymbol{B}^k,\boldsymbol{Z}^k)\} \\ \boldsymbol{B}^{k+1} \in arg\min_{s.t.\boldsymbol{B}\in R^{m\times n}} \{L(\boldsymbol{A}^{\boldsymbol{k+1}},\boldsymbol{B},\boldsymbol{Z}^k)\} \\ \boldsymbol{Z}^{k+1} = \boldsymbol{Z}^k - \beta(\boldsymbol{A}^{k+1}+\boldsymbol{B}^{k+1}-\boldsymbol{C}) \end{cases} \textbf{ (4.65)}$$

Then, as it is explained in details in [72], the two first optimization problem of **(4.65)** can be solved as follows:

$$\begin{cases} \boldsymbol{A}^{k+1} = \frac{1}{\beta}\boldsymbol{Z}^k - \boldsymbol{B}^{\kappa} + \boldsymbol{C} - P_{\Omega_\infty^{\gamma/\beta}}\left[\frac{1}{\beta}\boldsymbol{Z}^k - \boldsymbol{B}^{\kappa} + \boldsymbol{C}\right] \\ \boldsymbol{B}^{k+1} = \boldsymbol{U}^{k+1} diag\left(max\left\{\sigma_i{}^{\kappa+1} - \frac{1}{\beta}, 0\right\}\right)(\boldsymbol{V}^{k+1})^T \end{cases} \textbf{ (4.66)}$$

, where $P_{\Omega_\infty^{\gamma/\beta}}$ denotes the *Euclidean Projection* onto the set:

$$\Omega_\infty^{\gamma/\beta} = \left\{\boldsymbol{X} \in R^{n\times n} \mid -\frac{\gamma}{\beta} \le X_{ij} \le \frac{\gamma}{\beta}\right\} \textbf{ (4.67)}$$

, while at the same time $\boldsymbol{U}^{k+1} \in R^{m\times r}$ and $\boldsymbol{V}^{k+1} \in R^{n\times r}$ are obtained via the following SVD:

$$\begin{cases} \boldsymbol{C} - \boldsymbol{A}^{k+1} + \frac{1}{\beta}\boldsymbol{Z}^k = \boldsymbol{U}^{k+1}\boldsymbol{\Sigma}^{k+1}(\boldsymbol{V}^{k+1})^T \\ \boldsymbol{\Sigma}^{k+1} = diag\left(\left\{\sigma_i^{k+1}\right\}_{i=1}^r\right) \end{cases} \textbf{ (4.68)}$$

Taking into mind the above approach, the ADM method can then be formulated as in the following image:

Algorithm: the ADM for SLRMD problem:

Step 1. Generate A^{k+1}:

$$A^{k+1} = \frac{1}{\beta}Z^k - B^k + C - P_{\Omega_\infty^{\gamma/\beta}}[\frac{1}{\beta}Z^k - B^k + C].$$

Step 2 Generate B^{k+1}:

$$B^{k+1} = U^{k+1}\,\mathrm{diag}(\max\{\sigma_i^{k+1} - \frac{1}{\beta}, 0\})(V^{k+1})^T,$$

where U^{k+1}, V^{k+1} and $\{\sigma_i^{k+1}\}$ are generated by the singular values decomposition of $C - A^{k+1} + \frac{1}{\beta}Z^k$, i.e.,

$$C - A^{k+1} + \frac{1}{\beta}Z^k = U^{k+1}\Sigma^{k+1}(V^{k+1})^T, \text{ with } \Sigma^{k+1} = \mathrm{diag}(\{\sigma_i^{k+1}\}_{i=1}^r).$$

Step 3. Update the multiplier:

$$Z^{k+1} = Z^k - \beta(A^{k+1} + B^{k+1} - C).$$

Image 15: The Alternating Direction Method Algorithm

4.5.2 SVD Computation

Obviously, as it is clear from Image 15, the most computational "thirsty" step of the ADM method is that related to the SVD computation, with complexity $O(n^3)$ ([30]). In order to make such a computation even more efficient, the aid of the PROPACK package ([41]) was once more proven priceless, as was also for most of the methods described in this Chapter.

4.5.3 Step-Size Parameters

As far as the step-size parameter β is concerned, in [72] is proposed to be set to the following constant value:

$$\beta = \frac{0.25mn}{\|\boldsymbol{C}\|_1} \textbf{ (4.69)}$$

Concerning an adaptive strategy for the update of the step-size parameter β, the reader is referred to [72] for further information as well as details about such a possibility.

4.5.4 Initialization Steps

In what has to do with the relaxation parameter γ, in [72] is proposed to be evaluated as follows:

$$\begin{cases} \gamma = \frac{t}{1-t} \\ t \in (0,1) \end{cases} \textbf{ (4.70)}$$

The rationale of such a choice is that the RPCA problem **(4.62)** now takes the following form:

$$\min_{s.t.\boldsymbol{B}+\boldsymbol{A}=\boldsymbol{C}} (1-t)\|\boldsymbol{B}\|_* + t\|\boldsymbol{A}\|_1 \textbf{ (4.71)}$$

, in which the involved parameter t lives in a finite interval compared to the infinite one concerning the primary relaxation parameter, as $\gamma \in (0, \infty)$. In order to achieve the best balance between the two terms of the objective function in **(4.71)**, an interesting strategy is proposed in [72] -based on the admission of t staying away from the extreme points of its "living space" (i.e. 0 and 1). After many practical tests, the parameter t is proposed in [72] to be set to $t = 0.1$, in order to achieve the aforementioned balance.

4.5.5 Stopping Criteria

The stopping criterion of the ADM method is related to the "amount" of the relative error between the original low-rank and sparse components we wish to recover, $(\boldsymbol{B}^*, \boldsymbol{A}^*)$, and the numerical solution obtained for them via the ADM method, $(\widehat{\boldsymbol{B}_t}, \widehat{\boldsymbol{A}_t})$, that can be tolerated. In other words, the ADM method is proposed in [72] to be terminated when the quality of recovery drops below a predefined tolerance $\varepsilon > 0$ (i.e. $\varepsilon = 10^{-6}$):

$$\frac{\left\|(\widehat{\boldsymbol{B}_t}, \widehat{\boldsymbol{A}_t}) - (\boldsymbol{B}^*, \boldsymbol{A}^*)\right\|_F}{\|(\boldsymbol{B}^*, \boldsymbol{A}^*)\|_F + 1} \leq \varepsilon \quad \textbf{(4.72)}$$

If we interpret the above criterion in terms of transition from a specific iteration, k, of the algorithm to its consecutive one, $k + 1$, then it takes the following form:

$$\frac{\left\|(\boldsymbol{B}^{k+1}, \boldsymbol{A}^{k+1}) - (\boldsymbol{B}^k, \boldsymbol{A}^k)\right\|_F}{\left\|(\boldsymbol{B}^k, \boldsymbol{A}^k)\right\|_F + 1} \leq \varepsilon \quad \textbf{(4.73)}$$

4.6 Comparison of Algorithms

Various methods were presented throughout this Chapter for dealing with the RPCA problem. Of course, each one of them has its pros and cons -depending each time on which specific application problem we wish to apply them. In order to avoid such an "unfair" regime, we decided at the present section of this thesis to carry out a comparison among them based on well-known metrics used for such purposes in this specific scientific field.

4.6.1 Simulation Conditions

Before mentioning those *Key Performance Indicators* (KPI) which will constitute the basis for the upcoming comparison, we should make clear at this point the circumstances under which this comparison is going to take place.

First of all, the implementation of the aforementioned algorithms was done in Matlab, as well as the simulation tests. The respective code of each one of them was kindly uploaded by the authors to the website maintained by the research group of Professor Yi Ma at the University of Illinois at Urbana-Champaign ([50]). All the simulations were conducted and timed on the same Toshiba Satellite laptop, with an Intel Core i7-3630QM CPU @ 2.40GHz processor and 4.00GB memory, running Windows 10 and Matlab version R2012b.

Furthermore, we denote the true solution of the RPCA problem as $(\boldsymbol{A}_0, \boldsymbol{E}_0)$, where:

$$\begin{cases} \boldsymbol{A}_0 \in R^{m \times m} \\ \boldsymbol{E}_0 \in R^{m \times m} \end{cases} \quad \textbf{(4.74)}$$

In our simulations, the low-rank component $\boldsymbol{A}_0$ is generated as a random $m \times m$ square matrix of rank r. This is achieved via the multiplication of two independent factors:

$$\begin{cases} \boldsymbol{L} \in R^{m \times r} \\ \boldsymbol{R}^T \in R^{r \times m} \end{cases} \quad \textbf{(4.75)}$$

, which are matrices whose elements are *Independent* and *Identically Distributed* (I.I.D. / i.i.d.) Gaussian random variables with zero mean and unit variance (see [15] for more details). Then, $\boldsymbol{A}_0$ is set as their respective product $\boldsymbol{L}\boldsymbol{R}^T$.

As far as the sparse component $\boldsymbol{E}_0$ is concerned, it is generated as a sparse matrix whose support is chosen uniformly at random, and whose non-zero elements are i.i.d. uniformly in the interval $[-500,500]$.

Then, the input matrix of the algorithmic schemes to be tested is set as:

$$\boldsymbol{D} = \boldsymbol{A}_0 + \boldsymbol{E}_0 \textbf{ (4.76)}$$

, while its output is denoted as the ordered pair $(\boldsymbol{A}, \boldsymbol{E})$.

Concerning the weighting parameter λ, it is universally selected for all the algorithmic schemes to be tested as proposed in Chapter 3 ([19]):

$$\lambda = \frac{1}{\sqrt{\max_{m,n}(m,n)}} = \frac{1}{\sqrt{m}} \textbf{ (4.77)}$$

Furthermore, each algorithm terminates its main iteration loop for finding the solution to the RPCA problem, with one of the following two different ways:

- The relative error of reconstruction of the data matrix has exceeded a predefined tolerance set to $\varepsilon = 10^{-6}$:

$$\frac{\|\boldsymbol{D}-\boldsymbol{A}-\boldsymbol{E}\|_F}{\|\boldsymbol{D}\|_F} < 10^{-6} \textbf{ (4.78)}$$

- The number of iterations performed by the algorithm has reached the threshold of maximum number of iterations permitted, which is set to $maxiter = 10000$.

In what has to do with the specific parameters of each algorithm separately, they are chosen as follows:

- **SVT algorithm**:
  - $\tau = 10000$: The threshold parameter τ is set to 10000 for accuracy reasons, as proposed in [69]. Nevertheless, if the dimension of the problem is low, a good choice would also be the following one:

$$\tau = 5\sqrt{mn} \textbf{ (4.79)}$$

  - $\delta = 0.9$: The value of the step-size parameter δ is chosen in this way for speed-up reasons of the algorithm.
- **APG algorithm**:
  - $\delta = 10^{-9}$: The value of the step-size parameter δ is chosen in this way for speed-up reasons of the algorithm.
  - $\eta = 0.9$: The value of this relaxation parameter is chosen exactly as proposed in [43].
- **DM algorithm**:
  - $\delta = 0.1$: The value of the sequence of the step-size parameters δ_k is initialized in this way for accuracy reasons of the algorithm, concerning its adaptive changing during the optimization process.
- **EALM algorithm**:
  - $\rho = 6$: The adaptive incremental of the step-size parameters' sequence $\{\mu_k\}$ is chosen exactly as proposed in **(4.55)** ([42]).

- **IAML algorithm**:
  - $\rho = 1.6$: The adaptive incremental of the step-size parameters' sequence $\{\mu_k\}$ is chosen exactly as proposed in **(4.58)** ([42]).
- **ADM algorithm**:
  - $t = \frac{\lambda}{\lambda+1}$: As it was made clear from **(4.70)** the relaxation parameter γ plays the role of the respective parameter λ for all the other algorithms, with the difference however that it depends on the parameter t for convenience reasons of narrowing the respective interval of possible values. For reasons of fairness we chose to select t in this way (and not as $t = 0.1$, which proposed in [72]), in order for the respective relaxation parameter to have the same "behavior" as in the rest of the algorithms.

4.6.2 Exact Recoverability

The first KPI has to do with the relative error of reconstruction concerning the low-rank as well as the sparse component of the data matrix, $\boldsymbol{A}$ and $\boldsymbol{E}$ respectively, produced by each one of the above algorithms. However, more emphasis should be given to the low-rank component rather than the sparse one, as it is clear from the respective papers describing the aforementioned algorithms. The only reasoning for adopting such a choice is the viewpoint of the whole theory developed in the previous Chapter, which confronts the RPCA problem as recovering a low-rank matrix from gross errors ([19]). In such a case, the relative reconstruction error of the sparse component plays a more subsidiary role rather than a central one -which is assigned to the low-rank component.

More precisely, for each triplet $\{m, rank(\boldsymbol{A}_0), \|\boldsymbol{E}_0\|_0\}$ the RPCA problem was solved for the same input data matrix $\boldsymbol{D}$ (created as mentioned before) using the algorithmic schemes cited in this Chapter, and at each time we measure the relative error of reconstruction of the low-rank and the sparse component of the data matrix:

$$\begin{cases} \frac{\|\boldsymbol{A}-\boldsymbol{A}_0\|_F}{\|\boldsymbol{A}_0\|_F} \\ \frac{\|\boldsymbol{E}-\boldsymbol{E}_0\|_F}{\|\boldsymbol{E}_0\|_F} \end{cases} \textbf{(4.80)}$$

Another interesting KPI is the number of iterations required for convergence of each algorithm, which indicates the amount of computational burden that has to be lifted by the specific solver.

Furthermore, the total computation time in seconds performed by each algorithm until convergence is reported likewise, as it consists a metric of speed of each solver which indicates the time it requires for coming up with the solution of the RPCA problem.

In addition, the number of SVDs performed by each algorithm is also recorded, as some of the algorithms use partial SVD techniques (such as the *Partial Accelerated Proximal Gradient Algorithm* -PAPG- which is mentioned below as a speed-up version of the classical APG algorithm) which lead to lower computational effort -a difference although which is clearer as well as important as the dimension of the problem augments.

There were adopted two different scenarios concerning the "level of low rankness" as well as the "level of sparsity" of the respective components of our input data matrix. In the first one, the experimental regime is as follows:

$$\begin{cases} rank(\boldsymbol{A}_0) = 0.05m \\ \|\boldsymbol{E}_0\|_0 = 0.05m^2 \end{cases} \textbf{(4.81)}$$

, while the second one is quite more challenging:

$$\begin{cases} rank(\boldsymbol{A}_0) = 0.05m \\ \|\boldsymbol{E}_0\|_0 = 0.10m^2 \end{cases} \quad \textbf{(4.82)}$$

The respective results of each one scenario are reported in the following tables:

Table 3: Comparison between different algorithmic schemes on the RPCA problem-Scenario 1

Dimension m	Algorithm	$\frac{\|A-A_0\|_F}{\|A_0\|_F}$	$\frac{\|E-E_0\|_F}{\|E_0\|_F}$	$rank(A)$	$\|E\|_0$	# SVDs	# Iterations	Time(s)
500	SVT	8.75×10^{-6}	9.09×10^{-7}	25	12512	3689	3689	835.92
	APG	9.23×10^{-6}	7.34×10^{-7}	25	12510	251	128	28.66
	PAPG	8.50×10^{-6}	7.32×10^{-7}	25	12510	251	128	24.44
	DM	8.01×10^{-6}	9.00×10^{-7}	25	12500	1461	256	120.65
	EALM	8.18×10^{-7}	2.00×10^{-7}	25	12500	28	5	3.26
	IALM	4.76×10^{-6}	1.02×10^{-6}	25	12498	15	15	1.23
	ADM	5.24×10^{-6}	9.26×10^{-7}	25	12498	39	39	3.69
1000	SVT	3.48×10^{-6}	1.00×10^{-6}	50	50122	7446	7446	11536.43
	APG	6.40×10^{-6}	7.18×10^{-7}	50	50104	251	128	169.58
	PAPG	6.71×10^{-6}	7.98×10^{-7}	50	50149	249	127	96.91
	DM	3.24×10^{-6}	9.98×10^{-7}	50	49993	3304	470	1347.00
	EALM	6.50×10^{-7}	2.15×10^{-7}	50	49996	28	5	17.46
	IALM	1.31×10^{-6}	4.07×10^{-7}	50	49993	17	17	5.25
	ADM	3.17×10^{-6}	1.04×10^{-6}	50	49991	50	50	28.35
1500	SVT	3.30×10^{-6}	1.04×10^{-6}	75	112501	6155	6155	29058.89
	APG	5.23×10^{-6}	7.16×10^{-7}	75	112683	252	128	608.61
	PAPG	5.48×10^{-6}	7.96×10^{-7}	75	112767	249	127	254.38
	DM	5.01×10^{-6}	8.08×10^{-7}	75	112489	7592	1004	9028.23
	EALM	4.00×10^{-7}	1.54×10^{-7}	75	112498	28	5	54.09
	IALM	2.27×10^{-6}	8.69×10^{-7}	75	112490	16	16	13.28
	ADM	2.47×10^{-6}	9.93×10^{-7}	75	112493	62	62	132.59

Table 4: Comparison between different algorithmic schemes on the RPCA problem-Scenario 2

Dimension m	Algorithm	$\frac{\|A-A_0\|_F}{\|A_0\|_F}$	$\frac{\|E-E_0\|_F}{\|E_0\|_F}$	$rank(A)$	$\|E\|_0$	# SVDs	# Iterations	Time(s)
500	SVT	1.22×10^{-5}	1.89×10^{-6}	31	25136	10000	10000	9160.13
	APG	1.15×10^{-5}	8.26×10^{-7}	25	25118	256	130	28.91
	PAPG	1.21×10^{-5}	9.17×10^{-7}	25	25145	254	129	24.92
	DM	6.54×10^{-6}	3.22×10^{-6}	25	24999	5507	630	620.83
	EALM	1.29×10^{-6}	1.82×10^{-7}	25	25000	32	5	3.64
	IALM	8.34×10^{-6}	1.01×10^{-6}	25	24997	17	17	1.44
	ADM	5.87×10^{-6}	7.56×10^{-7}	25	24997	108	108	10.26
1000	SVT	4.73×10^{-6}	1.07×10^{-6}	82	100504	10000	10000	21456.78
	APG			50	100365	256	130	172.11

	PAPG	8.13×10^{-6}	8.24×10^{-7}	50	101622	255	130	98.62
	DM	7.84×10^{-6}	8.30×10^{-7}	50	100002	6493	783	2510.69
	EALM	5.05×10^{-6}	1.05×10^{-6}	50	99995	32	5	19.79
	IALM	8.73×10^{-7}	1.75×10^{-7}	50	99990	17	17	5.12
	ADM	5.72×10^{-6}	1.03×10^{-6}	50	99988	99	99	55.08
		4.71×10^{-6}	1.02×10^{-6}					
1500	SVT	3.89×10^{-6}	1.02×10^{-6}	113	225532	10000	10000	49652.18
	APG	6.62×10^{-6}	8.20×10^{-7}	75	225722	256	130	624.77
	PAPG	6.48×10^{-6}	8.17×10^{-7}	75	225488	256	130	259.77
	DM	4.08×10^{-6}	9.61×10^{-7}	75	225001	9507	1110	15904.94
	EALM	6.16×10^{-7}	1.39×10^{-7}	75	224999	32	5	60.08
	IALM	3.95×10^{-6}	8.70×10^{-7}	75	224983	17	17	14.19
	ADM	3.84×10^{-6}	1.03×10^{-6}	75	224987	91	91	198.02

As it is obvious from the above tables, there are many interesting conclusions arising from both the scenarios examined. First of all, concerning the SVT algorithm, it should be mentioned that on the one hand it achieves good accuracy results for the low-rank component but on the other hand the time needed for such a result is prohibitive. The reason for that to happen is obviously that it has to compute a SVD per iteration, which makes its computational burden quite significant. As a result, as the dimension of the problem increases the algorithm does not scale well, leading to computational time of several hours for dimensions exceeding 1000 -which although is usually the case in practice.

As far as the APG algorithm is concerned, it achieves comparable results in terms of the reconstruction error of the low-rank component with those of the SVT algorithm. Nevertheless, it converges much faster than the SVT algorithm, as the continuation technique involved in it plays a crucial role towards that direction. Consequently, the total computational time is significant less than that of the SVT algorithm, as well as the number of iterations needed for convergence -especially when the dimension of the problem rises. We should also mention the fact that if we adopt the PAPG version of the APG algorithm, the computational time can be further reduced, as the SVDs per iteration are not computed exactly but partially. Although at the beginning the gain may seem meaningless, it becomes quite clearer as we augment the dimension of the problem.

Concerning the DM algorithm, it is obvious from both Tables 3 and 4 that its accuracy of recovering the low-rank component is better than that of the APG algorithm as well as that of the SVT one. Of course such an advantage has its trade-off, which in that case is depicted to the iterations needed for convergence and consequently the total computational time. As a result, the DM algorithm constitutes a good alternative if we care about accuracy, but not a good one if we do not have the time to achieve it.

In what has to do with the ALM algorithm, we should highlight the fact that it achieves the best performance concerning both the accuracy of recovering the low-rank component as well as the time required to achieve it. It is significantly faster than all the other algorithms, a fact that arises from the adoption of an ADM approach for exploiting the separable nature of the RPCA problem. Furthermore, the ALM method scales quite well as the dimension of the problem increases, requiring less than a minute to recover the solution for matrix dimensions exceeding 1000. We should not forget to mention here the fact that if we adopt the IALM version, the time needed for convergence is significantly reduced -as becomes clear from Tables 3 and 4. The reason for that to happen is that

the required number of partial SVDs for the IALM version is significantly less than that of the EALM one, leading to less computational time.

Finally, the ADM approach also behaves quite well in what has to do with both its accuracy and its computational time. More precisely, it overcomes the SVT algorithm as well the APG and the DM ones. At the same time, the time needed to achieve such a performance is also less than that of the aforementioned algorithms, making it a good alternative for tackling the RPCA problem. A fact that should be also mentioned here is that the ADM algorithm seems less accurate as well as slower than both the EALM and IALM algorithms. Such a thing holds true, but does not tell the whole truth: in fact, both versions of the ALM algorithm adopt the approach proposed in [72] for updating separately the low-rank and sparse components, but at the same time the involved step-size parameters are adaptively updated. Such a regime does not hold true in the ADM approach proposed in [72], in which the step-size parameter β has a constant value from the beginning of the algorithm -as it is obvious from **(4.69)**. Consequently, the accuracy of the recovered low-rank component and the time needed to achieve it are a little worse than those of the EALM and IALM algorithms. Although, the credits of treating separately the low-rank and the sparse components should be given to the ADM algorithm.

4.7 A Case Study: Image De-Noising

In the final section of this thesis, we chose to cope with the well-known problem of *Image De-Noising*. More precisely, we selected to formulate the image de-noising task in terms of the RPCA problem, and subsequently try to tackle it with the algorithmic methods described in this Chapter, in order to have a more descriptive view of their performance in a more realistic application than the initial-random experimental evaluation which took place in the previous section.

As it is known from the respective bibliography, image de-noising is the process of removing noise from a noise-infected image. In general, it is a quite challenging task, as there are many different types of noise that can affect an image either accidentally or on purpose. For example, in salt and pepper noise (sparse light and dark disturbances), pixels in the image are very different in color or intensity from their surrounding pixels; the defining characteristic is that the value of a noisy pixel bears no relation to the color of surrounding pixels. Generally this type of noise will only affect a small number of image pixels. When viewed, the image contains dark and white dots, hence the term salt and pepper noise. As another example, we could refer to the probably most well-known type of noise, the Gaussian noise: in Gaussian noise, each pixel in the image will be changed from its original value by a (usually) small amount. It owes its name to its histogram visualization, a plot of the amount of distortion of a pixel value against the frequency with which it occurs, which in that case shows a normal distribution of noise. Among many reasons that have made it so popular (i.e. easy mathematical manipulation), probably the most important one comes from the field of statistics, and has the following intuitive interpretation: while other distributions are possible, the Gaussian (normal) distribution is usually a good model, due to the *Central Limit Theorem* that states that the sum of different noises tends to approach a Gaussian distribution.

Of course, there are many algorithms designed to tackle these problems -each one trying to take advantage from the specific application in which is used probably. What all algorithmic methods which cope with the image de-noising task weight in general, are the following factors:

- **Available time and computer power**: As an example to set our notation, we could think that it is much more difficult for a digital camera to apply image de-noising in

a fraction of a second using a tiny onboard CPU, than for a desktop computer which has much more power and time

- **Desired level of details**: In many cases we have to decide whether sacrificing some real detail is acceptable if it allows more noise to be removed (in other words, we have to decide how "aggressive" we should be vis-à-vis variations in the image that can be treated as noise)

4.7.1 Simulation Conditions

Taking the above information into mind, we decided to formulate the image de-noising pursuit in terms of the RPCA problem as follows: we select a low-rank image (or an approximately one) to be our low-rank component $\boldsymbol{A}_0 \in R^{m\times m}$. Afterwards, we contaminate it with outlier noise ($\boldsymbol{E}_0 \in R^{m\times m}$) selected uniformly at random -exactly as in the previous experimental section. The "sparsity level" for this component was chosen as in **(4.81)** to be $\|\boldsymbol{E}_0\|_0 = 0.05m^2$. As a result, their superposition data matrix $\boldsymbol{D}$ is going to be the noise-infected image which we seek to de-noise.

Mainly for time-saving reasons we chose to address this problem via the use of the three fastest algorithmic methods described in the previous section (i.e. EALM, IALM, and ADM). The respective parameters remain the same as was the case before, mostly for "fairness reasons". In what has to do with the KPIs used for the upcoming ranking of the algorithms, they also remain the same with those provided in Tables 3 and 4, but due to the application studied an additional one was also put into the frame: the *Peak-Signal-to-Noise-Ratio* (PSNR). PSNR is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale. PSNR is most commonly used to measure the quality of reconstruction of lossy compression codecs (i.e. image compression). The signal in this case is the original data, and the noise is the error introduced by compression. In our case, the noise-free signal is going to be the original images to be tested each time, while the noise term is going to be dominated by the sparse outliers.

In mathematical terms, the PSNR metric is defined via the MSE between our original noise-free image (NFI) and its noisy approximation (NAI):

$$MSE = \frac{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[NFI(i,j)-NAI(i,j)]^2}{mn} \quad \textbf{(4.83)}$$

Then the PSNR (in dB) is then defined as follows:

$$PSNR = 10\log_{10}\left(\frac{(max_{NFI})^2}{MSE}\right) \quad \textbf{(4.84)}$$

, where max_{NFI} is the maximum possible pixel value of the image. In our experiments we coped with square images (m=n), and the pixels are represented using 8 bits per sample. As a result, max_{NFI} value is going to be 255.

The selection of the PSNR metric was based on the one hand on its relative easy computation, while on the other hand when comparing image de-noising algorithms the PSNR metric is an approximation to human perception of reconstruction quality. In other words, this metric gives an intuitive argument of whether the performance of a specific algorithm was satisfactory or not.

For the better understanding of the algorithms' behavior under different real-application circumstances, we sketched two different scenarios: the noiseless and the noisy one. Both of them are described in details in the upcoming sections, together with their results and the respective conclusions occurring from them.

4.7.2 Noiseless Scenario

In the first scenario, the experimental regime is the one described just above. In other words, the data matrix $\boldsymbol{D}$ is the superposition of the original image and the outlier noise added to it. The test set included three different images, in terms of image contrast as well as image resolution. In plain English, we used images of different dimensions and different "level of low-rankness". More precisely, the first test-image was one with resolution of 512×512 while the upcoming two had a double resolution each time (1024×1024 and 2048×2048 respectively). In order to provide a better understanding of the whole de-noising process, for each one of them, we present the following figures:

- **Original image**: The original image to be infected with outlier noise
- **Noisy image**: The noise-infected image to be de-noised by the three different algorithmic methods
- **Clean-Reconstructed image**: The de-noised image produced by each one of the three algorithmic methods
- **Estimated outlier noise**: The portion of the outlier noise captured as the sparse component by each one of the three algorithmic methods

Furthermore, the performance of each one of the three used algorithmic methods, according to the KPIs mentioned above, is recorded and subsequently plotted versus the number of iterations needed for convergence -in order to have a visualization of their relative changes during the de-noising process. The respective results are summarized into a, similar to Tables 3 and 4, result-table -which will help in the conclusion extraction process and therefore is presented at the end of the whole process.

