DEVELOPMENT & IMPLEMENTATION OF ALGORITHMS FOR FAST IMAGE RECONSTRUCTION

A thesis submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
University of Canterbury
2011

BY RACHAEL TAPPENDEN

Department of Mathematics and Statistics
University of Canterbury
New Zealand

ABSTRACT

Signal and image processing is important in a wide range of areas, including medical and astronomical imaging, and speech and acoustic signal processing. There is often a need for the reconstruction of these objects to be very fast, as they have some cost (perhaps a monetary cost, although often it is a time cost) attached to them. This work considers the development of algorithms that allow these signals and images to be reconstructed quickly and without perceptual quality loss.

The main problem considered here is that of reducing the amount of time needed for images to be reconstructed, by decreasing the amount of data necessary for a high quality image to be produced. In addressing this problem two basic ideas are considered. The first is a subset selection problem where the aim is to extract a subset of data, of a predetermined size, from a much larger data set. To do this we first need some metric with which to measure how 'good' (or how close to 'best') a data subset is. Then, using this metric, we seek an algorithm that selects an appropriate data subset from which an accurate image can be reconstructed. Current algorithms use a criterion based upon the trace of a matrix. In this work we derive a simpler criterion based upon the determinant of a matrix. We construct two new algorithms based upon this new criterion and provide numerical results to demonstrate their accuracy and efficiency. A row exchange strategy is also described, which takes a given subset and performs interchanges to improve the quality of the selected subset.

The second idea is, given a reduced set of data, how can we quickly reconstruct an accurate signal or image? Compressed sensing provides a mathematical framework that explains that if a signal or image is known to be sparse relative to some basis, then it may be accurately reconstructed from a reduced set of data measurements. The reconstruction process can be posed as a convex optimization problem. We introduce an algorithm that aims to solve the corresponding problem and accurately reconstruct the desired signal or image. The algorithm is based upon the Barzilai-Borwein algorithm and tailored specifically to the compressed sensing framework. Numerical experiments show that the algorithm is competitive with currently used algorithms.

Following the success of compressed sensing for sparse signal reconstruction, we consider whether it is possible to reconstruct other signals with certain structures from reduced data sets. Specifically, signals that are a combination of a piecewise constant part and a sparse component are considered. A reconstruction process for signals of this type is detailed and numerical results are presented.

ACKNOWLEDGMENTS

My first thanks go to Dr Bob Broughton for introducing me to research and showing me that it wasn't an impossible task. It is actually a hugely interesting and satisfying pursuit. Dr Peter Renaud initially planted the idea of doing a PhD in my head, so it is thanks to him and his suggestion that I am at this point now. Thank you to Dr Ian Coope for agreeing to be my senior supervisor and preparing me for the life of a researcher.

Ian, Bob and Peter have been fantastic supervisors. They have given me a huge amount of expertise and support throughout this process and I am extremely grateful for their time, guidance, and patience. Being part of the BIRP team has been really great!

I would also like to thank my family, friends, and especially Tim, for believing in me and encouraging me throughout this process. They have smiled and nodded when I have talked about maths and supported me through the many ups and downs of the research process.

Sharing an office with Anna for the past three years has been a real pleasure and a privilege. I am lucky to have such a great friend to share in the research experience, which has made it so much more enjoyable.

Thank you also to Scott 'it's Graybill time', and my fellow postgraduate students for making my time at UC so enjoyable. The bountiful supply of coffee and games of 500 have made it all worthwhile!

Thank you to all the staff from the Department of Mathematics and Statistics for always taking an interest in the students and providing a really positive working environment. It has been a pleasure to come into the office each day — something I have greatly missed over the past couple of months, with the Erskine building out of action following February's earthquake.

Finally, I gratefully acknowledge the financial support from the Department of Mathematics and Statistics at the University of Canterbury and from the New Zealand Institute of Mathematics and its Applications. Without this support, this thesis would not have been possible.

CONTENTS

NOTATION

Let A be a general matrix.

A^T	Transpose
$\overline{A}$	Complex conjugate
A^H	Complex conjugate transpose
A^+	Moore-Penrose (Pseudo-) Inverse
A^{-H}	$(A^H)^{-1} = (A^{-1})^H$
A^{-2}	$(A^2)^{-1}$
I	The (appropriately sized) identity matrix
$\mathbf{1}$	A vector whose entries are all one

CHAPTER 2

a_i	A row of A
$\mathcal{B}$	Index subset of selected rows
$\mathcal{N}$	Index subset of non selected rows
$.*$	Element-wise multiplication
$./$	Element-wise division

CHAPTER 3

n	Signal length
s	Number of non-zeros in signal

CHAPTER 1

INTRODUCTION

1.1 SIGNALS AND IMAGES: THE DIGITAL AGE

We live in a world alive with images. Whether watching a DVD, downloading and streaming images from the internet, or using web cameras and Skype to interact with one another, we are constantly communicating electronically with digitised images. This use is not restricted to our everyday lives and entertainment but is also an integral part of modern day science and medicine. Examples include electron microscopes, satellite and aerial imagery, high powered telescopes and all types of medical scanning devices.

We consider signals as being represented by discrete (and therefore digitised) data, collected by some sampling process. For example, geological recording of seismological data for earthquake research, the collection of data from medical scanning machines and video surveillance cameras.

An obvious tool for collecting data is the digital camera. Although a digital camera is capable of taking high resolution images, to be able to send the image electronically via email, the amount of data must first be reduced or it will take up too much bandwidth and hence time. In practice we use tools to considerably lower the quantity of data, whilst retaining the integrity of the image to enable speedy and efficient transmission.

The digitization of signals and images requires computer technology to process the data. The type of tasks involved include filtering, sharpening, deblurring, compressing, decompressing, reconstruction and restoration. For example, current street surveillance cameras often record data that is not detailed enough to produce high quality images and requires image enhancement to identify subjects, particularly when the recording is in darkness and from a distance. Another illustration is the restoration of historic, degraded paintings. Here the amount of information is limited from faded, flaking or non-existent paintwork and the task is to restore the image to its former glory.

In other instances it is simply not possible to collect high resolution information because it is unrealistic in terms of time and resources. This occurs in astronomical imaging where the collection and transmission of data from space is severely constrained. In medical scanning we wish to minimize the time that patients are subjected to scanning devices, while obtaining

the desirable quality of image. In both of these instances we require reconstruction techniques.

Since biomedical imaging, image and video processing, neuroscience, and visualisation are all areas of application exploiting sampling and reconstruction theory, it is apparent that interaction with mathematicians, electrical engineers, medical personnel and scientists is involved. It can also be seen that this interdisciplinary area provides a large and ever expanding field of research.

Since a digital image may be represented by a mathematical model, based on arrays of data, it is not surprising that mathematics has a central role to play in this area. Mathematics is the thread that ties these disciplines and ideas together. The area lends itself to high level mathematics such as Fourier analysis and modern wavelet techniques in the development of theory and the design of algorithms. There are many books that illustrate the strong mathematical links in signal processing including [51, 52, 69, 85, 101].

1.1.1 SIGNALS, IMAGES AND MATHEMATICS

Because the work in this thesis is based upon signal and image processing, it is important that we understand what these concepts mean mathematically. They are defined now.

A signal is a one-dimensional function $f(x)$ that takes a continuous input and gives a continuous output. The independent variable x is often time, while the value $f(x)$ is the height of the signal at each instance. This work is concerned with discrete, digitised signals. These are continuous signals that have been sampled evenly at a discrete number of points so that $f(x)$ is also discrete. In this way we think of a signal as a vector whose entries correspond to the sampled values of $f(x)$.

An image is usually defined as a two-dimensional function $f(x, y)$ where x and y are the plane coordinates and $f(x, y)$ is the intensity or grey-scale value of the image at that point. Again, we restrict our attention to digitised images where the quantities x, y and $f(x, y)$ are all discrete. An image is therefore written as a matrix whose dimensions dictate the size of the image. Each element in the matrix (and therefore the image) is commonly known as a picture element or *pixel*. By concatenating the columns (or rows) of the image matrix we form a long vector and in this way we can think of images and signals as being essentially the same.

We can similarly describe higher-dimensional objects, for example videos, although in this work we are primarily concerned with the 1D and 2D cases.

1.1.2 SIGNALS, IMAGES AND MODERN APPLICATIONS

We have already alluded to the fact that signal and image reconstruction is fundamental to modern day living and occurs seamlessly in the background without many people even noticing. We describe a few more reconstruction applications here.

In mobile phone technology, the human voice is a sound wave that is sampled and digitised by a chip in the phone before being transmitted to a receiving phone where the process is reversed to produce an audio output. This all happens in real time.

Another application occurs in music. People enjoy listening to music and like to download new songs (which can be thought of as digitised signals). With a slow internet connection this can take a considerable amount of time but with careful compression techniques, for example songs stored as MP3s, the size of the song can be significantly reduced and downloading will therefore be much faster. The compressed version is not the same as the original but most users cannot distinguish it from the original. This is because the data has been manipulated in a compression algorithm to remove data that the user will not miss — data outside the auditory resolution of the common user.

This idea of compression without loss is also at the heart of every digital camera. The new JPEG2000 compression algorithm replaces the discrete cosine transform based compression algorithm with a wavelet-based method. The algorithm samples the image and performs the discrete wavelet transform. The largest coefficients are retained and stored while the remainder are discarded. The JPEG2000 standard uses lossless (and lossy) compression provided by a reversible integer wavelet transform.

Signal and image processing is central to many of today's technological advances. These techniques are also widely applied in medical imaging, from x-ray to computed tomography, from ultrasound to MRI. The original motivation for this work arises specifically in magnetic resonance imaging. For this reason it is worthwhile describing MRI in slightly more detail.

1.1.3 MAGNETIC RESONANCE IMAGING

One of the most important medical applications currently employing image reconstruction techniques is Magnetic Resonance Imaging (MRI). MRI is commonly used as a diagnostic tool because it allows the internal physical and chemical characteristics of an object to be observed, using externally measured nuclear magnetic resonance signals. Unlike computed tomography, these signals do not expose the object to any ionising radiation. Clearly this has great advantages in medicine, as diagnoses can be made without invasive procedures. One feature that makes MRI so popular is that the images provide good soft tissue contrast, while other imaging modalities cannot. This makes MRI particulary useful, for example, in the diagnosis of tumours and the imaging of blood vessels (angiography).

One of the main disadvantages of MRI is that a scan can take a long time and the bulk of this time is spent on the data collection process. Reducing the time required for data collection has many benefits, several of which are described below.

1. The scanner itself is very expensive (currently it costs around US$1 million per tesla with a hospital grade magnet ranging from 1–3 tesla). If scan time can be reduced, the scanner itself is used more efficiently, which has the added benefit of reduced patient waiting times.

2. The patient being scanned must remain in the scanner until all images have been produced, throughout which time they must remain perfectly still, otherwise the image will suffer from motion artifacts. This can cause great discomfort and is particularly difficult for young children and the elderly, so reducing the data collection time reduces the discomfort for patients.

3. Reducing the data collection time widens the scope of imaging that may be done by the scanner and it has the potential to allow for real-time imaging, for example, of cardiac cycles. (Currently for these types of images, the data is gated into different time windows so only a very small amount of data is available for image reconstruction).

The physics underpinning MRI is based on the fact that protons (Hydrogen atoms) are abundant in the body. Particles spin on their own axis so a proton can be thought of as a continuously rotating positive charge that creates its own tiny magnetic field. When a proton is placed in a strong external magnetic field (as is the case in MRI), it attempts to align itself with this field. Taking the vector sum of the individual protons gives net magnetization aligned with the external field. Applying a radiofrequency (RF) pulse at the Larmor frequency (approximately 42.6MHz for protons, [92, p. 103]) causes resonance as the protons are all precessing at this same frequency. It forces the protons to flip perpendicular to the external magnetic field. Once the application of the RF pulse stops, the protons slowly relax to align with the external field direction. During the relaxation period, the response within the patient's tissue is measured and recorded by the MRI machine. Different tissues have different densities of protons so respond differently to the RF pulse, and therefore show up in the MR image at different intensities (or brightness).

On a larger scale, a magnetic gradient is set up along the length of the patient (the z coordinate direction). Applying an RF pulse causes resonance in a particular slice of the patient. The specific frequency of the RF pulse isolates the corresponding slice along the body, while the strength of the z gradient dictates the width of the slice. Each slice is an x, y plane. The x direction corresponds to frequency encoding (FE) and the y direction corresponds to phase encoding (PE).

The MR measurements (or samples) are made in the Fourier domain. This raw data space is commonly known as k-space in the MR community. Samples are usually acquired on a regular, rectangular grid around the origin and the sample spacing is dictated by the Nyquist rate[1]. Subsequently, k-space is a matrix. The number of signals collected (echoes) determines the number of rows of the k-space matrix (phase encoding direction) while the number of sample points of each echo determines the number of columns in the k-space matrix (frequency encoding direction). When a full Nyquist set has been selected (the k-space matrix has been filled with data) an image can be produced using the fast Fourier transform (with zero padding of the data if necessary).

[1]The Nyquist rate is twice the maximum frequency found in the signal. This will be described in Chapter 3 although further information can be found in [86].

The k-space matrix is filled in the following way. A slice selective gradient (z direction) and an RF pulse are applied to the object. Next, gradients in both the PE and FE directions are applied, followed by data acquisition. The strength of the phase encoding gradient dictates which row of k-space is being filled. Depending on the length of the RF pulse, and strength of the FE gradient, a single entry or an entire row in the FE direction is filled. We therefore have complete control over the order in which the elements of k-space are filled. For further information on MRI, see for example [38, 67, 102].

1.2 MATHEMATICAL INTERPRETATION

In this section we provide a mathematical description of the MRI problem set up to show the systems of equations that arise and to provide some motivation for the algorithms described in later chapters.

Mathematically we have a system of linear equations

$$Fx = y. \tag{1.1}$$

The matrix F is an $n \times n$ Fourier matrix and x is a length n vector representing the $\sqrt{n} \times \sqrt{n}$ image to be reconstructed. (Recall from section 1.1.1 that we can reshape an image into a vector, for example, by concatenating the columns of the matrix.) The vector y represents the observations to be acquired by the scanner and also has length n. Equation (1.1) demonstrates that we are essentially acquiring the Fourier transform of the image, rather than measuring the image directly. The quantities F, x and y are all complex.

The following two subsections describe two approaches to the MRI reduced data problem. The main objectives are to investigate reducing image reconstruction time by reducing the number of observations. The two main questions that must be addressed are:

1. How do we choose the data to observe, while ensuring a high quality signal or image is reproducible?

2. How do we accurately reconstruct a signal, given a sub-sampled data set?

Approaches to questions 1 and 2 are found in each of the following two subsections respectively. The approach to question 1 uses a priori knowledge of the image to formulate an approach to the subset selection problem. The approach to question 2 in the second subsection assumes no knowledge about the image and focuses on reconstructing an image from a reduced data set.

1.2.1 THE SUBSET SELECTION APPROACH

Often in MRI a so-called 'scout image' of the object is quickly computed before the actual scan to define the Region of Interest (ROI). For example, in a brain scan the ROI will be an oval shape just containing the brain. From this scout image we know that values of x outside the ROI will be zero (corresponding to air). A binary mask can therefore be applied

to x. That is, premultiplying the image vector by a diagonal matrix with binary entries, ones corresponding to x inside the ROI and zeros elsewhere. This is in turn equivalent to zeroing columns of the Fourier matrix. The zero columns of F and the zero entries in x can be deleted without losing any information. Let m be the number of non-zeros in the binary mask (corresponding to m pixels inside the ROI). Then define $\hat{F}$ to be the resulting $n \times m$ partial Fourier matrix and define $\hat{x}$ to be the resulting vector of m unknowns. The corresponding system of equations $\hat{F}\hat{x} = y$, is now overdetermined. Furthermore, each of the elements in the vector y represents a location in k-space. Selecting a reduced set of data is equivalent to selecting a certain number of elements in the vector y to observe, which is the same as selecting certain rows of the partial Fourier matrix $\hat{F}$.

The problem is, select a predetermined number of rows from the matrix $\hat{F}$ that ensure an accurate reconstruction of $\hat{x}$ (and subsequently x) is possible. Once the rows of $\hat{F}$ have been chosen, the corresponding locations in k-space can be observed by the MRI machine and the measured data stored in the vector y.

While this is a specific application that supplied the motivation for much of this work, a more general form of this problem is considered in chapter 2 and based upon the following. From a complex system of $m \times n$ $(m > n)$ equations, $Ax = b$, select k rows of A (where $n < k < m$), corresponding to k equations, that ensure x can be reconstructed accurately. This row-subset selection problem is considered in more detail in Chapter 2.

1.2.2 THE RECONSTRUCTION APPROACH

Another way to approach the MRI problem assumes no a priori knowledge of the region of support of the image. In MR, images can often be sparsely represented in a particular basis, for example a Fourier or a wavelet basis. Other images, such as an angiogram (blood vessel images), are already sparse in the pixel domain. The question arises: if we know that few elements of the image are non-zero, is the full Nyquist data set required to accurately recover the image? Because no assumptions are being made about the support of the image, a random sampling scheme is recommended (this choice will be justified in chapter 3) and few equations are needed to reconstruct x accurately. For this application we again have the system of equations $\hat{F}x = y$, but this time $\hat{F}$ is of size $m \times n$ where $m \ll n$ and the rows of $\hat{F}$ are randomly chosen.

The relatively new mathematical theory that supports this approach is *compressed sensing* (also known as compressive sensing or compressed sampling). This problem is the motivation for Chapter 3, in which we propose a new reconstruction algorithm. However, as for the preceding approach, we consider this problem in a more general form: from the (real) system of $m \times n$ $(m < n)$ equations $Ax = b$, find the solution that is most sparse. This can be posed as a regularized least squares problem and the reconstruction of such signals is based upon optimisation techniques.

1.3 THESIS ORGANISATION

The work in this thesis is separated into three parts, each of which is described in a separate chapter.

Chapter 2 discusses the idea of observation subset selection. The problem is, given some data matrix, how do we choose the best size k row subset? When considering this problem we must decide upon a measure of 'best'. The first part of this chapter introduces a known subset selection criterion that is based upon minimizing the trace of a matrix. It then introduces two currently used algorithms that adopt the trace criterion. These algorithms perform well but are suboptimal. The idea of a row exchange process is therefore considered to improve the selected subset and a row exchange criterion based upon the trace is derived. We describe a row exchange algorithm that uses the new row exchange criterion.

We introduce a new selection criterion that replaces the arithmetic mean (trace) with the geometric mean (determinant). Two algorithms based on this new criterion are described, followed by the derivation of a determinant-based row exchange criterion. A section containing numerical results is included to demonstrate the quality of the subset selection process.

Chapter 3 is concerned with the relatively new idea of compressed sensing. This concept also deals with reconstruction of signals from only a small amount of data, but here we assume that the reduced data set has already been collected and the focus is obtaining a high quality signal reconstruction. In compressed sensing, the signal to be reconstructed is assumed to be sparse. The underlying mathematical theory is introduced and the reconstruction process is posed as a convex optimization problem. The remainder of the chapter is concerned with deriving a new algorithm to solve the associated l_1-regularized least squares problem formulation. The algorithm uses the Barzilai-Borwein step lengths and we derive a result showing that a constant step length is calculated if the measurement matrix has orthonormal rows. We also derive an upper bound on the elements of the solution vector, allowing us to pose the problem as a box-constrained quadratic programming problem. An algorithm tailored to this problem is detailed and numerical results are presented to show that the algorithm is accurate, robust and efficient.

Chapter 4 considers extending the ideas of compressed sensing to signals that have other known structures. Piecewise constant signals can be reconstructed accurately from limited data sets through use of a discrete derivative operation. We specifically consider the reconstruction of signals that are a combination of a piecewise constant part and a sparse component. The reconstruction problem can be posed as an optimization problem in $2n$ unknowns. The reconstruction process involves separating the single optimization problem into two optimization problems, each in n unknowns. The first optimization procedure locates the position of the discontinuities in the piecewise constant signal. A projection step is then applied and the sparse signal is reconstructed. The sparse and piecewise constant signals are then recombined to give the unknown signal. The optimization problems are solved by the algorithm described in Chapter 3 and numerical results are given.

CHAPTER 2

OBSERVATION SUBSET SELECTION

2.1 PROBLEM BACKGROUND

Many problems in signal and image processing can be modeled by the linear equation

$$b = Ax + w, \tag{2.1}$$

where b is a vector of observations, $A \in \mathbb{C}^{m\times n}$ is a measurement matrix ($m > n$ and $\text{rank}(A) = n$) and w is additive noise. Noise is essentially a component that is uncorrelated with the desired signal and we often assume it has certain properties, for example, it is Gaussian. Determining the original, unknown signal (or image) is an inverse problem and an efficient yet accurate method for reconstructing x is required.

Problem (2.1) represents an overdetermined system of equations and the standard way of solving this kind of problem uses a least squares approach. The normal equations for (2.1) are

$$A^H A\hat{x} = A^H(b - w).$$

If A has full column rank then $\hat{x}$ is the unique solution for these equations and gives the reconstruction that minimizes the noise w in the 2-norm sense. (There is a difficulty here as w is unknown. This is addressed later in Section 2.2.3.)

As $m > n$, the matrix A has linearly dependent rows. The question arises, "Can we accurately reconstruct the vector x using a subset of the rows available from A?" That is, does the system (2.1) contain redundant information? This is an important problem to consider because in practical situations, for example medical imaging, the vector b may represent observations that are costly to obtain. The problem becomes: given some a priori information about the matrix A, select a subset of rows of A in such a way as to preserve the quality of the vector x to be reconstructed. To select the 'best' subset of rows of a matrix is a combinatorial problem and to take such an approach is computationally infeasible. The multiplicative formula for the binomial coefficient tells us that there are

$${}^mC_k = \frac{m!}{(m-k)!\,k!}$$

combinations of k rows out of a total of m possible rows. Even for quite small values of m and k, ${}^mC_k$ is prohibitively large. (For example, if $m = 30$ and $k = 20$ there are over 30 million combinations.) Thus we need a criterion that allows us to decide which rows to choose without looking at all possible combinations of rows.

2.1.1 CHAPTER ORGANISATION

Chapter 2 is concerned with the formulation of an observation subset selection criterion and the implementation of algorithms that select data according to that criterion. The aim is to select a reduced set of data, thereby reducing data acquisition time, while preserving the reconstruction quality.

In Section 2.2 a practical observation subset selection criterion is derived. Current algorithms that use this selection criterion are discussed in Section 2.3, together with some of their advantages and disadvantages and examples of practical applications. Section 2.4 derives an extension to the existing algorithms. A new subset selection criterion is proposed in Section 2.5 and this forms the basis of several new subset selection algorithms. Finally Section 2.6 presents numerical experiments to compare the performance of the existing and new algorithms.

2.2 DERIVATION OF A SUBSET SELECTION CRITERION

This section is concerned with deriving a practical observation subset selection criterion. Several mathematical concepts used in the derivation of the criterion are presented, followed by the criterion itself.

2.2.1 THE MOORE-PENROSE INVERSE

Consider the system of equations $Ax = b$ where $A \in \mathbb{C}^{m\times n}$ and $m \geq n$. The normal equations are

$$A^H A\hat{x} = A^H b. \tag{2.2}$$

If A has full rank then $A^H A$ is nonsingular and (2.2) has a unique solution. We call the matrix

$$A^+ = (A^H A)^{-1} A^H, \tag{2.3}$$

the Moore-Penrose Inverse (also known as the generalised inverse or pseudoinverse) and $x = A^+ b$ is the solution to $Ax = b$ that has smallest Euclidean norm.

Remarks:

1. If A has full rank with $m < n$ then the Moore-Penrose Inverse is defined as

$$A^+ = A^H (AA^H)^{-1}.$$

2. If $m = n$ and A^{-1} exists, then

$$A^+ = (A^H A)^{-1} A^H = A^{-1}(A^{-H} A^H) = A^{-1}.$$

3. The pseudoinverse may also be defined as the unique Frobenius norm solution to the problem

$$\min_{X \in \mathbb{R}^{n \times m}} \|AX - I_m\|_F,$$

where the Frobenius norm is

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}.$$

For further details, see for example Golub and Van Loan [50, p.257] or Watkins [100, p.277].

2.2.2 THE SINGULAR VALUE DECOMPOSITION

An important mathematical technique is the singular value decomposition (SVD). In the mentioned signal and imaging applications, the aim is to reduce the amount of data to be acquired, without losing any essential information. An image is generally made up of different parts, some containing a large amount of detail such as a person's face, and other parts with less detail, for example the sky in the background of a photograph. The singular values correspond to image detail. The largest singular values contribute most of the information needed for reconstruction, while the smaller singular values contain only a small amount of detail. Formally, the singular value decomposition of an $m \times n$ matrix A is

$$A = U \Sigma V^H, \tag{2.4}$$

where $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ are both unitary matrices and Σ is a diagonal matrix. Let $r = \text{rank}(A)$. Then $\Sigma \in \mathbb{C}^{m \times n}$ is the diagonal matrix with elements

$$\sigma_1 \geq \cdots \geq \sigma_r > 0, \qquad \sigma_{r+1} = \cdots = \sigma_{\min(m,n)} = 0,$$

and $\sigma_1, \ldots, \sigma_r$ are called the singular values of A. The r singular values of A and the non-zero eigenvalues (λ_j) of both $A^H A$ and AA^H are linked by the formula $\sigma_j^2 = \lambda_j$. For a full-rank matrix ($A \in C^{m \times n}$ and $m \geq n$) all the singular values are strictly positive.

We can write the SVD of A in outer product form:

$$A \;=\; \sum_{j=1}^{r} \sigma_j u_j v_j^H \;=\; \sigma_1 u_1 v_1^H + \cdots + \sigma_r u_r v_r^H,$$

where u_j and v_j are the jth columns of U and V respectively. For compression purposes we can truncate this sum to get rid of the smaller singular values, which contribute only minimally to the image reconstruction. Thus we have the approximation

$$A \;\approx\; \sum_{j=1}^{p} \sigma_j u_j v_j^H \;=\; \sigma_1 u_1 v_1^H + \cdots + \sigma_p u_p v_p^H, \qquad p \leq r. \tag{2.5}$$

The pseudoinverse of a full-rank matrix can also be defined in terms of the SVD. Recall equations (2.3) and (2.4) and that U and V are unitary matrices. Note also that, for a unitary matrix, $(V^{-1})^H = (V^H)^H = V$. Then

$$\begin{aligned} A^+ &= (A^H A)^{-1} A^H, \\ &= (V\Sigma^H U^H U \Sigma V^H)^{-1} V \Sigma^H U^H, \\ &= (V\Sigma^H \Sigma V^H)^{-1} V \Sigma^H U^H, \\ &= V^{-H} (\Sigma^H \Sigma)^{-1} \Sigma^H U^H, \\ &= V \Sigma^+ U^H, \end{aligned} \tag{2.6}$$

where

$$\Sigma^+ = \left(\begin{array}{ccc|c} \frac{1}{\sigma_1} & & & \\ & \ddots & & 0 \\ & & \frac{1}{\sigma_n} & \end{array} \right). \tag{2.7}$$

When A is rank-deficient with rank r ($< n \leq m$), U and V can be decomposed and the SVD can be expressed as follows (see for example [34]):

$$A = \begin{pmatrix} U_1 & | & U_2 \end{pmatrix} \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1 & | & V_2 \end{pmatrix}^H = U_1 \Sigma_1 V_1^H,$$

where $\Sigma_1 = \text{diag}\left(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_r}\right)$ is a diagonal matrix. This means that the pseudoinverse is still defined for a rank-deficient matrix through

$$A^+ = V_1 \Sigma_1^+ U_1^H.$$

For the remainder of this chapter, A is assumed to have full column rank. The SVD and the Moore-Penrose inverse (2.6) can also be used to define the following expression, which will be useful in the next subsection:

$$(A^+)^H (A^+) = U \left((\Sigma^+)^H (\Sigma^+) \right) U^H. \tag{2.8}$$

Let

$$D = \left((\Sigma^+)^H \Sigma^+ \right), \tag{2.9}$$

so, from (2.7), D is an $m \times m$ diagonal matrix with diagonal elements $d_j = 1/\sigma_j^2$ for $j = 1 \ldots n$, and $d_j = 0$ for $j = n+1 \ldots m$. Then (2.8) becomes

$$(A^+)^H (A^+) = U D U^H. \tag{2.10}$$

Now we have the tools to describe the solution of (2.1) in terms of the SVD. In the noiseless case ($w = 0$),

$$x = A^+ b = V \Sigma^+ U^H b = \sum_{j=1}^{n} \frac{1}{\sigma_j} (u_j^H b)\, v_j.$$

However when noise is present,

$$x \;=\; \sum_{j=1}^{n} \frac{1}{\sigma_j}\,(u_j^H b)\; v_j \;+\; \sum_{j=1}^{n} \frac{1}{\sigma_j}\,(u_j^H w)\; v_i. \tag{2.11}$$

Clearly, the second term of (2.11) can be very large for small values of σ_j. That is, small singular values tend to amplify noise. This supports the idea of using a truncated sum (2.5) and eliminating small singular values. So, one way to approach the observation subset selection problem is to compute the singular value decomposition of A and choose a set of k rows from the observation matrix with this in mind. Unfortunately, computing the singular values of a matrix is computationally expensive and therefore selecting the subset of rows of A using a reduced set of the singular values is not a viable method in practice. Thus we would like a selection criterion that does not rely on prior knowledge of the singular values.

2.2.3 A SUBSET SELECTION CRITERION

Let x be the solution to the noiseless linear system

$$Ax = b \quad \Rightarrow \quad x = A^{+}b. \tag{2.12}$$

When noise w is present, let

$$\hat{x} = A^{+}(b + w). \tag{2.13}$$

Substituting (2.12) into (2.13) gives

$$\hat{x} \;=\; x + A^{+}w. \tag{2.14}$$

Rearranging (2.14) and taking norms gives

$$\|\hat{x} - x\|_2^2 \;=\; \|A^{+}w\|_2^2\,,$$

which means that the error in the reconstruction of x is proportional to the noise in the observations. We wish to minimize the error $\|\hat{x} - x\|_2^2$. However the problem is not well posed because we do not know x or $\hat{x}$.

