COMPARATIVE ANALYSIS OF ORTHOGONAL MATCHING PURSUIT AND LEAST ANGLE REGRESSION

By

Mazin Abdulrasool Hameed

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Electrical Engineering

2012

ABSTRACT

COMPARATIVE ANALYSIS OF ORTHOGONAL MATCHING PURSUIT AND LEAST ANGLE REGRESSION

By

Mazin Abdulrasool Hameed

The problem of finding a unique and sparse solution for an underdetermined linear system has attracted significant attention in recent years. In this thesis, we compare two popular algorithms that are used for finding sparse solutions of underdetermined linear systems: Orthogonal Matching Pursuit (OMP) and Least Angle Regression (LARS). We provide an in-depth description of both algorithms. Subsequently, we outline the similarities and differences between them. OMP and LARS solve different optimization problems: OMP attempts to find an approximate solution for the ℓ_0-norm minimization problem, while LARS solves the ℓ_1-norm minimization problem. However, both algorithms depend on an underlying greedy framework. They start from an all-zero solution, and then iteratively construct a sparse solution until some convergence is reached. By reformulating the overall structure and corresponding analytical expressions of OMP and LARS, we show that many of the steps of both algorithms are almost identical. Meanwhile, we highlight the primary differences between the two algorithms. In particular, based on our reformulation, we show that the key difference between these algorithms is how they update the solution vector at each iteration. In addition, we present some of the salient benefits and shortcomings of each algorithm. Moreover, we exploit parallel processing techniques to speed-up the convergence of the algorithms.

ACKNOWLEDGMENTS

First of all, I would like to express my deep gratitude to my advisor Professor Hayder Radha for his guidance, inspiration and encouragement. Without his invaluable advice and support, this work would not be possible.

In addition, I would like to thank the government of my country (Iraq) represented by ministry of higher education and scientific research for granting me the scholarship to study in the United State and supporting me financially during my study in Michigan State University.

Furthermore, I would like to thank my colleague in WAVES lab Mohammad Aghagolzadeh for helping me a lot in this work. Also, I am thankful to my friends Wahhab Albazrqaoe and Mohammed Alqizwini. Their support and encouragement are appreciated.

Finally, a special thanks to my parents and my wife for their encouragement and support throughout my study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CS	Compressed Sensing
DCT	Discrete Cosine Transform
LARS	Least Angle Regression
LASSO	Least Absolute Shrinkage and Selection Operator
MP	Matching Pursuit
MSE	Mean Square Error
OMP	Orthogonal Matching Pursuit
PSNR	Peak Signal to Noise Ratio
StOMP	Stagewise Orthogonal Matching Pursuit

NOTATIONS

x	Solution vector which is usually sparse
y	Observation or measurement vector
n	Size of the solution vector x
m	Size of the measurement vector y
k	Sparsity (number of nonzero entries of the solution vector x)
A	Projection matrix of size $m \times n$
a_i	i^{th} column of the matrix A
I	Active set
I^c	Inactive set
t	Iteration counter
r_t	Residual vector at iteration t
c_t	Correlation vector at iteration t
x_t	Current solution vector at iteration t
Δx_t	Solution update vector at iteration t (i.e. $\Delta x_t = x_t - x_{t-1}$)
d_t	Updated direction of LARS at iteration t
γ_t	Step size of LARS at iteration t
λ_t	The largest absolute entry in the correlation vector at iteration t, and represents the absolute correlation of the active columns in LARS

Chapter 1
Introduction

1.1 Background

Various applications in science and nature can be modeled as an underdetermined linear system which is characterized by having fewer equations than unknowns. Therefore, finding a solution for an underdetermined linear system is a central issue for a wide range of areas and applications. Some of these areas include: Compressed Sensing (CS) [1,2,3,4,5], error correction [6,7], minimum distance problems in coding theory [8], and a wide range of inverse problems [9]. In this thesis, we focus on presenting the problem of finding a solution for an underdetermined linear system in the context of the area of compressed sensing. In that context, instead of sensing a signal using a high sampling rate (e.g. the Nyquist rate), CS senses a compressive representation of that signal using a number of measurements that is smaller than the signal dimension. As a result, CS generates fewer measurements than traditional signal sampling methods that have been based on the Nyquist criterion for decades. Consequently, the CS framework leads to solving the following underdetermined linear system:

$$y = Ax \tag{1.1}$$

Here, $y \in R^m$ is the measurement vector (compressive samples), A is a known $m \times n$ projection matrix with $m < n$, and $x \in R^n$ is an unknown vector that represents the signal that we need to find.

In the system of Equation (1.1), the number of unknowns is more than the number of equations. Therefore, there are either infinitely many solutions, or no solution if the vector y is not in the space spanned by the columns of the matrix A. To avoid the case of having no solution, we assume that the matrix A is a full rank matrix, and its columns span the whole space R^m . In practice, and for virtually all real applications, a unique solution for this system is required. It has been well-established under the area of compressed sensing and related literature that if the signal x has a sparse representation (includes only a few nonzero entries) in some space (transform domain), it can be recovered uniquely [2,4,10,1,11,3,12].

To find the sparse solution for the underdetermined linear system, the following optimization problem should be solved:

$$\min_x \|x\|_0 \qquad subject \;\; to \quad Ax = y \tag{1.2}$$

where $\|x\|_0$ is ℓ_0-norm, which represents the number of nonzero entries in a vector x (i.e. $\|x\|_0 = \{\# \, of \; i : \; x(i) \neq 0\}$. When $\|x\|_0 \ll n$, x is consider to be sparse.

Solving the optimization problem characterized by Equation (1.2) yields to a sparse solution that represents the given vector y as a linear combination of the fewest columns of the matrix A. Suppose that the vector x has only k nonzero entries with $k < m < n$. To find such k nonzero entries of the vector x, k columns of the matrix A that best approximate the vector y should be identified. For example in Figure 1.1, when shaded columns are recognized, nonzero shaded coefficients of the vector x can be computed.

The ℓ_0-norm minimization problem expressed by Equation (1.2) is an NP-hard problem [13]. To obtain a global solution, we need to examine the feasibility of all 2^n sparsity patterns of the vector x. Hence, the computation complexity of solving problem (1.2) is $O(2^n)$, which is not practical when n is very large. As a result, some efforts in this area tend to find an approximate solution with tractable complexity by tolerating some error:

$$\min_{x} \|x\|_0 \qquad subject\ to \quad \|Ax - y\|_2 < \epsilon \tag{1.3}$$

where $\epsilon > 0$ is a small constant.

The problem characterized by Equation (1.3) is commonly known as "*sparse approximation*" [13].

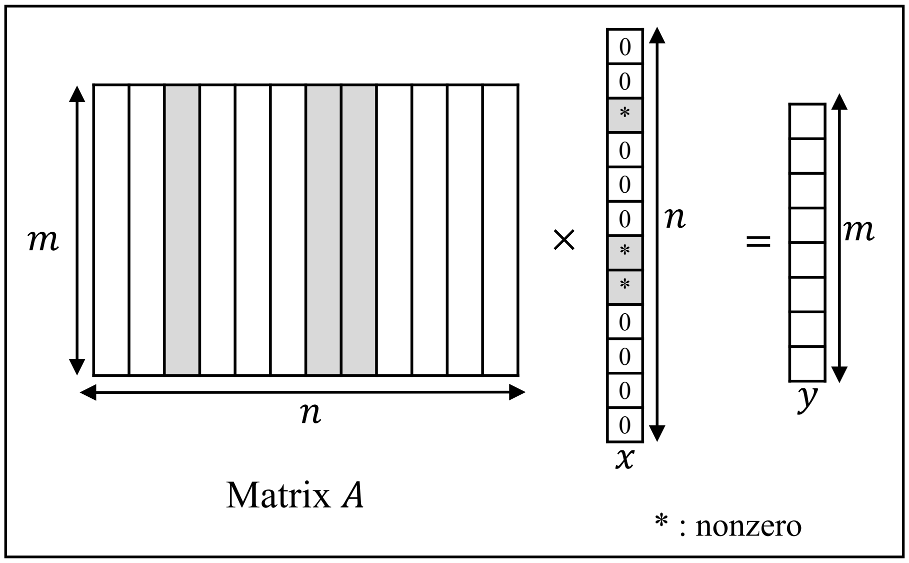


Figure 1.1: A sparse solution of an underdetermined linear system

Because ℓ_0-norm is a nonconvex function, one approach for finding a suboptimal solution of the problem (1.2) is relaxing the ℓ_0-norm by an ℓ_1-norm which is convex function [14,15,16]:

$$\min_{x} \|x\|_1 \qquad subject\ \ to \quad Ax = y \tag{1.4}$$

In the signal processing literature, the ℓ_1-norm minimization problem (1.4) is known as "*Basis Pursuit*" [14]. The solution that is obtained by solving problem (1.4) is equivalent to the solution of ℓ_0-norm minimization problem (1.2) under some conditions [17,18,19,20,21]. The ℓ_1-norm minimization (1.4) is a convex optimization problem, and therefore, it can be solved globally and efficiently by adopting some convex techniques such as the interior point method [22].

Another general direction for finding a sparse solution of the underdetermined linear system (1.1) is to use greedy algorithms [23]. There has been a wide range of greedy algorithms proposed in the literature for finding sparse solutions for the CS problem. One popular family of greedy algorithms, which received a great deal of attention, is the family of greedy pursuit algorithms. Some of these algorithms include: Matching Pursuit (MP) [24], Orthogonal Matching Pursuit (OMP) [25,26,27], Stagewise Orthogonal Matching Pursuit (StOMP) [28], Regularized Orthogonal Matching Pursuit (ROMP) [29], Orthogonal Complementary Matching pursuit (OCMP) [30], Gradient Pursuit (GP) [31], and Compressive Sampling Matching pursuit (CoSaMP) [32]. Another popular framework, which can be considered greedy in nature, for solving the underdetermined system (1.1) is Least Angle Regression (LARS) [33].

Before elaborating further on both of these families of algorithms, namely matching pursuit and least angle regression, we briefly highlight the overall strategy used by these and other greedy algorithms. First, it is worth emphasizing that one of the primary motivations for pursuing greedy algorithms is that they are faster than convex techniques. However, greedy algorithms do not always yield as an accurate solution as convex techniques do. Being iterative, a greedy algorithm usually makes the choice that appears the best at that particular iteration of the algorithm. In essence, a greedy algorithm makes a locally optimal choice at each iteration with the hope that this choice will lead to a globally optimal solution. In other words, these algorithms iteratively make one greedy choice after another until the final solution is reached. Thus, the choice made by a greedy algorithm may depend on past choices, but not on future choices. Equally important, greedy algorithms, in general, never change their pervious choices that are made so far.

To find a solution for the optimization problem (1.2), greedy algorithms typically begin from an all-zero solution, and then iteratively build a sparse solution by selecting a set of columns from the matrix A that best approximates the vector y. Furthermore, such algorithms usually update the solution x at each iteration in a way that depends on the selected columns up to that point. This update leaves a *residual* signal, which represents the part of the measurement vector y that have not been used to recover the solution vector (up to that point); and hence, this residual will be used at the next iteration. These steps are repeated until the current residual falls below a certain very small value.

1.2 Contributions

In this thesis, we focus on analyzing and comparing the two, arguably most popular greedy algorithms that are used for finding a sparse solution of an underdetermined linear system: Orthogonal Matching Pursuit (OMP) and Least Angle Regression (LARS). OMP approximates the ℓ_0-norm minimization problem (1.2), while LARS solves the ℓ_1-norm minimization problem (1.4). Although they solve different optimization problems, we show that both OMP and LARS have almost identical steps after reformulating the underlying analytical expressions of these algorithms. The significant difference between them is how they update the solution vector. Even then, we illustrate some of the subtle similarities and differences between their update strategies; and try to shed some light on their strengths and weaknesses in that respect.

More specifically, our key contributions include:

- Providing a comparative analysis of OMP and LARS in terms of how they find a sparse solution of an underdetermined linear system. Comparing such popular and widely used algorithms is important to their understanding that could lead to better and more efficient implementations.

- Providing a thorough insight and detailed derivation of computing the *updated direction* vector and *step size* parameter that are used in LARS. Prior work, including the original LARS paper [33] did not provide the same level and detailed analysis that is presented in this thesis.
- Reformulating some algorithmic steps of OMP and LARS to discover similarities and differences between them.
- Presenting some of the benefits and shortcomings of OMP and LARS.
- Simulating the comparative analysis of OMP and LARS through different examples covering a wide range of cases.

1.3 Thesis Organization

The rest of the thesis is organized as follows:

- In chapter 2, we review three greedy pursuit algorithms that are used for finding a sparse solution of an underdetermined linear system, and concentrate on explaining the OMP algorithm. We start from the basic MP algorithm, which is considered the ancestor of OMP. Then, we describe OMP in detail. We end the chapter by explaining StOMP, which is a modified version of OMP.
- In chapter 3, we review how LARS can be used for finding a sparse solution of an underdetermined linear system [34]. First, we describe the basic LARS algorithm [33] and the derivations of its main steps. Then, we explain how LARS can be modified to solve the well-

known *Least Absolute Shrinkage and Selection Operator* (LASSO) problem [35], which can be defined as a constrained version of ordinary least square for solving the ℓ_1-norm minimization problem.

- In chapter 4, we compare OMP and LARS in terms of their algorithmic steps and performance. We reformulate some steps of OMP and LARS to discover the similarities and differences between them. To study the performance of OMP and LARS, we compare the convergence time and solution accuracy in each algorithm.
- In chapter 5, we simulate the comparative analysis of OMP and LARS using MATLAB. In general, we go through different examples to demonstrate the similarities and differences between OMP and LARS from various perspectives. First, we employ OMP and LARS to reconstruct some images from their compressive samples. Afterward, we exploit parallel processing techniques to speed-up convergence of the algorithms, and observe the efficiency of using different number of processors. Next, we examine the performance of OMP and LARS in terms of mean square error (MSE) as a function of the measurement size m and the sparsity k. To study the difference in updating process of OMP and LARS, we examine and plot the coefficients of the solution and *correlation* vectors over iterations. Note that the correlation vector represents the correlations of columns of the matrix A with current residual vector r. At last, we generate two different examples, and utilize OMP and LARS to reconstruct the sparse vector in each example. At each iteration of the algorithms, we plot the updating process in three dimensions (3D) view.
- Finally, in chapter 6, we state the conclusion of this thesis and some potential future work.

Chapter 2

Greedy Pursuit Algorithms

Greedy pursuit algorithms attempt to find an approximate solution for the ℓ_0-norm minimization problem (1.2). In this chapter, we review three greedy pursuit algorithms that are used for finding a sparse solution of an underdetermined linear system, and concentrate on explaining the OMP algorithm. First, we explain the basic MP algorithm which is considered the ancestor of OMP. Then, we describe OMP in detail. We end this chapter by explaining StOMP, which is a modified version of OMP.

2.1 Matching Pursuit (MP)

Matching Pursuit (MP) [24] is a basic iterative greedy algorithm that searches for a sparse solution of an underdetermined linear system. The fundamental vector in MP is the *residual vector* $r \in R^m$, which represents the remaining part of the measurement vector y after updating the solution vector. Henceforth, we assume that the columns of the matrix A are normalized to have unit ℓ_2-norm (i.e. $\|a_i\|_2 = 1, i = \{1,2, \ldots., n\}$, where a_i is the i^{th} column of the matrix A). MP starts from an all zero solution, and initializes the residual with the measurement vector y (i.e. $r_0 = y$). Then at each iteration t, MP computes the correlation vector by multiplying columns of the matrix A with the residual vector r from the previous iteration as follows:

$$c_t = A^T r_{t-1} \tag{2.1}$$

Next, it selects a column from the matrix A that is highly correlated with the current residual r. Indeed, the MP algorithm selects a column from the matrix A that corresponds to the entry in the vector c_t that has the largest magnitude (i.e. the largest entry in the vector c_t is determined by considering the absolute value of each entry in the vector c_t). This can be expressed by the following equation:

$$i = \arg \max_{1 \leq j \leq n} |c_t(j)| \tag{2.2}$$

where i is the index of the selected column.

Afterward, the i^{th} solution coefficient $x(i)$ associated with the selected column is updated:

$$x_t(i) = x_{t-1}(i) + c_t(i) \tag{2.3}$$

The last step of MP is computing the current approximation to the vector y via multiplying the matrix A by the current solution vector x_t, and finding a new residual vector by subtracting the vector y from the current approximation:

$$r_t = y - Ax_t \tag{2.4}$$

The residual vector r_t will be used in the subsequent iteration $(t + 1)$ to calculate a new correlation vector c_{t+1}, and select a new column. These steps are repeated until the norm of the residual r_t falls below a certain very small value (ϵ). The MP algorithm can be summarized in Figure 2.1.

- **Input**: the vector $y \in R^m$, the matrix $A \in R^{m \times n}$, and the termination threshold for the residual norm ϵ.
- **Output**: the sparse vector $x \in R^n$.
- **Task**: approximate the vector y by using the fewest columns of the matrix A.

---

1. Initialization: $x_0 = 0, r_0 = y, t = 1$
2. Compute the correlation vector: $c_t = A^T r_{t-1}$
3. Find a column index of the matrix A that is best correlated with the current residual vector. This can be achieved by determining the index of the largest absolute entry in the vector c_t:
$$i = \arg \max_{1 \le j \le n} |c_t(j)|$$
4. Update the ith solution coefficient: $x_t(i) = x_{t-1}(i) + c_t(i)$
5. Compute the new residual vector : $r_t = y - Ax_t$
6. If $\|r_t\|_2 < \epsilon$, terminate and return $x = x_t$ as the final solution. Else, increase the iteration counter: $t = t + 1$ and return to step 2

Figure 2.1: The MP algorithm

The MP algorithm is simple and intuitive for finding a sparse solution of an underdetermined linear system. However, it suffers from slow convergence because it requires a large unbounded number of iterations to find a solution. The computational complexity of the MP algorithm is $O(mnT)$, where T is the number of iterations that are required for MP to coverage to the final solution.

2.2 Orthogonal Matching Pursuit (OMP)

In many applications, MP is not practical since its complexity increases linearly with the number of required iterations (T). Therefore, MP was revised to limit the number of required iterations by adding an orthogonalization step. The modified MP is known as *Orthogonal Matching Pursuit* (OMP) [25,26,27]. OMP inherits many steps from MP. The primary modification that is made by OMP is using a least square formula to achieve better approximation to the measurement vector y over the selected columns. OMP keeps indices of the selected columns in a set called the *active set* I. Therefore, the selected columns are called the *active columns*.

Like MP, OMP starts from an all-zero solution, and initializes the residual to the measurement vector y. At each iteration, OMP selects a column from the matrix A that is best correlated with the residual vector r. Then, OMP appends the index of the selected column to the active set. The next step is to find the *active entries* (the entries that correspond to the active set I) of the solution vector x by solving the following least square problem over the active columns:

$$x_t(I) = \arg\min_{v} \|y - A_I v\|_2^2 \tag{2.5}$$

Here, A_I is a sub-matrix that is formed by the active columns, and $x(I)$ is a sub-vector which holds the active entries of the vector x.

By using well-known linear algebra techniques, problem (2.5) can be solved by projecting the vector y onto the space spanned by the active columns. This can be achieved by solving for $x_t(I)$ in the following equation:

$$A_I^T A_I x_t(I) = A_I^T y \tag{2.6}$$

Note that Equation (2.6), which is widely known as the *normal equation*, represents a linear system with a unique solution.

Then, and as in the MP algorithm, a new residual vector is computed using Equation (2.4). These steps are repeated until the norm of the current residual falls below a certain very small value ϵ. When OMP terminates, the vector y is spanned by the set of active columns. More importantly, OMP ensures that the residual vector r is orthogonal to all active columns at each iteration. Consequently, the correlations of the active columns will be zeros at the next iteration. Therefore in OMP, no column is selected twice, and the active set grows linearly with the iteration counter. OMP can be summarized in Figure 2.2.

- **Input**: the vector $y \in R^m$, the matrix $A \in R^{m\times n}$, and the termination threshold for the residual norm ϵ.
- **Output**: the sparse vector $x \in R^n$.
- **Task**: approximate the vector y by using the fewest columns of the matrix A.

---

1. Initialization: $x_0 = 0, r_0 = y, t = 1, I = \emptyset$
2. Compute the correlation vector: $c_t = A^T r_{t-1}$
3. Find a column index of the matrix A that is best correlated with current residual vector. This can be achieved by determining the index of the largest absolute entry in the vector c_t:

$$i = \arg \max_{j \in I^c} |c_t(j)|$$

where I^c is the inactive set (the set have indices of columns of the matrix A that are not in the active set

4. Add i to the active set: $I = I \cup \{i\}$
5. Solve the least square problem: $A_I^T A_I x_t(I) = A_I^T y$
6. Compute the new residual vector : $r_t = y - Ax_t$
7. If $\|r_t\|_2 < \epsilon$, terminate and return $x = x_t$ as the final solution. Else, increase the iteration counter: $t = t + 1$ and return to step 2

Figure 2.2: The OMP algorithm

To further illustrate the OMP algorithm, we consider the following example.

