Addis Ababa University
Addis Ababa Institute of Technology
School of Electrical & Computer Engineering

Image Deblurring with Compressive Sensing

A Thesis submitted to the School of Graduate Studies of Addis Ababa Institute of Technology in partial fulfillment of the Degree of Master of Science in Electrical and Computer Engineering

By: Rahel Berhanu

Advisor: Dr. Fitsum Assamnew

August 2019

Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical & Computer Engineering

Image Deblurring with Compressive Sensing

By: Rahel Berhanu

Approval by Board of Examiners

Dr. Yalemzewd Negash

Dean, School of Electrical and Computer Engineering _____

Signature

Dr. Fitsum Assamnew

Advisor _____

Signature

_____ _____

Internal Examiner **Signature**

_____ _____

External Examiner **Signature**

DECLARATION

I, the undersigned, certify that this thesis research titled Image Deblurring with Compressive Sensing is a presentation of my original work and has not been presented for a degree in any other university. Wherever contributions of others are involved, every effort has been made to indicate clearly with due reference to the literature.

Rahel Berhanu _____

Name Signature

Date of Submission: August, 2019

Place: Addis Ababa

This thesis has been submitted for examination with my approval as a university advisor.

Dr. Fitsum Assamnew

Advisor's Name _____

Signature

Abstract

Compressive sensing is a technique which enables recovery of signals that are represented by an underdetermined system of equations. Such a recovery of an original signal is made possible if the samples are represented in a sparse manner provided an appropriate measuring matrix is used for the modeled system. Blurred images are examples of signals that are sparse especially in transform domains. Different researches have been done to show the possibility of recovering blurred images that use sparse representation of transform domains by applying compressive sensing. In this thesis, however, a model has been used that doesn't require transforming into other domains. In addition, a box-wise approach has been used that derives the underdetermined system matrix from 7x7 segmented boxes of the blurred image. Then compressive sensing algorithms are used to recover the whole image iteratively. This method is shown to have a much better computational complexity, for example, than the traditional Lucy-Richardson deblurring but it has limitations due to approximations used in the 7x7 boxes during modeling. Thus, with this improved computational complexity, the study provides an initial platform to deblur images using box-wise method and compressive sensing theories.

Dedication

This thesis work is dedicated to the Lord Jesus Christ, God the Father and the Holy Spirit.

Acknowledgement

First and foremost, I would like to thank the Almighty God. Without His enormous help and mercy, I wouldn't have been able to begin or finish this study.

I would like to thank my advisor Dr. Fistum Assamnew whose advice and support has been indispensable throughout this thesis work.

My heartfelt gratitude goes to my husband and son whose precious love has given me great drive and also to my mother, brother and sister whose constant encouragement, love and support gave me great strength to finish this work.

Contents

Chapter 1 Introduction

1.1 Background

Image processing is a field which finds various applications in everyday life. Image processing can be classified as digital or analog. In analog image processing, the image is represented by two dimensional analog signals and the processing is done by manipulating these signals. While in digital signal processing the analysis is done using computer algorithms on the digitized image (1). Due to its various advantages such as faster and cheaper processing, easier fixing and retouching and so on, digital image processing is commonly used currently.

There are different types operations that can be done in digital image processing including image editing, filtering and image restoration. In filtering, images are blurred or sharpened using digital filters (2). While image editing comprises the process of altering images such as changing colors, red eye removal, zooming, scaling, rotating, up to the more complex tasks of image layer formation and noise reduction. On the other hand, image restoration is the process of recovering an original image from a noisy or blurred one (3). One part of image restoration is image deblurring in which the blur is removed from a corrupted image by different techniques. And image deblurring is the focus of this thesis research.

In image deblurring, it is desired to have a restored image that is very close to the original one. It is also desired to attain such an output with less computational resources. Presently, there exist different types of deblurring techniques. The Lucy-Richardson algorithm, neural network approach, deblurring with Noisy Image pairs and deblurring with Handling Outliers are few of them (4).

During image restoration of blurred images, various types of blurs can be encountered including zooming blur, out-of-focus blur, motion blur and Gaussian blur. As the name indicates, zooming blur occurs when an image is either zoomed in or out. While out-of-focus blur happens if the camera is out of focus during image capturing. On the other hand, a motion blur can come about while taking the picture of a moving object. Gaussian blur occurs in images taken in astronomy or medicine such as MRI. When a Gaussian blur is modeled mathematically, it is the outcome of manipulating an image using a Gaussian function.

Compressive sensing is a method which allows finding solution of equations which are underdetermined (5). Suppose the deblurring problem involves a system of equations of size n. Compressive sensing theorems promise reconstruction of the image using m set of equations where $m<<n$ provided that a certain number of preconditions are fulfilled.

There are different kinds of reconstruction algorithms that are used in compressive sensing and these algorithms mainly use nonlinear optimization techniques. For reconstruction to be effective in compressive sensing, sparsity is one of the preconditions (6). That means the sampled signal

must be represented using few non-zero coefficients. Due to this, compressive sensing has potentially become suitable for the recovery of sparse signals (7). This application of compressive sensing includes restoring blurred images because blurred images can be presented in sparse manner in various domains. As a result, recovery of images using compressive sensing has become a recent research area of interest.

Image deblurring can be divided in two categories based on the knowledge of the kernel involved. A kernel is a mathematical structure that is used to model a certain blur. When the deblurring is done without the knowledge of its kernel, it is called blind image deblurring. The other is when the deblurring is done with an identified kernel known as non-blind image deblurring.

This thesis research investigates the possibility of recovering an image blurred with a known Gaussian kernel using compressive sensing techniques. Apart from examining whether compressive sensing theories can be applied to image deblurring, the study assesses whether the given method is efficient in terms of storage and computation time as compared with other deblurring algorithms.

1.2 Problem Statement

As mentioned earlier, images could be blurred due to motion, lack of focus or the effect of zooming. Such blurred images occur in military where photographs may be taken by radars. They may also occur in medical images such as MRI or in astronomy where images are taken by telescopes and satellites. Obviously, these blurs are common encounters in everyday photography taken by an ordinary person or a professional camera man. Whether the blur be of any kind, it is normally required to get rid of it and that is when the deblurring problem arises.

There are various kinds of existing deblurring algorithms. For example, the Lucy-Richardson algorithm is one of them. This algorithm uses an iterative method to reconstruct a blurred image with a known point spread function where point spread function is equivalent to impulse response of a focused optical system (8) (9). The Lucy-Richardson algorithm is commonly used in astronomy. But its efficiency is regarded as being low and various researches are done to come about with its further modifications (10) (11) (12). There are other techniques of deblurring which give better result than the Lucy-Richardson algorithm. A survey done in a certain research compared algorithms like the Lucy-Richardson, deblurring with neural network, deblurring with handling outliers and deblurring with Adaptive Sparse Domain Selection. In the result, the deblurred images were found to have PSNR (Peak Signal to Noise Ratio) that ranges from 21 up to 30 which were regarded as efficient up to very efficient (4).

Although, there are already existing deblurring algorithms which are considered as very efficient, researches continue to be done to come up with even better PSNR(Peak Signal to Noise Ratio). And that is the reason why this research has been conducted to investigate the possibility of deblurring images with compressive sensing theory.

It has been stated that there are different kinds of blurs that occur in images. Out of these blurs, Gaussian blur has been selected to be the focus of this study. When expressed mathematically, Gaussian blur is similar to convolving an image with a Gaussian function. As will be discussed later, this convolution involves manipulating the image with a kernel which is a matrix like

structure. Gaussian blur appears in different occasions. For example, when astronomic images are affected by atmospheric turbulence, they are usually blurred in a Gaussian fashion (13). Gaussian blur also occurs when there is statistical light scatter during image capture and when sampling is done by receptive fields having Gaussian profile. In medical imaging, Gaussian blur causes decreased acuity in the presence of cataracts leading to less accurate diagnosis (14). Because of such common occurrence of Gaussian blur in these essential areas, this thesis deals with the deblurring of Gaussian blur using compressive sensing techniques.

1.3 Objective

The general objective of this research is to investigate the possibility of deblurring images using compressive sensing techniques.

1.3.1 Specific Objectives

The research has the following specific objectives.

I. To examine the possibility of removing a Gaussian blur using a compressive sensing algorithm.
II. To determine whether the applied method using a compressive sensing algorithm is more efficient than other existing methods.

1.4 Methodology

The methodology used in this thesis research is the deductive approach. It is known that in such an approach, a given theory is proven to be true for a certain application. Compressing sensing theories provide methods for finding the solution of an underdetermined system of equations efficiently. Accordingly, the hypothesis in this thesis is stated below.

"Given a blurred image with a known Gaussian kernel and an appropriate model designed satisfying the preconditions of compressive sensing, the blurred image can be reconstructed with a good PSNR and better computational efficiency."

Thus, the course followed in the thesis includes developing a proper model for the deblurring process with which compressive sensing can be applied. After that, a selection of compressive algorithms is done by which the deblurring of images is performed. Finally, the deblurring of images is carried out and its efficiency is evaluated based on different parameters.

Phases	**Actions**
I	Develop a suitable model with which compressing sensing can be applied
II	Selecting an algorithm from existing compressive sensing algorithms
III	Performing the deblurring of images and evaluating its efficiency

Table 1.1 Phases taken during the research

1.5 Significance

Image deblurring has important applications including restoring blurred shots in everyday life or in the field of multi media. Deblurring of images serves great purpose in the fields of astronomy, medicine, criminal investigation and military. Compressive sensing is a recently developed technique in which different researches are being undertaken. This study which is done to investigate the deblurring of images using compressive sensing will serve as input to such researches being done. Especially, the method followed in this thesis that can potentially minimize the computational complexity of the deblurring can serve as a starting point for researches that are carried out in a similar topic.

1.6 Organization of the Research Report

This research report consists of five chapters. In the first chapter an introduction is given about the thesis research. The second chapter gives an overview of compressive sensing including definition, existing algorithms and related works. The third chapter details the modeling process used in this thesis in order to apply a compressive sensing algorithm. The fourth chapter includes the results obtained by applying the compressive sensing algorithm and their comparison with other deblurring methods. Finally, conclusion and recommendation are given in the fifth chaper.