We also do not know w but because the noise is assumed to be random we consider expected values. Using equation (2.10)

$$\begin{aligned} \mathbb{E}\{\|A^{+}w\|^2\} &= \mathbb{E}\{w^H (A^{+})^H (A^{+}) w\}, \\ &= \mathbb{E}\{w^H U D U^H w\}. \end{aligned}$$

Let $y = U^H w$. Then

$$\begin{aligned} \mathbb{E}\{\|A^{+}w\|^2\} &= \mathbb{E}\,\{y^H D y\}, \\ &= \mathbb{E}\Big\{\sum_{j=1}^{n} |y_j|^2 d_j\Big\}, \qquad \text{as } D \text{ is a diagonal matrix (2.9)} \\ &= \sum_{j=1}^{n} d_j\; \mathbb{E}\{|y_j|^2\}, \qquad \text{by linearity of expected value} \\ &= \sum_{j=1}^{n} \frac{1}{\sigma_j^2}\; \mathbb{E}\{|y_j|^2\}. \end{aligned}$$

The variance is defined as

$$\begin{aligned} \text{var}(X) &= \mathbb{E}((X - \mathbb{E}(X))^2), \\ &= \mathbb{E}(X^2) - (\mathbb{E}(X))^2. \end{aligned}$$

It is usually assumed that the y_i's are i.i.d. with mean 0 and variance σ^2. (In this case, $\mathbb{E}\{y_j\} = 0$, so $\mathbb{E}\{|y_j|^2\} = \sigma^2$). Therefore,

$$\mathbb{E}\{\|A^+ w\|^2\} = \sigma^2 \left(\sum_{j=1}^{n} \frac{1}{\sigma_j^2} \right). \tag{2.15}$$

But the sum in (2.15) is simply the trace of $(A^H A)^{-1}$, so

$$\mathbb{E}\{\|A^+ w\|^2\} = \sigma^2 \, \text{trace}(A^H A)^{-1}. \tag{2.16}$$

Equation (2.16) explains that the expected value of the noise in the reconstruction is proportional to $\text{trace}(A^H A)^{-1}$. This means that to minimize the noise in the reconstructed signal we should maximize the sum of the eigenvalues of $A^H A$. Obviously $\text{trace}(A^H A)^{-1}$ is fixed for a fixed matrix so what this means in the context of the subset selection problem is that the rows of A should be chosen in such a way as to maximize the sum of the singular values of the resulting submatrix.

Let $\mathcal{B}$ represent a subset of row indices corresponding to the rows of A that are selected in the subset selection process. That is $\mathcal{B} \subseteq \{1, 2, \ldots, m\}$. Then $A_{\mathcal{B}}$ is a submatrix of A whose rows are indexed by $\mathcal{B}$. The subset selection criterion now becomes:

$$\min_{\mathcal{B}} \ \text{trace}(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1}. \tag{2.17}$$

Now that we have a means of deciding on suitable rows to select, we need an algorithm to optimize this criterion. The next section discusses algorithms that aim to solve problem (2.17).

2.3 EXISTING ALGORITHMS

Reeves and Heck [78, 79] have considered the observation subset selection problem and have formulated it in the following way. Begin with an $m \times n$ matrix A $(m > n)$. Choose an integer k $(n < k < m)$, where k is the number of observations allowed. Select k rows from A to obtain a $k \times n$ submatrix $A_{\mathcal{B}}$, in such a way as to find

$$\min_{\mathcal{B}} \ \text{trace}(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1}.$$

Now that we have a mathematical formulation for the subset selection problem, we consider two currently used algorithms that approximately solve expression (2.17).
In these algorithms, at each iteration one index is added to, or deleted from, the index set $\mathcal{B}$. The algorithms terminate when $|\mathcal{B}| = k$.

In the remainder of this chapter we follow the notation in [79] and denote the jth row of the matrix A by the *row* vector a_j. We also assume, without loss of generality, that A has full column rank.

2.3.1 THE SEQUENTIAL BACKWARD SELECTION ALGORITHM

The Sequential Backward Selection (SBS) algorithm sequentially eliminates rows from the original matrix A until k rows remain. Initially $\mathcal{B} = \{1, 2, \ldots, m\}$ and at each iteration the cardinality of $\mathcal{B}$ reduces by one. To determine which row a_j (for $j \in \mathcal{B}$) to delete from $A_\mathcal{B}$, the rank-1 update $(A_\mathcal{B}^H A_\mathcal{B} - a_j^H a_j)^{-1}$ is evaluated. The Sherman-Morrison-Woodbury matrix inversion formula [5, p. 67] for a non-singular matrix M and *column* vectors x and y is

$$(M + xy^H)^{-1} = M^{-1} - \frac{M^{-1}xy^H M^{-1}}{(1 + y^H M^{-1} x)}. \tag{2.18}$$

Applying (2.18) to $(A_\mathcal{B}^H A_\mathcal{B} - a_j^H a_j)^{-1}$ and taking the trace gives

$$\text{trace}(A_\mathcal{B}^H A_\mathcal{B} - a_j^H a_j)^{-1} = \text{trace}(A_\mathcal{B}^H A_\mathcal{B})^{-1} + \frac{a_j (A_\mathcal{B}^H A_\mathcal{B})^{-2} a_j^H}{1 - a_j (A_\mathcal{B}^H A_\mathcal{B})^{-1} a_j^H}. \tag{2.19}$$

For expression (2.19) we note that the trace is unchanged under cyclic permutations of the matrices, i.e., $\text{trace}(CDE) = \text{trace}(DEC)$ [74, p. 5]. As $\text{trace}(A_\mathcal{B}^H A_\mathcal{B})^{-1}$ is independent of row j, the optimisation criterion that the SBS algorithm uses is

$$\min_j \quad \frac{a_j (A_\mathcal{B}^H A_\mathcal{B})^{-2} a_j^H}{1 - a_j (A_\mathcal{B}^H A_\mathcal{B})^{-1} a_j^H}. \tag{2.20}$$

Therefore the row deleted at each iteration of the SBS algorithm is the row that minimizes (2.20). If row a_j is deleted then $\mathcal{B} \leftarrow \mathcal{B} \setminus \{j\}$.

Before we describe the implementation of the SBS algorithm we first describe the QR decomposition, along with the economy QR decomposition, as these are used throughout the remainder of this chapter.

THE QR DECOMPOSITION

Formally, the QR decomposition (or factorisation) can be defined as follows [50, p. 223].

Definition 2.3.1. *The QR factorisation of a complex $m \times n$ matrix A ($m \geq n$) is given by*

$$A = Q\hat{R},$$

where $Q \in \mathbb{C}^{m \times m}$ is unitary and $\hat{R} \in \mathbb{C}^{m \times n}$ is upper triangular.

Note that if A has full column rank then the first n columns of Q form an orthonormal basis for range(A). The matrix Q can be decomposed to give

$$A \;=\; Q\hat{R} \;=\; \begin{bmatrix} Y & Z \end{bmatrix} \begin{bmatrix} R \\ \mathbf{0} \end{bmatrix} \;=\; YR,$$

where $Y \in \mathbb{C}^{m \times n}$, $Z \in \mathbb{C}^{m \times (m-n)}$, $R \in \mathbb{C}^{n \times n}$, and $\mathbf{0}$ is an $(m - n) \times n$ matrix of all zeros. We therefore have the following definition.

Definition 2.3.2. *The economy QR decomposition of a complex $m \times n$ matrix A ($m \geq n$) is*

$$A = YR,$$

where $Y \in \mathbb{C}^{m \times n}$ has orthonormal columns and $R \in \mathbb{C}^{n \times n}$ is upper triangular.

We again note that if A has full column rank then the columns of Y form an orthonormal basis for range(A).

IMPLEMENTING THE SBS ALGORITHM

The SBS algorithm was implemented using the QR factorisation because of its numerical stability. We predominantly use the economy QR factors (which we call Y and R, and sometimes refer to as the YR factors) as these are cheaper to compute than the full QR factorisation, especially if $m \gg n$.

The SBS algorithm begins with the entire matrix A and at each iteration, one row is deleted until only k remain (where $n < k < m$ is a user-defined parameter chosen before the algorithm begins). The row to be deleted at each iteration is the row that minimizes (2.20). Criterion (2.20) must be efficiently calculated at each iteration and this is the focus of the remainder of this subsection.

Initially, the index set for the SBS algorithm is $\mathcal{B} = \{1, \ldots, m\}$ so that $A_{\mathcal{B}} = A$. Recall Definition 2.3.2 and let $A_{\mathcal{B}} = YR$. Then

$$A_{\mathcal{B}}^H A_{\mathcal{B}} = R^H Y^H Y R = R^H R,$$

because $Y^H Y = I$, so $(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} = R^{-1} R^{-H}$, where $R^{-H} \equiv (R^H)^{-1}$. Consider the numerator of (2.20) and let $C = (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1}$. As C is Hermitian,

$$a_j C^2 a_j^H \;=\; (a_j C)(a_j C)^H \;=\; \|a_j C\|_2^2. \tag{2.21}$$

In order to determine the numerator of (2.20) we only need to calculate $a_j C$. In terms of the YR factors, this is

$$a_j C = a_j R^{-1} R^{-H}. \tag{2.22}$$

Rather than calculating (2.22) for each row of A separately, we can compute it for all rows of $A_{\mathcal{B}}$ at once. As a_j is a row of $A_{\mathcal{B}}$:

$$A_{\mathcal{B}} C = A_{\mathcal{B}} R^{-1} R^{-H} = Y(RR^{-1})R^{-H} = YR^{-H}.$$

Define

$$V \;=\; A_{\mathcal{B}} C \;=\; YR^{-H}. \tag{2.23}$$

Then each row vector $a_j C$ corresponds to a row of V. The SBS algorithm was implemented in MATLAB so V in (2.23) can be found easily using MATLAB's forward-slash command.

The values $\|a_j C\|^2$, for $j = 1, \ldots, |\mathcal{B}|$ in (2.21) are the Euclidean norms of the rows of the matrix V. To give the values $\|a_j C\|^2$ in a single vector, simply compute $V.^*\overline{V}$ (where $.^*$ is element-wise multiplication) and then sum along the rows of the resulting matrix.

A similar process is used to determine the denominator of (2.20) and proceeds as follows. The denominator is $1 - a_j C a_j^H$, where

$$a_j C a_j^H = a_j (R^H R)^{-1} a_j^H = (a_j R^{-1})(a_j R^{-1})^H.$$

To calculate the value $a_j C a_j^H$ for all rows of $A_\mathcal{B}$ at once, compute

$$\begin{aligned} A_\mathcal{B} C A_\mathcal{B}^H &= A_\mathcal{B} R^{-1} R^{-H} A_\mathcal{B}^H, \\ &= (A_\mathcal{B} R^{-1})(A_\mathcal{B} R^{-1})^H, \\ &= Y Y^H. \end{aligned} \tag{2.24}$$

(Note that $YY^H \neq I$). The diagonal entries of the $|\mathcal{B}| \times |\mathcal{B}|$ matrix YY^H, correspond to the values $a_j C a_j^H$ (while the off-diagonals are redundant). Rather than computing this large matrix and using only the diagonal entries, we instead compute the smaller $(|\mathcal{B}| \times n)$ matrix $Y.^*\overline{Y}$, and sum along the rows to give the vector whose elements are the values $a_j C a_j^H$.

Using (2.23) and (2.24), we see that the row to discard at each iteration of the SBS algorithm corresponds to

$$\min_j \quad \left(\sum_{i=1}^{|\mathcal{B}|} (V.^*\overline{V})_{ji} \right) ./ \left(\mathbf{1} - \sum_{i=1}^{|\mathcal{B}|} (Y.^*\overline{Y})_{ji} \right),$$

where $\mathbf{1}$ is a vector of ones and $./$ is element-wise division. Hence we can evaluate (2.20) efficiently using the economy QR factors.

The pseudocode for the SBS algorithm is given below.

ALGORITHM: SBS

---

1. **Input:** A, k and initialize $\mathcal{B} = \{1, \ldots, m\}$
2. **for** $l = 1$ to $(m - k)$ **do**
3. $\quad j^\star \leftarrow \min_j \dfrac{a_j (A_\mathcal{B}^H A_\mathcal{B})^{-2} a_j^H}{1 - a_j (A_\mathcal{B}^H A_\mathcal{B})^{-1} a_j^H}.$
4. $\quad \mathcal{B} \leftarrow \mathcal{B} \setminus \{j^\star\}$
5. **endfor**

Remark: At each iteration of the SBS algorithm the economy QR factors of $A_{\mathcal{B}}$ are required. The submatrix $A_{\mathcal{B}}$ is large, so determining the economy QR factors is an expensive process and the SBS algorithm could therefore benefit from an updating scheme. Currently MATLAB has a built-in command to perform a rank-1 update to the QR factors but not to the *economy* QR factors. Therefore in the SBS algorithm an economy QR rank-1 'down dating' scheme was also implemented, although the details are not described here. See, for example, [33], or a more recent discussion in [91].

2.3.2 THE SEQUENTIAL FORWARD SELECTION ALGORITHM

An alternative to the SBS algorithm is the Sequential Forward Selection (SFS) algorithm, which starts with an empty matrix and sequentially adds rows a_i from the matrix A until the submatrix $A_{\mathcal{B}}$ has k rows. Let $A_{\mathcal{B}}$ be the matrix that is being grown. Recall that a_i is a *row* vector. The Sherman-Morrison-Woodbury matrix inversion formula (2.18) gives the rank-1 update,

$$\text{trace}(A_{\mathcal{B}}^H A_B + a_i^H a_i)^{-1} = \text{trace}(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} - \frac{a_i (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-2} a_i^H}{1 + a_i (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_i^H}. \tag{2.25}$$

Again $\text{trace}(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1}$ is independent of row a_i, so to minimize expression (2.25) the SFS algorithm uses the criterion,

$$\max_i \ \frac{a_i (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-2} a_i^H}{1 + a_i (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_i^H}\,. \tag{2.26}$$

Therefore, the row to be added to $A_{\mathcal{B}}$ at each iteration of the SFS algorithm is the row that maximizes criterion (2.26). If a_i is the row to be added then the index set is updated to $\mathcal{B} \leftarrow \mathcal{B} \cup \{i\}$.

There is a difficulty here because $(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1}$ is not defined unless $A_{\mathcal{B}}$ has at least n rows. The question arises: "how do we select the first n rows?" A simple approach is to randomly select the first n rows from A to form $A_{\mathcal{B}}$, where $|\mathcal{B}| = n$. Then equation (2.25) is valid, and (2.26) can be used as the submatrix grows from n to k rows.

A better alternative is to note that when $A_{\mathcal{B}}$ has n rows, $\text{trace}(A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-1} = \text{trace}(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1}$ and that $(A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-1}$ *is* defined when $A_{\mathcal{B}}$ has fewer than n rows. Since the matrices $A_{\mathcal{B}}^H A_{\mathcal{B}}$ and $A_{\mathcal{B}} A_{\mathcal{B}}^H$ have the same non-zero eigenvalues, the expression (2.25) can be modified, as in [45]. In this case, the modified criterion is used to select the first n rows to add to the submatrix $A_{\mathcal{B}}$, and the original criterion (2.26) is used as the submatrix grows from n to k rows. We derive the modified criterion now.

Suppose that $|\mathcal{B}| < n$. Adding the row a_i to the matrix $A_{\mathcal{B}}$, leads to

$$(A_{\mathcal{B}} A_{\mathcal{B}}^H)_+ = \begin{bmatrix} A_{\mathcal{B}} \\ a_i \end{bmatrix} \begin{bmatrix} A_{\mathcal{B}}^H & a_i^H \end{bmatrix} = \begin{bmatrix} A_{\mathcal{B}} A_{\mathcal{B}}^H & A_{\mathcal{B}} a_i^H \\ a_i A_{\mathcal{B}}^H & a_i a_i^H \end{bmatrix}.$$

Furthermore, the matrix inverse is (see for example [5, p.67])

$$(A_{\mathcal{B}}A_{\mathcal{B}}^H)_+^{-1} = \frac{1}{\triangle}\begin{bmatrix} \left(\triangle I + (A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1}(A_{\mathcal{B}}a_i^H a_i A_{\mathcal{B}}^H)\right)(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} & -(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1}A_{\mathcal{B}}a_i^H \\ -a_i A_{\mathcal{B}}^H (A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} & 1 \end{bmatrix},$$

where

$$\triangle = a_i a_i^H - a_i A_{\mathcal{B}}^H (A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H.$$

The modified criterion relies on determining the trace of $(A_{\mathcal{B}}A_{\mathcal{B}}^H)_+^{-1}$. Recall that the trace of a scalar is the scalar itself and that the trace is invariant under cyclic permutations of matrices. Then

$$\begin{aligned} \text{trace}(A_{\mathcal{B}}A_{\mathcal{B}}^H)_+^{-1} &= \text{trace}\left((A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} + \frac{1}{\triangle}(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1}(A_{\mathcal{B}}a_i^H a_i A_{\mathcal{B}}^H)(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1}\right) + \frac{1}{\triangle} \\ &= \text{trace}(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} + \frac{1}{\triangle}\text{trace}\left((A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1}(A_{\mathcal{B}}a_i^H a_i A_{\mathcal{B}}^H)(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1}\right) + \frac{1}{\triangle} \\ &= \text{trace}(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} + \frac{1}{\triangle}\text{trace}\left((A_{\mathcal{B}}a_i^H a_i A_{\mathcal{B}}^H)(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-2}\right) + \frac{1}{\triangle}. \end{aligned}$$

By noticing that

$$(A_{\mathcal{B}}a_i^H a_i A_{\mathcal{B}}^H)(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-2} = (A_{\mathcal{B}}a_i^H)(a_i A_{\mathcal{B}}^H (A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-2})$$

is an outer product, and using the identity (for *column* vectors x and y)

$$\text{trace}(xy^H) = x^H y,$$

we can further simplify to

$$\begin{aligned} \text{trace}(A_{\mathcal{B}}A_{\mathcal{B}}^H)_+^{-1} &= \text{trace}(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} + \frac{1}{\triangle}\text{trace}\left(a_i A_{\mathcal{B}}^H (A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-2} A_{\mathcal{B}} a_i^H\right) + \frac{1}{\triangle} \\ &= \text{trace}(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} + \frac{1}{\triangle}\left(1 + a_i A_{\mathcal{B}}^H (A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-2} A_{\mathcal{B}} a_i^H\right). \end{aligned}$$

Therefore, the modified criterion for the SFS algorithm when the submatrix $A_{\mathcal{B}}$ has fewer than n rows is

$$\text{trace}(A_{\mathcal{B}}A_{\mathcal{B}}^H)_+^{-1} = \text{trace}(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} + \frac{1 + a_i A_{\mathcal{B}}^H (A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-2} A_{\mathcal{B}} a_i^H}{a_i a_i^H - a_i A_{\mathcal{B}}^H (A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H}. \tag{2.27}$$

The row a_i is chosen to minimize

$$\min_i \quad \frac{1 + a_i A_{\mathcal{B}}^H (A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-2} A_{\mathcal{B}} a_i^H}{a_i a_i^H - a_i A_{\mathcal{B}}^H (A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H}. \tag{2.28}$$

The SFS algorithm is therefore split into two optimization phases: phase 1 employs the modified formula (2.28) and runs until A_B is square. Phase 2 uses the original formula (2.26) and the algorithm proceeds until k rows have been selected.

IMPLEMENTING THE SFS ALGORITHM

The SFS algorithm begins with an empty matrix $A_{\mathcal{B}}$ and at each iteration, one row of A is added to the submatrix $A_{\mathcal{B}}$ until it contains k rows. If the submatrix $A_{\mathcal{B}}$ contains fewer than n rows, then the row to be added to the submatrix $A_{\mathcal{B}}$ at each iteration is the row that minimizes (2.28). If the submatrix has at least n rows, then the row to be added to the submatrix $A_{\mathcal{B}}$ at each iteration is the row that maximises (2.26).

The SFS algorithm requires two index sets. Let $\mathcal{B}$ denote the set of indices corresponding to the rows of A that have been selected and let $\mathcal{N}$ denote the set of row indices corresponding to the rows of A that have *not* been selected. Note that $\mathcal{B} \cup \mathcal{N} = \{1, 2, \ldots, m\}$ and $\mathcal{B} \cap \mathcal{N} = \{\emptyset\}$. Initially $\mathcal{B} = \{\emptyset\}$ and $\mathcal{N} = \{1, 2, \ldots, m\}$. Define the matrices $A_{\mathcal{B}}$ and $A_{\mathcal{N}}$ to be submatrices of A whose rows correspond to the index sets $\mathcal{B}$ and $\mathcal{N}$ respectively. Note that at each iteration, the row to be added to the submatrix $A_{\mathcal{B}}$ is selected from the submatrix $A_{\mathcal{N}}$.

There is a slight difficulty here because $\mathcal{B}$ (and $A_{\mathcal{B}}$) is initially empty, so that (2.28) is undefined at the first iteration. Hence equation (2.27) becomes

$$\min_i \ \operatorname{trace}(a_i a_i^H)^{-1}.$$

Because $a_i a_i^H$ is a scalar, the problem is reposed as

$$\max_i \ a_i a_i^H \ \equiv \ \max_i \ \|a_i\|^2 \, . \tag{2.29}$$

Expression (2.29) provides a way of selecting the initial row of the submatrix — it is chosen as the row of A with maximum Euclidean norm.

Now that (2.28) is initialised, we enter phase 1 of the SFS algorithm to choose a further $n-1$ rows to add to submatrix $A_{\mathcal{B}}$ according to (2.28). (That is, phase 1 runs until $A_{\mathcal{B}}$ is square.) As for the SBS algorithm, economy QR factors are used throughout the SFS algorithm. Let $A_{\mathcal{B}}^H = YR$ so that

$$(A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-1} \ = \ (R^H R)^{-1}.$$

The denominator of (2.28) is

$$a_i a_i^H - a_i A_{\mathcal{B}}^H (A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H. \tag{2.30}$$

The scalar value $a_i a_i^H$ for $i \in \mathcal{N}$ is required at each of the first n iterations of the SFS algorithm. Fortunately, this value has already been computed for every row of A before starting phase 1 (see (2.29)), so these values can be stored in a vector $a_{\mathcal{N}}$ say, and need not be computed again.

Now consider the product term in (2.30). The factors $A_{\mathcal{B}}^H = YR$ give

$$\begin{aligned} a_i A_{\mathcal{B}}^H (A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H \ &= \ a_i Y R (R^H R)^{-1} R^H Y^H a_i^H \\ &= \ a_i Y R R^{-1} R^{-H} R^H Y^H a_i^H \\ &= \ (a_i Y)(a_i Y)^H. \end{aligned}$$

Note that a_i is a row of $A_{\mathcal{N}}$ and define $V = A_{\mathcal{N}}Y$. The rows of the matrix V correspond to the row vectors a_iY (for $i = 1$ to $|\mathcal{N}|$). Following the same process as for the SBS algorithm, we can sum along the rows of the matrix $V.^*\overline{V}$ to obtain a vector whose entries are the values $a_iA_{\mathcal{B}}^H(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1}A_{\mathcal{B}}a_i^H$.

Consider now the product term in the numerator of fraction (2.28):

$$a_iA_{\mathcal{B}}^H(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-2}A_{\mathcal{B}}a_i^H \;=\; \|a_iA_{\mathcal{B}}^H(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1}\|_2^2.$$

The term $a_iA_{\mathcal{B}}^H(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1}$ can be computed for all rows of $A_{\mathcal{N}}$ at once via

$$A_{\mathcal{N}}A_{\mathcal{B}}^H(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} = A_{\mathcal{N}}YR(R^HR)^{-1} = A_{\mathcal{N}}YR^{-H} = VR^{-H}.$$

Define $W = VR^{-H}$, which is computed using the forwardslash '/' command in MATLAB.

Recall that $a_{\mathcal{N}}$ is the vector (of length $|\mathcal{N}|$) whose entries are the values $a_ia_i^H$ for $i \in \mathcal{N}$. The row to be added to the submatrix $A_{\mathcal{B}}$ at each of the first $n-1$ iterations of the SFS algorithm is the row that minimizes

$$\min_i \left(\mathbf{1} + \sum_{j=1}^{|\mathcal{N}|}(W.^*\overline{W})_{ij}\right) \;./\; \left(a_{\mathcal{N}} - \sum_{j=1}^{|\mathcal{N}|}(V.^*\overline{V})_{ij}\right).$$

After each iteration, $\mathcal{B} \leftarrow \mathcal{B} \cup \{i\}$ and $\mathcal{N} \leftarrow \mathcal{N} \setminus \{i\}$. Note that the set $\mathcal{B}$ must be updated before the set $\mathcal{N}$. Phase 1 of the SFS algorithm continues until $A_{\mathcal{B}}$ has n rows.

Phase 2 of the SFS algorithm then commences and the criterion to be optimized is (2.26). Note that at the start of phase 2, the matrix $A_{\mathcal{B}}$ has n rows so $\text{trace}(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} = \text{trace}(A_{\mathcal{B}}^HA_{\mathcal{B}})^{-1}$. Using economy QR factors,

$$A_{\mathcal{B}} = YR, \tag{2.31}$$

so

$$(A_{\mathcal{B}}^HA_{\mathcal{B}})^{-1} = (R^HR)^{-1}.$$

The product term in the denominator of (2.26) is

$$a_i(A_{\mathcal{B}}^HA_{\mathcal{B}})^{-1}a_i^H = \|a_iR^{-1}\|_2^2.$$

Now define $V = A_{\mathcal{N}}R^{-1}$. We sum across the rows of $V.^*\overline{V}$ to get the values $a_i(A_{\mathcal{B}}^HA_{\mathcal{B}})^{-1}a_i^H$ in a single vector.

From (2.31) the numerator of (2.26) is

$$a_i(A_{\mathcal{B}}^HA_{\mathcal{B}})^{-2}a_i^H = \|a_iR^{-1}R^{-H}\|_2^2.$$

Recall that $V = A_{\mathcal{N}}R^{-1}$ and let $W = VR^{-H}$. Summing across the rows of $W.^*\overline{W}$ gives a vector whose entries are the values $a_i(A_{\mathcal{B}}^HA_{\mathcal{B}})^{-2}a_i^H$.

The row to be added to the submatrix $A_{\mathcal{B}}$ at each iteration of (phase 2 of) the SFS algorithm is the row that maximizes

$$\max_i \left(\sum_{j=1}^{|\mathcal{N}|} (W.^* \overline{W})_{ij} \right) ./ \left(\mathbf{1} + \sum_{j=1}^{|\mathcal{N}|} (V.^* \overline{V})_{ij} \right).$$

The index sets $\mathcal{B}$ and $\mathcal{N}$ are updated in the same way as for Phase 1: $\mathcal{B} \leftarrow \mathcal{B} \cup \{i\}$ and $\mathcal{N} \leftarrow \mathcal{N} \setminus \{i\}$. The algorithm terminates when the submatrix $A_{\mathcal{B}}$ has k rows.

The pseudocode for the SFS algorithm is as follows.

ALGORITHM: SFS

---

1. **Input:** A, k and initialise $\mathcal{N} = \{1, \ldots, m\}$
2. Find $i^\star \leftarrow \max_i \ \|a_i\|_2^2$
3. $\mathcal{B} = \{i^\star\}$
4. $\mathcal{N} \leftarrow \{1, \ldots, m\} \setminus \{i^\star\}$
5. **for** $l = 2$ to n **do**
6. $\quad i^\star \leftarrow \min_i \ \dfrac{1 + a_i A_{\mathcal{B}}^H (A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-2} A_{\mathcal{B}} a_i^H}{a_i a_i^H - a_i A_{\mathcal{B}}^H (A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H}$
7. $\quad \mathcal{B} \leftarrow \mathcal{B} \cup \{i^\star\}$
8. $\quad \mathcal{N} \leftarrow \mathcal{N} \setminus \{i^\star\}$
9. **endfor**
10. **for** $l = n + 1$ to k **do**
11. $\quad i^\star \leftarrow \max_i \ \dfrac{a_i (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-2} a_i^H}{1 + a_i (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_i^H}$
12. $\quad \mathcal{B} \leftarrow \mathcal{B} \cup \{i^\star\}$
13. $\quad \mathcal{N} \leftarrow \mathcal{N} \setminus \{i^\star\}$
14. **endfor**

Remarks:

1. The economy QR factors of $A_{\mathcal{B}}^H$ are used for phase 1 of the SFS algorithm. As previously mentioned, MATLAB does not have a built-in code for updating the economy QR factors. In our implementation of the SFS algorithm we have included an updating process, although the details are omitted. See [33].

2. In phase 2 of the SFS algorithm, only R from the economy QR factors of $A_{\mathcal{B}}$ is actually used. Therefore, rather than using the economy QR factors, we could use the Cholesky factors in the computation of each row selection criterion instead. The advantage is that MATLAB has a built-in code for the computation of a rank-1 update to the Cholesky factors. Also, the work is significantly less, and adding rows is stable without Q or Y.

3. At step 2 of the pseudocode for the SFS algorithm, if all the rows have the same length, then $\max_i \|a_i\|_2^2$ is not unique. In this case the programming language's built-in sorting algorithm will decide the choice. (Note that MATLAB chooses the first instance.)

2.3.3 APPLICATIONS

Both the SBS and SFS algorithms are currently being used for a wide range of applications. Many medical applications make use of the SBS and SFS algorithms, in particular, magnetic resonance imaging (MRI). Using the sequential algorithms we determine a limited set of information to observe so that data collection times, and therefore overall scan times, can be significantly reduced. This is the basis of much current research; for example [46, 47, 48, 105, 106].

The algorithms are also used for feature selection in areas such as mammography [66, 97], malignancy in brain glioma [63], and the classification of uterine myoma [107]. Non-medical applications include the selection of samples acquired by a color digital camera [73] and sensor array configurations for optimal image reconstruction [87].

2.3.4 COMPARISON OF THE SBS AND SFS ALGORITHMS

Although the SBS and SFS algorithms are generally suboptimal, they both provide good practical results and they avoid the combinatorial problem.

The SBS algorithm has two drawbacks. First, it requires a higher computational effort than the SFS algorithm. This is simply because SBS operates on the full data matrix whereas SFS starts with the empty matrix. This means that the matrix factorizations are much more expensive for SBS than for SFS. In fact, for the SFS algorithm, at the first iteration when the submatrix $A_{\mathcal{B}}$ contains a single row, the YR factors are very simple. We have $Y = A_{\mathcal{B}}^H / \|A_{\mathcal{B}}\|_2$ and $R = \|A_{\mathcal{B}}\|_2$.

Second, in many imaging applications, for example MRI, it is highly desirable for the data collection process to run in parallel with the selection process. The SBS algorithm must run to completion so that the entire observation set is known, before data acquisition can begin.

On the other hand, if the SFS algorithm is used, once a row of A has been selected and added to the candidate set $A_{\mathcal{B}}$, the corresponding data point (in b) can immediately be acquired by the actual scanning device. This allows data processing to begin as the candidate matrix is still being grown, so the SFS algorithm has that practical advantage.

Neither algorithm is optimal although both generally perform well. (Upper bounds on the 'worst-case' trace value can be found in [80, 81].) In most cases the SBS algorithm returns a lower trace value corresponding to a 'better' set of candidate rows, while the SFS algorithm is faster. There is a trade-off between efficiency of the algorithm and accuracy of the solution.

2.4 EXTENDING THE EXISTING ALGORITHMS

One issue with the SBS and SFS algorithms that has been highlighted in [6, 84] is that there is no opportunity to improve the selected row subset, because any row deleted (SBS) or added (SFS) is not considered again. With this in mind, we consider how we could implement a row-exchange system so that a better row subset is selected.

2.4.1 A ROW-EXCHANGE CRITERION

Suppose we have selected a sub-matrix $A_{\mathcal{B}} \in \mathbb{C}^{k\times n}$ (where $k \geq n$). Let $M = A_{\mathcal{B}}^H A_{\mathcal{B}}$, which is square and has full rank. The aim is to improve the selected row subset $A_{\mathcal{B}}$ by exchanging a row from $A_{\mathcal{B}}$ with a row from $A_{\mathcal{N}}$. (That is, a row in $A_{\mathcal{B}}$ is replaced by a row from $A_{\mathcal{N}}$ so that the resulting submatrix $A_{\mathcal{B}'}$ gives a lower trace value.)