Example 2.1: assume that the matrix A contains two columns (a_1, a_2) as illustrated in Figure 2.3. In this case, the solution coefficients $(x(1), x(2))$ that generate the vector y as a linear combination of the columns a_1, a_2 should be found:

$$y = x(1)\, a_1 + x(2) a_2$$

OMP starts by determining which column is highly correlated with the current residual. Recall that the residual vector is initialized to the vector y at the beginning of the first iteration. In this example, the vector y is correlated more with the vector a_1 than the vector a_2. For this reason, OMP selects the vector a_1 during the first iteration. Then, OMP takes the largest possible step in the direction of the vector a_1; this is achieved by projecting the vector y onto the vector a_1 to get the current solution coefficient $x_1(1)$ (red bold line in Figure 2.3). This leaves some error value that is represented by the residual vector r_1. This residual is orthogonal to the vector a_1 which in this example represents the active set. At the second iteration, a new column is selected which represents the best correlated column with the current residual r_1. In this simple illustrative example, the column a_2 which was left out from the first iteration, is selected now and its index is added to the active set I. Next, the vector y is projected onto the space spanned by the columns a_1 and a_2 to obtain the current solution coefficients $x_2(1)$ and $x_2(2)$ (blue bold line in Figure 2.3). Since the vector y belongs to the space spanned by a_1 and a_2, there is no residual signal left after the second iteration (i.e. $r_2 = 0$). In this case, OMP terminates and returns the vector x_2 as the final solution.

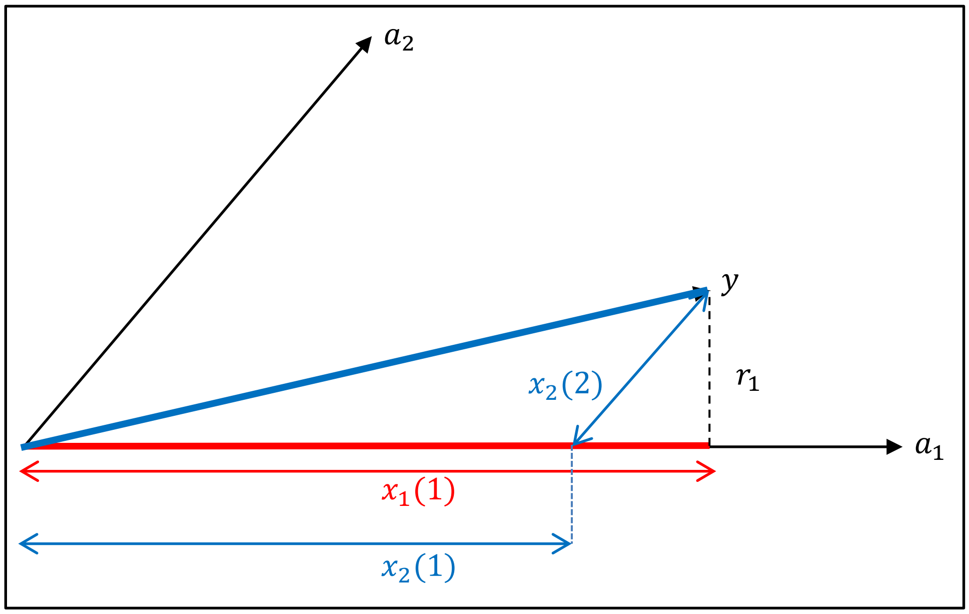


Figure 2.3: OMP approximates the vector y by using the columns a_1, a_2 in Example 2.1
The Red bold line represents the approximation to the vector y at the first iteration; while the blue bold line represents the approximation at the second iteration. For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this thesis.

The orthogonalization step expedites convergence of the OMP algorithms to the final solution. Because of this step, the OMP algorithm is faster than the basic MP algorithm. In OMP, a column is selected and added to the active set only once. This implies that there is no chance for the same column to be selected twice since the residual vector has zero correlation with the active columns, because it is always orthogonal to all active columns after updating the solution coefficients. On the other hand, a column may be selected more than once during MP's computation. Therefore, MP requires more iterations than OMP to converge to the final solution. Nevertheless, each iteration in the OMP algorithm requires a higher computational cost than an MP's iteration. This is because of the orthogonalization step (solving a least square problem) that is used by the OMP algorithm.

In the OMP algorithm, there is a high probability to recover a k sparse signal (includes only k nonzero entries) in at most k iterations using the vector y and the matrix A [36]. As a result, the computational complexity of the OMP algorithm would be $O(mnk)$. Hence, OMP is an efficient algorithm for finding a sparse solution of an underdetermined linear system when the *sparsity* k is relatively small. Note that the sparsity k of any vector refers to the number of nonzero entries in that vector. It is worth noting, in the statistical literature, there is a similar algorithm to OMP that is known as "*Forward Stepwise Regression*" [37].

2.3 Stagewise Orthogonal Matching Pursuit (StOMP)

In many signal processing applications, such as image inpainting [38] and cartoon-texture decomposition [39,40], the underdetermined linear system is extremely large in scale, and the unknown vector (x) is not very sparse (i.e. includes many nonzero entries). In such cases, OMP is not a practical algorithm to find the solution vector x because the computational complexity of OMP increases linearly with the number of nonzeros k. Therefore, OMP has been revised in a way to find less accurate solutions for such large scale underdetermined linear systems in a reasonable time. The modified version of OMP is called *Stagewise Orthogonal Matching Pursuit (*StOMP) [28]. The StOMP is characterized by adding several columns to the active set at each iteration instead of considering a single column as in the case of OMP. StOMP selects columns that their absolute correlations with the current residual exceed a specific chosen threshold value. Thus, the convergence of StOMP is faster than OMP since StOMP requires less number of iterations than OMP to find the solution vector x. The drawback of the StOMP algorithm is the accuracy of the resulting solution. In other words, although StOMP is faster than OMP, the calculated solution by StOMP is less accurate than the calculated one by OMP. It should be clear that StOMP offers a trade-off between the accuracy of the final solution and the required time to find that solution.

The algorithmic steps of StOMP are similar to OMP's steps, except that in OMP, a single column is added to the active set at each iteration, while in StOMP, more than one column can be added to the active set at each iteration. It is important to mention that in the StOMP algorithm, the threshold value (thr) is calculated at each iteration by the following equation:

$$thr = s\sigma \tag{2.7}$$

where $2 \leq s \leq 3$ is the threshold parameter, and $\sigma = \frac{\|r\|_2}{\sqrt{n}}$ is the formal noise level.

Note that the parameter σ is updated at each iteration since it depends on the residual vector r. The StOMP algorithm can be summarized in Figure 2.4.

- **Input**: the vector $y \in R^m$ the matrix $A \in R^{m\times n}$, the termination threshold for the residual norm ϵ, and the threshold parameter s.
- **Output**: the sparse vector $x \in R^n$.
- **Task**: approximate the vector y by using the fewest columns of the matrix A.

```
1. Initialization: $x_0 = 0, r_0 = y, t = 1, I = \emptyset$
2. Compute the correlation vector: $c_t = A^T r_{t-1}$
3. Compute the threshold: $thr_t = s\sigma_t$ where $2 \leq s \leq 3$ and $\sigma = \frac{\|r_{t-1}\|_2}{\sqrt{n}}$.
4. Find indices of columns that their absolute correlations with the residual vector exceed the
   threshold value:
       $i = \{ j: |c_t(j)| > thr_t , j \in I^c\}$
   where $I^c$ is the inactive set (the set have indices of columns of the matrix $A$ that are not in the
   active set
5. Add $i$ to the active set: $I = I \cup \{i\}$
6. Solve the least square problem: $A_I^T A_I x_t(I) = A_I^T y$
7. Compute the new residual : $r_t = y - Ax_t$
8. If $\|r_t\|_2 < \epsilon$, terminate and return $x = x_t$ as the final solution. Else, increase the iteration
   counter: $t = t + 1$ and return to step 2.
```

Figure 2.4: The StOMP algorithm

Chapter 3

Least Angle Regression (LARS)

LARS was originally proposed by Efron, Hastie, Johnstone and Tibshirani as a new model selection algorithm to solve overdetermined linear systems [33]. In addition, Efron et al demonstrated how LARS can be modified to solve the known LASSO problem in a more efficient way than traditional convex optimization techniques. Later, Donoho and Tsaig proposed to use the modified LARS to find a sparse solution of an underdetermined linear system, which is known as "*homotopy algorithm*" [34]. In this chapter, we review and study how LARS can be used for finding a sparse solution of an underdetermined linear system [33,34]. First, we describe the basic LARS algorithm [33]. Then, we provide a thorough insight about the modified LARS for solving the LASSO problem which is considered an effective way for solving the ℓ_1-norm minimization problem (1.4) [34]. In addition, we provide a detailed derivation of LARS main steps, which was not explained in the prior work [33,34] at the same level we present here.

3.1 Basic LARS Algorithm

In Section 2.2, we have seen that the OMP algorithm relies on solving the least square problem to update the solution coefficients at each iteration. As we stated earlier, OMP adopts the largest possible step in the least square direction of the active columns. Therefore, OMP is considered an extreme fitting method that can be overly greedy. OMP may not select some columns that are significant in terms of accuracy of the targeted sparse solution. This is because these columns are highly correlated with columns that have already been in the active set. To overcome this problem, the Least Angle Regression (LARS) was used to find a sparse solution of an underdetermined linear system [34]. LARS is considered less greedy than OMP since it adopts appropriate step that is fairly smaller than the OMP step. In other words, LARS increases the coefficients of the solution vector that associated with the active set as much as needed (as explained further below).

Similar to OMP, LARS starts by initializing: the solution vector x with an all-zero, the residual vector r with the measurement vector y, and the active set I with an empty set. Then, at each iteration, a new column is selected from the matrix A, and its index is added to the active set. At the first iteration, LARS selects a column, say a_j, that is highly correlated with the residual vector r. This implies that the residual vector r has a smaller angle with the column a_j than other columns of the matrix A. Then, LARS increases the coefficient $x(j)$ that is associated with the selected column a_j. This causes the absolute correlation value of a_j with the current residual to decrease as $x(j)$ is increased. LARS takes the smallest possible steps in the direction of the column a_j until another column, say a_k, has as much absolute correlation value with the current residual as the column a_j. Now, instead of continuing in the direction of a_j, LARS moves forward in the direction that is equiangular with the selected columns (a_j, a_k) until a third column, say a_m, has much absolute correlation value with the current residual as much as a_j and a_k. The procedure is continued to add one column until no remaining column has correlation with the current residual. The name "*least angle*" comes from a geometrical interpretation of the LARS process, which chooses the updated direction that makes the smallest and equal angle with all active columns.

To further illustrate the LARS algorithm, we consider the two-dimensional system that was described in Example 2.1. LARS initializes the solution vector to an all zero and the residual to the measurement vector y. As shown in Figure 3.1, LARS begins to select the column a_1

because it has absolute correlation with the initial residual (the vector y) more than a_2 (i.e. $\theta_1(1) < \theta_1(2)$, where $\theta_t(i)$ is the angle between the column a_i and the current residual r at iteration t). Then, the LARS algorithm proceeds in the direction of a_1 by adding the step size γ_1 (the red bold line in Figure 3.1) which is chosen in a way that makes columns a_1 and a_2 have the same absolute correlation with the current residual at the next iteration (i.e. $\theta_2(1) = \theta_2(2)$). When the solution vector x is updated, the solution coefficient $x_1(1) = \gamma_1$. At the second iteration, LARS adds the column a_2 to the active set, and proceeds in the direction that is equiangular with the columns a_1 and a_2. Because there is no remaining column, LARS adds step size γ_2 (the blue bold line in Figure 3.1) that leads to the vector y. LARS terminates since the residual is zero, and the solution coefficients equals to: $x_2(1) = \gamma_1 + \gamma_2 d_2(1)$ and $x_2(2) = \gamma_2 d_2(2)$, where d_2 is the updated direction at the second iteration which is equiangular with the active columns (a_1, a_2).

In the LARS algorithm, two important parameters are required to be computed: the *updated direction* vector d that should be equiangular with the active columns, and the *step size* γ which is a scalar that is multiplied by the updated direction d to update the solution vector x. The remaining steps of the LARS algorithm are roughly similar to the corresponding OMP's steps that were described in Section 2.2.

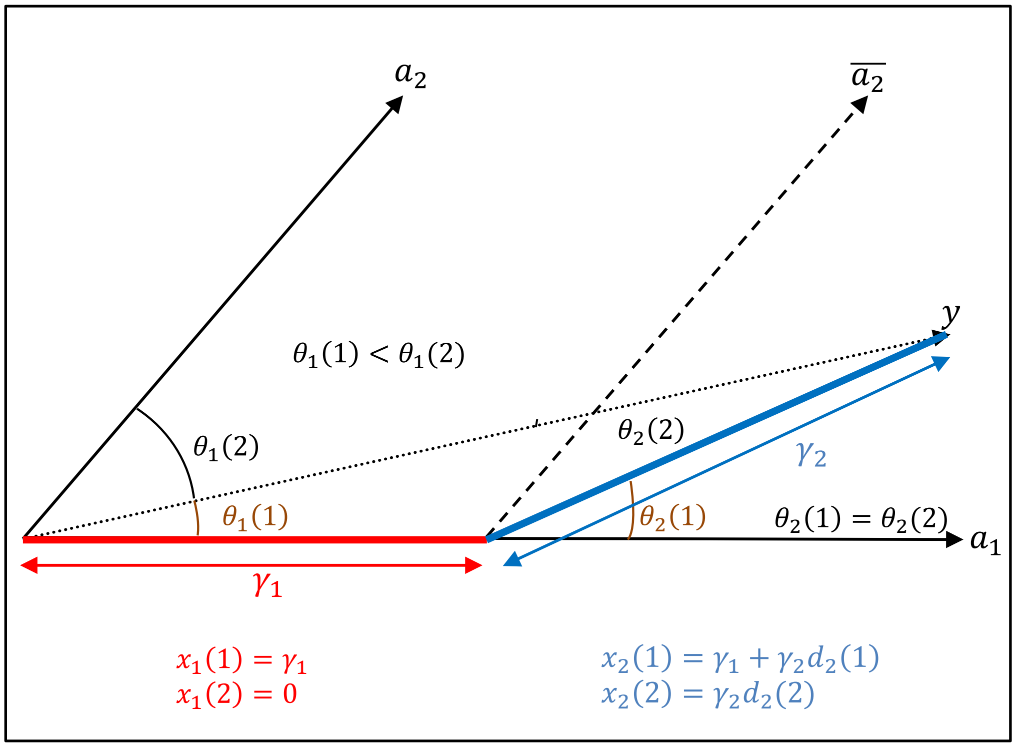


Figure 3.1: LARS approximates the vector y by using the columns a_1, a_2
The red bold line represents the approximation to the vector y at the first iteration, and the blue bold line represents the change that adds to the previous approximation to obtain the approximation at the second iteration. $a_2 \parallel \overline{a_2}$. $\theta_t(i)$ is the angle between the column a_i and the current residual at iteration t. d_2 is the updated direction at the second iteration.

3.3.1 Derivation of Updated Direction

In this section, we provide our derivation of computing the updated direction that is used in [34]. The updated direction should form an equal angle with the active columns toward the residual vector r. We start the derivation of the update direction $d_t \in R^n$ at iteration t by projecting the previous residual vector r_{t-1} onto the space spanned by the active columns. This is achieved by solving for $\tilde{x}$ in the following normal equation:

$$A_I^T A_I \tilde{x} = A_I^T r_{t-1} \tag{3.1}$$

By using Equation (2.1), the active entries of the correlation vector can be found as follows:

$$c_t(I) = A_I^T r_{t-1} \tag{3.2}$$

Substituting (3.2) into (3.1):

$$A_I^T A_I \tilde{x} = c_t(I) \tag{3.3}$$

At each iteration of the LARS algorithm, the active columns have the same absolute correlation with the residual vector. This correlation value is represented by the parameter λ_t. Consequently, we can factorize $c_t(I)$ as follows:

$$c_t(I) = \lambda_t \; sign\big(c_t(I)\big) \tag{3.4}$$

By using (3.4), Equation (3.3) can be rewritten as:

$$A_I^T A_I \tilde{x} = \lambda_t \; sign \; (c_t(I)) \tag{3.5}$$

λ_t is a scalar; dividing Equation (3.5) by the value of λ_t leads to:

$$A_I^T A_I d_t(I) = \; sign \; (c_t(I)) \tag{3.6}$$

where $d_t(I) = \tilde{x}/\lambda_t$ is a sub-vector of the updated direction vector d_t that holds the entries corresponding to the active set I. The goal of Equation (3.6) is to calculate the entries of the sub-vector $d_t(I)$ which ensures that the absolute correlations of the active columns are declined equally. LARS sets entries that are not in the active set to zero (i.e. $d_t(I^c) = 0$).

3.1.2 Derivation of Step Size

At each iteration, the LARS algorithm computes the step size γ to update the solution coefficients that correspond to the active set. This kind of update is necessary to cause a column from the inactive set to be included in the active set at the next iteration. This can be achieved by making the active columns similar to the selected column in terms of the absolute correlation with the current residual vector.

Derivation of closed-form expression for calculating the step size γ was briefly stated in the original LARS paper [33]. Therefore, In this section, we obviously provide our complete and detailed derivation of that expression. We start the derivation of the step size γ by assuming the LARS algorithm is currently at iteration t. In this case, the step size γ_t is required to be calculated. In LARS, the current residual vector is computed in the same way as OMP, which is expressed in Equation (2.4):

$$r_t = y - Ax_t \tag{3.7}$$

And the previous residual is obtained by:

$$r_{t-1} = y - Ax_{t-1} \tag{3.8}$$

To find the relationship between the current and the previous residual vectors, we subtract Equation (3.8) from Equation (3.7):

$$r_t - r_{t-1} = y - Ax_t - y + Ax_{t-1}$$

Equivalently:

$$r_t = r_{t-1} - A(x_t - x_{t-1}) \tag{3.9}$$

At each iteration, LARS updates the solution vector x via adding the multiplication of the scalar γ_t by the vector d_t to the previous solution vector. This is expressed as follows:

$$x_t = x_{t-1} + \gamma_t d_t \tag{3.10}$$

By substituting (3.10) in (3.9), we obtain:

$$r_t = r_{t-1} - \gamma_t A\, d_t \tag{3.11}$$

Because $d(I^c) = 0$, Equation (3.11) can be simplified to:

$$r_t = r_{t-1} - \gamma_t A_I d_t(I) \tag{3.12}$$

The correlation vector c in the next iteration is computed by:

$$c_{t+1} = A^T r_t \tag{3.13}$$

By substituting (3.12) in (3.13), we get:

$$\begin{aligned} c_{t+1} &= A^T(r_{t-1} - \gamma_t A_I d_t(I)) \\ &= c_t - \gamma_t A^T A_I d_t(I) \end{aligned} \tag{3.14}$$

At each iteration of LARS, the parameter λ represents the absolute correlation of the active columns. In fact, λ represents the largest absolute value over entries of the correlation vector. This can be expressed as follows:

$$\lambda = \|c\|_\infty = |c(I)| \tag{3.15}$$

where $\| \quad \|_\infty$ is the maximum norm, and $\|c\|_\infty \overset{\text{def}}{=} \max(|c(1)|, |c(2)|, \dots\dots\dots, |c(n)|)$.

Hence, λ can be computed for the next iteration $t + 1$ as follows:

$$\lambda_{t+1} = |c_{t+1}(I)|$$

By using Equation (3.14), we get:

$$\lambda_{t+1} = |c_t(I) - \gamma_t A_I^T A_I d_t(I)| \tag{3.16}$$

From (3.6), we have:

$$d_t(I) = (A_I^T A_I)^{-1} sign(c_t(I)) \tag{3.17}$$

By substituting $d_t(I)$ of (3.17) into (3.16), we obtain:

$$\lambda_{t+1} = |c_t(I) - \gamma_t A_I^T A_I (A_I^T A_I)^{-1} sign(c_t(I))|$$

$$\lambda_{t+1} = |c_t(I) - \gamma_t \, sign(c_t(I))| \tag{3.18}$$

Substituting (3.4) in (3.18):

$$\lambda_{t+1} = \left|\lambda_t \, sign\big(c_t(I)\big) - \gamma_t \, sign(c_t(I))\right|$$

$$\lambda_{t+1} = \left|sign\left(c_t(I)\right)(\lambda_t - \gamma_t)\right|$$

$$\lambda_{t+1} = |\lambda_t - \gamma_t| \tag{3.19}$$

Recall that at each iteration, the updated direction d ensures the absolute correlations of the active columns are declined equally. In other word, the value of λ is decreased at each iteration. Moreover, LARS selects the smallest step size that causes a column from inactive set to join the active set at next iteration. Therefore, the value of γ_t should be positive and less than λ_t. As a result, Equation (3.19) can be simplified to:

$$\lambda_{t+1} = \lambda_t - \gamma_t \tag{3.20}$$

We assume the i^{th} column is added to the active set at the next iteration $t + 1$. Therefore, the absolute value of its correlation should equal to λ_{t+1}.

$$|c_{t+1}(i)| = \lambda_{t+1} \tag{3.21}$$

where $i \in I_t^c$, and I_t^c is the inactive set at iteration t.