For the 512×512 image resolution case, the selected image was one depicting one of the famous San Francisco bridges, and is shown below:

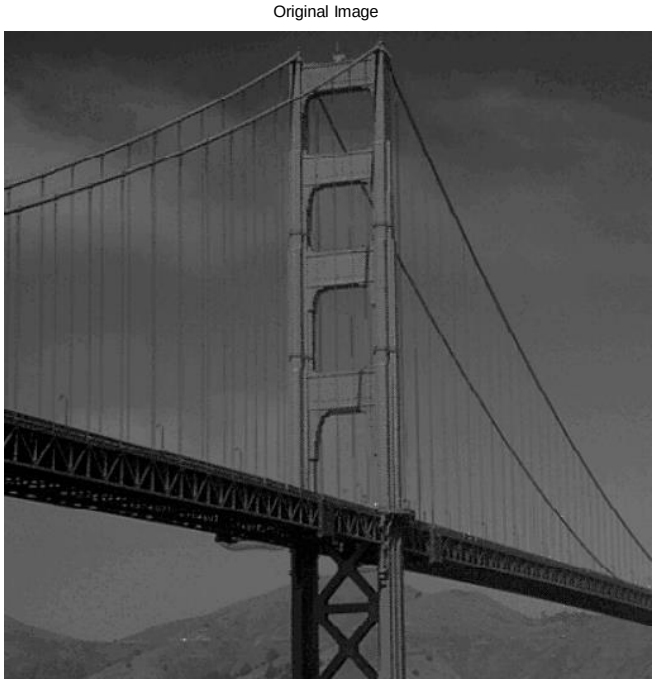

Image 16: Original Image-Bridge

Its noisy counterpart is then depicted in the following image:

Noisy Image

Image 17: Noisy Image-Bridge

The task of de-noising the above noisy image is tackled by the three fastest algorithms examined in this Chapter, as mentioned earlier. For the EALM algorithm, the reconstructed image as well as the estimated outlier noise are depicted in the following images:

Reconstructed Image-EALM

Image 18: Reconstructed Image-Bridge-EALM

Outliers-EALM

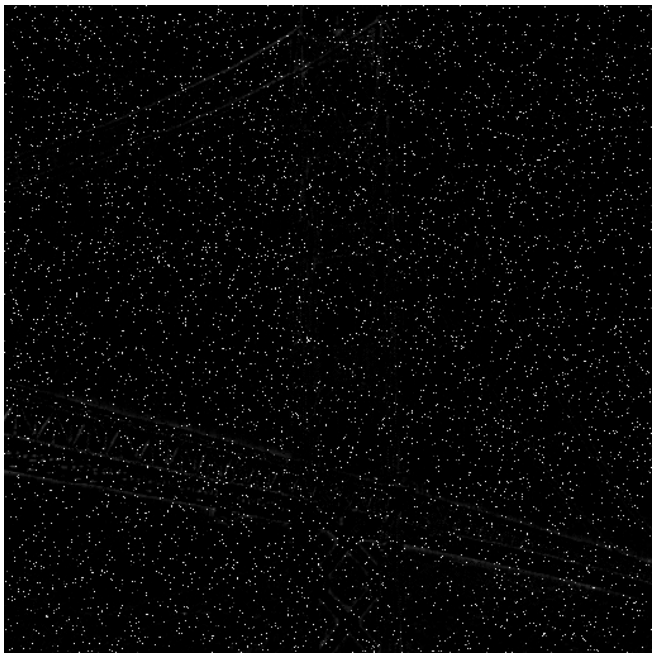

Image 19: Outliers-Bridge-EALM

The respective images for the IALM algorithm are the following ones:

Reconstructed Image-IALM

Image 20: Reconstructed Image-Bridge-IALM

Outliers-IALM

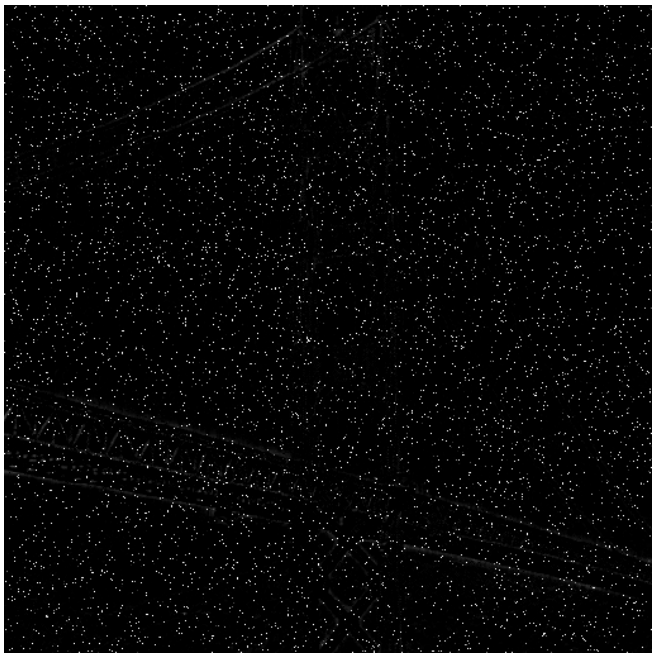

Image 21: Outliers-Bridge-IALM

As far as the ADM algorithm is concerned, its results are shown in the upcoming images:

Reconstructed Image-ADM

Image 22: Reconstructed Image-Bridge-ADM

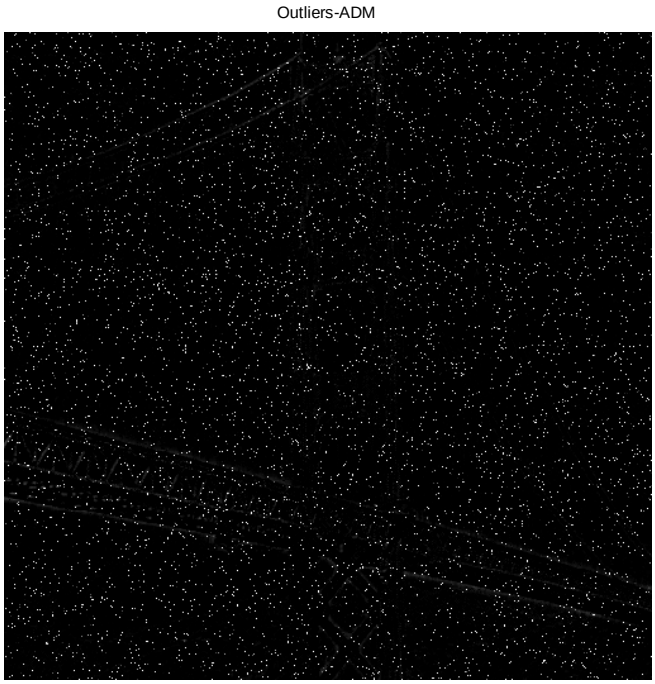

Image 23: Outliers-Bridge-ADM

After the image de-noising process figures, it is deemed appropriate to show the quality metrics used to measure the performance of the aforementioned algorithms. More precisely, the relative error for the low-rank component (i.e. the reconstructed image) is shown in the following figure:

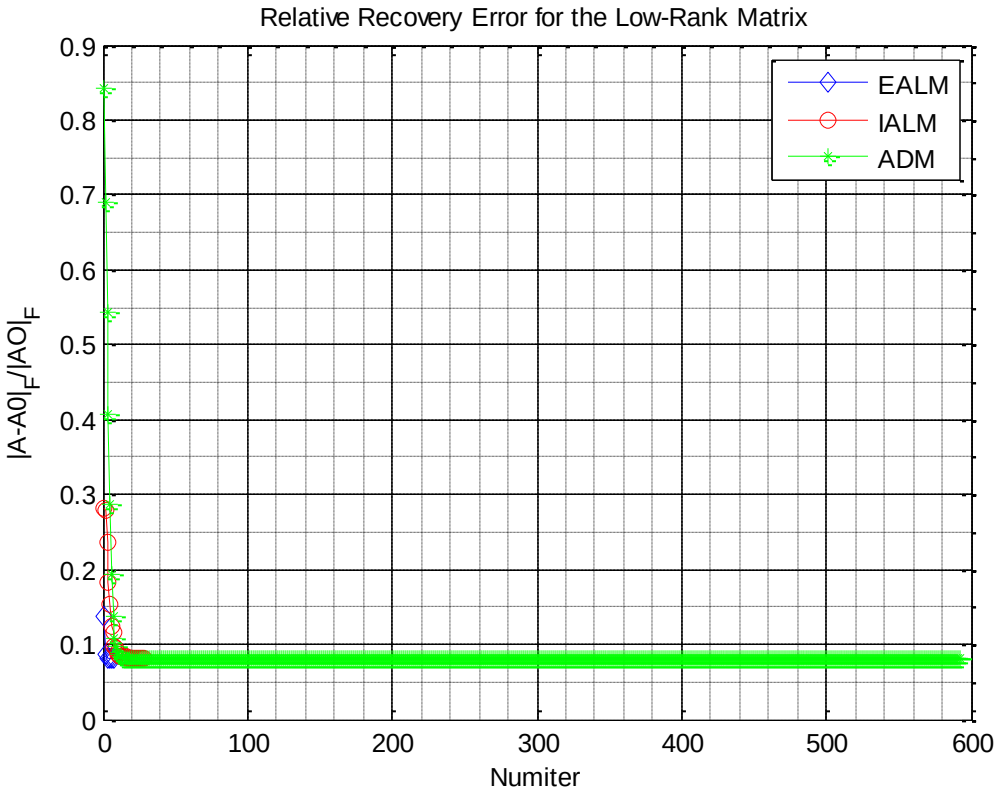

Figure 2: RELR-Bridge

Subsequently, the relative error for the sparse component (i.e. the outliers) is depicted in the following figure:

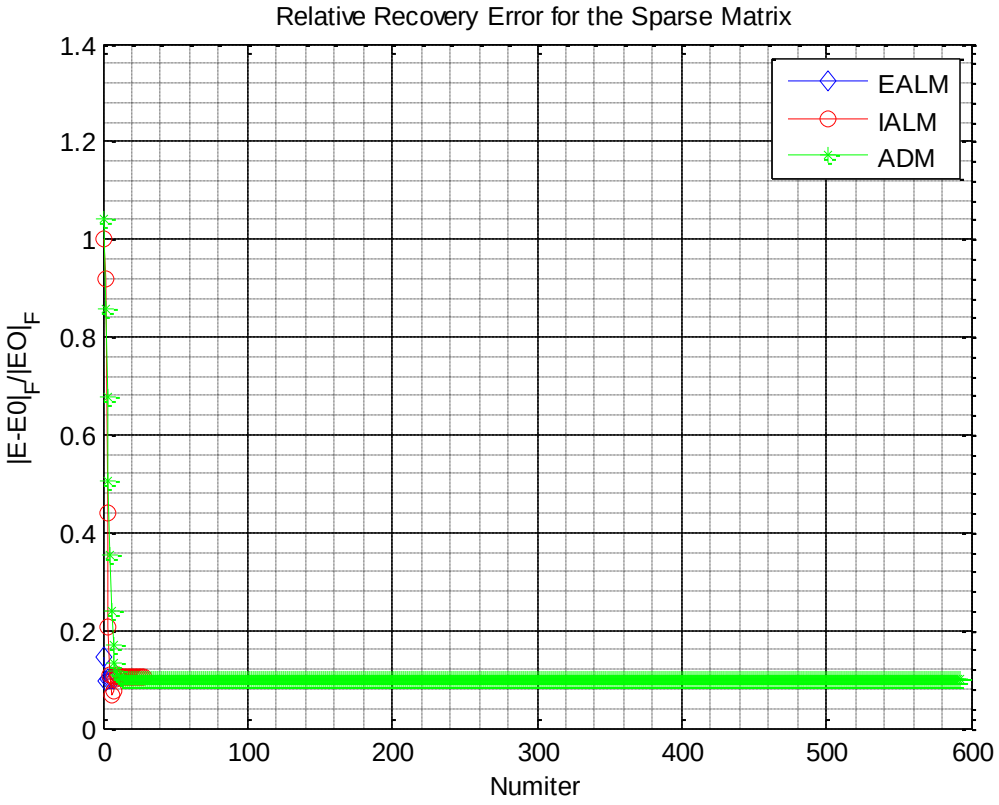

Figure 3: RES-Bridge

Furthermore, the rank of the low-rank component is shown below:

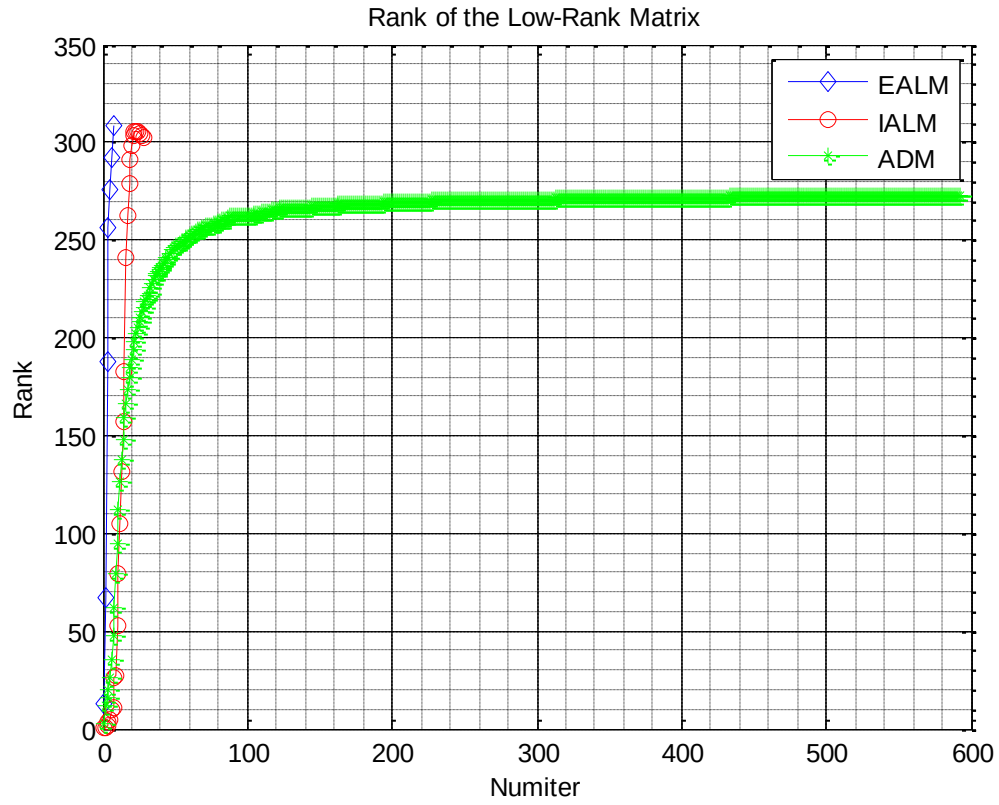

Figure 4: Rank-Bridge

At the same time, the cardinality of the sparse component is depicted below:

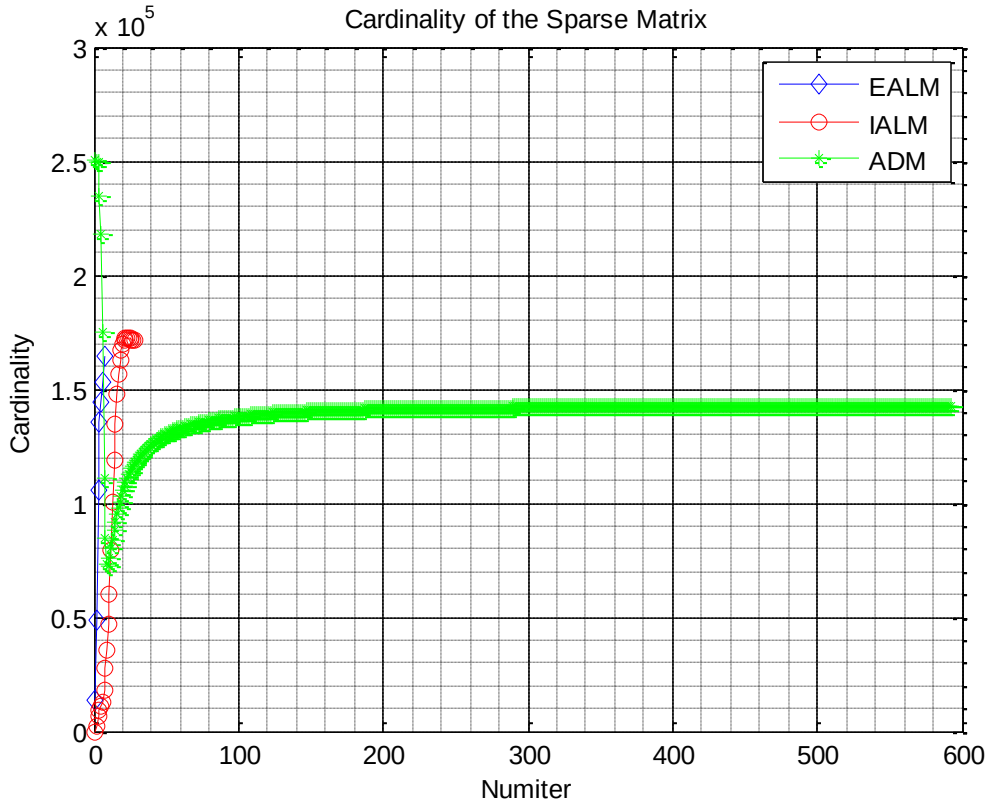

Figure 5: Cardinality-Bridge

Finally, the PSNR metric is depicted in the following figure:

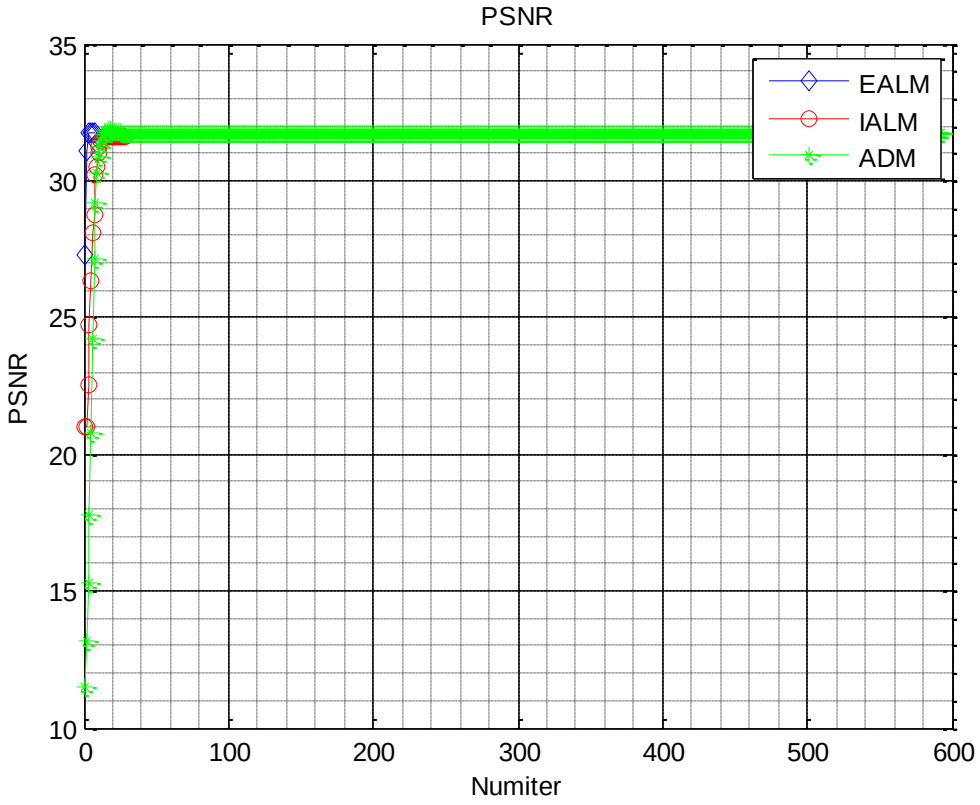

Figure 6: PSNR-Bridge

For the 1024×1024 image resolution case, the selected image was one depicting lights in a dark background, which is shown below:

Image 24: Original Image-Lights

Its noisy counterpart is then depicted in the following image:

Image 25: Noisy Image-Lights

For the EALM algorithm, the reconstructed image as well as the estimated outlier noise are depicted in the following images:

Image 26: Reconstructed Image-Lights-EALM

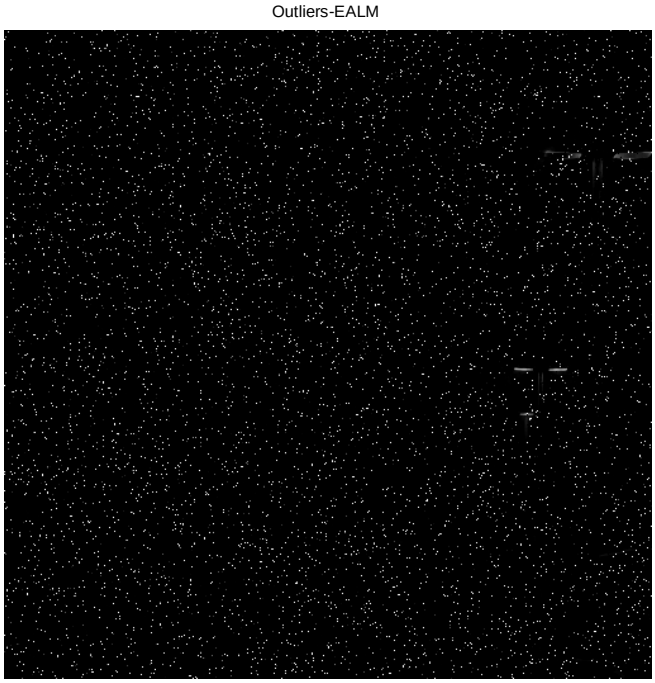

Image 27: Outliers-Lights-EALM

The respective images for the IALM algorithm are the following ones:

Image 28: Reconstructed Image-Lights-IALM

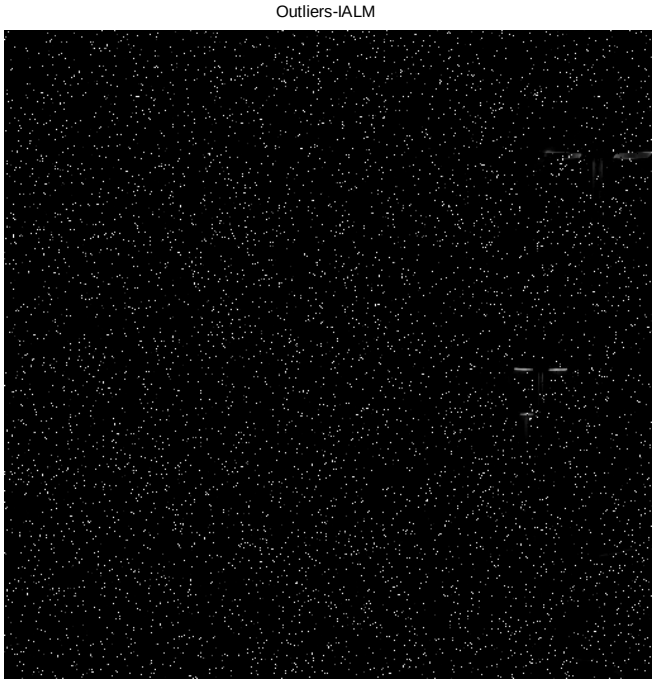

Image 29: Outliers-Lights-IALM

As far as the ADM algorithm is concerned, its results are shown in the upcoming images:

Image 30: Reconstructed Image-Lights-ADM

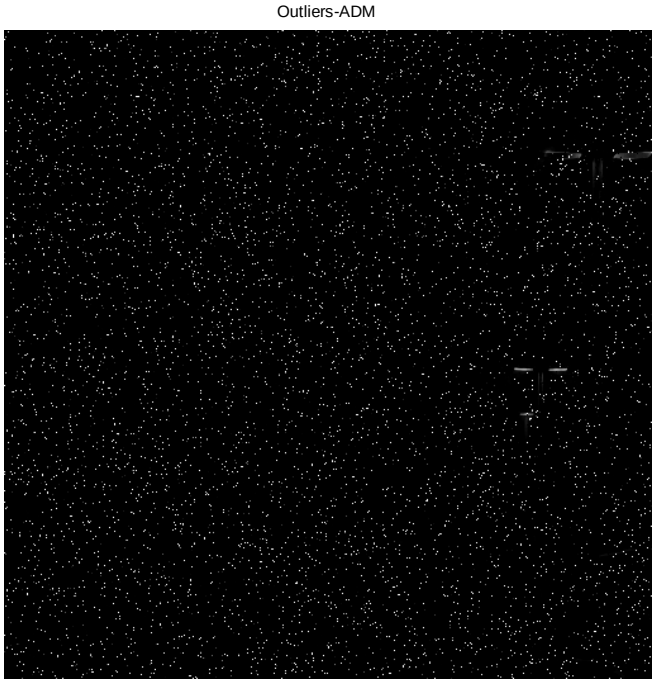

Image 31: Outliers-Lights-ADM

In what has to do with the performance metrics for this case, the relative error for the low-rank component is shown in the following figure:

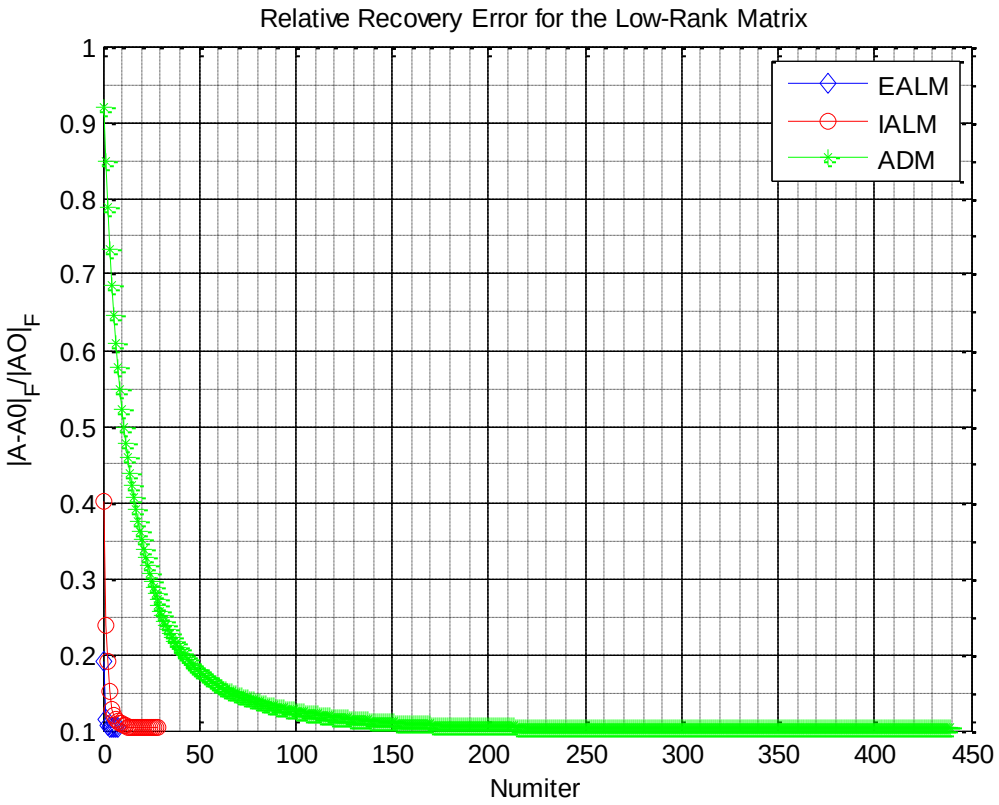

Figure 7: RELR-Lights

As for the relative error for the sparse component, it is depicted in the following figure:

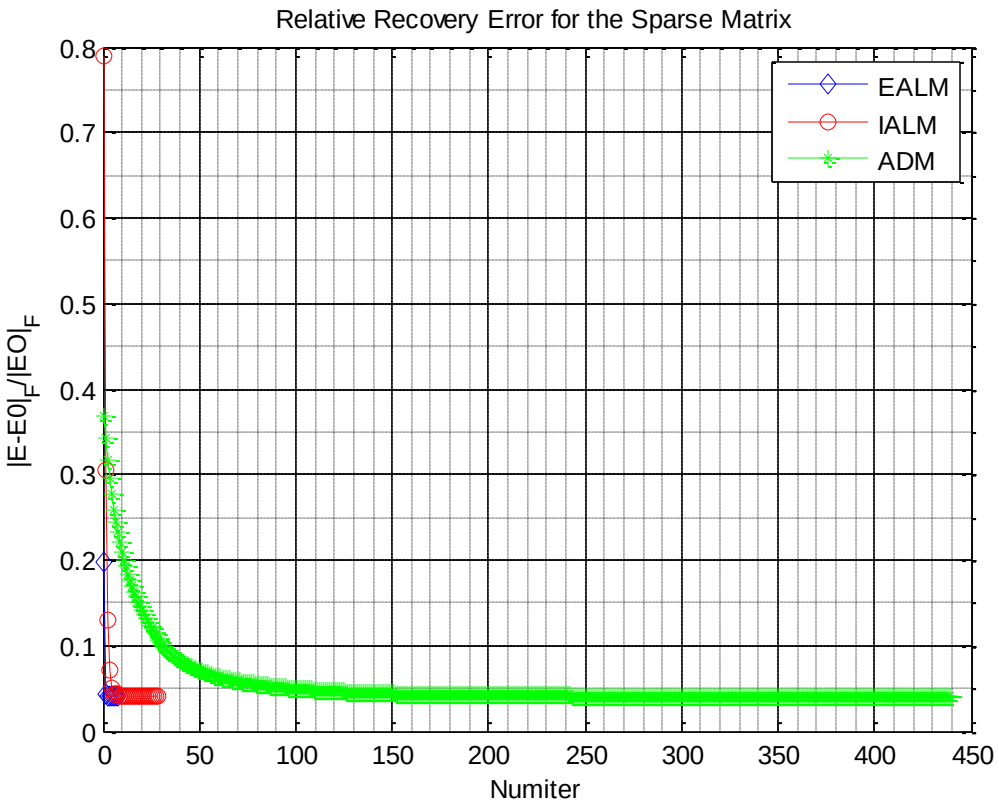

Figure 8: RES-Lights

Furthermore, the rank of the low-rank component is shown below:

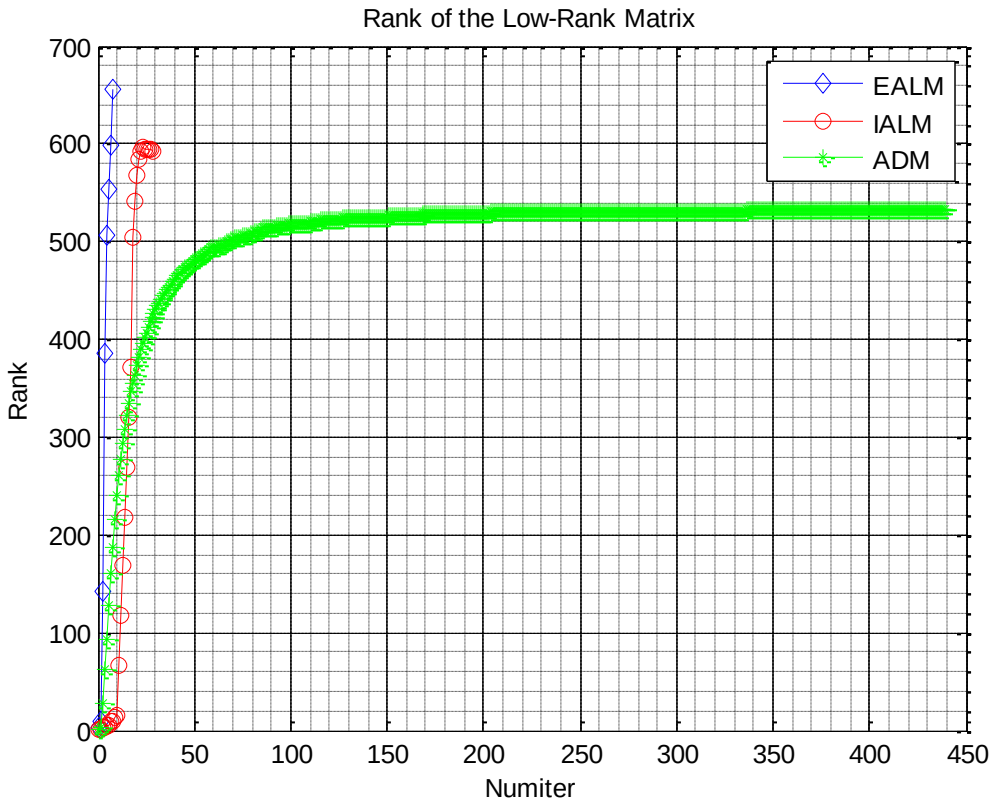

Figure 9: Rank-Lights

As for the cardinality of the sparse component, it is depicted below:

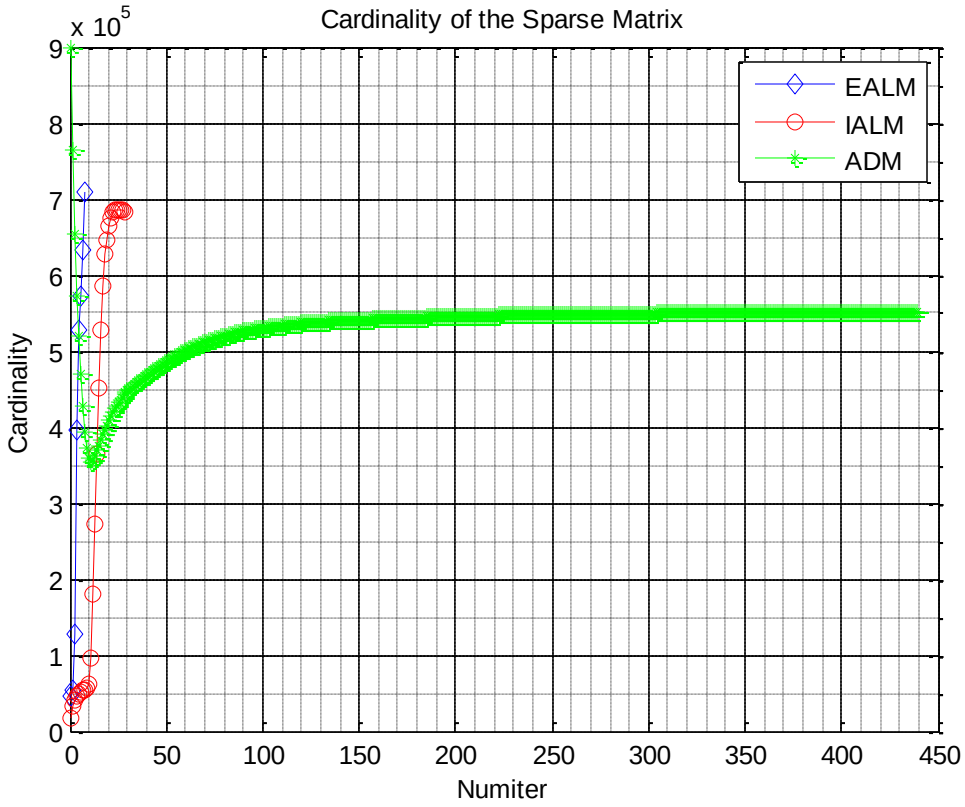

Figure 10: Cardinality-Lights

Finally, the PSNR metric is depicted in the following figure:

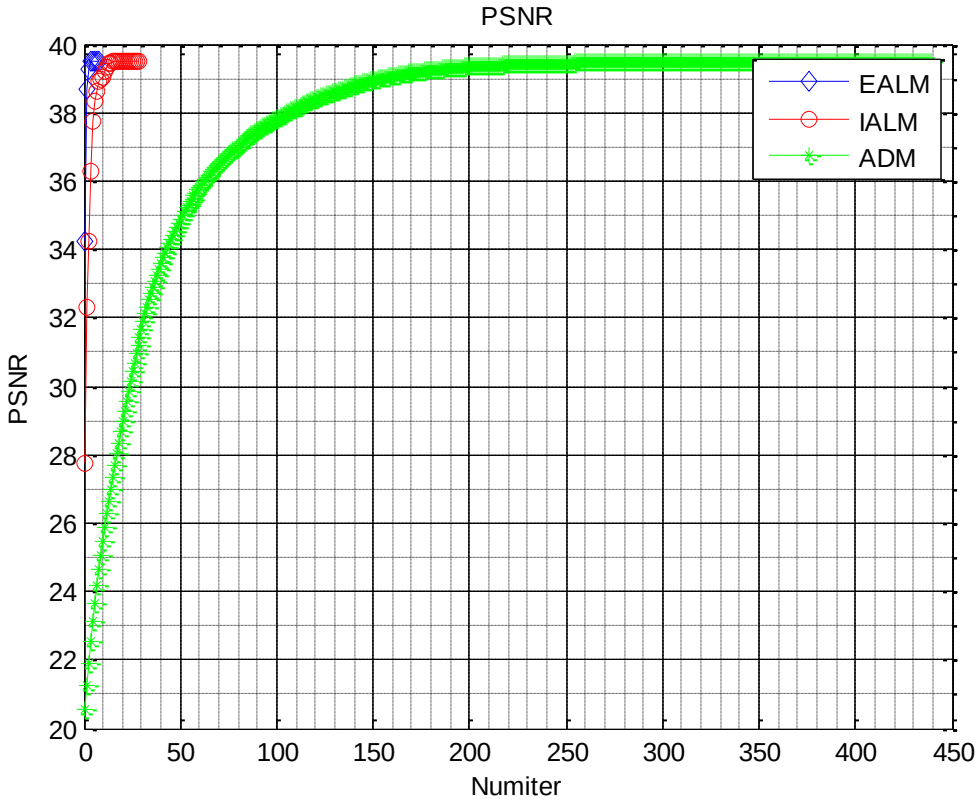

Figure 11: PSNR-Lights

Finally, in the case of the 2048×2048 image resolution, the selected image was one depicting stones in a woody background, which is shown below:

Image 32: Original Image-Stones

Following the same pattern as above, its noisy counterpart is depicted below:

Image 33: Noisy Image-Stones

The reconstructed image as well as the estimated outlier noise “produced” by the EALM algorithm are depicted below:

Image 34: Reconstructed Image-Stones-EALM

Outliers-EALM

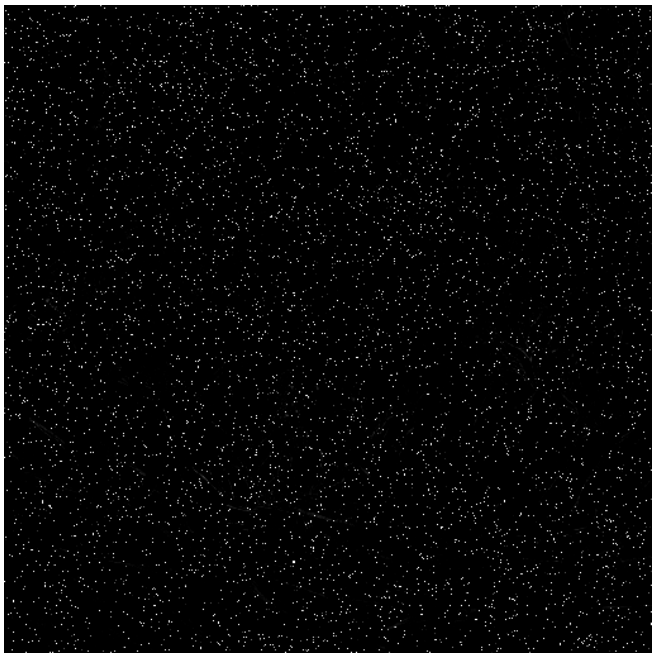

Image 35: Outliers-EALM-Stones

Furthermore, the respective images “produced” by the IALM algorithm are shown below:

Reconstructed Image-IALM

Image 36: Reconstructed Image-Stones-IALM

Outliers-IALM

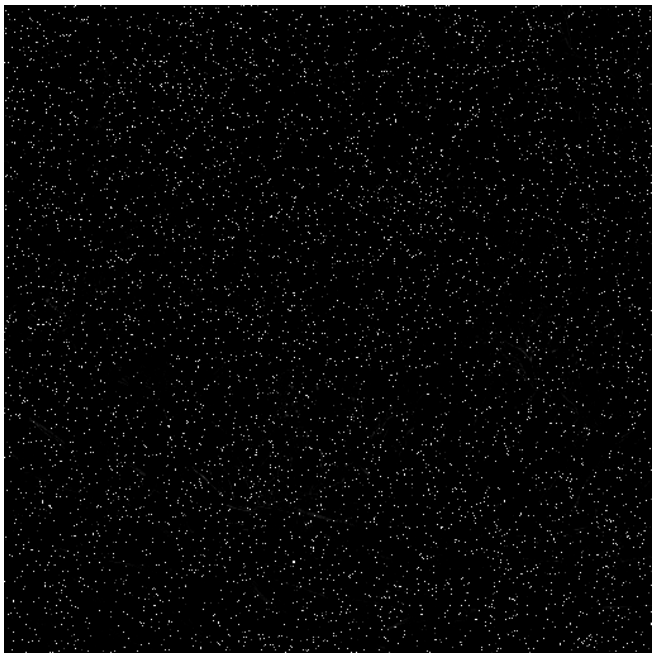

Image 37: Outliers-Stones-IALM

As for the ADM algorithm’s results, they are shown straightaway:

Reconstructed Image-ADM

Image 38: Reconstructed Image-Stones-ADM

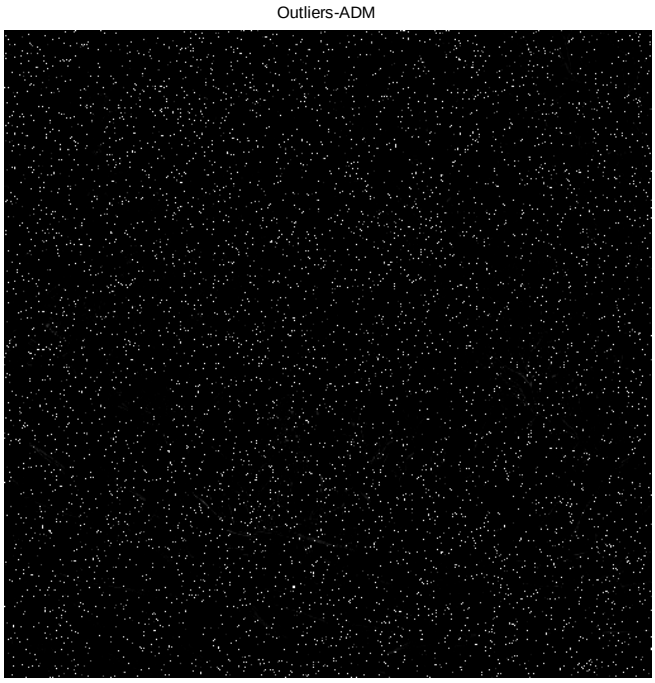

Image 39: Outliers-Stones-ADM

As far as the performance metrics for the highest-resolution case are concerned, the relative error for the low-rank component is depicted in the forthcoming figure:

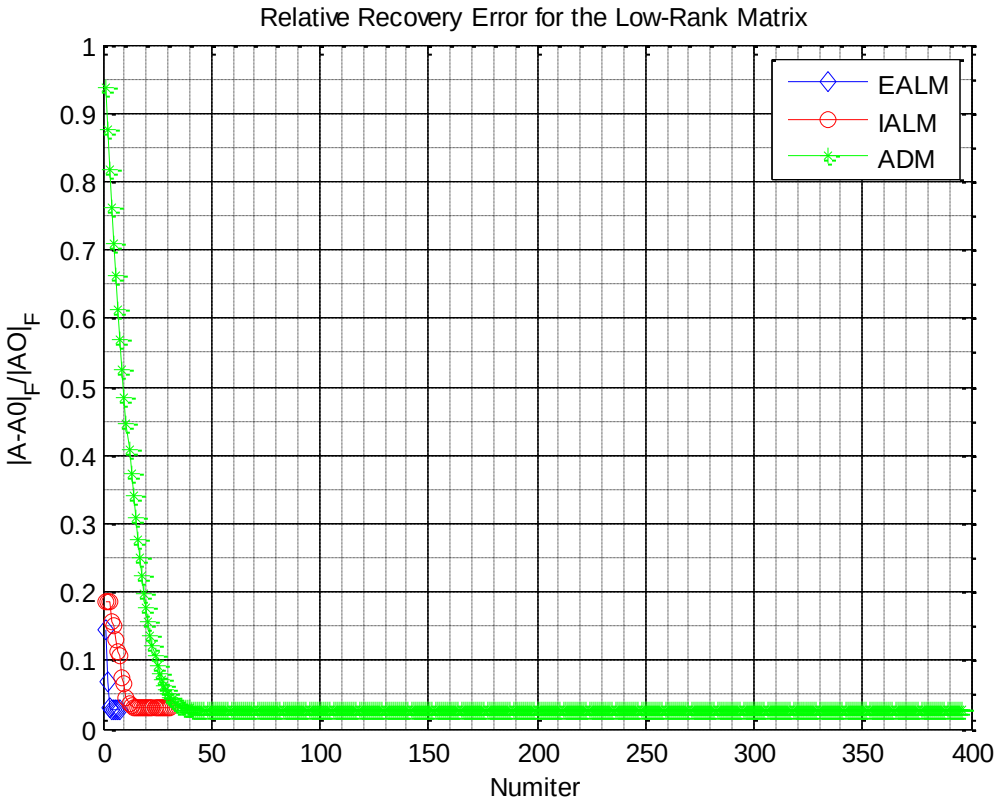

Figure 12: RELR-Stones

At the same time, the respective relative error for the sparse component is shown below:

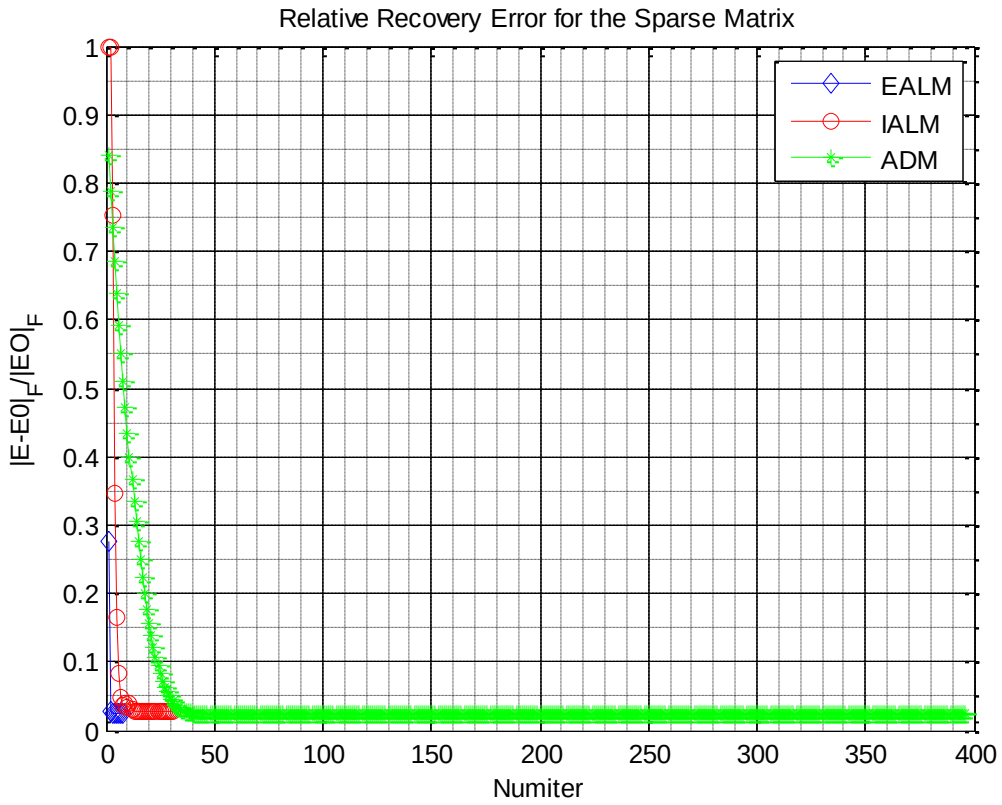

Figure 13: RES-Stones

The rank of the low-rank component follows up:

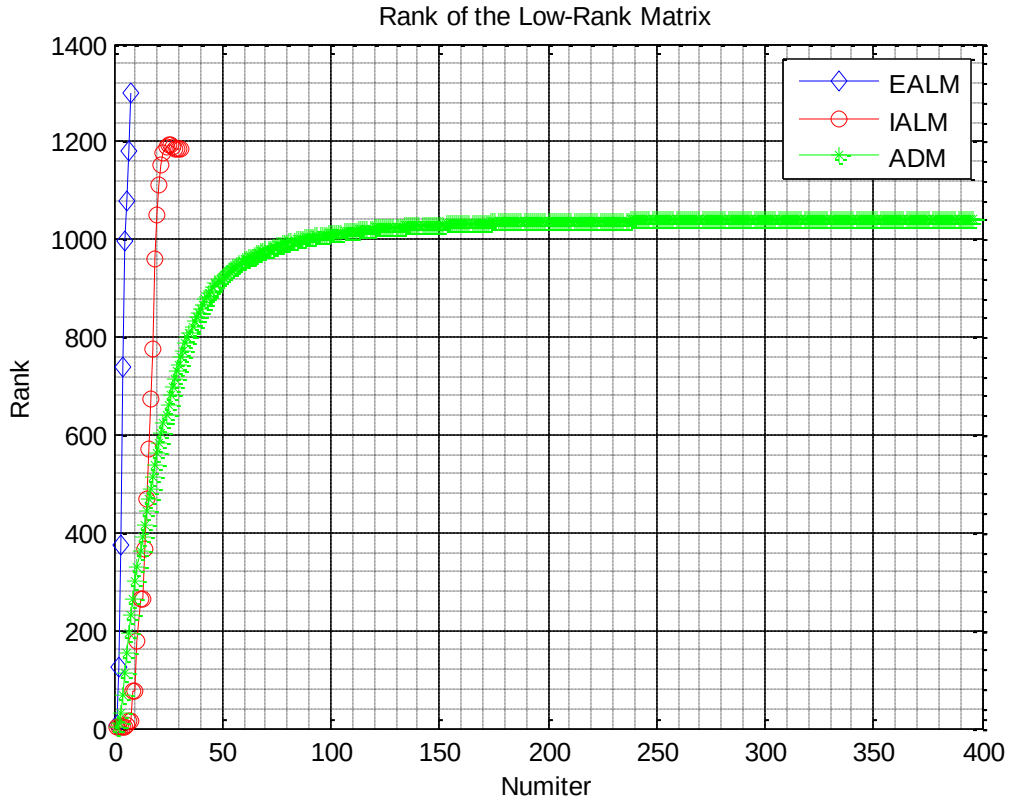

Figure 14: Rank-Stones

As for the cardinality of the sparse component, it is depicted in the following figure:

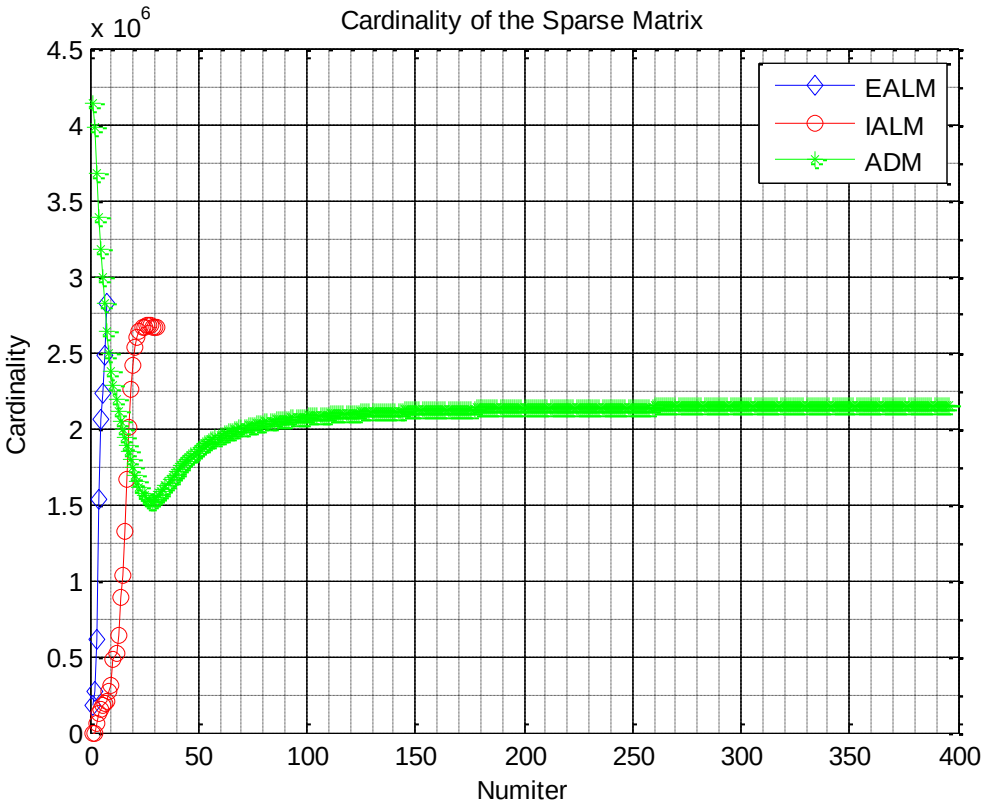

Figure 15: Cardinality-Stones

Finally, the PSNR metric is shown below:

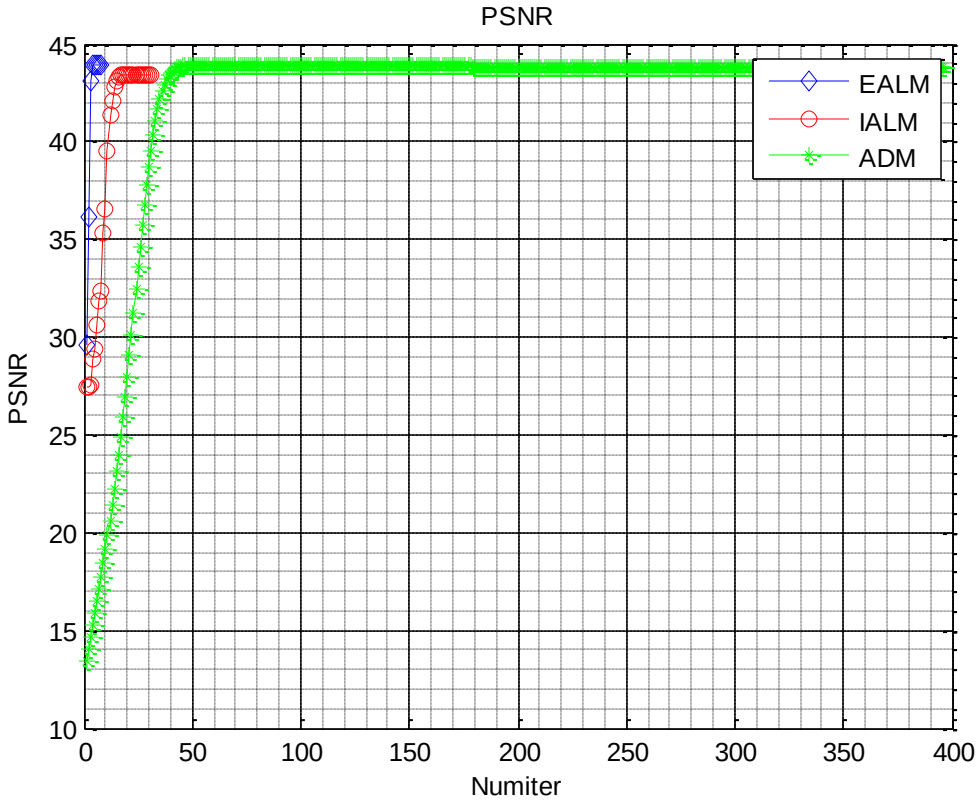

Figure 16: PSNR-Stones

The above results can be summarized in the following table, which contains all the useful information -for each different resolution image- which is needed to extract some initial

conclusions concerning the performance of the employed algorithms towards the image de-noising pursuit:

Table 5: Comparison between different algorithmic schemes on the Image De-noising pursuit- Noiseless Scenario

Image Resolution	Algorithm	$\frac{\|A-A_0\|_F}{\|A_0\|_F}$	$\frac{\|E-E_0\|_F}{\|E_0\|_F}$	$rank(A)$	$\|E\|_0$	PSNR	# SVDs	# Iterations	Time(s)
512×512	EALM	8.14×10^{-2}	1.02×10^{-1}	309	164646	31.764	756	7	180.16
	IALM	8.26×10^{-2}	1.03×10^{-1}	302	171560	31.632	29	29	4.12
	ADM	8.14×10^{-2}	1.02×10^{-1}	273	142541	31.766	593	593	67.24
1024×1024	EALM	1.04×10^{-1}	4.20×10^{-2}	656	710035	39.491	612	8	1163.90
	IALM	1.04×10^{-1}	4.19×10^{-2}	593	683701	39.498	28	28	38.40
	ADM	1.04×10^{-1}	4.19×10^{-2}	534	551198	39.502	439	439	311.99
2048×2048	EALM	2.81×10^{-2}	2.53×10^{-2}	1300	2821847	43.890	523	8	9427.62
	IALM	2.95×10^{-2}	2.66×10^{-2}	1185	2666190	43.456	31	31	282.79
	ADM	2.82×10^{-2}	2.53×10^{-2}	1042	2149958	43.873	396	396	2573.47

If we take a deeper look at the aforementioned results, there are many interesting conclusions that can come up into surface.

First of all, it is quite evident that in terms of the relative error of the low-rank component, EALM and ADM achieve a slightly better performance than IALM. The situation is pretty much the same in terms of the relative error for the sparse component also, although the latter practically is a metric of less importance as explained also in the previous section of this thesis. So if we care about more accurate results, picking one method among EALM and ADM would seem a reasonable choice. The gain although does not seem so astonishing.

In what has to do with the rank of the recovered solution, clearly the ADM method is the leading one. The most interesting remark though concerning this metric is that as the resolution of the image increases so does the gap between the “low-rankness” level of the three examined methods. As a result, for handling images of even higher resolution, it is reasonable to expect that the recovered solution obtained by the ADM method is going to be the lowest-rank one, whose “gain” to that metric vis-à-vis the other three methods is going to be proportional to the increase of the image resolution.

As far as the sparsity level of the obtained sparse component is concerned, even if its utility is not as important as the above metric due to the conceptual structure of the RPCA problem (recover a low-rank matrix from gross errors), the situation bears a strong resemblance to that of the rank metric. In other words, the ADM method recovers the sparsest solution followed by IALM and EALM. The difference of the sparsity level among them is again proportional to the image resolution augmentation, which suggests that for images of even higher resolution than those examined in the present thesis the regime is going to ameliorate in favor of the ADM method.

In terms of the PSNR metric, the difference among the algorithms are almost insignificant. Such a thing can be proven either by observing the respective values of the metric at each different image resolution scenario (which differ either on the first or the second decimal digit) or by observing the respective figures of the PSNR metric in which all three algorithms reach almost the same final value. Furthermore, the reconstructed images obtained by each algorithm bear a strong resemblance, which is indicative of the aforementioned argument. What should be highlighted at this point though is the fact of the clear improvement of the distinctive ability of the processed images as we increase

the resolution of the test-image. Furthermore, to this direction contributes also the higher-level of “low-rankness” of a specific image, which results in better performance of the algorithms as the contrast among different objects of the image becomes less significant. Such a conclusion is depicted in terms of the PSNR metric, whose value has increased about 12dBs from the lowest image resolution case to the highest one. This metric is of crucial importance in image processing as it depicts the reconstruction quality of an algorithm, and its amelioration up to more than 10dBs must give the credits to the respective methods.

As far as the computational load is concerned, the number of SVDs required to obtain a specific solution reveals two main conceptual results: the first one is that clearly the IALM method is the less computational-thirsty one (with the respective number of SVDs staying almost constant as the dimension of the problem increases), while the second one is that the more we cope with images of low-contrast the better the performance of each algorithm is going to be. Such a result of course owes its validity again to the constructional nature of these algorithms for tackling problems which contain low-rank structures in general.

Moreover, another result occurring from the above experiments is that the number of required iterations for convergence stays pretty much the same for EALM and IALM methods with a slight increase if we double the image resolution. Although this is not the case with the ADM method, its performance ameliorates as the resolution of the image increases as well as its “low-rankness” level.

Last but not least, there are some interesting results occurring from the computational time needed for each algorithm to solve a problem of specific image resolution. If we extend this argument a bit further, this metric is quite an important one -as it indicates the performance of each algorithm in real applications where the lavishness of time is not always guaranteed or even accepted. More precisely, the first conclusion is that clearly IALM outperforms the other two methods -a difference which becomes quite evident in the case of a 2048×2048 image resolution where the time required by IALM is up to 4.7 minutes while for ADM approaches 42 minutes and for EALM 2.5 hours.

The most interesting results although occur if we take a deeper look to the time needed for each algorithm to transit from a lower-resolution image to a higher one as a relative ratio: we observe that the EALM method requires about 6.4 times the time needed to solve to 512×512 problem for coping with the 1024×1024 one, a situation which gets a bit worse for the 2048×2048 case in which the respective ratio augments up to 8.1 times. At the same time, the regime for the ADM method is 4.6 times for the transition from 512×512 to 1024×1024 and 8.2 times for that from 1024×1024 to 2048×2048. On the contrary, IALM may initially require 9.5 its first computational time for the first transition, but as the dimension of the problem augments, this ratio is reduced to 7.4 times. As a result, we could say that as the image resolution increases so does the gap between the time needed for each algorithm to solve the problem. Consequently, the adaptive nature of the IALM method (in terms of the step-sizes) combined with its non-exact computation of the updates for the low-rank and sparse components can be proved very efficient for tackling real application problems even for high-resolution images that lie into thousands of dimensions.

Another interesting comment could be made about the relative speed of the fastest method (i.e. IALM) compared to the other 2 as we transit to higher-resolution images. In plain English, we observe that as for the 512×512 problem IALM is about 45 times faster than EALM and 16 times faster than ADM, this relative ratio decreases as we double the resolution of the image to 1024×1024 to 30 times for EALM and 8 times for ADM. Further increase of the image resolution up to 2048×2048 does not change the situation

significantly, as the relative time ratios slightly augment up to 33 times and 9.1 times respectively. As a consequence, we could say that as we increase the dimensionality of the problem, the two slowest methods (i.e. ADM and EALM) will retain their current performance compared to the fastest one (i.e. IALM) as their dependence seems to look like a linear one with different scaling factors, as well as their relative ranking based on the time metric.

4.7.3 Noisy Scenario

In the scenario studied in the above section, we supposed that our image was corrupted only by outlier noise of very intense magnitude. However, in many real world applications in the field of image processing, the observed image is often corrupted with noise (which may be stochastic or deterministic) affecting every pixel of the image. Therefore, for the techniques studied in the present thesis to be a good alternative for image de-noising purposes, results that examine stability and accurate recovery in the presence of entry-wise noise must be provided.

The whole experimental concept remains pretty much the same as in the aforementioned scenario, but now the measurement model includes another one term standing for the extra entry-wise noise added to each pixel of the image. In mathematical terms, our data matrix is going to have to following form:

$$\boldsymbol{D} = \boldsymbol{A}_0 + \boldsymbol{E}_0 + \boldsymbol{P}_0 \quad \textbf{(4.85)}$$

, where $\boldsymbol{A}_0$ and $\boldsymbol{E}_0$ represent once more the low-rank and the sparse component respectively, while $\boldsymbol{P}_0$ is a small-perturbation noise term.

From a theoretical point of view, the significance of the validity of the model described in **(4.85)** is quite important in the sense that it provides sophisticated as well as apt arguments for the stability of the RPCA decomposition of a matrix in its low-rank plus sparse components in the presence of small entry-wise noise. At the same time, the practical impact is also considerable as in many occasions imperfections that affect the whole available data may happen during the acquisition process -leading to situations that can be perfectly modeled like **(4.85)**. Of course, at this point of this thesis, the goal is by no means to provide theoretical results concerning the applicability of **(4.85)**. The diligent reader is referred to [74] for more details.

Before presenting the respective images for the present scenario, it is deemed appropriate to refer to the kind of perturbation noise term $\boldsymbol{P}_0$ mentioned in **(4.85)**. For our experimental purposes we assumed that each entry of $\boldsymbol{P}_0$ is i.i.d. Gaussian random variable with zero mean and 0.01 variance. This choice was made mainly to ensure that the portion of the entry-wise noise is not insignificant, and really affects the image, in order to be able to extract some useful conclusions about the performance of the three different employed algorithmic methods under more challenging situations.

Taking the above information into mind, we continue by presenting the de-noising process right away. For the 512×512 image resolution case, the original image which depicts one of the famous San Francisco bridges is shown below:

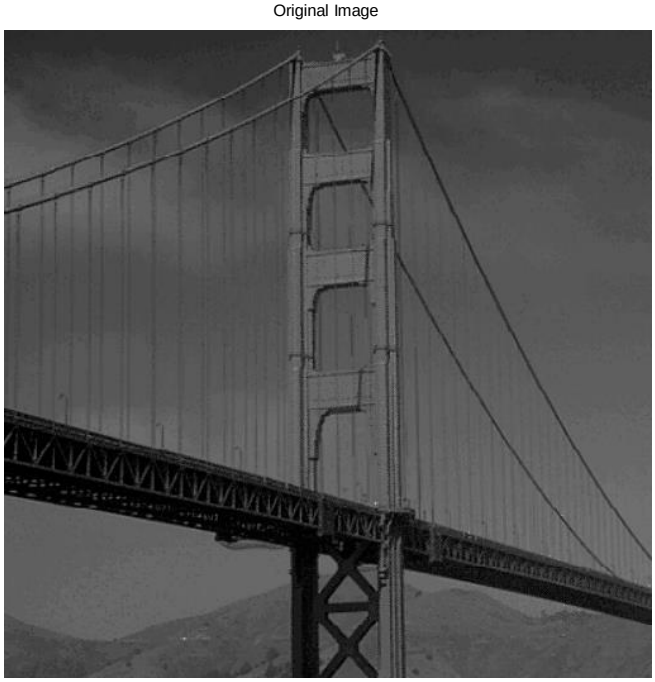

Image 40: Original Image-Bridge(Noisy)

Its noisy counterpart is then depicted in the following image:

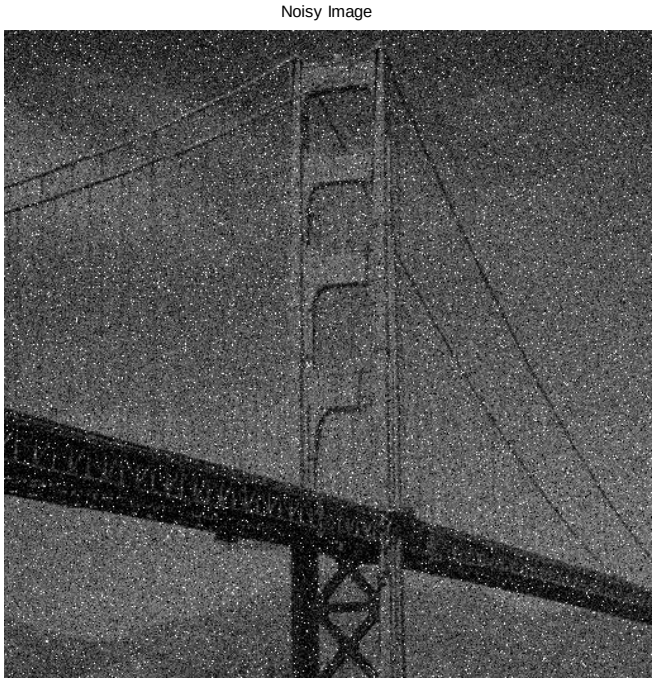

Image 41: Noisy Image-Bridge(Noisy)

For the EALM algorithm, the reconstructed image as well as the estimated outlier noise are depicted in the following images:

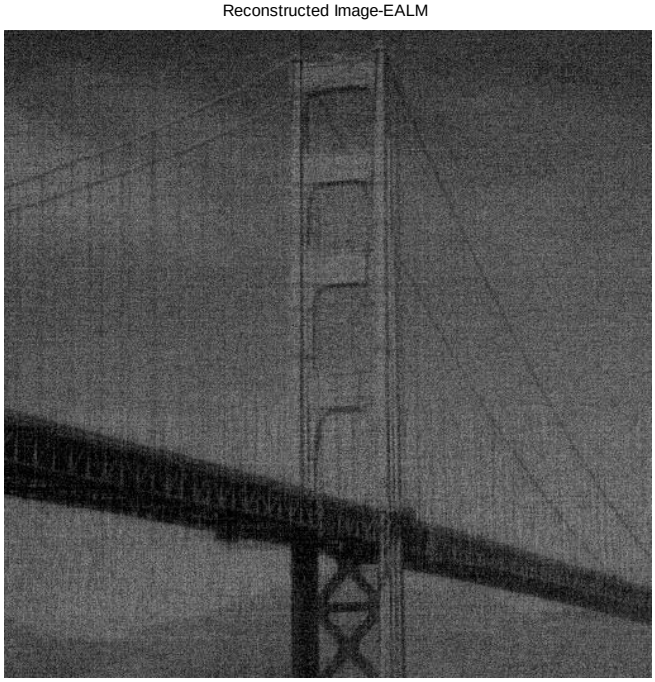

Image 42: Reconstructed Image-Bridge(Noisy)-EALM

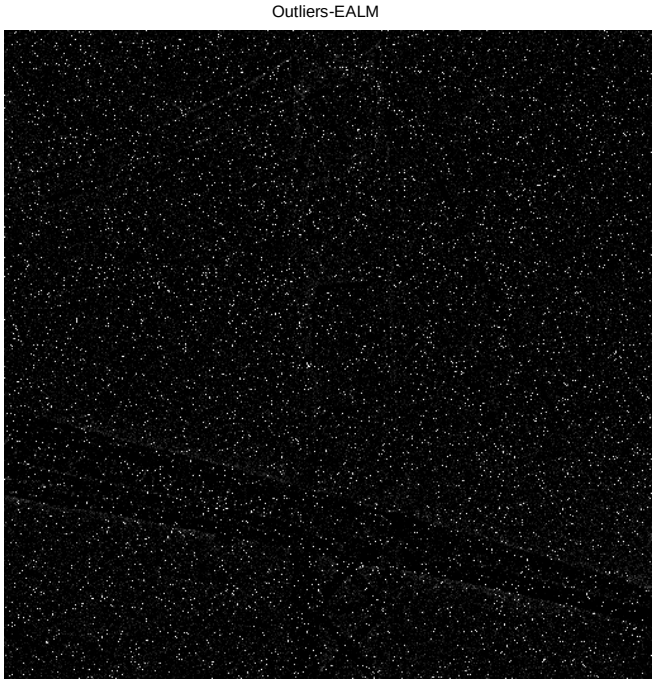

Image 43: Outliers-Bridge(Noisy)-EALM

The respective images for the IALM algorithm are the following ones:

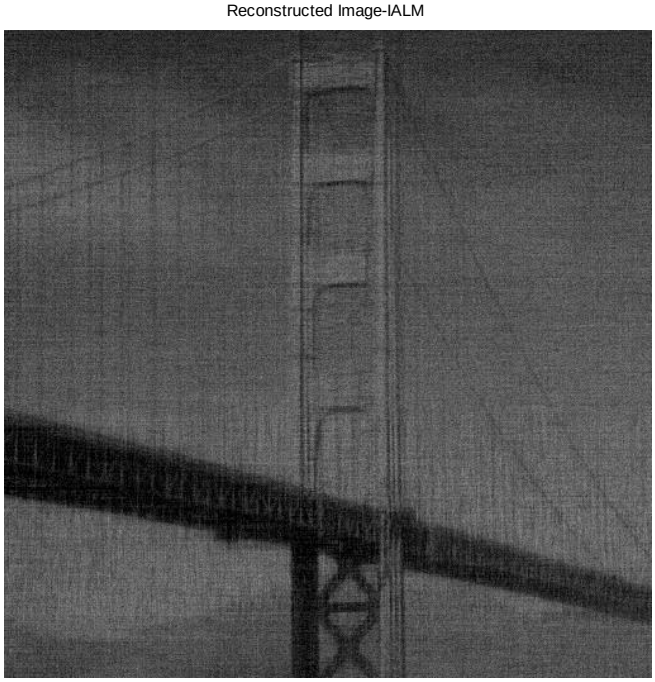

Image 44: Reconstructed Image-Bridge(Noisy)-IALM

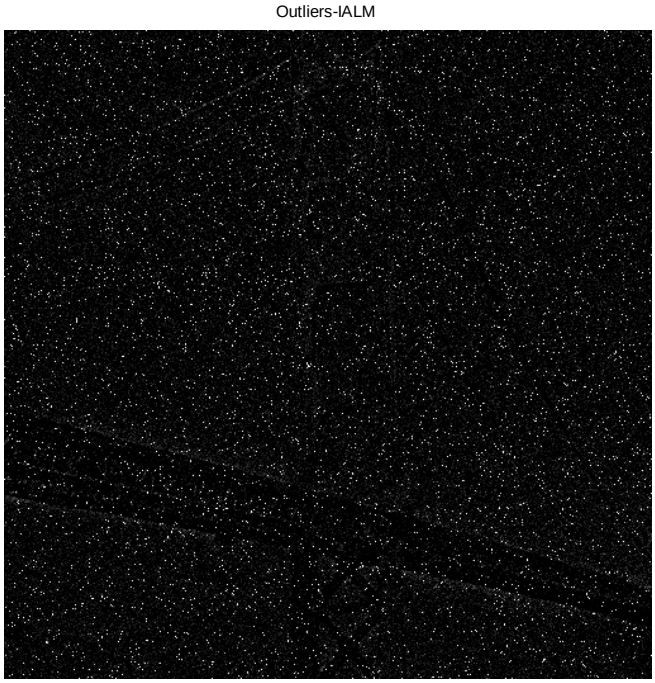

Image 45: Outliers-Bridge(Noisy)-IALM

As far as the ADM algorithm is concerned, its results are shown in the upcoming images:

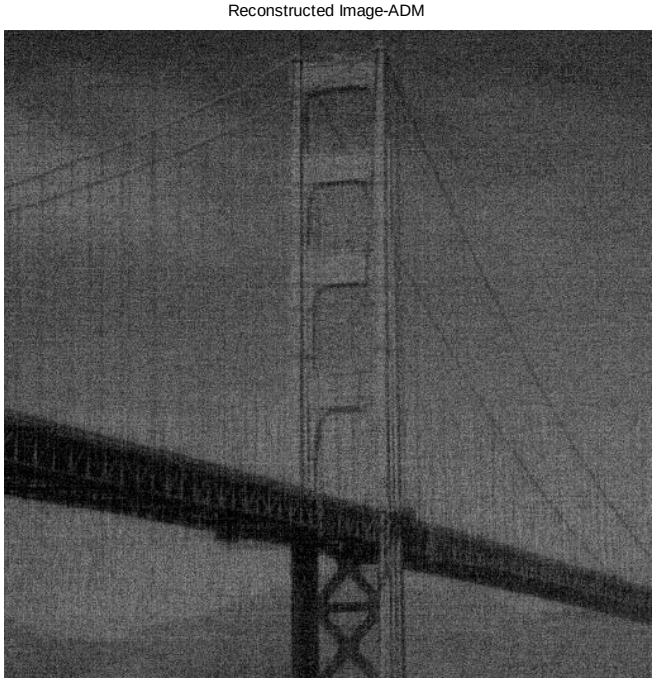

Image 46: Reconstructed Image-Bridge(Noisy)-ADM

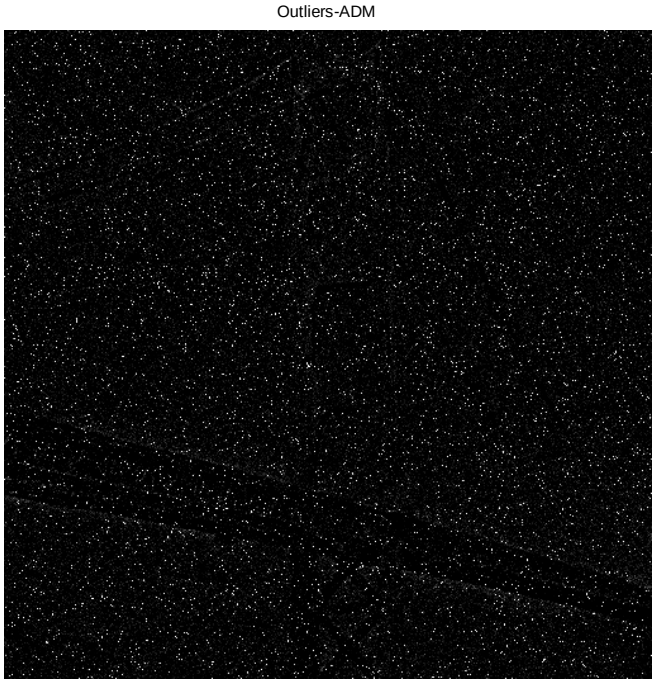

Image 47: Outliers-Bridge(Noisy)-ADM

Concerning the quality metrics used to measure the performance of the aforementioned algorithms in the present scenario, they are presented straight away. More precisely, the relative error for the low-rank component is shown in the following figure:

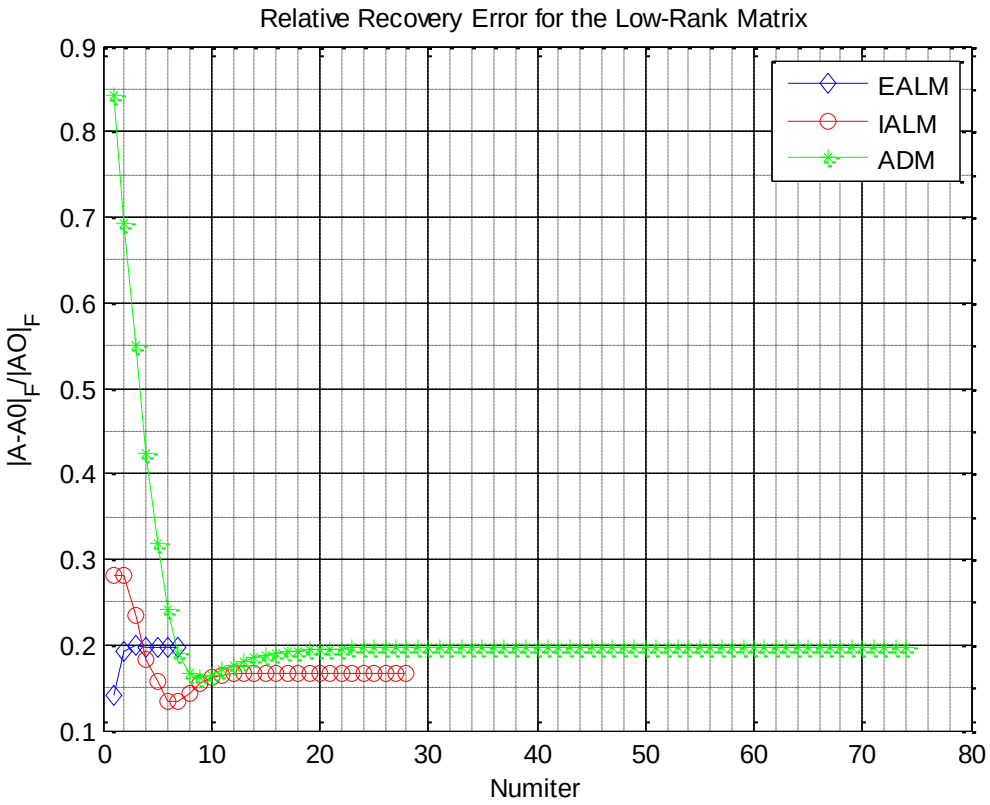

Figure 17: RELR-Bridge(Noisy)

Subsequently, the relative error for the sparse component is depicted in the following figure:

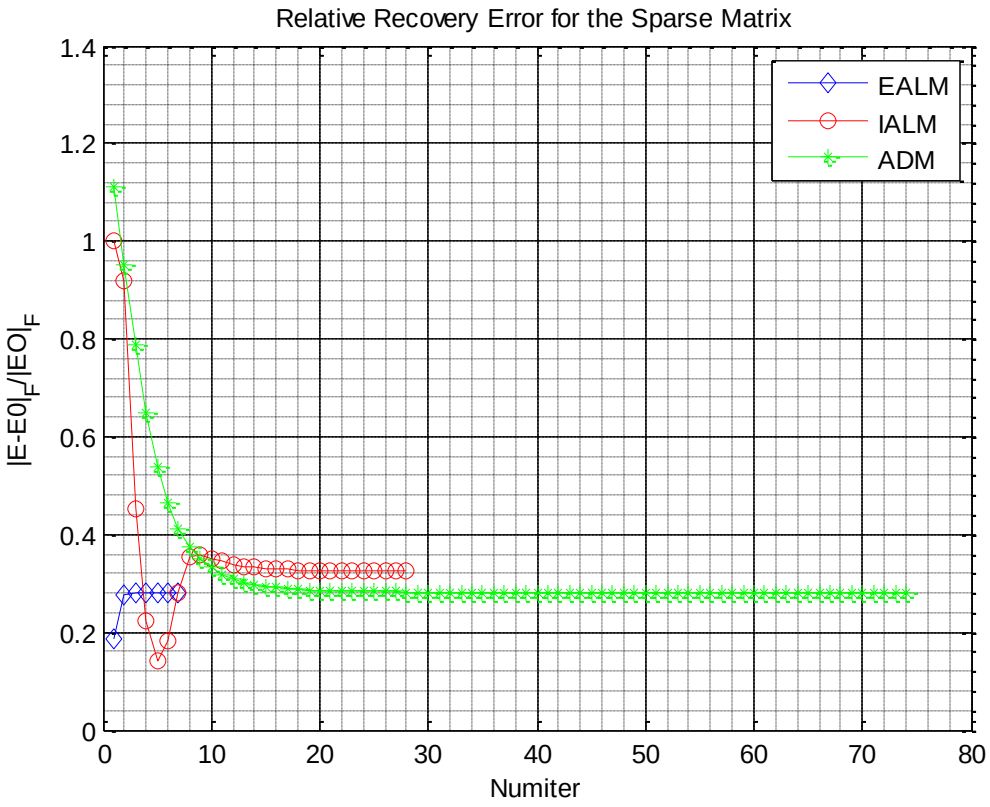

Figure 18: RES-Bridge(Noisy)

Furthermore, the rank of the low-rank component is shown below:

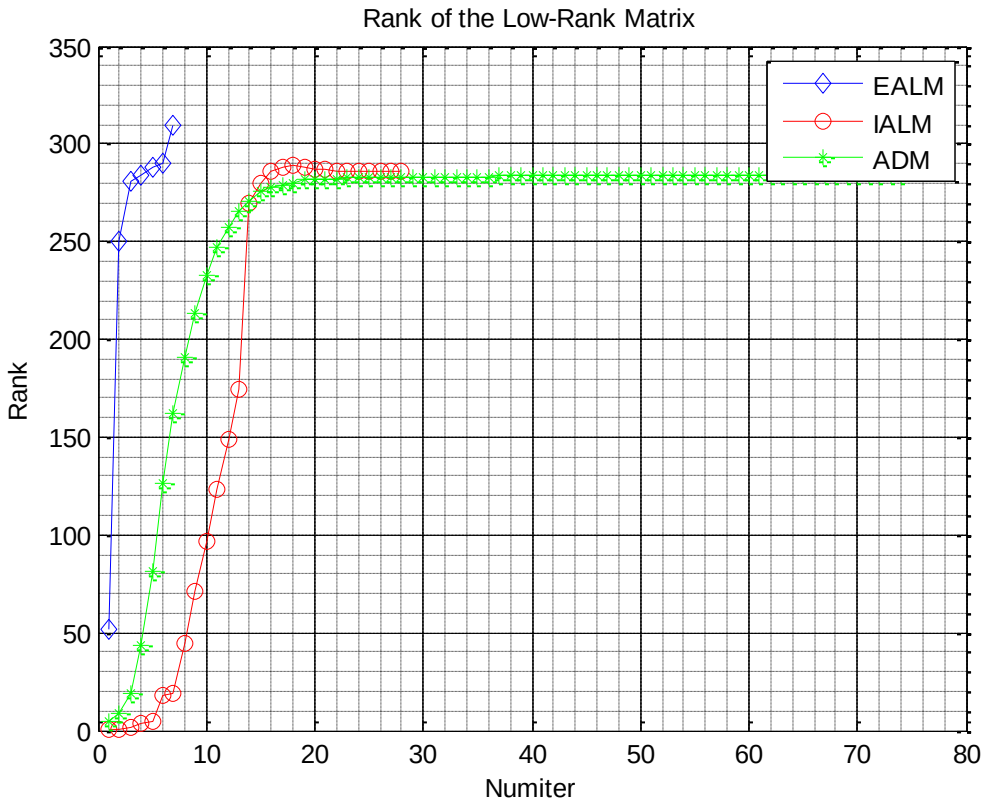

Figure 19: Rank-Bridge(Noisy)

At the same time, the cardinality of the sparse component is depicted below:

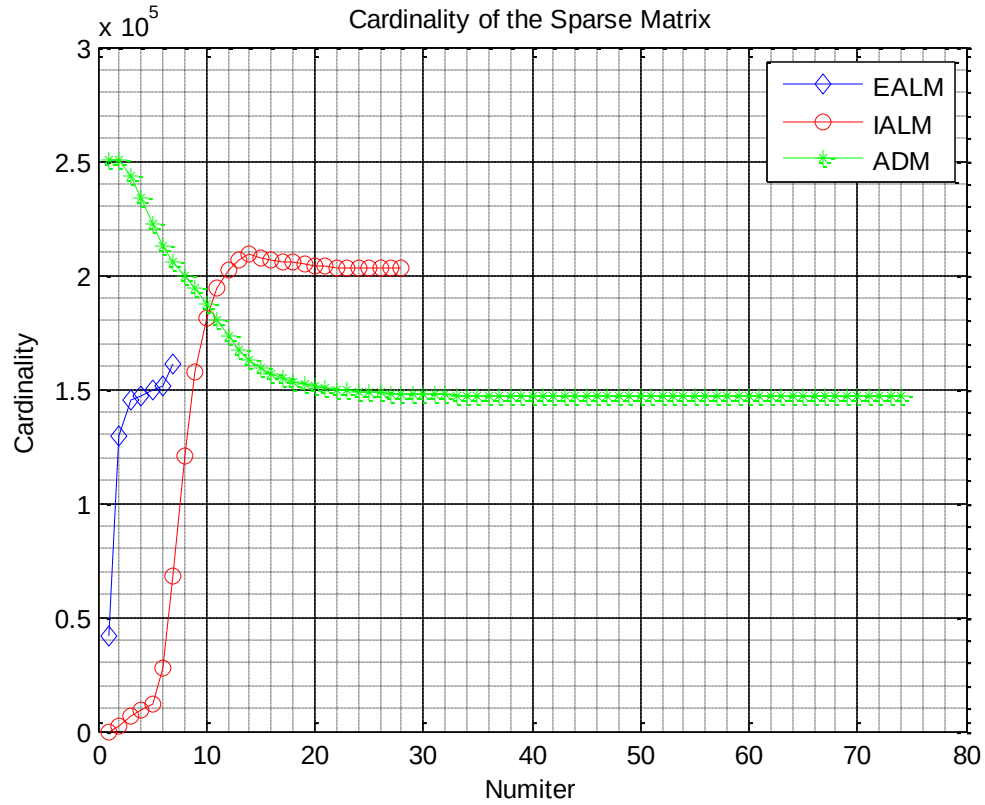

Figure 20: Cardinality-Bridge(Noisy)

Finally, the PSNR metric is depicted in the following figure:

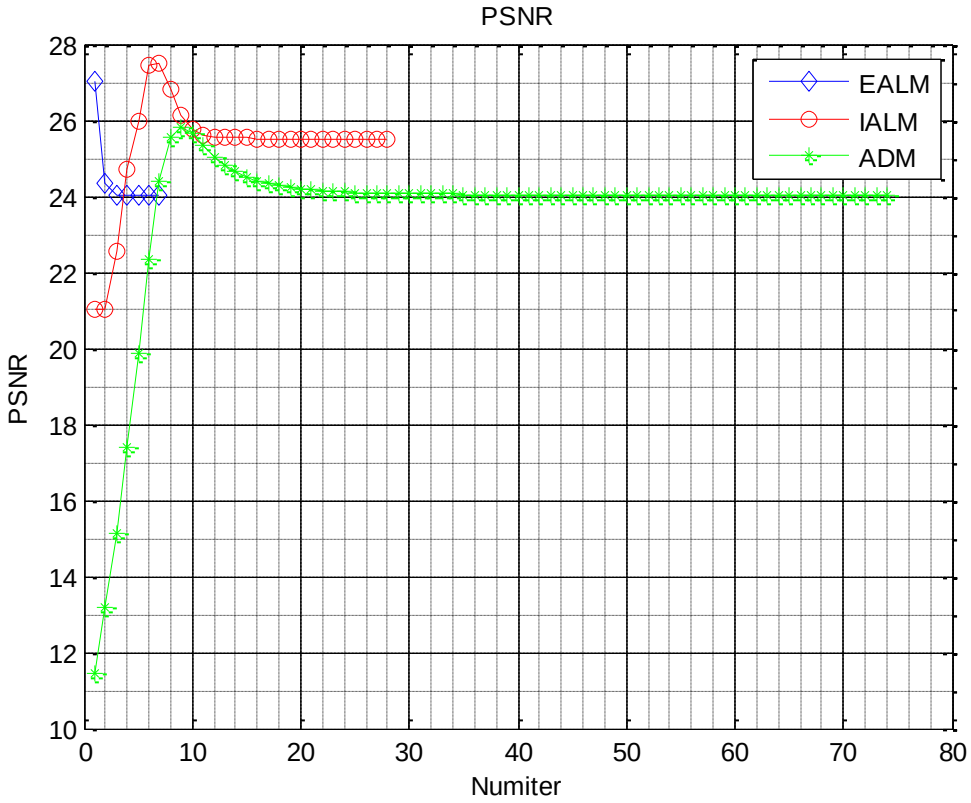

Figure 21: PSNR-Bridge(Noisy)

For the 1024×1024 image resolution case, the original image which depicts lights in a dark background, is shown below:

Image 48: Original Image-Lights(Noisy)

Its noisy counterpart is then depicted in the following image:

Image 49: Noisy Image-Lights(Noisy)

For the EALM algorithm, the reconstructed image as well as the estimated outlier noise are depicted in the following images:

Image 50: Reconstructed Image-Lights(Noisy)-EALM

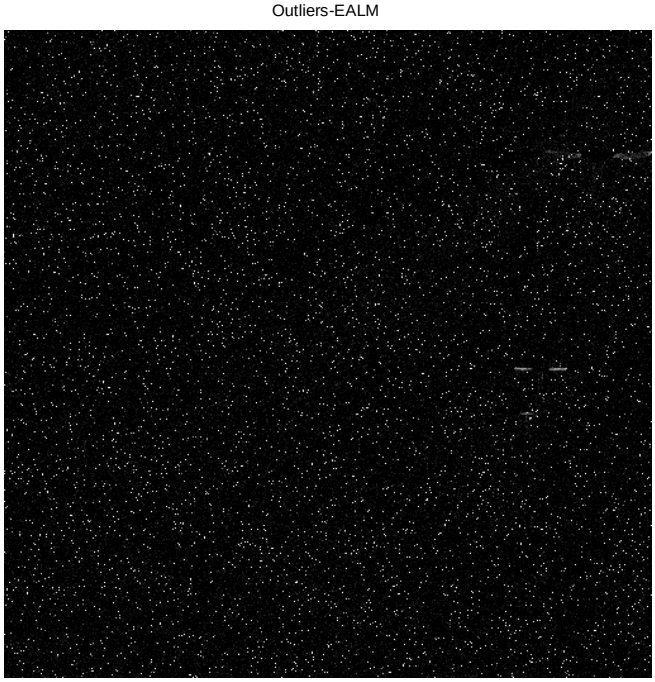

Image 51: Outliers-Bridge(Noisy)-EALM

The respective images for the IALM algorithm are the following ones:

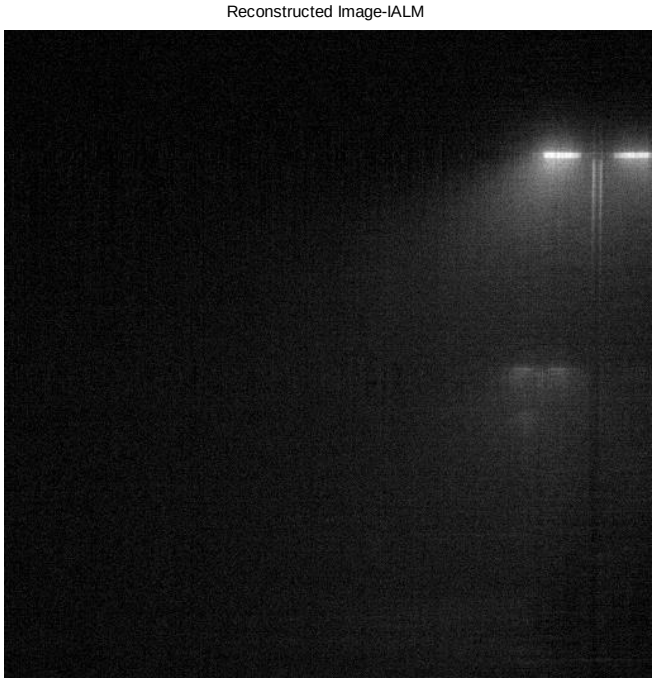

Image 52: Reconstructed Image-Lights(Noisy)-IALM

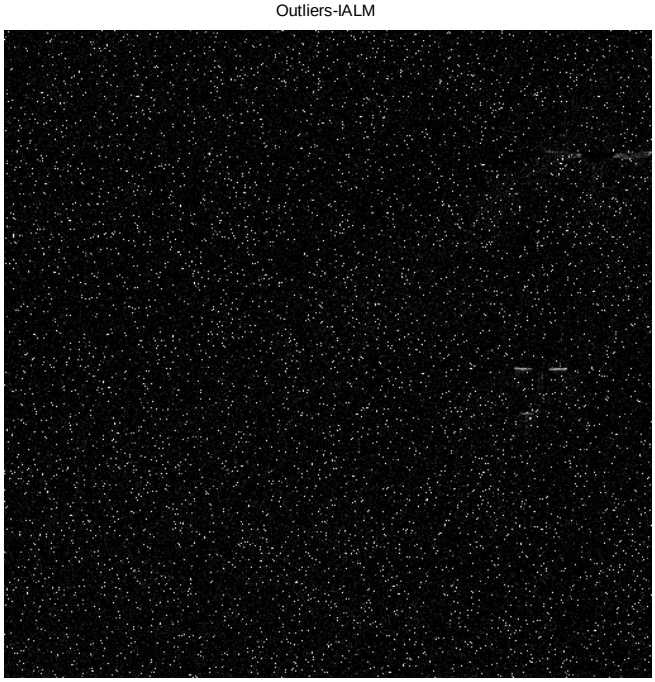

Image 53: Outliers-Lights(Noisy)-IALM

As far as the ADM algorithm is concerned, its results are shown in the upcoming images:

Image 54: Reconstructed Image-Lights(Noisy)-ADM

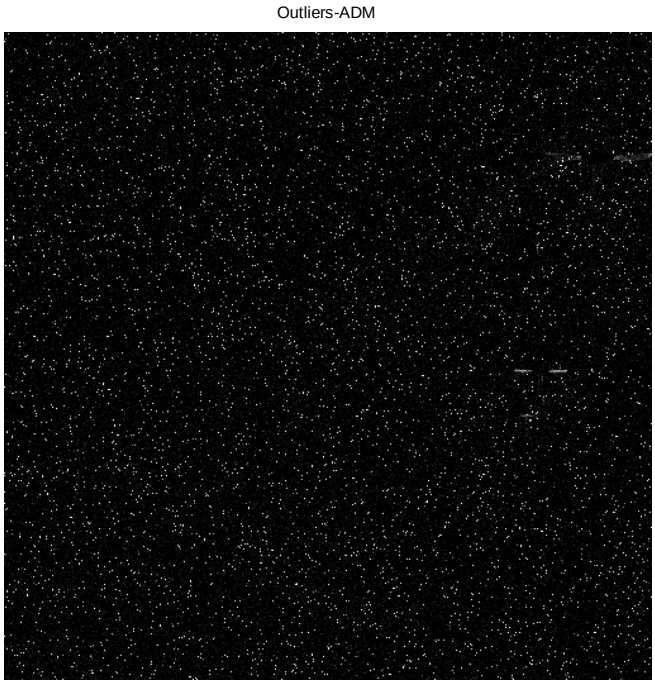

Image 55: Outliers-Lights(Noisy)-ADM

In what has to do with the performance metrics for this case, the relative error for the low-rank component is shown in the following figure:

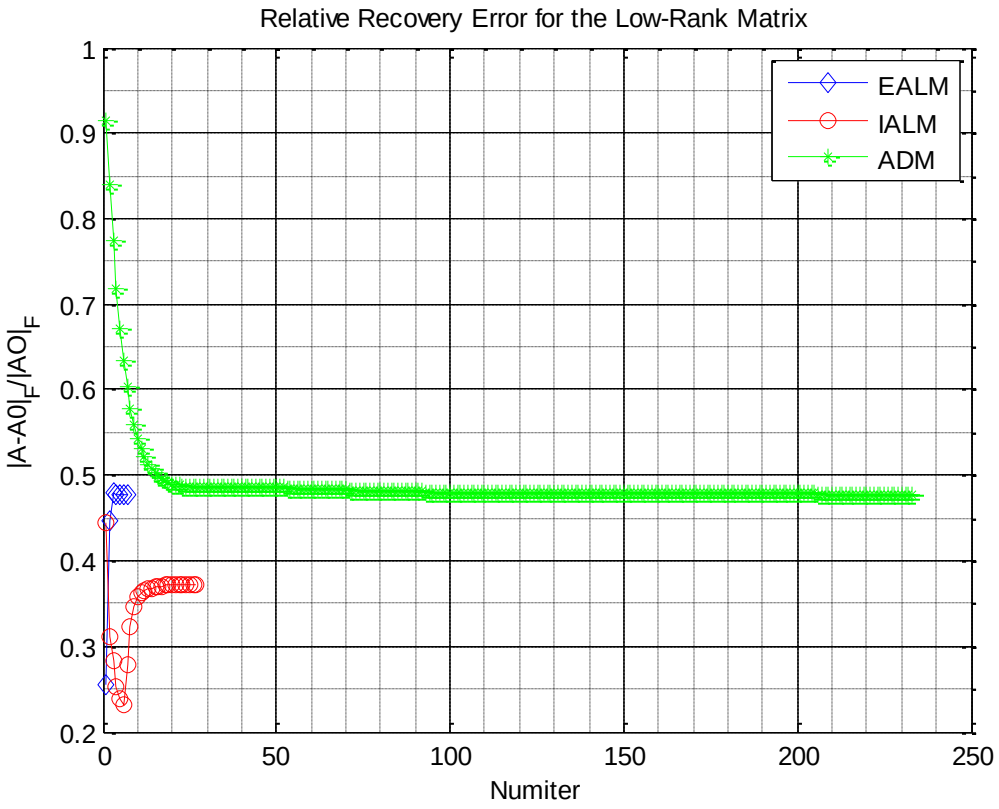

Figure 22: RELR-Lights(Noisy)

As for the relative error for the sparse component, it is depicted in the following figure:

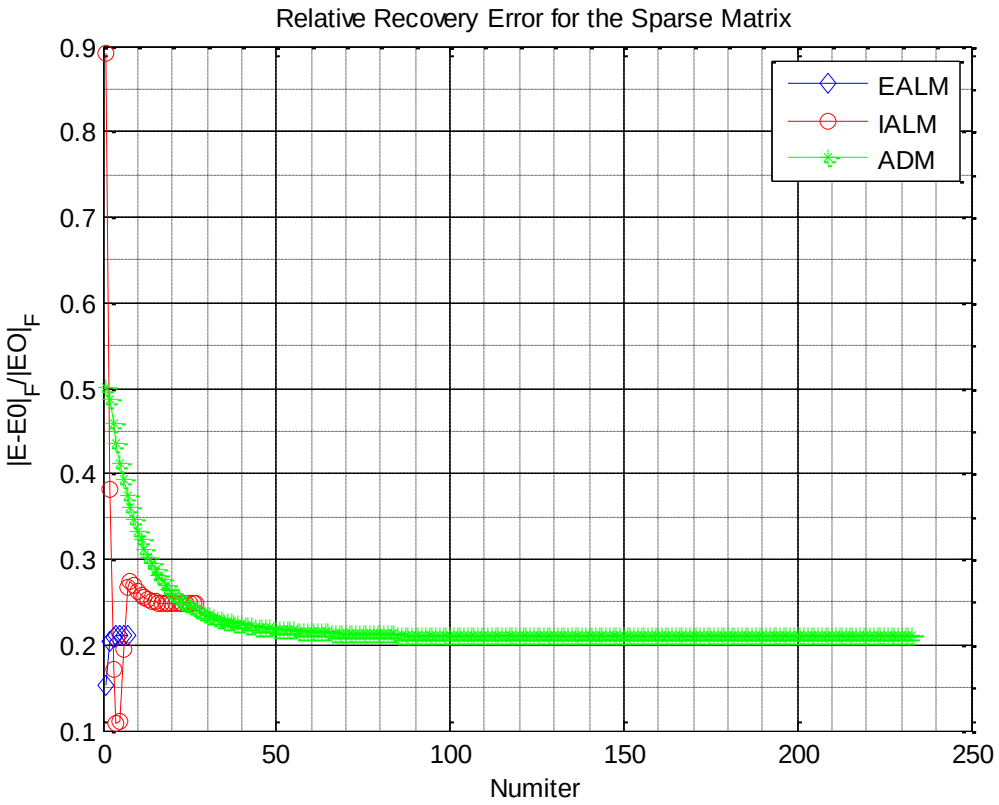

Figure 23: RES-Lights(Noisy)

Furthermore, the rank of the low-rank component is shown below:

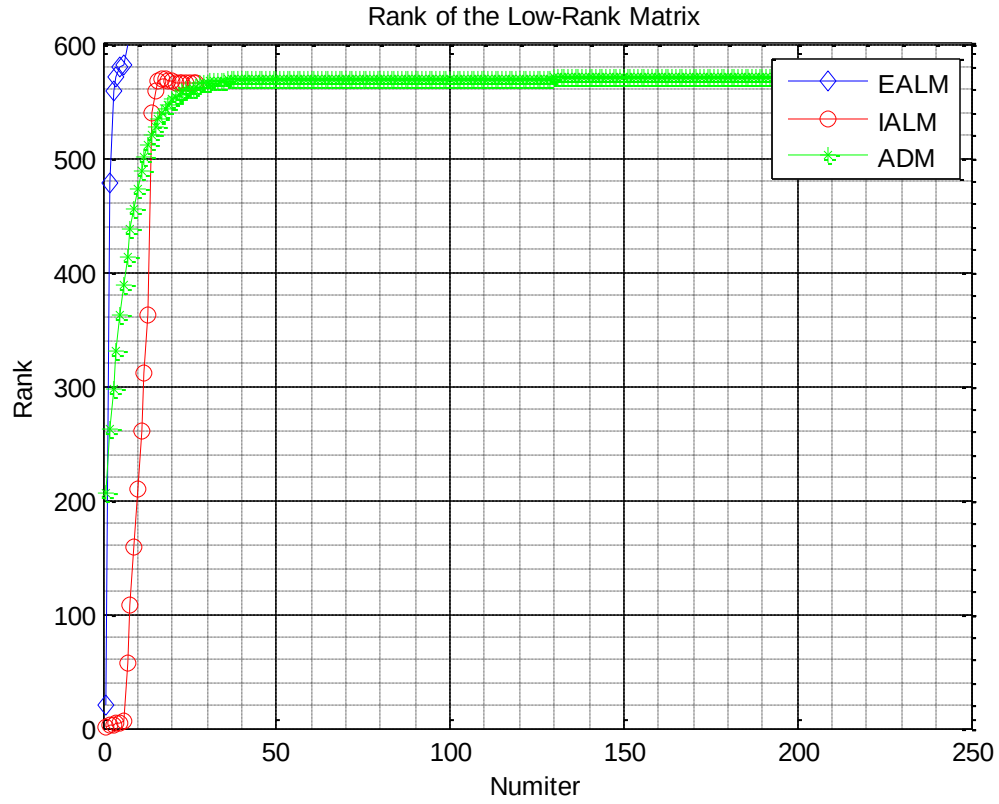

Figure 24: Rank-Lights(Noisy)

As for the cardinality of the sparse component, it is depicted below:

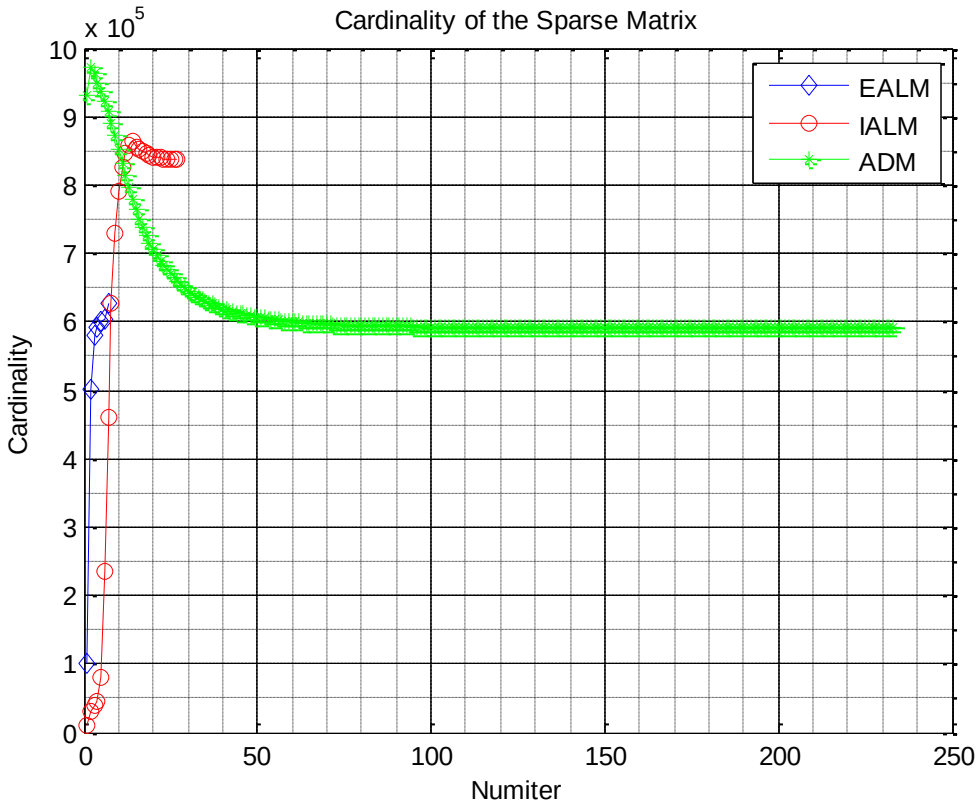

Figure 25: Cardinality-Lights(Noisy)

Finally, the PSNR metric is depicted in the following figure:

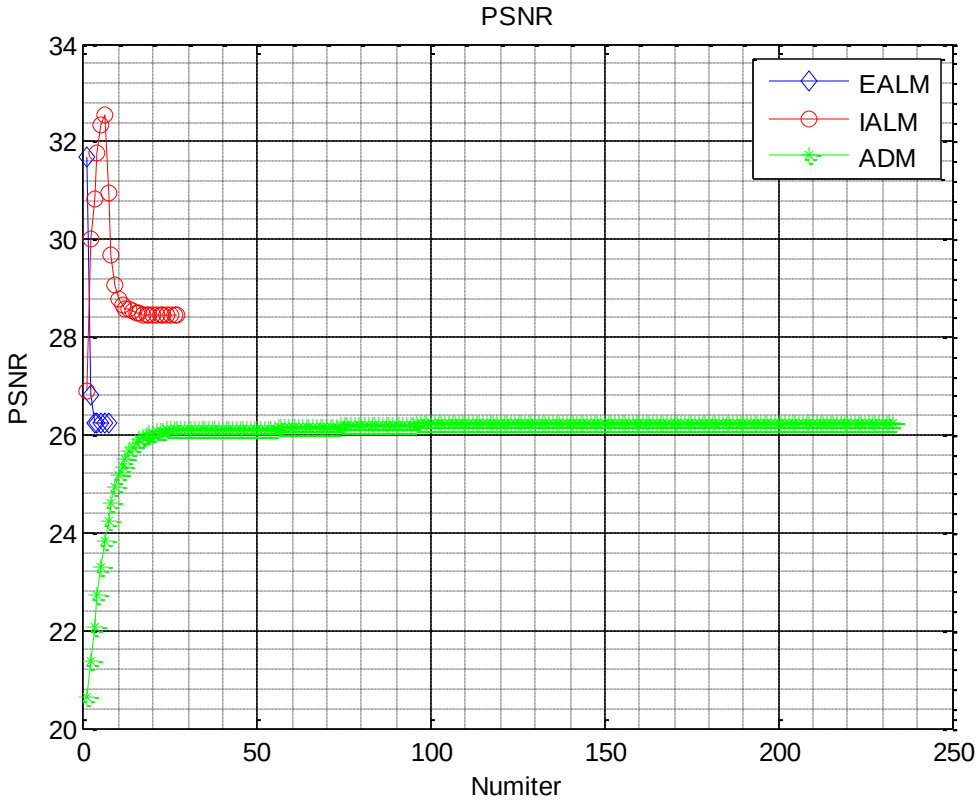

Figure 26: PSNR-Lights(Noisy)

Finally, in the case of the 2048×2048 image resolution, the original image which depicts stones in a woody background, is shown below:

Image 56: Original Image-Stones(Noisy)

Following the same pattern as above, its noisy counterpart is depicted below:

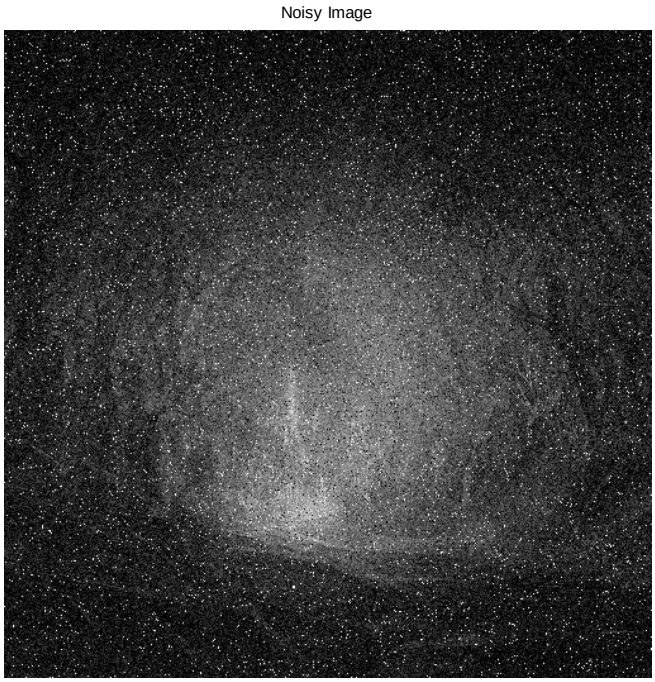

Image 57: Noisy Image-Stones(Noisy)

The reconstructed image as well as the estimated outlier noise “produced” by the EALM algorithm are depicted below:

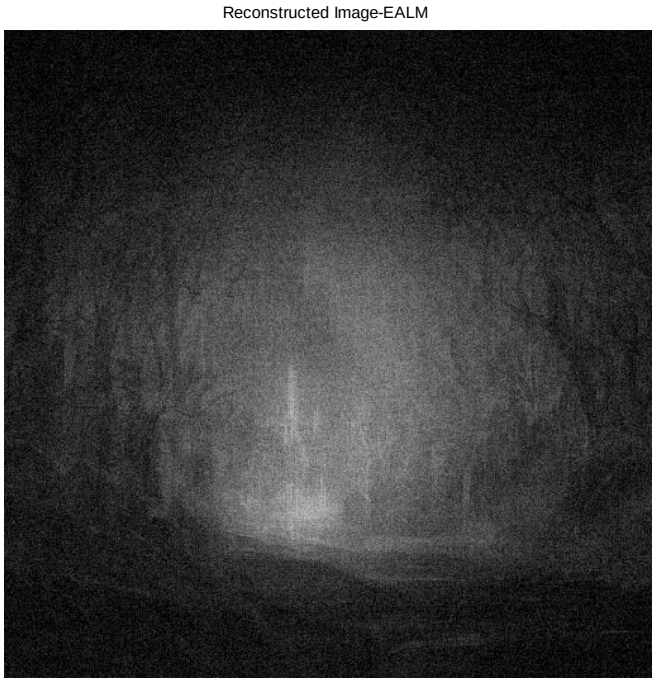

Image 58: Reconstructed Image-Stones(Noisy)-EALM

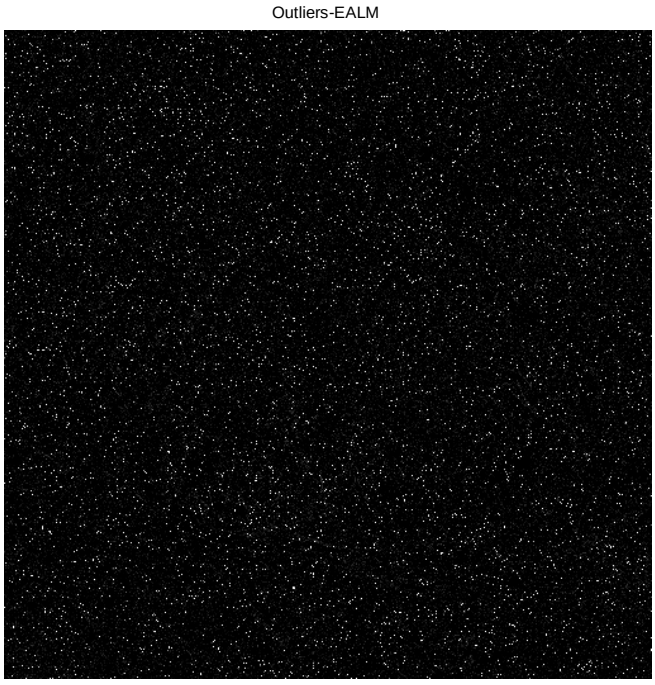

Image 59: Outliers-Stones(Noisy)-EALM

Furthermore, the respective images “produced” by the IALM algorithm are shown below:

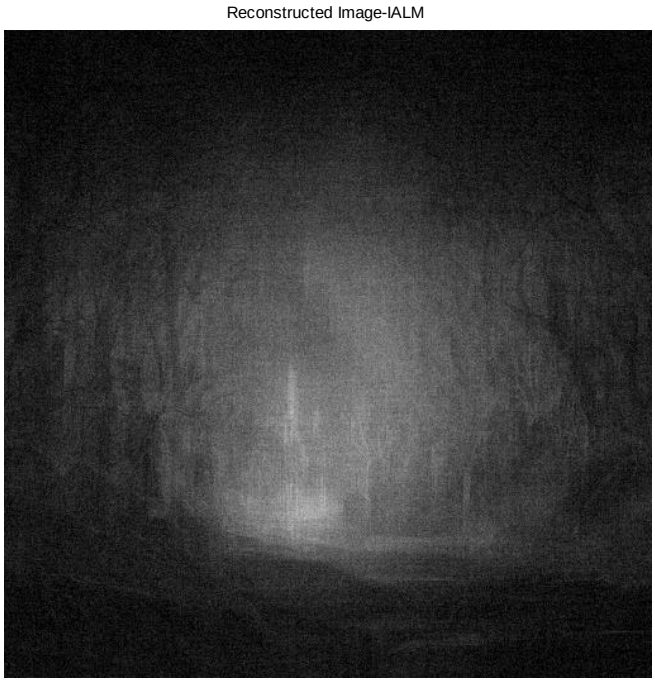

Image 60: Reconstructed Image-Stones(Noisy)-IALM

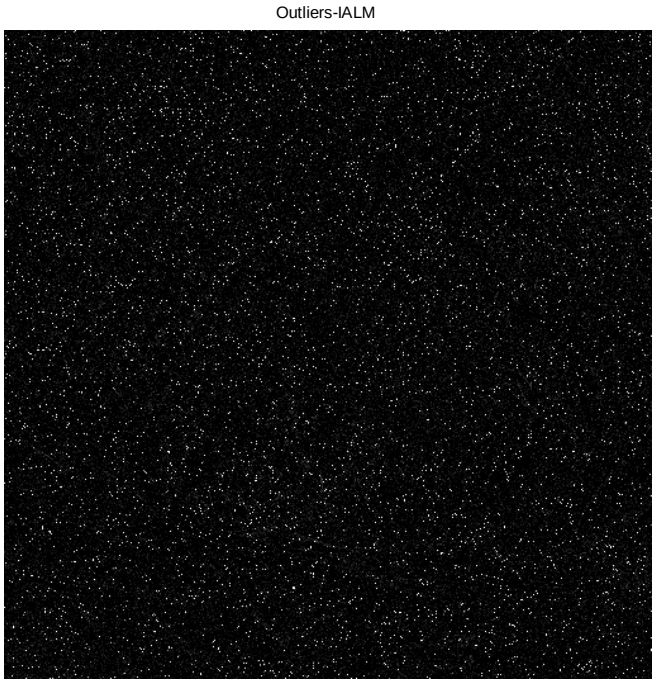

Image 61: Outliers-Stones(Noisy)-IALM

As for the ADM algorithm’s results, they are shown straightaway:

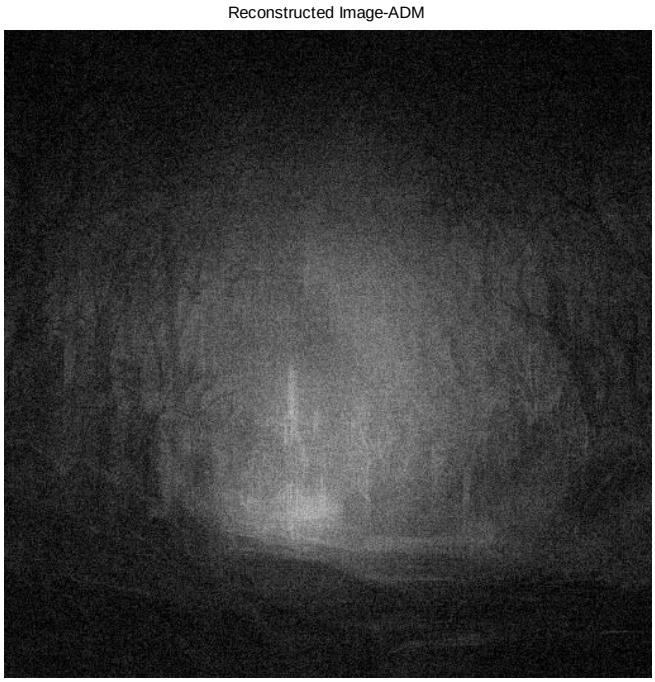

Image 62: Reconstructed Image-Stones(Noisy)-ADM

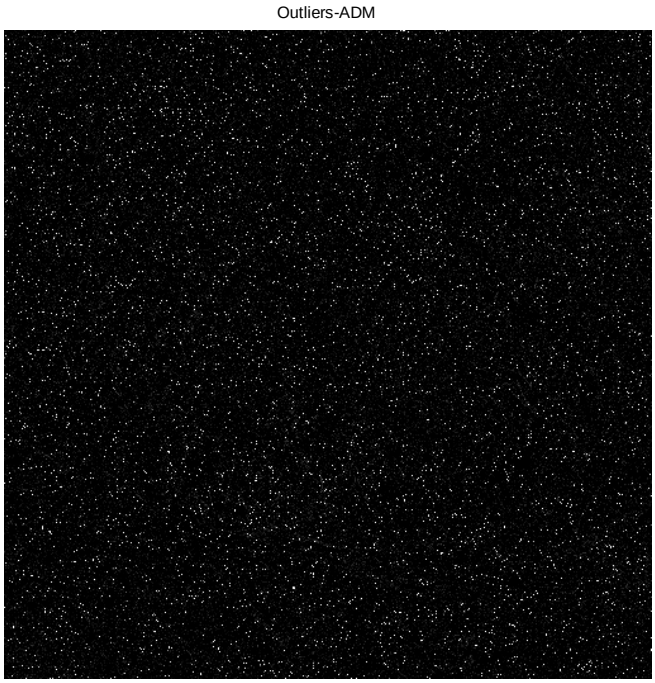

Image 63: Outliers-Stones(Noisy)-ADM

As far as the performance metrics for the highest-resolution case are concerned, the relative error for the low-rank component is depicted in the forthcoming figure:

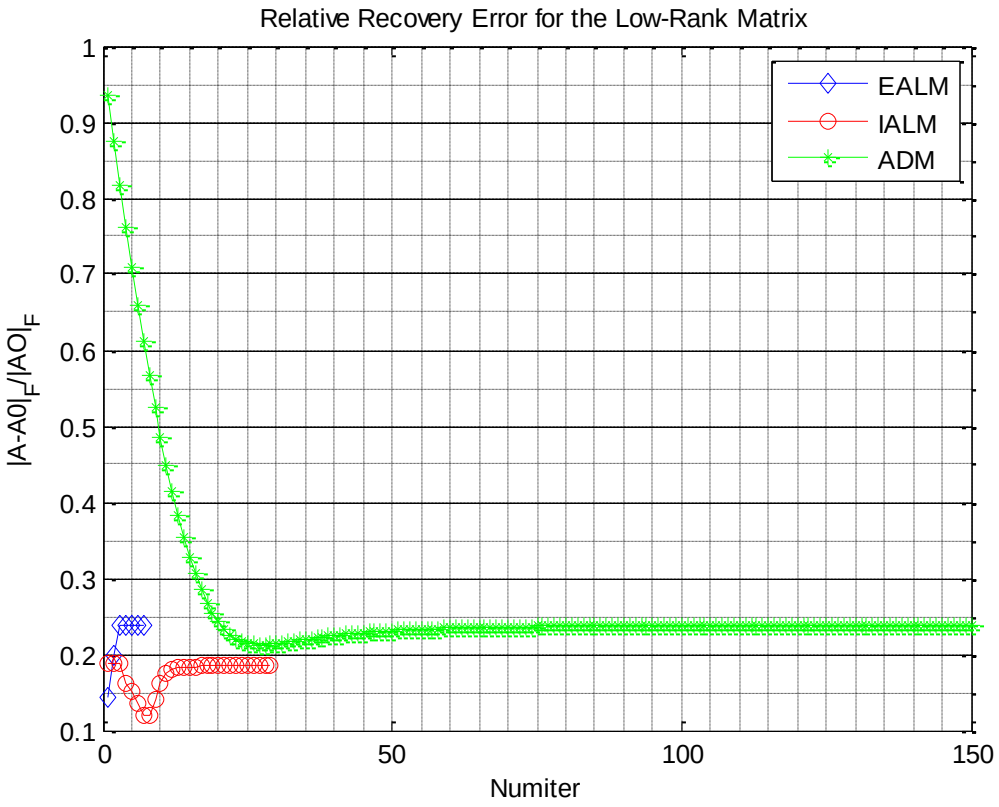

Figure 27: RELR-Stones(Noisy)

At the same time, the respective relative error for the sparse component is shown below:

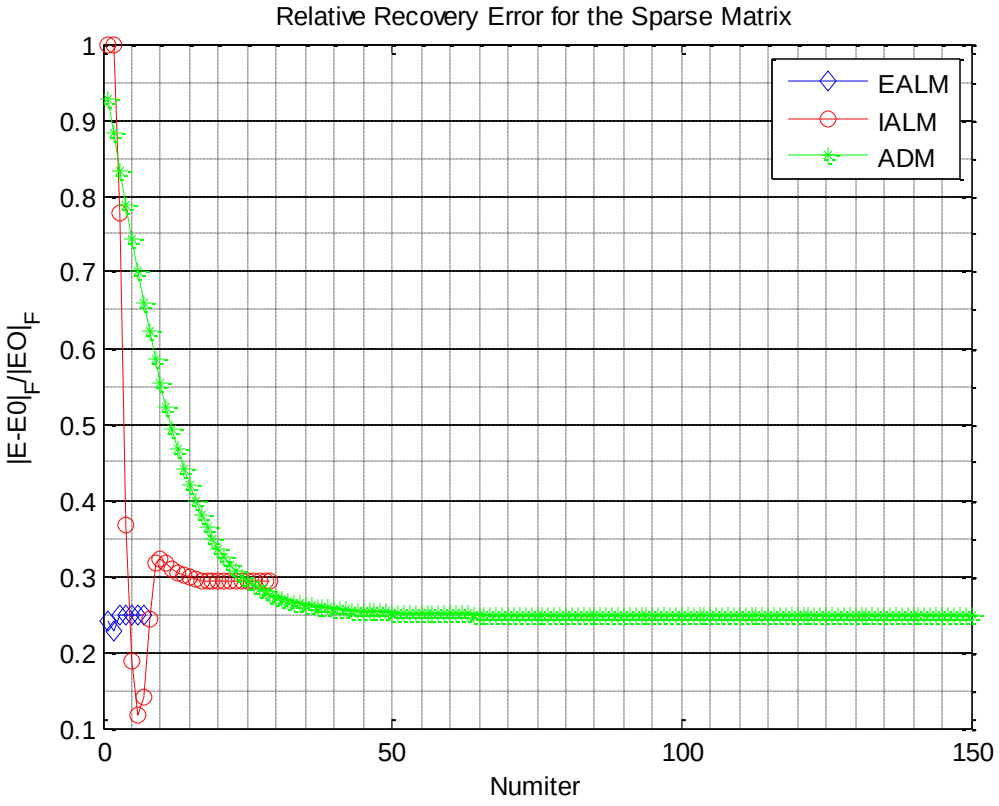

Figure 28: RES-Stones(Noisy)

The rank of the low-rank component follows up:

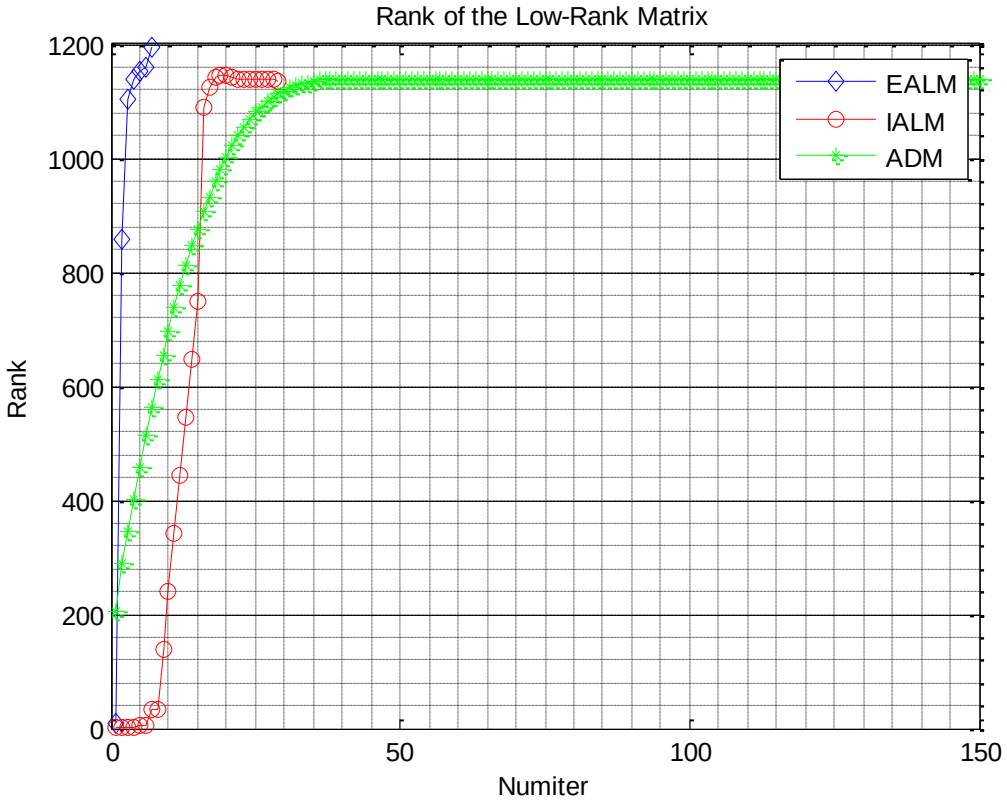

Figure 29: Rank-Stones(Noisy)

As for the cardinality of the sparse component, it is depicted in the following figure:

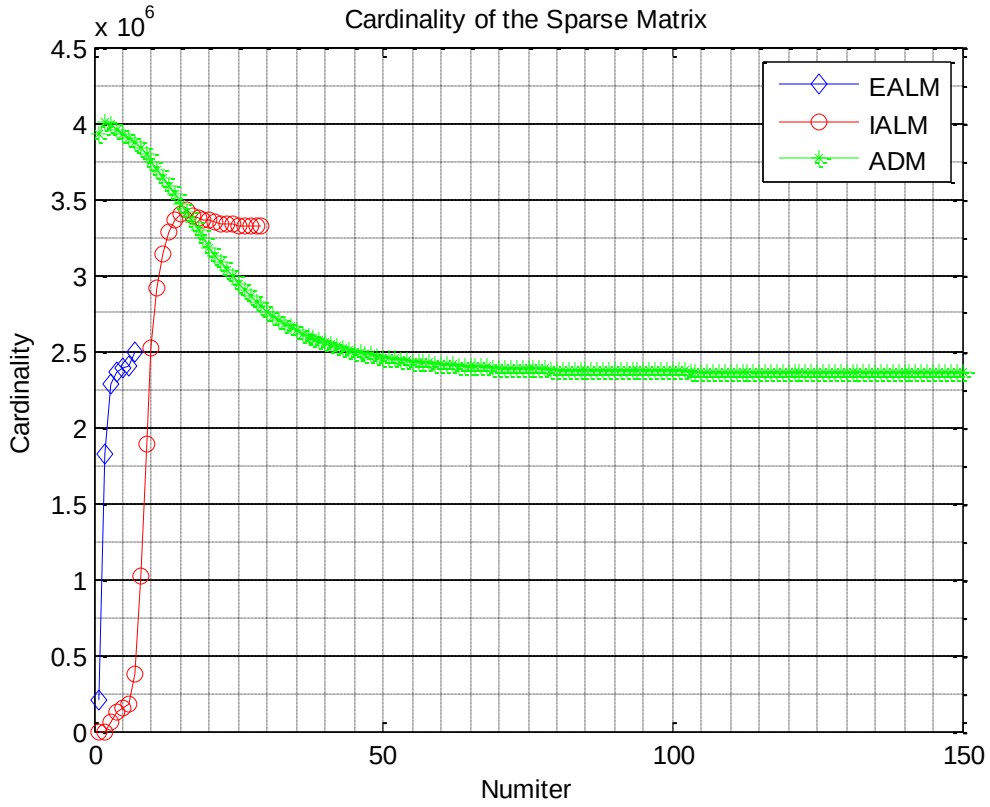

Figure 30: Cardinality-Stones(Noisy)

Finally, the PSNR metric is shown below:

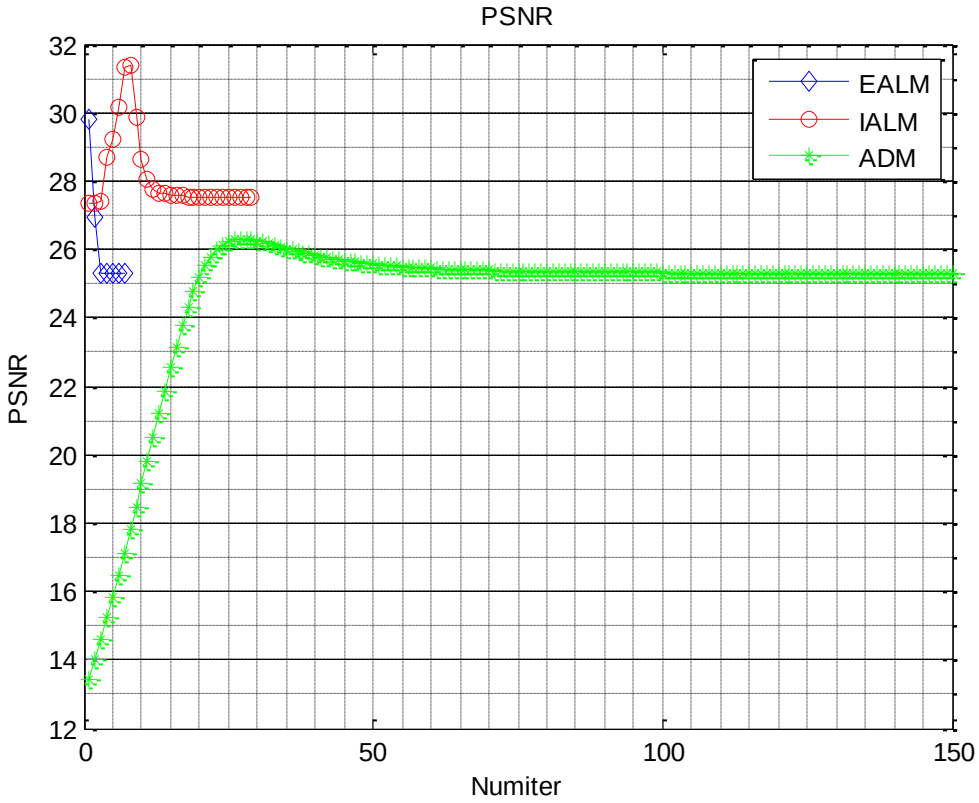

Figure 31: PSNR-Stones(Noisy)

The above results can again be summarized in a table similar to Table 5. In this case, this table is going to contain all those key elements allowing us to extract useful information concerning the stability of the employed algorithmic methods under entry-wise noise added to each image pixel:

Table 6: Comparison between different algorithmic schemes on the Image De-noising pursuit-Noisy Scenario

Image Resolution	Algorithm	$\frac{\|A-A_0\|_F}{\|A_0\|_F}$	$\frac{\|E-E_0\|_F}{\|E_0\|_F}$	$rank(A)$	$\|E\|_0$	PSNR	# SVDs	# Iterations	Time(s)
512×512	EALM	1.98×10^{-1}	2.82×10^{-1}	310	161326	24.036	1202	7	109.44
	IALM	1.68×10^{-1}	3.27×10^{-1}	286	203026	25.493	28	28	4.03
	ADM	1.98×10^{-1}	2.81×10^{-1}	284	146675	24.036	74	74	8.08
1024×1024	EALM	4.78×10^{-1}	2.10×10^{-2}	600	626535	26.261	1312	7	830.15
	IALM	3.72×10^{-1}	2.48×10^{-2}	565	838184	28.451	27	27	32.77
	ADM	4.78×10^{-1}	2.10×10^{-2}	571	591403	26.260	233	233	159.79
2048×2048	EALM	2.39×10^{-1}	2.48×10^{-1}	1196	2489325	25.317	1190	7	7547.67
	IALM	1.85×10^{-1}	2.93×10^{-1}	1136	3325088	27.507	29	29	281.24
	ADM	2.38×10^{-1}	2.48×10^{-1}	1138	2362938	25.318	150	150	982.62

If we look closer the results of the above table, there are many useful conclusions that arise.

First of all, it is quite evident that compared to the noiseless scenario the reconstruction error for the low-rank component is higher. Of course, this is a quite reasonable result as the problem now is tougher than before due to the extra entry-wise Gaussian noise. Although, if we address problems which contain low-contrast images (mainly which these

algorithms tackle better), the relative error drops, a quite encouraging result for handling images of higher resolution.