Chapter 2 Compressive Sensing Overview

2.1 Compressive Sensing

Compressive sensing is a theory first developed by David L. Donoho (5) and Candes, Romberg and Tao (15) in 2006 that can be used to solve a system of equations which is underdetermined. A system of equations is called underdetermined when it involves n number of unknowns with m equations where $m<n$. In such situations, it is not possible to find the solution by using direct methods like Gaussian elimination.

Starting from the first development of compressing sensing in 2006, various algorithms have emerged to enable to solve such an underdetermined system of equations. These algorithms have preconditions that need to be met for successful recovery of the solution. The first precondition is that the system of equations should be represented in sparse manner (16). The other common precondition for faithful recovery is incoherence which requires the matrix which represents the underdetermined system of equations to have low coherence (17). The coherence of a matrix is defined as the absolute value of the maximum cross-correlations between its columns (18).

Let the underdetermined system of equations be represented by $\phi(mxn)$ which is obtained by multiplying some original set of equations, $\psi(nxn)$, by a measuring matrix M(mxn) where $m<<n$. And let the data to be recovered be a vector X of length n. Then the system of equations which is underdetermined can be represented as (5):

$$Y = M\psi X \,, \qquad [1]$$

where ψ: nxn matrix

M: mxn matrix

X: n sized vector to be restored

Y: $mx1$ known vector called also observation vector

$$Y = \phi X, \qquad [2]$$

where $\phi = M\psi$: mxn matrix

For successful reconstruction to take place by compressive sensing, the matrix ϕ which is the product ψ and M should be sparse and have low coherence (16) (17).

2.1.1 Restricted Isometry Property (RIP)

In compressive sensing, a parameter known as Restricted Isometric Property(RIP) has been defined that must be satisfied to ensure the faithful recovery of signals. The importance of this parameter relies on the fact that low coherence of the ϕ matrix is closely related to it (17).

The Restricted Isometric Property of the matrix, ϕ, given by $\phi = M\psi$, is defined as follows. Assume a matrix $\phi_{mxn,}$ and an integer *s* such that $1 \leq s \leq n$. If there exists a constant $\delta_s \in (0,1)$ such that, for every *m x s* submatrix ϕ_s of $\phi_,$ and for every vector z of dimension *s* (6),

$$(1 - \delta_s)\|z\|_2^2 \leq \|\phi_s z\|_2^2 \leq (1 + \delta_s)\|z\|_2^2 \qquad [3]$$

holds true, then the matrix ϕ satisfies the *s*-restricted isometry property with the restricted isometry constant δ_s.

Most compressive sensing algorithms specify that the RIP value of the matrix ϕ to be less than a certain constant so that a successful reconstruction takes place. For example, Cande (17) specifies $\delta_{2s} < \sqrt{2} - 1$ for successful recovery in compressive sensing.

2.1.2 Sparsity

A vector or a matrix is said to be sparse if it consists of mainly zero elements. A vector or a matrix is said to be s-sparse if it has utmost s non-zero elements (19). Sparsity enables to bring about efficient solutions in compressive sensing and different algorithms depend on it to recover a signal (16) (20).

In the work by Donoho and Tanner (2006) (21), it is stated that compressive sensing algorithms can recover most sparse signal if s is given by:

$$s = \frac{m}{2 \log n} \qquad [4]$$

2.1.3 Incoherence

In compressive sensing, it is important that the measurement matrix M is selected in a way such that it has the lowest possible coherence with ψ (18).

The mutual coherence, μ, for the matrix $\phi = M\psi$ which needs to be minimized is described by the expression given below (17).

$$\mu(\phi) = \frac{max}{i \neq j, 1 \leq i,j \leq n} \left\{ \frac{|\phi_i^T \phi_j|}{\|\phi_i\| . \|\phi_j\|} \right\} \qquad [5]$$

where ϕ_i is the i[th] column of ϕ.

In the work by Tropp (22) (23), it is stated that the coherence value, μ, need to satisfy

$$s < \frac{1}{2}(\mu^{-1} + 1) \qquad [6]$$

for an accurate reconstruction to take place in the algorithms Orthogonal Matching Pursuit and Basic Pursuit.

2.2 Reconstruction Algorithms

In compressive sensing, there are different categories of reconstruction algorithms. The Matching Pursuit and Iterative Thresholding algorithms have been summarized as follows.

2.2.1 Matching Pursuit

This class of algorithms tries to represent a signal by linear expansion functions that form a dictionary (24). Then the Matching Pursuit algorithm optimally selects dictionary elements that can best approximate the signal. The Orthogonal Matching Pursuit (OMP) (25) and Compressive Sampling Matching Pursuit (CoSaMP) (26) are examples of Matching Pursuit algorithms.

2.2.2 Iterative Thresholding Algorithms

Compressive sensing algorithms in this category try to recover a signal iteratively. And they use a thresholding function $\mathbf{H}_s(\mathbf{x})$ at each iteration to set components of a vector X which are less than some number ε to zero and leave the rest of the components untouched. Iterative thresholding algorithms include the Iterative Hard thresholding algorithms and Iterative Soft thresholding algorithms.

2.2.2.1 Iterative Hard thresholding algorithm

T. Blumensath and M. Davis (27), introduced iterative hard thresholding algorithm as:

$$X^i = H_s\left(X^{i-1} + \emptyset^T\left(Y - \emptyset X^{i-1}\right)\right), \qquad [7]$$

where $\mathbf{H}_s(\mathbf{x}) = \begin{cases} x_i, & x_i \geq \varepsilon \\ 0, & x_i < \varepsilon \end{cases}$.

ε is the s-largest element of X while *s* is the sparsity of X that is predefined. It is possible also that ε can be selected randomly whenever there is no unique set of *s* number of elements which are the largest in the vector X. The iterative hard thresholding algorithm is proven to converge given the condition that the second norm of the matrix ϕ, $\|\phi\|_2$, is less than one (27).

Below is the complete listing of the hard iterative thresholding algorithm.

Input

- s the sparsity of X

-y ε R^m *and* ϕ ε R^{mxn}

Output:

- *X' such that y =* $\phi x'$
  1. $X^{(0)} = 0$
  2. *for i=1,…do*
  3. $X^{(i)} = Hs(X^{(i-1)} + \phi^T(y - \phi X^{(i-1)}))$
  4. *end for*
  5. $X' = X^{(i)}$

Theoretical maximum number of Iterations

T. Blumensath and M. Davis (28) derived the theoretical maximum number of iteration required to solve the compressive sensing equation, $Y = \phi X + e$. Their theory states the following.

Suppose Y is a noisy observation given by,

$$Y = \phi X + e, \qquad [8]$$

where e represents the noise.

Let X^s be the s-sparse approximation of Y for which $\|X - X^s\|_2$ is minimal. If ϕ has modified restricted isometry property (RIP) $\beta_{3s} < 1/8$ in which $\beta_s = 1 - \frac{1-\delta_s}{1+\delta_s}$, then an approximation X^s can be recovered using Iterative Hard Thresholding satisfying,

$$\left\|X - X^k\right\|_2 \leq 2^{-k}\|X^s\|_2 + 5\varepsilon_s, \qquad [9]$$

where

$$\varepsilon_s = \|X - X^s\|_2 + \frac{1}{\sqrt{s}}\|X - X^s\|_1 + \|e\|_2 \qquad [10]$$

In the algorithm a maximum of k^* iterations is defined which equals to:

$$k^* = \left\lceil \log_2\left(\frac{\|X^s\|_2}{\varepsilon_s}\right)\right\rceil \qquad [11]$$

And after a maximum of k^* iterations, the Iterative Hard Thresholding algorithm can approximate X with accuracy given by:

$$\left\|X - X^{k*}\right\|_2 \leq 6\left[\|X - X^s\|_2 + \frac{1}{\sqrt{s}}\|X - X^s\|_1 + \|e\|_2\right] \qquad [12]$$

The computational complexity and memory requirement of Iterative Hard Thresholding algorithm per iteration is found to be O(*mn*) (28).

2.2.2.2 Iterative Soft thresholding algorithm

Soft thresholding uses similar algorithm to hard thresholding except that it uses a modified thresholding function. The thresholding function in iterative soft thresholding relaxes the thresholding function by an amount τ as shown below. The soft thresholding algorithm uses equation [13] to iteratively find the solution for an underdetermined system of equations, $Y = \phi X + e$ (29).

$$X^i = S_\tau\left(X^{i-1} + \emptyset^T Y - \emptyset^T \emptyset X^{i-1}\right), \qquad [13]$$

Where $\mathbf{S}_\tau(\mathbf{x}) = \begin{cases} x_i - \tau, & x_i > \tau \\ 0, & |x_i| \leq \tau \\ x_i + \tau, & x_i < -\tau \end{cases}$

2.3 Application of Compressive Sensing

Compressive Sensing is finding applications in various fields which require efficiency in data collection and recovery. One example is the application of compressive sensing in the advent of single pixel camera by Rice University. In single pixel camera, a sensor is used to collect the input image as a linear combination of different pixels that sum up to a single pixel (30).

Compressive sensing is actively investigated for applications in Magnetic Resonance Imaging(MRI). MRI images exhibit sparsity in transform domains like Fourier and wavelet basis. Hence, this property is exploited to develop algorithms that improve image quality at the same time by reducing number of collected measurements (31).

Compressive sensing is also finding application in network manangement, ultra wideband systems, multimedia coding and communication, network traffic monitoring and anomaly detection and more as summarized in (32).

2.4 Related works

Image denoising based on compressive sensing was done by A. Tavakoli and A. Pourmohammad (7) in which an additive noise was used to model the compressive sensing equation.

$$Y = \phi(X + Z)$$

The authors performed compressive sensing denoising using existing algorithms namely, Orthogonal Matching Pursuit(OMP) and Iterative Hard Thresholding(IHT) in which they illustrated that IHT is faster than OMP. They also compared compressive sensing denoising with classical filters like Wiener filter, Median filter, Wavelet denoising and Gaussian filters. And the results showed that compressive sensing denoising gave the same result as some of the classical filters or fairly better result than the rest of the existing methods.

In the work by Bruno Amizic et al. (33), blind image deconvolution is performed using compressive sensing. The authors experimented to show that a blurred image is mostly compressible in the transform domain. Based on this fact, the authors proposed a new algorithm that solves a contrained optimization problem. In doing so, they extended compressive sensing algorithms for use in blind image deconvolution and the experimental results from the work shows fairly better outputs than that of existing algorithms such as CoSamp.