Using Equation (3.14), the correlation of the i^{th} column can be computed by the following equation:

$$c_{t+1}(i) = c_t(i) - \gamma_t a_i^T A_I d_t(I) \tag{3.22}$$

By substituting (3.22) and (3.20) in Equation (3.21), we get:

$$\left|c_t(i) - \gamma_t a_i^T A_I d_t(I)\right| = \lambda_t - \gamma_t$$

$$\pm(c_t(i) - \gamma_t a_i^T A_I d_t(I)) = \lambda_t - \gamma_t$$

$$\pm c_t(i) \mp \gamma_t a_i^T A_I d_t(I) = \lambda_t - \gamma_t$$

$$\gamma_t \mp \gamma_t a_i^T A_I d_t(I) = \lambda_t \mp c_t(i)$$

$$\gamma_t(1 \mp a_i^T A_I d_t(I)) = \lambda_t \mp c_t(i)$$

$$\gamma_t = \frac{\lambda_t \mp c_t(i)}{1 \mp a_i^T A_I d_t(I)}$$

Let $v_t = A_I d_t(I)$

$$\gamma_t = \frac{\lambda_t \mp c_t(i)}{1 \mp a_i^T v_t}$$

The LARS algorithm finds the minimum positive step size γ_t that makes one column from the inactive set I^c join the active set I at the next iteration $t + 1$.

$$\gamma_t = \min_{i \in I^c} \left\{ \frac{\lambda_t - c_t(i)}{1 - a_i^T v_t}, \frac{\lambda_t + c_t(i)}{1 + a_i^T v_t} \right\} \tag{3.23}$$

In (3.23), only positive components within every choice of i are considered when the minimum is taken.

After the updated direction d_t and the step size γ_t are calculated, LARS updates the solution vector as expressed by Equation (3.10), and computes the new residual as expressed by Equation (3.7). These steps are repeated as long as there is still a column from the inactive set that has correlation with the current residual. When no remaining columns have correlation with the current residual (i.e. λ approaches zero), LARS terminates and the vector x_t is returned as the final solution. The basic LARS algorithm can be summarized in Figure 3.2.

- **Input**: the vector $y \in R^m$ the matrix $A \in R^{m \times n}$, and the termination threshold for the residual norm ϵ.
- **Output**: the sparse vector $x \in R^n$.
- **Task**: approximate the vector y by using the fewest columns of the matrix A.

```
1.Initialization: $x_0 = 0, r_0 = y, t = 1, I = \emptyset$.
2.Compute the correlation vector: $c_t = A^T r_{t-1}$
3.Compute the maximum absolute value in the correlation vector: $\lambda_t = \|c_t\|_\infty$
4. If $\lambda_t$ is zero or approaches a very small value, LARS is terminated and the vector $x_t$ is
   returned as the final solution, otherwise the following steps are implemented.
5. Find the active set: $I = \{j: |c_t(j)| = \lambda_t\}$
6. Solve the following least square problem to find active entries of the updated direction:
       $A_I^T A_I d_t(I) = sign(c_t(I))$
   where $sign(c_t(I))$ returns the sign of the active entries of the correlation vector $c_t$
7. Set the inactive entries of the updated direction to zero: $d_t(I^c) = 0$
8. Calculate the step size:
       $\gamma_t = \min_{i \in I^C} \left\{ \frac{\lambda_t - c_t(i)}{1 - a_i^T v_t}, \frac{\lambda_t + c_t(i)}{1 + a_i^T v_t} \right\}$
   where $v_t = A_I d_t(I)$
9. Update the solution vector: $x_t = x_{t-1} + \gamma_t d_t$
10. Compute the new residual vector: $r_t = y - Ax_t$
11. If $\|r_t\|_2 < \epsilon$, terminate and return $x = x_t$ as the final solution. Else, increase the
    iteration counter: $t = t + 1$ and return to step 2.
```

Figure 3.2: The basic LARS algorithm

3.2 Modified LARS for solving the LASSO problem

In certain circumstances, the ℓ_1-norm minimization problem stated in (1.4) can successfully find the sparsest solution while OMP fails [14,27,15]. The ℓ_1-norm minimization problem can be solved by using the standard convex optimization techniques. However, the convex optimization methods require heavy computations especially when the underdetermined system is very large. The Least Absolute Shrinkage and Selection Operator (LASSO) [35] was proposed by Tibshirani to solve the ℓ_1-norm minimization problem (1.4) efficiently. LASSO minimizes the least square error subject to the ℓ_1-norm of the solution vector being less than some threshold (q):

$$\min_x \|y - Ax\|_2^2 \quad subject \ \ to \quad \|x\|_1 \leq q \tag{3.24}$$

In other words, LASSO finds the least square solution subject to an ℓ_1-norm constraint on the solution vector. LASSO (3.24) can be rewritten equivalently as an unconstrained optimization problem:

$$\min_x \frac{1}{2}\|y - Ax\|_2^2 + \lambda\|x\|_1 \tag{3.25}$$

where $\lambda \in [0, \infty)$ is a scalar regularization parameter that handles the trade-off between the mean square error and the ℓ_1-norm of the solution vector x.

To see how Equation (3.25) solves the ℓ_1-norm minimization problem (1.4), the vector x is initialized with an all zero for large λ, and then λ is decreased gradually. When λ approaches zero, the solution of LASSO (3.25) converges to the solution of ℓ_1-norm minimization problem (1.4) [34].

LARS with a minor modification has been used to solve LASSO (3.25) faster than traditional convex optimization methods [33,34]. To explain this, we start solving (3.25) as a minimization problem by setting its gradient to zero and solve the following equation:

$$\partial_x \left(\frac{1}{2}\|y - Ax\|_2^2 + \lambda\|x\|_1\right) = 0 \tag{3.26}$$

This can be simplified to:

$$-A^T(y - Ax) + \lambda\partial_x(\|x\|_1) = 0 \tag{3.27}$$

Because the residual $r = y - Ax$, Equation (3.27) can be rewritten as:

$$-A^T r + \lambda\, \partial_x\, (\|x\|_1\,) = 0 \tag{3.28}$$

Recall that the correlation is calculated by: $c = A^T r$, Equation (3.28) can be expressed as:

$$\begin{gathered} -c + \lambda\, \partial_x(\|x\|_1\,) = 0 \\ c = \lambda\, \partial_x(\|x\|_1\,) \end{gathered} \tag{3.29}$$

The gradient of $\|x\|_1$ cannot be found because ℓ_1-norm is discontinuous at zero. Therefore, subgradient of ℓ_1-norm can be found as follows:

$$\partial_x\|x\|_1 = \left\{ v \in R^n : \begin{array}{ll} v(i) = sign\big(x(i)\big) & if \;\; x(i) \neq 0 \\ v(i) \in [-1,1] & if \;\; x(i) = 0 \end{array} \right\} \tag{3.30}$$

At each iteration of LARS, only coefficients of the solution vector x corresponding to the active set I are nonzeros (i.e. $x(i) \neq 0\, , i \in I$). The other coefficients are zeros (i.e. $x(i) = 0, i \in I^c$). Hence, by using (3.30), Equation (3.29) can be expressed for the active set as follows:

$$c(I) = \lambda\, sign(x(I)) \tag{3.31}$$

and for the inactive set, Equation (3.29) can be expressed as follows:

$$|c(I^c)| \leq \lambda \tag{3.32}$$

Thus, LARS can be used to solve the LASSO problem if it satisfies the constraints stated by Equations (3.31) and (3.32) at each iteration. In other words, to solve LASSO, LARS should ensure:

1- The absolute correlations of the active columns are equal to λ.

2- The signs of the correlation and the solution vectors are matched for the active entries.

3- The absolute correlations of the inactive columns are equal or less than λ.

LARS maintains the condition stated by equality (3.32) at each iteration because the absolute correlations of the inactive columns are always less than λ. If they are equal to λ, the columns should be in the active set (step 5 of the LARS algorithm in Figure 3.2). Also LARS partially maintains the condition stated by Equation (3.31) which is the absolute correlations of the active columns are equal to λ. However, LARS does not enforce the signs matching of the active entries between the correlation and solution vectors. Some of the solution coefficients associated with the active set may change signs while the signs of corresponding correlation coefficients are still the same. Therefore, LARS should be modified to maintain the sign matching constraint that states in the following equation:

$$sign\left(x(I)\right) = sign\left(c(I)\right) \tag{3.33}$$

LARS violates the condition (3.33) when one of the solution coefficients that is associated with the active set crosses zero. Therefore, if any solution coefficient associated with the active set hits zero during the LARS procedures, the corresponded column to this coefficient should be removed from the active set, and the updated direction d is recalculated by solving Equation (3.6). Hence, we need to determine the step size γ which causes one of the solution coefficients that is associated with the active set to be zero after updating solution process. By using Equation (3.10), the i^{th} solution coefficient is updated as follows:

$$x_t(i) = x_{t-1}(i) + \gamma_t d_t(i)$$

The value of $x_t(i)$ would be zero if:

$$\gamma_t = -\frac{x_{t-1}(i)}{d_t(i)}$$

The smallest positive step size that causes the i^{th} solution coefficient associated with the active set to be zero, can be found by the following minimization problem:

$$\gamma_t = \min_{i \in I} \left\{ -\frac{x_{t-1}(i)}{d_t(i)} \right\} \tag{3.34}$$

where the minimum is taken only over positive value.

Therefore, at each iteration, the modified LARS for solving LASSO computes two step sizes: one adds a column to the active set as stated by Equation (3.23), and the other drops a column from the active set as stated by Equation (3.34). To distinguish between these two values of the step size, γ^+ is used to refer to the step size computed by (3.23), and γ^- is used to refer to the step size computed by (3.34). The modified LARS selects the smallest step size:

$$\gamma_t = min\ \{\gamma_t^+, \gamma_t^-\} \tag{3.35}$$

where $\gamma_t^+ = \underset{i \in I^c}{min} \left\{ \frac{\lambda_t - c_t(i)}{1 - a_i^T v_t}, \frac{\lambda_t + c_t(i)}{1 + a_i^T v_t} \right\}$ and $\gamma_t^- = \underset{i \in I}{min} \left\{ -\frac{x_{t-1}(i)}{d_t(i)} \right\}$

The modified LARS uses the step size computed by (3.35) instead of the step size computed by (3.23) in the basic LARS algorithm. Equally important, for updating the active set, if $\gamma_t = \gamma_t^+$, the i^{th} column would be added to the active set at the next iteration $t + 1$ as follows:

$$I = I \cup \{i\} \tag{3.36}$$

In contrast, if $\gamma_t = \gamma_t^-$, the i^{th} column would be removed from the active set at the next iteration $t+1$ as follows:

$$I = I - \{i\} \tag{3.37}$$

The remaining steps are the same as they are in the basic LARS algorithm stated in Section 3.1.

The modified LARS requires more iterations than the basic LARS because in the modified LARS, some columns are added to and dropped from the active set, while in the basic LARS, columns are always added to the active set. The computational complexity of modified LARS is $O(mnT)$, where T is the number of required iterations which is normally equal or greater than the number of nonzero entries of the solution vector x (i.e. $\mathrm{T} \geq k$). Note that the constant that hides in asymptotic notation (big O) of the modified LARS is much higher than the one of OMP, because the modified LARS executes many additional steps to compute the step size γ while OMP does not. However, the modified LARS is considered an efficient algorithm for solving the LASSO problem because it requires less computational cost than traditional convex optimization methods.

Chapter 4

Comparative Analysis of OMP and LARS

In the signal processing and statistical literatures, OMP and modified LARS for solving LASSO are considered popular algorithms for finding a sparse solution of an underdetermined linear system. In this chapter, we compare OMP and modified LARS in terms of the algorithmic steps and performance. We reformulate some steps of OMP and modified LARS to identify the similarities and differences between them. To study the performance of OMP and modified LARS, we compare the convergence time and the solution accuracy in each algorithm.

4.1 Algorithmic Steps

Pervious work by Donoho and Tsaig [34] showed that the basic LARS algorithm that was stated in Section 3.1 and OMP have the same algorithmic steps, except the updating solution step. As stated earlier, OMP updates the active entries of the solution vector x by solving the least square problem. This is achieved by projecting the vector y onto the space spanned by active columns; which is conducted by solving the following normal equation:

$$A_I^T A_I x_t(I) = A_I^T y \tag{4.1}$$

On the other hand, the basic LARS algorithm updates the active entries of the solution vector x by solving the following penalized least square problem:

$$A_I^T A_I \, x_t(I) = A_I^T y - \lambda_{t+1} \, sign\big(c_t(I)\big) \tag{4.2}$$

Equations (4.1) and (4.2) are identical if the penalization term $\{\lambda_{t+1} \, sign\big(c_t(I)\big)\}$ is removed from Equation (4.2). However, Donoho et al [34] did not demonstrate the derivation of (4.2). Therefore, we formulate our derivation of (4.2) in appendix A.

In our work, we compare OMP and modified LARS for solving LASSO from a different perspective. We reformulate some steps in the algorithms to reflect more similarities and differences between these algorithms. For simplicity, hereafter in this chapter and the next ones of this thesis, we use the abbreviation “*LARS*” to refer to the modified LARS algorithm that is specialized for solving the LASSO problem.

OMP and LARS solve different optimization problems: OMP is used to find an approximate solution for the ℓ_0-norm minimization problem stated by Equation (1.2), while LARS is used to solve the ℓ_1-norm minimization problem stated by Equation (1.4). Nevertheless, both OMP and LARS depend on an underlying greedy framework. Indeed, they have almost the same fundamental algorithmic steps. Both algorithms start from an all-zero solution, an empty active set, and the measurement vector y as initial residual. They depend on the correlation between the columns of the matrix A and the current residual to maintain the active set. They compute the residual vector in the same manner. Furthermore, in both algorithms, it is required to solve the least square problem over the active columns to update the solution vector x.

However, OMP and LARS are different in some steps. At each iteration of LARS, a column is added to or removed from the active set, while in OMP, a column is always added to the active set. In addition, each algorithm adopts a different way to update the solution vector x. OMP takes the largest possible step in the least square direction of the active columns. OMP achieves this through updating the active entries of the solution vector x by projecting the vector y onto the space spanned by the active columns. On the other hand, LARS takes the smallest possible step that forces a column from the inactive set to join the active set, or drops a column from the active set. The algorithmic steps of OMP and LARS are compared in Table 4.1.

	OMP	LARS
Initialization	$x_0 = 0, r_0 = y, t = 1, I = \emptyset$	$x_0 = 0, r_0 = y, t = 1, I = \emptyset$
Compute correlation	$c_t = A^T r_{t-1}$	$c_t = A^T r_{t-1}$

Update active set	$i = \arg\max_{i \in I^c} \lvert c_t(j) \rvert$
	$I = I \cup \{i\}$
		$\lambda_t = \lVert c_t \rVert_\infty$
		If ($\gamma_{t-1} = \gamma_{t-1}^{+}$ or $t = 1$) // add column
		$i = \{ j \in I^c : \lvert c_t(j) \rvert = \lambda_t \}$
		$I = I \cup \{i\}$
		If ($\gamma_{t-1} = \gamma_{t-1}^{-}$) // remove column
		$i = \{ j \in I : x_{t-1}(j) = 0\}$
		$I = I - \{i\}$

Update solution	$A_I^T A_I x_t(I) = A_I^T y$ $x_t(I^c) = 0$	$A_I^T A_I d_t(I) = sign(c_t(I))$ $d_t(I^c) = 0$
		$v_t = A_I d_t(I)$ $\gamma_t^{+} = \min_{i \in I^c} \left\{ \frac{\lambda_t - c_t(i)}{1 - a_i^T v_t}, \frac{\lambda_t + c_t(i)}{1 + a_i^T v_t} \right\}$ $\gamma_t^{-} = \min_{i \in I} \left\{ -\frac{x_{t-1}(i)}{d_t(i)} \right\}$ $\gamma_t = min\ \{\gamma_t^{+}, \gamma_t^{-}\}$ $x_t = x_{t-1} + \gamma_t d_t$
Compute residual	$r_t = y - Ax_t$	$r_t = y - Ax_t$
Stopping condition	$\lVert r_t \rVert_2 < \epsilon$	$\lVert r_t \rVert_2 < \epsilon$
Increase iteration counter	$t = t + 1$	$t = t + 1$

Table 4.1: The algorithmic steps of OMP and LARS

From Table 4.1, it is observable that the algorithmic steps are identical in both algorithms, except the steps for updating the active set and the solution vector. To find more commonality between OMP and LARS, we reformulate the steps for updating active set and the solution vector in the OMP algorithm.

4.1.1 Reformulating step for updating the active set

As we stated early on, one of differences between OMP and LARS is the way of updating the active set. To compare them in terms of updating active set step, we reformate the step in the OMP algorithm. At each iteration, OMP selects a column from the inactive set I^c that has the highest absolute correlation with current residual. This can be expressed as follows:

$$i = \arg\max_{i \in I^c} |c_t(j)| \tag{4.3}$$

where i is the index of the selected column.

To make the updating active set step of OMP comparable with the corresponding step of LARS, we divide the step of finding the index i in Equation (4.3) into two steps:

1- Computing the largest absolute entry in the correlation vector: $\lambda_t = \|c_t\|_\infty$.

2- Determining an index of column from the inactive set that its absolute correlation with the current residual vector equals to λ_t. This can be described as follows:

$$i = \{\, j \in I^c \colon |c_t(j)| = \lambda_t \,\} \tag{4.4}$$

Then, i is added to the active set as follows: $I = I \cup \{i\}$.

In Table 4.2, we state the original step for updating the active set and the reformulated one in OMP and LARS. It is noticeable that the step for updating the active set is the same in both algorithms if LARS always adds columns to the active set. Therefore, the basic LARS algorithm that is stated in Section 3.1 is similar to OMP in the way of updating the active set, because it always adds columns to the active set and never removes them from the set. On the other hand, the modified LARS for solving the LASSO problem differs from OMP in the step for updating the active set. The modified LARS adds or removes columns to/from the active set, while OMP always adds columns to the active set.

	OMP	LARS
Update active set (original)	$i = \arg\max_{i \in I^c} \lvert c_t(j) \rvert$ $I = I \cup \{i\}$	$\lambda_t = \lVert c_t \rVert_\infty$ If $(\gamma_{t-1} = \gamma_{t-1}^{+}$ or $t = 1)$ // add column $i = \{\, j \in I^c : \lvert c_t(j) \rvert = \lambda_t \,\}$ $I = I \cup \{i\}$ If $(\gamma_{t-1} = \gamma_{t-1}^{-})$ // remove column $i = \{\, j \in I : x_{t-1}(j) = 0\}$ $I = I - \{i\}$
Update active set (after reformulating)	$\lambda_t = \lVert c_t \rVert_\infty$ add column $i = \{\, j \in I^c : \lvert c_t(j) \rvert = \lambda_t \,\}$ $I = I \cup \{i\}$	$\lambda_t = \lVert c_t \rVert_\infty$ If $(\gamma_{t-1} = \gamma_{t-1}^{+}$ or $t = 1)$ // add column $i = \{\, j \in I^c : \lvert c_t(j) \rvert = \lambda_t \,\}$ $I = I \cup \{i\}$
		If $(\gamma_{t-1} = \gamma_{t-1}^{-})$ // remove column $i = \{\, j \in I : x_{t-1}(j) = 0\}$ $I = I - \{i\}$

Table 4.2: The original step for updating the active set and the reformulated one in OMP and LARS

4.1.2 Reformulating step for updating the solution vector

The other difference between OMP and LARS is the way of updating the solution vector. To compare them clearly, we reformulate the step for updating the solution vector in OMP to make it comparable with corresponding step in LARS. This can be explained as follows: let I be the active set at iteration t, and $\bar{I}$ be the active set at iteration $t-1$. We assume that OMP is currently at iteration t. The active entries of the solution vector x are computed via solving Equation (4.1):

$$A_I^T A_I \, x_t(I) = A_I^T y \tag{4.5}$$

The residual vector at iteration $t-1$ is obtained by:

$$r_{t-1} = y - Ax_{t-1} \tag{4.6}$$

Because only active entries of x_{t-1} are nonzeros, we can rewrite (4.6) as follows:

$$r_{t-1} = y - A_{\bar{I}} \, x_{t-1}(\bar{I})$$

$$y = A_{\bar{I}} \, x_{t-1}(\bar{I}) + r_{t-1} \tag{4.7}$$

By substituting y of (4.7) in (4.5), we obtain:

$$A_I^T A_I \, x_t(I) = A_I^T (A_{\bar{I}} \, x_{t-1}(\bar{I}) + r_{t-1})$$

By simplifying this expression, it follows:

$$A_I^T (A_I \, x_t(I) - A_{\bar{I}} \, x_{t-1}(\bar{I})) = A_I^T \, r_{t-1}$$

Recall Equation (3.2) that computes the active entries of the correlation vector: $c_t(I) = A_I^T r_{t-1}$. This leads to:

$$A_I^T (A_I \, x_t(I) - A_{\bar{I}} \, x_{t-1}(\bar{I})) = c_t(I) \tag{4.8}$$

At each iteration of OMP, one column is added to the active set. Therefore, the active set I at iteration t has one column that is not included in the active set $\bar{I}$ of the previous iteration $(t-1)$. The coefficient of x_{t-1} that is corresponding to this column is zero. Hence, we get:

$$A_{\bar{I}} \, x_{t-1}(\bar{I}) = A_I x_{t-1} \, (I) \tag{4.9}$$

By substituting (4.9) in (4.8), the following equation is obtained:

$$A_I^T (A_I \, x_t(I) - A_I \, x_{t-1}(I)) = c_t(I)$$

Equivalently:

$$A_I^T A_I \left(x_t(I) - x_{t-1}(I)\right) = c_t(I) \tag{4.10}$$

At this point, let us define the vector $\Delta x_t \in R^n$ where $\Delta x_t(I) = x_t(I) - x_{t-1}(I)$ and $\Delta x_t(I^c) = 0$. Therefore, Equation (4.10) can be expressed as follows:

$$A_I^T A_I \, \Delta x_t(I) = c_t(I) \tag{4.11}$$

The active entries of the vector Δx_t can be obtained by solving Equation (4.11), while the remaining entries of the vector are set to zero (i.e. $\Delta x_t(I^c) = 0$). The formula for updating the solution vector x can be rewritten in terms of the vector Δx as follows:

$$x_t = x_{t-1} + \Delta x_t \tag{4.12}$$

As a result, we divide the step for updating the solution vector that is expressed by Equation (4.1) in the OMP algorithm into two steps:

1- Computing active entries of the vector Δx as expressed in Equation (4.11), and setting the remaining entries to zero.