We consider how $\text{trace}(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1}$ changes when a row a is deleted from $A_{\mathcal{B}}$ and a row b from $A_{\mathcal{N}}$ is inserted. (The row b replaces the row a.) We are working toward an expression for the rank-2 update $\text{trace}(A_{\mathcal{B}}^H A_{\mathcal{B}} - a^H a + b^H b)^{-1}$. The matrix involved is

$$(M - a^H a + b^H b)^{-1} = (M + X^H Y)^{-1}, \tag{2.32}$$

where

$$X^H = \begin{bmatrix} a^H & | & b^H \end{bmatrix} \in \mathbb{C}^{n\times 2} \qquad \text{and} \quad Y = \begin{bmatrix} -a \\ b \end{bmatrix} \in \mathbb{C}^{2\times n}.$$

The Sherman-Morrison-Woodbury formula [5, p.67] gives

$$(M + X^H Y)^{-1} = M^{-1} - M^{-1}X^H(I + YM^{-1}X^H)^{-1}YM^{-1}. \tag{2.33}$$

Let $C = M^{-1}$. Taking the trace of (2.33) and using $\text{trace}(CDE) = \text{trace}(DEC)$ gives

$$\text{trace}(M + X^H Y)^{-1} = \text{trace}(C) - \text{trace}\left((I + YCX^H)^{-1}(YC^2X^H)\right).$$

Expressions for the matrices $(I + YCX^H)^{-1}$ and (YC^2X^H) are now derived. We begin with the matrix $(I + YCX^H)$, which can be expanded as

$$I + YCX^H \;=\; I + \begin{bmatrix} -a \\ b \end{bmatrix}\begin{bmatrix} Ca^H & Cb^H \end{bmatrix} \;=\; \begin{bmatrix} 1 - aCa^H & -aCb^H \\ bCa^H & 1 + bCb^H \end{bmatrix}.$$

Because $(I + YCX^H)$ is a 2×2 matrix, using the formula for the inverse of a 2×2 matrix gives

$$(I + YCX^H)^{-1} \;=\; \frac{1}{\Delta}\begin{bmatrix} 1 + bCb^H & aCb^H \\ -bCa^H & 1 - aCa^H \end{bmatrix},$$

where

$$\Delta = (1 - aCa^H)(1 + bCb^H) + (aCb^H)(bCa^H). \tag{2.34}$$

Now consider the matrix YC^2X^H, which can be expanded as

$$YC^2X^H = \begin{bmatrix} -a \\ b \end{bmatrix}\begin{bmatrix} C^2a^H & C^2b^H \end{bmatrix} \;=\; \begin{bmatrix} -aC^2a^H & -aC^2b^H \\ bC^2a^H & bC^2b^H \end{bmatrix}.$$

The matrix multiplication $(I + YCX^H)^{-1}(YC^2X^H)$ is

$$\frac{1}{\Delta}\begin{bmatrix} -(1 + bCb^H)(aC^2a^H) + aCb^HbC^2a^H & -(1 + bCb^H)(aC^2b^H) + aCb^HbC^2b^H \\ bCa^HaC^2a^H + (1 - aCa^H)bC^2a^H & bCa^HaC^2b^H + (1 - aCa^H)bC^2b^H \end{bmatrix},$$

and taking the trace gives the rank-2 update:

$$\text{trace}(M - a^Ha + b^Hb)^{-1} \;=\; \text{trace}(C) - \tag{2.35}$$

$$\frac{(aCb^H)(bC^2a^H) - (1 + bCb^H)(aC^2a^H) + (1 - aCa^H)(bC^2b^H) + (bCa^H)(aC^2b^H)}{\Delta}.$$

Therefore the criterion used to determine the rows to exchange is

$$\max_{a,b} \frac{(1 - aCa^H)(bC^2b^H) - (1 + bCb^H)(aC^2a^H) + 2|(aCb^H)(bC^2a^H)|}{\Delta}. \tag{2.36}$$

Remark: Note that substituting $a = 0$ and $b = a_i$ in (2.36) gives the criterion for the SFS algorithm (2.26). Similarly substituting $b = 0$ and $a = a_j$ in (2.36) gives (2.19), the criterion for the SBS algorithm.

2.4.2 THE SEQUENTIAL HYBRID SELECTION ALGORITHM

The row subset selected by either the SBS or the SFS algorithm is likely to be suboptimal. The Sequential Hybrid Selection (SHS) algorithm aims to improve the selected subset by including a row-exchange process.

There are many ways in which a row-exchange process based upon (2.35) can be implemented. We discuss one option here. The SFS algorithm is employed to grow the sub-matrix $A_{\mathcal{B}}$ until it has a predetermined number of k rows. Next, row exchanges are implemented according to (2.36) to improve the subset of selected rows. A row exchange should only take place when (2.36) is positive, thus reducing the trace. Therefore the algorithm terminates when all row

combinations give (2.36) ≤ 0.

There are two sub-matrices involved in this problem, one each for the forward and backward selection choices. Let the initial matrix be $A \in C^{m \times n}$ and $A_{\mathcal{B}}$ be the sub-matrix formed after k iterations of the SFS algorithm. At this point row interchanges are used. The row to be deleted from $A_{\mathcal{B}}$ is chosen from a sub-matrix with k rows. The row being added to $A_{\mathcal{B}}$ is selected from the sub-matrix $A_{\mathcal{N}}$, which has $(m-k)$ rows. Overall this gives $k \times (m-k)$ possible row combinations. So the benefits of the SHS algorithm do come with an associated computational cost.

In the following pseudocode for the SHS algorithm, a is a row of $A_{\mathcal{B}}$ with associated row index j and b is a row of $A_{\mathcal{N}}$ with associated index i. Note that at each iteration of the SHS algorithm, the index set $\mathcal{B}$ should be updated before the index set $\mathcal{N}$.

Algorithm: SHS

---

1. **Input:** A, k and initialize $\mathcal{B} = \{\emptyset\}$, $\mathcal{N} = \{1, \ldots, m\}$
2. **until** $|\mathcal{B}| = k$ **do**
3. SFS algorithm
4. **enduntil**
5. $i^\star, j^\star \leftarrow \max\limits_{a,b}$ (2.36)
6. **while** (2.36) > 0 **do**
7. $\mathcal{B} \leftarrow \mathcal{B} \cup \{i^\star\}$
8. $\mathcal{N} \leftarrow \mathcal{N} \cup \{j^\star\}$
9. $\mathcal{N} \leftarrow \mathcal{N} \setminus \{i^\star\}$
10. $\mathcal{B} \leftarrow \mathcal{B} \setminus \{j^\star\}$
11. **endwhile**

Remark: For each of the sequential algorithms, if every row in the matrix A has the same length then the initial choice of row is not unique. This situation arises, for example, if A is a Fourier matrix. The sorting algorithm will instead determine the initial row. A row exchange process using the SHS algorithm can help in this situation because the SHS algorithm can exchange the original row for a better one, if required.

2.5 A NEW SELECTION CRITERION

The subset selection process is to determine the 'best' submatrix, of a predetermined size, from the available data, by minimising the trace expression (2.17). Because the trace of a matrix is the sum of its eigenvalues, minimising (2.17) is equivalent to minimising the arithmetic mean of the eigenvalues of the matrix $(A^H A)^{-1}$. If instead, we consider minimising the geometric mean, this leads to minimising the product of the eigenvalues, or equivalently minimising the determinant of $(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1}$. This is in turn, equivalent to maximising the determinant of the matrix $(A_{\mathcal{B}}^H A_{\mathcal{B}})$.

Thus we propose the following alternative problem:

$$\max_{\mathcal{B}} \quad \det(A_{\mathcal{B}}^H A_{\mathcal{B}}) \tag{2.37}$$

over all index sets $\mathcal{B}$ containing k indices.

From problem (2.37), two new subset selection algorithms naturally arise from the question: "What happens to the determinant if a row is added to, or deleted from, the submatrix $A_{\mathcal{B}}$?" The remainder of this section concentrates on this question, and describes two new sequential algorithms that aim to optimize (2.37).

2.5.1 THE SEQUENTIAL DETERMINANT BACKWARD SELECTION ALGORITHM

A Sequential Backward Selection Algorithm based upon the determinant maximization problem (2.37) is now discussed. The idea is the same as for the SBS algorithm, where we begin with a matrix A and then sequentially delete rows until only k remain. Let $A_{\mathcal{B}}$ be the submatrix of A from which we are deleting rows. The problem is

$$\max_{j} \quad \det(A_{\mathcal{B}}^H A_{\mathcal{B}} - a_j^H a_j)\,. \tag{2.38}$$

Let $M = A_{\mathcal{B}}^H A_{\mathcal{B}}$. Then, using the identity $\det(I + x^H y) = (1 + xy^H)$ [5, p. 66], we have

$$\begin{aligned} \det(A_{\mathcal{B}}^H A_{\mathcal{B}} - a_j^H a_j) &= \det(M(I - M^{-1} a_j^H a_j)) \\ &= (\det M)(\det(I - (M^{-1} a_j^H) a_j)) \\ &= (\det M)(1 - a_j M^{-1} a_j^H)\,. \end{aligned} \tag{2.39}$$

As $\det M$ is independent of the choice of row a_j, the row to discard is the row which solves

$$\min_{j} \quad a_j (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_j^H\,. \tag{2.40}$$

Remarks:

1. Note that to maximize the term $(1 - a_j (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_j^H)$ in (2.39) we must subtract as little as possible so problem (2.40) is a minimization problem.

2. Note also that the expression $(1 - a_j (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_j^H)$ is equivalent to the denominator of (2.20).

We call the algorithm that optimizes (2.40) the Sequential Determinant Backward Selection (SDBS) algorithm.

IMPLEMENTING THE SDBS ALGORITHM

The SDBS algorithm begins with the full original matrix A and at each iteration, a row is deleted from the submatrix $A_{\mathcal{B}}$ until only k remain. The row that is deleted at each iteration is the row that minimizes (2.40).

As for the SBS algorithm, the economy QR factors were used to implement SDBS algorithm. Initially $\mathcal{B} = \{1, 2, \ldots, m\}$. Let $A_{\mathcal{B}} = YR$. Then

$$a_j(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_j^H = a_j(R^H R)^{-1} a_j^H = \|a_j R^{-1}\|_2^2.$$

Because a_j is a row of $A_{\mathcal{B}}$ we can compute $a_j R^{-1}$ for all rows of $A_{\mathcal{B}}$ through,

$$(A_{\mathcal{B}} R^{-1})(A_{\mathcal{B}} R^{-1})^H = YY^H \, . \tag{2.41}$$

Recall that $YY^H \neq I$. We are only interested in the diagonal terms of the matrix YY^H because these correspond to the values $a_j(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_j^H$. (The off-diagonal elements correspond to the terms $a_i(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_j^H$, which are not relevant here.) Therefore, instead of forming the matrix YY^H and taking the diagonal elements, we form the product $Y.^*\overline{Y}$ and take the sum along the rows to give the values $a_j(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_j^H$ for $j \in \mathcal{B}$ in a single vector. (This is equivalent to computing the row norms of the matrix Y.)

At each iteration of the SDBS algorithm, the expression

$$\min_j \quad \sum_{i=1}^{|\mathcal{B}|} (Y.^*\overline{Y})_{ji}$$

is computed to determine which row to delete from the submatrix $A_{\mathcal{B}}$.

The pseudocode for the SDBS algorithm is given below.

Algorithm: SDBS

---

1. **Input:** A, k and initialize $\mathcal{B} = \{1, \ldots, m\}$
2. **for** $l = 1$ to $(m - k)$ **do**
3. $\quad j^\star \leftarrow \min_j \quad a_j(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_j^H$
4. $\quad \mathcal{B} \leftarrow \mathcal{B} \setminus \{j^\star\}$
5. **endfor**

SOME RESULTS ON THE SDBS ALGORITHM

In this section we are interested in studying the growth of the determinant and trace as the SDBS algorithm progresses. Recall that the algorithm terminates when the submatrix $A_{\mathcal{B}}$ has size $k \times n$ where $k > n$. We assume the original matrix A has full rank.

Theorem 2.5.1. *If the original matrix $A \in \mathbb{C}^{m\times n}$ has full rank then at each iteration of the SDBS algorithm, the submatrix $A_{\mathcal{B}} \in \mathbb{C}^{|\mathcal{B}|\times n}$ has full rank.*

Proof. Suppose at the lth iteration of the SDBS algorithm, the matrix $A_{\mathcal{B}} \in \mathbb{C}^{|\mathcal{B}|\times n}$ with $n < k < |\mathcal{B}|$ has rank n. Choose a set $\mathcal{S}$ of indices corresponding to n linearly independent rows of $A_{\mathcal{B}}$. Note that $\mathcal{S} \subset \mathcal{B}$ and $|\mathcal{S}| < |\mathcal{B}|$. The SDBS algorithm now selects a row to delete from the index set $\mathcal{B}$ while optimising (2.38). Suppose the SDBS algorithm selects a row a_j where $j \notin \mathcal{S}$. Let $\mathcal{B}' = \mathcal{B} \setminus \{j\}$. Then $A_{\mathcal{B}'}$ still has n linearly independent rows and therefore has rank n. In this case

$$\det(A_{\mathcal{B}'}^H A_{\mathcal{B}'}) = \det(A_{\mathcal{B}}^H A_{\mathcal{B}} - a_i^H a_i) > 0 \,.$$

Since the SDBS algorithm aims to maximize criterion (2.38), it will never select a row for which $\det(A_{\mathcal{B}'}^H A_{\mathcal{B}'}) = 0$. But

$$\det(A_{\mathcal{B}'}^H A_{\mathcal{B}'}) = 0 \quad \Leftrightarrow \quad \operatorname{rank}(A_{\mathcal{B}'}) < n \,.$$

Therefore, at each iteration of the SDBS algorithm, the submatrix $A_{\mathcal{B}}$ has full rank. □

Lemma 2.5.1. *The sequence of determinant values found at each iteration of the SDBS algorithm is positive and strictly decreasing.*

Proof. From the previous Theorem, when the algorithm terminates, $A_{\mathcal{B}}$ has full rank so at each iteration of the SDBS algorithm, $\det(A_{\mathcal{B}}^H A_{\mathcal{B}}) > 0$. Subsequently we have that $\det(A_{\mathcal{B}}^H A_{\mathcal{B}} - a_j^H a_j) > 0$. The matrix $A_{\mathcal{B}}^H A_{\mathcal{B}}$ is positive definite so $(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1}$ exists. Therefore $a_j (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_j^H > 0$. Using $\det(I - DE) = \det(I - ED)$ (which is Sylvester's Determinant identity [5, p.66]) gives

$$\det(A_{\mathcal{B}}^H A_{\mathcal{B}} - a_j^H a_j) = \det(A_{\mathcal{B}}^H A_{\mathcal{B}})(1 - a_j (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_j^H) < \det(A_{\mathcal{B}}^H A_{\mathcal{B}}).$$

□

As a consequence of the previous Lemma we also have the following.

Lemma 2.5.2. *If a_j is a row of $A_{\mathcal{B}}$ then $0 < a_j (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_j^H < 1$.*

Proof. This follows from (2.41),

$$a_j (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_j^H = \|a_j R^{-1}\|_2^2 = \|y_j\|_2^2,$$

where y_j is a row of Y (part of a unitary matrix). □

2.5.2 THE SEQUENTIAL DETERMINANT FORWARD SELECTION ALGORITHM

We now introduce a variation of the SFS algorithm that uses the determinant criterion (2.37). We call this the Sequential Determinant Forward Selection (SDFS) algorithm. As for the SFS algorithm, the SDFS algorithm begins with an empty submatrix and sequentially adds a row at each iteration until a total of k rows have been chosen.

There are two index sets for this algorithm, $\mathcal{B}$ and $\mathcal{N}$. The set $\mathcal{B}$ corresponds to the row indices of A that have been selected, while the set $\mathcal{N}$ corresponds to the set of row indices not yet selected. Define $A_{\mathcal{B}}$ and $A_{\mathcal{N}}$ to be the submatrices of A whose rows correspond to the index sets $\mathcal{B}$ and $\mathcal{N}$ respectively. Note that $A_{\mathcal{B}}$ is the submatrix being grown, while $A_{\mathcal{N}}$ is the submatrix containing the remaining row choices.

To determine a selection criterion for the SDFS algorithm we investigate what happens to the determinant when a new row is added to $A_{\mathcal{B}}$. Consider the following derivation (which is similar to that in (2.39)):

$$\begin{aligned} \det(A_{\mathcal{B}}^H A_{\mathcal{B}} + a_i^H a_i) &= \det(M(I + M^{-1} a_i^H a_i)) \\ &= (\det M)\det(I + (M^{-1} a_i^H) a_i) \\ &= (\det M)(1 + a_i M^{-1} a_i^H). \end{aligned} \tag{2.42}$$

The term $\det(A_{\mathcal{B}}^H A_{\mathcal{B}})$ is independent of the choice of row a_i, so the expression to be optimized at each iteration of the SDFS algorithm is

$$\max_i \quad a_i (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_i^H. \tag{2.43}$$

This forms the basis of the Sequential Determinant Forward Selection algorithm.

As for the SFS algorithm, (2.43) is invalid when $A_{\mathcal{B}}$ has fewer than n rows because $A_{\mathcal{B}}^H A_{\mathcal{B}}$ is singular. For this reason we modify the determinant criterion to give a selection criterion that holds when $|\mathcal{B}| < n$. We use the same approach as for the SFS algorithm and consider a rank-1 update to $\det(A_{\mathcal{B}} A_{\mathcal{B}}^H)$ as $A_{\mathcal{B}}$ grows from 0 to n rows.

A modified determinant forward selection criterion is derived now. A row a_i of $A_{\mathcal{N}}$ is to be added to the sub-matrix $A_{\mathcal{B}}$ to give

$$(A_{\mathcal{B}} A_{\mathcal{B}}^H)_+ = \begin{bmatrix} A_{\mathcal{B}} \\ a_i \end{bmatrix} \begin{bmatrix} A_{\mathcal{B}}^H & a_i^H \end{bmatrix} = \begin{bmatrix} A_{\mathcal{B}} A_{\mathcal{B}}^H & A_{\mathcal{B}} a_i^H \\ a_i A_{\mathcal{B}}^H & a_i a_i^H \end{bmatrix}. \tag{2.44}$$

The determinant of (2.44) [5, p. 67] is

$$\det(A_{\mathcal{B}} A_{\mathcal{B}}^H)_+ = \det(A_{\mathcal{B}} A_{\mathcal{B}}^H)(a_i a_i^H - a_i A_{\mathcal{B}}^H (A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H). \tag{2.45}$$

Note that the term $\det(A_{\mathcal{B}} A_{\mathcal{B}}^H)$ is independent of row i so the expression to be optimised at each of the first n iterations of the SDFS algorithm is

$$\max_i \quad a_i a_i^H - a_i A_{\mathcal{B}}^H (A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H \ . \tag{2.46}$$

For the first n iterations of the SDFS algorithm the modified criterion (2.46) is used to grow the submatrix $A_{\mathcal{B}}$ until it has n rows. When $A_{\mathcal{B}}$ is square, $\det(A_{\mathcal{B}}^H A_{\mathcal{B}}) = \det(A_{\mathcal{B}} A_{\mathcal{B}}^H)$. Then, the original criterion (2.43) is used as the submatrix $A_{\mathcal{B}}$ grows from n to k rows.

Remarks:

1. Notice that the term $1 + a_i(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_i^H$ in (2.42) is equivalent to the denominator of the SFS criterion (2.26).

2. Notice that the term $a_i a_i^H - a_i A_{\mathcal{B}}^H (A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H$ in (2.46) is equivalent to the denominator of the SFS modified criterion (2.28).

IMPLEMENTING THE SDFS ALGORITHM

The SDFS algorithm requires two index sets. Let $\mathcal{B}$ denote the set of indices corresponding to the rows of A selected by the SDFS algorithm. Let $\mathcal{N}$ denote the set of row indices corresponding to the rows of A that have *not* been selected. Note that $\mathcal{B} \cup \mathcal{N} = \{1, 2, \ldots, m\}$ and $\mathcal{B} \cap \mathcal{N} = \{\emptyset\}$. Initially $\mathcal{B} = \{\emptyset\}$ and $\mathcal{N} = \{1, 2, \ldots, m\}$. Define the matrices $A_{\mathcal{B}}$ and $A_{\mathcal{N}}$ to be submatrices of A whose rows correspond to the index sets $\mathcal{B}$ and $\mathcal{N}$ respectively.

There is a slight difficulty here because $\mathcal{B}$ (and subsequently $A_{\mathcal{B}}$) is initially empty, so the (2.46) is initially undefined. In this case (2.46) becomes

$$\max_i \ \det(a_i a_i^H) \quad \equiv \quad \max_i \ a_i a_i^H$$

(because $a_i a_i^H$ is a scalar). This provides a way of selecting the initial row for the submatrix—it is chosen as the row of A with maximum Euclidean norm. Let $a_{\mathcal{N}}$ denote the vector whose ith entry corresponds to the Euclidean norm of the ith row of the matrix A—the value $a_i a_i^H$.

When $|\mathcal{B}| < n$, the SDFS modified criterion (2.46) can be efficiently calculated by making use of the YR factors. Let $A_{\mathcal{B}}^H = YR$. The product term in the modified criterion (2.45) becomes

$$\begin{aligned} a_i A_{\mathcal{B}}^H (A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H &= a_i A_{\mathcal{B}}^H (R^H R)^{-1} A_{\mathcal{B}} a_i^H \\ &= (a_i A_{\mathcal{B}}^H R^{-1})(a_i A_{\mathcal{B}}^H R^{-1})^H \\ &= (a_i Y)(a_i Y)^H . \end{aligned} \tag{2.47}$$

The term (2.47) can be calculated for each row a_i of $A_{\mathcal{N}}$ simultaneously. Let $V = A_{\mathcal{N}} Y$. Then criterion (2.46) becomes

$$\max_i \quad a_{\mathcal{N}} - \sum_{j=1}^{|\mathcal{B}|} (V .^* \overline{V})_{ij}. \tag{2.48}$$

The row to be added to the submatrix $A_{\mathcal{B}}$ at each iteration of the SDFS algorithm is the row that minimizes (2.48). After each iteration, $\mathcal{B} \leftarrow \mathcal{B} \cup \{i\}$ and $\mathcal{N} \leftarrow \mathcal{N} \setminus \{i\}$. The index set $\mathcal{B}$ should be updated before the set $\mathcal{N}$ in practice. Phase 1 of the SDFS algorithm continues as

$A_{\mathcal{B}}$ grows from 0 to n rows.

Phase 2 commences once $|\mathcal{B}| = n$ and we now consider how the economy QR factors of $A_{\mathcal{B}}$ can be used to efficiently compute (2.43). Note that a_i is a row of $A_{\mathcal{N}}$ and let $A_{\mathcal{B}} = YR$. The term $a_i(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_i^H$ in equation (2.43) can be computed for all rows of $A_{\mathcal{N}}$ simultaneously through

$$A_{\mathcal{N}}(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} A_{\mathcal{N}}^H \;=\; (A_{\mathcal{N}} R^{-1})(A_{\mathcal{N}} R^{-1})^H.$$

Let $V = A_{\mathcal{N}} R^{-1}$. The diagonal elements in the matrix VV^H correspond to the values $a_i(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_i^H$ so we compute

$$\max_i \quad \sum_{j=1}^{|\mathcal{B}|} (V.^* \overline{V})_{ij}. \tag{2.49}$$

The pseudocode for the SDFS algorithm is given below.

Algorithm: SDFS

---

1. **Input:** A, k and initialize $\mathcal{N} = \{1, \ldots, m\}$
2. Find $i^\star \leftarrow \max_i \; a_i a_i^H$
3. $\mathcal{B} = \{i^\star\}$
4. $\mathcal{N} \leftarrow \{1, \ldots, m\} \setminus \{i\}$
5. **for** $l = 2$ to n **do**
6. $\quad i^\star \leftarrow \max_i \quad a_i a_i^H - a_i A_{\mathcal{B}}^H (A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H$
7. $\quad \mathcal{B} \;\leftarrow\; \mathcal{B} \cup \{i^\star\}$
8. $\quad \mathcal{N} \;\leftarrow\; \mathcal{N} \setminus \{i^\star\}$
9. **endfor**
10. **for** $l = n + 1$ to k **do**
11. $\quad i^\star \leftarrow \max_i \quad a_i (A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_i^H$
12. $\quad \mathcal{B} \;\leftarrow\; \mathcal{B} \cup \{i^\star\}$
13. $\quad \mathcal{N} \;\leftarrow\; \mathcal{N} \setminus \{i^\star\}$
14. **endfor**

Remark: As for the SFS algorithm, the 'Y' of the YR factors is not needed for the computation of the expression (2.49). Alternately, we could use the Cholesky decomposition of the matrix $A_{\mathcal{B}}^H A_{\mathcal{B}}$, rather than computing the YR decomposition of $A_{\mathcal{B}}$.

SOME RESULTS ON THE SDFS ALGORITHM

In this section we are interested in studying the growth of the determinant and trace as the SDFS algorithm progresses. We initially consider the case $|\mathcal{B}| \geq n$ before turning our attention to the case $|\mathcal{B}| < n$.

The following two results assume that $|\mathcal{B}| \geq n$ and that the corresponding matrix $A_{\mathcal{B}}$ (and the original matrix A) have full column rank. Recall that the algorithm terminates when the submatrix $A_{\mathcal{B}}$ has size $k \times n$ where $k > n$.

Lemma 2.5.3. *The sequence of determinant values found at each iteration of the SDFS algorithm using criterion (2.43) form a positive and increasing sequence.*

Proof. Because $A_{\mathcal{B}}^H A_{\mathcal{B}}$ is a positive definite matrix $\det(A_{\mathcal{B}}^H A_{\mathcal{B}}) > 0$, $(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1}$ exists and is also positive definite. For any row a_i of A, $a_i(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_i^H > 0$ and

$$\det(A_{\mathcal{B}}^H A_{\mathcal{B}} + a_i^H a_i) \;>\; (\det A_{\mathcal{B}}^H A_{\mathcal{B}})(1 + a_i(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_i^H) \;=\; \det(A_{\mathcal{B}}^H A_{\mathcal{B}}).$$

□

We also have the following result, which shows the growth of the trace using the determinant based selection criterion.

Lemma 2.5.4. *The sequence of trace values found at each iteration of the SDFS algorithm using criterion (2.43) form a positive and decreasing sequence.*

Proof. Let $a_i \neq 0$ be a row of A. From the previous result, $1 + a_i(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_i^H > 0$. Furthermore, if B is a symmetric positive definite matrix, then so too is B^2, which implies that $\|(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_i^H\|_2^2 > 0$. This gives

$$\frac{a_i(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-2} a_i^H}{1 + a_i(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_i^H} > 0\,.$$

Therefore

$$\operatorname{trace}(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} > \operatorname{trace}(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} - \frac{a_i(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-2} a_i^H}{1 + a_i(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1} a_i^H} = \operatorname{trace}(A_{\mathcal{B}}^H A_{\mathcal{B}} + a_i^H a_i)^{-1}\,.$$

This means that the sequence of trace values found at each iteration of the SDFS algorithm form a decreasing sequence. Furthermore, if $\mathcal{B} = \{1, 2, \ldots, m\}$, then $A_{\mathcal{B}} = A$ and $\operatorname{trace}(A^H A)^{-1} > 0$ because A has full rank, so the sequence of trace values generated is positive. □

This is an interesting result because even though we are selecting rows according to a criterion based upon the determinant, the original trace criterion is still being decreased.

We now derive similar results for the SDFS algorithm using the modified criterion (2.45), used when the submatrix $A_{\mathcal{B}}$ has fewer than n rows ($|\mathcal{B}| < n$).

We first show that at each of the first n iterations of the SDFS algorithm, the rank of the submatrix $A_{\mathcal{B}}$ increases.

Theorem 2.5.2. *At each of the first n iterations of the SDFS algorithm using the modified determinant criterion (2.46), rank$(A_{\mathcal{B}}) = |\mathcal{B}|$.*

Proof. The proof follows by induction. Suppose $|\mathcal{B}| = 1$, then $\text{rank}(A_{\mathcal{B}}) = 1 = |\mathcal{B}|$ so the result holds. Assume that at the $(i-1)$th iteration of the SDFS algorithm, the submatrix $A_{\mathcal{B}}$ has $\text{rank}(A_{\mathcal{B}}) = |\mathcal{B}| = i - 1$ (where $1 \leq (i-1) \leq n$). Now at the ith iteration, the determinant modified criterion is

$$\det(A_{\mathcal{B}} A_{\mathcal{B}}^H)_+ \;=\; \det(A_{\mathcal{B}} A_{\mathcal{B}}^H)(a_i a_i^H - a_i A_{\mathcal{B}}^H (A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H) \; .$$

Let $P = A_{\mathcal{B}}^H (A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}}$. Then $P = P^H$ and $P^2 = P$ so P is a projection matrix. Let a_i be a row of the matrix A to be added to the submatrix $A_{\mathcal{B}}$. We have two cases.

1. Suppose that a_i is a linear combination of the rows of $A_{\mathcal{B}}$. Then

   $$\|a_i P\| = \|a_i\| \quad \Rightarrow \quad \|a_i P\|^2 = \|a_i\|^2 \; ,$$

   and

   $$a_i a_i^H - a_i A_{\mathcal{B}}^H (A_{\mathcal{B}} A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H = a_i a_i^H - (a_i P)(a_i P)^H = 0.$$

   so $\det(A_{\mathcal{B}} A_{\mathcal{B}}^H)_+ = 0$.

2. Suppose that a_i is not a linear combination of the rows of $A_{\mathcal{B}}$. Then

   $$\|a_i P\|^2 < \|a_i\|^2 \quad \Rightarrow \quad a_i a_i^H - (a_i P)(a_i P)^H > 0.$$

   so $\det(A_{\mathcal{B}} A_{\mathcal{B}}^H)_+ > 0$.

Because A has rank n, there is always at least one row a_i that is independent of the rows of $A_{\mathcal{B}}$ (because $|\mathcal{B}| < n$). The SDFS algorithm will select such a row because it is maximising $\det(A_{\mathcal{B}} A_{\mathcal{B}}^H)_+$ and any independent row will give $\det(A_{\mathcal{B}} A_{\mathcal{B}}^H)_+ > 0$ while a dependent row would give 0. The index set $\mathcal{B}$ is updated as $\mathcal{B} \leftarrow \mathcal{B} \cup \{i\}$ and therefore $\text{rank}(A_{\mathcal{B}}) = |\mathcal{B}| = i$. □

We can also show how the trace evolves using the modified determinant criterion (again $|\mathcal{B}| < n$). Recall that $(A_{\mathcal{B}} A_{\mathcal{B}}^H)_+$ denotes the submatrix $A_{\mathcal{B}}$ with the row a_i added to it.