In what has to do with the level of “low-rankness” of the acquired solution, the IALM method seems turning the tide and achieving the best performance -surpassing the ADM method which was the leading one in the noiseless scenario in terms of this performance metric. The most interesting remark though concerning this metric is that as the resolution of the image increases so does the gap between the “low-rankness” level of the two aforementioned methods in contrast with the EALM one. As a result, for handling images of even higher resolution, it is reasonable to expect that the recovered solution obtained by the IALM and ADM methods is going to be of even lower rank than that by EALM, a “gain” which is going to be proportional to the increase of the image resolution.

Concerning the sparsity level of the obtained sparse component, even if its utility is not as important as the above metric (as also mentioned earlier), the situation bears a strong resemblance to that of the noiseless case. In other words, the ADM method recovers once more the sparsest solution but this time is followed by EALM and afterwards IALM. The IALM method seems more prone to the addition of entry-wise noise to the image than the other two methods in terms of this quality metric, a difference which becomes quite clear as the resolution of the image increases. As a consequence, the more we augment the noise power, the more favorable is going to be the case for the ADM method.

Furthermore, in perfect alignment with the above remark is another one concerning the distinctive ability of the images occurring from the processing via the above mentioned methods. More precisely, if we take a deeper look at them compared to the respective ones occurred in the noiseless scenario, it is quite obvious that the de-noising process has lower quality results. Of course such a result is quite logical, as our test images are contaminated with quite a lot amount of noise -which makes the de-noising process tougher. This difference is also depicted in terms of the PSNR metric (mentioned below), which is clearly lower than in the noiseless case.

As far as the computational load is concerned, the number of SVDs required to obtain a specific solution reveals two main conceptual results: the first one is that clearly the IALM method is the less computational-thirsty one (with the respective number of SVDs staying almost constant as the dimension of the problem increases) -just as in the noiseless case, while the second one is that as we added extra entry-wise noise the performance of the ADM method ameliorated while that of the EALM one deteriorated. Such a result indicated that clearly the IALM method is the best one according to this performance metric even if we add further noise to our test image, but at the same time the ADM method is also a good alternative.

In terms of the PSNR metric, the situation is quite clear. More precisely, the odds are clearly in favor of the IALM method, as the additional entry-wise noise leaded to its overtaking in this performance index. Getting started from a 1.5dBs gain against the other two methods for the lowest image resolution case, this gain of IALM seems widening and consolidating over 2.2dBs as we increase the resolution of the test image. Furthermore, the more noise we add to the image, the more this gap is widened. Such a thing can be proven either by observing the respective values of the metric at each different image resolution scenario or by observing the respective figures of the PSNR metric in which all the aforementioned gap is quite evident. As a final result we could consequently say that the IALM method seems being more stable to the addition of further noise to our image, concerning the PSNR metric.

Moreover, another result occurring from the above experiments is that the number of required iterations for convergence stays pretty much the same for EALM and IALM methods, just as in the noiseless case. Although this is not the case with the ADM method,

its performance seems ameliorating as we added extra noise to the test image. Such a thing is a quite encouraging one, as even if its performance is still the worst one among the three employed methods it seems more stable to the addition of extra entry-wise noise.

Last but not least, there are some interesting results occurring again from the computational time needed for each algorithm to solve a problem of specific image resolution. More precisely, the first conclusion is that clearly IALM once more outperforms the other two methods -a difference which becomes quite evident in the case of the highest image resolution where the time required by IALM to solve the problem is up to 4.6 minutes while for ADM approaches 16 minutes and for EALM 2 hours.

The most interesting results although occur if we take a deeper look to the time needed for each algorithm to transit from a lower-resolution image to a higher one as a relative ratio: we observe that the EALM method requires about 7.58 times the time needed to solve to 512×512 problem for coping with the 1024×1024 one, a situation which gets a bit worse for the 2048×2048 case in which the respective ratio augments up to 9.09 times. At the same time, the regime for the ADM method is 19.77 times for the transition from 512×512 to 1024×1024 and 6.14 times for that from 1024×1024 to 2048×2048. On the contrary, IALM initially requires 8.13 its first computational time for the first transition, but as the dimension of the problem augments, this ratio is slightly augmented up to 8.5 times. As a result, we could say that as the image resolution increases both 3 algorithms seem coping respectively well with that, but the rate of doing so is a bit higher for the ADM method. On the contrary, the IALM method seems achieving a constant rate of up to 8 times its previous required time, which may seem not having a decreasing rate, but still is the fastest one as well as the more stable one.

Another interesting comment could be made about the relative speed of the fastest method (i.e. IALM) compared to the other 2 as we transit to higher-resolution images. In plain English, we observe that as for the 512×512 problem IALM is about 27.15 times faster than EALM and 2 times faster than ADM, this relative ratio changes as we double the resolution of the image to 1024×1024 to 25.33 times for EALM and 4.87 times for ADM. Further increase of the image resolution up to 2048×2048 does not change the situation significantly, as the relative time ratios are slightly modified to 26.83 times and 3.49 times respectively. As a consequence, we could say that as we increase the dimensionality of the problem, the two slowest methods (i.e. ADM and EALM) seem ameliorating their current performance compared to the fastest one (i.e. IALM). Such an argument contributes to the fact of them coping better with the extra entry-wise noise in terms of time metric than the IALM method, but at the same time, one should not surpass the fact that the IALM method is still the fastest one and at the same time the one achieving the best performance regarding the (crucial in this kind of application) PSNR performance metric.

5. CONCLUSIONS

The present thesis was focused on the theoretical as well as the algorithmic aspects of the RPCA problem. In the beginning, we presented the classical PCA technique for dimensionality reduction, mentioning those advantages that have made it so popular among scientists from different fields and with different backgrounds. Subsequently, the main disadvantage of it (i.e. its sensitivity to outliers) was clarified, triggering at the same time the question of whether the whole method could become robust vis-à-vis gross errors. Towards this direction, the whole theoretical aspects of the RPCA problem was stated, concerning basic theorems, constructional limits occurring from them, as well as applications of real life where the RPCA plays a key role.

Furthermore, the main research part of this thesis was focused on algorithmic schemes which were developed exactly to tackle the RPCA problem. After each one of them was presented in details -according to the respective publications- several experimental tests among them took place in order to evaluate their performance under different experimental circumstances. Subsequently, a real-application case-study from the scientific field of image processing took place, in order to examine the performance of the three fastest algorithmic methods studied in this thesis in more realistic datasets.

Based on widely used KPIs in this scientific field, which were presented in Chapter 4, we could highlight the following conclusions:

- Concerning the relative error of reconstruction of the low-rank component, the ALM algorithm (both the EALM and the IALM versions) is the most accurate one, followed by the ADM, the DM, the SVT and finally by the APG algorithm.
- As far as the relative error of reconstruction of the sparse component is concerned, the ALM algorithm (both the EALM and the IALM versions) is again the most accurate one, followed this time by the APG, the ADM, the DM and finally by the SVT algorithm.
- Concerning the number of SVDs computed by each algorithmic scheme until convergence, the IALM version of the ALM algorithm does the less number of computations. It is followed by the EALM version of the ALM algorithm, the ADM, the APG, the DM and finally by the SVT algorithm. Obviously, the adoption of versions of algorithms which adopt partial SVDs (such as PAPG and IALM) rather than full ones could prove extremely useful concerning this KPI, especially as the dimension of the problem augments.
- As far as the number of iterations required for convergence is concerned, the EALM version of the ALM algorithm is the leading one, followed by the IALM version of the ALM algorithm, the ADM, the APG, the DM and finally the SVT algorithm. This KPI is related with the aforementioned one, as the SVD computation is the most computational “thirsty” operation done by each algorithm.
- Concerning the total computational time of each algorithm, once more the ALM algorithm proved to be the best one -independently of which one version was adopted- followed by the ADM, the APG, the DM and the SVT algorithm. Once more the number of the SVDs computed by each algorithm plays an important role also in this KPI, for the same reason as was the case with the number of iterations until convergence.

Given the above conclusions, it may seem easy to conclude that the ALM algorithm is the best one among all the algorithmic schemes which were presented and tested in this thesis. Such a thing holds true for most of the experimental circumstances as well as the

KPIs chosen for the testing process. In general the ALM algorithm enriched with an ADM approach scales well for several optimization problems, and the RPCA problem could not be an exception to the rule. The combination of these two approaches with an adaptive update of the step-size parameters leads to significant results which deserve the attention of anyone involved in this field of research.

Furthermore, it is deemed appropriate to mention the fact that the above comparison took place under the specific simulation conditions described in the final section of Chapter 4. The main idea was that all the involved parameters should be tuned in such a way that the upcoming comparison would be “as fair as possible”. For most of them, there were chosen the values proposed in the respective publications, but there were some cases in which this was not the case: in the DM algorithm we chose to do an exact line search (rather than an inexact one which was proposed in the respective publication) in order to determine the respective step-size parameter, while on the ADM algorithm we chose a different value than the proposed one for the relaxation parameter γ. Both these “initiatives” were adopted in the “spirit of fairness” mentioned above, as well as for not throwing away key ideas of the algorithms presented in this thesis.

Concerning the case-study of the image de-noising problem, we should mention the following results:

- In what has to do with the relative error of reconstruction of the low-rank component, both the EALM, IALM and ADM methods achieve equally good performance either we add extra entry-wise noise to the image or not.
- As far as the relative error of reconstruction of the sparse component is concerned, the situation is pretty much the same as above for the noiseless case, while if we add further noise to the image there are observed small differences in favor of the EALM and ADM methods. Although, this metric is of less importance than the above one, one should not surpass it without mentioning the occurring ranking of the methods employed.
- Concerning the level of “low-rankness” of the de-noised images, the ADM method is the best choice in the noiseless case, while in the noisy one the IALM method scales better. Although the gap between them is not wide enough, this turnaround is indicative of the fact that the IALM method seems more stable than the ADM one if we add extra noise to our test image.
- In what has to do with the cardinality metric, the ADM method seems achieving the best score -both in the noiseless scenario as well as in the noisy one. What should also be mentioned here is that the performance of the IALM method seems deteriorating if we add further noise to the image -leading to “more dense” estimations of the outlier noise.
- In terms of the PSNR metric, which probably is the most crucial one due to the nature of the application studied, in the noiseless case all three algorithms seem achieving similar performance. Nevertheless, as we add further noise to the image, the IALM method clearly outperforms the other two methods -indicating that it is more stable in the presence of extra noise at each pixel of the image.
- Concerning the number of SVDs computed by each algorithmic scheme until convergence, clearly the IALM method is the leading one both in the noiseless as well as in the noisy case. Nevertheless, we should underline here the fact that the ADM method seems scaling better with the addition of further noise in the image than the EALM one.

- As far as the number of iterations required for convergence is concerned, the EALM and IALM methods seem the best alternative -either we add further noise to the image or not. Although, it is worth-mentioning here the fact that even if the performance of the ADM method is the worst one according to this metric, it seems quite stable to the addition of extra entry-wise noise to the image.
- Concerning the total computational time of each algorithm, clearly the IALM method is the most trust-worthy one in any case. Its relative performance over the other two employed method in this case-study seems constant in the noiseless case, whereas the addition of extra entry-wise noise seems on the one hand ameliorating the situation for the other two slowest methods although on the other hand this amelioration seems not enough to dethrone the IALM method from the leading position.

Finally, we should not forget to refer to the fact that the specific application to which these algorithms are used plays an important role. In the case-study examined here, the selection of the algorithms to be tested was based mainly on their performance in terms of the computational time index computed in the first experiments of this thesis. Perhaps, in another application where some other KPIs play a more important role (i.e. the accuracy of the low-rank component), also other algorithms studied in this thesis could be an equally good alternative (i.e. the DM / SVT algorithms). As a result, our decision of which algorithm should be chosen for a specific application is in the end a matter of on the one hand the “nature” of the application and on the other hand of which KPI we desire each time to be the leading one.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Big Data	Μεγάλα Δεδομένα
Curse of Dimensionality	Κατάρα της Διαστατικότητας
Intrinsic Dimension	Εγγενής Διάσταση
Dimensionality Reduction	Μείωση της Διάστασης
Feature Selection	Επιλογή Χαρακτηριστικών
Feature Extraction	Εξαγωγή Χαρακτηριστικών
Principal Component Analysis	Ανάλυση Κύριων Συνιστωσών
Transformation Matrix	Μητρώο Μετασχηματισμού
Signal-to-Noise Ratio	Λόγος Σήμα-προς-Θόρυβος
Variance	Διακύμανση / Διασπορά
Cartesian Axis System	Σύστημα Καρτεσιανών Συντεταγμένων
Mean	Μέση Τιμή
Covariance	Συνδιακύμανση / Συνδιασπορά
Random Variables	Τυχαίες Μεταβλητές
Covariance Matrix	Μητρώο Συνδιακύμανσης / Συνδιασποράς
Correlation Matrix	Μητρώο Συσχέτισης
Eigenvectors	Ιδιοδιανύσματα
Eigenvalues	Ιδιοτιμές
Principal Axis Theorem	Θεώρημα Κύριων Αξόνων
Karhunen-Loève Transform	Μετασχηματισμός Karhunen-Loève
Stochastic Process	Στοχαστική Διαδικασία
Fourier Series	Σειρά Fourier
Gaussian Distribution	Gaussian / Κανονική Κατανομή
Redundancy	Πλεονασμός
Correlation	Συσχέτιση
Occam’s Razor Principle	Αρχή του Ξυραφιού του Occam
Free Parameters	Ελεύθερες Παράμετροι
Mean Square Error	Μέσο Τετραγωνικό Σφάλμα
Trace	Ίχνος
Linear Transform Invariant System	Γραμμικό Χρονικά Αναλλοίωτο Σύστημα
Kernel Principal Component Analysis	Ανάλυση Κύριων Συνιστωσών με χρήση Συναρτήσεων Πυρήνα
Independent Component Analysis	Ανάλυση Ανεξάρτητων Συνιστωσών
Eckart-Young-Mirsky Theorem	Θεώρημα Eckart-Young-Mirsky
Rank	Τάξη / Βαθμός
Singular Values	Ιδιάζουσες Τιμές
Singular Vectors	Ιδιάζοντα Διανύσματα
Singular Value Decomposition	Ανάλυση Ιδιάζουσων Τιμών
Outlier	Ακραία Τιμή
Robust Principal Component Analysis	Εύρωστη Ανάλυση Κύριων Συνιστωσών
Basis Vectors	Διανύσματα Βάσης
Column Space	Χώρος Στηλών
Row Space	Χώρος Γραμμών
Objective Function	Αντικειμενική Συνάρτηση
Netflix Prize	Βραβείο Netflix
Matrix Completion	Συμπλήρωση Πινάκων
Interior-Point Methods	Μέθοδοι Εσωτερικού-Σημείου

Method of Conjugate Gradients	Μέθοδος των Συζυγών Διευθύνσεων
Condition Number	Δείκτης Κατάστασης
Singular Value Thresholding Algorithm	Αλγόριθμος Κατωφλίωσης των Ιδιάζουσων Τιμών
Step-Size Parameter	Παράμετρος βήματος
Compressed Sensing / Compressive Sampling	Συμπιεστική Δειγματοληψία
Karush-Kuhn-Tucker Conditions / First-Order Optimality Conditions	Συνθήκες Karush-Kuhn-Tucker / Συνθήκες Βελτιστοποίησης Πρώτης-Τάξης
Relative Reconstruction Error	Σχετικό Σφάλμα Ανακατασευής
Iterative Thresholding	Επαναληπτική Κατωφλίωση
Gradient Ascend Algorithm	Αλγόριθμος Ανάδυσης Κλίσης
Dual Problem	Δυϊκό Πρόβλημα
Hilbert Space	Χώρος Hilbert
Gradient	Κλίση / Βαθμίδα / Διανυσματική Παράγωγος
Sub-Gradient	Υπο-κλίση / Υπο-βαθμίδα / Υπο-Διανυσματική Παράγωγος
Cost Function	Συνάρτηση Κόστους
Dual Method	Δυϊκή Μέθοδος
Primal Problem	Αρχικό Πρόβλημα
Steepest Ascend Algorithm	Αλγόριθμος Απότομης Ανάδυσης
Normal Cone	Κανονικός Κώνος
Alternating Projection Scheme	Σχήμα Εναλλασσόμενων Προβολών
Power Method	Μέθοδος των Δυνάμεων
Principal Singular Spaces	Κύριοι Ιδιάζοντες Χώροι
Armijo’s Rule	Κανόνας του Armijo
Augmented Lagrange Multiplier Method	Προσαυξημένη Μέθοδος των Πολλαπλασιαστών Lagrange
Penalty Methods	Μέθοδοι Ποινών
Lagrange Multiplier	Πολλαπλασιαστής Lagrange
Method of Lagrange Multipliers	Μέθοδος των Πολλαπλασιαστών Lagrange
Lagrangian Function	Συνάρτηση Lagrange / Λαγκρανζιανή Συνάρτηση
Method of Multipliers	Μέθοδος των Πολλαπλασιαστών
Augmented Lagrangian Function	Προσαυξημένη Συνάρτηση Lagrange / Προσαυξημένη Λαγκρανζιανή Συνάρτηση
Trace Inner Product	Εσωτερικό Γινόμενου Ίχνους
Exact Augmented Lagrange Multiplier Method	Ακριβής Προσαυξημένη Μέθοδος των Πολλαπλασιαστών Lagrange
Inexact Augmented Lagrange Multiplier Method	Μη-Ακριβής Προσαυξημένη Μέθοδος των Πολλαπλασιαστών Lagrange
Alternating Direction Method	Μέθοδος των Εναλλασσόμενων Διευθύνσεων
Alternating Direction Method of Multipliers	Μέθοδος των Εναλλασσόμενων Διευθύνσεων των Πολλαπλασιαστών
Gauss-Seidel Method	Μέθοδος Gauss-Seidel
Dual Variable	Δυϊκή Μεταβλητή
Jacobi Method	Μέθοδος Jacobi
Euclidean Projection	Ευκλίδεια Προβολή

Key Performance Indicators	Βασικοί Δείκτες Απόδοσης
Independent and Identically Distributed	Ανεξάρτητες και Όμοια Κατανεμημένες
Image De-Noising	Από-θορυβοποίηση Εικόνας
Central Limit Theorem	Κεντρικό Οριακό Θεώρημα
Peak-Signal-to-Noise-Ratio	Μέγιστος Λόγος Σήματος προς Θόρυβο
Absolute Value	Απόλυτη Τιμή
Norm	Νόρμα
Triangle Inequality	Τριγωνική Ανισότητα
Sum-Absolute-Value Norm / Taxicab Norm / Manhattan Norm	Αθροιστική-κατ' Απόλυτη Τιμή Νόρμα / Manhattan Νόρμα
Euclidean Norm	Ευκλείδια Νόρμα
Infinity Norm / Chebyshev Norm	Άπειρη Νόρμα / Chebyshev Νόρμα
Hamming Distance	Απόσταση Hamming
Unit Ball	Μοναδιαία Σφαίρα
Euclidean Space	Ευκλείδειος Χώρος
Matrix Norm	Νόρμα Μητρώου
Operator Norm	Τελεστική Νόρμα
Rayleigh Quotient	Πηλίκο Rayleigh
Spectral Norm	Φασματική Νόρμα
Max-Column-Sum Norm	Μέγιστη-κατά Στήλες-Αθροιστική Νόρμα
Max-Row-Sum Norm	Μέγιστη-κατά Γραμμές-Αθροιστική Νόρμα
Frobenius Norm / Hilbert-Schmidt Norm	Frobenius Νόρμα / Hilbert-Schmid Νόρμα
Frobenius Inner Product	Frobenius Εσωτερικό Γινόμενο
Inner Product	Εσωτερικό Γινόμενο
Sum-Absolute-Value Norm	Αθροιστική-κατ' Απόλυτη Τιμή Νόρμα
Maximum-Absolute-Value Norm	Μέγιστη-κατ' Απόλυτη Τιμή Νόρμα
Dual Norm	Δυϊκή Νόρμα
Nuclear Norm	Πυρηνική Νόρμα
Schatten 1-Norm	Schatten 1-Νόρμα
Ky Fan r-norm	Ky Fan r-Νόρμα
Hölder's Inequality	Ανισότητα του Hölder
Linear Programming	Γραμμικός Προγραμματισμός
Non-Linear Programming	Μη-Γραμμικός Προγραμματισμός
Sparsity	Αραιότητα
Sparse	Αραιός
Underdetermined System	Αόριστο Σύστημα
Convex Relaxation	Κυρτή Χαλάρωση
Convex Hull / Convex Envelope	Κυρτή Θήκη / Κυρτό Περίβλημα
Nullspace	Μηδενοχώρος
Mutual Coherence	Αμοιβαία Συνοχή
Welch Bound	Όριο Welch
Grassmanian Frames	Grassmanian Πλαίσια
Restricted Isometry Property	Ιδιότητα της Περιορισμένης Ισομετρίας

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

ID	Intrinsic Dimension
PCA	Principal Component Analysis
SNR	Signal-to-Noise Ratio
KLT	Karhunen-Loève Transform
MSE	Mean Square Error
LTI	Linear Transform Invariant
KPCA	Kernel Principal Component Analysis
ICA	Independent Component Analysis
SVD	Singular Value Decomposition
RPCA	Robust Principal Component Analysis
PCP	Principal Component Pursuit
MC	Matrix Completion
SDP	Semi-Definite Programming
SVT	Singular Value Thresholding
CS	Compressed Sensing / Compressive Sampling
KKT	Karush-Kuhn-Tucker
IT	Iterative Thresholding
APG	Accelerated Proximal Gradient
PG	Proximal Gradient
DM	Dual Method
AP	Alternating Projection
PM	Power Method
ALM	Augmented Lagrange Multiplier
EALM	Exact Augmented Lagrange Multiplier
IALM	Inexact Augmented Lagrange Multiplier
ADM	Alternating Direction Method
ADMM	Alternating Direction Method of Multipliers
KPI	Key Performance Indicators
I.I.D. / i.i.d.	Independent and Identically Distributed
PAPG	Partial Accelerated Proximal Gradient
PSNR	Peak-Signal-to-Noise-Ratio
LP	Linear Programming
NLP	Non-Linear Programming

RIP	Restricted Isometry Property

APPENDIX I: MEASURE THEORY AND FUNCTIONAL ANALYSIS

In this Appendix we give a brief review of some basic concepts from measure theory as well as from functional analysis. The treatment is by no means complete, and is meant mostly to set out our notation.

Vector Norms

Norms give a rough measure of the magnitude of the entries in vectors and matrices. They generalize the notion of *Absolute Value* for real numbers. In general, a function $f: R^n \to R$ is called a *Norm* if it respects the following four requirements:

1. $f(\boldsymbol{x}) \geq 0$, for all $\boldsymbol{x} \in R^n$ (f is non-negative)
2. $f(\boldsymbol{x}) = 0 \leftrightarrow \boldsymbol{x} = \boldsymbol{0}$ (f is definite)
3. $f(c\boldsymbol{x}) = |c| f(\boldsymbol{x})$, for all $\boldsymbol{x} \in R^n$ and $c \in R$ (f is positively homogenous)
4. $f(\boldsymbol{x} + \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y})$, for all $\boldsymbol{x}, \boldsymbol{y} \in R^n$ (f satisfies the *Triangle Inequality*)

The most widely used notation for norm functions is $f(x) = \|\boldsymbol{x}\|$, in order to indicate that in reality they form a generalization of the absolute value on R.

Of course there are many different kinds of norms, according to what we desire to compute. In order to have a general symbolism for all of them, it is commonly used the notation $\|\boldsymbol{x}\|_{subscript}$ where the subscript indicates exactly to which norm we refer to.

$\boldsymbol{L_p}$ (Holder) Vector Norms

The most widely used vector norms are the well-known l_p-norms (with $p \geq 1$), which are defined as follows:

Definition A.1.1: For a given vector $\boldsymbol{x} = (x_1, x_2, \dots, x_n)^T \in R^n$, its l_p-norm is defined as:

$$\|\boldsymbol{x}\|_p \equiv \left(\sum_{i=1}^n |x_i|^p\right)^{\frac{1}{p}} = \left(|x_1|^p + |x_2|^p + \cdots + |x_n|^p\right)^{\frac{1}{p}} \textbf{ (A.1.1)}$$

The simplest l_p-norm is the l_1-norm, which is given by equation **(A.1.1)** for $p = 1$:

$$\|\boldsymbol{x}\|_1 \equiv \sum_{i=1}^n |x_i| = |x_1| + |x_2| + \cdots + |x_n| \textbf{ (A.1.2)}$$

, which is also known as the *Sum-Absolute-Value Norm* or the *Taxicab Norm* or the *Manhattan Norm*.

Perhaps the most widely used l_p-norm is the l_2-norm, which is given by equation **(A.1.1)** for $p = 2$:

$$\|\boldsymbol{x}\|_2 \equiv \left(\sum_{i=1}^n |x_i|^2\right)^{\frac{1}{2}} = \left(|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2\right)^{\frac{1}{2}} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} \textbf{ (A.1.3)}$$

, which of course is the well-known *Euclidean Norm*.

Another norm which is used in practice is the l_∞-norm, which is derived by the definition of the l_p-norm if we let $p \to \infty$:

$$\|\boldsymbol{x}\|_\infty \equiv \lim_{p\to\infty} \|\boldsymbol{x}\|_p = \lim_{p\to\infty} \left(\left|x_{i_{max}}\right|^p \sum_{i=1}^n \left(\frac{|x_i|}{|x_{i_{max}}|}\right)^p\right)^{\frac{1}{p}} = \left|x_{i_{max}}\right| = \max_{1\leq i\leq n}\{|x_i|\} \textbf{ (A.1.4)}$$

, which is also known as the *Infinity Norm* or the *Chebyshev Norm*.

The four requirements for a function to be a norm guarantee that in fact it is also a convex function. This is a very important property, due to the fact that the minimization of a convex function leads to a unique solution (i.e. a global optimum), as it is known from optimization theory. As a result, the choice of norm-based functions as objective functions in optimization problems seems quite reasonable.

Last but not least, a widely used norm in many application fields (such as signal processing and statistics) is the l_0-norm, which is derived by the definition of the l_p-norm if we let $p \to 0$:

$$\|\boldsymbol{x}\|_0 \equiv \lim_{p\to 0} \|\boldsymbol{x}\|_p^p = \lim_{p\to 0} \sum_{i=1}^n |x_i|^p = \#\{i: x_i \neq 0\} \textbf{ (A.1.5)}$$

, and which represents the number of the non-zero coordinates of the vector $\boldsymbol{x}$, or otherwise the *Hamming Distance* of $\boldsymbol{x}$ from zero. It should be mentioned here that in reality the l_0-norm is not a true norm, because it does not fulfil the third requirement for

being one as: $\|a\boldsymbol{x}\|_0 \neq |a|\|\boldsymbol{x}\|_0, \forall a \neq 1$. However, it has come to be considered as a norm, with a slight abuse of the strict sense of the definition.

Unit Ball and Geometrical Interpretation of l_p-Norms

Definition A.1.2: The set of all vectors with norm less than or equal to one:

$$UB = \{\boldsymbol{x} \in R^n : \|\boldsymbol{x}\| \leq 1\} \quad \textbf{(A.1.6)}$$

is called the *Unit Ball* of the norm $\|\cdot\|$. This set is very important for the geometrical interpretation of the norm functions, because it exhibits the following properties:

1. UB is symmetric about the origin
2. UB is convex
3. UB is closed, bounded, and has nonempty interior

The concept of unit ball is different in different norms. Its' geometrical depiction is based on the isovalue curves on the *Euclidean Space* that correspond to the respective norm. More precisely, these isovalue curves for the l_0, l_1, l_2 and l_∞-norms are depicted in the figure below:

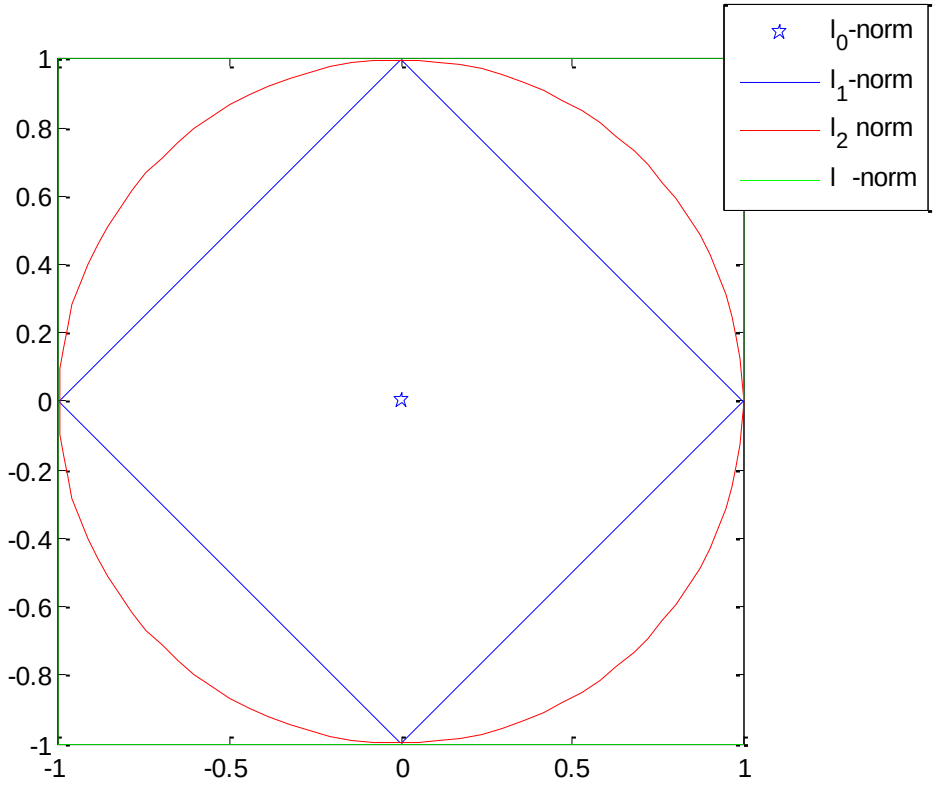

Figure 32: Unit ball for different values of p

As it is clear from Figure 32, the unit ball in R^2 is a:

- Single point at the origin of the coordinate axes $(0,0)$, for the l_0-norm
- Rhombus, for the l_1-norm
- Circle, for the l_2-norm
- Square, for the l_∞-norm

Generally, for any l_p-norm, it is a superellipse (with congruent axes). The fact that the isovalue curve of the l_0-norm consists of only one point is a direct consequence of its discrete nature, as well as that it is not a norm in the strict sense.

Matrix Norms

As well as vector norms, there exists also matrix norms. In most cases, they are defined in terms of the respective vector norms. Their meaning is to have a measure of the size of a matrix, as happens with vectors and vector norms respectively.

Just as in the definition of vector norms, a function $\|\boldsymbol{A}\|: R^{m\times n} \to R$ is called a *Matrix Norm* if it satisfies the following five requirements:

1. $\|\boldsymbol{A}\| \geq 0$, for all $\boldsymbol{A} \in R^{m\times n}$ ($\|\boldsymbol{A}\|$ is non-negative)
2. $\|\boldsymbol{A}\| = 0 \leftrightarrow \boldsymbol{A} = \boldsymbol{0}$ ($\|\boldsymbol{A}\|$ is definite)
3. $\|c\boldsymbol{A}\| = |c|\|\boldsymbol{A}\|$, for all $\boldsymbol{A} \in R^{m\times n}$ and $c \in R$ ($\|\boldsymbol{A}\|$ is positively homogenous)
4. $\|\boldsymbol{A}+\boldsymbol{B}\| \leq \|\boldsymbol{A}\| + \|\boldsymbol{B}\|$, for all $\boldsymbol{A},\boldsymbol{B} \in R^{m\times n}$ ($\|\boldsymbol{A}\|$ satisfies the triangle inequality)
5. $\|\boldsymbol{A}\boldsymbol{x}\| \leq \|\boldsymbol{A}\|\|\boldsymbol{x}\|$ and $\|\boldsymbol{A}\boldsymbol{B}\| \leq \|\boldsymbol{A}\|\|\boldsymbol{B}\|$, for all $\boldsymbol{A},\boldsymbol{B} \in R^{m\times n}$ and $\boldsymbol{x} \in R^n$

In contrast with vector norms, matrix norms must additionally satisfy the fifth above requirement, a consequence of the fact that matrices are more than just vectors as they multiply. That specific requirement demands that the size (growth) of this multiplication must stay under control.