2- Updating the solution vector x as expressed in Equation (4.12).

With respect to LARS, the vector Δx can be calculated by using Equation (3.10) as follows:

$$\Delta x_t = \gamma_t d_t \tag{4.13}$$

The original step for updating the solution vector and the reformulated one are listed for OMP and LARS in Table 4.3.

	OMP	LARS
Update solution (original)	$A_I^T A_I x_t(I) = A_I^T y$ $x_t(I^C) = 0$	$A_I^T A_I d_t(I) = sign(c_t(I))$ $d_t(I^C) = 0$
		$v_t = A_I d_t(I)$ $\gamma_t^+ = \min_{i \in I^C} \left\{ \frac{\lambda_t - c_t(i)}{1 - a_i^T v_t}, \frac{\lambda_t + c_t(i)}{1 + a_i^T v_t} \right\}$ $\gamma_t^- = \min_{i \in I} \left\{ -\frac{x_{t-1}(i)}{d_t(i)} \right\}$ $\gamma_t = min\ \{\gamma_t^+, \gamma_t^-\}$ $x_t = x_{t-1} + \gamma_t d_t$
Update solution (after reformulating)	$A_I^T A_I\ \Delta x_t(I) = c_t(I)$ $\Delta x_t(I^C) = 0$	$A_I^T A_I d_t(I) = sign(c_t(I))$ $d_t(I^C) = 0$
		$v_t = A_I d_t(I)$ $\gamma_t^+ = \min_{i \in I^C} \left\{ \frac{\lambda_t - c_t(i)}{1 - a_i^T v_t}, \frac{\lambda_t + c_t(i)}{1 + a_i^T v_t} \right\}$ $\gamma_t^- = \min_{i \in I} \left\{ -\frac{x_{t-1}(i)}{d_t(i)} \right\}$ $\gamma_t = min\ \{\gamma_t^+, \gamma_t^-\}$ $\Delta x_t = \gamma_t d_t$
	$x_t = x_{t-1} + \Delta x_t$	$x_t = x_{t-1} + \Delta x_t$

Table 4.3: The original step for updating the solution vector and the reformulated one in OMP and LARS

By using the reformulated steps that we derive in Sections 4.1.1 and 4.1.2, the algorithmic steps of OMP and LARS can be rewritten as shown in Table 4.4.

	OMP	LARS
Initialization	$x_0 = 0, r_0 = y, t = 1, I = \emptyset$	$x_0 = 0, r_0 = y, t = 1, I = \emptyset$
Compute correlation	$c_t = A^T r_{t-1}$	$c_t = A^T r_{t-1}$
Update active set	$\lambda_t = \lVert c_t \rVert_\infty$ add column $i = \{ j \in I^c: \lvert c_t(j) \rvert = \lambda_t \}$ $I = I \cup \{i\}$	$\lambda_t = \lVert c_t \rVert_\infty$ If ($\gamma_{t-1} = \gamma_{t-1}^{+}$ or $t = 1$) // add column $i = \{ j \in I^c: \lvert c_t(j) \rvert = \lambda_t \}$ $I = I \cup \{i\}$
		If ($\gamma_{t-1} = \gamma_{t-1}^{-}$) // remove column $i = \{ j \in I: x_{t-1}(j) = 0\}$ $I = I - \{i\}$
Compute solution update vector	$A_I^T A_I \, \Delta x_t(I) = c_t(I)$ $\Delta x_t(I^c) = 0$	$A_I^T A_I d_t(I) = sign(c_t(I))$ $d_t(I^c) = 0$
		$v_t = A_I d_t(I)$ $\gamma_t^{+} = \min_{i \in I^c} \left\{ \frac{\lambda_t - c_t(i)}{1 - a_i^T v_t}, \frac{\lambda_t + c_t(i)}{1 + a_i^T v_t} \right\}$ $\gamma_t^{-} = \min_{i \in I} \left\{ -\frac{x_{t-1}(i)}{d_t(i)} \right\}$ $\gamma_t = min \; \{\gamma_t^{+}, \gamma_t^{-}\}$ $\Delta x_t = \gamma_t d_t$
Update solution	$x_t = x_{t-1} + \Delta x_t$	$x_t = x_{t-1} + \Delta x_t$
Compute residual	$r_t = y - A x_t$	$r_t = y - A x_t$
Stopping condition	$\lVert r_t \rVert_2 < \epsilon$	$\lVert r_t \rVert_2 < \epsilon$
Increase iteration counter	$t = t + 1$	$t = t + 1$

Table 4.4: The algorithmic steps of OMP and LARS after reformulating

Form Table 4.4, it is observable that the algorithmic steps of OMP and LARS are almost identical; the only differences are in updating the active set and computing the solution update vector (the vector Δx). For updating the active set, OMP always adds a column to the active set at each iteration, and this column would never be removed later. On the other hand, in LARS, a column is added to or removed from the active set at each iteration. For computing the vector Δx, both OMP and LARS solve the least square problems stated in Equations (4.11) and (3.6) respectively. However, each algorithm relies on different parameters to achieve this. OMP uses the active entries of the correlation vector c, while LARS uses only the signs of them. Equally important in OMP, the vector Δx is directly computed by solving the least square problem as stated in Equation (4.11). In contrast, in LARS, the vector Δx is computed by the following steps:

1. Determine the updated direction d by solving the least square problem (3.6).
2. Compute the step size γ by Equation (3.35).
3. Multiply γ by d to obtain the vector Δx as express in Equation (4.13).

4.2 Performance Analysis

To analyze the performance of OMP and LARS, we compare these two algorithms in terms of the convergence time and the accuracy of the final solution.

4.2.1 Convergence time

In convergence to the final solution, OMP is much faster than LARS because:

1. Generally speaking, OMP requires less number of iterations than LARS to converge to the final solution. This is because OMP always adds columns to the active set, while LARS adds or removes columns to/from the active set.

2. At each iteration, OMP computes the vector Δx in one step by solving the least square problem (4.11). Whereas in LARS, computing the vector Δx involves many steps as stated in Section 4.1. These steps require more computations than solving the least square problem.

4.2.2 Accuracy

In terms of the recovered sparse solution, OMP is considered less accurate than LARS when some columns of the matrix A are highly correlated. OMP takes the largest possible step in the least square direction of the active columns by solving the least square problem to update the solution vector x. As a result, the residual vector will be orthogonal to all active columns at the next iteration. In other words, the residual vector will have zero correlation with all active columns. Consequently, columns that are highly correlated with the active columns would have small correlation with the current residual. Recall that at each iteration, OMP select a column that has the largest absolute correlation with current residual. Therefore, OMP does not select these columns even though they are significant for recovering the sparse solution. On the other hand, LARS does not this problem because it uses a fairly smaller step than the OMP step, and increases the coefficients of the solution vector associated with the active set as much as needed. For this reason, LARS selects important columns for recovering the sparse solution even though they may be highly correlated with the columns that have already been in the active set .To demonstrate this, we consider the following example:

Example 4.1: assume the matrix A contains three columns (a_1, a_2, a_3), where the columns a_1 and a_2 are highly correlated as shown in Figure 4.1. We generate the vector $x \in R^3$ which its first and second entries are nonzeros, and the third one is zero:

$$x = \begin{bmatrix} 0.9613 \\ 0.2757 \\ 0 \end{bmatrix}$$

The measurement vector y is obtained via multiplying the matrix A by the vector x (i.e. $y = Ax$). As a result, the vector y would be a linear combination of the columns a_1 and a_2.

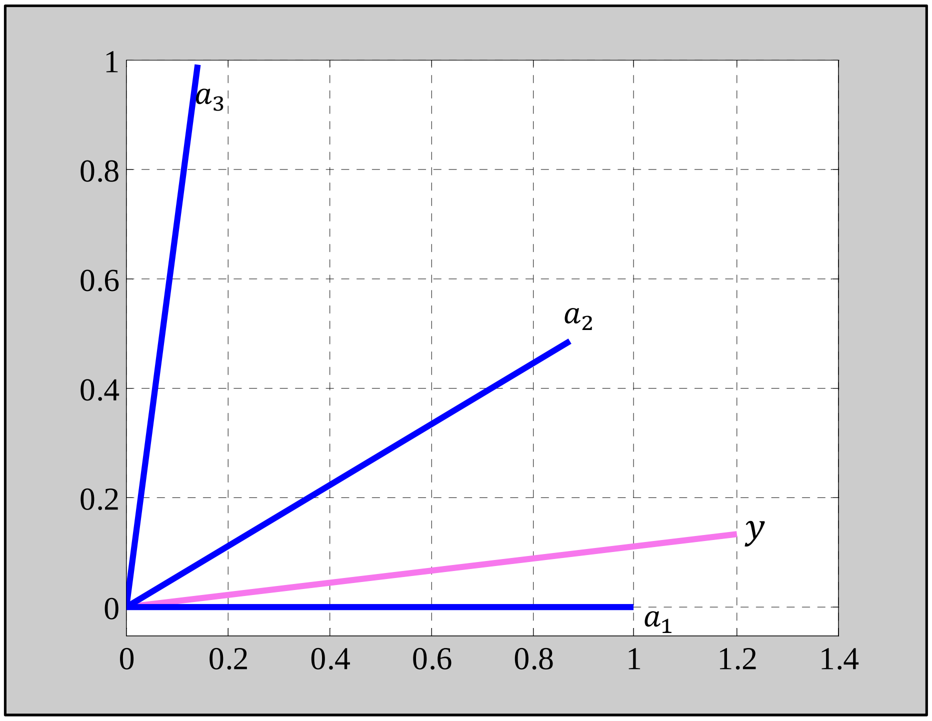


Figure 4.1: Columns of the matrix A and the measurement vector y in Example 4.1

We use both OMP and LARS to find the vector $\hat{x}$ that represents the vector y as a linear combination of columns of the matrix A. In Figure 4.2 and Figure 4.3, we illustrate the *approximation change* to the vector y which is accomplished by OMP and LARS at each iteration. Note that the approximation change is calculated by multiplying the matrix A by the current solution update vector Δx_t.

OMP starts by selecting a column from the matrix A that has maximum absolute correlation with the initial residual (i.e. the vector y). In this example, the column a_1 is selected at the first iteration because it is highly correlated with initial residual as shown in Figure 4.2, and added to the active set. Then, OMP takes the largest possible step in the direction of the column a_1 by projecting the vector y onto the column a_1. This leaves some error represented by the residual vector r which is orthogonal to a_1. At the second iteration, the column a_2 would be less correlated with the residual vector since it is highly correlated with the column a_1. In this case, the column a_3 has the largest absolute correlation with the current residual. Therefore, the column a_3 is selected and added to the active set as shown in Figure 4.3. Then, OMP takes the largest possible step in the space spanned by the columns (a_1,a_3) toward the vector y. After updating the solution vector, the residual will be zero and OMP terminates. Note that the column a_2 is never selected even though it is important for recovering the original vector x.

On the other hand, similar to OMP, LARS starts by adding the column a_1 to the active set at the first iteration as shown in Figure 4.2. However, LARS moves in the direction of the column a_1 until the column a_2 has absolute correlation with the current residual as much as a_1. At the second iteration, LARS adds the column a_2 to the active set, and moves in the direction that is equiangular with both a_1 and a_2 toward the vector y as shown in Figure 4.3. The residual will be zero after updating the solution vector. Therefore, LARS terminates at the end of the second iteration. Note that LARS selects the column a_2 which is essential to reconstruct the original vector x, while OMP does not.

Figure 4.4 shows the coefficients of the reconstructed vector $\hat{x}$ over iterations of OMP and LARS. As shown in the figure, LARS achieves very small error norm (almost zero), while OMP obtains a high error norm. Note that the error is computed by the following equation:

$$error = x - \hat{x} \tag{4.14}$$

where x is the original sparse vector, and $\hat{x}$ is the reconstructed one by the algorithms

Therefore, this implies that LARS can reconstruct the sparse vector x when two or more columns of the matrix A are highly correlated, while OMP fails. However, LARS is slower than OMP.

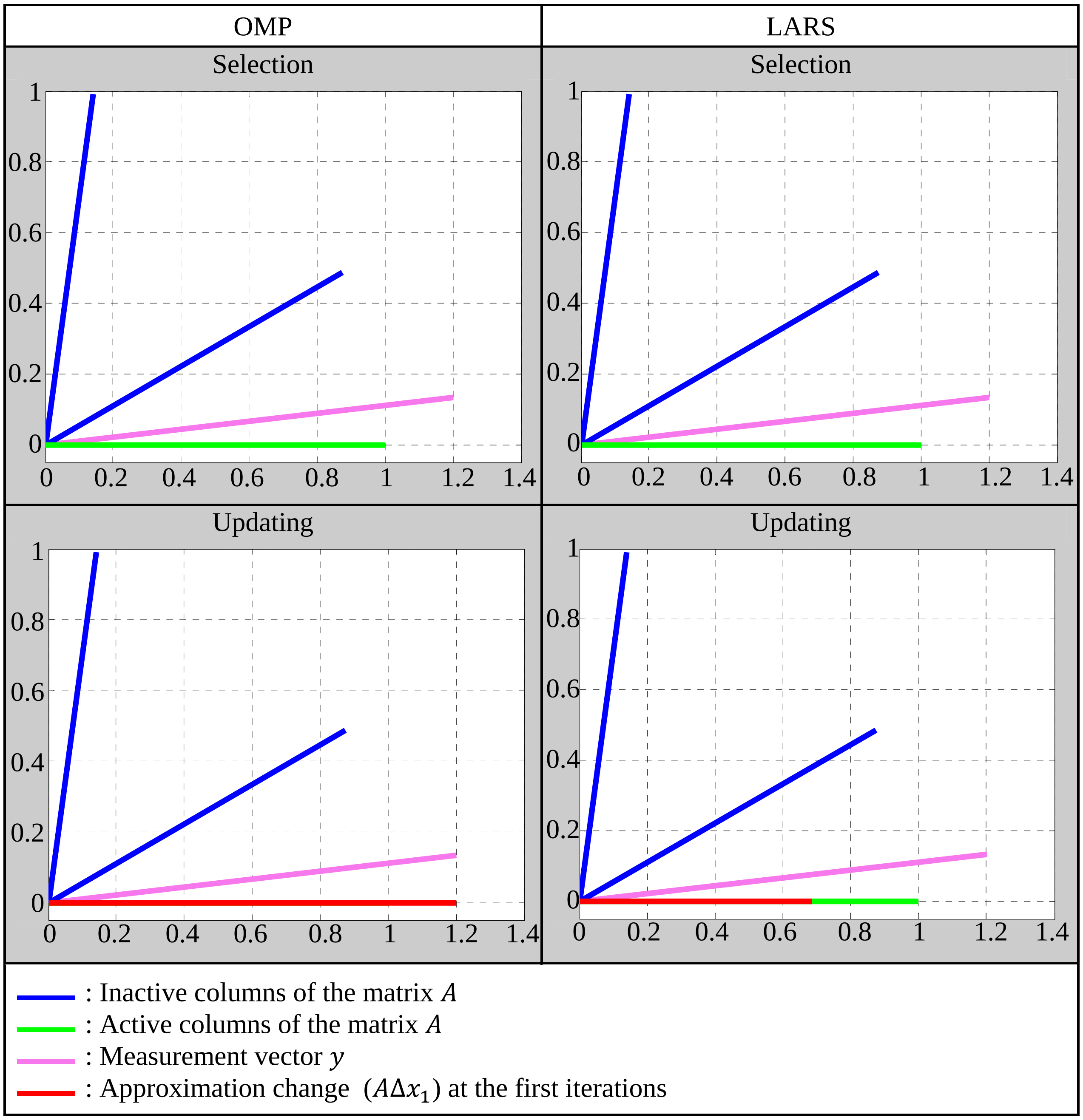


Figure 4.2: The selection and updating steps at the first iteration of OMP and LARS in Example 4.1.The approximation change is obtained via multiplying the matrix A by the current solution update vector Δx_t

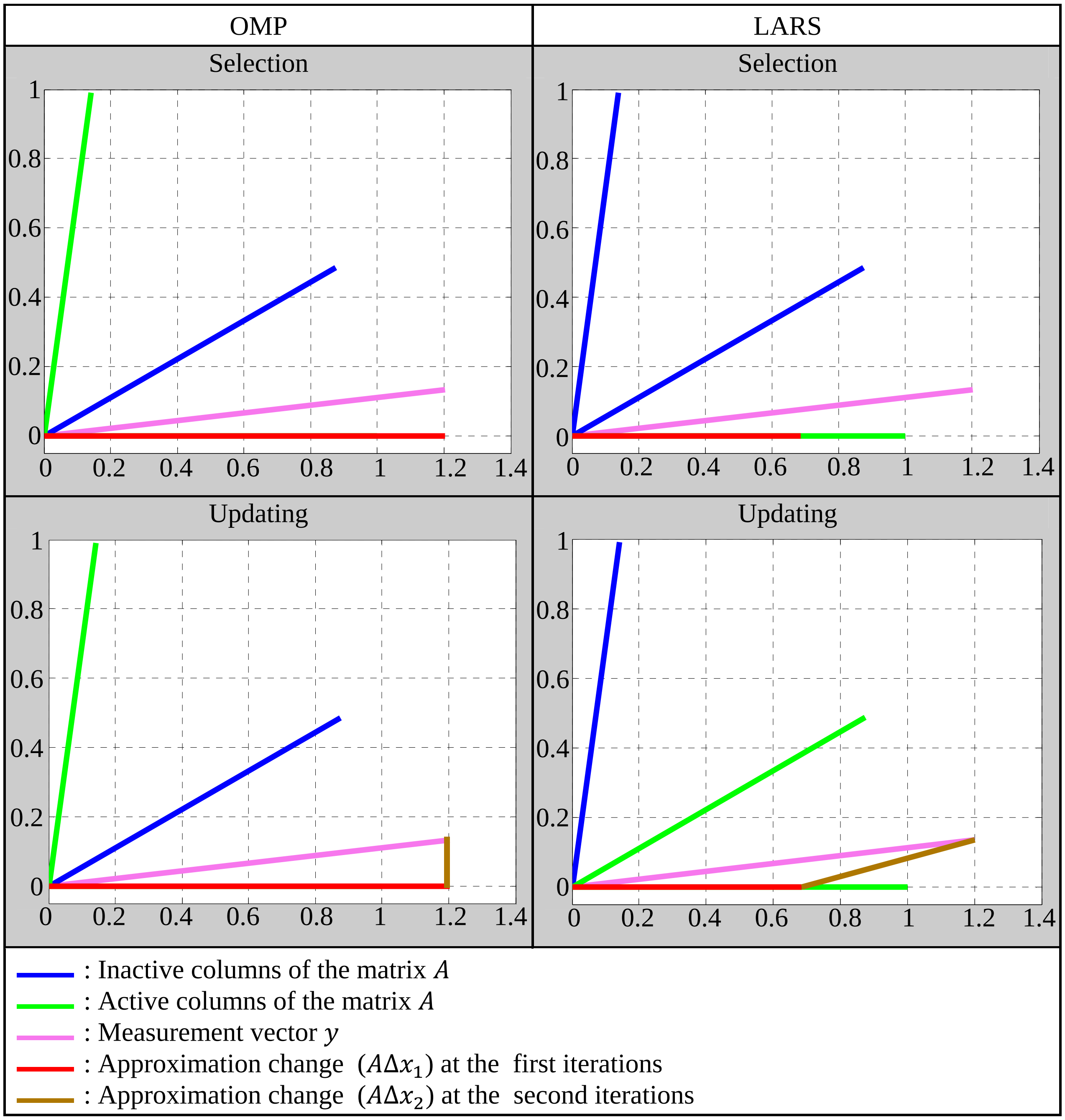


Figure 4.3: The selection and updating steps at the second (last) iteration of OMP and LARS in Example 4.1. The approximation change is obtained via multiplying the matrix A by the current solution update vector Δx_t