Lemma 2.5.5. *Let $A_{\mathcal{B}}$ denote the matrix with rank$(A_{\mathcal{B}}) < n$, formed by selecting rows of the observation matrix A according to the modified selection criterion (2.45). Then the values of the trace found at each iteration of the SDFS algorithm using the criterion (2.45) form a positive, increasing sequence.*

Proof. Because $A_{\mathcal{B}}A_{\mathcal{B}}^H$ is a positive definite matrix, $0 < \text{trace}(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1}$. Similarly, $0 < \text{trace}(A_{\mathcal{B}}A_{\mathcal{B}}^H)_+^{-1}$. By previous arguments, $a_i a_i^H - a_i A_{\mathcal{B}}^H (A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H > 0$ and $(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-2}$ is a positive definite matrix, so

$$\begin{aligned} \text{trace}(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} &< \text{trace}(A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} + \frac{1 + a_i A_{\mathcal{B}}^H (A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-2} A_{\mathcal{B}} a_i^H}{a_i a_i^H - a_i A_{\mathcal{B}}^H (A_{\mathcal{B}}A_{\mathcal{B}}^H)^{-1} A_{\mathcal{B}} a_i^H} \\ &= \text{trace}(A_{\mathcal{B}}A_{\mathcal{B}}^H)_+^{-1} . \end{aligned}$$

□

Remark: In fact, Lemmas 2.5.1 to 2.5.5 can also be proven as a consequence of Cauchy's interlace theorem for eigenvalues of symmetric matrices. See for example [50, p. 396].

2.5.3 A DETERMINANT ROW-EXCHANGE CRITERION

As for the trace criterion, we can derive a determinant row-exchange criterion. Suppose $|\mathcal{B}| > n$. Let $M = A_{\mathcal{B}}^H A_{\mathcal{B}}$, $C = M^{-1}$ and with the notation of Section 2.4,

$$\det(M - a^H a + b^H b) = \det(M + X^H Y) = \det(I + X^H Y M^{-1}) \det M.$$

Since also $\det(I + DE) = \det(I + ED)$ [5], we have

$$\det(M - a^H a + b^H b) = \det(I + YCX^H) \det M = \Delta \det M, \tag{2.50}$$

where Δ is defined in (2.34). Expression (2.50) is the rank-2 update to the determinant. Hence, the determinant row-exchange criterion is

$$\max_{a,b} \Delta \;=\; \max_{a,b} (1 - aCa^H)(1 + bCb^H) + (aCb^H)(bCa^H). \tag{2.51}$$

The rows a and b are chosen to maximize Δ. A row exchange should only take place if $\Delta > 1$, thus increasing the determinant.

A special case of equation (2.50) occurs when the sub-matrix $A_{\mathcal{B}}$ is square ($k = n$) and is therefore nonsingular. Because a is a row of $A_{\mathcal{B}}$ it follows that $aA_{\mathcal{B}}^{-1}$ is a row of the identity matrix and hence $aCa^H = 1$. The expression for Δ now simplifies to

$$\Delta' = (aCb^H)(bCa^H) = \left|aCb^H\right|^2 . \tag{2.52}$$

In this case we have

$$\det(M - a^H a + b^H b) \;=\; (\det M) \left|aCb^H\right|^2 ,$$

so the determinant row-exchange criterion, when the submatrix $A_{\mathcal{B}}$ is square, is

$$\max_{a,b} \left|aCb^H\right|^2 . \tag{2.53}$$

The rows a and b are chosen to maximize (2.53) and a row exchange will take place if the value $\Delta' > 1$.

2.5.4 THE SEQUENTIAL DETERMINANT HYBRID SELECTION ALGORITHM

The Sequential Determinant Forward Selection algorithm can be combined with the determinant row-exchange criterion (2.50) to give a Sequential Determinant Hybrid Selection algorithm. This is also a row-exchange system but is much simpler than that described in Section 2.4 as we are only computing the denominator of expression (2.36), namely Δ.

A row-exchange algorithm based upon the determinant-exchange criterion proceeds as follows: the SDFS algorithm is used to grow the sub-matrix from 0 to k rows to give a submatrix $A_{\mathcal{B}}$ of size $k \times n$. At this point row interchanges are implemented to maximize (2.50). The row-exchange process continues until $\Delta \leq 1$, at which point further exchanges do not increase the determinant and the algorithm terminates.

In the following pseudocode for the SDHS algorithm, let a be a row of $A_{\mathcal{B}}$ that is being deleted (with associated row index j) and let b be a row of $A_{\mathcal{N}}$ that is being added to the submatrix $A_{\mathcal{B}}$ (with associated row index i).

ALGORITHM: SDHS

---

1. **Input:** A, k and initialize $\mathcal{B} = \{\emptyset\}$, $\mathcal{N} = \{1, \ldots, m\}$
2. **until** $|\mathcal{B}| = k$ **do**
3. SDFS algorithm
4. $i^\star, j^\star \leftarrow \max\limits_{a,b}$ (2.50)
5. **while** (2.50) > 1 **do**
6. $\mathcal{B} \leftarrow \mathcal{B} \cup \{i^\star\}$
7. $\mathcal{N} \leftarrow \mathcal{N} \cup \{j^\star\}$
8. $\mathcal{N} \leftarrow \mathcal{N} \setminus \{i^\star\}$
9. $\mathcal{B} \leftarrow \mathcal{B} \setminus \{j^\star\}$
10. **endwhile**

2.5.5 COMPARISON OF THE TRACE AND DETERMINANT FORMULATIONS

There are surprising parallels between the trace and determinant formulations. The observation subset criteria based upon the determinant are the denominators of the corresponding trace based criteria. For example, (2.51) is the denominator of (2.36).

We also see that the exchange criteria become the regular forward and backward selection criteria when either $a_i = 0$ or $a_j = 0$ respectively.

The determinant criteria are simpler expressions than the corresponding trace expressions because the maximization process does not rely so heavily on knowledge of the matrix inverse. Because of this the formulae can be computed more quickly, thereby leading to faster subset selection algorithms.

2.6 NUMERICAL EXPERIMENTS

In this section we present numerical results obtained using the algorithms described in this chapter. All algorithms were coded in MATLAB and the numerical examples were tested on an Intel Xeon 3.2GHz processor with 4GB of memory, under Linux.

The first experiment we present is that described in Reeves and Heck [79] and aims to show that our implementation of the algorithms are working as demonstrated in that paper. The problem is analogous to an MRI reconstruction where region of support information is available. Consider a 28-point 1D signal with a region of support given by the entries of the vector $n_l = [0, 1, \ldots, 9, 20, \ldots, 24]$. Let A be the 28×15 partial Fourier matrix with entries described by $a_{k,l} = \exp(-i2\pi k n_l/28)$. A plot of the results is shown in Figure 2.1.

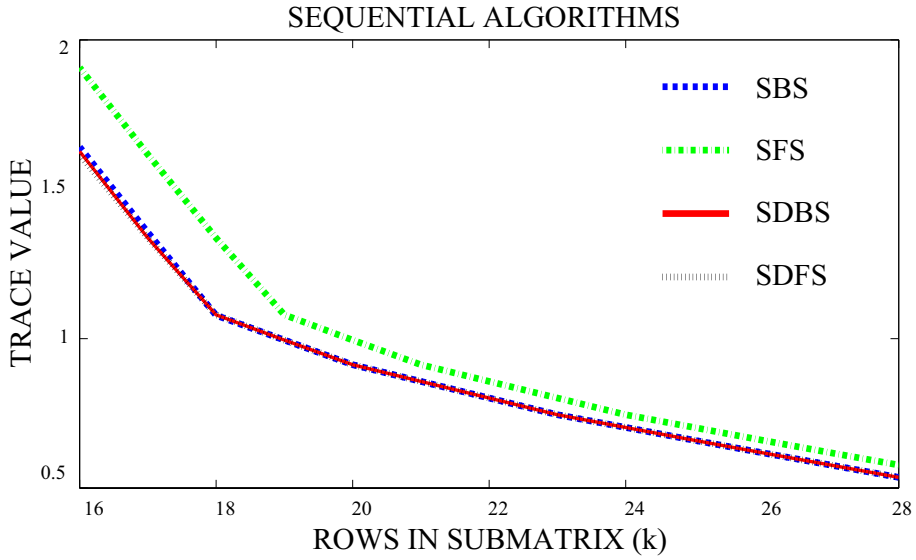


Figure 2.1: A comparison of the algorithms tested on data described by Reeves and Heck [79]. All the algorithms are performing as expected.

For this experiment Figure 2.1 presents results for subsets of size $|\mathcal{B}| = n = 16$ to $|\mathcal{B}| = m = 28$ because the Sequential Backward Selection Criterion (for both the SBS and SDBS algorithms) is undefined for fewer than n rows. The SFS and SDFS algorithms ran as usual (from $|\mathcal{B}| = 0$ to $|\mathcal{B}| = m$) using the appropriate modified selection criterion, although the trace values for subsets smaller than n are not displayed in the figure.

The results show that each algorithm is performing as demonstrated in [79], so we have confidence that the results presented in this section are fair. Note that in [79] they refer to the Mean Square Error (MSE). By equation (2.16) this is equivalent to the value trace$(A_{\mathcal{B}}^H A_{\mathcal{B}})^{-1}$, which is in turn equivalent to the value of 'tmetric' in [7, 8].

2.6.1 TRACE (MSE) COMPARISONS

In this subsection we run the algorithms on Complex and Fourier submatrix examples, to consider how the trace changes as rows are selected, and added, to the submatrix $A_{\mathcal{B}}$. Note that the Backward Selection Criterion is undefined for fewer than n rows. For the SHS algorithm, the SFS algorithm selects the new row to be added to the submatrix at each iteration, and a maximum of two row-exchanges are allowed at each iteration.

A RANDOM COMPLEX MATRIX EXAMPLE

The algorithms were tested on a 1000×200 matrix with randomly generated (complex) entries. The results are shown in Figure 2.2.

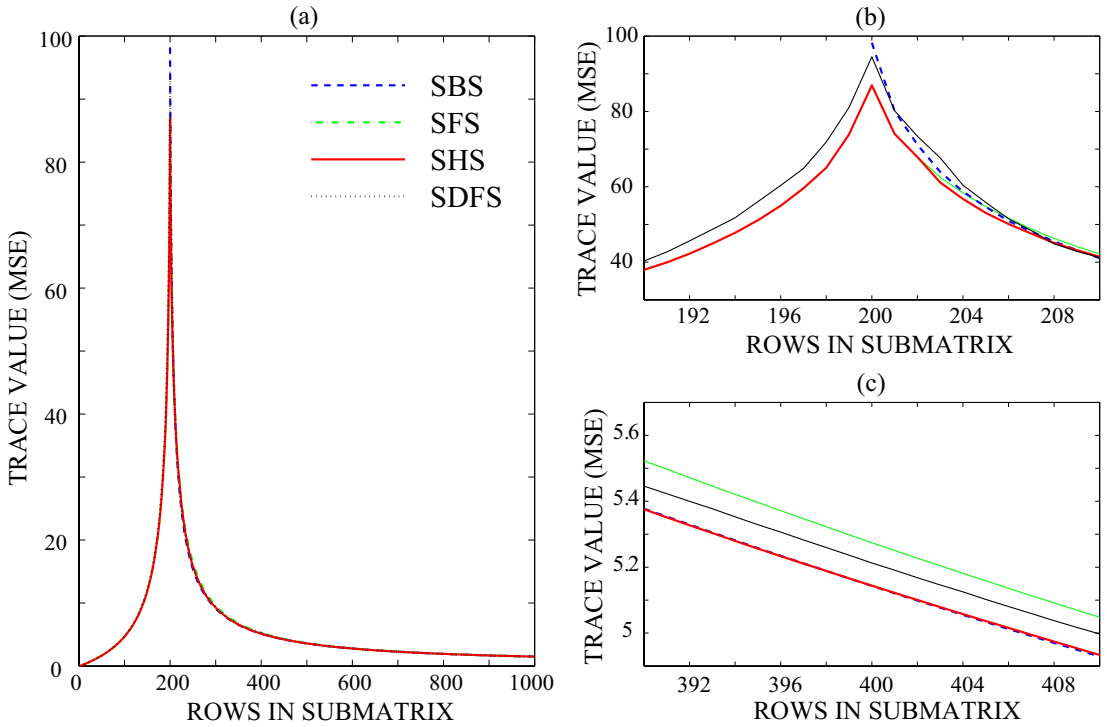


Figure 2.2: This figure shows the evolution of the trace as more rows are added to the submatrix $A_{\mathcal{B}}$. (a) shows the trace as the submatrix is grown from 0 to 1000 rows. (b) shows rows 190 to 210, around the peak in part (a). This is when the submatrix is square. (c) is zoomed in when the submatrix is larger than square (overdetermined).

Figure 2.2 shows the evolution of the trace as more rows are added to the submatrix. We see that when fewer rows than columns have been selected, the trace value increases until it peaks when the submatrix is square. (Note that the SBS criterion is not defined for a

submatrix with fewer than n rows, which is why that plot is only shown for more than 200 rows.) Then, as expected, the value of the trace reduces monotonically. The algorithms are behaving similarly and converging to the same value of the trace as more rows are added to the submatrix.

We also see that even though the subset of n rows chosen by the SDFS algorithm has a higher trace value, as more rows are added to the submatrix the SDFS algorithm chooses a better subset than the SFS algorithm. This is interesting because it shows that even if one algorithm chooses a better size n row subset, this is not an indicator that it will from then on continue to have a lower trace value. Figure 2.2 also shows that the SDFS algorithm is performing at least as well as the SFS algorithm, which is somewhat surprising because the SDFS algorithm is choosing rows based on the determinant criterion yet it still also gives as good a value of the trace-based criterion.

A FOURIER MATRIX EXAMPLE

A Discrete Fourier Transform matrix of size 1000×1000 was constructed and 200 columns were randomly selected to form a partial Fourier matrix. The results of the experiment are shown in Figure 2.3.

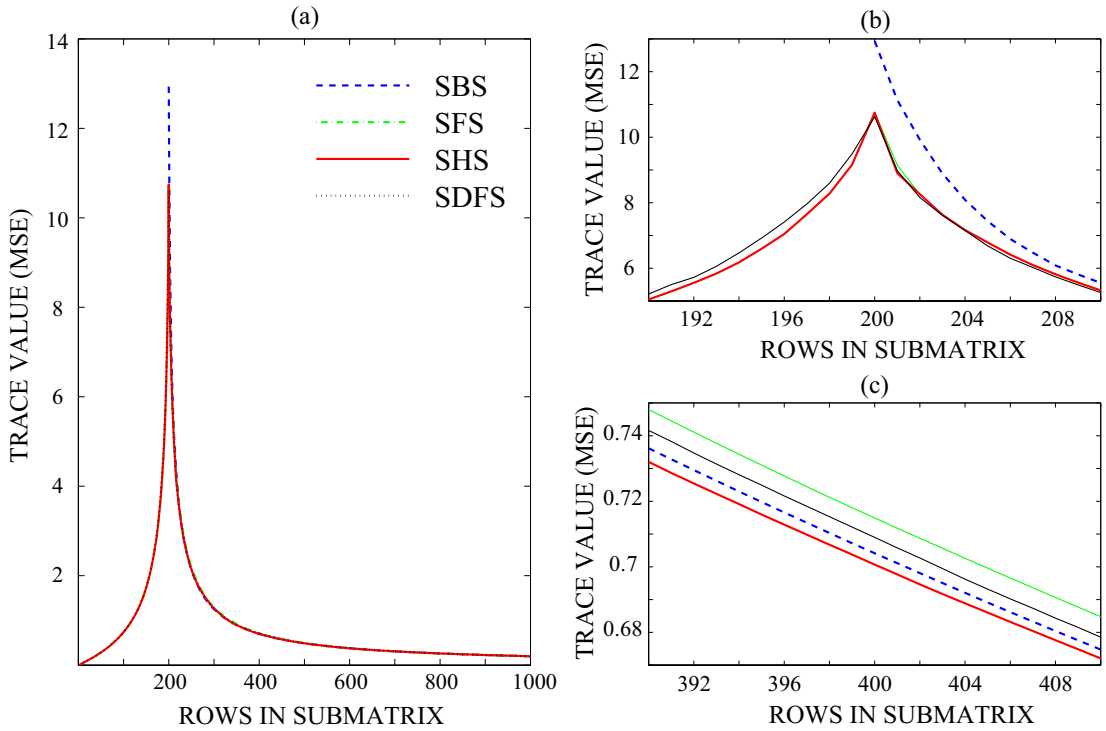


Figure 2.3: Evolution of the trace on a partial Fourier matrix. Plot (a) shows the trace as the submatrix is grown from 0 to 1000 rows. (b) shows rows 190 to 210, around the peak in part (a) when the submatrix is square. (c) is zoomed in when the submatrix is larger than square (overdetermined).

We see that the SHS algorithm gives the submatrix with the lowest trace value over almost all row subsets. The SDFS algorithm also performs very well, competing favourably with the SFS algorithms, and the SDFS is also the fastest of all four algorithms. The SBS algorithm only works for subsets of at least n rows otherwise criterion (2.20) is undefined. We expect the SBS algorithm to select a better subset than the other algorithms, although in this instance the SBS algorithm selects a subset of size n that has a much higher trace value, so is not performing as well as the competing algorithms.

2.6.2 ALGORITHM RUN-TIME COMPARISON

Another numerical experiment was performed to compare the running times of the SDFS and SFS algorithms. Ten Fourier submatrices of size 1024×128 were formed and the SDFS and SFS algorithms ran until they had chosen $k = 256$ rows of the submatrix. The same experiment was also performed on ten 1000×100 random matrices with the SDFS and SFS algorithms choosing $k = 250$ rows to form the submatrix $A_{\mathcal{B}}$. The average times over the ten runs are displayed in the Table 2.1.

Table 2.1: The average cputime in seconds (over 10 runs) of the SDFS and SFS algorithms on both Fourier and Random matrices.

	Fourier Matrix	Random matrix
SDFS	2.053	0.306
SFS	6.781	1.097

We see that the SDFS algorithm is significantly faster than the SFS algorithm — it requires less than 1/3 the computation time. Figures 2.2 and 2.3 and Table 2.1 show that the SDFS algorithm is performing as accurately as the SFS algorithm and is significantly faster. Clearly the SDFS algorithm is a significant improvement upon the standard SFS algorithm.

2.6.3 IMAGE RECONSTRUCTION EXPERIMENTS

We consider two image-reconstruction experiments. The first is performed on the Shepp-Logan phantom, while the second is performed on a small MRI data set.

For the Shepp-Logan Phantom, the pixel matrix measures 512×373. The 2D Fourier Transform of the pixel matrix was computed to imitate a k-space data set (recall Section 1.1.3). The SDFS and SFS algorithms were run until they had selected 350 rows of the 'k-space' data set. The results are shown in Figure 2.4.

The SDFS algorithm selects a sub-matrix from the original data set which leads to an excellent image reconstruction. The SFS image reconstruction is poor and suffers from severe artifacts. This is because the SFS algorithm has not selected enough data from the center of the k-space matrix. (Around the bright white spot in the center of the images on the left.) The center of k-space contains most of the high frequency information, which corresponds to image detail, so it is important that enough of this data is selected. The SDFS algorithm selects enough of the high frequency data to allow a high quality image reconstruction.

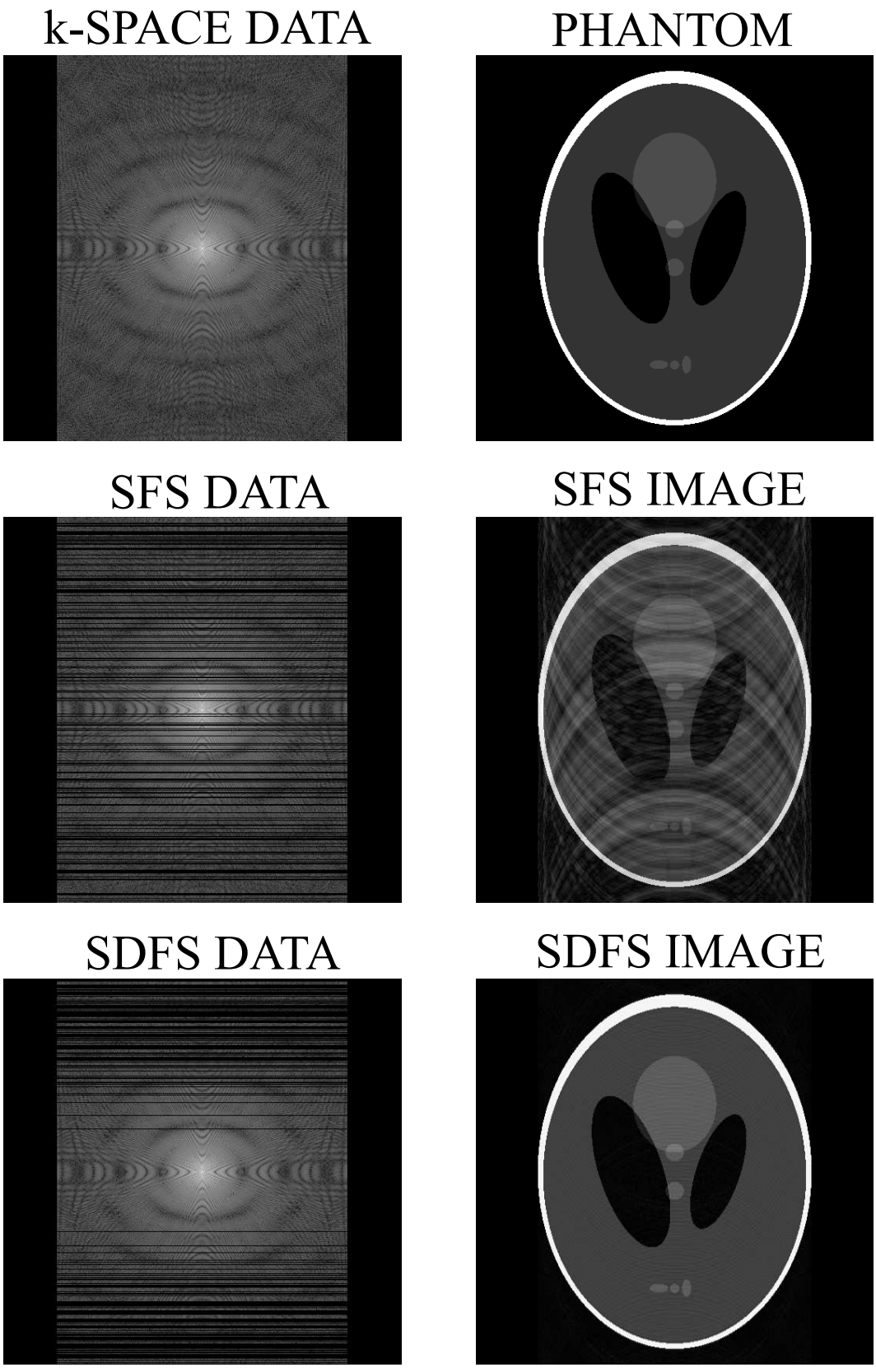


Figure 2.4: Top row: Full k-space matrix and Phantom. Middle Row: The data sub-matrix chosen by the SFS algorithm and corresponding phantom reconstruction. The reconstruction suffers from artifacts. Bottom row: The data sub-matrix chosen by the SDFS algorithm and corresponding phantom reconstruction. The reconstructed image is of high quality.

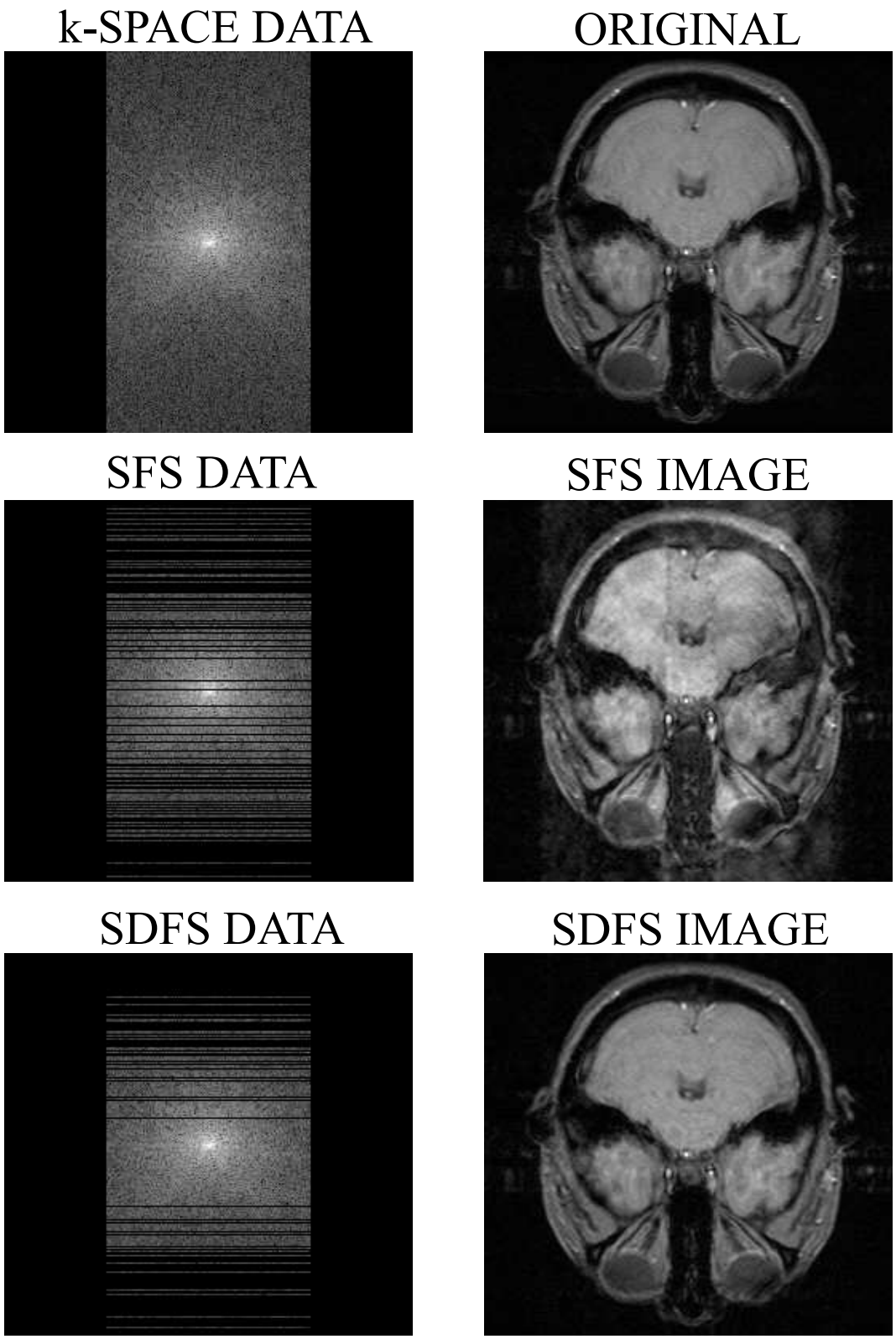


Figure 2.5: Top row: k-space data (Fourier matrix of size 256×128) and the original MRI image. Middle row: The 135 rows of k-space data chosen by the SFS algorithm and corresponding image reconstruction. The reconstructed image is blurry. Bottom row: The 135 rows of k-space data selected by the SDFS algorithm and the corresponding image reconstruction. Again, the reconstruction is very good.

The second experiment is an image-reconstruction experiment performed on an MRI data set. The k-space data matrix has size 256×128. The SDFS and SFS algorithms selected $k = 135$ rows from the matrix (just over half) and the results are shown in Figure 2.5.

Again we see that the SDFS algorithm does an excellent job with the image reconstruction process. There is very little difference between the original image and the SDFS reconstruction. The SFS fails to produce a usable image with so little data. Again the SFS data matrix is missing rows through the middle of k-space, which results in the poor quality image.

2.6.4 HYBRID SELECTION EXPERIMENT

The results in Section 2.6.3 show that the SFS algorithm is choosing data subsets that do not allow for a quality image to be reconstructed. This is where the SHS algorithm is useful. This experiment uses the data set chosen by the SFS algorithm on the MRI reconstruction described in Section 2.6.3. The SHS algorithm improved this data subset by performing row-exchanges according to criterion (2.35). The results are shown in Figure 2.6.

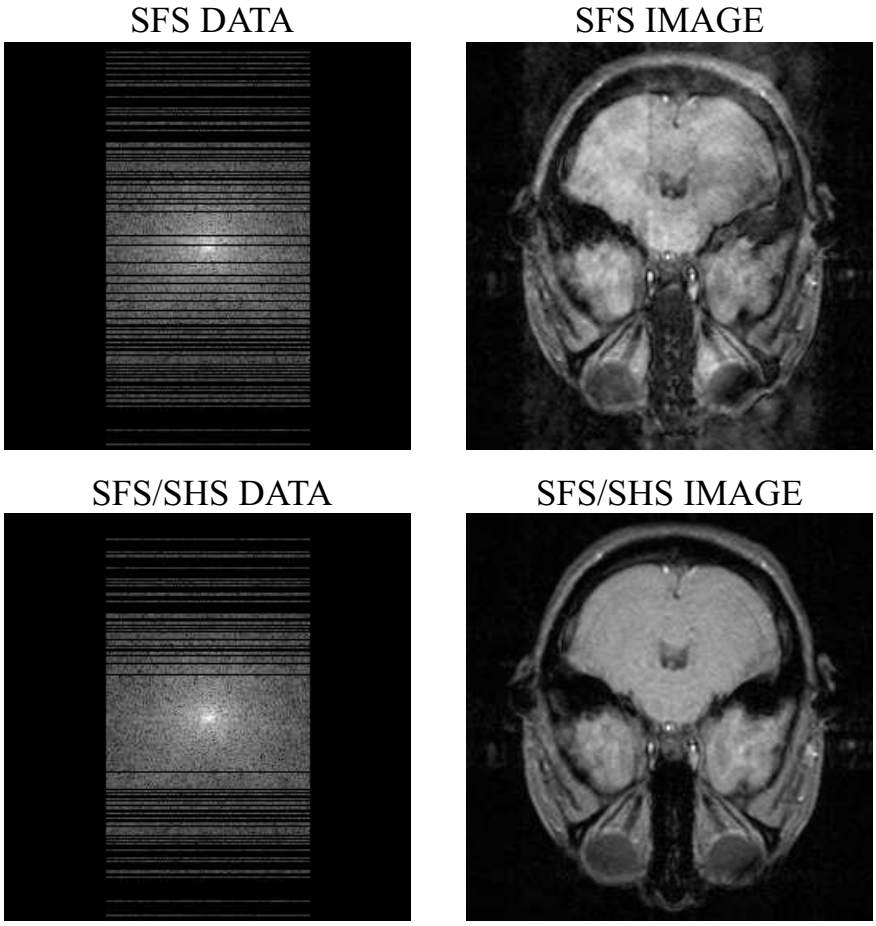


Figure 2.6: Top row: The data chosen by the SFS algorithm and corresponding image reconstruction. Bottom row: The data set chosen by the SFS algorithm with row exchanges by the SHS algorithm and the corresponding image reconstruction. The image reconstructed using the subset chosen by the SHS algorithm is much better than that using the SFS algorithm alone.

We see that the SFS algorithm coupled with the SHS algorithm leads to a high quality image that is now comparable with that produced by the SDFS algorithm. This is a vast improvement over the SFS algorithm alone. The SHS algorithm swaps rows from the outside of the k-space matrix with those closer to the center which means enough of the high frequency information has been selected to allow for this good reconstruction. The SHS algorithm needed only 10 row exchanges for this large improvement. (We stress here that the SFS/SHS data set still has only 135 out of the 256 possible rows.)

As previously mentioned, the SFS algorithm needed to select 180 rows of the original data matrix for a quality image to be produced. Here we compare the time needed to select these 180 rows, versus the time taken for the SFS/SHS algorithm to select the improved 135 rows. The results are shown in Table 2.2.