Operator Norms

Definition A.1.3: Suppose there is available a matrix $\boldsymbol{A} \in R^{m\times n}$ and a norm $\|\cdot\|_a$ on R^m as well as one $\|\cdot\|_b$ on R^n. Then, the *Operator Norm* of matrix $\boldsymbol{A}$, which is induced by the norms $\|\cdot\|_a$ and $\|\cdot\|_b$, is defined as:

$$\|\boldsymbol{A}\|_{a,b} = sup\{\|\boldsymbol{A}\boldsymbol{x}\|_a : \|\boldsymbol{x}\|_b \leq 1\} = sup\left\{\frac{\|\boldsymbol{A}\boldsymbol{x}\|_a}{\|\boldsymbol{x}\|_b} : \boldsymbol{x} \neq \boldsymbol{0}\right\} \textbf{ (A.1.7)}$$

Furthermore, it can be proved (via the use of the famous *Rayleigh Quotient*) that when $\|\cdot\|_a$ and $\|\cdot\|_b$ are both Euclidean norms, the operator norm of the matrix $\boldsymbol{A}$ is its maximum singular value, and is defined as:

$$\|\boldsymbol{A}\|_2 = \sigma_{max}(\boldsymbol{A}) = \left(\lambda_{max}(\boldsymbol{A}^T\boldsymbol{A})\right)^{\frac{1}{2}} = \sqrt{\lambda_{max}(\boldsymbol{A}^T\boldsymbol{A})} \textbf{ (A.1.8)}$$

, which is also known as the l_2-norm or the *Spectral Norm*. If the matrix $\boldsymbol{A}$ is a square symmetric matrix then: $\|\boldsymbol{A}\|_2 = |\lambda_{max}(\boldsymbol{A})|$.

The matrix norm which corresponds to the l_1-norm on R^m and R^n is defined as:

$$\|\boldsymbol{A}\|_1 = \max_{1\leq j\leq n}\left\{\sum_{i=1}^{m}\left|a_{ij}\right|\right\} \textbf{ (A.1.9)}$$

, which is also known as the *Max-Column-Sum Norm*.

Finally, the matrix norm which corresponds to the l_∞-norm on R^m and R^n is defined as:

$$\|\boldsymbol{A}\|_\infty = \max_{1\leq i\leq n}\left\{\sum_{j=1}^{n}\left|a_{ij}\right|\right\} \textbf{ (A.1.10)}$$

, which is also known as the *Max-Row-Sum Norm*.

From a computational point of view, we should not forget to mention that the computation of $\|\boldsymbol{A}\|_2$ is generally more demanding than those of $\|\boldsymbol{A}\|_1$ and $\|\boldsymbol{A}\|_\infty$.

"Elementwise" matrix norms

These matrix norms treat a $m \times n$ matrix $\boldsymbol{A}$ as a (huge) vector of size $m \times n$, and afterwards use one of the familiar vector norms for the respective definition.

For example, using the l_p-norm for vectors, we get the following definition for the *"elementwise" matrix norms*:

$$\|\boldsymbol{A}\|_p = \|vec(\boldsymbol{A})\|_p = \left(\sum_{i=1}^{m}\sum_{j=1}^{n}\left|a_{ij}\right|^p\right)^{\frac{1}{p}} = (|a_{11}|^p + |a_{12}|^p + \cdots + |a_{mn}|^p)^{\frac{1}{p}} \textbf{ (A.1.11)}$$

An "elementwise" matrix norm which is commonly used is the *Frobenius Norm* of a matrix $\boldsymbol{A} \in R^{m\times n}$, which is defined as:

$$\|\boldsymbol{A}\|_F = \left(tr(\boldsymbol{A}^T\boldsymbol{A})\right)^{\frac{1}{2}} = \left(\sum_{i=1}^{m}\sum_{j=1}^{n}\boldsymbol{X}_{ij}^2\right)^{\frac{1}{2}} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\boldsymbol{X}_{ij}^2} \textbf{ (A.1.12)}$$

, which is also called the *Hilbert-Schmidt Norm* -though the latter term is often reserved for operators on Hilbert spaces. Its name comes from the well-known *Frobenius Inner Product* on the space of all matrices, which is the component-wise *Inner Product* of two matrices as though they were vectors. In reality, the Frobenius norm of a matrix is the Euclidean norm of the vector obtained by listing the coefficients of the matrix. At this point we should emphasize the fact that the Frobenius norm of a matrix does not coincide with its l_2-norm defined above.

Two equally well-known "elementwise" matrix norms are the *Sum-Absolute-Value Norm*, which arises from equation **(A.1.11)** for $p = 1$ and is defined as:

$$\|\boldsymbol{A}\|_{sav} = \sum_{i=1}^{m}\sum_{j=1}^{n}\left|a_{ij}\right| \textbf{ (A.1.13)}$$

, and the *Maximum-Absolute-Value Norm*, which arises from equation **(A.1.11)** for $p = \infty$ and is defined as:

$$\|\boldsymbol{A}\|_{mav} = \max_{1\le i\le m, 1\le j\le n}\left\{\left|a_{ij}\right|\right\} \textbf{ (A.1.14)}$$

Dual Norm

Definition A.1.4: Let $\|\cdot\|$ be a norm on R^n. The associated *Dual Norm*, denoted $\|*\|_d$, is defined as:

$$\|\boldsymbol{y}\|_d = sup\{\|\boldsymbol{y}^T\boldsymbol{x}\|: \|\boldsymbol{x}\| \le 1\} = sup\left\{\frac{\|\boldsymbol{y}^T\boldsymbol{x}\|}{\|\boldsymbol{x}\|}: \boldsymbol{x} \ne \boldsymbol{0}\right\} \textbf{ (A.1.15)}$$

As expected, the dual norm of the dual norm is the original norm:

$$\|\boldsymbol{x}\|_{d_d} = \|\boldsymbol{x}\| \textbf{ (A.1.16)}$$

, for all $\boldsymbol{x}$.

For all known l_p-norms there exists a respective dual norm l_q, where the subscripts satisfy the equation:

$$\frac{1}{p} + \frac{1}{q} = 1 \textbf{ (A.1.17)}$$

Taken this into mind, it can be easily proved that:

- The dual norm of the l_1-norm is the l_∞-norm
- The dual norm of the l_∞-norm is the l_1-norm
- The dual norm of the Euclidean norm is the Euclidean norm

A more interesting example though is that of the dual norm of the spectral or l_2-norm, as we refer to an operator norm. Then, applying the definition of the dual norm, we have:

$$\|\boldsymbol{A}\|_{2_d} = \|\boldsymbol{A}\|_* = sup\{\|\boldsymbol{A}^T\boldsymbol{x}\|: \|\boldsymbol{x}\|_2 \le 1\} = sup\left\{\frac{\|\boldsymbol{A}^T\boldsymbol{x}\|}{\|\boldsymbol{x}\|_2}: \boldsymbol{x} \ne \boldsymbol{0}\right\} = tr(\boldsymbol{A}^T\boldsymbol{A})^{\frac{1}{2}} = \sqrt{tr(\boldsymbol{A}^T\boldsymbol{A})} = \sigma_1(\boldsymbol{A}) + \sigma_2(\boldsymbol{A}) + \cdots + \sigma_r(\boldsymbol{A}) \textbf{ (A.1.18)}$$

, where $r = rank(\boldsymbol{A})$. This norm is known as the *Nuclear Norm*, though it has nothing to do with nuclear physics. It is also known by several other names such as the *Schatten* 1*-norm* (the Schatten p-norms arise when applying the l_p-norm to the vector of singular values of a matrix) or the *Ky Fan* r*-norm* (in general, the Ky Fan r-norm of a given matrix $\boldsymbol{A}$ is the sum of its r largest singular values).

Equivalence of norms

We conclude this first Appendix with a significant result from functional analysis, which in reality states that all norms (vector and matrix) are equivalent.

More precisely, supposed that $\|\cdot\|_a$ and $\|\cdot\|_b$ are norms on R^n, then there exist positive real numbers $c > 0$, $d > 0$ such that:

$$c\|\boldsymbol{x}\|_a \le \|\boldsymbol{x}\|_b \le d\|\boldsymbol{x}\|_a \textbf{ (A.1.19)}$$

, for all vectors $\boldsymbol{x} \in R^n$.

A more general form of this result states that on C^n, if $0 < a < b$, then:

$$\|\boldsymbol{x}\|_b \le \|\boldsymbol{x}\|_a \le n^{\left(\frac{1}{a}-\frac{1}{b}\right)}\|\boldsymbol{x}\|_b \textbf{ (A.1.20)}$$

, for all $\boldsymbol{x} \in R^n$.

In particular, it can be easily proven that the following equations are true:

- $\frac{1}{\sqrt{n}}\|\boldsymbol{x}\|_1 \le \|\boldsymbol{x}\|_2 \le \|\boldsymbol{x}\|_1$
- $\frac{1}{n}\|\boldsymbol{x}\|_1 \le \|\boldsymbol{x}\|_\infty \le \|\boldsymbol{x}\|_1$
- $\|\boldsymbol{x}\|_2 \le \|\boldsymbol{x}\|_1 \le \sqrt{n}\|\boldsymbol{x}\|_2$
- $\frac{1}{\sqrt{n}}\|\boldsymbol{x}\|_2 \le \|\boldsymbol{x}\|_\infty \le \|\boldsymbol{x}\|_2$
- $\|\boldsymbol{x}\|_\infty \le \|\boldsymbol{x}\|_1 \le n\|\boldsymbol{x}\|_\infty$
- $\|\boldsymbol{x}\|_\infty \le \|\boldsymbol{x}\|_2 \le \sqrt{n}\|\boldsymbol{x}\|_\infty$

The important thing about equivalent norms is that they define the same notions of continuity and convergence and for many purposes do not need to be distinguished. To

be more precise, the uniform structure defined by equivalent norms on the vector space is uniformly isomorphic. In fact, if the vector space is a finite-dimensional real or complex one, all norms are equivalent. On the other hand, in the case of infinite-dimensional vector spaces, this result need not hold.

As for vector norms, there holds a similar equivalence result for matrix norms as well. More precisely, supposed that $\|\cdot\|_a$ and $\|\cdot\|_b$ are norms on R^n, then there exist positive real numbers $c > 0$, $d > 0$ such that:

$$c\|\boldsymbol{A}\|_a \leq \|\boldsymbol{A}\|_b \leq d\|\boldsymbol{A}\|_a \quad \textbf{(A.1.21)}$$

, for all matrices $\boldsymbol{A} \in R^{m\times n}$. In this occasion, the equivalence of norms means that they induce the same topology on $R^{m\times n}$. Of course, just as before, such a result holds true because the vector space $R^{m\times n}$ has a finite dimension equal to $m \times n$.

In particular, just as above, it can be easily proven that the following equations concerning matrix norms' equivalence are true:

- $\|\boldsymbol{A}\|_2 \leq \|\boldsymbol{A}\|_F \leq \sqrt{r}\|\boldsymbol{A}\|_2$
- $\|\boldsymbol{A}\|_F \leq \|\boldsymbol{A}\|_* \leq \sqrt{r}\|\boldsymbol{A}\|_F$
- $\|\boldsymbol{A}\|_{max} \leq \|\boldsymbol{A}\|_2 \leq \sqrt{mn}\|\boldsymbol{A}\|_{max}$
- $\frac{1}{\sqrt{n}}\|\boldsymbol{A}\|_\infty \leq \|\boldsymbol{A}\|_2 \leq \sqrt{m}\|\boldsymbol{A}\|_\infty$
- $\frac{1}{\sqrt{m}}\|\boldsymbol{A}\|_1 \leq \|\boldsymbol{A}\|_2 \leq \sqrt{n}\|\boldsymbol{A}\|_1$

, where $\boldsymbol{A} \in R^{m\times n}$, $r = rank(\boldsymbol{A})$ and $\|\boldsymbol{A}\|_p$ refers to the matrix norm induced by the respective l_p-vector norm (operator norm).

Finally, a useful inequality between matrix norms which is commonly used in practice is the following one:

$$\|\boldsymbol{A}\|_2 \leq \sqrt{\boldsymbol{A}_1\boldsymbol{A}_\infty} \quad \textbf{(A.1.22)}$$

, which is nothing more than a special case of the famous *Hölder's Inequality*.

APPENDIX II: CONVEX OPTIMIZATION AND CONVEX ANALYSIS

In this Appendix we give a brief review of some basic concepts from convex optimization and convex analysis. As in the previous Appendix, the treatment is also here by no means complete, and again is meant mostly to set out our notation.

Mathematical Optimization Problems

As it is known from optimization theory, a mathematical optimization problem has the following general form:

$$\min_{s.t. f_i(\boldsymbol{x}) \leq b_i} f(\boldsymbol{x}), i = 1,2,\dots,m \quad \textbf{(A.2.1)}$$

, where:

- $\boldsymbol{x}$ is the optimization variable
- f is the objective function
- f_i are the constraint functions
- b_i are constants, which represent the limits for the constraints (of course there can be equality constraints instead of inequality ones)

The goal of such a problem is to find the optimal vector $\boldsymbol{x}_*$ that fits the constraints and has the minimum objective value, among all possible vectors fitting the constraints.

Depending on the "form" of the objective function, as well as that of the constraint ones, there arise different classes of optimization programs. For example, if the objective function and the constraint functions are linear functions, then the respective optimization problem is called a *Linear Programming* (LP) program. On the other hand, if they are not linear, the arising optimization problem is called a *Non-Linear Programming* (NLP) program.

A very interesting class of optimization problems are the so called convex optimization problems, in which class the objective and constraint functions are convex functions. The reason why this class of problems is so interesting is that the resulting solution is guaranteed to be not only a local optimum solution to the problem, but in fact the global one.

A solution method for a particular optimization problem lies in finding an algorithmic scheme that computes the desired solution. Of course, for every different class of optimization problems, there is available a variety of algorithms which are able to solve them. Each one of them has its special features and characteristics, making it the "appropriate choice" for particular problems. It is exactly this adjective, i.e. "particular", that discriminates an effective algorithm for an optimization problem from a non-effective one. Reading between the lines, that means that there exist certain factors which affect the behavior of an algorithm aiming to solve an optimization problem, some of whom are the particular form of the objective and constraint functions, the number of the variables and the constraints, and the possible special structures among them-such as *Sparsity* (a problem is *Sparse* if each constraint function depends on only a small number of the variables).

Norm-Based Objective Functions

As mentioned before, a widely occurring class of optimization problems are the convex optimization problems, due to the fact that the convexity of the objective function guarantees the uniqueness of the solution. As it became quite clear in the Appendix I, the l_p-norms ($p \geq 1$) are convex functions, so they form a potential choice as objective functions. The goal of this Appendix is to demonstrate the utility of such a choice in problems at which some extra information is known about the structure of the solution, concerning more specifically sparsity and low rankness. In order to achieve that, we formulate the desired goal as the following optimization program:

Suppose we are given a system of linear equations:

$$\boldsymbol{Ax} = \boldsymbol{b} \quad \textbf{(A.2.2)}$$

, with $\boldsymbol{b} \in R^m, \boldsymbol{x} \in R^n, \boldsymbol{A} \in R^{m \times n}, rank(\boldsymbol{A}) = m$ and $m < n$. Of course, such a system has in general infinitely many solutions, due to the fact that it has fewer equations than unknowns, and is widely known as an *Underdetermined System*. For a fixed set of measurements $\boldsymbol{b}$, and a specific transformation matrix $\boldsymbol{A}$, the set of possible solutions obviously depends directly on $\boldsymbol{x}$ and its structure. Therefore, if we desire to narrow the choice to one well-defined solution, additional criteria concerning $\boldsymbol{x}$ are needed.
A very interesting idea is to impose the sparsity constraint on the solution $\boldsymbol{x}$ (i.e. we seek for a vector $\boldsymbol{x}$ with few non-zero entries), as we could be able to solve the resulting optimization problem using techniques from convex analysis. Taken that into mind, there exist three possible norm-based objective functions to be minimized in order to achieve our goal of obtaining the sparsest solution of our underdetermined system, which are examined in details below:

1. l_2-norm
2. l_0-norm
3. l_1-norm

$\boldsymbol{l_2}$-Norm Minimization

In this case, the goal is to choose the optimal vector $\boldsymbol{x}_*$ that is consistent with the underdetermined system **(A.2.2)** in such a way that it has the minimum l_2-norm. In other words, we have to solve the following optimization problem:

$$\min_{s.t. \boldsymbol{a}_n^T \boldsymbol{x} = \boldsymbol{b}_n} \|\boldsymbol{x}\|_2^2, n = 1,2,\dots m \quad \textbf{(A.2.3)}$$

Of course, this problem has a unique solution, which is given by the following formula:

$$\boldsymbol{x}_* = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1}\boldsymbol{b} \quad \textbf{(A.2.4)}$$

, a fact that may seem quite encouraging as the solution is given in closed form. But there is one drawback in the whole process, which unfortunately is directly connected with our primal goal of recovering a solution as sparse as possible. This can become more evident if we consider the geometrical interpretation of the l_2-norm minimization process, which is depicted in the following image:

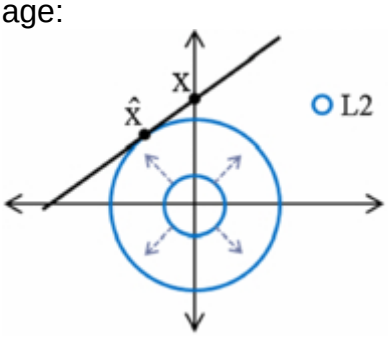

Image 64: $\boldsymbol{l_2}$-norm minimization

What is obvious from Image 64 is that by minimizing the l_2-norm we cannot expect to recover the sparsest solution. This is explained in the following sense: let's suppose that the sparse solution we seek to recover is the point $\boldsymbol{X}$ in Image 64 (it is sparse because its coordinate on the x-axis is zero whereas that on the y-axis is not), of which we make one measurement. As it is known from middle-school mathematics, the (feasible) set of all x that share the same measurement value is a line, and more precisely the black one in Image 64. The task of finding the point on this line with the minimum l_2-norm is accomplished by expanding the radius of the spherical and completely isotropic Euclidean l_2 ball until it intersects the line. By definition, the first point at which this happens is the one with minimum l_2-norm, and constitutes the solution we are seeking for. As it is clear from Image 16, this point (i.e. the point $\widehat{\boldsymbol{X}}$) does not have to be sparse as we desired. Although in low dimensions the obtained solution via the l_2-norm minimization looks close to the desired one, the occurring situation in high dimensions is a disaster. As a result, even if the minimization of the l_2-norm is a well-posed and tractable optimization

program, it gives no guarantee that the recovered solution will be a sparse one -as we desire.

l_0-Norm Minimization

In this case, the goal is to choose the optimal vector $\boldsymbol{x}_*$ that is consistent with the underdetermined system **(A.2.2)** in such a way that it has the minimum l_0-norm. In other words, we have to solve the following optimization problem:

$$\min_{s.t.\boldsymbol{a}_n^T\boldsymbol{x}=\boldsymbol{b}_n} \|\boldsymbol{x}\|_0 , n = 1,2,\dots,m \quad \textbf{(A.2.5)}$$

, which seems to be exactly what we should be doing in order to obtain the sparsest possible solution -given the meaning of the l_0-norm. Although the optimization problem in **(A.2.5)** looks like that of in **(A.2.3)**, in reality they bear significant differences. The solution to the l_2-norm minimization problem is always unique, and can be computed via the usage of innovative numerical linear algebra methods and tools. On the other hand, the solution to the l_0-norm minimization problem is a more complex task. This comes from the fact that the l_0-norm is not a true norm function in the strict sense of the definition, as mentioned in Appendix I, but in reality is a discrete and discontinuous function. The bad news that arise from this fact are that all the available tools from optimization theory of convex functions are automatically useless, making the questions of uniqueness and verification of the solutions seem at least daunting.

To make things even worse, it has been proved in [47] that a direct solution to the l_0-norm minimization problem under linear equality constraints is infeasible, and the problem is generally considered as a NP-hard one. This comes out essentially from the fact that in reality it is a classical combinatorial problem of exhaustive search, in which we should systematically enumerate all possible candidates for the solution and check whether each candidate satisfies the problem's statement. While an exhaustive search is simple to implement, and will always find a solution if it exists, its cost is proportional to the number of candidate solutions -which in many practical problems tends to grow very quickly as the size of the problem increases. In our case, the complexity of adopting such a technique is proved to be exponential in n, making it prohibitive for real-world problems.

l_1-Norm Minimization

From so far analysis, the gap between the l_0-norm minimization problem and the l_2-norm minimization problem may seem unbridgeable. This gap comes to fill in another l_p-norm, and more precisely the l_1-norm. Now, the optimization problem has the following form:

$$\min_{s.t.\boldsymbol{a}_n^T\boldsymbol{x}=\boldsymbol{b}_n} \|\boldsymbol{x}\|_1 , n = 1,2,\dots,m \quad \textbf{(A.2.6)}$$

As in the l_2-norm case, the geometrical interpretation of the minimization process is illustrated in the following image:

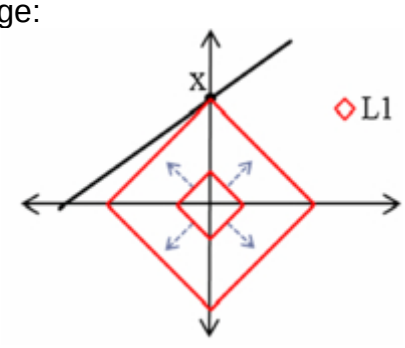

Image 65: l_1-norm minimization

Following the same reasoning as before, the radius of the anisotropic and “pointy” along the axes l_1 ball is expanded until it intersects the black line representing the (feasible) set of solutions. The difference now is that the flatness of the line (in general, of the plane) of solutions combined with the diamond-shaped l_1 ball results in picking a solution point which is extremely sparse, as we desired. As a result, the minimization of the l_1-norm

seems a good proxy for the minimization of the l_0-norm, which identically we would desire, compared to that of the l_2-norm. Despite the seemingly negligible differences among their definitions, and even the fact that they are both convex functions and as a result the task of finding the optimal solution becomes computationally tractable, the l_1-norm is the only one among the l_p-norms ($p \geq 1$) which respects small values and results in recovering sparse solutions. From an algorithmic point of view, the l_1-norm minimization task can be recast as a LP program, and consequently be solved by anyone of the related methods. Last but not least, we should highlight the fact that sparse signals (which we seek for) have small l_1-norms relative to their energy -as stated in [54]- a fact that overbids the preference of minimizing the l_1-norm instead of the l_2 one.

Convex Relaxation: A Priceless Treasure

As mentioned before, minimizing the l_1-norm would be a good idea, in the context of recovering sparse solutions to underdetermined system of equations, instead of trying to cope with the combinatorically hard task of minimizing the l_0-norm. But such a choice is not as simple as it seems, given the fact that the l_1-norm is a totally different "creature" from the l_0-norm -which is not even a norm in the strict sense as mentioned in Appendix I. The transition from a non-convex objective function to a convex one should be very cautious, even if it is widely known that convex optimization techniques play an important role in problems that are not convex. In fact, convex optimization is the basis for several heuristics for solving non-convex problems, and this is justified by the fact that convexity implies that every local minimum / maximum is also a global one -which is not guaranteed when considering non-convex problems.

When facing situations in which our optimization problem (convex or not) is a NP-hard one (as here with the minimization of the l_0-norm), a common practice is to relax it. What practically relaxation techniques do is approximating difficult problems of constrained optimization by simpler ones, which are solvable in polynomial time. As a result, a solution to the relaxed problem is an approximate solution to the original problem, and therefore provides useful information. Given the fact that norm-based objective functions are convex functions as mentioned in the Appendix I, the relative relaxation is called *Convex Relaxation*, and is defined as below:

Definition A.2.1: For a non-convex function $f: S \rightarrow R$, where S is a non-empty convex subset of R^n, a convex function $u: S \rightarrow R$ will be called a convex relaxation of $f(\boldsymbol{x})$ if it obeys the following inequality:

$$u(\boldsymbol{x}) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in S \quad \textbf{(A.2.7)}$$

The above inequality condition is depicted in the Euclidean plane below, so as its' geometrical interpretation to become sufficiently clarified:

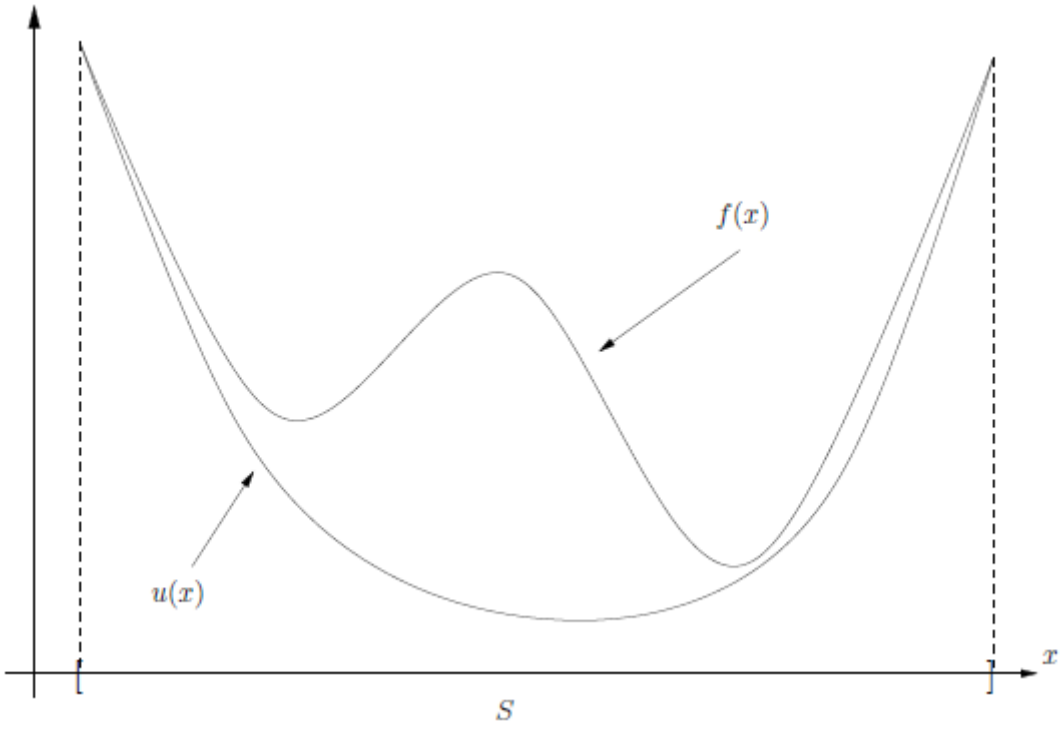

Image 66: Geometrical interpretation of convex relaxation

It is worth mentioning that among all convex relaxations $u(\boldsymbol{x})$ of a non-convex function $f(\boldsymbol{x})$, the one for which holds true the following inequality:

$$u(\boldsymbol{x}) \leq f_S(\boldsymbol{x}), \forall \boldsymbol{x} \in S \textbf{ (A.2.8)}$$

is called the *Convex Hull* / *Convex Envelope* of $f(\boldsymbol{x})$ over S. In other words, the convex hull is the tightest possible convex relaxation of a nonconvex function.

Taking into mind the above definitions, it would seem apparent to consider the l_1-norm minimization task as the convex relaxation of the l_0-norm minimization one. In fact, such a conclusion holds true, as it is shown below. But before this, it is wise to reveal those conditions under which both tasks recover a unique solution to the underdetermined system of equations (**A.2.2**).

Uniqueness of the Solution recovered by l_1-Norm Minimization

Concerning the uniqueness of the solution of the l_1-norm minimization task, the following Lemma is stated without proof (the diligent reader is referred to the relative reference below):

Lemma A.2.1: An element $\boldsymbol{x} \in X$, where X is the set of the solutions of the underdetermined system of equations **(A.2.2)**, has minimal l_1-norm if and only if it obeys the following inequality condition:

$$\left|\sum_{i:x_i \neq 0} sgn(x_i) z_i\right| \leq \sum_{i:x_i=0} |z_i|, \forall \boldsymbol{z} \in null(X) \textbf{ (A.2.9)}$$

, where $null(X)$ is the *Nullspace* of X. Furthermore, the solution is unique if and only if the above inequality condition is a strict one $\forall \boldsymbol{z} \neq \boldsymbol{0}$, as stated in [51].

From linear algebra basics it is known that for the dimension of the nullspace of $\boldsymbol{A}$ holds the following equality:

$$dim\big(null(\boldsymbol{A})\big) = n - rank(\boldsymbol{A}) = n - m \textbf{ (A.2.10)}$$

Taken that in mind, and combining the above Lemma as well as the fact that a vector $\boldsymbol{x}$ is called S-sparse if it has at most S non-zero coordinates, it becomes clear that if there

exists a unique solution to the underdetermined system of equations (**A.2.2**) then it will be an S-sparse vector with $S \leq m$. This is a very important result, considering that it defines a strict upper bound for the number of the non-zero elements of the unique solution of **(A.2.2)**, as it states that for sure the recovered solution will be a sparse one.

A final remark that should be made concerning the minimization of the l_1-norm is that there are specific conditions that must hold true (as it became clear from the above Lemma) for the solution to be unique, which is clearly not the case for the respective minimization of the l_2-norm. This is a direct result of the "nature" of the l_1-norm which is not a strict convex function, unlike the l_2-norm which is one-and whose minimization will always lead to a unique solution.

Uniqueness of the Solution recovered by l_0-Norm Minimization

Following the same reasoning as before, the sufficient conditions that must hold true in order for the solution arising via the l_0-norm minimization task to be unique, are presented in form of two important Lemmas (again, the relevant proofs are omitted as they are beyond the scope of this thesis, however the adventurous reader is given the required references in order to confirm their validity).

Both of them contain a key matrix quantity, called *Spark*, which was defined in [24] as the smallest number of linearly dependent columns of a full rank matrix $\boldsymbol{A} \in R^{m \times n}, m > n$. By definition, the spark of a square $m \times m$ matrix $\boldsymbol{A}$ is m+1, and any number of columns of cardinality beyond the spark of a matrix are necessarily linearly independent.

A very interesting corollary arising from the definition of the spark of a matrix is stated in the above Lemma:

Lemma A.2.2: If $null(\boldsymbol{A})$ is the nullspace of matrix $\boldsymbol{A}$, then the following inequality holds true:

$$\|\boldsymbol{x}\|_0 \geq spark(\boldsymbol{A}), \forall \boldsymbol{x} \in null(\boldsymbol{A}), \boldsymbol{x} \neq \boldsymbol{0} \textbf{ (A.2.11)}$$

, where $spark(\boldsymbol{A})$ is the spark of the full-rank matrix $\boldsymbol{A} \in R^{m \times n}, m > n$.

Although the definition of the spark of a matrix bears a strong resemblance to that of its rank, in reality not only they differ conceptually, but in the same time the necessary effort needed for their computation deviates significantly. In fact, the spark of a matrix is NP-hard to compute, as a combinatorial search over all possible subsets of columns from $\boldsymbol{A}$ is needed, as stated in [24].

At first sight, the definition of the spark combined with the difficulty of its computation would seem redundant. On the contrary, this is clearly not the case, as it plays a crucial role in our attempt to discover those conditions needed to hold true for the uniqueness of the solution recovered via the l_0-norm minimization task, as it becomes crystal clear from the following Lemma ([24], [16]):

Lemma A.2.3: If the underdetermined system of equations **(A.2.2)** has a solution that satisfies the following inequality condition:

$$\|\boldsymbol{x}\|_0 < \frac{spark(\boldsymbol{A})}{2} \textbf{ (A.2.12)}$$

, then this solution is unique, and at the same time the sparsest one.

Although the result of the above Lemma is elementary, it is also a very interesting one. If we take a deeper look at inequality **(A.2.12)** we could easily understand that in reality it constitutes a sufficient condition for checking the optimality of a solution to an NP-hard problem of combinatorial flavor. And more precisely, this check concerns the global

optimality of the solution, and not the local one as it is usually the case when dealing with combinatorial optimization problems.

As it became clear from Lemma A.2.3, it is of crucial importance the value of the spark to be as large as possible, so that the sufficiency condition **(A.2.12)** holds true for a respective range as wide as possible. Fortunately, the range of sparks' values is known by definition, as also stated in [11], to be:

$$1 \leq spark(\boldsymbol{A}) \leq m + 1 \textbf{ (A.2.13)}$$

, and the construction of random matrices $\boldsymbol{A}$ with i.i.d. entries has been proven to lead to $spark(\boldsymbol{A}) = m + 1$, as we would ideally desire.

Clearly, the result of Lemma A.2.3 provides us the sufficient condition we were seeking for, but bearing into mind that the computation of the spark is a tough combinatorial problem would be a discouraging factor for sure. This fundamental obstacle is overcomed by the usage of another significant quantity of a matrix instead of its spark, which is much easier to compute, and is called *Mutual Coherence*. More formally, the mutual coherence is defined as follows:

Definition A.2.2: For a given matrix $\boldsymbol{A} \in R^{m \times n}$, its' mutual coherence is given by the following equation ([44], [24], [17], [11]):

$$\mu(\boldsymbol{A}) \equiv \max_{1 \leq i < j \leq n} \frac{|\boldsymbol{a}_i^T \boldsymbol{a}_j|}{\|\boldsymbol{a}_i\| \|\boldsymbol{a}_j\|} \textbf{ (A.2.14)}$$

, where $\boldsymbol{a}_i, i = 1,2, \dots, n$ denotes the $i - th$ column of matrix $\boldsymbol{A}$.