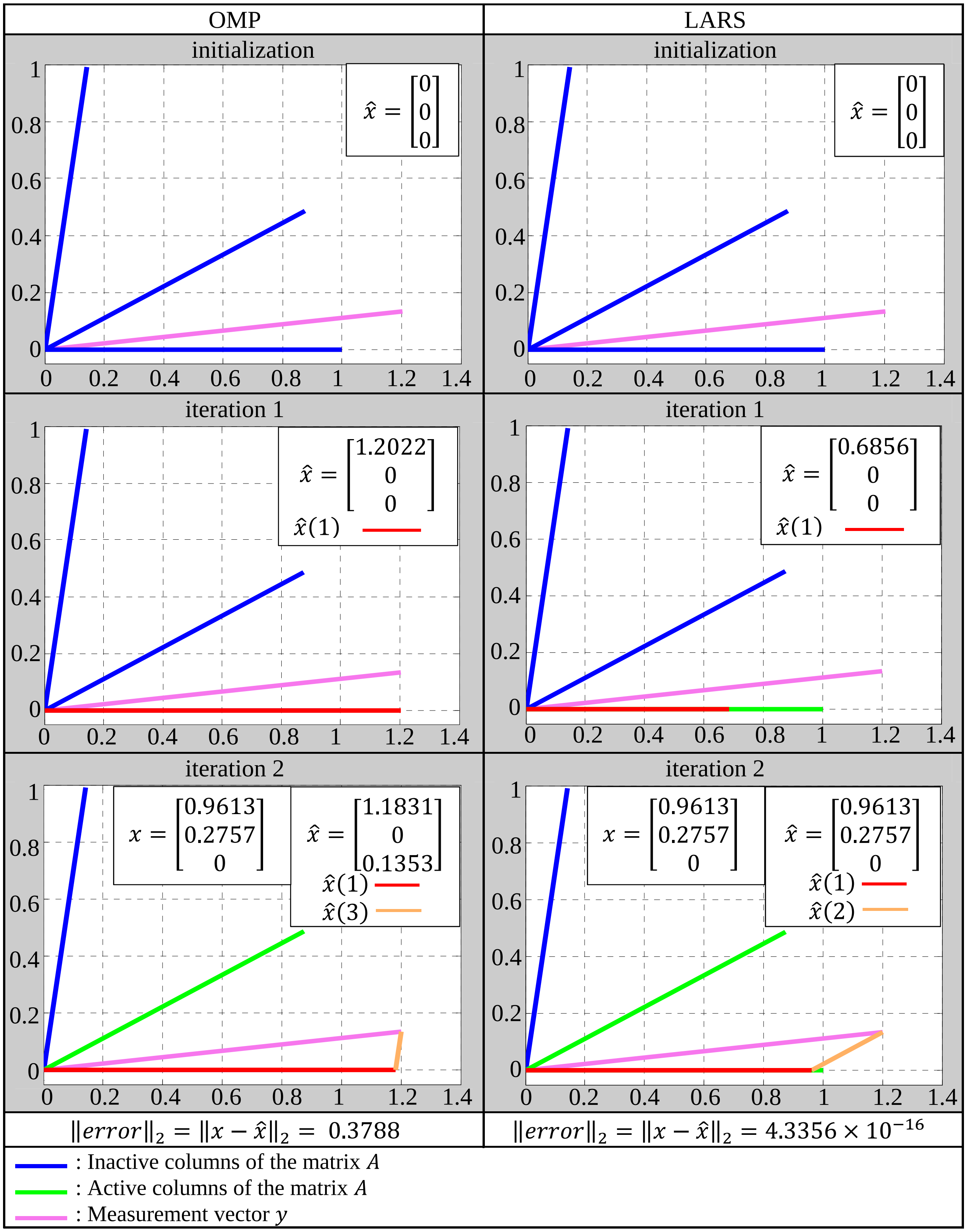


Figure 4.4: The solution coefficients of the reconstructed vector $\hat{x}$ over iterations of OMP and LARS in Example 4.1, and Euclidean norm of error between the original sparse vector x and the reconstructed vector $\hat{x}$ after the algorithms terminate

Chapter 5

Simulation Results and Discussion

In this chapter, we simulate the comparative analysis of the OMP and LARS algorithms stated in previous chapters by using MATLAB. In general, we would go through different examples to demonstrate the similarities and differences between OMP and LARS from different perspectives. First, we employ OMP and LARS to reconstruct some images from their compressive samples. Afterward, we exploit parallel processing techniques to speed-up convergence of the algorithms, and observe the efficiency of using different number of processors. Next, we examine the performance of OMP and LARS in terms of mean square error (MSE) as a function of the measurement size m and the sparsity k. To study the difference in the updating process of OMP and LARS, we observe the coefficients of the solution and correlation vectors over iterations. Finally, to study the performance of OMP and LARS from different aspect, we generate two different examples of an underdetermined linear system. We utilize OMP and LARS to reconstruct the sparse vector in each example. At each iteration of the algorithms, we plot the updating process in three dimensions (3D) view.

5.1 Reconstructing images from their compressive samples

To demonstrate the efficiency of OMP and LARS, we reconstruct gray images from their compressive samples by using both algorithms, and measure the error between the original image and the reconstructed one. To estimate the convergence speed of each algorithm, we record the time and the number of iterations that algorithms require to reconstruct the images.

We use images of size (512×512) pixels. First, we divide each image into 4096 patches of size (8×8) pixels. We reshape every patch into a vector of size (64×1) as shown in Figure 5.1. Each patch vector is normalized through dividing its entries by the maximum value found in the image.

The matrix A is obtained by multiplying two matrices: $m \times n$ random matrix ϕ, and $n \times n$ transformation matrix ψ:

$$A_{m \times n} = \phi_{m \times n} \times \psi_{n \times n} \tag{5.1}$$

where $n = 64$.

The random matrix ϕ is generated by using Gaussian distribution, and the transformation matrix ψ is generated by using Discrete Cosine Transform (DCT) [41].

To obtain the measurement vector y, the random matrix ϕ is multiplied by each patch of image:

$$y^i = \phi \times p^i \tag{5.2}$$

where p^i represents the i^{th} patch, and $i = 1,2, \ldots \ldots ,4096$

Now, the underdetermined linear system is expressed as follows:

$$A \times x^i = y^i \tag{5.3}$$

where x^i is the sparse representation of the patch p^i.

To obtain x^i for each i, we employ OMP and LARS with the system (5.3) as shown in Figure 5.1. Then, the transformation matrix ψ is multiplied by x^i to obtain the reconstructed patch $\hat{p}^i$:

$$\hat{p}^i = \psi \times x^i \tag{5.4}$$

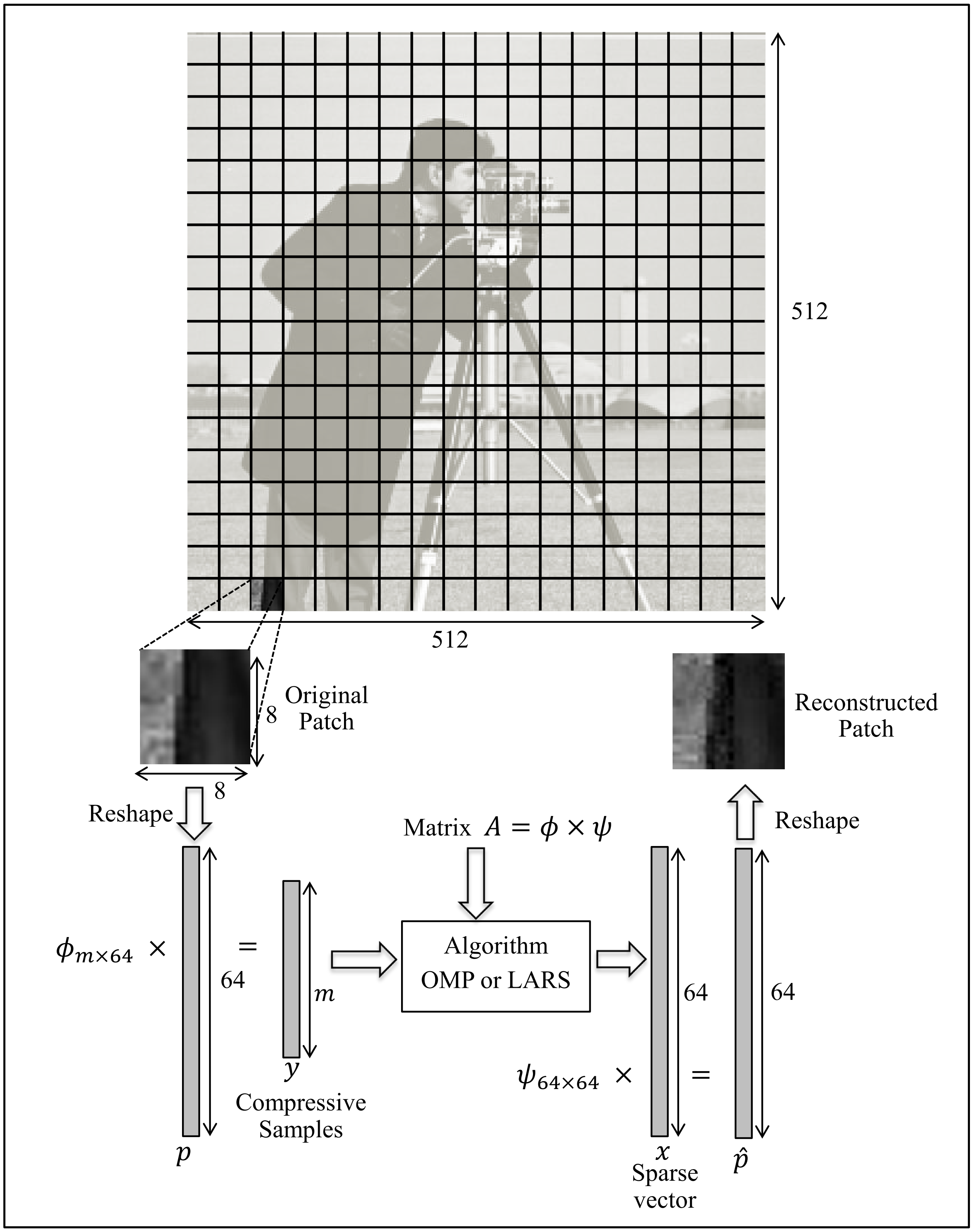


Figure 5.1: Reconstruct a patch of an image from its compressive samples by using OMP or LARS

After reconstructing all patches, we form the recovered image. Figure 5.2 and Figure 5.3 show the recovered images for measurement size $m = 32$ and $m = 16$ respectively.

Then, we compute the mean square error (MSE) between the original patch and the reconstructed one:

$$MSE^i = \frac{1}{n}\sum_{j=1}^{n}\left(p^i(j) - \hat{p}^i(j)\right)^2 \tag{5.5}$$

After computing MSE for all 4096 patches, we calculate the average mean square error over them:

$$Average\ MSE = \frac{1}{4096}\sum_{i=1}^{4096} MSE^i \tag{5.6}$$

We depend on the value of Peak Signal to Noise Ratio (PSNR) to determine the performance of the algorithms which is computed by:

$$PSNR = 10\log_{10}\left(\frac{MAX(x)}{Average\ MSE}\right) \tag{5.7}$$

where $MAX(x)$ is the maximum possible entry in the patches. Note that $MAX(x) = 1$ because the image patches are normalized.

If an algorithm has high PSNR, this implies that it can successfully reconstruct the original images with small error. The values of PSNR for OMP and LARS with measurement size $m = 32$ and $m = 16$ are shown in Figure 5.2 and Figure 5.3 respectively.

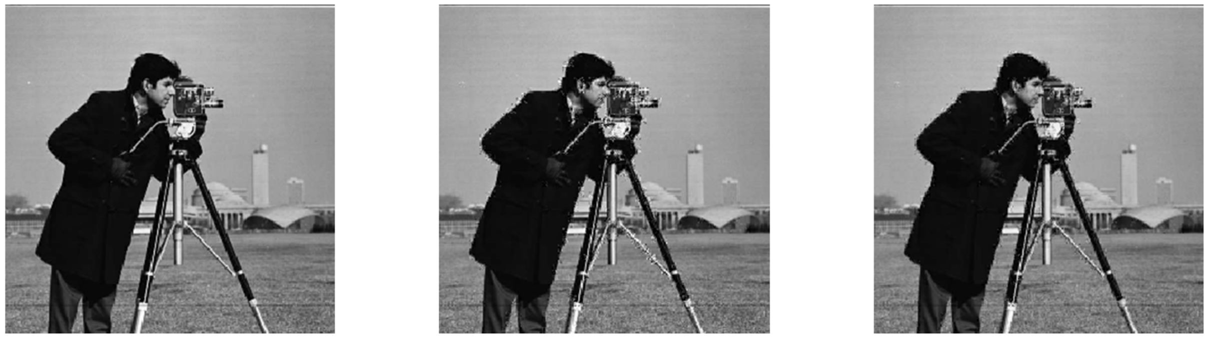
To estimate the convergence speed of the algorithms, we record the number of iterations that each algorithm requires to converge to the final solution for each patch. Then, we compute the average number of iterations over all patches in the same way that we did with MSE. However, the number of iterations is not an accurate measure of the algorithms' speed because computation involved at each iteration of LARS takes longer time than the computation of OMP iteration. To get a better measurement of the algorithms' speed, we also record the required time for each algorithm to recover the whole image in a certain computer. In this simulation, we use a Dell laptop that is equipped with CPU Intel (i3, 2.13 GHz) and (4 GB) of memory. The average number of iterations and the required time for OMP and LARS with measurement size $m = 32$ and $m = 16$ are shown in Figure 5.2 and Figure 5.3 respectively.

From Figure 5.2 and Figure 5.3, it is observable that the number of iterations required by OMP is less than the number of iterations required by LARS to recover the same images. This is because OMP always adds columns to the active set, while LARS adds or removes columns to/from the active set. Equally important, OMP consumes less time than LARS. Nevertheless, LARS obtains higher PSNR than OMP. The justification for this is that LARS selects the most important columns from the matrix A to recover the sparse solution even though these columns are highly correlated, while OMP does not.

original image reconstructed image (OMP) reconstructed image (LARS)

	OMP	LARS
PSNR	26.3780	28.2098
Time (second)	7.58966	22.27399
Average iteration	32	36

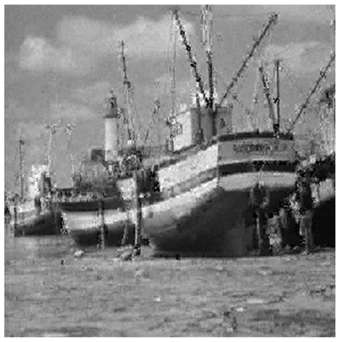
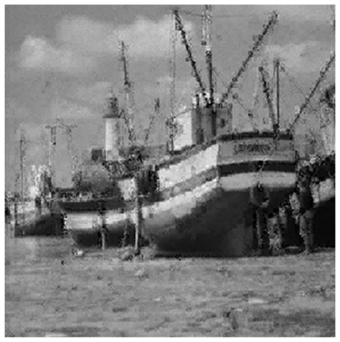
original image reconstructed image (OMP) reconstructed image (LARS)



	OMP	LARS
PSNR	28.3421	30.0823
Time (second)	5.42805	16.819573
Average iteration	23	27

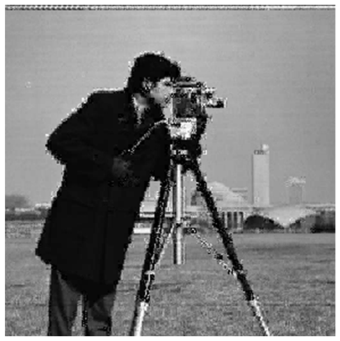
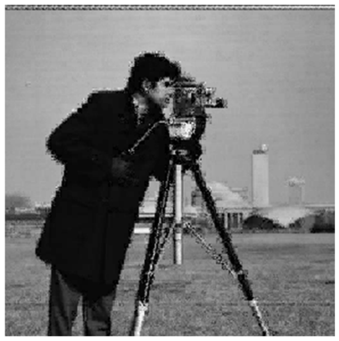
Figure 5.2: Reconstructed images from their compressive samples by using OMP and LARS with $m = 32$ and $n = 64$

original image reconstructed image (OMP) reconstructed image (LARS)

	OMP	LARS
PSNR	21.8220	22.4618
Time (second)	2.452597	8.807559
Average iteration	16	19

original image reconstructed image (OMP) reconstructed image (LARS)

	OMP	LARS
PSNR	21.1888	22.5433
Time (second)	2.138599	7.632700
Average iteration	14	16

Figure 5.3: Reconstructed images from their compressive samples by using OMP and LARS with $m = 16$ and $n = 64$

5.2 Using parallel processing to speed-up convergence of the algorithms

In the previous section, we divided an image into a number of patches. After that, the compressive samples of each patch were obtained. Next, the algorithms (OMP and LARS) were used to reconstruct the original patches as illustrated in Figure 5.1. Because samples of each patch are independent of samples of other patches, and the same algorithmic steps are executed to samples of all patches, we can utilize the parallel processing techniques to expedite the convergence of the algorithms.

According to the Flynn classification [42], reconstructing all patches of an image from their compressive samples falls under *single instruction multiple data* (SIMD) class. This is because identical algorithmic steps are performed to samples of different patches to recover the whole image. In this case, samples of patches are distributed over different processers. All processors execute the same algorithmic steps to recover their own patches. Subsequently, the recovered patches are combined from processers to form the entire image.

Both OMP and LARS calculate the Gramian matrix ($A^T A$) to solve the least square problem over active columns. Computing the Gramian matrix involves heavy computation. Equally important, the matrix A is identical for all patches. Therefore, for efficiency reason, the Gramian matrix ($A^T A$) is off-line computed first, and then it is distributed to all processors [43].

To examine impact of using many processors on convergence time, we reconstruct an image from its compressive samples as we did in Section 5.1. However, in this section, we use more than one processor to run OMP and LARS. We illustrate variation of the convergence time with respect to the number of used processors as shown in Figure 5.4. It is noticeable that the convergence time reduces as the number of used processors increases. In addition, we observe that the reduction in convergence time is not significant as the number of processors increases above 6. This is because the communication overhead among processors grows as the number of used processors increases.

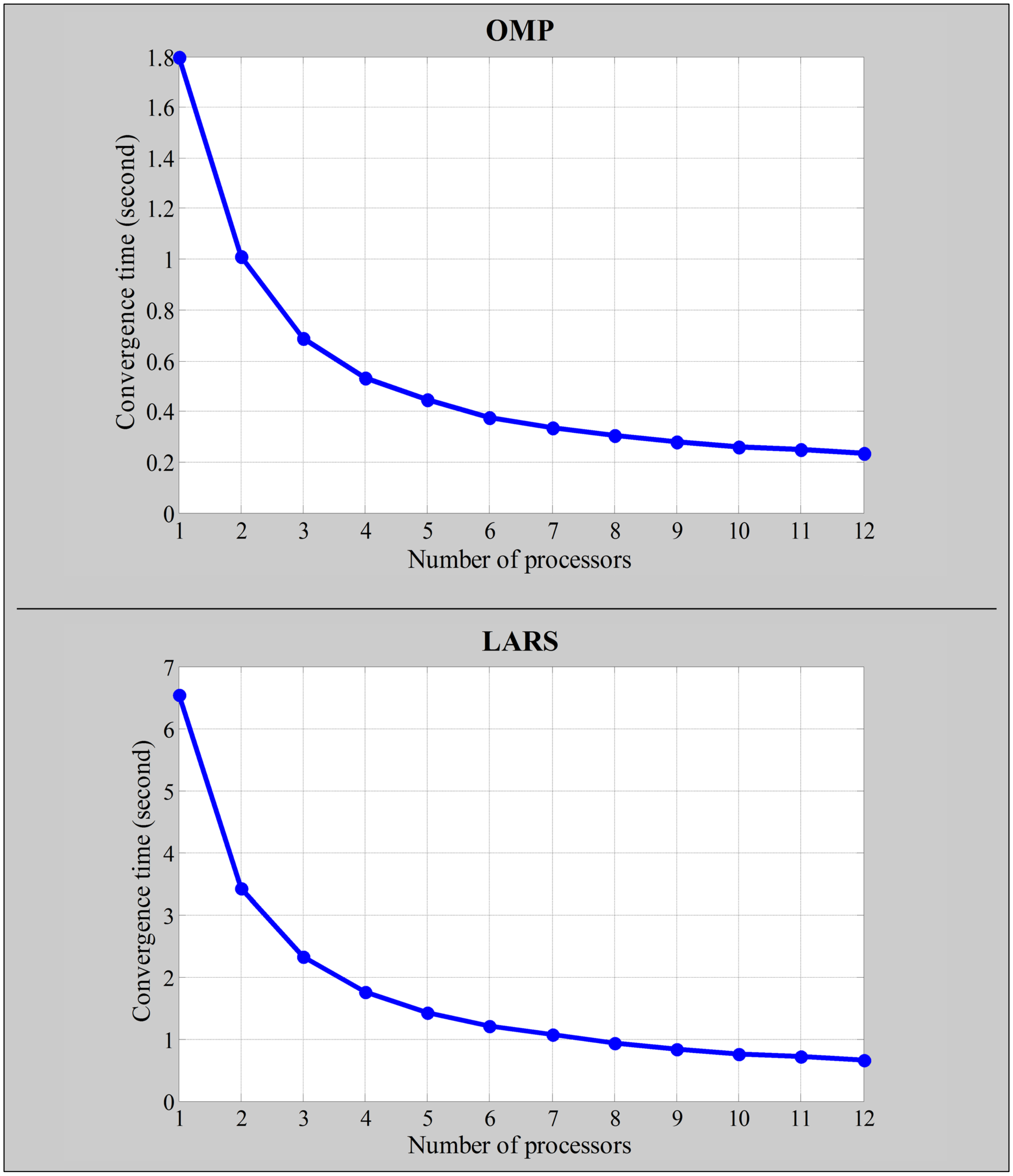


Figure 5.4: The convergence time of OMP and LARS against the number of used processors

To calculate the gain of using many processors, we compute the speedup ratio by using the following equation:

$$Speedup = \frac{T(1)}{T(p)} \tag{5.8}$$

where $T(1)$ is the convergence time of an algorithm by using one processor, $T(p)$ is the convergence time of an algorithm by using p processors, and $1 \leq Speedup \leq p$.