Blind image deblurring using compressive sensing has also been performed by J. Yu et al (34). The work exploited the fact that similar structures usually recur in a natural image. The authors also exploited the fact that a natural image exhibits multiple similar patches or structures when the image is down sampled. Thus, the authors used the down sampled version of the blurred image in order to find sparse representation of the original image. Using structural multi-similarity and sparse representation a blind motion deblurring method was developed which was shown to have 98.88% success rate.

C. Metzler et al. (35) applied compressive sensing to already existing denoising methods. The authors integrated the existing denoising framework AMP(Approximate message passing) with compressive sensing recovery. In this method, they illustrated that using Denoising AMP and

compressive sensing together gives state of the art recovery while operating ten times faster than existing denoising algorithms.

Compressive sensing image denoising is also done by W.Kang et al (36). In this work, the image was decomposed into edge and flat regions. In addition, an 8x8 measurement matrix was designed which was applied to the first three wavelet coefficients of the blurred image. Then from the existing compressive sensing algorithms, OMP (Orthogonal Matching Pursuit) was applied to construct each block in the image. Different error thresholds were used based on the block being in edge or flat region. Based on the experiments done by the authors, the proposed method gives better results than other existing methods.

Chapter 3 Applying Compressive Sensing to deblur Gaussian blur

3.1 Representing Gaussian blur by a sparse matrix using convolution

Compressive sensing allows to solve an underdetermined system of equations given the sparsity and incoherence conditions are satisfied. It is known that there are different existing methods for sparse representation of signals. One of these methods which is commonly used for sparse representation is transformation of the observed signal, in this case the blurred image, into domains such as wavelet.

In this thesis however, another approach has been used that doesn't require transforming the blurred image into another domain. As it has been stated before, the deblurring to be done in this thesis is a non-blind one with a known kernel. For this study, the kernel chosen is a two dimensional 7x7 Gaussian kernel with standard deviation $\sigma = 2$.

0.005084	0.009377	0.013539	0.015302	0.013539	0.009377	0.005084
0.009377	0.017296	0.024972	0.028224	0.024972	0.017296	0.009377
0.013539	0.024972	0.036054	0.040749	0.036054	0.024972	0.013539
0.015302	0.028224	0.040749	0.046056	0.040749	0.028224	0.015302
0.013539	0.024972	0.036054	0.040749	0.036054	0.024972	0.013539
0.009377	0.017296	0.024972	0.028224	0.024972	0.017296	0.009377
0.005084	0.009377	0.013539	0.015302	0.013539	0.009377	0.005084

Table 3.1 Two Dimensional 7x7 Gaussian kernel $\sigma = 2$

In deriving a model for the deblurring, image convolution with the above kernel is utilized. Convolution of an image involves replacing the pixels in an image with the linear combination of the neighboring pixels according to the values in a certain kernel (37). Whenever the kernel is symmetric, the convolution is done as follows.

The central cell of the kernel is placed at a pixel of the image. In doing so, the kernel will also cover the neighboring pixels. Then, the value of each kernel cell will be multiplied with the pixel value it has overlapped with and the products will be added. The sum of the products will be used to replace the value of the pixel with which the central cell of the kernel has overlapped with. This process is repeated on the other pixels of the original image until the whole image is covered.

By applying this standard procedure of convolution between the above Gaussian kernel and a certain image with width *w* and height *h* pixels, there will be a resulting *wxh* system of equations. Thus given an image blurred with this kernel, if one needs to find the original image, the following system of equations needs to be solved.

$$Y = \psi X,$$

where Y - the blurred image vector of length $s=wxh=wh$

ψ - an $s \times s$ matrix where $s= wxh=wh$.

X - the original image vector of length $s=wxh=wh$ to be restored

Basically, wxh is the size or the total no of pixels of the image. Therefore, one will have a total number of equations which is equal to the total number of pixels in the image. To solve these equations in the direct way, the inverse of the matrix ψ has to be solved which usually has computional complexity of $O(s^3)$, where $s= wh$.

As mentioned earlier, the system of equations $Y = \psi X$ is obtained by convolving each pixel of the image with the convolution kernel. In these equations, since a small fraction of the total number of pixels (7x7 or 49 neighbouring pixels) is used to replace a pixel with their linear combinations, the resulting equations are sparse mainly consisting of zeros. And the sparsity found in these equations has made it possible to apply compressive sensing algorithms without transforming into other domains.

3.2 Specification of the approach used for compressive sensing image deblurring

While it is possible to apply compressive sensing algorithms to the whole set of equations which represents the total number of pixels, in this thesis however, an alternative method has been used. In this approach, the image has been segmented into 7x7 boxes measured in pixels. Then, 49 equations were derived resulting from convolution of every pixel in each box with the 7x7 Gaussian kernel. Out of the 49 equations, about half of them will be selected by a measurement matrix M. And to these selected equations, a compressive sensing algorithm is applied to retrieve the 49 original pixels. Finally this procedure is executed at each box iteratively until the whole image is covered.

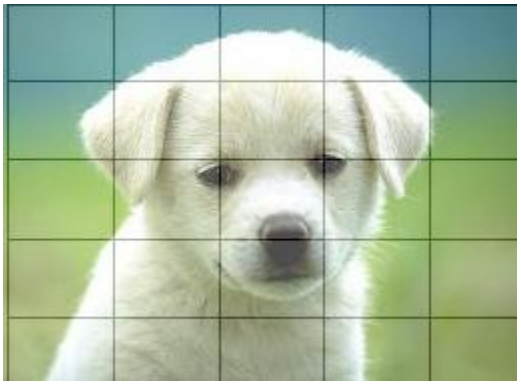


Fig.3.1 Image segmented into boxes

The reason behind following this approach was initially to simplify the process of computing the RIP of the matrix to which compressive sensing was to be applied. In the end, however, the computation of the RIP parameter was not found to be necessary. This was because the hard iterative thresholding algorithm used was found to converge since the norm-2 of the matrix

involved was less than one. Yet, the approach used has resulted in an efficient computational complexity as to be seen later.

3.3 Derivation of sparse basis matrix $\boldsymbol{\psi}$ from a Gaussian kernel

As the mentioned earlier, the method used for deblurring an image in this thesis involves segmentation of the image into 7x7 boxes.

Thus, for an image with height *h* and width *w* pixels, the maximum number of the resulting boxes horizontally and vertically along the image will be:

$$Max.no.of\ boxes\ horizontally = \frac{w}{7} + 1$$

$$Max.no.of\ boxes\ vertically = \frac{h}{7} + 1$$

If the width *w* and *h* are multiples of 7 the number of boxes will reduce to simply:

$$No.of\ boxes\ horizontally = \frac{w}{7}$$

$$No.of\ boxes\ vertically = \frac{h}{7}$$

As a result, the maximum value of the total number of boxes given by:

$$Max.value\ of\ the\ total\ no.of\ boxes = (\frac{w}{7} + 1\ )(\frac{h}{7} + 1)$$

Now, when a single 7x7 box is convolved with a 7X7 Gaussian kernel, there will be a resulting 49 set of equations. These equations constitute the ψ matrix. The ψ matrix is shown below partially where $C_{i,j}$ represents elements of the 2D Gaussian kernel shown earlier.

Appendix I contains the whole list of equations.

Fig. 3.2 Partial view of the ψ matrix

When a 7x7 box is convolved with a 7x7 Gaussian kernel only the central pixel at (3,3) is covered completely by the Gaussian kernel. Because of that, the convolution result will not be accurate at the rest of the pixels. Such effect is also observed when doing convolution at the edge of any image. For a Gaussian kernel for example, this results in blackening of the image at the edges instead of whitening it. There are different existing methods (37) that can be used to correct this such as:

1. Wrapping or mirroring an image

2. Ignoring edge pixels

3. Duplicating edge pixels. This involves considering (m, n) pixels [where n is non-positive] to have the same value as (m, 1), where m is m^{th} row in the image.

For the image convolution of 7x7 boxes at hand, ignoring the edge pixels cannot be an option because this will mean ignoring the whole image. But mirroring the image has been found a better option because symmetry is found frequently in nature and consequently in most pictures.

After applying mirroring and doing the convolution of the 49 pixels of single box, the ψ matrix will have the following set of equations which are shown partially. Appendix II contains the whole list of equations after mirroring.

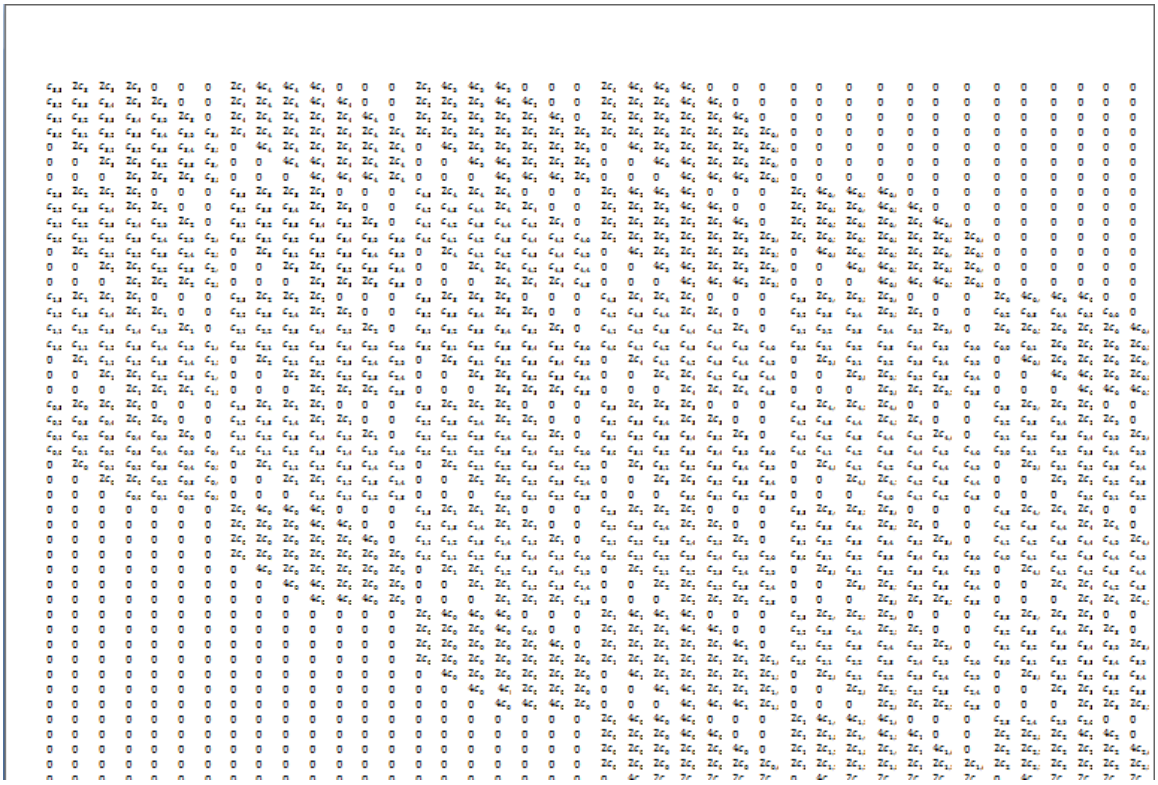


Fig.3.3 Partial view of the ψ matrix after applying mirroring

3.4 Derivation of the Measurement matrix - M

The measurement matrix is *mxn* in size where $m<n$. The task of the measurement matrix is to reduce the *n* number of equations to *m* equations and it can be of different compositions. After doing the required computations and tests, finally the measurement matrix has been designed in such a way that it selects the even numbered rows from the main equation and is shown below. This measurement matrix is found to give the lowest coherence in the matrix ϕ which in turn gives favorable results in compressive sensing.