Table 2.2: An experiment comparing the CPU time taken by the SFS algorithm to select 135 rows, the CPU time taken by the SFS algorithm to select 135 rows and for the SHS algorithm to perform row exchanges to improve the quality of the selected subset, and the CPU time taken by the SFS algorithm to select 180 rows.

	SFS (135 Rows)	SFS/SHS	SFS (180 Rows)
Times (s)	0.54	0.63	0.87

Table 2.2 shows that it is much faster to use a row-exchange process than to continue selecting rows until the image is of high quality.

2.6.5 COMPARISON WITH RANDOM SELECTION

In this experiment we demonstrate the difference between the sequential algorithms and a randomly selected subset. The SFS, SDFS and SBS algorithms were run as normal on a partial Fourier matrix of size 100×20. A vector p containing a random permutation of the numbers 1 to 100 was then generated. The index set $\mathcal{B}$ was extended so that at the lth iteration, $\mathcal{B} = \{p(1), \ldots, p(l)\}$, and the trace value of the corresponding submatrix was found. The results are shown in Figure 2.7.

The superiority of the sequential algorithms is clear. The trace value of the randomly chosen subset can be very large, particularly when the chosen submatrix has n rows. Although it is difficult to see in the plot, the randomly chosen subset has a much higher trace value throughout the selection process, even as the subset grows.

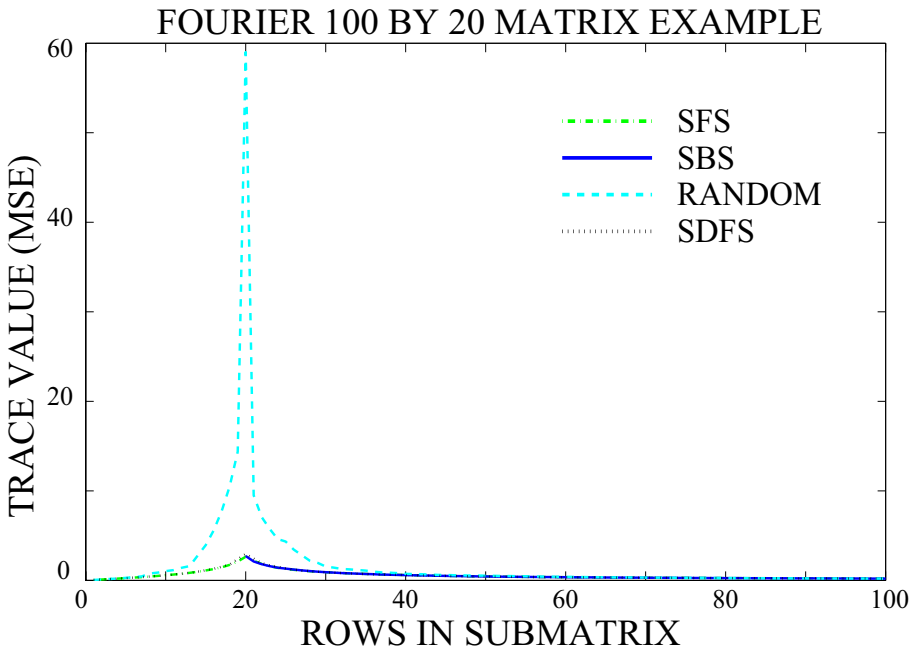


Figure 2.7: Plot showing the sequential algorithms versus a randomly chosen subset. The plot shows that all of the sequential algorithms result in submatrices with a significantly lower trace value than the submatrices formed via a random selection approach.

2.7 CONCLUSION

Observation subset selection is a very important problem that arises in a wide range of areas. In this chapter we have devised a new criterion for choosing a subset of observations based upon the determinant of a matrix. This determinant criterion is simpler than the existing trace criterion and is therefore less expensive to compute. We have also described an algorithm based upon the determinant criterion, called the Sequential Determinant Forward Selection (SDFS) algorithm. Numerical results show that this algorithm is faster than the SFS algorithm, and as accurate. This new algorithm shows great potential, specifically in the field of Magnetic Resonance Imaging where the SFS and SBS algorithms are currently being used.

We have also devised a hybrid scheme for observation subset selection and have derived a row-exchange criterion based upon the trace of matrix. A simpler determinant row-exchange criterion was also described. Row exchanges are a means of improving the observation subset, and the numerical results shown in this chapter (and in [14]) are very promising.

CHAPTER 3

COMPRESSED SENSING

In the previous chapter we considered the problem of reconstructing signals (or images) when there is an overdetermined, system of linear equations. This chapter also models the image reconstruction problem as

$$b = Ax + w, \tag{3.1}$$

but now we consider (3.1) for an *underdetermined* system of equations. In (3.1), $A \in \mathbb{R}^{m \times n}$ is a measurement matrix (where $m < n$), $b \in \mathbb{R}^m$ is a vector of observations, $x \in \mathbb{R}^n$ is the vector of unknowns and w is noise. The aim is to determine x, so (3.1) is an inverse problem. This problem is ill-posed because there are infinitely many solutions.

Chapter 3 discusses compressed sensing—a relatively new mathematical framework that allows signals to be reconstructed accurately from few observations or measurements.

Section 3.1 provides an introduction to compressed sensing. The basic mathematical principles are discussed and the important concepts, including sparsity and incoherence, are described. Section 3.2 describes how the mathematical theory underlying compressed sensing is put into practice using optimization algorithms. It also describes some of the problem formulations and currently used algorithms. In Section 3.3 a new algorithm for compressed sensing called the Box-Constrained Gradient Projection (BCGP) algorithm is presented. Section 3.4 details numerical experiments using the new algorithm, and in Section 3.5 we provide some concluding remarks.

3.1 AN INTRODUCTION TO COMPRESSED SENSING

Until recently, almost all signal acquisition protocols follow the celebrated Shannon sampling theorem [86], which explains that to ensure accurate signal reconstruction, the sampling rate must be twice the maximum frequency present in the signal (the so-called Nyquist rate). Compressed Sensing is a new and novel sampling paradigm that goes against this common wisdom. The following section gives an introduction to compressed sensing and explains why it may be possible to reconstruct signals and images from very little data.

3.1.1 SPARSITY

Consider a discrete signal $f \in \mathbb{R}^n$. (Usually such a signal is time-dependent ($f(t)$) but for simplicity we will only consider an *instance* of the time-dependent signal in this work.) There is no loss of generality in assuming that f is a signal (vector) because if f is a higher-dimensional object, for example an image, then the object can be reshaped into a single long vector using some specified ordering. Any signal can be represented in an orthonormal basis. Furthermore, many signals can be represented 'concisely' in a particular orthonormal basis. By concisely we mean that many of the transform coefficients are zero and we say that f has a sparse representation in that basis. Suppose we have such an orthonormal basis $\Psi = [\psi_1, \psi_2, \ldots, \psi_n]$ (often referred to as the representation basis) and a sparse coefficient sequence for f: $x_j = \langle f, \psi_j \rangle$. It is convenient to express this in matrix form as

$$f = \Psi x. \tag{3.2}$$

Usually we would have to sample all n of the transform coefficients x_j to ensure accurate reconstruction of f, but because we know that the coefficient vector is sparse, one may hope that we could sample fewer of the coefficients and still get an accurate reconstruction. Consider x_s, the vector x with all but the largest s coefficients set to zero. By definition, $f_s = \Psi x_s$ and we say that x_s is s-sparse. Because Ψ is an orthonormal basis we have the relation

$$\|f - f_s\|_2 = \|x - x_s\|_2 \ , \tag{3.3}$$

which explains that if x is s-sparse or can be well represented by a sparse vector x_s then the error in the reconstructed signal f_s is small.

In general, to recover f accurately, all n of the coefficients of x must be sampled. However, in many real-world situations, acquiring a complete set of data is so costly as to make it impractical. Suppose we take $m < n$ measurements of the signal f using the basis Φ (often referred to as the *sampling* or *sensing* basis), giving

$$b = \Phi f, \tag{3.4}$$

where Φ is $m \times n$ with $m < n$. Combining equations (3.2) and (3.4) we have

$$b = \Phi f = \Phi \Psi x.$$

Letting $A = \Phi\Psi$ gives $b = Ax$, which is equation (3.1) in the noiseless case ($w = 0$). There are infinitely many candidate signals $\hat{x}$ that satisfy $A\hat{x} = b$, so the problem is ill-posed. The chance of choosing the $\hat{x}$ that correctly recovers f seems very small, which is why sparsity is so important. If we know that the signal f is sparse in some basis then the problem becomes: of all the candidate signals $\hat{x}$ satisfying $A\hat{x} = b$, choose the one that is the most sparse.

This seems a very nice idea in theory. However, to make compressed sensing viable in practice there must also be an efficient way to recover the signal. One may wish to use the "l_0-norm",

which simply counts the number of non-zeros in a given vector, and pose the signal recovery problem as

$$\min_x \ \|x\|_0 \quad \text{subject to} \quad Ax = b.$$

However, the l_0-norm is not a true norm and to determine the signal that has the fewest non-zero elements is a combinatorial problem with ${}^nC_s$ possibilities. As we saw in Chapter 2, combinatorial problems can be computationally intractable and we prefer to avoid them when possible.

One may naturally consider whether another norm might be more appropriate. Mathematically, a norm is defined for $p \geq 1$ by the relation

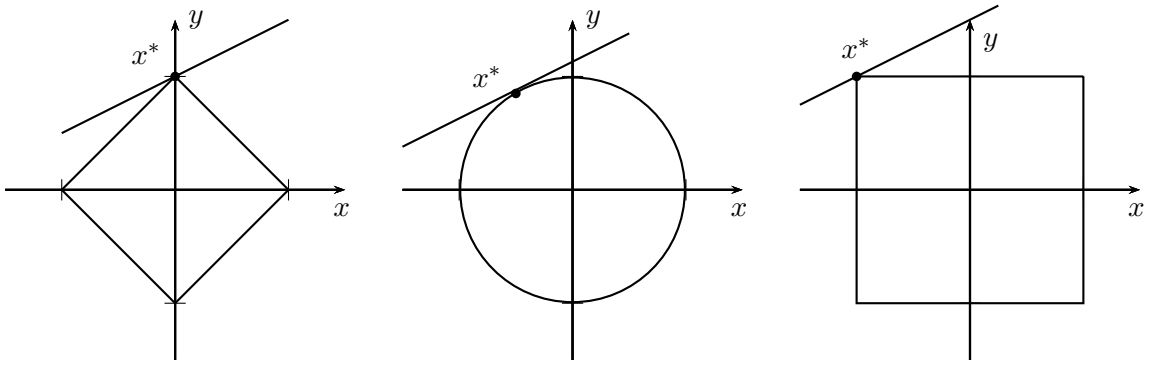
$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}.$$

The signal recovery problem may be posed as

$$\min_x \ \|x\|_p \quad \text{subject to} \quad Ax = b. \tag{3.5}$$

For $p \geq 1$, (3.5) is a more computationally tractable problem, because norm minimization subject to linear equality constraints is a convex optimization problem. This has nice properties; for example, a local minimizer is a global minimizer.

The three most commonly used norms are for $p = 1$, $p = 2$, and $p = \infty$. We consider the associated norm balls to decide the norm that would be most appropriate for use in compressed sensing. The norm balls and the line $a^T x - b$ (corresponding to a hyperplane in the n-dimensional case) are shown in Figure 3.1, where the p-norm ball has been blown up until it first meets the line.



(a) The l_1-norm ball **(b)** The l_2-norm ball **(c)** The l_∞-norm ball

Figure 3.1: The 1, 2, and ∞ norm balls have been 'blown up' until they meet with the line $a^T x - b$. The point x^* indicates the solution to problem (3.5) for the corresponding p-norm.

The points x^* in Figure 3.1 (a), (b), and (c) correspond to the solution of problem (3.5) for the corresponding p-norm. For the two and infinity norms, the x and y coordinates corresponding to the contact point x^* are both non-zero. However for the 1-norm, the contact point occurs across the y-axis and therefore has one zero and one non-zero coordinate. In general (n-dimensions), the 1-norm will give a sparse solution to (3.5) while the 2- and ∞-norm will not. This intuitively leads us to choose the 1-norm for compressed sensing problems as we are seeking a sparse solution. We pose the signal recovery problem as

$$\min_x \quad \|x\|_1 \quad \text{subject to} \quad Ax = b. \tag{3.6}$$

Indeed the use of the 1-norm as a sparsity-inducing tool goes back some time, for example [28, 62]. More recently came work on the Least Absolute Shrinkage and Selection Operator (LASSO) [94], and on the Basis Pursuit (BP) and Basis Pursuit Denoising (BPDN) problems [26], which both aim to solve systems of equations for which a sparse solution is sought.

While we prefer to solve problem (3.5) for $p = 1$ (giving formulation (3.6)), others have preferred to use $p < 1$. The problem is no longer convex and as mentioned, $\|\cdot\|_p$ for $p < 1$ is not a true norm. However, positive results have been reported. We will not comment further on these non-convex problems, but further information can be found, for example, in [23, 24, 88].

Remark: We have explained that compressed sensing is based upon the idea that a signal may be sparse in some basis. A signal is said to be s-sparse if all but s of its entries are zero. However, this idea also works when the signal is not sparse but *compressible*. Compressibility is the notion that, subject to a reordering, a signal has a finite region of support, outside of which the coefficients decay rapidly. If a signal is compressible, it can be well represented by an s-sparse signal, in which case (3.3) holds and the error in the reconstruction of f is small.

3.1.2 INCOHERENT SAMPLING

So far we have seen that if a signal is sparse in some basis, then accurate reconstruction from few measurements should be possible. However, for a given representation basis Ψ, we cannot choose just any sensing basis Φ and hope to reconstruct the signal. This is demonstrated by the following example. Suppose that the sensing basis is the same as the representation basis. In this case we have $\Psi^H\Psi = I$. All n coefficients must be sampled or guaranteed recovery of f is impossible. This example leads to the notion of *incoherence*, which is an important concept for compressed sensing.

Definition 3.1.1 ([21])**.** *The coherence between the sensing system Φ and the representation system Ψ is defined as*

$$\mu(\Phi, \Psi) = \sqrt{n} \cdot \max_{1 \le j,k \le n} |\langle \varphi_j, \psi_k \rangle|.$$

The value of $\mu(\Phi, \Psi)$ lies in the interval $[1, \sqrt{n}]$. The upper bound occurs because the inner product between two unit vectors is at most one, and the lower bound is a consequence of

Parseval's identity [58, p. 8–16].

Up to this point we have not quantified how sparse a signal must be, or how few observations can be made while ensuring accurate signal reconstruction. The following theorem offers bounds on how few measurements are required for accurate reconstruction, and the bound is dependent on the sparsity level of the signal.

Theorem 3.1.1 ([18]). *Fix $f \in \mathbb{R}^n$ and suppose that the coefficient sequence x of f in the basis Ψ is s-sparse. Select m measurements in the Φ domain uniformly at random. Then if*

$$m \geq c \cdot \mu^2(\Phi, \Psi) \cdot s \cdot \log n \ ,$$

for some positive constant c, the solution to the convex optimization problem (3.6) is exact with overwhelming probability.

(In fact, the probability of success exceeds $1 - \delta$ if $m \geq c \cdot \mu^2(\Phi, \Psi) \cdot s \cdot \log(n/\delta)$.)

This theorem makes it clear why incoherence is so important: the smaller the value of $\mu(\Phi, \Psi)$, the smaller the number of measurements m that are required for exact signal recovery. For the earlier example when Ψ was chosen as both the representation and sensing basis, we get maximal coherence and Theorem 3.1.1 supports the earlier conclusion that we cannot undersample the data.

Fortunately there are sensing/representation matrix pairs for which we get low or even optimal $\mu(\Phi, \Psi) = 1$ coherence. We discuss a few examples here that are presented in more detail in [21].

1. Suppose the sensing basis is a spike basis $\varphi_k(t) = \delta(t - k)$ and the representation basis is a Fourier basis $\psi_j(t) = n^{-1/2} e^{i2\pi jt/n}$. Then we have $\mu(\Phi, \Psi) = 1$, which is maximal incoherence.

2. A random matrix usually displays low coherence with any fixed basis Ψ. If n vectors of unit length are sampled independently and then orthonormalised, with high probability the coherence between this Φ and a fixed Ψ is approximately $\sqrt{2 \log n}$.

3. Many signals are sparse in a wavelet basis, and a sensing basis that is largely incoherent with a wavelet basis is a noiselet basis [29]. The coherence between a noiselet sensing basis and a Haar wavelet representation basis is $\sqrt{2}$. Noiselets have the advantage that they have very fast transforms and the basis does not need to be stored to be applied to a vector. We say that noiselets form a *fast operator*.

3.1.3 THE RESTRICTED ISOMETRY PROPERTY

So far we have discussed the important roles sparsity and incoherence play in compressed sensing. However, we have not discussed which kinds of matrices will enable exact reconstruction, and this brings into play the Restricted Isometry Property (RIP). The RIP is a

measure of how well a given matrix preserves Euclidean length, and we have the following relation:

$$(1-\delta_s)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1+\delta_s)\|x\|_2^2.$$

The constant δ_s is called the restricted isometry constant. If δ_s is close to zero then the matrix approximately preserves Euclidean length. Recall that $A = \Phi\Psi$. The RIP is needed for the following result, which shows that if a matrix satisfies the restricted isometry property of 'order s' (where s is the number of non-zeros in the signal) then the reconstruction via l_1 minimization is accurate.

Theorem 3.1.2 ([20]). *Assume that A is defined above and has the RIP property such that $\delta_{2s} < \sqrt{2}-1$. Then the solution $\hat{x}$ to (3.6) satisfies*

$$\|\hat{x}-x\|_2 \leq \frac{c}{\sqrt{s}} \cdot \|x_s - x\|_1 \tag{3.7}$$

for some scalar c depending upon the instance, where x_s is as defined previously.

3.1.4 NOISY MEASUREMENTS

The results described so far have been based on a noise-free model with $w = 0$ in equation (3.1). However, many of the results in compressed sensing have been extended to the case when noise is present in the linear model. The optimization problem

$$\min_x \|x\|_1 \quad \text{subject to} \quad \|Ax-b\|_2 \leq \sigma\ , \tag{3.8}$$

may then be preferred over (3.6), where σ is proportional to the noise in the measured data. Theorem 3.1.2 is now extended as follows.

Theorem 3.1.3 ([19]). *Assume that $\delta_{2s} < \sqrt{2}-1$. Then the solution $\hat{x}$ to (3.8) obeys*

$$\|\hat{x}-x\|_2 \leq \frac{c_0}{\sqrt{s}} \cdot \|x_s - x\|_1 + c_1\sigma\ ,$$

for some scalars c_0 and c_1.

Theorem 3.1.3 places a bound on the error in the reconstructed signal when there is noise present in the measured data.

Many other results in compressed sensing can also be extended to the case when noise is present in the measured data, although we will not go into further detail here. Instead we suggest [19, 36].

3.2 CURRENT ALGORITHMS

The previous sections show that there has been much research on the theory underlying compressed sensing and there is now a solid mathematical framework within which to work. This section demonstrates that there has also been a great deal of research on algorithms to solve the associated optimization problems. Several current algorithms for compressed sensing are described in this section.

3.2.1 BASIS PURSUIT

We have already seen the convex optimization problem formulation (3.6):

$$\min_{x} \ \|x\|_1 \quad \text{subject to} \quad Ax = b,$$

which falls into the Basis Pursuit (BP) category [25, 26]. Theorem 3.1.1 explains that if there are enough observations, signal recovery using this optimization formulation is exact with very high probability. Problem (3.6) can be recast as the linear programming problem

$$\begin{aligned} \min_{x,u} \ \sum_{j=1}^{n} u_j \qquad \text{subject to} \qquad -u_j \leq \ & x_j \ \leq u_j, \\ Ax \ &= \ b, \end{aligned}$$

or as

$$\begin{aligned} \min_{u,v} \ \sum_{j=1}^{n} u_j + v_j \qquad \text{subject to} \qquad A(u-v) \ &= \ b, \\ u_j, v_j \ &\geq \ 0 \end{aligned}$$

Linear programming is a well studied area of optimization and many robust algorithms exist. Compressed sensing has brought linear programming back into the spotlight. One of the first algorithms specifically designed to solve problems from compressed sensing was the l_1-magic algorithm by Candès and Romberg [17]. They wrote a suite of MATLAB programmes, all generally referred to as l_1-magic, to solve problem (3.6) and its various relaxations, for example, (3.8). The MATLAB code for l_1-magic is freely available at `http://www.l1magic.org`. These were the first programmes to show that the compressed sensing theory could be realised in practice via optimization algorithms. Since l_1-magic, there has been a flurry of activity to find ever-more efficient algorithms to solve the optimization problems from compressed sensing. The Rice University webpage is a frequently updated site that contains a wealth of information on this topic: `http://dsp.rice.edu/cs` [53]. There are also several excellent review articles, including [21, 82, 110].

3.2.2 QUADRATIC PROGRAMMING

Another approach to solving the signal recovery problems from compressed sensing is to use the unconstrained convex optimization formulation

$$\min_{x} \ \|Ax - b\|_2^2 + \lambda \|x\|_1 \ , \tag{3.9}$$

which is the l_1-regularized least-squares approach (BPDN). The scalar λ is a regularization parameter that determines the balance between the quadratic loss term and the sparsifying l_1 term. The parameter λ is usually small and in [60] it is shown that λ must satisfy

$$\lambda < 2 \ \|A^T b\|_\infty \ , \tag{3.10}$$

otherwise the solution to (3.9) is the zero vector. Problem (3.9) can be reformulated as a quadratic programme with linear inequality constraints as follows:

$$\min_{u,x} \ \|Ax - b\|_2^2 \ + \ \lambda \sum_{j=1}^{n} u_j \qquad \text{subject to} \qquad -u_j \leq x_j \leq u_j. \tag{3.11}$$

In [59, 60] Kim et al. describe a truncated Newton interior point method to solve problem (3.11). The algorithm is called l_1-ls and the associated MATLAB code is freely available at `http://www.stanford.edu/~boyd/l1_ls/`.

In [25, 26, 40], problem (3.9) is reformulated via the substitution $x = u - v$, to obtain the differentiable problem

$$\begin{aligned} \min_{u,v} \quad & \frac{1}{2} \|A(u-v) - b\|_2^2 + \tau \sum_{j=1}^{n} (u_j + v_j) , \\ \text{subject to} \quad & u_j, v_j \geq 0 , \end{aligned} \tag{3.12}$$

where the regularization parameter τ is related to the previous regularization parameter (3.10) through $\lambda = 2\tau$. At the solution of problem (3.12) we have $u_j = 0$ or $v_j = 0$. Problems (3.9) and (3.12), although different, share a common minimizer.

3.2.3 OTHER ALGORITHMS

Many other algorithms exist with applications to compressed sensing and the associated signal and image processing problems. For example, the SpaRSA (Sparse Reconstruction by Separable Approximation) algorithm [103, 104] and the FISTA (Fast Iterative Shrinkage-Thresholding) algorithm [4]. Other current algorithms include a projected Barzilai-Borwein-type algorithm with applications in computed tomography [98], the Gradient Projection (GP) algorithm (along with the Steplength Selection for Gradient Projection (GPSS) variant [64]). There is also a gradient descent algorithm that uses a thresholding step to encourage sparsity [49] and in [96] a range of problems that arise in compressed sensing are discussed.

Greedy algorithms have also proved popular for compressed sensing problems. These algorithms include Orthogonal Matching Pursuit (OMP) [95], Stagewise Orthogonal Matching Pursuit (StOMP) [35] and Compressed Sampling via Matching Pursuit (CoSaMP) [70].

In the remaining part of this chapter we focus on creating a new algorithm for compressed sensing. This algorithm should be able to solve the convex optimization problems arising in compressed sensing, accurately and efficiently.

3.3 A NEW ALGORITHM FOR COMPRESSED SENSING

In this section we propose a new algorithm that can be applied to problems in compressed sensing. The algorithm uses a new formulation of the optimization problem (3.9), employs the Barzilai-Borwein step lengths (which are described in the following subsection), a projection operator, and a non-traditional backtracking line search loop based on recent work by Dai and Fletcher [30].

Our approach uses two levels of iteration (or nested loops), an outer loop defining a search direction and new candidate point x, and an inner backtracking line search loop. The back-

tracking line search is included as a safeguard and was suggested in [30] to prevent iterates cycling and to ensure convergence of the algorithm. The algorithm only enters the backtracking line search loop under certain conditions that in practice rarely arise.

The approach only requires matrix-vector products involving A and A^T. At each iteration there is one vector multiplication with A and one with A^T, unless the inner loop is required, in which case there is an additional multiplication with A in the backtracking line search. The computational effort is therefore low in each iteration.

In the remainder of this section, we introduce the different features of our algorithm, which is described in detail in Section 3.3.9.

3.3.1 THE BARZILAI–BORWEIN ALGORITHM

In 1988 Barzilai and Borwein [3] devised a novel gradient method for unconstrained optimization problems, based on a two-point approximation to the Secant method [3]. Their algorithm has the unusual property that at some iterates the function value increases. Despite this property, the algorithm performs very well in practice. In fact, forcing a monotonic decrease in the function value at each iteration can seriously impair the practical performance of the algorithm [30].

In [3] the algorithm is proven to converge in the unconstrained, 2-dimensional, positive definite, quadratic case. This was extended to the n-dimensional, strictly convex quadratic case [75], and in [44] the convergence of the algorithm in the unconstrained, quadratic case when the Hessian is positive semi-definite is proved.

The Barzilai-Borwein algorithm is a steepest-descent-type algorithm. For an objective function $f(x)$, the steepest-descent method updates the candidate point at each iteration via

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k).$$

That is, from the point x_k, a step of size α_k is taken in the direction of the negative gradient. The step length α_k is chosen such that the objective function at the new point x_{k+1} is lower than the function value at the current point x_k. The Barzilai-Borwein algorithm has two specific formulae for calculating α_k and these do not necessarily result in a reduced objective function value at the next iterate. The two formulae for the step lengths are derived in the following subsection.

There has been much interest in this algorithm more recently: the implementation of dynamical retards [65], a Cauchy Barzilai-Borwein method [77], analysis of convergence properties [32] and the introduction of a cyclic Barzilai-Borwein variant [31]. (See also the review by Fletcher [43].)

3.3.2 THE STEP LENGTH FORMULAE

The Barzilai-Borwein algorithm is designed to solve a general unconstrained optimization problem. Let $f(x)$ be the objective function and denote the gradient vector by $g_k = \nabla f(x_k)$, and the Hessian matrix by $H_k = H(x_k) = \nabla^2 f(x_k)$. Define y_k and s_k as follows:

$$y_k = g_k - g_{k-1} \qquad \text{(the change in gradient)} \tag{3.13}$$

$$s_k = x_k - x_{k-1} \qquad \text{(the change in position).} \tag{3.14}$$

At the kth iteration we have the approximation

$$y_k \approx H_k s_k, \tag{3.15}$$

and if $f(x)$ is a quadratic function, then the Hessian is constant and (3.15) holds with equality.

To find the first Barzilai-Borwein step length we approximate the Hessian by a multiple of the identity matrix ($H_k \approx \alpha_k^{-1} I$, where $\alpha_k > 0$), and solve

$$\alpha_k^{BB_1} = \arg\min_{\alpha} \|y_k - \frac{1}{\alpha} s_k\|_2^2 = \frac{s_k^T s_k}{s_k^T y_k}. \tag{3.16}$$

Similarly the second Barzilai-Borwein step length is found using $\alpha_k I$ to approximate H_k^{-1} and solving

$$\alpha_k^{BB_2} = \arg\min_{\alpha} \|\alpha y_k - s_k\|^2 = \frac{s_k^T y_k}{y_k^T y_k}. \tag{3.17}$$

Formulae (3.16) and (3.17) are the two step lengths used in the Barzilai-Borwein algorithm and the step lengths used in the algorithm we propose. Note that $\alpha_k > 0$ if and only if $s_k^T y_k > 0$. The case $s_k^T y_k < 0$ cannot occur in the quadratic case when the constant Hessian is positive semi-definite, but $s_k^T y_k = 0$ is possible in this case so we need extra safeguards that are described later.

Because we are using the given step lengths it is possible that $f(x_{k+1}) > f(x_k)$! That is, the function value may increase with either choice of step length. To force a decrease in the objective function at each iteration essentially reduces the Barzilai-Borwein method to the steepest-descent method.

A SPECIAL CASE

We now consider the following constrained optimization problem, where the objective is a quadratic function subject to non-negativity bounds on the variables and the constant Hessian is positive semi-definite. The problem is a re-scaled version of (3.12):

$$\min_{u,v} \qquad \|A(u-v) - b\|_2^2 + \lambda \sum_{j=1}^{n} (u_j + v_j) \,, \tag{3.18}$$

$$\text{subject to} \qquad u_j, v_j \geq 0.$$

Formulation (3.18) is differentiable and the associated gradient is

$$g = \begin{bmatrix} 2A^T(A(u-v)-b) + \lambda \mathbf{1} \\ -2A^T(A(u-v)-b) + \lambda \mathbf{1} \end{bmatrix}, \tag{3.19}$$

(where $\mathbf{1}$ is a vector of ones) and the Hessian is

$$H = 2\begin{bmatrix} A^TA & -A^TA \\ -A^TA & A^TA \end{bmatrix}. \tag{3.20}$$

For the function (3.18) we have derived the following result for the step length formula (3.17).

Theorem 3.3.1. *For the function (3.18), if $A \in \mathbb{R}^{m\times n}$ $(m < n)$ has orthonormal rows, and for $y_k \neq 0$, then the Barzilai-Borwein step length (3.17) satisfies*

$$\alpha_k^{BB_2} = \frac{1}{4}.$$

Proof. Let $x_k = u_k - v_k$, $\tilde{u} = u_k - u_{k-1}$ and let $\tilde{v} = v_k - v_{k-1}$. Then from (3.19),

$$y_k = g(x_k) - g(x_{k-1}) = \begin{bmatrix} 2A^TA(\tilde{u}-\tilde{v}) \\ -2A^TA(\tilde{u}-\tilde{v}) \end{bmatrix},$$

so

$$\begin{aligned} y_k^T y_k &= 8(\tilde{u}-\tilde{v})^T A^T A A^T A(\tilde{u}-\tilde{v}) \,, \\ &= 8(\tilde{u}-\tilde{v})^T A^T A(\tilde{u}-\tilde{v}) \,, \end{aligned}$$

as $AA^T = I$. The objective function in (3.18) is quadratic so $y_k = Hs_k$ where H is the Hessian matrix (3.20). Therefore $s_k^T y_k = s_k^T H s_k$, so

$$s_k^T H s_k = 2\begin{bmatrix} \tilde{u}^T & \tilde{v}^T \end{bmatrix} \begin{bmatrix} A^TA & -A^TA \\ -A^TA & A^TA \end{bmatrix} \begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix} = 2(\tilde{u}-\tilde{v})^T A^T A(\tilde{u}-\tilde{v}).$$

Thus

$$\alpha_k^{BB_2} \;=\; \frac{s_k^T y_k}{y_k^T y_k} \;=\; \frac{2(\tilde{u}-\tilde{v})^T A^T A(\tilde{u}-\tilde{v})}{8(\tilde{u}-\tilde{v})^T A^T A(\tilde{u}-\tilde{v})} \;=\; \frac{1}{4}\,.$$

□

Remarks:

1. If the function (3.12) is used instead of the function (3.18), then $\alpha_k^{BB_2} = \frac{1}{2}$.
2. The Barzilai-Borwein algorithm is a variant of steepest descent. The literature on the BB algorithm recommends that a strategy that alternates between the two step length formulae works best in practice; however, steepest descent is infamous for performing poorly when a constant step length is used. Theorem 3.3.1 suggests that the BB step length formula (3.17) should not be used when the sensing matrix is known to have orthonormal rows. This explains why in [40], no improvement is reported when the step lengths alternate between (3.16) and (3.17) and they instead use only (3.16) even though the literature would discourage this.