The ideal speedup ratio equals to the number of used processors (p). Normally, the speedup ratio is less than p because of time lost in communication overhead among processors. The ideal and actual speedup ratios for the OMP algorithm are shown in Figure 5.5.

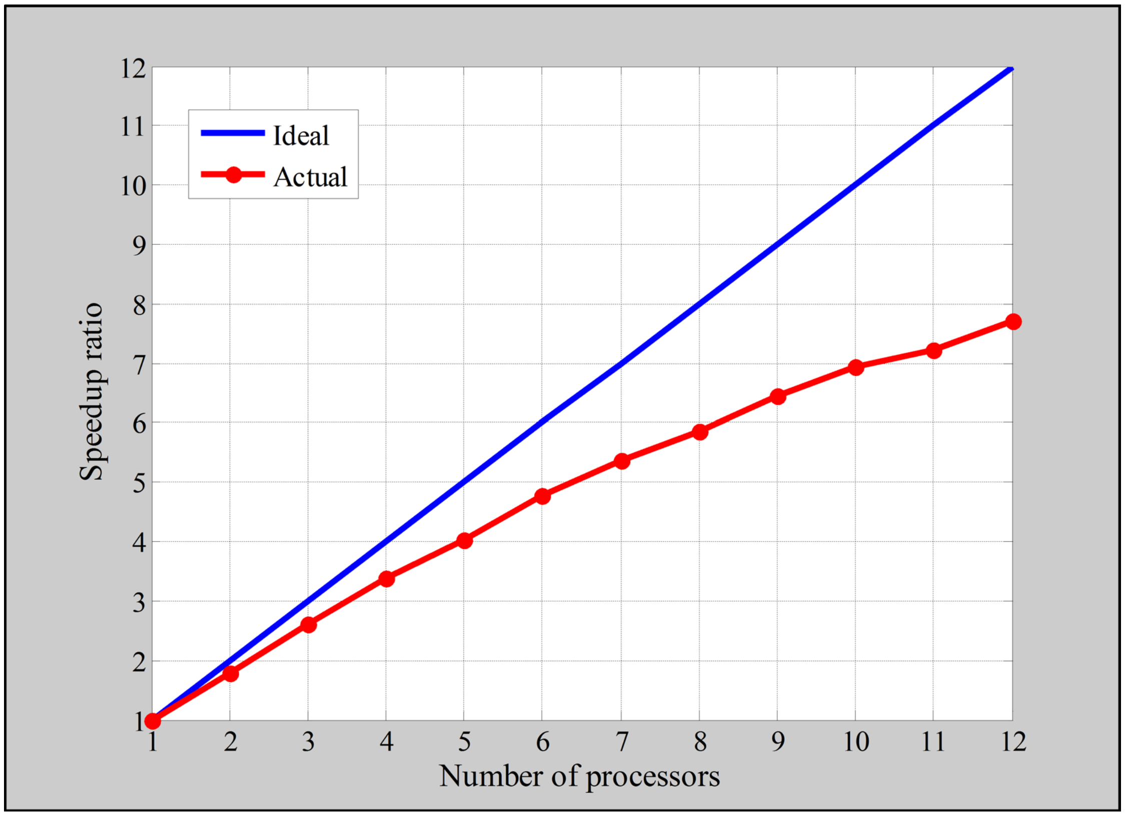


Figure 5.5: The ideal and actual speedup ratios for the OMP algorithm against the number of used processors

5.3 Impact of measurement size and sparsity on MSE

In this section, we examine the performance of OMP and LARS in terms of mean square error (MSE) as a function of the measurement size m and the sparsity k. We consider various underdetermined linear systems that have different measurement size m and different sparsity k. For all of the considered underdetermined linear systems, the value of n is fixed to 1000. For each trial of underdetermined linear system, we randomly generate the matrix A of size $m \times 1000$ by using Gaussian distribution. The columns of the matrix A are normalized to have unit ℓ_2 norm. Then we generate the sparse vector $x \in R^{1000}$ which has k nonzero entries in random positions. The nonzero entries of the vector x are also obtained from Gaussian distribution, and all other entries are set to zero. The vector x is also normalized. We multiply the matrix A by the sparse vector x to compute the measurement vector y:

$$y = A \times x \tag{5.9}$$

At this point, we employ OMP and LARS to reconstruct the original sparse vector of undetermined linear system described by Equation (5.9). Then, we compute MSE between the original sparse vector and the reconstructed one as follows.

$$MSE = \frac{1}{n}\sum_{j=1}^{n}\left(x(j) - \hat{x}(j)\right)^2 \tag{5.10}$$

where $\hat{x}$ is the reconstructed sparse vector by the algorithms

In Figure 5.6, the average MSE of 100 trials is illustrated as a function of m for different values of k with using OMP and LARS. For each value of k, we observe that the average MSE decreases as the value of m increases, until it settles to a very small value about (10^{-34}) after specific value of m, say $\widetilde{m}$. Having the average MSE approaching such small value implies that the algorithms successfully reconstruct the original sparse vector. Hence, the value $\widetilde{m}$ is considered as the minimum size of measurements to make the algorithms able to recover the original k sparse vector. From the figure, we observe that OMP and LARS require more measurement m to successfully recover the sparse vector as number of nonzeros k is increased. In addition, we notice that LARS recovers the sparse vector with less average MSE than OMP for $m < \tilde{m}$. Therefore, LARS obtains better results when the measurement size m is smaller than $\tilde{m}$.

To examine the performance of the algorithms from a different viewpoint, we plot the average MSE as a function of k for different values of m as shown in Figure 5.7. We can see that the average MSE starts from almost zero; then after a certain value of k, say $\tilde{k}$, it obviously increases to some values above zero. Therefore, by using the measurement vector of size m, the algorithms can recover a k sparse vector if $k < \tilde{k}$. Furthermore, we observe that LARS recovers a k sparse vector with less average MSE than OMP when $k > \tilde{k}$. As a result, we can state that LARS is better to be used than OMP when the sparsity k is fairly large.

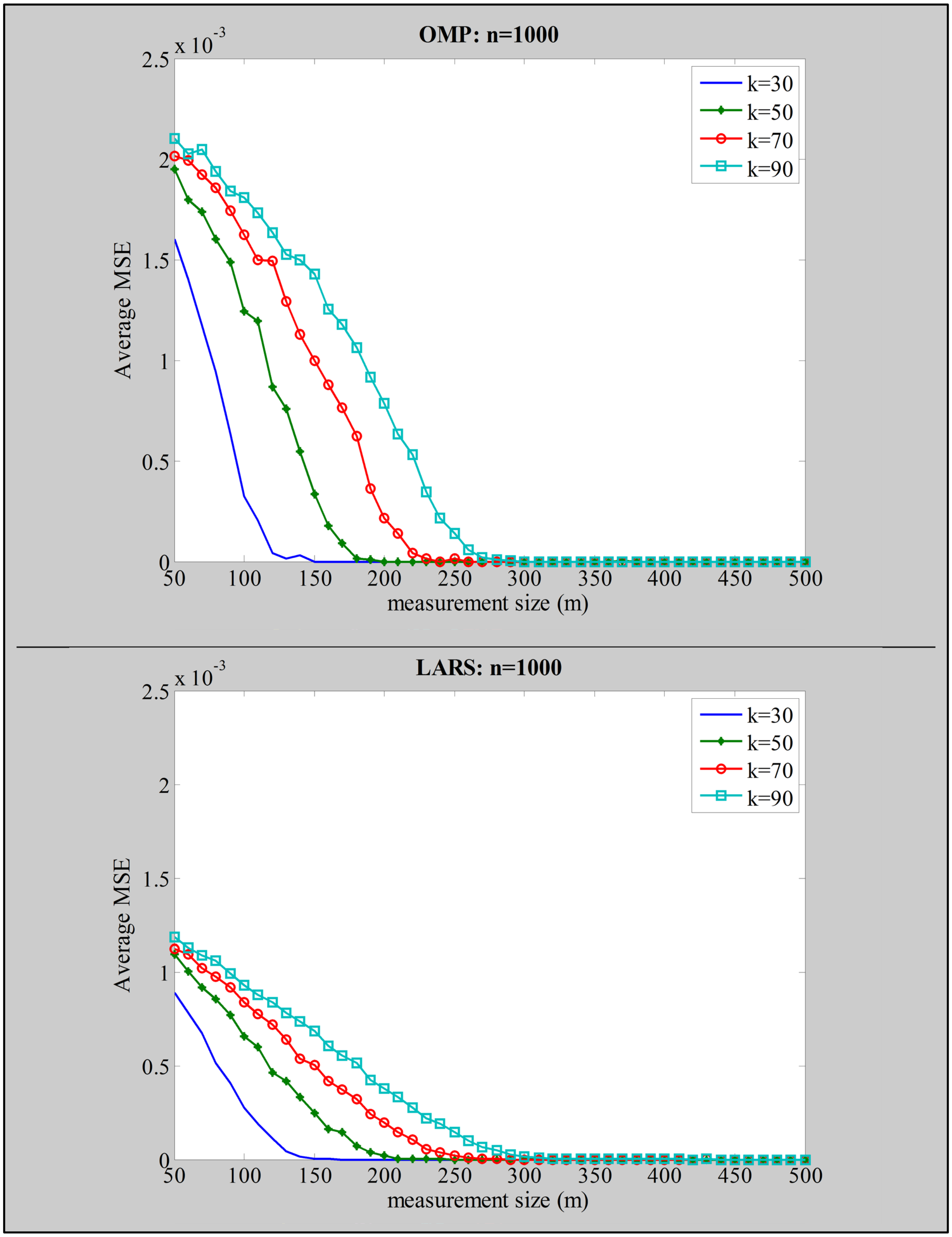
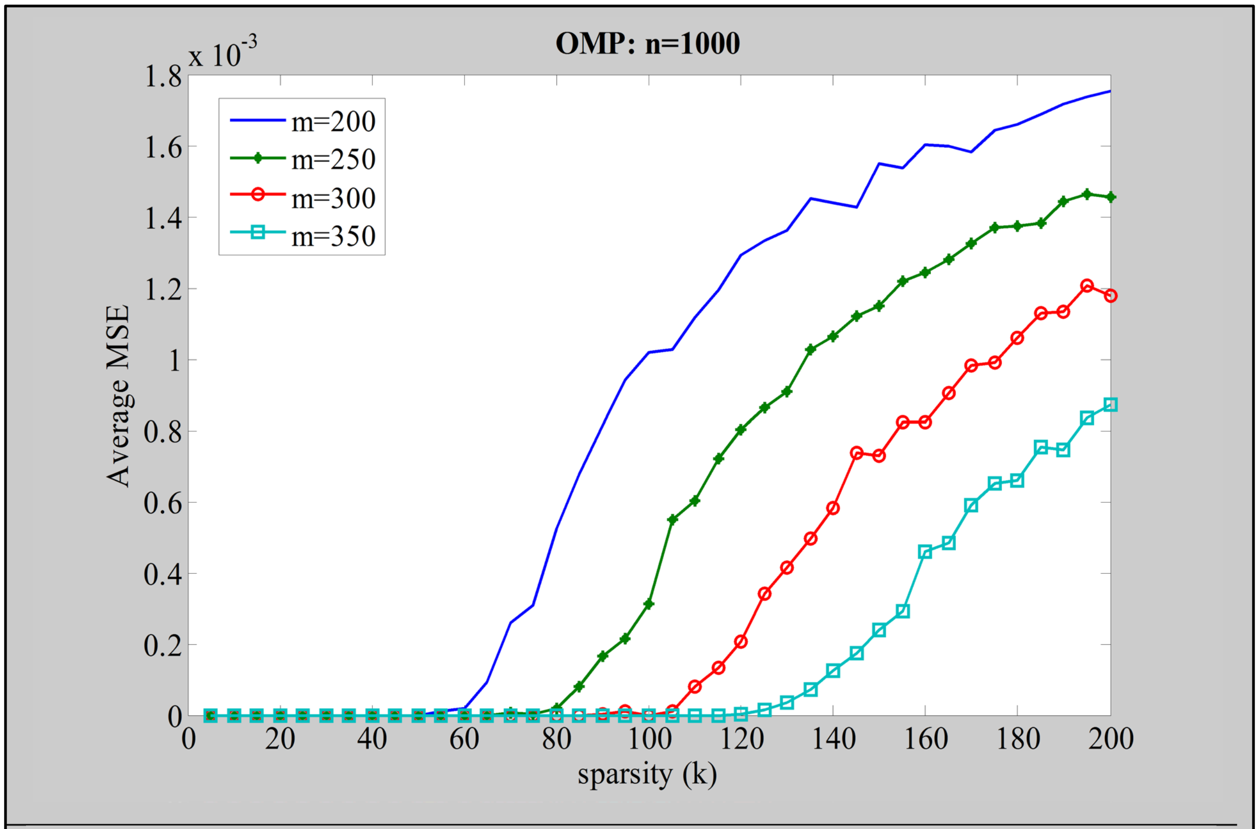


Figure 5.6: Average MSE of 100 trials between the original sparse vectors and the reconstructed ones by OMP and LARS as a function of measurement size m for different sparsity k



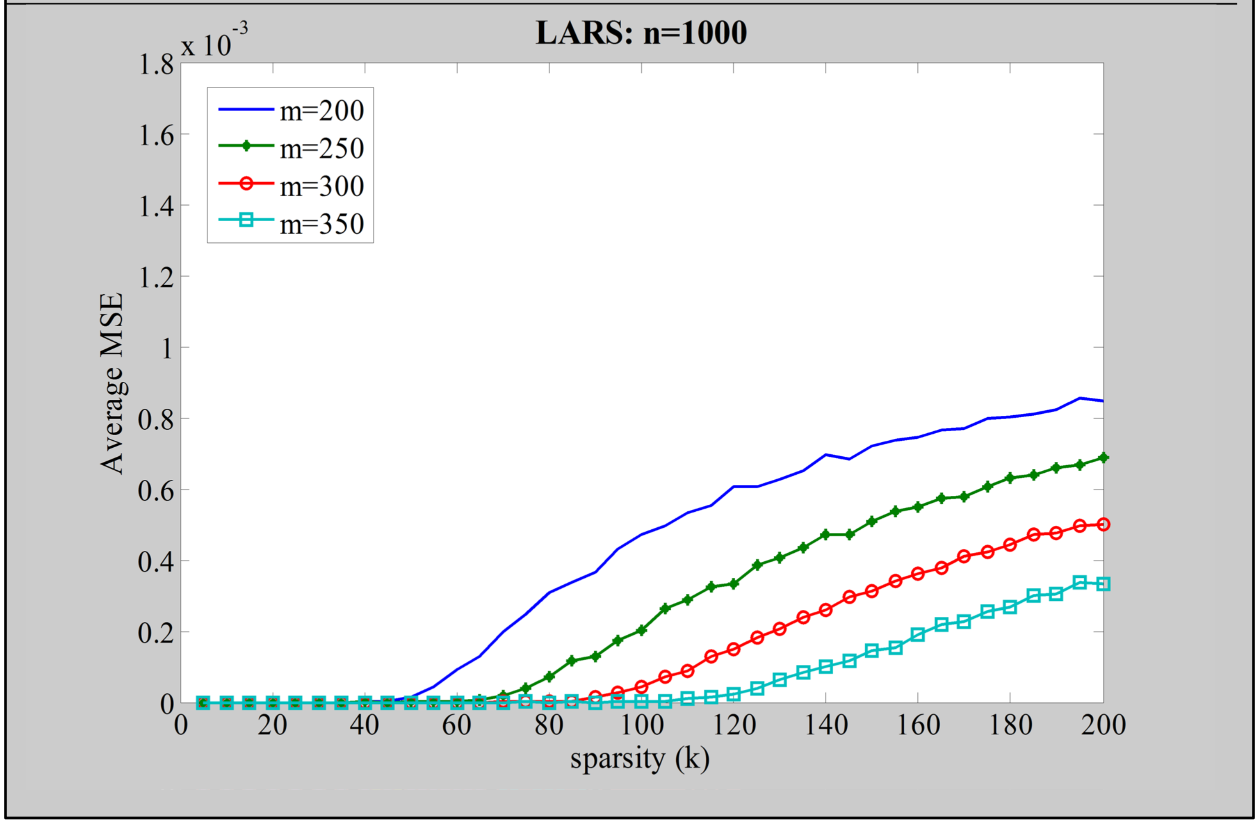


Figure 5.7: Average MSE of 100 trials between the original sparse vectors and the reconstructed ones by OMP and LARS as a function of sparsity k for different measurement size m

5.4 Variation of the solution and correlation coefficients over iterations

In this section, we aim to monitor the updating process of OMP and LARS to the solution vector over iterations. We use both OMP and LARS to recover a sparse solution of an underdetermined linear system. At each iteration of OMP and LARS, we plot the solution coefficients. To run an example, we consider an underdetermined linear system with $m = 50$, $n = 100$, and $k = 10$. We generate the matrix A and the sparse vector x of the system by using Gaussian distribution in the same way we did in Section 5.3. To obtain the measurement vector y, we multiply the matrix A by the sparse vector x as express in Equation (5.9). At this point, we employ OMP and LARS to reconstruct the original sparse vector. The solution coefficients that corresponding to the active columns are shown over iterations of OMP and LARS in Figure 5.8.

Form Figure 5.8, we observe that both OMP and LARS terminate with the same solution coefficients. However, they are different in the way of updating the solution coefficients. In OMP, when a column is added to the active set, the corresponding solution coefficient is increased to the largest possible value. The coefficients of the active set may decrease slightly in the subsequent iterations, because they will be affected by other columns that would be added to the active set later. On the other hand, at each iteration of LARS, all coefficients corresponding to the active set are increased to the smallest possible step in a way that makes a column from the inactive set join the active set at the next iteration. In addition, we observe that columns are added to the active set in different order in the algorithms as shown in Figure 5.8. Moreover, LARS requires one more iteration than the case of OMP to converge to the final solution, since it adds one more column (column 72 in this example) to active set, and increase its corresponding solution coefficients to a very small value (almost zero). The reason behind this is that LARS solves the ℓ_1 norm minimization problem. Therefore, some coefficient that are zero in original sparse vector would be equal to very small value but not exactly zero in the reconstructed sparse vector by LARS.