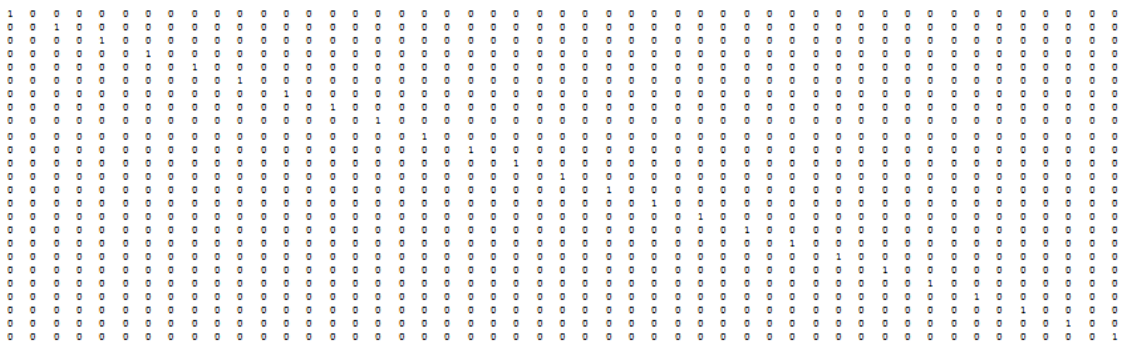


Fig.3.4 The measurement matrix M used in the compressive sensing deblurring

Accordingly, the ϕ matrix which is given by $\phi = M\psi$ becomes as shown below.

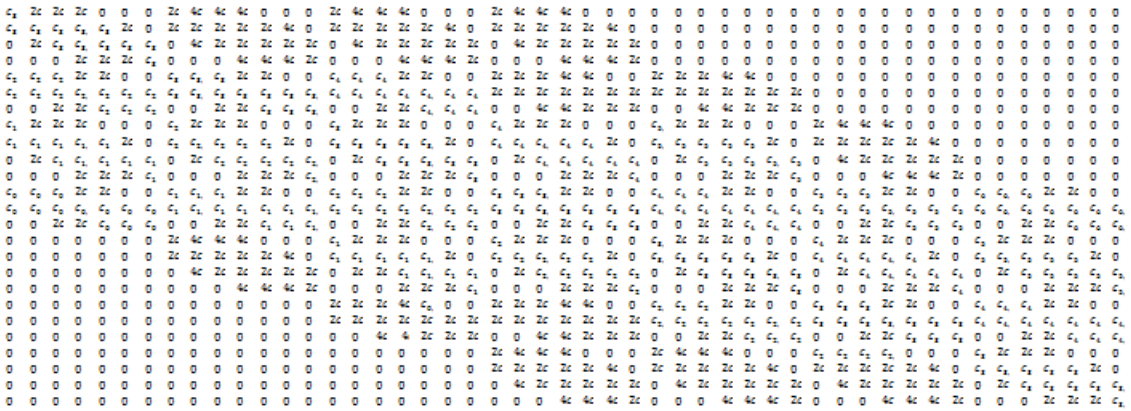


Fig.3.5 The ϕ matrix used in the compressive sensing deblurring

3.5 Selection of a reconstruction algorithm

There are different algorithms of compressed sensing that can be applied to a given problem. In this study, the ϕ matrix, which is derived from the image's convolution with the 2D-Gaussian kernel shown earlier, consists of coefficients that are all less than one. As a result, the norm-2 of the matrix becomes less than 1 which makes it suitable to apply iterative hard thresholding algorithm of compressed sensing. The iterative hard thresholding algorithm is said to converge whenever the norm-2 of the matrix ϕ is less than one (27). For the ϕ matrix displayed in Fig.3.5 the second norm is found to be:

$$\|\phi\|_2 = 0.7649 \quad \text{for 2D Gaussian kernel with standard deviation 2}$$

With this value of $\|\phi\|_2$ being less than 1, it is found suitable to use the Hard Iterative algorithm for the image deblurring.

3.6 Computation of parameters required for use in iterative hard thresholding algorithm

Before applying the Iterative Hard Thresholding algorithm, some parameters need to be predetermined. The Iterative Hard Thresholding algorithm is listed below again.

Input

- s the sparsity of X

-$y \varepsilon R^m$ and the measurement matrix $\phi \varepsilon R^{mxn}$

Output:

- *X' such that $y = \phi x'$*
  1. $X^{(0)} = 0$
  2. *for i=1,…do*
  3. $X^{(i)} = Hs(X^{(i-1)} + \phi^T(y - \phi X^{(i-1)}))$
  4. *end for*
  5. $X' = X^{(i)}$

where Hs is hard thresholding function which sets all components of the vector **x** to zero except the s largest components.

$$\mathbf{H}_s(\mathbf{x}) = \begin{cases} x_i, & x_i \geq \varepsilon \\ 0, & x_i < \varepsilon \end{cases}$$

ε is the s-largest component of X

1. *Setting m*

The integer *m* is the number of rows of the matrix ϕ. For each 7x7 box of the image 49 equations were derived making *n=49*. Compressive sensing uses underdetermined system of equation where *m<n*.

Let *m = 16*.

2. *Setting the sparsity value s*

According to equation [3], *s* is calculated as:

$$s = \frac{m}{2\log n} = \frac{16}{2\log 49} = 4.733 \cong 5$$

3. *Setting the upper bound ε of the thresholding function*

It has been indicated earlier along with equation [7] that it is possible to set this upper bound value, *ε,* randomly whenever there is no unique set of s-largest elements. The pixel values in an image range from 0-255. And most images contain much larger number of pixels than 255. As a result there usually does not exist a unique set of *s* number of elements which are the largest pixel values in an image. Because of this, *ε* has been set randomly. Thus, let *ε* be equal to the sparsity value *s* giving:

ε = 5

4. *Computing the theoretical maximum number of iterations* k^*

The theoretical maximum number of iterations hold when the matrix ϕ has modified restricted isometry property(RIP) $\beta_{3s} < 1/8$ in which $\beta_s = 1 - \frac{1-\delta_s}{1+\delta_s}$. The calculation of RIP is a computationally intensive process and this is also stated in the work by Blumnensath and Davies (28). Due to this, the Restricted Isometric, δ_s, has not been computed for the matrix ϕ. As it is mentioned in **Sec. 3.6**, the precondition for convergence of the iterative hard thresholding algorithm is satisfied with the norm-2 of the matrix ϕ being less than one. With the RIP value of the matrix ϕ being unknown, the theoretical maximum number of iterations may not hold true. Thus, for this study, the theoretical maximum number of iterations is taken as the starting number of iterations with which the iterative hard thresholding algorithm is applied.

The theoretical maximum number of iterations, k^*, can be computed using equations [10] and [11].

Given, $Y = \phi X + e$ and X' as the k^{th} approximation,

$$k = \left\lceil \log_2 \left(\frac{\|X'\|_2}{\varepsilon_s} \right) \right\rceil$$

$$\varepsilon_s = \|X - X'\|_2 + \frac{1}{\sqrt{s}} \|X - X'\|_1 + \|e\|_2$$

$\|X'\|_2$ can have different values depending on the pixel color. Here, the maximum value of $\|X'\|_2$ is computed. The maximum value that a pixel value can have is 255. X is n-dimensional with *n=49* and setting all the *n* elements to 255 gives:

$$\|X'\|_2 = \sqrt{nx\ 255^2} = \sqrt{49x\ 255^2} = 7x255 \cong 7x256 = 1{,}792$$

Assuming maximum of 1 pixel difference between the actual solution and the approximation vector:

$$\left\| X - X' \right\|_2 = \sqrt{nx1^2} = \sqrt{49} = 7$$

$$\frac{\left\| X - X' \right\|_1}{\sqrt{s}} = \frac{\sqrt{49}}{\sqrt{5}} \cong 21$$

And for this study, the noise assumed to be zero giving $\|e\|_2 = 0$.

Thus,

$\varepsilon_s = \|X - X'\|_2 + \frac{1}{\sqrt{s}} \|X - X'\|_1 + \|e\|_2 = 7 + 21 = 28$ giving,

$$k = \left\lceil \log_2 \left(\frac{\|X'\|_2}{\varepsilon_s} \right) \right\rceil = \left\lceil \log_2 \left(\frac{7x256}{28} \right) \right\rceil = 6$$

5. *Initial approximation value*

The initial approximation vector has been set to have a value of white pixel which is 255.

$X^{(0)} = 255$

This is because taking zero value for the initial approximation vector will totally blacken the output since zero stands for black color in pixel representation.

Chapter 4 Results and Discussion

4.1 Specification of the inputs used in deblurring

In the code used to test the deblurring of images using Iterative Hard Thresholding, 24 bit bitmap images have been used as inputs. Thus, the algorithm has been applied three times in each image for the R(red), G(green), B(blue) array of pixels.