3.3.3 UPPER BOUNDS ON THE SOLUTION VECTOR

At the unique minimizer x^* to the BPDN problem (3.9) we have

$$f(x^*) \leq f(x) = \|Ax - b\|_2^2 + \lambda\|x\|_1$$

for any x. Setting $x = 0$ gives,

$$\lambda\|x^*\|_1 \;\leq\; \|Ax^* - b\|_2^2 + \lambda\|x^*\|_1 \;\leq\; b^T b\,.$$

This implies that

$$\|x^*\|_1 \leq \frac{b^T b}{\lambda}\,,$$

and thus

$$|x_j^*| \leq \frac{b^T b}{\lambda} \qquad \text{for } j = 1, \ldots, n. \tag{3.21}$$

Expression (3.21) is an a priori upper bound on the components of x^* and hence an a priori upper bound on the components of both u and v. It is possible to improve this bound dynamically but numerical trials suggest this is not worthwhile and we do not expect this bound to be active at the solution.

There are other ways of deriving upper bounds on the elements of x^*. For example, let $\hat{x}$ minimize $\|Ax - b\|_2^2$. If A has full rank, $\min_x \|Ax - b\|_2^2 = 0$ and $\hat{x} = A^T(AA^T)^{-1}b$, so

$$\lambda\|x^*\|_1 \;\;\leq\;\; \|A\hat{x} - b\|_2^2 + \lambda\|\hat{x}\|_1 \;\;=\;\; \lambda\|A^T(AA^T)^{-1}b\|_1.$$

This means that

$$\|x^*\|_1 \;\;\leq\;\; \|A^T(AA^T)^{-1}b\|_1$$

and

$$|x_j^*| \;\;\leq\;\; \|A^T(AA^T)^{-1}b\|_1 \qquad \text{for } j = 1, \ldots, n.$$

In practice we would not compute $(AA^T)^{-1}$, but if A has orthonormal rows then $AA^T = I$ and in this case,

$$\|x^*\|_1 \;\;\leq\;\; \|A^T b\|_1.$$

Recall that the value $\|A^T b\|_\infty$ gives an appropriate value for the regularization parameter λ (3.10). From the property that [5, p. 366],

$$\|x\|_\infty \;\;\leq\;\; \|x\|_1 \;\;\leq\;\; n\,\|x\|_\infty\,,$$

we have

$$\|x^*\|_1 \;\;\leq\;\; n\|A^T b\|_\infty\,,$$

so

$$|x_j^*| \;\;\leq\;\; n\|A^T b\|_\infty\,. \tag{3.22}$$

As $\|A^T b\|_\infty$ has already been calculated, this second upper bound on the elements x_j^* is also very cheap to compute if A has orthonormal rows. This upper bound can also be changed dynamically if desired.

3.3.4 A BOX CONSTRAINED QUADRATIC PROGRAMME

Now that we have established upper bounds on the solution vector, problem (3.18) can be reformulated as a box-constrained quadratic programme

$$\min_{u,v} \qquad \|A(u-v)-b\|_2^2 + \lambda \sum_{j=1}^{n} u_j + \lambda \sum_{j=1}^{n} v_j \tag{3.23}$$

$$\text{subject to} \qquad 0 \leq u_j, v_j \leq \text{ub},$$

where ub is an upper bound defined by either (3.21) or (3.22). We prefer this new problem formulation because it is a convex, differentiable, quadratic programming problem for which an efficient algorithm can be created. The gradient and Hessian are as for problem (3.18), which we re-state here for convenience. The gradient is

$$g = \begin{bmatrix} 2A^T(A(u-v)-b) + \lambda \mathbf{1} \\ -2A^T(A(u-v)-b) + \lambda \mathbf{1} \end{bmatrix}$$

(where $\mathbf{1}$ is a vector of ones) and the associated Hessian is

$$H = 2 \begin{bmatrix} A^TA & -A^TA \\ -A^TA & A^TA \end{bmatrix}.$$

3.3.5 UPPER BOUND ON THE STEP LENGTH

The step length formula (3.16) is related to the eigenvalues of the Hessian H through the following expression:

$$\alpha_k^{BB_1} \;=\; \frac{s_k^T s_k}{s_k^T y_k} \;=\; \frac{s_k^T s_k}{s_k^T H s_k}\,, \tag{3.24}$$

where y_k and s_k are defined in (3.13) and (3.14) respectively. The right-hand term in (3.24) is the reciprocal of the Rayleigh quotient so the step length (3.16) is proportional to the reciprocal of eigenvalues of the Hessian. Moreover, the eigenvalues and eigenvectors of the Hessian correspond to the magnitude and direction of curvature of the objective function respectively.

In the strictly convex, quadratic, unconstrained case, the step lengths are automatically bounded by the reciprocal of the smallest eigenvalue of the Hessian [75] because the smallest eigenvalue is strictly greater than zero. In our case the objective function (3.23) has a positive *semi*-definite Hessian (3.20) so the smallest eigenvalue is zero. This means that the reciprocal is infinite and the corresponding step length is unbounded. In [76] the use of an upper bound on the step length, $\alpha_{\max} = 10^{30}$, is discussed as a safeguard, which is implemented if the algorithm finds a direction of zero (or near zero) curvature at any iterate. The step length $\alpha_{\max} = 10^{30}$ is still extremely large and could destroy numerical precision if encountered. For this reason it seems prudent to include a more reasonable upper bound on the step length.

We can determine the nature of the eigenvalues of the objective function (3.23) by writing the Hessian as the Kronecker product

$$H = A^T A \otimes \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}.$$

From the properties of the eigenvalues of a Kronecker product (see for example [5]), H has $2n-m$ zero eigenvalues corresponding to $2n-m$ directions of zero curvature. The remaining m positive eigenvalues of H are given by the positive eigenvalues of $4(A^T A)$. In the simple case when A has orthonormal rows, the positive eigenvalues of $A^T A$ are all 1 so the positive eigenvalues of the Hessian are all 4.

The dimension of the subspace of directions of zero curvature is large; thus there is an increased probability of encountering search directions for which the change in gradient would be tiny, resulting in very large values of α_k for the next iteration. For this reason it is important to have a reasonable estimate for $\alpha_{\max}$ so that numerical precision is retained throughout the algorithm.

Based on arguments from Section 3.3.3, we have an upper bound on each element of the vector x. Furthermore, we have both upper and lower (simple) bounds on the vectors u and v. Because u and v live inside a box (feasible region) we can also place a more realistic bound on the maximum allowable step length rather than resorting to $\alpha_{\max} = 10^{30}$. This step length is the distance from the origin to the furthermost corner of the box, (the point $\frac{b^T b}{\lambda}\mathbf{1}$). Let p be the search direction. The maximum step length is

$$\alpha_{\max} \;=\; \text{ub} \times \frac{\sqrt{n}}{\|p\|} \;=\; \frac{b^T b}{\lambda} \; \frac{\sqrt{n}}{\|p\|}. \tag{3.25}$$

Alternatively, the upper bound (3.22) may be used in place of (3.21) to determine $\alpha_{\max}$ if the rows of A are known to be orthonormal. Using (3.25) as the maximum step length ensures that numerical precision is likely to be maintained if a direction of zero (or near zero) curvature is encountered.

Remarks:

1. Finding a direction of zero curvature is not an indication that x_k is near the solution. Most often the direction will simply be tangential to the contours.

2. If a direction of zero curvature is located, taking a large step in this direction is not a good idea. The problem is that with the BB formula, the step length will be infinite, which destroys numerical precision. A more realistic upper bound on the step length is essential.

3.3.6 THE PROJECTION OPERATOR

Another key feature of the algorithm we propose is the projection operator. Because we have a constrained optimization problem (3.23), once a search direction and step length have been

determined the projection operator ensures that the new candidate point is feasible. If we define the feasible set of (3.23) to be

$$\mathcal{F} = \{x : \text{lb} \leq x_j \leq \text{ub}\} ,$$

where lb and ub are lower and upper bounds respectively, then the projection operator onto $\mathcal{F}$ is

$$P_{\mathcal{F}}(x) = \text{mid}(\text{lb}, x, \text{ub}). \tag{3.26}$$

Here mid(lb,x,ub) is the vector whose jth component is the median of the set $\{\text{lb}, x_j, \text{ub}\}$. This operator ensures that any x is kept within the feasible region.

Figure 3.2 provides a 2-dimensional example to illustrate how the projection operator works, assuming lb= 0.

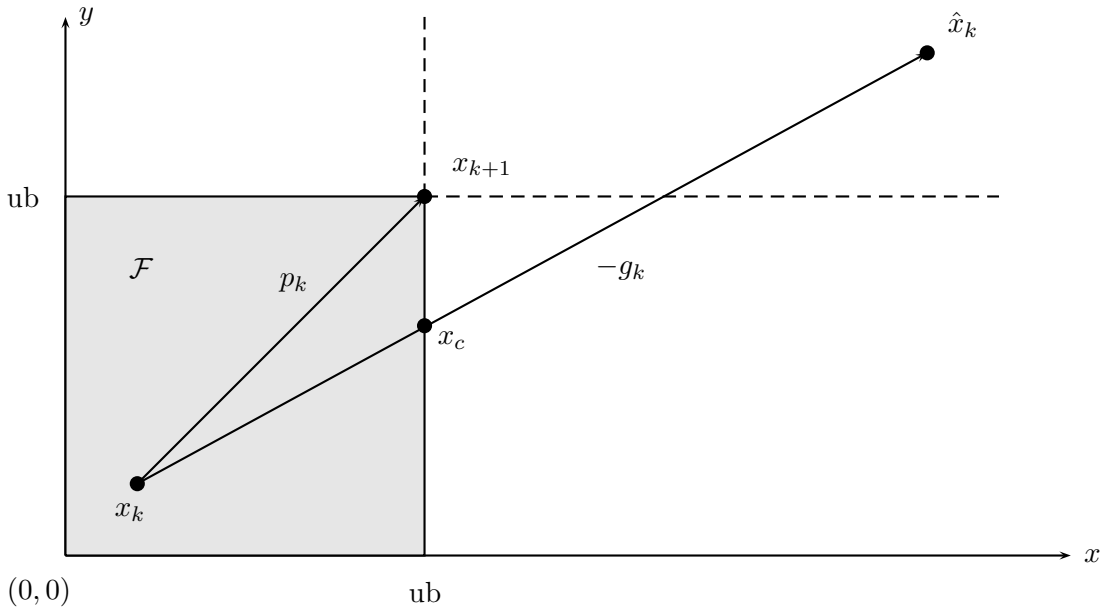


Figure 3.2: A diagram depicting the feasible region (shaded) and how the projection operator determines the new point, x_{k+1}.

Figure 3.2 can be further described as follows. At the kth iterate, evaluate $\hat{x}_k = x_k - \alpha g_k$, where α_k is found using (3.16) or (3.17). Suppose the point $\hat{x}_k$ lies outside the feasible region as shown in Figure 3.2. The point x_{k+1} is found by projecting $\hat{x}_k$ onto the feasible region using (3.26). This essentially means we have used the search direction p_k rather than the negative gradient.

There are also algorithms that use the so-called "chopped" gradient. These proceed by taking a step in the direction of the negative gradient and if the resulting point is outside the feasible region, then the gradient is essentially 'chopped off' when it hits a bound. This is also shown in Figure 3.2. Starting at the point x_k, we again evaluate $\hat{x}_k = x_k - \alpha g_k$. Rather than accepting $\hat{x}_k$ as the new candidate point, we move back to the point along the negative gradient that lies on the boundary of the feasible region. In Figure 3.2 this is the point x_c, which becomes the new candidate point x_{k+1}.

3.3.7 AN ADAPTIVE NON-MONOTONE LINE SEARCH

Many optimization algorithms employ a backtracking line-search loop to enforce reduction of the objective function value at each iteration. A key property of the Barzilai-Borwein algorithm is that a reduction in function value at every iteration is not required, so a traditional backtracking line-search loop is not appropriate. However, the algorithm is not guaranteed to converge in the box-constrained case without a line search, and [30] provides an example demonstrating that in the constrained case, the algorithm can in fact cycle, even for a quadratic with a positive definite Hessian.

Dai and Fletcher [30] have devised a so-called "adaptive non-monotone line search" loop that does not rely on the standard "sufficient descent property" [42], yet it ensures that global convergence can be proven. This non-traditional loop is employed in the algorithm we propose and does not appear to have been implemented in previous algorithms for Compressed Sensing. The following description of the so-called "adaptive non-monotone line-search" embedded in our algorithm is taken directly from [30]. For a function $f(x)$, set a reference value f_r, a candidate value f_c and the lowest objective value encountered so far f_{best}. (These are usually all initialized as $f_r = f_c = f_{best} = \infty$.) The update strategy is described by the following pseudo-code.

Procedure: Reference Value Updating Scheme

---

1. **Initialize:** $f_{best} = f_r = f_c = \infty, \;\; l = 0, \;\; L = 4$
2. **if** $f(x_k) < f_{\text{best}}$ **do**
3. $\quad f_{\text{best}} = f(x_k), \;\; f_c = f(x_k), \;\; l = 0$
4. **else**
5. $\quad f_c = \max(f_c, f(x_k)), \;\; l = l + 1$
6. $\quad$ **if** $l = L$ **do**
7. $\quad\quad f_r = f_c, \;\; f_c = f(x_k), \;\; l = 0$
8. $\quad$ **end**
9. **end**

This code resets the reference function value f_r to the candidate function value f_c if f_{best} has not been improved upon after L iterations. This is enough to enforce convergence while still allowing non-monotone behaviour.

The choice of parameter L is important. It represents the number of iterations allowed before a function decrease is enforced. For example, $L = 1$ implies that the function value must be

decreased at each iteration (a monotonic decrease in the objective function). As mentioned in Section 3.3.1, forcing a decrease in the objective function at each iteration can impair the practical performance of the Barzilai-Borwein algorithm. It is suggested in [30] that suitable choices are $L = 4$ or $L = 10$. Initial testing showed little difference between the two choices so in the numerical results presented in Section 3.4, $L = 4$ is used.

THE INTRODUCTION OF A SHIFT

Suppose a shift Δ is introduced. Then u and v become $u \leftarrow u + \Delta$ and $v \leftarrow v + \Delta$, and x remains unchanged as

$$x = (u + \Delta) - (v + \Delta) = u - v.$$

The gradient (3.19) is also independent of this shift although the objective function value (3.23) increases because

$$\lambda \sum_{i=1}^{n} ((u_i + \Delta) + (v_i + \Delta)) \quad = \quad \lambda \sum_{i=1}^{n} ((u_i + v_i) + 2\lambda\Delta \ .$$

Therefore in the algorithm presented in Section 3.3.9, the convergence test evaluates (3.9), rather than (3.23), because (3.9) gives a lower value of the objective function.

However, we must be careful that the values f_c, f_{best} and f_r are updated by using the larger objective value computed using u and v. Otherwise it is possible for the reference value f_r to increase, which affects the convergence of the algorithm.

3.3.8 STOPPING CRITERION

An appropriate stopping criterion for any optimization algorithm is paramount to ensure that an accurate solution is located. A standard approach is to terminate when the norm of the projected gradient is sufficiently small, indicating a stationary point has been found. However Kim et al. [60] discuss the use of the duality gap as a stopping criterion. We discuss this in more detail here.

In [60] problem (3.9) is re-written as

$$\min_{z,x} \qquad z^T z + \lambda \|x\|_1 \ , \tag{3.27}$$

$$\text{subject to} \qquad z = Ax - b \ . \tag{3.28}$$

The Lagrangian associated with this problem is

$$\mathcal{L}(z, x, \nu) \quad = \quad z^T z \quad + \quad \lambda \sum_{i=1}^{n} |x_i| + \nu^T (z - Ax + b) \ . \tag{3.29}$$

The dual function is the infimum of the Lagrangian. To find the infimum of (3.29) we first differentiate with respect to z:

$$\nabla_z \ \ \mathcal{L}(z, x, \nu) = 2z + \nu \ ,$$

which is zero at $z = -\frac{1}{2}\nu$. Substituting this into (3.29) gives

$$\begin{aligned}\mathcal{L}\left(-\frac{1}{2}\,\nu, x, \nu\right) &= \frac{1}{4}\,\nu^T\nu \;+\; \lambda\sum_{i=1}^{n}|x_i| + \nu^T\left(-\frac{1}{2}\,\nu - Ax + b\right)\\ &= -\frac{1}{4}\,\nu^T\nu + \nu^T b + \lambda\sum_{i=1}^{n}|x_i| - \nu^T Ax\end{aligned}$$

We now consider the terms involving x. Consider the subproblem

$$\inf_{x_i}\;\left(\lambda|x_i| - x_i(A^T\nu)_i\right) \tag{3.30}$$

If x_i is negative then (3.30) is bounded below by zero. However if x_i is positive then (3.30) is unbounded below unless $\|A^T\nu\|_\infty \leq \lambda$. This means that the Lagrange dual function is

$$\inf_{x,z}\;\mathcal{L}(z,x,\nu) \;=\; \begin{cases} -\frac{1}{4}\,\nu^T\nu + \nu^T b, & \text{if} \quad \|A^T\nu\|_\infty \leq \lambda,\\ -\infty, & \text{otherwise.}\end{cases}$$

The Lagrange dual problem of (3.27) is therefore

$$\begin{aligned}\max_{\nu} \quad & G(\nu) = -\tfrac{1}{4}\;\nu^T\nu + \nu^T b\\ \text{s.t.} \quad & |(A^T\nu)_i| \leq \lambda.\end{aligned} \tag{3.31}$$

For more on duality see [12, 42, 72]. A dual feasible point ν allows a lower bound on the optimal value of the primal problem to be calculated. Furthermore, as (3.9) satisfies Slater's condition, the optimal value of the primal objective is equal to the optimal objective of the dual. Thus we define the duality gap to be

$$\eta = \|Ax - b\|_2^2 + \lambda\|x\|_1 - G(\nu). \tag{3.32}$$

When the duality gap is sufficiently small, it indicates a solution has been located and as such, the duality gap can be used as a stopping criterion. Usually we use the relative change in the duality gap, which we define to be

$$\frac{\eta}{f(x)} < \epsilon \tag{3.33}$$

where $f(x)$ is given by (3.9) and ϵ is some user defined tolerance.

3.3.9 A BOX-CONSTRAINED GRADIENT PROJECTION ALGORITHM

Here we propose an algorithm for solving problem (3.23) using the ideas in the previous sections. We refer to this algorithm as the Box-Constrained Gradient Projection (BCGP) algorithm. The algorithm has been tailored to problem (3.23) with novel bounds on the elements of the solution vector and allowable step lengths, as well as a non-monotone line-search to ensure algorithm convergence.

Algorithm: Box Constrained Gradient Projection (BCGP)

1. **Initialize all parameters:** $\lambda = 0.01\|A^T b\|_\infty$, lb$= 0$, ub$= b^T b/\lambda$, initial point $x_0 = \mathbf{0}$, $\alpha_{max} =$ub$\sqrt{n}$, $\alpha_0 = 1/4$, $l = 0$, $f_r = f_c = f_{best} = \infty$, tolerance ϵ, iteration $k = 0$.

2. **Compute gradient and step length**. Let

$$z_k = \begin{bmatrix} u_k \\ v_k \end{bmatrix} \qquad \text{where} \qquad u_k = \max(x_k, 0), \qquad v_k = -\min(x_k, 0).$$

   Compute the gradient g_k using (3.19) and the step-length α_k using (3.16) for even k and (3.17) for odd k. From the point z_k, take the step

$$\hat{z} = z_k - \alpha_k g_k \ , \tag{3.34}$$

3. **Projection Operator**. Project $\hat{z}$ on the feasible region using the projection operator (3.26):

$$z_P = \begin{bmatrix} u_P \\ v_P \end{bmatrix} \ = \ P_{\mathcal{F}}(\hat{z}) \ = \ \text{mid}(\text{lb}, \hat{z}, \text{ub}).$$

4. **Perform convergence test**. Compute the primal objective value f, using

$$f = \|A(u_P - v_P) - b\|_2^2 + \lambda\|u_P - v_P\|_1 \ .$$

   Calculate the duality gap (3.32) and check whether the relative gap is less than ϵ. If so, exit the algorithm, otherwise continue with the following steps.

5. **Adaptive non-monotone line search**. If $f < f_r$ go to the next step. Otherwise for the search direction $p_k = z_P - z_k$, and perform a backtracking line-search,

$$z^+ = z_k + \beta p_k \ ,$$

   (where $\beta = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots$), until $f(z^+) < f_r$.

6. Update the reference values l, f_r, f_c and f_{best} according to the pseudo-code described in Section 3.3.7. Rename: $z_{k+1} = z^+$.

7. **End of Loop**. Update: $k \leftarrow k + 1$, $z_k \leftarrow z^+$, $x_k \leftarrow u_P - v_P$. Return to step 2.

3.4 NUMERICAL EXPERIMENTS

In this section we present numerical results obtained using the algorithm outlined in Section 3.3.9. These results illustrate the performance of the BCGP algorithm and how it compares with other recent algorithms, namely GPSR [40], SpaRSA [104] and FPC-BB [55] algorithms. Each of the algorithms compared in this section was coded in MATLAB, and all experiments were conducted on an Intel Xeon 3.2GHz processor with 4GB of memory, under Linux.

3.4.1 A SIMPLE SIGNAL RECONSTRUCTION PROBLEM

The first numerical example is a sparse signal recovery problem. A signal $x \in \mathbb{R}^{4096}$ consisting of 160 randomly placed spikes of amplitude ± 1 was generated. A measurement matrix $A \in \mathbb{R}^{1024\times 4096}$ (representing 1024 observations of the signal x) was constructed with Gaussian $\mathcal{N}(0,1)$ entries, and then the rows were orthonormalized (as in [17, 60]). The observation vector b was formed according to (3.1), where w was drawn from a Gaussian distribution with zero mean and variance $\sigma^2 = 10^{-4}$. The regularization parameter λ was chosen to be

$$\lambda = 0.01\|A^T b\|_\infty.$$

As discussed in Section 3.3.7, the value $L = 4$ was used along with a tolerance of $\epsilon = 10^{-6}$ on the relative duality gap. The algorithm was initialized at the origin: $x_0 = 0$. Figure 3.3 shows the signal reconstruction results. The top plot shows the original signal. The middle plot shows the signal reconstructed using the BCGP algorithm which does an excellent job of finding the positions of the non-zero components in the signal. The bottom plot shows the minimum energy solution $x = A^T b$. Figure 3.3 also gives the mean squared error, MSE, for both signal reconstructions. Here we follow [40] and define

$$\text{MSE} \;=\; \frac{1}{n}\|x - \hat{x}\|_2^2 \;,$$

where x is the original signal and $\hat{x}$ is the reconstructed signal. The BCGP algorithm finds a solution with a low MSE, indicating an accurate reconstruction.

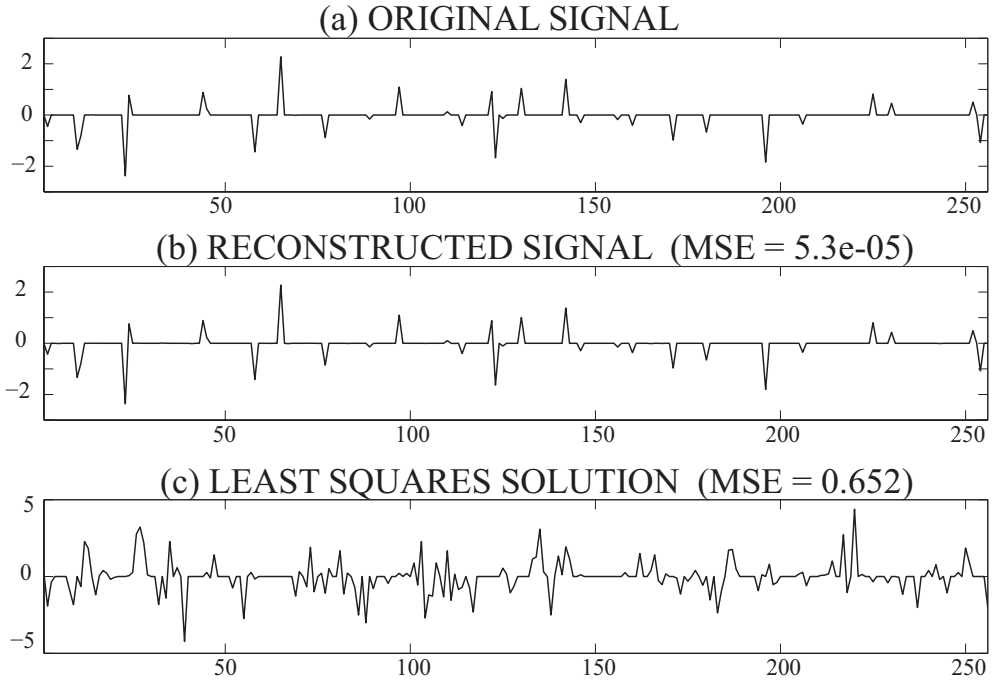


Figure 3.3: Sparse signal reconstruction. From top to bottom: the original signal; the reconstruction from noisy observations; the minimum energy solution $x = A^T b$.

3.4.2 δ–ρ PHASE PLOTS

Phase plots are a way of demonstrating how accurately the BCGP algorithm is able to reconstruct signals as the number of observations, and the sparsity level, varies. To create phase plots for the BCGP algorithm the following problem set-up was used. The length of the signal was set to $n = 400$ and a $\rho = s/m$ by $\delta = m/n$ grid was created. (Here n is the signal length, m is the number of observations and s is the number of spikes in the signal.) For each of the (ρ, δ) coordinates on the grid, the BCGP algorithm solved 40 signal reconstruction problems. We follow the approach in [55] in choosing error thresholds: in the noiseless case each element in the signal must be reconstructed to 10^{-4} accuracy, while in the noisy case the condition $\|x - x_s\|_2/\|x_s\|_2 < 10^{-2}$ is used. The colour on the grid represents the accuracy of the BCGP signal reconstruction, where darker shades represent poorer reconstruction.

The phase plots are shown in Figure 3.4. The first row ((a) and (b)) represents experiments on randomly generated matrices with Gaussian entries, while the second row ((c) and (d)) represents experiments on partial Discrete Cosine transform (partial DCT) matrices. In the first column ((a) and (c)) the results in the noiseless case are given, while in the second column ((b) and (d)) the reconstruction from noisy measurements is given.

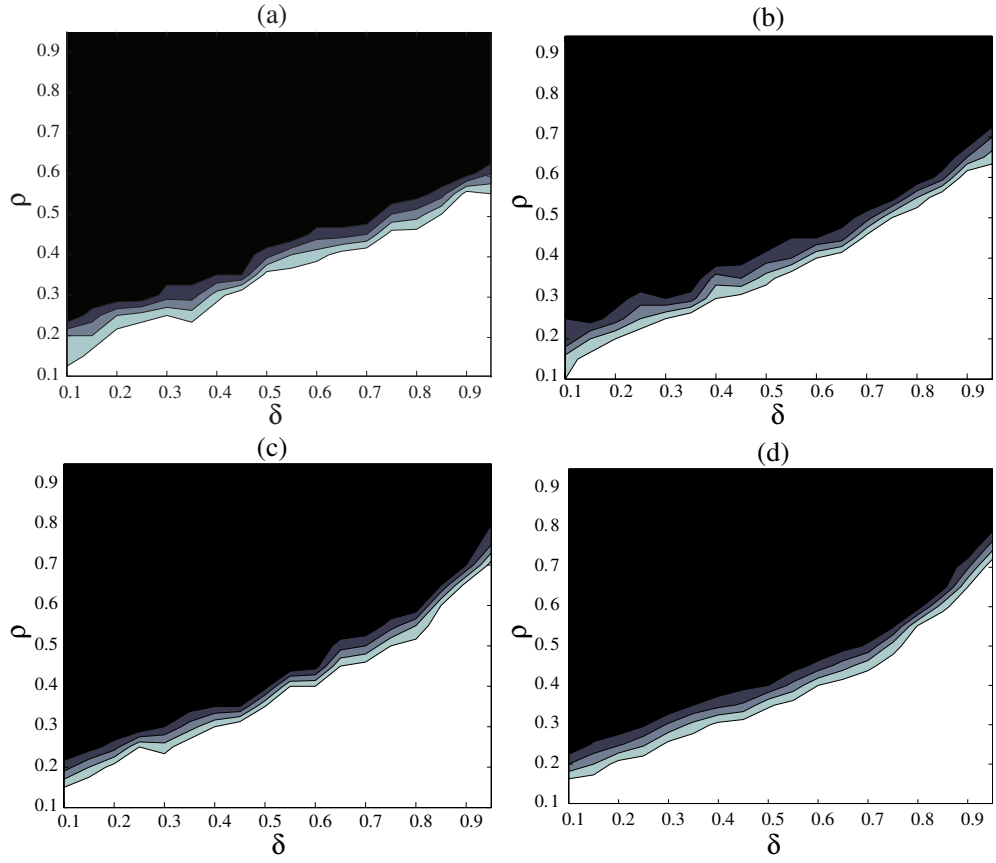


Figure 3.4: Phase plot of the BCGP algorithm being tested on (a) A Gaussian matrix with no noise; (b) A Gaussian matrix with noise $\sigma = 10^{-3}$; (c) A Partial DCT matrix in the noiseless case; (d) A Partial DCT matrix with noise $\sigma = 10^{-3}$.

The white regions in Figure 3.4 show parameter ranges where 100% of the reconstruction attempts were successful (x satisfied the error bounds). Black regions show that reconstruction was not possible. Middle shades show varying success percentages.

Figure 3.4 shows that the BCGP algorithm accurately reconstructs signals over a wide range of parameter space, demonstrating the robustness of the algorithm.

3.4.3 THE REGULARIZATION PARAMETER

Here we present an example to demonstrate the sensitivity of the BCGP algorithm as the regularization parameter λ varies.

An observation matrix was chosen to be a sub-matrix of a DCT matrix. The matrix had dimensions $2^{10} \times 2^{12}$. A sparse signal with 2^7 spikes of height ± 1 was generated and the observation vector b had noise added. A range of values of the regularization parameter λ were tested according to $\lambda = \gamma \|A^T b\|_\infty$ with $\gamma = 0.1, 0.05, 0.01, 0.005, 0.001$. The algorithms tested in this experiment all use different stopping criteria. To ensure the experiment was as fair as possible, the parameter values for each algorithm were chosen in such as way so that when the algorithm terminated, the recovered signal $\hat{x}$ gave the MSE$\approx 10^{-5}$. Table 3.1 compares the runtimes of the MATLAB implementation of our method and the existing methods using the different γ values.

Table 3.1: Average CPU Times (in seconds) over 10 runs on the Partial DCT Matrix Experiment with $m = 2^{10}$ and $n = 2^{12}$, using varying values of the regularization parameter λ.