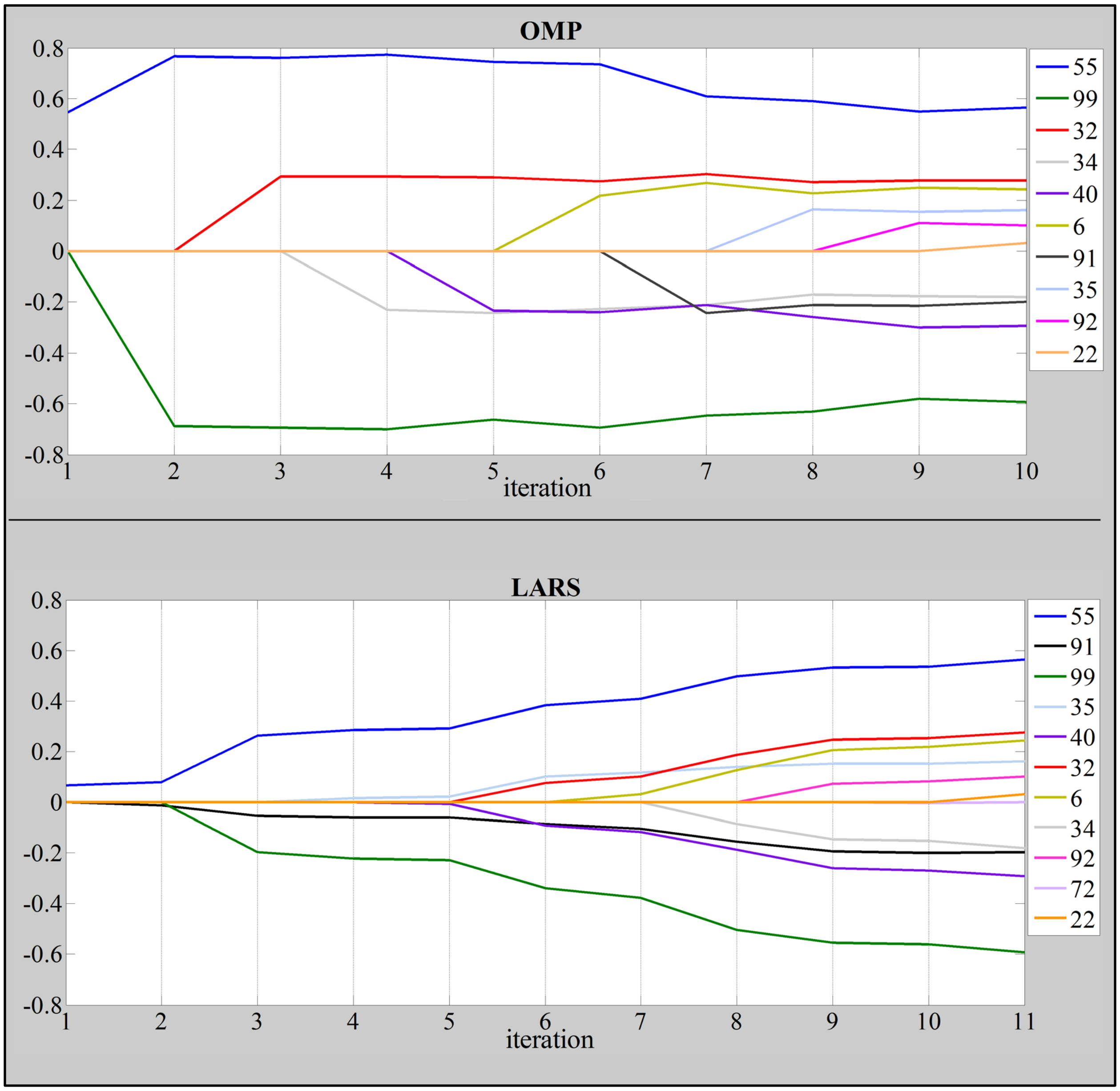


Figure 5.8: The solution coefficients that correspond to the active columns over iterations of OMP and LARS

Furthermore, to see how the correlations of columns of the matrix A with the current residual are changed, we illustrate them over some iterations of OMP and LARS as shown in Figure 5.9 and Figure 5.10 respectively.

At each iteration, OMP adds a column that has maximum absolute correlation with current residual vector to the active set, and updates the solution coefficients to make its correlation zero at the next iteration. This is because OMP ensures that the current residual vector is always orthogonal to all active columns. For example in Figure 5.9, OMP adds the 99^{th} column to the active set at the second iteration, and its correlation would be zero at the third iteration. OMP continues to add a column to the active set until the norm of the current residual approaches zero.

On the other hand, LARS first selects a column that has the maximum absolute correlation with the initial residual (the vector y), and then updates the corresponding solution coefficient in a way that makes the absolute correlation of selected column equal to the largest absolute correlation over columns that are not in the active set. For instance in Figure 5.10, LARS selects the 55^{th} column at the first iteration, and updates the solution coefficient to make its absolute correlation similar to the absolute correlation of the 91^{th} column. Therefore, the 91^{th} column would be added to the active set at the second iteration. In similar manner, LARS iteratively adds a column to the active set, and decreases the absolute correlations of the active columns until no column has correlation with the current residual.

We conclude that OMP updates the solution vector to make absolute correlations of the active columns equal to zero, while LARS updates the solution vector to make absolute correlations of the active columns equal to same value (λ) as follows:

$$\begin{array}{ll} \text{OMP:} & |c(i)| = 0 \\ \text{LARS:} & |c(i)| = \lambda \end{array} \qquad \forall\, i \in I \tag{5.11}$$

Note that λ is decreased at each iteration; when $\lambda = 0$, LARS converges to the final solution vector.

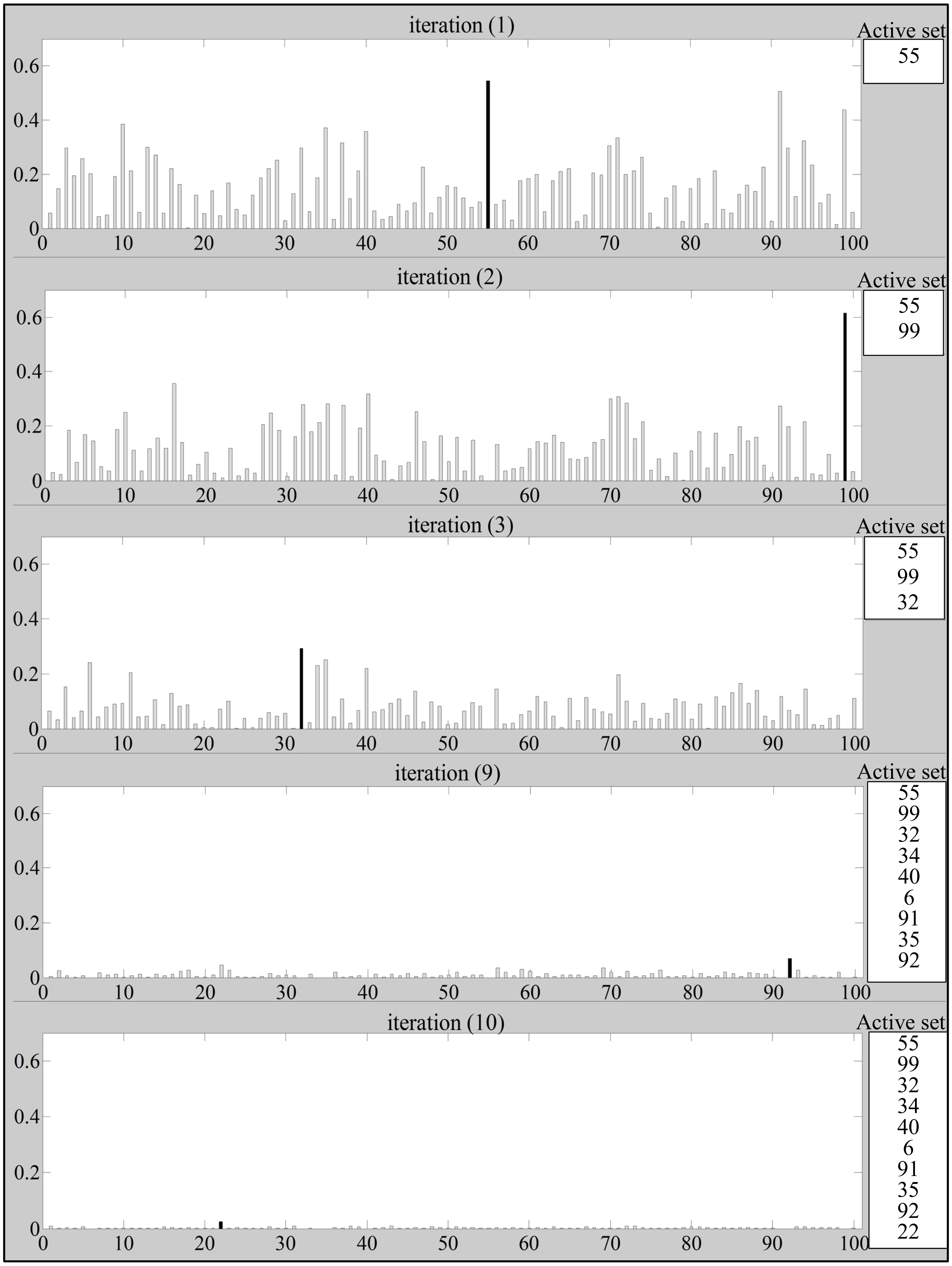


Figure 5.9: Absolute correlation of columns of the matrix A in some iterations of OMP

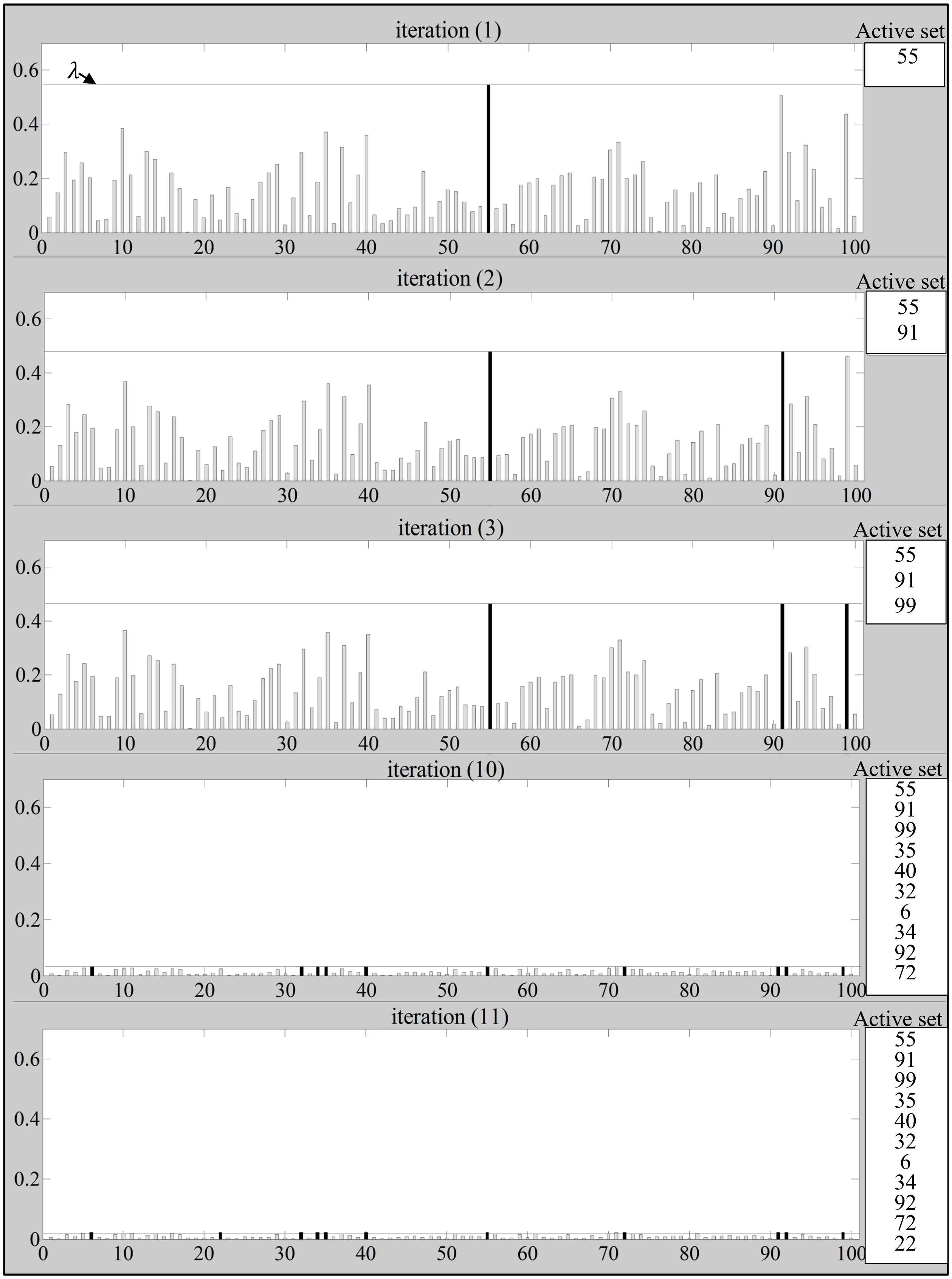


Figure 5.10: Absolute correlation of columns of the matrix A in some iterations of LARS

5.5 Study the algorithms performance through three dimensions (3D) examples

In this section, we study the performance of OMP and LARS from different perspective by using three dimensional systems. We create two different examples of underdetermined linear system with $m = 3$, $n = 6$, and $k = 2$. Then, we use OMP and LARS to reconstruct the sparse vector of these examples. We illustrate the updating process of algorithms at each iteration in three dimensions (3D) view.

In both examples, we generate the matrix A and the sparse vector x by using Gaussian distribution in the same way we did in Section 5.3. We multiply the matrix A by the sparse vector x to obtain the measurement vector y as expressed in Equation (5.9). Now, by having the matrix A and the vector y, we use OMP and LARS to reconstruct the sparse vector x. To determine the performance of the algorithms, we compute MSE between the original sparse vector x and the reconstructed one by the algorithms as stated in Equation (5.10).

Example 5.1: we generate the matrix A in this example to have columns that are slightly correlated. To quantify the correlation between columns of the matrix A, we use the *mutual coherence* μ which is computed as follows:

$$\mu \overset{\text{def}}{=} arg \max_{i \neq j} |< a_i, a_j >| \tag{5.12}$$

where a_i is the i^{th} column of the matrix A

In this example, $\mu = 0.5124$ which is considered a fairly small. The columns of the matrix A and the measurement vector y are drawn in Figure 5.11. We illustrate the approximation change ($A\Delta x$) to the vector y at each iteration of OMP and LARS in Figure 5.12. We notice that both OMP and LARS obtain almost the same MSE which is approximately zero. This implies that

OMP and LARS achieve the same performance and succeed to reconstruct the sparse vector when columns of the matrix A are slightly correlated.

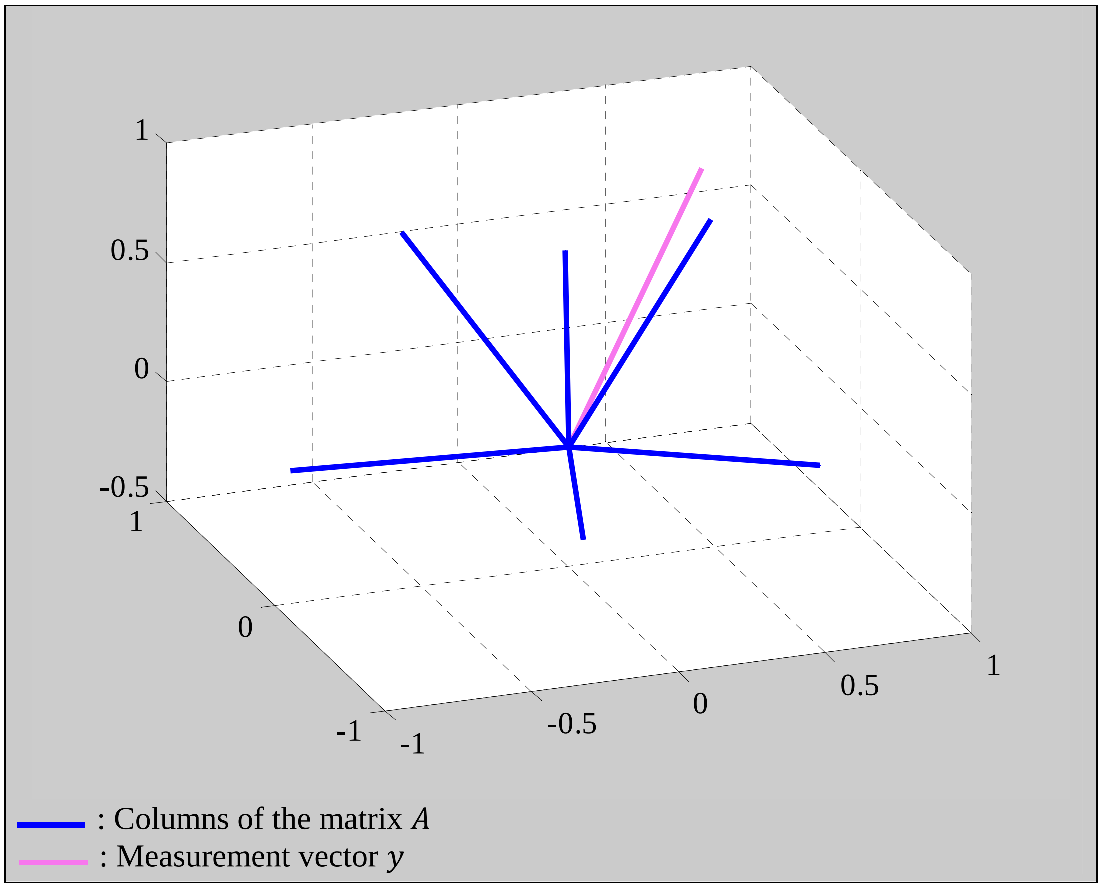
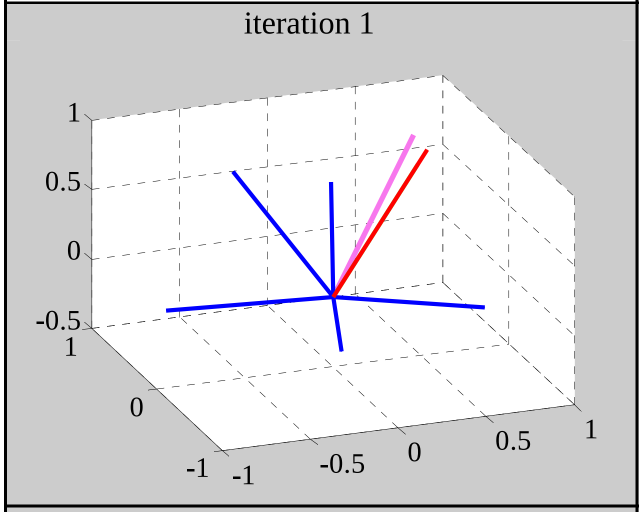
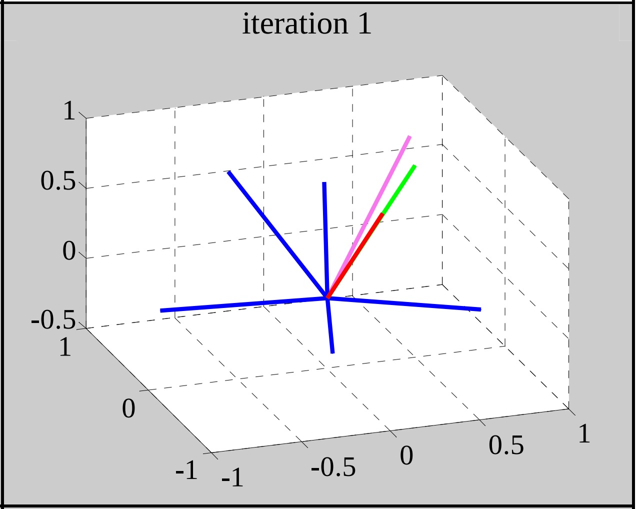
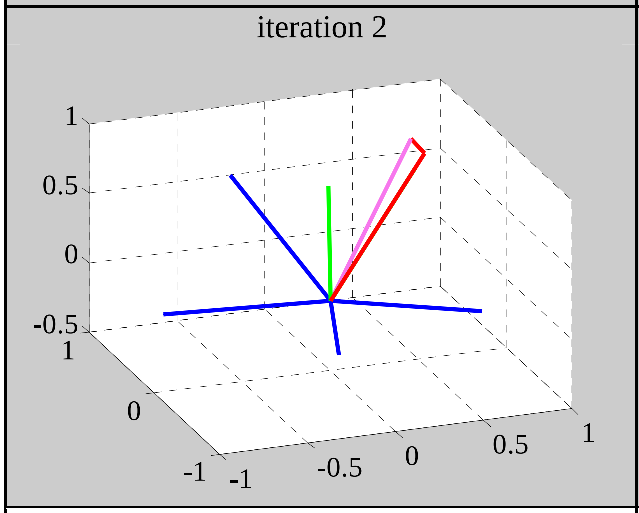
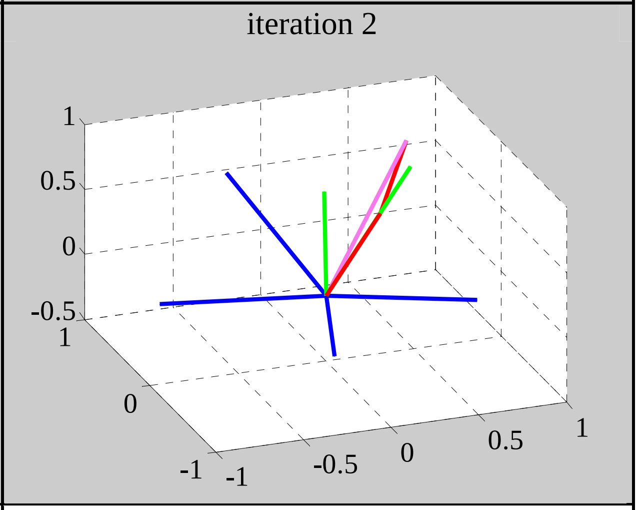


Figure 5.11: The columns of the matrix A and the measurement vector y in Example 5.1

Equally important, for lower dimensional underdetermined linear systems, a rectangular matrix (matrix contains more columns than rows) that has small mutual coherence μ is rarely existed and hard to find. To obtain the matrix $A \in R^{3\times 6}$ that has $\mu = 0.5124$ in this example, we generate 10 million matrices by using Gaussian distribution, and select the matrix that has the smallest mutual coherence μ.