The blurred images that are used as inputs in the deblurring process have been convolved box-wise. This means that when performing convolutions on the original images, the images are segmented into 7x7 pixel boxes and the convolution is done iteratively on each box separately. And the mirroring of pixels near the borders of each box has been done during convolution. This is not the natural way of performing convolution. It has been done in order to match the input blurred image with the model used for deblurring. Results for ordinarily or non-box wise convolved images will be discussed later.

4.2 Applying IHT using Initial parameters and results obtained

The initial parameters, which are computed in **sec. 3.6** to be used in the deblurring process, are displayed below.

No of rows of ϕ(m)	***Upper bound of thresholding(ε)***	***No of Iteration(k)***
16	5	6

Table 4.1 Initial Parameters

The image used is the Electric_Lines image convolved with 2D-Gaussian kernel (std=2). Both the original image and the convolved images are shown below.

Fig.4.1 The original Electric_Lines image

Fig.4.2 Box wise 2D-Gaussian convolved image

For the convolved image displayed in Fig.4.2, the deblurring using IHT with the initial parameters resulted in an image shown below.

Fig.4.3 Deblurring result for k=6, m=16

In the result, it can be seen that an acceptable outcome is not obtained. It can be observed from Fig.4.3 that each box has not been restored sufficiently and grids have been formed all over the image as a result.

To improve the deblurring, m has been varied to different values and good results were obtained for m=25 by selecting even number of rows from ψ. This is because the ϕ matrix has the lowest correlation when such a selection is done. The table below gives few samples of correlation values of ϕ for different types of measuring matrix M where $\phi = M\psi$.

Type of Measurement matrix(M)	**Value of *m***	**Correlation of ϕ**
Selects rows of ψ of which are multiples of 3	16	0.979235
Selects rows of ψ of which are even	25	0.953705
Selects rows of ψ randomly	32	0.975379

Table 4.2 Correlation values of different values of the measurement matrix

4.3 Applying IHT with Improved Parameters and Results obtained

Iterative hard thresholding is applied to the convolved Electric_Lines images again with the improved parameters shown below.

No of rows of ϕ(m)	***Upper bound of thresholding(ε)***	***No of Iteration(k)***
25	5	6

Table 4.3 Improved Parameters

The deblurring result for the Electric_Lines image using *m=25* and the other parameters unchanged is given below in Fig.4.4. In the figure, it can be seen that the 7x7 boxes are restored sufficiently. However, the restored image lacks smoothness around the electrical lines and towers. To remove these undesired effects, the iteration was increased and it was observed that the deblurred image got sharpened during that process. The number of iterations was increased successively and good results were obtained for k=45. Results for k=12 and k=45 are shown below.

Fig.4.4 Deblurring result for m=25, k=12

Fig.4.5 Deblurring result for m=25, k=45

Iterative hard thresholding has been applied to a dog and flower images and the results are shown below.

Fig.4.6 Original Flower Image

Fig.4.7 Box-wise 2D Gaussian convolved flower image (std=2)

Fig.4.8 Deblurring result for k=12

Fig.4.9 Deblurring result for k=45

Fig.4.10 Original Dog Image

Fig.4.11 Box-wise 2D Gaussian convolved dog image (std=2)

.4.12 Deblurring result for k=12

Fig.4.13 Deblurring result for k=45

4.4 MSE and PSNR of the deblurred images

Mean square error (MSE) of the deblurred images is shown below.

Image	**MSE of Blurred image(RGB)**	**No. of Iterations-*k***	**MSE of Deblurred image**	**Improvement**
Electric_Lines	(R) 374.6 (G) 343.7 (B) 341.2	6	(R) 376.9	-0.6%
			(G) 342.0	0.4%
			(B) 337.9	1%
		12	(R) 349.5	6.7%
			(G) 316.0	8%
			(B) 308.8	9.4%
		45	(R) 317.2	15.5%
			(G) 289.1	15.9%
			(B) 272.0	20.3%
Flower	(R) 175.3 (G) 128.5 (B) 175.9	6	(R) 168.4	3.9%
			(G) 130.2	-1.3%
			(B) 169.7	3.5%
		12	(R) 123.5	30%
			(G) 113.5	11.7%
			(B) 133.0	22.4%
		45	(R) 105.2	40%
			(G) 115.3	10.3%
			(B) 120.7	31.4%
Dog	(R) 199 (G) 184.7	6	(R) 212.6	-6.8%
			(G) 196.6	-6.17%
			(B) 168.1	3.5%
			(R) 191.1	4%

	(B) 174.2	12	(G) 190.0	-2.9%
			(B) 133.5	23.4%
		45	(R) 242.2	-21.7%
			(G) 241.6	-30.8%
			(B) 121.6	30.2%

Table 4.4 MSE of the deblurred images

Peak Signal to noise ratio(PSNR) of the deblurred images is shown in the table below.

Image	**PSNR of Blurred image(RGB)-db**	**No. of Iterations-*k***	**PSNR of Deblurred image-db**
Electric_Lines	(R) 22.4 (G) 22.8 (B) 22.8	6	(R) 22.3
			(G) 22.8
			(B) 22.9
		12	(R) 22.7
			(G) 23.1
			(B) 23.2
		45	(R) 23.1
			(G) 23.5
			(B) 23.8
Flower	(R) 25.7 (G) 27.0 (B) 25.7	6	(R) 25.9
			(G) 27.0
			(B) 25.9
		12	(R) 27.2
			(G) 27.6
			(B) 26.9
			(R) 27.9

		45	(G) 27.5
			(B) 27.3
Dog	(R) 25.1 (G) 25.5 (B) 25.7	6	(R) 24.5
			(G) 25.2
			(B) 25.9
		12	(R) 25.3
			(G) 25.3
			(B) 26.9
		45	(R) 24.3
			(G) 24.3
			(B) 27.2

Table 4.5 PSNR of the deblurred images

The results show the Iterative hard thresholding has deblurred the images to a measurable degree. In all the images, the deblurring is negligible or negative at the theoretical maximum iteration *k=6*. As it has been stated earlier, this theoretical maximum iteration has been used as starting number of iterations since the RIP of the matrix ϕ is unknown. When the number iterations is increased to *k=12*, there is a considerable improvement in the deblurring ranging from 4%-30%, except for the green pixels in the dog image which exhibits -2.9% degradation. At iteration number *k=45*, Electric_Lines and flower images show a much better deblurring ranging from 10%-40%. At this iteration value, the dog image showed deterioration. During the test, it has been observed that the number of iterations cannot be increased from a certain upper bound beyond which the image will be degraded. Thus for the dog image this value is *k=12*.

It is to be noted that all the percentages stated work for the MSE measurement. The PSNR is measured in db which is a logarithmic scale and the changes are not much visible in it.

4.5 Computational complexity of the deblurring method and its comparison with similar approaches

In the subsequent the computational complexity analysis, it should be noted that the computational complexity refers to both the time and storage requirement.

As stated earlier the Iterative hard thresholding algorithm has computational complexity of O(*mn*) (28) per iteration. This has been derived taking into consideration that the maximum number of computation occurs when evaluating the product $\emptyset^T\emptyset$.

$$X^i = H_s\left(X^{i-1} + \emptyset^T\left(Y - \emptyset X^{i-1}\right)\right),$$

where $\mathbf{H}_s(\mathbf{x}) = \begin{cases} x_i, & x_i \geq \varepsilon \\ 0, & x_i < \varepsilon \end{cases}$

For *k* number of iterations the total computational complexity becomes O(*kmn*).

In this thesis, a deblurring method is followed in which the input image is segmented into boxes and iterative hard thresholding is applied to each box until the whole image is covered. The computational complexity includes the cost of evaluating the hard thresholding algorithms times the number of iterations. In addition, it includes the cost of evaluating ψ and ϕ matrices and also the segmenting overhead. For this method, the computational complexity is calculated as shown below.

4.5.1 Evaluating the computational complexity of the deblurring method that used box-wise approach

1. Computational complexity of the evaluating the ψ matrix

For the analysis below,

n - represents the total number of pixels in a segmented box for this case a 7x7 box making *n=49*

m - represents the number of rows in the ∅ matrix making *m=25*

k - represents the number of iterations

Evaluating the ψ matrix includes representing the convolution of a 7x7 pixel box of the image with the 7x7 2D Gaussian kernel by 49x49 matrix. This processes includes the following steps.

i. Initial representation of the convolution coefficients and the 7x7 box pixel indexes by *nxn* matrix whose elements need to be ordered further.

This first step has computational complexity of $O(n^2)$

ii. Ordering of each row of the *nxn* matrix based on pixel indexes using a sorting algorithm. This step gives the initial ψ matirx before mirroring.

The ordering or sorting algortihm used for this second step is insertion sorting. Insertion sort has computational complexity $O(n^2)$ and it is applied for each of the *n* rows of the unordered ψ matrix. The resulting total computational complexity is $O(n^3)$**.**

iii. Finally, the ψ matrix is mirrored to account for the incomplete convolution near the edges of the box. This final step has computational complexity of $O(nxn) = O(n^2)$**.**

Thus, the total computational complexity of evaluating the ψ matrix is the sum of the complexities of the above processes and it is equivalent to $O(n^3)$.

2. Computational complexity of the evaluating the ∅ matrix

The $\emptyset$ matrix is given by $\emptyset = M\psi$ where M is *mxn* and ψ is *nxn*. This matrix multiplication has $O(mxnxn) = O(mn^2)$ computational complexity.

3. Computational complexity of applying IHT on a single 7x7 box

As stated above, iterative hard thresholding, IHT, has computational complexity of *O(kmn)*. In the box-wise approach, there is a cost of segmenting the image at each iteration which increases the computational complexity by a factor of 49. Thus, the total computational complexity becomes *O(kmnSO)* where *SO = n = 49* is the segmenting overhead.