	$\gamma = 0.1$	$\gamma = 0.05$	$\gamma = 0.01$	$\gamma = 0.005$	$\gamma = 0.001$
BCGP	0.115	0.120	0.965	2.545	7.820
SpaRSA	0.120	0.140	0.867	2.606	8.980
GPSR	0.139	0.150	1.147	1.591	21.350
FPC-BB	0.353	0.290	1.488	3.622	5.750
l_1-ls	2.090	2.170	8.391	15.694	28.290

The BCGP algorithm performs well using a range of γ values and produces an accurate signal reconstruction even for small values. Clearly all algorithms perform more quickly with a large γ (≈ 0.1) value.

3.4.4 SCALABILITY ASSESSMENT

The next experiment is a scalability assessment that considers the computational effort required as problem size increases. Several random sparse matrices are considered whose entries are normally distributed. The dimensions of these matrices are $m \times n$, where n ranges from 10^4 to 5×10^6 and $m = 0.2n$. The sparsity of A is controlled to have approximately $30n$ nonzero elements. For each data set, x is also generated to be sparse with $0.01m$ randomly

placed components of height ± 1. The observed measurements y are formed using (3.1), and are corrupted with Gaussian noise of variance $\sigma^2 = 10^{-4}$. For each data set the regularization parameter used is $\lambda = 0.1\|A^T b\|_\infty$. Note that from (3.9), $\lambda = 2\tau$ and for the FPC-BB algorithm (`http://www.caam.rice.edu/~optimization/L1/fpc/#soft`), $\mu = 1/\tau$ was used.

The algorithms BCGP, GPSR, SpaRSA, and FPC-BB were tested using this experiment set-up. For each size n, twenty matrices were generated and the average CPU time for each algorithm was found. The signal length vs average CPU time for each algorithm is shown in Figure 3.5. We see that the BCGP algorithm is performing very competitively with the other algorithms.

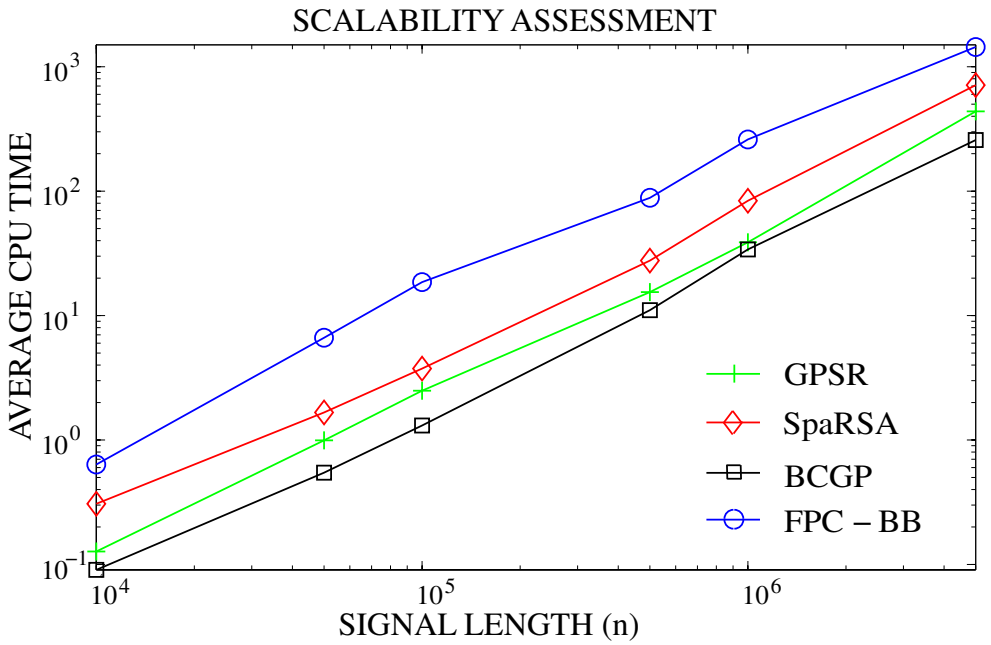


Figure 3.5: A plot of the signal length versus CPU time (measured in seconds) for the BCGP, GPSR, and SpaRSA algorithms. The BCGP algorithm is the fastest algorithm for this choice of regularization parameter λ.

The data from the scalability assessment was also used to estimate the computational complexity of each algorithm. That is, assume the computational cost is $\mathcal{O}(n^\alpha)$ and estimate α based upon CPU times as n increases. Table 3.2 gives the empirical estimates of the exponent α, and the average CPU time for each method when $n = 5 \times 10^6$.

Table 3.2: Empirical estimate of the exponent of each algorithm (over 20 runs) on the sparse scalability assessment. The average CPU time is also given when $n = 5 \times 10^6$.

Algorithm	α Value	CPU Time
BCGP	1.300	257.74
GPSR	1.286	437.62
SpaRSA	1.257	711.68
FPC-BB	1.226	1442.30

Table 3.2 shows that there is very little difference between the empirical computational complexity of each algorithm, and that the BCGP algorithm is significantly faster than its competitors for the given λ and n. An indirect comparison with other codes can be made by referring to the numerical results sections in [104].

3.5 CONCLUSION

The problem of finding sparse solutions to large underdetermined linear systems in the presence of noise is an important one in signal and image processing. In this chapter we have introduced a new algorithm called the Box-Constrained Gradient Projection (BCGP) algorithm, which aims to solve the convex optimization problems arising in compressed sensing. The algorithm uses the Barzilai-Borwein step lengths and employs novel upper bounds on the solution vector. Because of this upper bound, the signal recovery problem can be formulated as a box-constrained quadratic programme.

We derived a result that shows, for the compressed sensing problem (3.18), the step length (3.17) is fixed if the observation matrix has orthonormal rows. This explains why a strategy that uses the two Barzilai-Borwein step lengths in an alternating manner may not be appropriate in this special case.

The algorithm provides safeguards in the case where the Hessian is positive semi-definite. These safeguards include the incorporation of an adaptive non-monotone line search, upper bounds on the maximum step length, and uses a stopping criterion which provides a known bound on the error in the primal objective value. The algorithm proposed has guaranteed convergence properties because of the inclusion of the adaptive non-monotone line search.

The numerical results in Table 3.1 show that our algorithm is competitive with other existing algorithms and noticeably faster when the problem size is small. These results are encouraging because the underlying algorithm does not include any continuation schemes, which could potentially improve performance further.

As the scalability experiment shows, as the magnitude of the problem is increased our method retains accuracy and efficiency, even for small values of the regularization parameter.

3.5.1 FUTURE RESEARCH

The area of compressed sensing is still growing at a rapid pace, so there is plenty of room for further research in this area. Some possible ideas for future research include the introduction of a continuation scheme (which varies the parameter λ through a range and then restarts) to speed up the algorithm even further. Another idea is to investigate how the algorithm may be adapted to account for complex signal reconstruction. This would mean that we could use partial Fourier matrices as the measurement matrices, which is advantageous because matrix-vector products can be computed very quickly in this case. Algorithms that can handle complex data can also be used in practical situations such as MRI, where the data is

known to be complex.

Another possibility is to investigate the link between the observation subset selection problem and compressed sensing. In particular, it would be interesting to see whether an observation subset selection approach resulted in matrices which satisfy the RIP condition of compressed sensing.

CHAPTER 4

RECONSTRUCTION OF STRUCTURED SIGNALS

Compressed sensing aims to find sparse solutions to underdetermined systems of equations. One of the reasons for its popularity is that many real-world signals and images, for example, Magnetic Resonance images, can be sparsely represented in some basis. This leads us to consider other signals that may not be sparse in a particular basis but whose important features can be represented in a concise way. In particular, piecewise constant signals are not sparse, but if there are sufficiently few discontinuities in the signal, only a small amount of information, namely the location and heights of the discontinuities, will allow the signal to be reconstructed accurately. This idea can be extended to images that have piecewise homogeneous regions; for example, in computed tomography [27, 61, 89, 109].

As for sparse signals, there has been a great deal of interest in reconstructing piecewise constant signals that have been under-sampled [9, 57, 93]. The reconstruction problem is again modeled by the system of equations

$$b = Ax + w, \tag{4.1}$$

where $A \in \mathbb{R}^{m\times n}$ is a measurement matrix ($m \ll n$), $b \in \mathbb{R}^m$ is a vector of observations and $w \in \mathbb{R}^m$ is additive noise. The aim is to determine the solution $x \in \mathbb{R}^n$.

To be able to reconstruct general signals accurately, sampling at the Nyquist rate is essential. Compressed sensing only allows us to reconstruct signals from under-sampled data because the signal to be reconstructed has a particular *known* structure that is being exploited. Usually the structure is sparsity, but, as mentioned, piecewise constant signals may also be reconstructed using the ideas of compressed sensing. The aim of this chapter is to investigate how one might reconstruct signals with a particular structure, from a limited set of observations.

Piecewise constant signals can be described using only a small amount of information. When reconstructing these types of signals from underdetermined systems of equations, rather than using an ℓ_1 minimization or regularization approach, we instead use a *total variation* approach. The optimization problem is formulated as

$$\min_x \|x\|_{\mathrm{TV}} \quad \text{subject to} \quad Ax = b\,, \tag{4.2}$$

where the 'TV-norm' is defined (for a 1-dimensional signal) to be

$$\|x\|_{\mathrm{TV}} = \sum_{i=1}^{n-1} |x_{i+1} - x_i| \ . \tag{4.3}$$

Note that this is not a true norm, although we follow the current literature and refer to it as the TV-norm. The TV-norm searches for the solution that varies the least, and its use in signal processing goes back some time (see for example [83]). Clearly this is particularly useful for piecewise constant signals. As for compressed sensing, the optimization problem (4.2) may similarly be posed as a regularized least squares problem

$$\min_x \|Ax - b\|_2^2 + \lambda \|x\|_{\mathrm{TV}} \ , \tag{4.4}$$

where $\lambda > 0$ is a regularization parameter.

A number of algorithms aim to solve problems (4.2) and (4.4) to reconstruct a piecewise constant signal from a small number of observations. These methods include an interior point method for nonconvex minimization problems when the image is known to be piecewise constant [71], and an Edge Guided Compressive Sensing (EdgeCS) algorithm that solves a weighted TV-norm minimization problem [54, 99]. There is also an algorithm that uses Toeplitz and Circulant matrices with an augmented Lagrangian and alternating direction algorithm to reconstruct piecewise constant signals and images [108]. This algorithm can detect dominant 'directions' in images. Another example is [2], which uses a total variation reconstruction and compares three algorithms that use difference approximations of the nonlinearity.

4.1 STRUCTURED SIGNALS

The aim of this chapter is to reconstruct signals or images that are a combination of a signal with certain known properties, and a sparse part. The added complication is that only m ($\ll n$) observations of the signal are taken so the reconstruction falls under a compressed sensing framework. The simplest example is when the signal of interest is the combination of a signal of constant height, plus a sparse component. This can be written as

$$x = x_s + \alpha \mathbf{1}, \tag{4.5}$$

where α is the constant signal offset, $\mathbf{1}$ is a vector of ones and x_s is a sparse signal. The signal x is not sparse, in the sense that none of its components is likely to be zero. However after an appropriate vertical shift, this signal becomes sparse. The added complication is that the shift is not known.

Expression (4.5) can be written more generally as

$$x = x_s + v, \tag{4.6}$$

where again x_s is sparse, but all we know about v is that it lies in some subspace M. (In (4.5), M is the one-dimensional subspace spanned by the vector $\mathbf{1}$.)

These types of problems, where the signal or image can be separated into a sparse part plus a dense part, arise commonly in medical imaging. For example, when an image is split into two parts for diagnosis, the dense part represents the 'healthy' part of the image while the sparse part represents any pathology [16]. In this situation both parts of the reconstructed signal are equally important. In other applications the dense part, or the sparse part of the signal, may be more important. This situation arises in astronomical imaging, where the dense signal may be of importance and the sparse part of the signal represents unwanted bursts of radiation [11, 90].

The problem of signal reconstruction when the subspace M is unknown is often referred to as 'blind' or 'unsupervised' reconstruction and is also the topic of much current research. Examples include the reconstruction of multi-band signals that do not require the band locations to be known in advance [68], the reconstruction of a blurred image when the blurring function is unknown [22], and unsupervised segmentation for linear inverse problems [71].

4.1.1 CHAPTER OUTLINE

In this chapter we discuss three signal reconstruction problems of increasing difficulty. Section 4.2 considers the simplest problem of reconstructing a signal of the form (4.5) and determining the unknown offset α.

The method used in Section 4.2 is generalised in Section 4.3 to the case where the unknown signal x is of the form (4.6), and the subspace in which v lies is known.

In Section 4.4 we consider the problem of signal reconstruction when the subspace M is unknown but can be specified in terms of a number of parameters. This 'blind' reconstruction case seems considerably more difficult and we restrict ourselves to signals that are piecewise constant, but with unknown endpoints, which have to be determined. (In essence, these are the parameters which describe M.)

Finally, Section 4.6 presents numerical examples that show reconstruction is possible to a high level of accuracy in each of the above cases.

4.2 THE VERTICAL OFFSET PROBLEM

We start with the simplest problem so far described: to reconstruct a signal that is sparse relative to some *unknown* offset α. We are looking for a solution to (4.1) of the form $x = x_s + \alpha\mathbf{1}$.

Following a standard compressed sensing procedure, the signal reconstruction problem can be reformulated as an unconstrained, non-differentiable, convex optimization problem

$$\min_{x_s,\alpha} \quad \|Ax_s + \alpha A\mathbf{1} - b\|_2^2 \; + \; \lambda\|x_s\|_1. \tag{4.7}$$

This problem has $n+1$ unknowns, the elements of $x_s \in \mathbb{R}^n$ and the value of $\alpha \in \mathbb{R}$, all of which we would like to recover accurately.

Remark: For this problem we assume that $A\mathbf{1} \neq \mathbf{0}$, where $\mathbf{0}$ is the zero vector. (That is, the row sums of A are not all 0.) This is an essential assumption because otherwise there is no unique "best" solution to (4.7). To see this, suppose $A\mathbf{1} = \mathbf{0}$. Then

$$Ax = Ax_s + \alpha A\mathbf{1} = b\,,$$

so $Ax_s = b$, which is satisfied for any α. Thus α can only be uniquely determined if $A\mathbf{1} \neq \mathbf{0}$.

Problem (4.7) can be split into

$$\min_{x_s} \quad \{\min_{\alpha} \quad \|Ax_s + \alpha A\mathbf{1} - b\|_2^2\} + \lambda\|x_s\|_1. \tag{4.8}$$

This means that to solve problems of the form (4.7) we instead solve each of the two subproblems described by (4.8) in turn. The first step is to eliminate α. The problem then becomes one of recovering the sparse vector x_s, which can be achieved using standard compressed sensing techniques. Once x_s has been determined, the second step is to recover α. For this class of problems, the second step is very simple.

Consider the subproblem

$$\min_{\alpha} \quad \|Ax_s + \alpha A\mathbf{1} - b\|_2^2\,. \tag{4.9}$$

Let $c = A\mathbf{1}$ and $d = b - Ax_s$. Then (4.9) becomes

$$\min_{\alpha} \quad \|\alpha c - d\|_2^2\,,$$

where

$$\|\alpha c - d\|_2^2 = (\alpha c - d)^T(\alpha c - d) = \alpha^2\, c^T c - 2\alpha\, c^T d + d^T d\,.$$

Differentiating with respect to α gives

$$\nabla_\alpha(\alpha^2\, c^T c - 2\alpha\, c^T d + d^T d) \;=\; 2\alpha c^T c - 2c^T d\,,$$

which is zero when

$$\alpha \;=\; \frac{c^T d}{c^T c}\,. \tag{4.10}$$

Now the term $\alpha A\mathbf{1}$ can be written as

$$\alpha A\mathbf{1} \;=\; \left(\frac{cc^T}{c^T c}\right)(b - Ax_s)\,. \tag{4.11}$$

Substituting (4.11) into problem (4.7) gives

$$\min_{x_s} \quad \|P(Ax_s - b)\|_2^2 \;+\; \lambda\|x_s\|_1,$$

where

$$P = I - \frac{cc^T}{c^T c}\,.$$

Notice that $P = P^T$ and $P^2 = P$ and $\{c\}$ is a linearly independent (and orthogonal) set, so P is a projection matrix. The vector c can be thought of as an orthogonal basis for a subspace of $\mathbf{R}^m$. Let $M = \text{span}\{c\}$. Then M has an orthogonal complement $M^\perp$ and $\text{proj}_{M^\perp}(c) = \mathbf{0}$, where $\mathbf{0}$ is the zero vector.

Taking this approach, the term $\alpha A\mathbf{1}$ has effectively been removed from the original problem (4.7). The next step is to reconstruct the sparse signal x_s. This can be achieved using a compressed sensing set-up:

$$\min_{x_s} \|PAx_s - Pb\|_2^2 + \lambda\|x_s\|_1 \ . \tag{4.12}$$

Chapter 3 described an algorithm to accurately and efficiently solve problems of this form, namely the BCGP algorithm [15], that was used in the numerical examples in Section 4.6.

After x_s is determined, it is straightforward to recover α using (4.10). The signal x, of the form (4.5), can then be reconstructed.

Remark: Solving the optimization problem (4.12) does not require significantly more computational effort compared with the standard compressed sensing problem. The matrix-vector product Pb is computed once before the algorithm is initialized. The multiplication PAx_s is computed as two matrix-vector products — this means there is only one extra cheap matrix-vector product per iteration in the BCGP algorithm.

4.2.1 VERTICAL OFFSET EXAMPLES

The following example demonstrates the signal reconstruction process when the signal is of the form $x = x_s + \alpha\mathbf{1}$. This is a simplified example of the practical problem of spectrum reconstruction arising in x-ray imaging [41]. In this application the primary photon energy spectrum of the x-ray source should be reconstructed using information about the energy deposition on the detector. The primary spectrum is somewhat similar to a bell curve with distinctive spikes.

For this example, a partial DCT matrix $A \in R^{1024\times4096}$ and an observation vector $b \in R^{1024}$ (measured in the presence of normally distributed random noise, $\sigma = 10^{-3}$), were generated. The signal $x = x_s + \alpha\mathbf{1}$ to be reconstructed was a combination of a sparse vector $x_s \in R^{4096}$ with 128 spikes of height ±1, and a constant vector of (randomly generated) height $\alpha = 0.70733$. The results are shown in Figure 4.1.

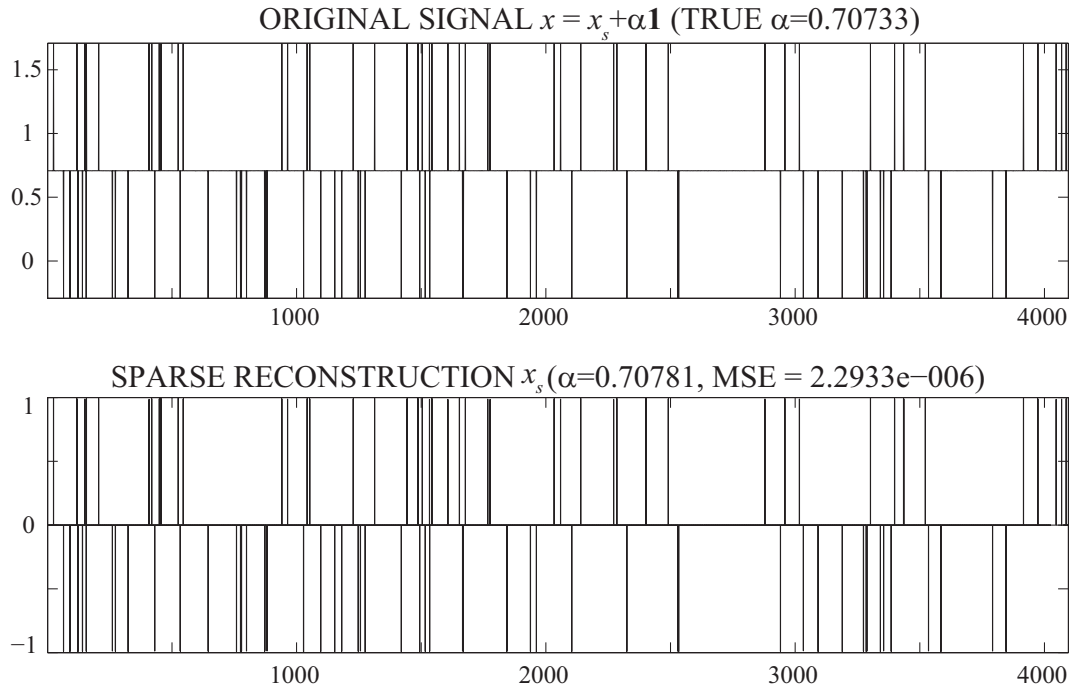


Figure 4.1: The top plot shows the original signal with offset $\alpha = 0.70733$. The bottom plot shows the sparse reconstruction x_s and the offset estimate $\alpha = 0.70781$. The algorithm accurately locates the positions of the spikes in the signal x_s, and the reconstruction $\hat{x} = x_s + \alpha\mathbf{1}$ (although not displayed here) leads to a small mean squared error (MSE).

Figure 4.1 shows the original signal x and the sparse part of the signal x_s, which has been reconstructed accurately. It also gives the offset $\alpha = 0.70781$ (estimated using (4.10)), which is close to the true value. Recall that the mean squared error, MSE, is a measure of the accuracy of the reconstructed signal and is defined as MSE $= \frac{1}{n}\|x - \hat{x}\|_2^2$ where x is the original signal and $\hat{x}$ is the reconstructed signal. The reconstructed signal $\hat{x} = x_s + \alpha\mathbf{1}$ (although $\hat{x}$ is not displayed in the figure) gives a value MSE$= 2.3 \times 10^{-6}$.

The next example considers signal reconstruction when the subspace M is spanned by a single *known* vector v, but instead of a constant vector, the entries of v form an arithmetic progression. In this case, the signal is modeled using $x = x_s + v$, where the vector v has entries of the form $v_1 = 0$ and $v_j = v_{j-1} + 0.001$ $(j = 2, \ldots, n)$ and x_s is sparse. The matrix $A \in \mathbb{R}^{200\times 1000}$ is a partial DCT matrix and $b \in \mathbb{R}^{200}$ is an observation vector measured in the presence of normally distributed noise $\sigma = 10^{-3}$. The sparse vector x_s has 20 randomly placed spikes of height ± 1. The results are shown in Figure 4.2.

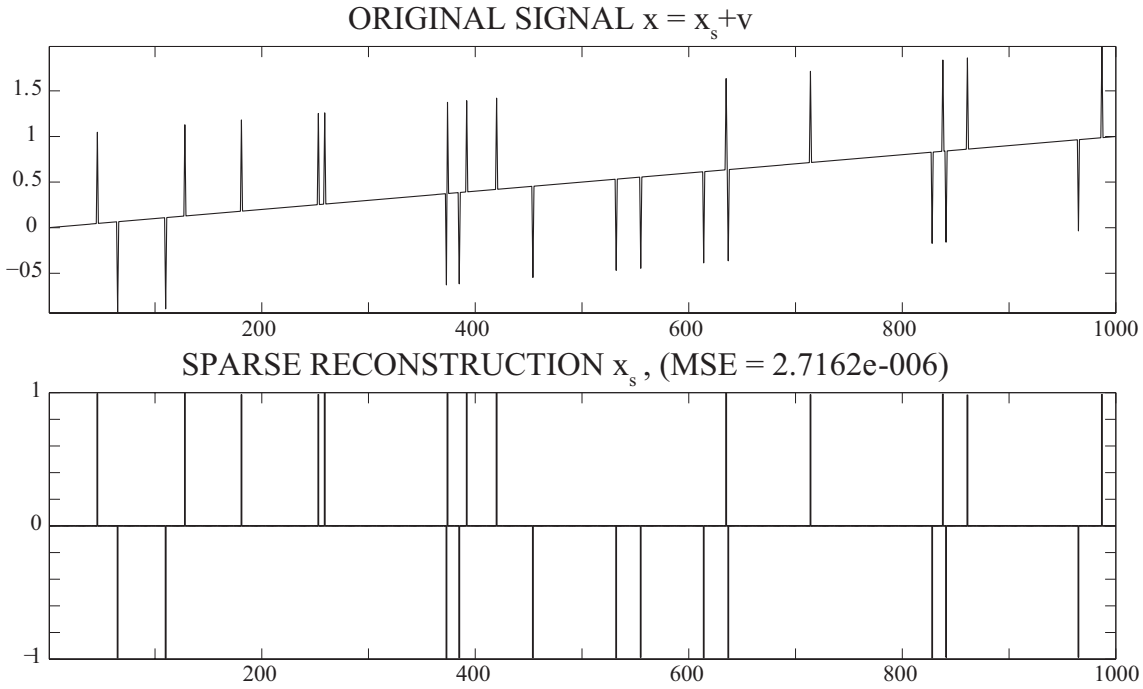


Figure 4.2: The first plot shows the original signal $x = x_s + v$. The second plot is the sparse reconstruction x_s. The location and height of each spike in x_s has been found accurately. The reconstruction $\hat{x} = x_s + v$ (although not displayed here) has a small MSE, which verifies that the reconstruction is accurate.

Figure 4.2 shows the original signal x and the reconstructed sparse part x_s. In this experiment it was assumed that v was known, so the reconstruction $\hat{x} = x_s + v$ has not been displayed, although the reconstruction $\hat{x}$ gave the MSE$= 2.7 \times 10^{-6}$. The MSE is low, indicating a high quality reconstruction. The high quality reconstruction is visible in the figure—the location and height of each spike in x_s has been reconstructed accurately.

4.3 A KNOWN SUBSPACE

This technique readily extends to the case where the component $\alpha\mathbf{1}$ of (4.7) is replaced by a vector v in some *known* subspace that is spanned by more than one vector. Specifically, if v is a fixed vector, $v \notin \text{null}(A)$, and $c = Av \neq 0$, reconstruction of a signal of the form $x = x_s + v$ is possible from a limited number of data measurements.

In this section we specifically consider the reconstruction of signals that are a combination of a piecewise constant part and a sparse signal. The added complication is that only m ($\ll n$) observations of the signal are made. Mathematically we wish to determine a solution to (4.1) of the form $x = x_s + v$, where v is piecewise constant and x_s is sparse. Because v is a piecewise constant vector it may be decomposed as

$$v = \sum_{l=1}^{p} \alpha_l u_l \quad (p \ll n), \tag{4.13}$$

where the vectors $u_1, \ldots, u_p$ (made up of zeros and ones) represent the various flat areas of the signal, while the parameters $\alpha_1, \ldots, \alpha_p$ represent the height of each flat region. We write (4.13) in matrix form as $v = U\underline{\alpha}$, where $U \in \mathbb{R}^{n\times p}$ is the matrix whose lth column is u_l and $\underline{\alpha} \in \mathbb{R}^p$ is the vector whose lth element is α_l.

To make this more concrete, consider the following example. Let $v \in \mathbb{R}^6$ be

$$v = \begin{bmatrix} 2 & 2 & 2 & 4 & 4 & 4 \end{bmatrix}^T.$$

Then v has one discontinuity and two flat areas, so $p = 2$ and

$$u_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}^T, \qquad u_2 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}^T.$$

Here $\alpha_1 = 2$ and $\alpha_2 = 4$ so that $v = \alpha_1 u_1 + \alpha_2 u_2$.

The aim is to solve the system $Ax = Ax_s + Av = Ax_s + AU\underline{\alpha} = b$. Following a standard compressed sensing procedure, the reconstruction problem can be reformulated as

$$\min_{x_s, v} \ \|Ax_s + Av - b\|_2^2 + \lambda \|x_s\|_1 \ , \tag{4.14}$$

which is an unconstrained convex optimisation problem in $2n$ unknowns. We split problem (4.14) into two subproblems in the following way:

$$\min_{x_s} \quad \{\min_{v} \quad \|Ax_s + Av - b\|_2^2\} + \lambda \|x_s\|_1. \tag{4.15}$$

To solve problems of the form (4.14) we can instead solve each of the two subproblems described by (4.15) in turn. The first step is to try to eliminate v from (4.15) and this is discussed further in the next subsection. Once v has been removed from the problem, the sparse vector x_s can be found and then v is straightforward to recover.

4.3.1 THE PROJECTION STEP

The aim of this section is to determine a way of removing v from problem (4.15) so that we are solving a simpler optimization problem involving only x_s.

If the vectors $u_1, \ldots, u_p$ are known, then so too is the subspace $M = \text{span}\{Au_1, \ldots, Au_p\}$. (This does not mean that we know the vector v because we do not know $\alpha_1, \ldots, \alpha_p$ and these

are not simple to determine.)

Following the procedure in Section 4.2, form the projection matrix P to project onto the subspace $M^{\perp}$ (the orthogonal complement of M). Let

$$c_l = Au_l \quad \text{for} \quad l = 1, \ldots, p.$$

Then the projection matrix is of the form

$$P = I - \sum_{l=1}^{p} \frac{c_l c_l^T}{c_l^T c_l} \, .$$

The vectors $c_1, \ldots, c_p$ must be orthogonal for P to be an appropriate projection matrix. Therefore, define C to be the matrix whose columns are the vectors $Au_1, Au_2, \ldots, Au_p$, and write $C = YR$ for its economy QR decomposition [50, p. 223]. (Note that the vectors $Au_1, \ldots, Au_p$ must be linearly independent for R to be nonsingular.) The columns of the matrix Y (denoted by $y_1, \ldots, y_p$) represent an orthonormal basis for the subspace M. We define the matrix

$$P = I - \sum_{l=1}^{p} y_l y_l^T = I - YY^T, \tag{4.16}$$

which is the projection onto $M^{\perp}$. Premultiplying expression (4.1) by P gives

$$\begin{aligned} Pb &= PAx_s + \alpha PAv + Pw \\ &= PAx_s + w \, . \end{aligned}$$

Using the projection matrix (4.16) we have essentially 'removed' αAv from problem (4.14) so we have the familiar convex optimization formulation

$$\min_{x_s} \; \|PAx_s - Pb\|_2^2 + \lambda \|x_s\|_1. \tag{4.17}$$

Once problem (4.17) has been solved for x_s, the vector v can be recovered as follows. Let $\underline{\alpha}$ denote the vector whose components are $\alpha_1, \ldots, \alpha_p$. Then we have

$$d \; = \; b - Ax_s \; = \; Av \; = \; C\underline{\alpha}. \tag{4.18}$$

Problem (4.18) is an over-determined system of m equations in p unknowns so can be solved using the normal equations. Alternatively, because the economy QR decomposition has already been computed to determine the projection matrix P, back-substitution on the system

$$R\underline{\alpha} = Y^T(b - Ax_s) = Y^T d \, , \tag{4.19}$$

also gives the vector $\underline{\alpha}$. Once the vector $\underline{\alpha}$ has been determined, v is recovered through (4.13), and the underlying signal $x = x_s + v$ is reconstructed.

4.3.2 KNOWN SUBSPACE EXAMPLES

The following examples consider signal reconstruction when the subspace M is known. In this case, if the signal is made up of a piecewise constant part, then the location of its discontinuities are assumed to be known, and therefore, so too are the basis vectors $u_1, \ldots, u_p$ and the subspace M.