OMP	LARS
	
	
$MSE = 4.1087 \times 10^{-33}$	$MSE = 4.6222 \times 10^{-33}$

🟦 : Inactive columns of the matrix A
🟩 : Active columns of the matrix A
🟪 : Measurement vector y
🟥 : Approximation change ($A\Delta x$)

Figure 5.12: The approximation change ($A\Delta x$) at each iteration of OMP and LARS in Example 5.1, where $\mu = 0.5124$, and the approximation change $= A \times \Delta x$

Example 5.2: we generate the matrix A in this example to have columns that are highly correlated, and $\mu = 0.8885$. The columns of the matrix A and the measurement vector y are drawn in Figure 5.13. As we did in Example 5.1, we illustrate the approximation change ($A\Delta x$) to the vector y at each iteration of OMP and LARS as shown in Figure 5.14. From the figure, we observe that the value of MSE in the case of LARS is very small (almost zero), while in OMP, it is high. Therefore, this implies that LARS succeeds to reconstruct the sparse vector x, while OMP fails. Equally important, LARS converges to the final solution in two iterations, while OMP requires three iterations to converge to the final solution. As a result, LARS achieves a better performance than OMP when columns of the matrix A are highly correlated.

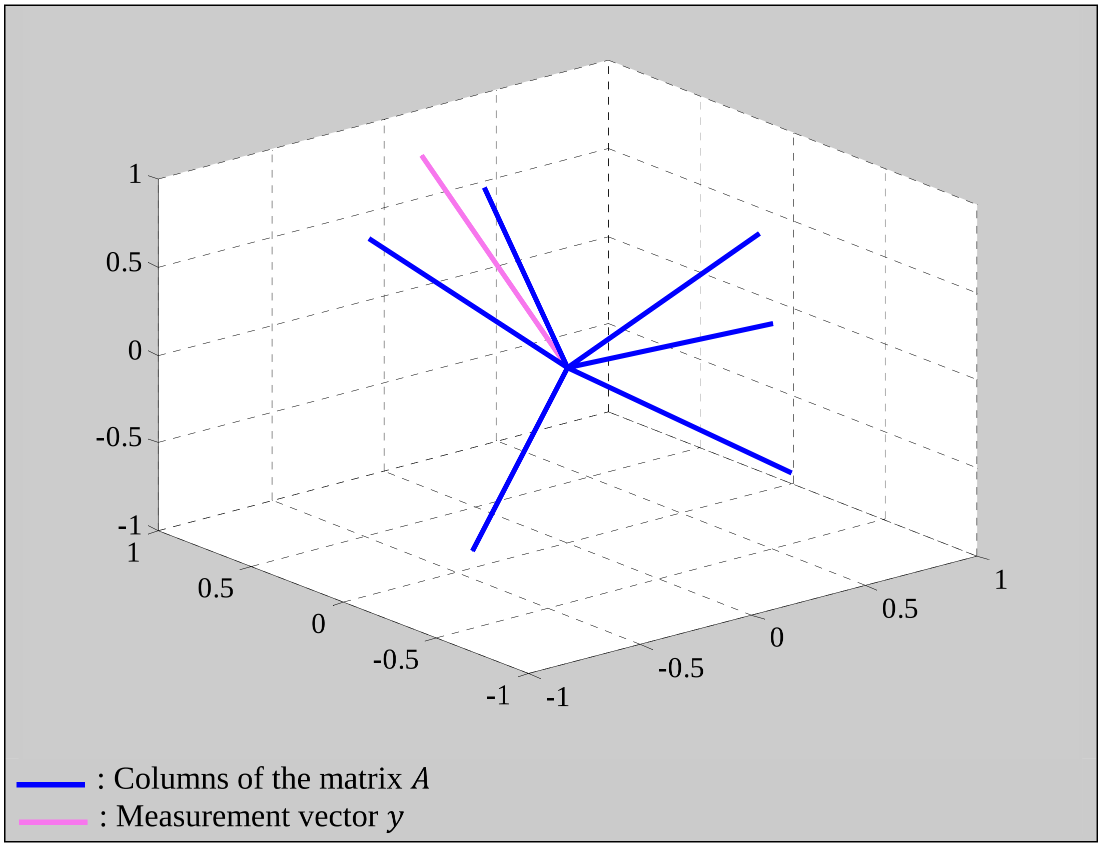


Figure 5.13: The columns of the matrix A and the measurement vector y in Example 5.2

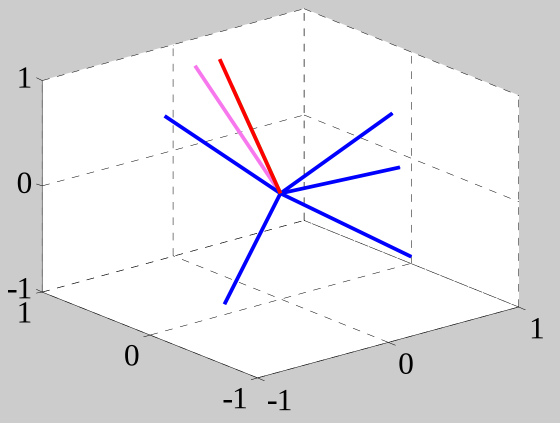
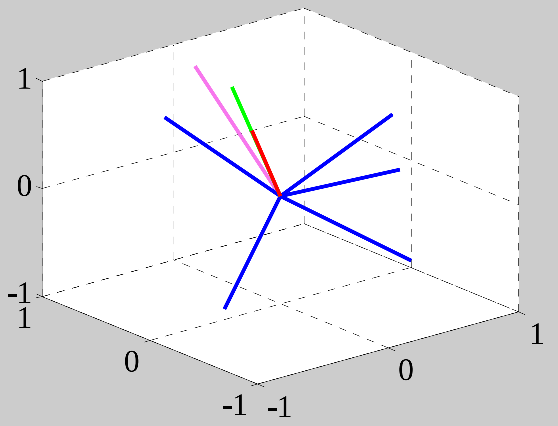
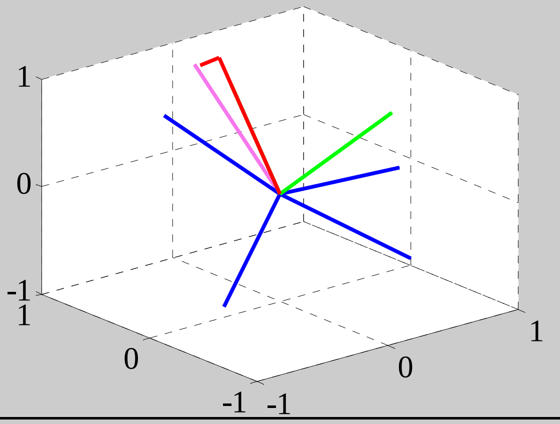
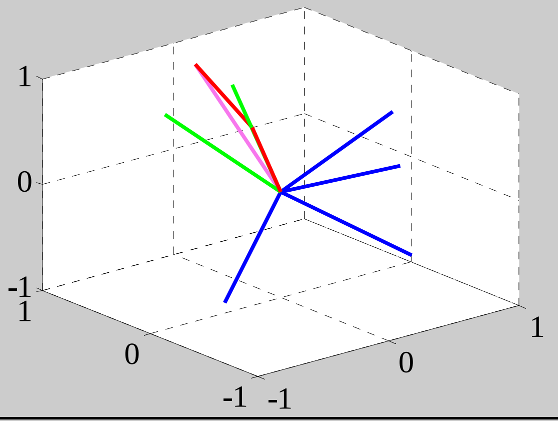
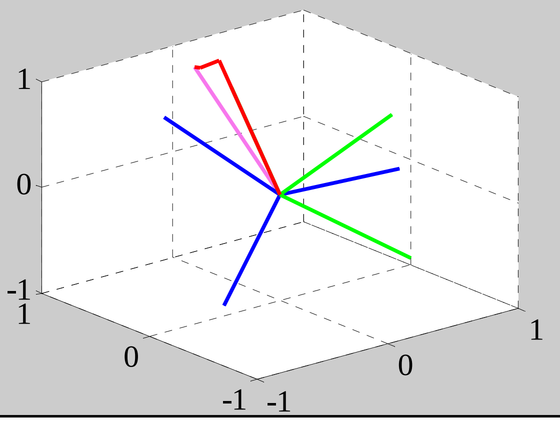
OMP	LARS
iteration 1 	iteration 1 
iteration 2 	iteration 2 
iteration 3 	
$MSE = 0.0384$	$MSE = 2.7785 \times 10^{-31}$
🟦 : Inactive columns of the matrix A	
🟩 : Active columns of the matrix A	
🟪 : Measurement vector y	
🟥 : Approximation change $(A\Delta x)$	

Figure 5.14: The approximation change $(A\Delta x)$ at each iteration of OMP and LARS in Example 5.2, where $\mu = 0.8885$, and the approximation change $= A \times \Delta x$

Chapter 6

Conclusion and Future Work

In this thesis, we developed a comprehensive comparison between two popular algorithms (OMP and LARS) that have been widely used for finding a sparse solution of an underdetermined linear system. We provided a thorough description of both algorithms. OMP attempts to find an approximate solution for the ℓ_0-norm minimization problem, while LARS solves the ℓ_1-norm minimization problem. Although OMP and LARS solve different minimization problems, they both depend on an underlying greedy framework. They start from an all-zero solution, and then iteratively construct a sparse solution based on correlation between columns of the matrix A and the current residual vector. They converge to the final solution when norm of the current residual vector approaches zero. We showed that both OMP and LARS have almost similar algorithmic steps after reformulating the corresponding analytical expressions. Nevertheless, they are different in maintaining the active set and updating the solution vector. To maintain the active set, LARS adds or drops columns to/from the active set, while OMP always adds columns to the active set and never removes them from the set. Moreover, OMP and LARS adopt different ways to update the solution vector. At each iteration, OMP updates the solution coefficients to the largest possible value that makes the residual vector is orthogonal to all active columns. In contrast, LARS updates the solution coefficients to the smallest possible value that forces a column from the inactive set to join the active set, or drops a column from the active set.

In addition, we analyzed the performance of OMP and LARS in terms of convergence time and accuracy of the final solution. In general, OMP converges to the final solution in less number of iterations than LARS because LARS may drop columns from the active set at some iterations, while OMP does not. Equally important, LARS computes the solution update vector (the vector Δx) which is used to update the solution vector x, in many steps, while OMP computes it in one step. Therefore, LARS executes many additional steps that OMP does not have. As a result, OMP requires less time than LARS to converge to the final solution.

However, in some scenarios especially when columns of the matrix A are highly correlated, LARS can successfully reconstruct the sparse vector, while OMP fails. The reason behind this is that OMP does not select columns that are highly correlated with columns that have already been in the active set, even though these columns are important to recover the sparse vector. In contrast, LARS does not face this problem.

With respect to future work, we recommend to study the impact of mutual coherence of the matrix A on the performance of OMP and LARS for different measurement size m and different sparsity k. In addition, we believe that there is a good room to improve the implementations of OMP and LARS, and propose a new algorithm that combines benefits of OMP and LARS.

APPENDICES

Appendix A

In this appendix, we formulate our derivation of penalized least square problem expressed by Equation (4.2), which can be exploited to compute active entries of the solution vector in the basic LARS algorithm that was stated in Section 3.1. Indeed, Equation (4.2) is used to compare LARS with OMP in [34]. We start the derivation by recalling Equation (3.6) that computes the updated direction of LARS:

$$A_I^T A_I d_t(I) = sign\big(c_t(I)\big) \tag{A.1}$$

where I is the active set at the iteration t.

Multiply Equation (A.1) by the step size γ_t which is a scalar:

$$A_I^T A_I \, \gamma_t d_t(I) = \gamma_t \, sign\big(c_t(I)\big) \tag{A.2}$$

From Equation (3.10), we obtain:-

$$\gamma_t d_t(I) = x_t(I) - \, x_{t-1}(I) \tag{A.3}$$

Substituting (A.3) into (A.2):

$$A_I^T A_I (x_t(I) - \, x_{t-1}(I)) = \gamma_t \, sign\big(c_t(I)\big)$$

$$A_I^T A_I \, x_t(I) = A_I^T A_I \, x_{t-1}(I) + \gamma_t \, sign\big(c_t(I)\big) \tag{A.4}$$

Recall the residual vector r at iteration $t - 1$ is computed by the following equation:

$$r_{t-1} = y - Ax_{t-1} \tag{A.5}$$

Because only active entries of x_{t-1} are nonzeros, we can rewrite (A.5) as follows:

$$r_{t-1} = y - A_{\bar{I}} \, x_{t-1}(\bar{I}) \tag{A.6}$$

where $\bar{I}$ is the active set at iteration $t - 1$.

As OMP, the basic LARS always adds columns to the active set and never removes them from the set. Therefore, the active set I at iteration t has one column that is not included in the active set $\bar{I}$ of the previous iteration $(t-1)$. The coefficient of x_{t-1} that corresponded to this column is zero. Hence, we obtain:

$$A_{\bar{I}}\, x_{t-1}(\bar{I}) = A_I x_{t-1}(I) \tag{A.7}$$

By substituting (A.7) into (A.6), we obtain:

$$A_I\, x_{t-1}(I) = y - r_{t-1} \tag{A.8}$$

Using (A.8) in (A.4):

$$A_I^T A_I\, x_t(I) = A_I^T(y - r_{t-1}) + \gamma_t\, sign\big(c_t(I)\big)$$

$$A_I^T A_I\, x_t(I) = A_I^T y - A_I^T r_{t-1} + \gamma_t\, sign\big(c_t(I)\big) \tag{A.9}$$

The active entries of the correlation vector are computed via: $c_t(I) = A_I^T r_{t-1}$. Hence we can rewrite Equation (A.9) as follows:

$$A_I^T A_I\, x_t(I) = A_I^T y - c_t(I) + \gamma_t\, sign\big(c_t(I)\big) \tag{A.10}$$

Substituting $c_t(I)$ of (3.4) into (A.10):

$$A_I^T A_I\, x_t(I) = A_I^T y - \lambda_t\, sign\big(c_t(I)\big) + \gamma_t\, sign\big(c_t(I)\big)$$

$$A_I^T A_I\, x_t(I) = A_I^T y - (\lambda_t - \gamma_t)\, sign\big(c_t(I)\big) \tag{A.11}$$

Recall Equation (3.20): $\lambda_{t+1} = \lambda_t - \gamma_t$, we can rewrite (A.11) as follows:

$$A_I^T A_I\, x_t(I) = A_I^T y - \lambda_{t+1}\, sign\big(c_t(I)\big)$$

This leads to the penalized least square problem expressed by Equation (4.2).

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Emmanuel J. Candès, Justin Romberg, and Terence Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489 - 509, February 2006.

[2] David L. Donoho, "Compressed Sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, April 2006.

[3] Emmanuel J. Candès and Terence Tao, "Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?," *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406 - 5425, December 2006.

[4] Emamnuel J. Candès, "Compressive sampling," *In Proceedings of the International Congress of Mathematicians, Madrid, Spain*, August 2006.

[5] Yaakov Tsaig and David L. Donoho, "Extensions of compressed sensing," *Elsevier Signal Processing*, vol. 86, no. 3, pp. 549-571, March 2006.

[6] Emmanuel J. Candes and Terence Tao, "Decoding by Linear Programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203-4215, December 2005.

[7] Mark Rudelson and Roman Vershynin, "Geometric approach to error-correcting codes and reconstruction of signals," *InternationalMathematics Research Notices*, vol. 2005, no. 64, pp. 4019-4041, December 2005.

[8] Alexander Vardy, "Algorithmic complexity in coding theory and the minimum distance problem," in *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of Computing*, New York, NY, USA, 1997, pp. 92–109.

[9] Albert Tarantola, *Inverse problem theory and methods for model parameter estimation*. Philadelphia, PA , USA: Society for Industrial and Applied Mathematics, 2005.

[10] Emmanuel J. Candès, Justin Romberg, and Terence Tao, "Quantitative Robust Uncertainty Principles and Optimally Sparse Decompositions," *Foundations of Computational Mathematics*, vol. 6, no. 2, pp. 227-254, 2006.

[11] Emmanuel J. Candès, Justin Romberg, and Terence Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207-1223, August 2006.

[12] Michael Elad, *Sparse and Redundant Representations From Theory to Applications in Signal and Image Processing*, 1st ed. New York, NY, USA: Springer, 2010.

[13] B. K. Natarajan, "Sparse Approximate Solutions to Linear Systems," *SIAM Journal on Computing*, vol. 24, no. 2, pp. 227-234, 1995.

[14] Scott S. Chen, David L. Donoho, and Michael A. Saunders, "Atomic Decomposition by Basis Pursuit," *SIAM Journal on Scientific Computing* , vol. 20, no. 33-61 , 1998.

[15] David L. Donoho, "For most large underdetermined systems of linear equations the minimal L1-norm solution is also the sparsest solution," *Communications on Pure and Applied Mathematics*, vol. 59, no. 6, pp. 797–829, June 2006.

[16] Jean-Jacques FUCHS, "Recovery of exact sparse representations in the presence of bounded noise ," *IEEE Transactions on Information Theory*, vol. 51, no. 10, pp. 3601 - 3608 , October 2005.

[17] Yaakov Tsaig and David L. Donoho, "Breakdown of equivalence between the minimal L1-norm solution and the sparsest solution," *Elsevier Signal Processing*, vol. 86, no. 3, pp. 533–548, March 2006.

[18] David L. Donoho and Philip B. Stark, "Uncertainty Principles and Signal Recovery," *SIAM Journal on Applied Mathematics*, vol. 49, no. 3, pp. 906-931, 1989.

[19] David L. Donoho and Xiaoming Huo, "Uncertainty principles and ideal atomic decomposition," *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2845 - 2862, November 2001.

[20] Michael Elad and Alfred M. Bruckstein, "A generalized uncertainty principle and sparse representation in pairs of bases ," *IEEE Transactions on Information Theory*, vol. 48, no. 9, pp. 2558 - 2567, September 2002.

[21] Rémi Gribonval and Morten Nielsen, "Sparse representations in unions of bases ," *IEEE Transactions on Information Theory*, vol. 49, no. 12, pp. 3320 - 3325, December 2003.

[22] Stephen Boyd and Lieven Vandenberghe, *Convex Optimization.*: Cambridge University Press, 2004.

[23] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms*, 3rd ed.: The MIT Press, 2009.

[24] Stephane G. Mallat and Zhifeng Zhang, "Matching Pursuits With Time-Frequency Dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397 - 3415 , December 1993.

[25] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *The Twenty-Seventh*

Asilomar Conference on Signals, Systems and Computers, November 1993.

[26] Geoffrey Davis, Stephana Mallat, and Marco Avellaneda, "Adaptive Greedy Approximations," *Constr. Approx*, vol. 13, pp. 57-98, 1997.

[27] Joel A. Tropp, "Greed is Good: Algorithmic Results for Sparse Approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231-2242, October 2004.

[28] David L. Donoho, Yaakov Tsaig, Iddo Drori, and Jean-Luc Starck, "Sparse solution of underdetermined linear equations by Stagewise Orthogonal Matching Pursuit," *Preprint*, 2006.

[29] Deanna Needell and Roman Vershynin, "Signal Recovery From Incomplete and Inaccurate Measurements Via Regularized Orthogonal Matching Pursuit," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 310-316, April 2010.

[30] Gagan Rath and Christine Guillemot, "Sparse approximation with an orthogonal complementary matching pursuit algorithm," *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP* , pp. 3325 - 3328, April 2009.

[31] Thomas Blumensath and Mike E. Davies, "Gradient Pursuits ," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2370 - 2382 , June 2008.

[32] D. Needell and J.A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301-321, May 2009.

[33] Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani, "Least Angle Regression," *The Annals of Statistics*, vol. 32, no. 2, pp. 407-451 , April 2004.

[34] David L. Donoho and Yaakov Tsaig, "Fast Solution of L1- norm Minimization Problems When the Solution May be Sparse," *IEEE Transactions on information Theory*, vol. 54, no. 11, pp. 4789-4812, November 2008.

[35] Robert Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society*, vol. 58, no. 1, pp. 267–288, 1996.

[36] Joel A. Tropp and Anna C. Gilbert, "Signal recovery from Random measurements via Orthogonal Matching Pursuit ," *IEEE Transactions on information Theory*, vol. 53, no. 12, pp. 4655–4666, December 2007.

[37] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The elements of statistical learning*, 2nd ed. New York, NY, USA: Springer, 2009.

[38] M. Elad, J.-L. Starck, P. Querre, and D. L. Donoho, "Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA)," *Applied and Computational Harmonic Analysis*, vol. 19, no. 3, pp. 340–358, November 2005.

[39] Jean-Luc Starck, Michael Elad, and David L. Donoho, "Image Decomposition: Separation of Texture from Piecewise Smooth Content," in *SPIE annual meeting*, San Diego, CA, USA, 2003.

[40] Jean-Luc Starck, Michael Elad, and David L. Donoho, "Redundant Multiscale Transforms and Their Application for Morphological Component Separation," *Advances in Imaging and Electron Physics*, vol. 132, pp. 287–348, 2004.

[41] N. Ahmed, T. Natarajan, K.R. Rao, "Discrete Cosine Transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90-93, January 1974.

[42] Behrooz Parhami, *Introduction to parallel processing*. New York, NY, USA: Plenum Press, 1999.

[43] Julien Mairal. SPArse Modeling Software. [Online]. http://www.di.ens.fr/willow/SPAMS/