Computational complexity of applying IHT a single 7x7 box $= O(kmnSO) = O(kmn^2)$

4. Computational complexity of applying IHT on the whole image

As it has been stated earlier, the total number of boxes in a 7x7 segmented image is given by:

$$Total\ no.of\ boxes = (\frac{w}{7} + 1\ )(\frac{h}{7} + 1)\ \cong \left(\frac{w}{7}\right)(\frac{h}{7})$$

Where *w* - width of the image

h - height of the image

Computational complexity of applying IHT on the whole image = $O\left(kmn^2\left(\frac{w}{7}\right)\left(\frac{h}{7}\right)\right)$

$= O(kmn^2\left(\frac{wh}{49}\right))$

$= O(kmn(wh))$

5. The total Computational complexity of deblurring the whole image

The total computational complexity of deblurring the whole image is the sum of the computational costs stated above. And that can be approximated by the process which has the largest order of computational complexity. The largest computational complexity is obtained when applying IHT on the whole image. Thus, the total computational complexity becomes:

The total Computational complexity of deblurring the whole image = $\boldsymbol{O(kmn(wh))}$

4.5.2 Evaluating the computational complexity of a deblurring method that used non-box-wise approach

If the Iterative Hard thresholding algorithm had been used without segmenting the image into boxes, the expected computational complexity would have been as follows. This is by assuming that the $\emptyset$ matrix has norm-2 which is less than one and the iterative hard thresholding algorithm converges.

- In the case where non-box-wise method is used, the ψ matrix has dimensions *sxs* where *s = wh.*

For the analysis below

n - represents the total number of equations in the ψ matrix making *n = s = wh.*

m - represents the number of equations selected from the ψ matrix making $m = \frac{n}{2} = \frac{wh}{2}$ assuming that half of the equations are selected similar to the box-wise case.

k - represents the number of iterations as before.

1. Computational complexity of the evaluating the ψ matrix

Evaluating this computational complexity comprises the copying into, ordering and applying mirroring to the ψ matrix which has dimension *sxs* where *s=wh.* With the values of *n=s=wh,* computational complexity of these processes amounts to $O((wh)^3)$.

2. Computational complexity of the evaluating the ∅ matrix

As explained earlier this computational complexity has order of multiplying the M*(mxn)* matrix by the $\psi(nxn)$ matrix to give the ∅*(mxn)* matrix. The *m* and *n* values are given above for the non-box wise case which makes the computational complexity of this step $O\left(\frac{(wh)^3}{2}\right) = O((wh)^3)$.

3. Computational complexity of applying IHT on the whole image

Substituting the values *k, n* and *m* for the non-box wise method stated above into the computational complexity of applying IHT gives:

Computational complexity of IHT without segmenting into boxes = $O(knm) = O(k\frac{(wh)^2}{2})$

$= O(k(wh)^2)$

4. The total Computational complexity of deblurring the whole image

The total computational complexity of deblurring the whole image by using non-box wise method is the largest computational complexity of the above steps. And this largest value is obtained when evaluating the ψ matrix giving computational complexity of $O((wh)^3)$.

4.5.3 Summary of comparison of the box-wise method of deblurring with other methods

The table below summarizes the computational complexities of the box-wise and non-box wise methods of applying iterative hard thresholding to a given image. Both these methods involve deriving the matrix ψ which represents the convolution of the original image pixels. For comparison purposes the computational complexity of inverting the ψ matrix by Gauss Jordan method is included which is the direct way of finding the solution. This is by taking the assumption that the matrix ψ is invertible.

Method	Computational complexity
Box-wise deblurring using IHT	$\boldsymbol{O\big(knm(wh)\big)}$
Non-box-wise deblurring using IHT	$\boldsymbol{O((wh)^3)}$
Direct matrix inversion(Gauss-Jordan)	$\boldsymbol{O((wh)^3)}$

Table 4.6 Comparison of computation complexities of different methods

The computational complexity for the box-wise method has been analyzed below taking examples of the dog and Electrical_Lines image. The dog image has *w=259* and *h=194*. For *k=45*, *n=49* and *m=25*,

$$knm(wh) = \big((\mathbf{55,125})\boldsymbol{x}(\mathbf{50,246})\big) \cong (\mathbf{50,246})^2 = (\boldsymbol{wh})^2$$

Thus, for the dog image which is relatively small, the box-wise method has computational complexity close to $\boldsymbol{O\big((wh)^2\big)}$. This result shows that the box wise method has better computational complexity than both the non-box-wise and the direct matrix inversion methods.

Method applied on the dog image	Computational complexity
Box-wise deblurring using IHT	$\boldsymbol{O\big((wh)^2\big)}$
Non-box-wise deblurring using IHT	$\boldsymbol{O((wh)^3)}$
Direct matrix inversion(Gauss-Jordan)	$\boldsymbol{O((wh)^3)}$

Table 4.7 Comparison of computational complexities for the dog image

Taking the example of the Electric_Lines image, which is relatively a larger image, the following results are obtained. The Electric_Lines image has *w=1024* and *h=768*. For *k=45*, *n=49* and *m=25*,

$$knm(wh) = (\mathbf{55,125})\boldsymbol{x}(\mathbf{786,432}) = \frac{(\mathbf{786,432})^2}{\mathbf{14}} \cong \frac{(\boldsymbol{wh})^2}{\mathbf{14}} \cong (\boldsymbol{wh})^2$$

It can be seen that, for the Electrical_Lines image, computational complexity of the box wise method has reduced by a factor of 14 though it is still in quadratic order given by $\boldsymbol{O\big((wh)^2\big)}$. Thus, it can be concluded that the box-wise method has computational complexity of $\boldsymbol{O\big((wh)^2\big)}$ for the average sized image which makes it better than the non-box-wise counterpart or the direct matrix inversion method. It can be seen be seen from these examples that the box wise method becomes more and more efficient as the image size becomes larger.

Method applied on the dog image	Computational complexity
Box-wise deblurring using IHT	$\boldsymbol{O}\left((\boldsymbol{wh})^{\boldsymbol{2}}\right)$
Non-box-wise deblurring using IHT	$\boldsymbol{O}((\boldsymbol{wh})^{\boldsymbol{3}})$
Direct matrix inversion(Gauss-Jordan)	$\boldsymbol{O}((\boldsymbol{wh})^{\boldsymbol{3}})$

Table 4.8 Comparison of computational complexities for the Electric_Lines image

The table below gives comparison of the methods discussed in terms of computational complexity. The Lucy-Richardson algorithm is also included here for comparison which is an existing method to deblur images including Gaussian blur. It can be seen that the box-wise method has a better computational complexity than the Lucy-Richardson algorithm.

Method	Computational complexity
Box-wise deblurring using IHT	$\boldsymbol{O}\left((\boldsymbol{wh})^{\boldsymbol{2}}\right)$
Non-box-wise deblurring using IHT	$\boldsymbol{O}((\boldsymbol{wh})^{\boldsymbol{3}})$
Direct matrix inversion(Gauss-Jordan)	$\boldsymbol{O}((\boldsymbol{wh})^{\boldsymbol{3}})$
Lucy-Richardson algorithm	$\boldsymbol{O}(\boldsymbol{k}(\boldsymbol{wh})^{\boldsymbol{3}})$

Table 4.9 Comparison of computational complexities

4.6 Results of the box-wise method applied to the deblurring of normally (non-box wise) convolved images

When the box-wise method was tested to deblur normally convolved images, the results were not positive. The result of the deblurring for the Electric_Lines image, which is convolved normally with a 2-dimensional Gaussian kernel with standard deviation=2, is shown in Fig. 4.15 and Fig.4.16 for iteration value of k=12 and k=45 respectively. The deblurring result for the flower image is also shown in Fig. 4.18 & 4.19 for the normally convolved flower image.

Fig. 4.14 Electric_Lines Image convolved normally with a 2-D Gaussian kernel std=2

Fig. 4.15 Deblurring results of a normally convolved Electric_Lines image for k=12

Fig. 4.16 Deblurring results of a normally convolved Electric_Lines image for k=45

Fig. 4.17 Flower image normally convolved with 2-D Gaussian kernel std=2

Fig. 4.18 Deblurring results of a normally convolved flower image for k=12

Fig. 4.19 Deblurring results of a normally convolved flower image for k=45

The mean square error of the images after the attempted deblurring is shown in the table below.

Image	**MSE of Blurred image(RGB)**	**No. of Iterations-*k***	**MSE of Deblurred image**	**Improvement**
Electric_Lines	(R) 324 (G) 300 (B) 284	6	(R) 373	-13%
			(G) 340	-13%
			(B) 333	-17%
		12	(R) 388	-20%
			(G) 339	-13%
			(B) 322	-13%
		45	(R) 495	-53%
			(G) 421	-40%
			(B) 377	-33%
			(R) 129	-35%

Flower	(R) 175.3 (G) 128.5 (B) 175.9	6	(G) 125	-28%
			(B) 146	-51%
		12	(R) 163	-70%
			(G) 150	-53%
			(B) 176	-81%
		45	(R) 283	-195%
			(G) 242	-147%
			(B) 277	-186%

Table 4.10 MSE of the deblurring results for normally convolved images

As the results show, when normally convolved images were attempted to be deblurred with the box-wise deblurring method, the process resulted in no deblurring at any number of iterations. Rather, the images deteriorated as the number of iterations increased. The reason behind this deterioration is the mirroring used in the 7x7 boxes of the images during deriving deblurring model. Here, the box size and the kernel size used are the same resulting in only the central pixel to be convolved in the normal way without mirroring. This has resulted in too much approximation in the box-wise method so that a normally convolved image couldn't be deblurred in the given way.

Chapter 5 Conclusion and Recommedation

5.1 Conclusion

In this thesis work, the possiblility of deblurring an image with compressive sensing theories has been studied. Among, the various existing algorithms of compressive sensing, the iterative hard thresholding algorithm has been used. In the process of applying this algorithm, a box-wise method was followed initially under with the intention of simplifying the calculation of RIP parameter. When the deblurring was performed the blurred images which are inputs to the algorithm were convolved in a box-wise manner. This was done to make the inputs have a parallel makeup to the deblurring model. And this deblurring model was intended to be also applicable to non-box wise or normally convolved images.

In the results, the deblurring was successful for blurred images which were convolved box-wise. As it is indicated earlier, the theoretical maximum number of iterations was used as a starting minimum point since the RIP of the modeled system of equations was unknown. At that iteration value no or negiligible deblurring occurred. As the number of iterations was increased, the deblurring was successful giving an improvement of up to 40%. During the testing a certain upper bound existed for the number of iterations beyond which the images get deteroriated instead of being deblurred.

Although accidentally followed, this box wise method has been found to be computationally more efficient than using the non-box wise counterpart or using the direct matrix inversion method. The method also exhibited a better computational efficiency than the well known Lucy-Richardson deblurring. In addition, it is shown that this method becomes more efficient relative to the other stated methods as the size of the image increases.