As it is obvious from its definition, mutual coherence measures the "similarity" between the columns of a matrix, computing for that reason the maximum value of the absolute inner product between them, renormalized properly. For general full-rank matrices with more columns than rows ($n > m$) mutual coherence is proven to satisfy the famous *Welch Bound* ([67]):

$$\sqrt{\frac{n-m}{m(n-1)}} \leq \mu(\boldsymbol{A}) \leq 1 \textbf{ (A.2.15)}$$

, with the equality being achieved for the so-called *Grassmanian Frames* matrices, as proven in [57]. Obviously, from the above inequality condition, square orthogonal matrices will always have zero mutual coherence. Another important conclusion is that if $m \ll n$ then the lower bound of **(A.2.15)** approaches $\frac{1}{\sqrt{m}}$.

Taking a deeper look at the definition of mutual coherence, we would surely desire its value to be as small as possible, for a matrix $\boldsymbol{A}$ satisfying the underdetermined system of equations **(A.2.2)**. This demand is in perfect alignment with the "service" that matrix $\boldsymbol{A}$ has to carry out: the linear combination of its columns with the respective coordinates of the unknown vector $\boldsymbol{x}$ must "absorb" as much information about $\boldsymbol{x}$ as possible, as the result of this process will derive the measurement vector $\boldsymbol{b}$. Therefore, it is of utmost importance the columns of matrix $\boldsymbol{A}$ to be as "less-similar" as possible, in order the recovery process of $\boldsymbol{x}$ to be as less toilsome as possible. Of course, the ideal scenario for matrix $\boldsymbol{A}$ is to have pairwise orthogonal columns (i.e. to be orthogonal), a regime however not very rare in practice ([17], [38]).

Having introduced the notion of mutual coherence as well as exposed its intuitive meaning, we should not forget the purpose of its usage. In reality, mutual coherence was mentioned in order to overcome the difficulty of computing the spark of a matrix, in our effort to check the uniqueness of the solution recovered via the l_0-norm minimization task.

It is time then to state the connection between them, in the form of a Lemma whose proof is available in [24]:

Lemma A.2.4: For any matrix $\boldsymbol{A} \in R^{m \times n}$ the following inequality holds:

$$spark(\boldsymbol{A}) \geq 1 + \frac{1}{\mu(\boldsymbol{A})} \textbf{ (A.2.16)}$$

The above inequality is quite important as it provides us a lower bound for the spark, which in turn can be computed much easier than the spark. We should not forget to mention here that it is clear from **(A.2.16)** that the spark of a matrix is inversely proportional to its mutual coherence, a fact that was in reality expected by their respective definitions: a small value of $\mu(\boldsymbol{A})$ indicates that its columns are not depending on each other, which of course leads to high values for the $spark(\boldsymbol{A})$ -as we desire.

It is now time to tie Lemmas A.2.3 and A.2.4 together, in order to conclude to a sufficient condition whose validity can be easily checked. The following important Theorem which achieves that purpose, was firstly stated in [24]:

Theorem A.2.1: If a system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$ has a solution $\boldsymbol{x}$ that satisfies the following condition:

$$\|\boldsymbol{x}\|_0 < \frac{1+\frac{1}{\mu(\boldsymbol{A})}}{2} \textbf{ (A.2.17)}$$

, then this solution is necessarily the sparsest one possible.

Obviously, the result of the above Theorem is of significant importance. As the mutual coherence of a matrix is not a difficult task to compute, we have an easily checkable sufficient condition in our hands. Of course, we should not forget that as the mutual coherence of a matrix constitutes a "relaxation" of its spark, so do their respective results concerning the sparsity level of the recovered solution. In fact, if we look closer to this tacit assumption, it can be easily proven that as the mutual coherence can never be lower than $\frac{1}{\sqrt{m}}$ (as stated below inequality condition **(A.2.15)**), then its corresponding bound from **(A.2.17)** will be $\frac{\sqrt{m}}{2}$. At the same time, an expected high value of the spark -proportional to m (identically, equal to m+1)- dictates that its corresponding bound from **(A.2.12)** will be $\frac{m}{2}$. This expansion of the sparsity level of the recovered solution is a direct consequence of the fact that in reality Lemma A.2.3 is far sharper than Theorem A.2.1, but at the same time far more difficult to check -a worthwhile and bearable tradeoff in practical applications though.

Equivalence of l_0-l_1-Norm Minimization Tasks

In the so far analysis that took place, those sufficient conditions that have to be fulfilled in order for the solutions recovered by the minimization of the l_0-norm as well as that of the l_1-norm to be unique became clear. As everyone could have suspected from the above analysis, the l_1-norm constitutes a convenient proxy for the l_0-norm, whose minimization is difficult combinatorial optimization task as we have seen so far. The crucial part though before adopting this powerful heuristic is to ensure that its minimization will lead to the same result as that of its discrete and discontinuous counterpart. Of course, in general such a conclusion can never always hold true (if it would, we would have solved practically the famous P=NP? problem!), but under suitable sufficient conditions it can. It is now time to state those important sufficient conditions:

Theorem A.2.2: For the system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$, where $A \in R^{m \times n}$ $(m < n)$ is a full-rank matrix, if a solution $\boldsymbol{x}$ exists, and at the same time it obeys the following inequality condition:

$$\|\boldsymbol{x}\|_0 < \frac{1+\frac{1}{\mu(A)}}{2} \textbf{ (A.2.17)}$$

, then this solution is the unique one of both the l_1-norm and l_0-norm minimization tasks.

The above very important Theorem A.2.2 was stated and proved independently in [24] and [32], while an elegant proof can also be found in [11]. Its validity is based on the notion of mutual coherence, defining a specific bound for the sparsity level of the recovered solution. Of course, this is not the only approach available concerning the equivalence of the l_0-l_1-norm minimization tasks. There exists a whole relative literature, containing numerous approaches seeking to prove this statement, each one adopting its own assumptions and producing its own results. Indicatively, the interested reader is referred to [20], [17], [64], [26], while a more informative list containing a summarized description of each method can be found in [11].

Obviously, a detailed presentation of each one of the methods available in the literature for achieving equivalence of the l_0-l_1-norm minimization tasks is beyond the scope of this Appendix -and this thesis in general. On the other hand, we should not forget to highlight the astonishing result of [17], in which is adopted a key notion that dominates the research directions on the CS framework -the so-called *Restricted Isometry Property* (RIP). More specifically, as it was pioneeringly stated in [17], we have the following definition:

Definition A.2.3: For each integer $S = 1,2, \ldots$, the isometry constant δ_S of a matrix $\boldsymbol{A}$ is defined as the smallest number such that the following inequality condition:

$$(1-\delta_S)\|\boldsymbol{x}\|_2^2 \leq \|\boldsymbol{A}\boldsymbol{x}\|_2^2 \leq (1+\delta_S)\|\boldsymbol{x}\|_2^2 \textbf{ (A.2.18)}$$

holds true for all S-sparse vectors $\boldsymbol{x}$. Furthermore, a matrix $\boldsymbol{A}$ is called to obey the RIP of order S if δ_S is not close to one.

What intuitively the RIP condition **(A.2.18)** says is that if it holds true, then the l_2-norm of $\boldsymbol{x}$ is "preserved" even after the operation of matrix $\boldsymbol{A}$ onto $\boldsymbol{x}$. An also interesting interpretation of the RIP condition **(A.2.18)** is that each submatrix $\boldsymbol{A}_S$ formed by combining at most S columns of matrix $\boldsymbol{A}$ has its nonzero singular values bounded below by $(1-\delta_S)$ and above by $(1+\delta_S)$ (a related work in [3] is also very informative), or in other words its columns are nearly orthogonal (they cannot be exactly orthogonal, since $m < n$).

In order for the importance of the above "preservation of the l_2-norm" to become clear, let's consider two S-sparse vectors $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and apply the RIP condition **(A.2.18)** to their difference-which in general is a $2S$-sparse vector:

$$(1-\delta_{2S})\|\boldsymbol{x}_1-\boldsymbol{x}_2\|_2^2 \leq \|\boldsymbol{A}(\boldsymbol{x}_1-\boldsymbol{x}_2)\|_2^2 \leq (1+\delta_{2S})\|\boldsymbol{x}_1-\boldsymbol{x}_2\|_2^2 \textbf{ (A.2.19)}$$

Clearly, **(A.2.19)** implies that all pairwise distances between S-sparse signals are well preserved in the measurement space, and therefore when δ_{2S} is "sufficiently small" the odds of recovering S-sparse vectors / signals with matrix $\boldsymbol{A}$ are surely not against us.

The significance of exhibiting the RIP is justified in the following Theorem ([17], [21], [18], [16], [14]):

Theorem A.2.3: Assume that $\delta_{2S} < \sqrt{2}-1$. Then, the solution $\boldsymbol{x}^*$ to the l_1-norm minimization task **(A.2.6)** satisfies the following two inequality conditions:

$$\begin{cases} \|\boldsymbol{x}^*-\boldsymbol{x}\|_1 \leq C_0\|\boldsymbol{x}-\boldsymbol{x}_S\|_1 \\ \|\boldsymbol{x}^*-\boldsymbol{x}\|_2 \leq \frac{C_0\|x-x_S\|_1}{\sqrt{S}} \end{cases} \textbf{ (A.2.20)}$$

, for some constant C_0, where $\boldsymbol{x}_S$ is the best sparse approximation of $\boldsymbol{x}$ if we knew exactly the locations and amplitudes of the S-largest entries of $\boldsymbol{x}$ (practically, $\boldsymbol{x}_S$ results from $\boldsymbol{x}$ if we set to zero all but its S-largest coordinates).

In particular, if $\boldsymbol{x}$ is S-sparse, the recovery is exact. On the other hand, if $\boldsymbol{x}$ is not S-sparse, then the accuracy of the recovered vector / signal depends, in some magic way, directly on the S-largest entries of $\boldsymbol{x}$. Furthermore, there is no piece of randomness in Theorem A.2.3, since everything is completely deterministic: if matrix $\boldsymbol{A}$ obeys the RIP condition with $\delta_{2S} < \sqrt{2} - 1$, then Theorem A.2.3 certifies that the recovered solution of the l_1-norm minimization task **(A.2.6)** would be either an exactly S-sparse vector or, in the worst case scenario, an approximately S-sparse one-whose S-largest values are definitely the "dominant" ones.

Advocating to the groundbreaking significance of the RIP condition, it would be a pity if we wouldn't mention that in the case of seeking an S-sparse solution (which is clearly our goal from the beginning), what Theorem A.2.3 claims is that in fact the l_1-norm minimization task and the l_0 one are formally equivalent. This assertion arises from two important interpretations of the isometry constant δ_{2S}([14]):

1. If $\delta_{2S} < 1$, then l_0-norm minimization task has a unique S-sparse solution -as it is shown in [16].
2. If $\delta_{2S} < \sqrt{2} - 1$, then the solution to the l_1-norm minimization task is the same as that of the l_0-norm minimization task. Of course, such a thing does not comes from the sky, and is makes completely sense: if we desire our convex relaxation to be exact, obviously we have also to decrease accordingly the range of values for the isometry constant δ_{2S}.

Concluding this section, it is deemed necessary to dwell on the fact that obviously we desire to deal with matrices for which the RIP condition holds true with as high value of S as possible. Unfortunately, there are no known large matrices with bounded restricted isometry constants, and -as with the computation of the spark of a matrix- computing these constants is a strongly NP-hard task ([61]). However, many random matrices have been shown to remain bounded. More precisely, it has been shown that with exponentially high probability, random Gaussian, Bernoulli, and partial Fourier matrices satisfy the RIP condition with number of measurements nearly linear in the sparsity level ([71]). The current smallest upper bounds for any large rectangular matrices are for those of Gaussian matrices according to [2].

Rank Minimization

Inspired by the reasoning developed above, a question of similar nature which is commonly addressed in practice (and in a sense, seems more generic) is whether it is possible to recover instead of an unknown vector $\boldsymbol{x}$ which is assumed to be sparse, an unknown matrix $\boldsymbol{D}$ which is assumed to be low-rank -given the fact that only a fraction of its entries are observed. In other words, as common sense approach dictates, we aim at solving the following optimization problem:

$$\min_{s.t.\boldsymbol{X}_{ij}=\boldsymbol{D}_{ij},(i,j)\in\Omega} rank(\boldsymbol{X}) \quad \textbf{(A.2.21)}$$

, where Ω is the set of observed entries of matrix $\boldsymbol{D} \in R^{m\times n}$, sampled uniformly at random, and $\boldsymbol{X}$ is the decision variable.

Although **(A.2.21)** may seem quite reasonable as an approach for tackling the rank minimization problem, in reality it is of quite limited practical use. The reason for that is on the one hand that is a problem of combinatorial nature -recasting it almost immediately as a NP-hard one- while on the other hand all known algorithmic schemes require time doubly exponential in the dimension of the matrix $\boldsymbol{D}$ (both in theory and practice) in order to provide the desired solution ([22], [66]).

l_*-Norm Minimization

Just as the rank minimization task resembles in flavor with the l_0-norm minimization one, so does the alternative used to overcome it: a handful proxy for the rank functional, which nevertheless can be efficiently optimized. With this guideline into mind, various heuristics have been developed. One of the most famous among them is the trace heuristic ([45]), which can be minimized in place of the rank functional when the matrix of interest is positive semidefinite. The main problem with this heuristic is that it is simply not applicable in situations where the desired matrix is non-symmetric or non-square, as in such cases the trace is not even defined.

To overcome such difficulties, M. Fazel championed a more "universal" heuristic both in [29] and [28], which at the same time constitutes the convexly relaxed counterpart of the rank functional in **(A.2.21)**: the nuclear norm. So replacing the rank with the nuclear norm in **(A.2.21)**, the optimization task now takes the following form:

$$\min_{s.t.\boldsymbol{X}_{ij}=\boldsymbol{D}_{ij},(i,j)\in\Omega} \|\boldsymbol{X}\|_* \quad \textbf{(A.2.22)}$$

As a result, the emerging optimization task in **(A.2.22)** is a convex optimization problem which can be solved efficiently via SDP -in contrast to its primal combinatorically hard counterpart. Furthermore, it is worth mentioning the fact that in the case where the matrix $\boldsymbol{X}$ is symmetric and positive semidefinite, its singular values coincide with its eigenvalues, and the computation of the nuclear norm of matrix $\boldsymbol{X}$ reduces to the computation of its trace. Of course, such a transition (from the rank functional to the nuclear norm) should be cautious, as the two objective functions are of different "nature": on the one hand, the rank function counts the number of the non-vanishing singular values of a matrix (interpreting it via its SVD), while on the other hand the nuclear norm sums their amplitude. Yet again, convexity plays the key role in this direction -as it becomes clearer below.

Prior Art Applied Once More

Bearing into mind the fact that the l_1-norm minimization task constitutes the convex relaxation of the l_0-norm minimization one, and more precisely that in fact it is the tightest one (convex hull / convex envelope), a similar analogy would also be convenient in our attempt to tackle the NP-hard rank minimization task. Fortunately, such a hope holds true, and the relative formality was given in [29] (as well as its proof) in the form of the following Theorem:

Theorem A.2.4: The convex envelope of the function $f(\boldsymbol{X}) = rank(\boldsymbol{X})$ on $C = \{\boldsymbol{X} \in R^{m\times n}: \|\boldsymbol{X}\| \leq 1\}$ is $u(\boldsymbol{X}) = \|\boldsymbol{X}\|_*$.

The significance of the above Theorem is exactly that of the technique of convex relaxation: the provision of useful information concerning the solution of the original NP-hard optimization problem, by finding that of the relaxed one. In other words, by solving the nuclear norm minimization task, we obtain helpful clues in our quest for the solution of the rank minimization task. In fact, as it is shown in [29], if the feasible set, C, is bounded by M, i.e. $\|\boldsymbol{X}\| \leq M$ for all $\boldsymbol{X} \in C$, (provided we can find such a bound M), then for all $\boldsymbol{X} \in C$ holds true the following inequality:

$$\frac{\|\boldsymbol{X}\|_*}{M} \leq rank(\boldsymbol{X}) \quad \textbf{(A.2.23)}$$

What practically **(A.2.23)** means is that the optimal solutions of the respective minimization tasks also obey a similar inequality equation:

$$\frac{os_*}{M} \leq os_{rank} \quad \textbf{(A.2.24)}$$

, where os_* denotes the optimal solution of the nuclear norm minimization task and os_{rank} denotes the optimal solution of the rank minimization task.

As it is obvious from **(A.2.24)**, the optimal solution of the rank minimization task is bounded below from that of the nuclear norm minimization task. As a result, by finding the solution to the heuristic problem we surely obtain an approximate value of the solution of our original NP-hard combinatorial problem.

Equivalence of Rank-l_*-Norm Minimization Tasks

As it became clear from the above analysis, the adoption of the nuclear norm heuristic for the rank minimization problem generalizes the results derived by the l_1-norm minimization in CS literature, exploiting the apparently strong parallels among them. As a consequence, it is reasonable to expect a similar analogy concerning those sufficient conditions under which the rank minimization and l_*-norm minimization tasks are equivalent and provide us the optimum solution. In fact, this is indeed the case, as it was shown in [53]. The elegant results of this paper build on the concept of restricted isometries introduced in [17] by Candès and Tao, deriving the homonymous property (RIP) under which the l_*-norm minimization task can be guaranteed to produce the minimum rank solution.

Generalizing the definition of the RIP from vectors to matrices, we have the following definition ([53]):

Definition A.2.4: Let $\boldsymbol{A}: R^{m\times n} \rightarrow R^{p}$ be a linear map. Without loss of generality, assume $m \leq n$. For every integer r with $1 \leq r \leq m$, define the r-restricted isometry constant to be the smallest number $\delta_r(\boldsymbol{A})$ such that:

$$\big(1-\delta_r(\boldsymbol{A})\big)\|\boldsymbol{X}\|_F \leq \|\boldsymbol{A}(\boldsymbol{X})\| \leq \big(1+\delta_r(\boldsymbol{A})\big)\|\boldsymbol{X}\|_F \quad \textbf{(A.2.25)}$$

holds for all matrices $\boldsymbol{X}$ of rank at most r.

The similarities between **(A.2.19)** and **(A.2.25)** are more than obvious, as it was expected: the l_2-norm is replaced by the Frobenius norm, and the l_0-norm by the rank functional. In a sense, we could say that **(A.2.25)** extends the concept of **(A.2.19)** in the diagonal case, as in this regime the Frobenius norm is equal to the Euclidean norm of the diagonal.

Nevertheless, there are also some differences among the two definitions, which are not of the same importance. The first one concerns the fact that in **(A.2.25)** the norms involved are not squared as it is the case in **(A.2.19)**, a choice although made by the authors of [53] just for simplification of the analysis following **(A.2.25)** and not for some exceptional (mathematical or not) advantage gained by adopting it. The second and most important difference however is that **(A.2.25)** cannot guarantee that all submatrices of the linear transform $\boldsymbol{A}$ of a certain size to be well conditioned, in the sense that the set of matrices $\boldsymbol{X}$ obeying **(A.2.25)** is not a finite union of subspaces.

Despite the aforementioned differences, and more precisely the second one which has a strong theoretical and conceptual flavor, the recovery results derived using **(A.2.25)** remain analogous to those using **(A.2.19)**. As before, the significance of the RIP becomes perceivable via the derivation of theorems resulting from it ([53]):

Theorem A.2.5: Suppose that $\delta_{2r} < 1$ for some integer $r \geq 1$. Then $\boldsymbol{X}_0$ is the only matrix of rank at most r satisfying $\boldsymbol{A}(\boldsymbol{X}) = \boldsymbol{b}$, where $\boldsymbol{X}_0$ is a matrix of rank r.

The above Theorem is quite important due to the fact that it leads us to derive an analogous of the Theorem A.2.3 in order to identify those conditions under which $\boldsymbol{X}_0$ is the minimum rank solution ($\boldsymbol{X}^*$) of $\boldsymbol{A}(\boldsymbol{X}) = \boldsymbol{b}$ ([53]):

Theorem A.2.6: Suppose that $r \geq 1$ is such that $\delta_{5r} < \frac{1}{10}$. Then $\boldsymbol{X}^* = \boldsymbol{X}_0$.

Technical details concerning the proofs of Theorems A.2.4 and A.2.5 are of course available in [53], but enlisting them here is out of the scope of this Appendix. An important

point although that has to be highlighted is the fact that the recovery condition on δ_{5r} is an absolute constant, which is independent both from the dimensions of the linear map $\boldsymbol{A}$ and from the rank of $\boldsymbol{X}_0$. Finally, concluding this section as well as the whole present Appendix, it would be a significant omission not to mention the fact that there are quite a lot linear transformations, arising from the sampling of widely used random matrices (Gaussian, Bernoulli), which exhibit the RIP property with overwhelming probability (as it is proven in details in [53]), making its' influence of utmost theoretical as well as practical importance at the same time.

REFERENCES

[1] S. Babacan, M. Luessi, R. Molina and A. Katsaggelos, Sparse bayesian methods for low-rank matrix estimation, *IEEE Transactions on Signal Processing*, vol. 60, no.8, pp. 3964-3977, May 2012.

[2] B. Bah and J. Tanner, Improved Bounds on Restricted Isometry Constants for Gaussian Matrices, *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 5, pp. 2882-2898, 2010.

[3] R. G. Baraniuk, M. Davenport, R. DeVore and M. B. Wakin, A simple proof of the restricted isometry property for random matrices, *Constructive Approximation*, vol. 28, pp. 253-263, 2008.

[4] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183-202, 2009.

[5] S. Becker, E. J. Candès and M. Grant, TFOCS: flexible first-order methods for rank minimization, *Low-rank Matrix Optimization Symposium*, *SIAM Conference on Optimization*, 2011.

[6] R. E. Bellman, ed., *Adaptive control processes: a guided tour*, Princeton University Press, 1961.

[7] R. E. Bellman, ed., *Dynamic programming*, Princeton University Press, 1957.

[8] D. P. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods, *Academic Press*, New York / London, 1982.

[9] D. P. Bertsekas, Nonlinear Programming, *Athena Scientific*, 1999.

[10] T. Bouwmans and E. Zahzah, Robust PCA via Principal Component Pursuit: A Review for a Comparative Evaluation in Video Surveillance, *Computer Vision and Image Understanding*, vol. 122, pp. 22-34, May 2014.

[11] A. M. Bruckstein, D. L. Donoho and M. Elad, From sparse solutions of systems of equations to sparse modeling of signals and images, *SIAM Review*, vol. 51, no. 1, pp. 34-81, 2009.

[12] J. Cai, E. J. Candès and Z. Shen, A singular value thresholding algorithm for matrix completion, *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956-1982, January 2010.

[13] J.-F. Cai, S. Osher and Z. Shen, Linearized Bregman iterations for compressed sensing, *Mathematics of Computation*, vol. 78, pp. 1515-1536, 2009.

[14] E. J. Candès, The Restricted Isometry Property and its Implications for Compressed Sensing, *Comptes Rendus Mathematique de l' Academie des Sciences*, Paris, France, vol. 346, no. 9, pp. 589-592, 2008.

[15] E. J. Candès and B. Recht, Exact Matrix Completion via Convex Optimization, *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717-772, 2009.

[16] E. J. Candès and J. Romberg, Practical signal recovery from random projections, *Proceedings of the SPIE 17th Annual Symposium on Electronic Imaging*, Bellingham, WA, 2005.

[17] E. J. Candès and T. Tao, Decoding by linear programming, *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203-4215, 2005.

[18] E. J. Candès and M. B. Wakin, An Introduction To Compressive Sampling, *Signal Processing Magazine*, IEEE, vol. 25, no. 2, pp. 21-30, March 2008.

[19] E. J. Candès, X. Li, Y. Ma and J. Wright, Robust Principal Component Analysis?, *Journal of the Applied and Computational Mathematics*, vol. 58, no. 3, Article no. 11, pp. 1-37, June 2011.

[20] E. J. Candès, J. Romberg and T. Tao, Robust uncertainty principles: Exact signal reconstruction from highly incomplete Fourier information, *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489-509, 2006.

[21] E. J. Candès, J. Romberg and T. Tao, Stable recovery from incomplete and inaccurate measurements, *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207-1223, 2006.

[22] A. L. Chistov and D. Yu. Grigoriev, Complexity of quantifier elimination in the theory of algebraically closed fields, *Proceedings of the 11th Symposium on Mathematical Foundations of Computer Science*, vol. 176 of *Lecture Notes in Computer Science*, pp. 17-31, 1984.

[23] X. Ding, L. He and L. Carin, Bayesian robust principal component analysis, IEEE Transaction on Image Processing, vol. 20, no. 12, pp. 3419-3430, January 2012.

[24] D. L. Donoho and M. Elad, Optimally sparse representation in general (nonorthogonal) dictionaries via l_1 minimization, *Proceedings of National Academy of Sciences*, pp. 2197-2202, 2003.

[25] D. L. Donoho and X. Huo, Uncertainty Principles and Ideal Atomic Decomposition, *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2845-2862, 2001.

[26] D.L. Donoho, For most large underdetermined systems of linear equations, the minimal l_1-norm solution is also the sparsest solution, *Communications on Pure and Applied Mathematics*, vol. 59, pp. 797–829, 2006.

[27] C. Eckart and G. Young, The approximation of one matrix by another of lower rank, *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[28] M. Fazel, "Matrix Rank Minimization with Applications", PhD thesis, Stanford University, 2002.

[29] M. Fazel, H. Hindi, and S. P. Boyd, A rank minimization heuristic with application to minimum order system approximation, *American Control Conference*, *Proceedings of the 2001*, pp. 4734-4739, 2001.

[30] G. H. Golub and C. F. van Loan, Matrix Computation, *The Johns Hopkins University Press*, 1996.

[31] I. F. Gorodnitsky and B. D. Rao, Sparse signal reconstruction from limited data using FOCUSS: A re-weighted minimum norm algorithm, *IEEE Transactions on Signal Processing*, vol. 45, no. 3, pp. 600-614, 1997.
[32] R. Gribonval and M. Nielsen, Sparse decompositions in unions of bases, *IEEE Transactions on Information Theory*, vol. 49, no. 12, pp. 3320-3325, 2003.
[33] C. Guyon, T. Bouwmans and E. Zahzah, Foreground detection via robust low-rank matrix decomposition including spatiotemporal constraint, *International Workshop on Background Model Challenges*, ACCV 2012, November 2012.
[34] C. Guyon, T. Bouwmans and E. Zahzah, Foreground detection via robust low-rank matrix factorization including spatial constraint with iterative reweighted regression, *International Conference on Pattern Recognition*, ICPR 2012, November 2012.
[35] C. Guyon, T. Bouwmans, and E. Zahzah, Moving object detection via robust low-rank matrix decomposition with IRLS scheme, *International Symposium on Visual Computing*, ISVC 2012, pp. 665-674, July 2012.
[36] M. R. Hestenes, Multiplier and gradient methods, *Journal of Optimization Theory and Applications*, vol. 4, pp. 303-320, 1969.
[37] H. Hotelling, Analysis of a complex of statistical variables into principal components, *Journal of Educational Psychology*, vol. 24, pp. 417–441 and 498–520, 1933.
[38] X. Huo, “Sparse Image representation via Combined Transforms”, Ph.D. thesis, Stanford, 1999.
[39] N. Jorge, S. J. Wright, Numerical Optimization, *Springer Series in Operations Research and Financial Engineering*, Berlin/New York, 2006.
[40] K. Karhunen, “Zur spektraltheorie stochastischer prozesse,” *Annales Academiae Scientiarum Fennicae*, vol. 37, 1946.
[41] R. M. Larsen, PROPACK-software for large and sparse SVD calculations, 2004; http://sun.stanford.edu/~rmunk/PROPACK/ [Accessed 29/06/2016].
[42] Z. Lin, M. Chen, L. Wu and Y. Ma, The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices, *UIUC Technical Report*, November 2009.
[43] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen and Y. Ma, Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix, *Computational Advances in Multi-Sensor Adaptive Processing* (*CAMSAP*), July 2009.
[44] S. Mallat and S. Zhang, Matching Pursuit in a time-frequency dictionary, *IEEE Transactions on Signal Processing*, vol. 41, pp. 3397-3415, 1993.
[45] M. Mesbahi and G. Papavassilopoulos, On the rank minimization problem over a positive semidefinite linear matrix inequality, *IEEE Transactions on Automatic Control*, vol. 42, no. 2, pp. 239-243, February 1997.
[46] L. Mirsky, Symmetric gauge functions and unitarily invariant norms, Quarterly Journal of Mathematics, vol. 11, no. 1, pp. 50–59, 1960.
[47] B. K. Natarajan, Sparse approximate solutions to linear systems, *SIAM Journal on Computing*, vol. 24, pp. 227-234, 1995.
[48] Y. Nesterov, A method of solving a convex programming problem with convergence rate $O\left(\frac{1}{k^2}\right)$, *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372-376, 1983.
[49] K. Pearson, On Lines and Planes of Closest Fit to Systems of Points in Space, *Philosophical Magazine*, vol. 2, no. 11, pp.559-572, 1901.
[50] Perception and Decision Lab, University of Illinois, Low-Rank Matrix Recovery and Completion via Convex Optimization, August 2012; http://perception.csl.illinois.edu/matrix-rank/home.html [Accessed 19/07/2016].
[51] A. M. Pinkus, On l_1-approximation, *Cambridge Tracts in Mathematics*, vol. 93, Cambridge University Press, 1989.
[52] M. J. D. Powell, edited by R. Fletcher, A method for nonlinear constraints in minimization problems, *Optimization*, Academic Press, New York, pp. 283-298, 1969.
[53] B. Recht, M. Fazel, and P. Parrilo, Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization, *SIAM Review*, vol. 52, no.3, pp. 471-501, 2010.
[54] J. Romberg, Imaging via Compressive Sampling, *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp.14-20, March 2008.
[55] J. Shlens, *A Tutorial on Principal Component Analysis*, Google Research, Version 3.02, April 2014.
[56] G. Strang, ed., *Introduction to Linear Algebra*, 4th edition, Wellesley-Cambridge Press, 2009.
[57] T. Strohmer and R. W. Heath, Grassmannian frames with applications to coding and communication, Applied and Computational Harmonic Analysis, vol. 14, pp. 257–275, 2003.
[58] G. Tang and A. Nehorai, Robust principal component analysis based on low-rank and block-sparse matrix decomposition, *Annual Conference on Information Sciences and Systems*, CISS 2011, 2011.
[59] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 4th edition, Elsevier, 2009.

[60] S. Theodoridis, A. Pikrakis, K. Koutroumbas and D. Cavouras, *An Introduction to Pattern Recognition-A Matlab Approach*, Elsevier, 2010.
[61] A. M. Tillmann and M. E. Pfetsch, The Computational Complexity of the Restricted Isometry Property, the Nullspace Property, and Related Concepts in Compressed Sensing, *IEEE Transactions on Information Theory*, vol. 60, no. 2, pp. 1248–1259, 2013.
[62] K. C. Toh, M. J. Todd and R.H. Tütüncü, A Matlab software package for semidefinite-quadratic-linear programming, version 3.0, August 2001; http://www.math.nus.edu.sg/~mattohkc/sdpt3.html [Accessed 29/06/2016].
[63] K. C. Toh and S. Yun, An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems, *Pacific Journal of Optimization*, vol. 6, no. 3, pp. 615-640, 2009.
[64] J. A. Tropp and A. A. Gilbert, Signal recovery from random measurements via orthogonal matching pursuit, *IEEE Transactions on Information Theory*, vol. 53, pp. 4655–4666, 2007.
[65] P. Tseng, On accelerated proximal gradient methods for convex-concave optimization, *Technical report*, University of Washington, Seattle, 2008.
[66] L. Vandenberghe and S. Boyd, Semidefinite programming, *SIAM Review*, vol. 38, no. 1, pp. 49-95, March 1996.
[67] L. R. Welch, Lower bounds on the maximum cross correlation of signals, *IEEE Transactions on Information Theory*, vol. 20, no. 3, pp. 397-399, 1974.
[68] B. Wohlberg, R. Chartrand and J. Theiler, Local principal component analysis for nonlinear datasets, *International Conference on Acoustics, Speech, and Signal Processing*, ICASSP 2012, pp. 3925-3928, March 2012.
[69] J. Wright, Y. Peng, Y. Ma, A. Ganesh and S. Rao. Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization, *Advances on Neural Information Processing Systems*, NIPS 2009, pp. 2080-2088, December 2009.
[70] H. Xu, C. Caramanis and S. Sanghavi, Robust PCA via outlier pursuit, *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3047-3064, 2012.
[71] F. Yang, S. Wang and C. Deng, Compressive Sensing of Image Reconstruction Using Multi-wavelet Transforms, *IEEE International Conference on Intelligent Computing and Intelligent Systems*, vol. 1, pp. 702–705, 2010.
[72] X. Yuan and J. Yang, Sparse and low-rank matrix decomposition via alternating direction methods, *Pacific Journal of Optimization*, vol. 9, no. 1, January 2009.
[73] T. Zhou and D. Tao, GoDec: randomized low-rank and sparse matrix decomposition in noisy case, *International Conference on Machine Learning*, ICML 2011, 2011.
[74] Z. Zhou, X. Li, J. Wright, E. J. Candès, Y. Ma. Stable principal component pursuit, *IEEE ISIT Proceedings*, pp. 1518-1522, June 2010.