However, when the box-wise deblurring method was applied to normally convolved images the results were not desirable. As It is stated earlier, when the convolution is done box-wise, mirroring is applied at those pixel points which the kernel cannot cover completely. When the box size and the kernel size are the same, in the given case 7x7, only the central pixel of the box gets convolved in a normal way and for the rest of the 48 pixel mirroring must be done. That means, only 2% of the box is convolved in the conventional way which makes the deblurring model and normally convolved images unparallel. Due to this, the box-wise method was unable to deblur normally (non-box-wise) convolved images.

5.2 Recommendation

It is seen that good results were obtained for the box-wise method used including that it is also computationally efficient. However, the deblurring method couldn't be extended for use in normally convolved images which are results of the natural way of performing convolution. There are certain things that could be adjusted in order to make use of the better computational efficiency of the box-wise deblurrring method and make it applicable to normally convolved images.

As it has been stated, in a 7x7 pixel box size, mirroring is applied to 98% of the pixels. This is clearly too much approximation. Increasing the box size will obviously decrease this approximation as the number of pixels to which mirroring is applied decreases. However, this will increase the computational complexity since the order of computations of the box-wise method is given by $O(kmn(wh)^2)$ where n is the box size. Finding an optimum design which minimizes the number of iterations and the value of m can compensate for the increase in computational complexity caused by increasing the box size. In doing these, the deblurring method could be applicable to normally convolved images with its computational complexity not being compromised.

Bibliography

1. *What is Signal?* **Chakravory, Praggnan.** 5, 2018, IEEE Signal Processing Magazine, Vol. 35, pp. 175-177.

2. **Gonzalez, Rafael C.** *Digital Image Processing.* 3. s.l. : Pearson Hall, 2008.

3. **Reeves, Stanley J.** Image Restoration: Fundamentals of Image Restoration. *mic Press Library in Signal Processing.* s.l. : Elseviear, 2014, pp. 165-192.

4. *A Survey on Various Image Deblurring Techniques.* **Singh, Dejee and Sahu, Raj Kumar.** 12, 2013, International Journal of Advanced Research in computer and communication engineering, Vol. 2.

5. *Compressive Sensing.* **Donoho, David L.** 4, 2006, IEEE Transactions on Information Theory, Vol. 52, pp. 1289-1306.

6. *Decoding by Linear Programing .* **Candes, Emmanuel J. and Tao, Terence.** 12, 2005, IEEE Transactions on Information Theory, Vol. 51, pp. 4203-4215.

7. *Image Denoising Based on Compressed Sensing.* **Tavakoli, Amin and Pourmohammad, Ali.** 2, 2012, International Journal of Computer Theory and Engineering, Vol. 4.

8. *Bayesian-Based Iterative Method of Image Restoration.* **Richardson, William H.** 1, 1972, Journal of the Optical Society of America, Vol. 62.

9. *An iterative technique for the rectification of observed distributions.* **Lucy, Leon B.** 6, 1974, Astronomical Journal, Vol. 79.

10. *Efficient deconvolution methods for astronomical imaging: Algorithms and IDL-GPU codes.* **Prato, Marko, Cavicchioli, Ricardo, Zanni, Luca, Boccacii, Patrizia and Bertero, M.** 2012, Astronomy & Astrophysics Manuscript, Vol. 539.

11. *A modified Richardson-Luccy algorithm for single image with adaptive reference maps.* **Cui, Guangmang, Feng, Huajun, Xu, Zhihai, Li, Qi and Chen Yueting.** 2014, Optics & Laser Technology, Vol. 58, pp. 100-109.

12. *Blind Image Deblurring with Modified Richardson-Lucy Deconvolution for Ringing Artifact Suppression.* **Yang, Hao Liang, Chiao, Yen Hao, Haung, Po Hao and Lai, Shang Hong.** Gwangju : s.n., 2011. Proceedings of 5th Pacific Rim conference on Advances in Image and Video Technology. Vol. 2, pp. 240-251.

13. *Deblurring Gaussian blur.* **Hummel, Robert A. and Zucker, Steven W.** 1, 1987, Computer Vision, Graphics and Image Processing, Vol. 38, pp. 66-80.

14. **Gonzalez, Rafael C., Woods, Richard E. and Eddins, Steven L.** *Digital Image Processing Using MatLab.* NJ : Pearson Prentice Hall, 2004.

15. *Robust uncertainty principle: Exact signal reconstruction from highly incomplete frequency information.* **Candes, Emmanuel J., Romberg, Justin and Tao, Terence.** 4, 2006, IEEE Transactions on Information Theory, Vol. 52, pp. 489-509.

16. *For most large underdetermined systems of linear equations the minimal 1-norm solution is also the sparsest solution.* **Donoho, David L.** 6, s.l. : Communications on Pure and Applied Mathematics, 2006, Vol. 59, pp. 797-829.

17. *The restricted isometry property and its implications for compressed sensing.* **Candes, Emmanuel J.** s.l. : Academic des Sciences, 2008.

18. *Just relax: Convex programming methods for identifying sparse signals in noise.* **Tropp, Joel A.** 3, s.l. : IEEE Transactions on Information Theory, 2006, Vol. 52, pp. 1030-1051.

19. **Davenport, Mark A., Daurte, Marco F., Eldar, Yonina C. and Kutyniok, Gitta.** Introduction to Compressive Sensing. [Online] webee.technion.ac.il.

20. **Cheng, Hong.** Modeling and Learning Visual Recongintion Theory Algorithms and Applications. [Online] 2015. www.Springer.com.

21. *Counting faces of random projected polytopes when the projections radically lowers dimensions.* **Donoho, David L. and Tanner, Jared.** 1, 2006, Journal of the American Mathematical Society, Vol. 22.

22. *Greed is good: algorithmic results for sparse approximation.* **Tropp, Joel A.** 10, s.l. : IEEE Transactions on Information Theory, 2004, Vol. 50, pp. 2231-2242.

23. *Signal Recovery from Random Measurement via Orthogonal Matching Pursuit.* **Tropp, Joel A. and Gilbert, Anna C.** 12, s.l. : IEEE Transactions on Information Theory, 2007, Vol. 53.

24. *Matching Pursuits with Time-Frequency Dictionaries.* **Mallat, Stephane Georges and Zhang, Zhifeng.** 1993, IEEE Transactions on Signal Processing.

25. *Adaptive time frequency decompostions with Matching Pursuits.* **Mallat, Stephane Georges, Davis, Geoff and Zhang, Zhifeng.** 1994, SPIE Journal of Optical Engineering, Vol. 33, pp. 2183-2191.

26. *CoSamp: Iterative signal recovery from incomplete and inaccurate samples.* **Needell, Deanna and Tropp., Joel A.** 3, 2009, Applied and Computational Harmonic Analysis, Vol. 26, pp. 301-321.

27. *Iterative thresholding for sparse approximations.* **Blumensath, Thomas and Davis, Mike E.** 5, 2008, Journal of Fourier Analysis and Applications, Vol. 14, pp. 629-654.

28. *Iterative hard thresholding for compressed sensing.* **Blumensath, Thomas and Davis, Mike E.** 3, 2009, Applied and Computational Harmonic Analysis, Vol. 27, pp. 256-274.

29. *An iterative thresholding algorithm for linear inverse problems with a sparisty constraint.* **Daubechies, Ingrid, Defrise, Micheal and Mol, Christine De.** 11, 2004, Communications on Pure and Applied Mathematics, Vol. 57, pp. 1413-1457.

30. *Single-pixel imaging via compressive sampling.* **Duarte, Marco F., Davenport, Mark A., Takhar, Dharmpal, Laska, Jason N., Sun, Ting, Kelly, Kevin F. and Baraniuk, Richard G.** 2, 2008, IEEE Signal Processing Magazine, Vol. 25, pp. 83-91.

31. *Compressive sensing MRI.* **Lustig, Micheal, Donoho, David L., Santos, Juan M. and Pauly, John M.** 2, 2008, IEEE Signal Processing Magazine, Vol. 25, pp. 72-82.

32. *Compressive sensing: From theory to application, a survey.* **Qaisar, Saad, Bilal, Runa Muhammed, Iqbal, Wafa, Naureen, Muqaddas and Lee, Sungyoung.** 5, 2013, Journal of Communications and Networks, Vol. 15, pp. 443-456.

33. *Compressive blind image deconvolution.* **Amizic, Bruno, Spinoulas, Leonidas, Molina, Rafael and Katsaggeolos, Aggelos K.** 10, 2013, IEEE Transactions on Image Processing, Vol. 22, pp. 3994-4006.

34. *Blind image deblurring based on sparse representation and structural self-similarity.* **Yu, Jing, Chang, Zhenchun, Xiao, Chuangbai and Sun, WD.** 2017. 42th International Conference on Accoustics, Speech and Signal Processing. pp. 1328-1332.

35. *From Denoising to Compressed Sensing.* **Metzler, Christopher A., Maleki, Arian and Baraniuk, Richard.** 2014, IEEE Transactions on Information Theory.

36. *Compresssive Sensing Based Image Denoising Using Adaptive Multiple Samplings and Reconstruction Error Control.* **Wonseok Kand, Eunsung Lee, Sangjin Kim, Doochun Seo and Joonki Paik.** 2012, Proceedings of SPIE- The International Society for Optical Engineering, Vol. 8365.

37. **Ludwig, Jamie.** Satellite Digital Image Analysis. [Online] web.pdx.edu.

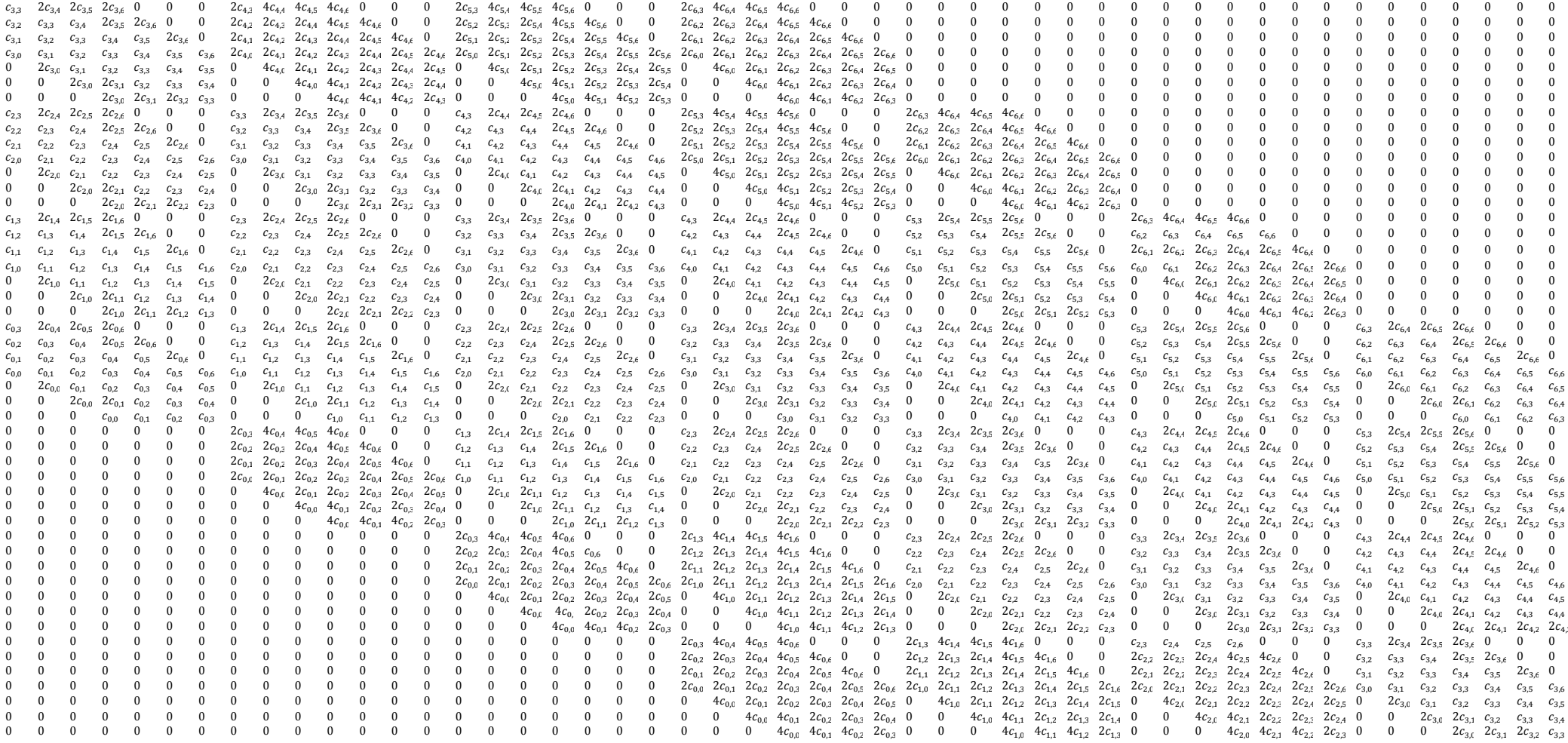
Appendix

I. The $\boldsymbol{\psi}$ matrix before mirroring

$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	0	0	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	0	0	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	0	0	0	$c_{6,3}$	$c_{6,4}$	$c_{6,5}$	$c_{6,6}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	0	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	0	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{6,6}$	0	0	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	$c_{6,5}$	$c_{6,6}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	0	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	$c_{6,5}$	$c_{6,6}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	$c_{6,5}$	$c_{6,6}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	0	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	$c_{6,5}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	0	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	0	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	0	0	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	0	0	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	0	0	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	0	0	0	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	0	0	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	0	0	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	0	0	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	0	0	0	$c_{6,3}$	$c_{6,4'}$	$c_{6,5}$	$c_{6,6}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	0	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	0	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	0	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	0	0	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	$c_{6,5}$	$c_{6,6}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	0	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	$c_{6,5}$	$c_{6,6}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	$c_{6,5}$	$c_{6,6}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	0	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	$c_{6,5}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	0	0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	0	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	0	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	0	0	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	0	0	0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	0	0	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	0	0	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	0	0	0	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	0	0	0	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	0	0	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	0	0	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	0	0	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	0	0	0	$c_{6,3}$	$c_{6,4'}$	$c_{6,5}$	$c_{6,6}$	0	0	0	0	0	0	0	0	0	0
$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	0	0	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	0	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	0	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	0	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	0	0	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	$c_{6,5}$	$c_{6,6}$	0	0	0	0	0	0	0	0	0
$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	0	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	0	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	$c_{6,5}$	$c_{6,6}$	0	0	0	0	0	0	0	0
$c_{1,0}$	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	$c_{6,5}$	$c_{6,6}$	0	0	0	0	0	0	0
0	$c_{1,0}$	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	0	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	0	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	$c_{6,5}$	0	0	0	0	0	0	0
0	0	$c_{1,0}$	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	0	0	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	0	0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	0	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	0	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	0	0	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4'}$	0	0	0	0	0	0	0
0	0	0	$c_{1,0}$	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	0	0	0	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	0	0	0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	0	0	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	0	0	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	0	0	0	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	0	0	0	0	0	0	0
$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	$c_{0,6}$	0	0	0	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	0	0	0	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	0	0	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	0	0	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	0	0	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	0	0	0	$c_{6,3}$	$c_{6,4}$	$c_{6,5}$	$c_{6,6}$	0	0	0
$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	$c_{0,6}$	0	0	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	0	0	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	0	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	0	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	0	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	0	0	$c_{6,2}$	$c_{6,3}$	$c_{6,4}$	$c_{6,5}$	$c_{6,6}$	0	0
$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	$c_{0,6}$	0	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	0	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	0	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4}$	$c_{6,5}$	$c_{6,6}$	0
$c_{0,0}$	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	$c_{0,6}$	$c_{1,0}$	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4}$	$c_{6,5}$	$c_{6,6}$
0	$c_{0,0}$	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	0	$c_{1,0}$	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	0	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	0	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4}$	$c_{6,5}$
0	0	$c_{0,0}$	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	0	0	$c_{1,0}$	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	0	0	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	0	0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	0	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	0	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	0	0	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$	$c_{6,4}$
0	0	0	$c_{0,0}$	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	0	0	0	$c_{1,0}$	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	0	0	0	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	0	0	0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	0	0	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	0	0	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	0	0	0	$c_{6,0}$	$c_{6,1}$	$c_{6,2}$	$c_{6,3}$
0	0	0	0	0	0	0	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	$c_{0,6}$	0	0	0	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	0	0	0	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	0	0	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	0	0	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	0	0	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	0	0	0
0	0	0	0	0	0	0	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	$c_{0,6}$	0	0	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	0	0	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	0	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	0	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	0	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	0	0
0	0	0	0	0	0	0	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	$c_{0,6}$	0	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	0	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$	0
0	0	0	0	0	0	0	$c_{0,0}$	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	$c_{0,6}$	$c_{1,0}$	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$c_{5,6}$
0	0	0	0	0	0	0	0	$c_{0,0}$	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	0	$c_{1,0}$	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	0	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$
0	0	0	0	0	0	0	0	0	$c_{0,0}$	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	0	0	$c_{1,0}$	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	0	0	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	0	0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	0	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	0	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$
0	0	0	0	0	0	0	0	0	0	$c_{0,0}$	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	0	0	0	$c_{1,0}$	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	0	0	0	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	0	0	0	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	0	0	0	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	0	0	0	$c_{5,0}$	$c_{5,1}$	$c_{5,2}$	$c_{5,3}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	$c_{0,6}$	0	0	0	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	0	0	0	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	0	0	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	0	0	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	$c_{0,6}$	0	0	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	0	0	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	0	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	0	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	$c_{0,6}$	0	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	0	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	0	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	0	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	$c_{0,0}$	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	$c_{0,6}$	$c_{1,0}$	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$c_{1,6}$	$c_{2,0}$	$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$c_{2,6}$	$c_{3,0}$	$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$c_{3,6}$	$c_{4,0}$	$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$c_{4,6}$

$$
\begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} & c_{0,4} & c_{0,5} & 0 & c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} & 0 & c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & c_{2,5} & 0 & c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} & c_{3,5} & 0 & c_{4,0} & c_{4,1} & c_{4,2} & c_{4,3} & c_{4,4} & c_{4,5} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} & c_{0,4} & 0 & 0 & c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & 0 & 0 & c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & 0 & 0 & c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} & 0 & 0 & c_{4,0} & c_{4,1} & c_{4,2} & c_{4,3} & c_{4,4} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} & 0 & 0 & 0 & c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} & 0 & 0 & 0 & c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} & 0 & 0 & 0 & c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} & 0 & 0 & 0 & c_{4,0} & c_{4,1} & c_{4,2} & c_{4,3} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{0,3} & c_{0,4} & c_{0,5} & c_{0,6} & 0 & 0 & 0 & c_{1,3} & c_{1,4} & c_{1,5} & c_{1,6} & 0 & 0 & 0 & c_{2,3} & c_{2,4} & c_{2,5} & c_{2,6} & 0 & 0 & 0 & c_{3,3} & c_{3,4} & c_{3,5} & c_{3,6} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{0,2} & c_{0,3} & c_{0,4} & c_{0,5} & c_{0,6} & 0 & 0 & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} & c_{1,6} & 0 & 0 & c_{2,2} & c_{2,3} & c_{2,4} & c_{2,5} & c_{2,6} & 0 & 0 & c_{3,2} & c_{3,3} & c_{3,4} & c_{3,5} & c_{3,6} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{0,1} & c_{0,2} & c_{0,3} & c_{0,4} & c_{0,5} & c_{0,6} & 0 & c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} & c_{1,6} & 0 & c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & c_{2,5} & c_{2,6} & 0 & c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} & c_{3,5} & c_{3,6} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} & c_{0,4} & c_{0,5} & c_{0,6} & c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} & c_{1,6} & c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & c_{2,5} & c_{2,6} & c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} & c_{3,5} & c_{3,6} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} & c_{0,4} & c_{0,5} & 0 & c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} & 0 & c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & c_{2,5} & 0 & c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} & c_{3,5} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} & c_{0,4} & 0 & 0 & c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & 0 & 0 & c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & 0 & 0 & c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} & 0 & 0 & 0 & c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} & 0 & 0 & 0 & c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} & 0 & 0 & 0 & c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3}
\end{matrix}
$$

Appendix- II ψ matrix after mirroring



Appendix III Measuring Matrix - M

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Appendix IV ϕ - Matrix