In these examples, the signal x is of the form $x = x_s + v$. In the first example v is a piecewise constant signal, while in the second example v is a vector whose entries form an arithmetic progression on piecewise regions. In both cases, the location of the discontinuities are known (i.e., the matrix U, and therefore the subspace M, is known), although the heights (i.e., the vector $\underline{\alpha}$) are unknown. In each case, a measurement matrix $A \in \mathbb{R}^{200\times 1000}$ with orthogonal rows and an observation vector $b \in \mathbb{R}^{200}$ were set up, with b measured in the presence of normally distributed Gaussian noise. In the first experiment the sparse signal x_s has 20 randomly placed spikes of height ± 1, while in the second experiment x_s has 10 randomly placed spikes of height $\pm\frac{1}{2}$. The signal reconstruction results are shown in Figures 4.3 and 4.4.

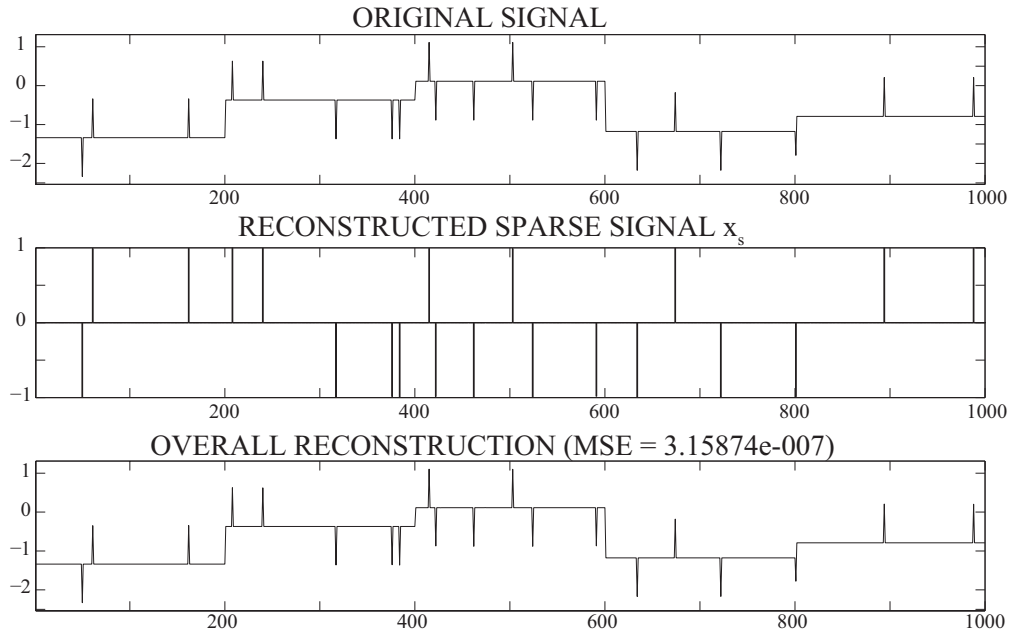


Figure 4.3: From top to bottom: The original signal $x = x_s + v$. The sparse reconstruction x_s found using the BCGP algorithm. The overall reconstruction $\hat{x} = x_s + v$.

Figure 4.3 shows the reconstruction of a signal that is a combination of a piecewise constant signal (v) and a sparse part (x_s). The reconstruction of x_s is accurate with all spikes located accurately. The overall reconstruction $\hat{x} = x_s + v$ is also very good, and this shows that the vector $\underline{\alpha}$ has also been recovered accurately. The MSE$\approx 10^{-7}$, and this supports our claim of a high quality reconstruction.

Figure 4.4 shows the reconstruction of a signal that is a combination of a sparse part x_s and a vector that is piecewise linear. The sparse part x_s and the overall signal are accurate.

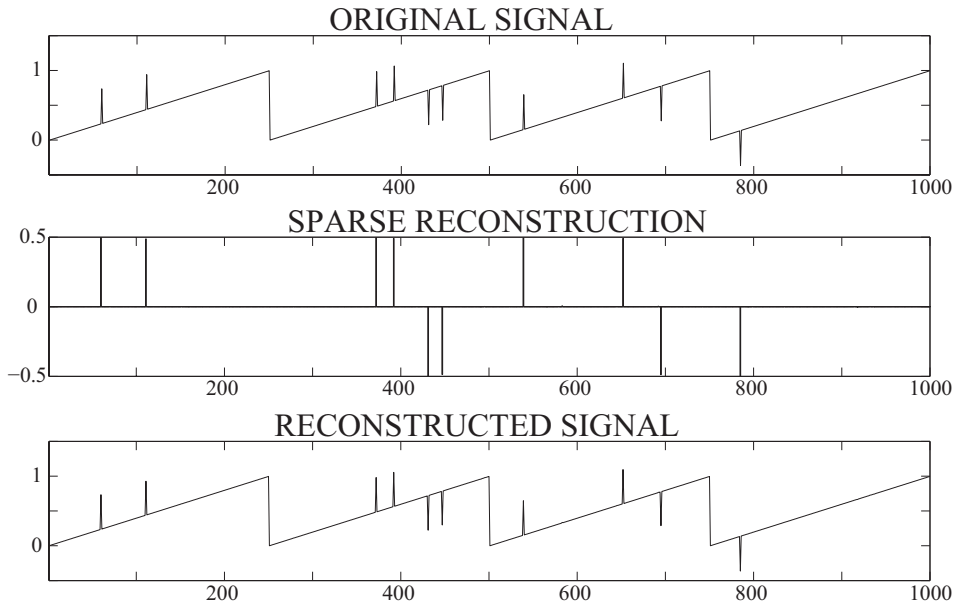


Figure 4.4: From top to bottom: The original signal x. The sparse reconstruction x_s using the BCGP algorithm. The reconstructed signal $x_s + v$.

4.4 BLIND SIGNAL RECONSTRUCTION

The problem of signal reconstruction when the subspace M is unknown is often referred to as 'blind' or 'unsupervised' reconstruction, and is the topic of much current research. So far we have assumed that in the decomposition $x = x_s + v$ the component v was in a known subspace. In the previous section, v was a piecewise constant function where the possible discontinuities were in certain known positions.

In this section we consider the more difficult problem of finding v when the subspace M is not known, but depends on some parameters that first have to be determined. Once they are found, the problem is solved by the procedure in Section 4.3. A typical problem of this nature occurs when v is a piecewise constant vector with jumps at unknown positions. These must first be identified as they act as the 'parameters' of the subspace M. We concentrate on this case for the remainder of this chapter. The signal x is decomposed as $x = x_s + v = x_s + U\underline{\alpha}$, where x_s, the columns of U (u_l), and $\underline{\alpha}$ (containing the values α_l), are all unknown.

At the beginning of the chapter, we introduced the total variation norm. Because we have a piecewise constant signal, adjacent signal entries are usually the same. That is, often $v_i - v_{i+1} = 0$. Furthermore, if $v_i - v_{i+1} \neq 0$ a discontinuity has been located. We now introduce the discrete derivative matrix D [56]:

$$D = \begin{pmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \\ & & & 1 \end{pmatrix}. \tag{4.20}$$

Left multiplication of a vector v by the matrix D yields a vector with entries $v_i - v_{i+1}$ (apart from the last entry). So (assuming $|v_n| \ll \|v\|_{\text{TV}}$)

$$\|Dv\|_1 \;=\; \sum_{i=1}^{n-1} |v_i - v_{i+1}| + |v_n| \;=\; \|v\|_{\text{TV}} + |v_n| \;\approx\; \|v\|_{\text{TV}} \; .$$

If v is a piecewise constant vector then we define

$$z = Dv \; , \tag{4.21}$$

where z is sparse and its non-zero entries correspond to the positions of the discontinuities in v. (Note that $z \neq x_s$).

The vector x is a combination of a piecewise constant vector *plus* a sparse component, and therefore posing the problem using TV regularization alone is unlikely to yield an accurate reconstruction. What we would like to do is break the blind reconstruction problem into two parts. If the subspace M can be determined, the method proceeds as in the previous section. The subspace M depends on knowledge of the vectors $u_1, u_2, \ldots, u_p$. If z from expression (4.21) is know accurately, then the vectors u_l for $l = 1, \ldots, p$ can be determined.

Because z is sparse and

$$Av = AD^{-1}z \approx b \; , \tag{4.22}$$

then z is approximated by solving the familiar convex optimization problem

$$\min_z \; \|AD^{-1}z - b\|_2^2 + \lambda \|z\|_1. \tag{4.23}$$

After z has been found, the subspace M is determined in the following way. The location of the spikes in the sparse vector z correspond to the location of the discontinuities in the piecewise constant vector v. From this information, the vectors $u_1, u_2, \ldots, u_p$ can be determined. Then $M = \text{span}\{Au_1, \ldots, Au_p\}$. (Note that as (4.22) is only an approximation, we do not accurately know the values $\alpha_1, \ldots, \alpha_p$ and in turn, we do not accurately know v.)

Once the subspace M is approximated, we return to the problem described in Section 4.3. That is, project out the subspace M (via (4.16)) and solve a second optimization problem (4.17) to determine x_s. Next the values $\alpha_1, \ldots, \alpha_p$ are found using (4.19), and x is reconstructed.

Remarks:

1. Usually when dealing with matrices, we do not actually compute a matrix inverse explicitly. The problems considered in this chapter need products of the form $AD^{-1}y_1$ and $D^{-T}A^Ty_2$ for various y_1 and y_2 and these products can be computed efficiently using backward and forward substitution with D and D^T respectively.

2. Noise is present in the observations b and problem (4.22) is an approximation. It is therefore likely that there will be small non-zero elements in the vector z so that there

may be more non-zeros than discontinuities in the piecewise constant vector v. We can still accurately form the projection matrix if there are more 'jumps' because the subspace M is still well approximated. However, we must be sure to locate all the true discontinuities in v, otherwise the method fails because we are projecting onto the wrong subspace.

4.5 THE GENERAL ALGORITHM

The following outlines a process for the reconstruction of signals that are a combination of a piecewise constant signal and a sparse component, using a limited set of data.

SIGNAL RECONSTRUCTION PROCESS

---

1. Solve the optimization problem

$$\min_z \ \|AD^{-1}z - b\|_2^2 + \lambda\|z\|_1$$

   to estimate the parameters z that define the subspace M.

2. Form the matrix $C = [Au_1, \ldots, Au_p]$ and perform an economy QR decomposition ($C = YR$) to determine orthonormal vectors that span the subspace M.

3. Projection out the subspace M.

4. Solve the transformed problem $\min_{x_s} \ \|PAx_s - Pb\|_2^2 + \lambda\|x_s\|_1$.

5. Determine the values $\alpha_1, \ldots, \alpha_p$ using $R\underline{\alpha} = Y^T(b - Ax_s)$.

6. Recover

$$v = \sum_{l=1}^{p} \alpha_l u_l, \quad \text{and} \quad x = x_s + v.$$

In Chapter 3 a number of algorithms were proposed which aim to solve the l_1-regularized least squares problems which feature heavily in this chapter. In the numerical results presented in Section 4.6, the BCGP algorithm is used to locate the discontinuities in the piecewise constant signal and to recover the sparse part of the signal.

4.6 NUMERICAL EXPERIMENTS

This section presents numerical examples to illustrate the practical performance of the reconstruction process outlined in this chapter. All experiments were performed on an Intel Xeon 3.2GHz processor with 4GB of memory, under Linux.

4.6.1 THE UNKNOWN SUBSPACE PROBLEM

This example considers a signal composed of an underlying piecewise constant signal with unknown discontinuities, and a sparse signal. A sensing matrix A of size 200×1000 was constructed and a vector $b \in R^{200}$ was observed. The observation vector was measured in the presence of Gaussian noise. The sparse signal had 10 spikes while the piecewise constant signal had 10 randomly located discontinuities of randomly generated heights. The results of the reconstruction procedure are shown in Figure 4.5.

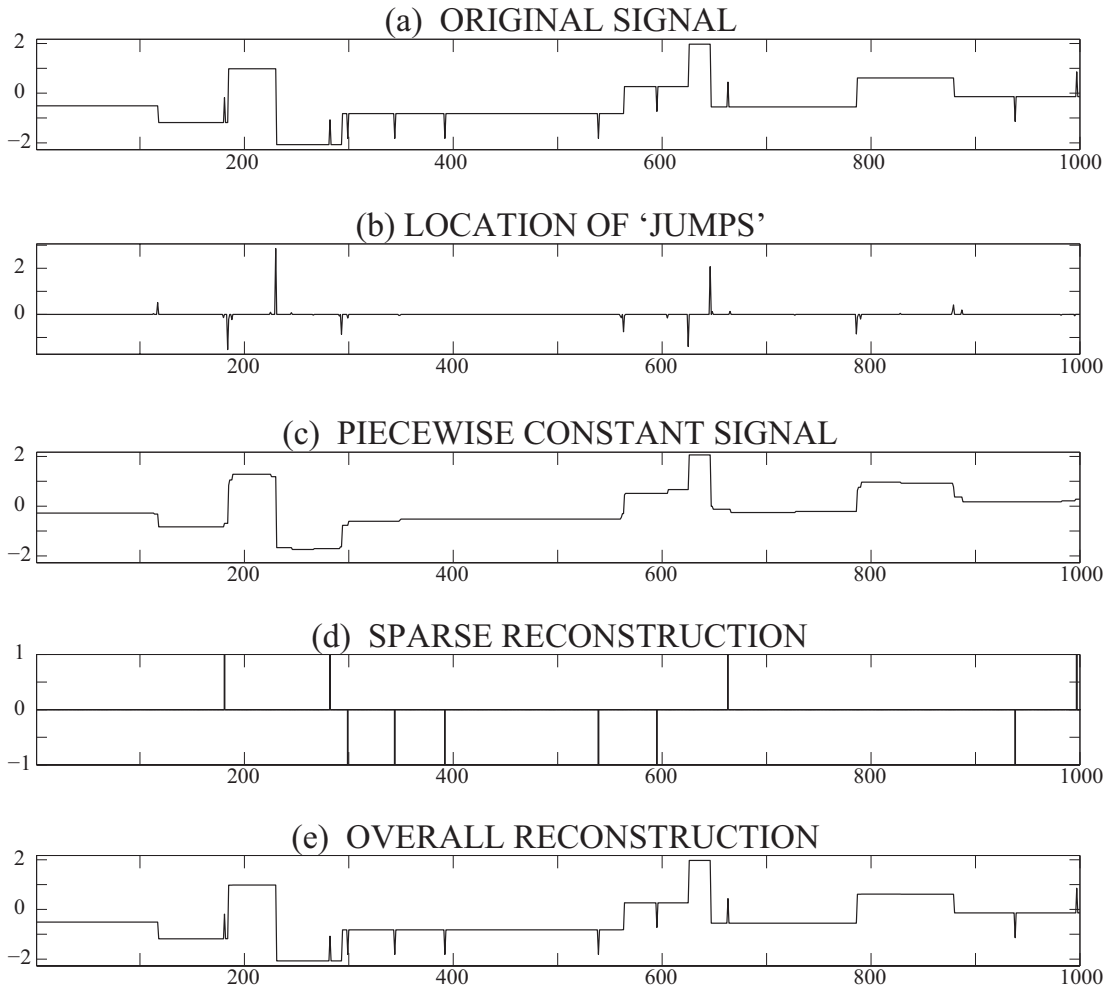


Figure 4.5: Reconstruction of a signal when the subspace M is unknown. (a) The original signal. (b) The sparse vector z that approximates the discontinuities/'jumps' in v. (c) The approximation using $v = D^{-1}z$. (d) The sparse vector x_s. (e) The overall reconstruction.

The reconstruction is excellent. The location and heights of the jumps in the piecewise constant signal have been accurately determined. Even the smaller jumps have been reconstructed accurately. For this example, the MSE was 1.8148×10^{-5}, which confirms the accuracy of the reconstruction. (Note that in Figure 4.5, the reconstruction in (c) is a result of the *approximation* $v = D^{-1}z$, while in (e), $\underline{\alpha}$ has been accurately recovered via $R\underline{\alpha} = Y^T(b - Ax_s)$.)

4.6.2 AN EXAMPLE FROM ASTRONOMY

Problems where a signal is a combination of a sparse part plus a piecewise constant part arise in astronomy. For example, a signal arriving from space may be corrupted by bursts of cosmic radiation. We can think of the piecewise constant part of the signal as the part that we wish to recover, and the sparse part of the signal as noise. As this is another example of unsupervised signal reconstruction we only know the sensing matrix $A \in R^{10^3 \times 10^4}$ and the observation vector $b \in R^{1000}$, which is measured in the presence of noise. Figure 4.6 illustrates this type of example.

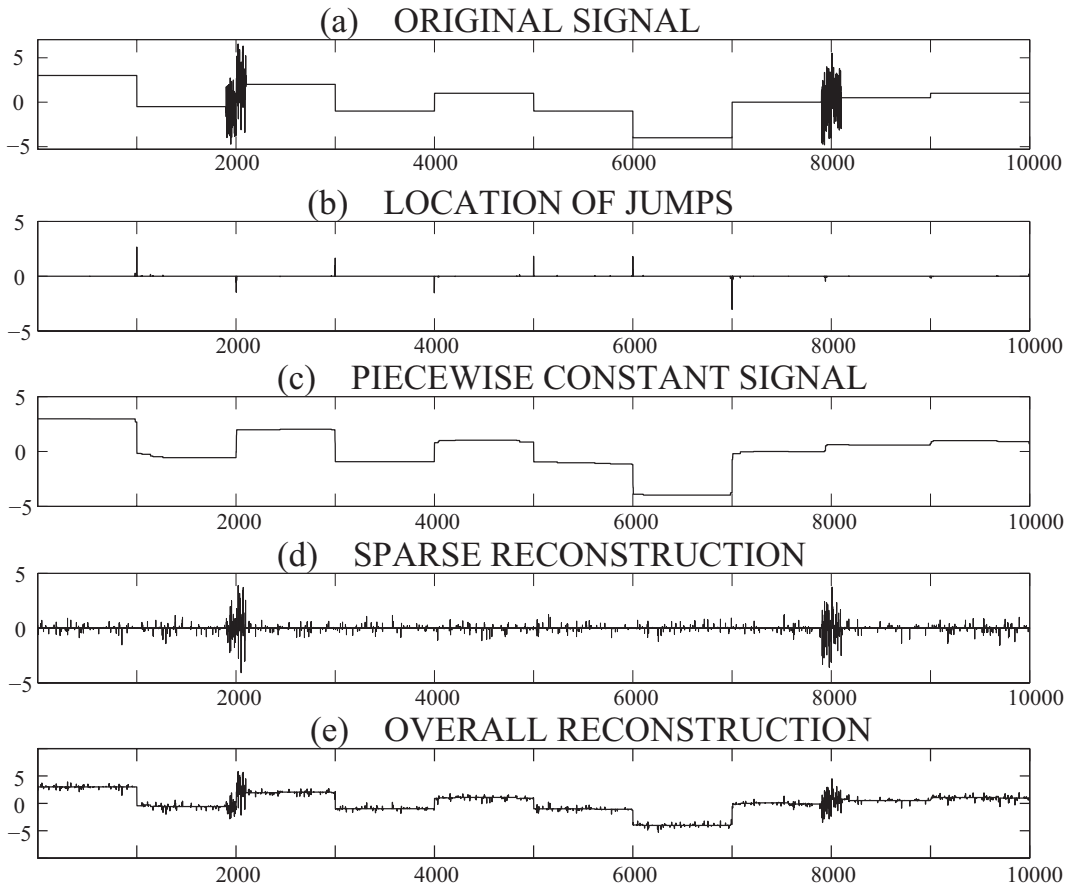


Figure 4.6: (a) The original signal; (b) the vector describing the location of the discontinuities in the piecewise constant signal; (c) the reconstructed piecewise constant signal; (d) the sparse reconstruction; (e) the final signal reconstruction.

The algorithm of Section 4.5 does an excellent job reconstructing the signal. This is a difficult example because the 'sparse' part of the signal is not sparse enough to be accurately reconstructed from so few measurements—because of the noise the signal actually has 410 spikes and as we have only $m = 1000$ observations, it violates the compressed sensing threshold of Theorem 3.1.1. Also, the bursts of radiation mask vital parts of the signal—the location of jumps. Despite this, the piecewise constant part of the signal is accurately reconstructed and although the sparse signal is very noisy, the location of the bursts are picked up accurately.

4.6.3 IMAGE RECONSTRUCTION

This example considers the reconstruction of the Shepp-Logan phantom. The phantom size is 50×50 pixels. The columns of the image are concatenated to represent the image as a column vector x of length 2500. The vector has 1204 non-zero elements so this is not a sparse image. However, Dx (where D is the matrix defined in (4.20)) is sparse with only 189 non-zero elements.

Recall Theorem 3.1.1 where $s = 189$ and $n = 2500$. Then A is the sensing matrix with $m = 1479$ orthogonal rows of length 2500, and an observation vector b of length m was measured in the presence of noise.

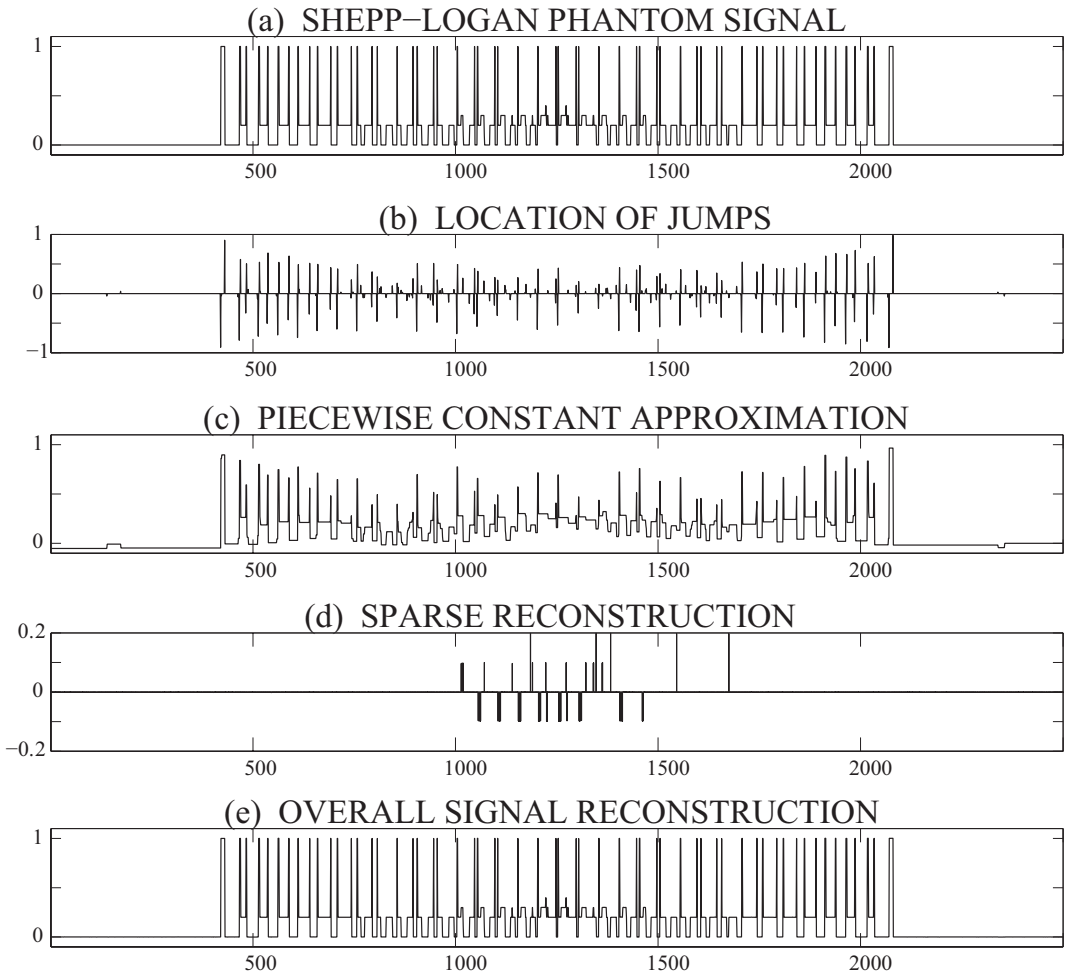


Figure 4.7: From top to bottom: (a) The signal representation of the Shepp-Logan Phantom; (b) The location of discontinuities in the piecewise constant signal; (c) The piecewise constant signal $v = D^{-1}z$; (d) The sparse reconstruction; (e) The final signal reconstruction.

Figure 4.7 shows the signal evolution at various stages of the reconstruction process. Plot (b) shows the location of discontinuities in the piecewise constant signal – that is, the vector z from (4.23). The piecewise constant reconstruction (c) is not exact, although this signal

has found the areas where there is a large contrast between neighbouring regions. This is the vector $v = D^{-1}z$. Next is the reconstructed sparse signal x_s, which can be thought of as an error correction step. It searches for finer detail that the first part of the algorithm may have missed. We can see that there need not be a large difference in neighbouring regions for this signal x_s to be accurately determined. The final plot (e) is the overall signal reconstruction. This is an excellent reconstruction. All fine detail is present. The MSE for this reconstruction is 1.37×10^{-8} which confirms very high accuracy of the final signal.

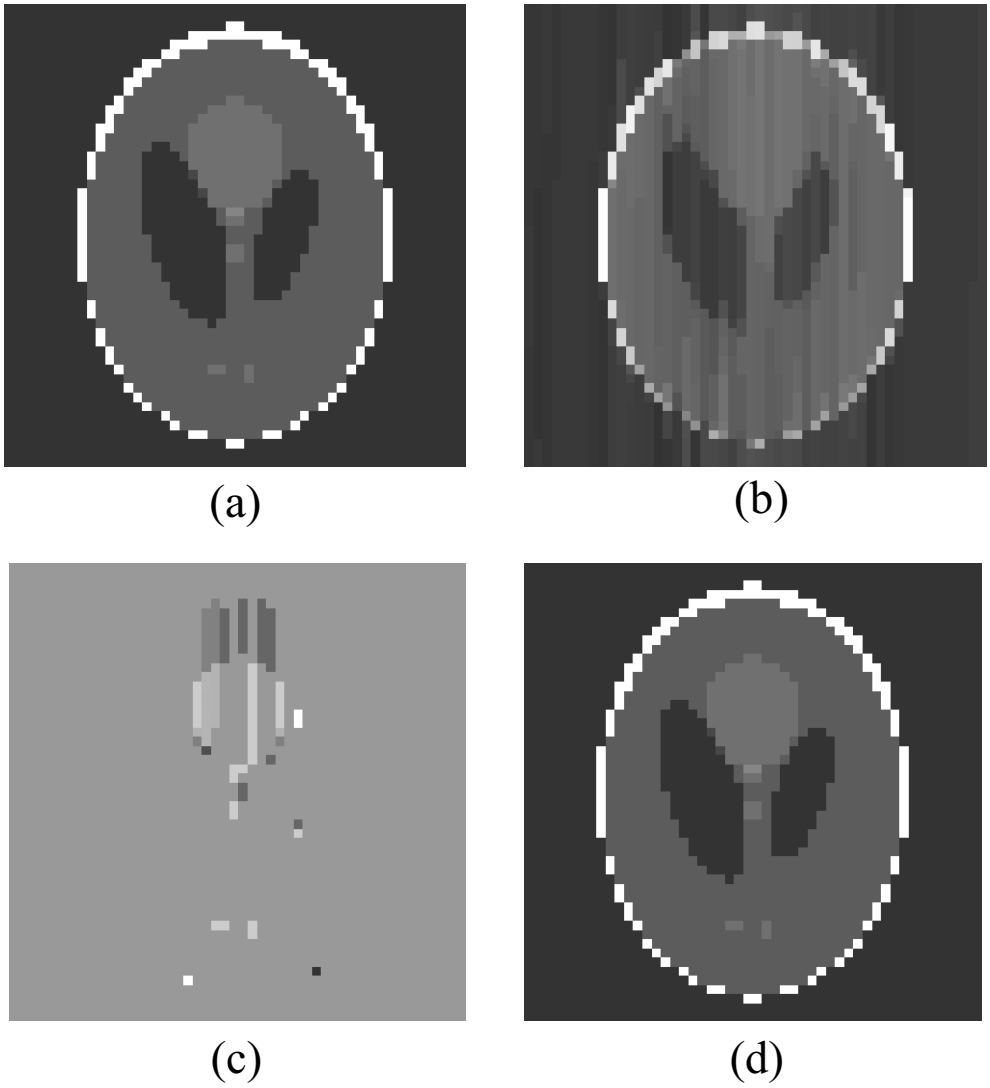


Figure 4.8: (a) The Shepp-Logan Phantom; (b) The Piecewise constant reconstruction, (corresponding to Figure 4.7(c)) with MSE = 0.0074619; (c) The Sparse reconstruction corresponding Figure 4.7(d)) with MSE = 0.060604; (d) The final signal reconstruction with MSE = 1.3701×10^{-8}.

Figure 4.8 shows the Shepp-Logan Phantom reconstruction. (The reconstructed length 2500 signal has been 'reshaped' into a 50×50 pixel image.) The reconstructed phantom captures all of the information of the original. Figure 4.8(b) is very stripey but provides information about the shape of the object and the neighbouring areas with high contrast. Figure 4.8(c)

provides the finer details of the object. During reconstruction of the sparse part of the signal, the subspace in which the piecewise constant signal lies is known, which means that the sparse signal can pick up areas that do not vary significantly from their neighbours. The background in this image is a flat grey, which tells us that the signal has already been accurately found in that region. This means we can be confident that the algorithm is working correctly. The final Phantom reconstruction is excellent. There is virtually no difference between the reconstruction and the original and this is supported by a very low MSE value.

4.6.4 TIME TRIAL

The average cputime (in seconds) over 10 runs for each of the examples in Section 4.6 are shown in Table 4.1. The total time has been split into 3 parts: the time taken to compute z, the time taken to compute the matrix C and its YR decomposition, and the time taken to compute x_s.

Table 4.1: Average CPU times (in seconds) over 10 runs on each of the examples described in Section 4.6.

	z	YR	x_s	Total
General	66.20	0.06	4.90	71.16
Phantom	87.51	0.07	5.02	92.60
Astronomy	97.58	0.11	10.82	108.51

The first part of the reconstruction process takes up the bulk of total computation time. This is to be expected as there are 4 matrix-vector products per iteration in the compressed sensing algorithm for step (4.23) of the overall algorithm. It is extremely fast to compute the matrix C and its YR decomposition. The final stage of the process is also fast although again there are again 4 matrix-vector products per iteration for the solution of (4.17) compared with only 2 matrix-vector products for the standard compressed sensing optimization problem. Nonetheless, these problems can still be solved in couple of minutes which seems reasonable. Future research will focus on speeding up the first part of the reconstruction process — for example, using the SALSA [39] or cSALSA [1] algorithms. These are basis pursuit type algorithms which have been shown to perform more quickly than l_1-ls.

4.7 CONCLUSION

In this chapter we considered reconstructing signals with specific structures from small amounts of data. We specifically considered signals that were a combination of a piecewise constant signal and a sparse component. The reconstruction process was posed as an l_1-regularized least squares problems in $2n$ unknowns. This can be separated into two optimization problems of the same type, for which there are efficient and accurate algorithms. We introduced a projection step that occurs between the two optimization problems, and this allowed us to solve for the vectors x_s and v independently.

The numerical results presented in this Chapter and in [13] were promising and showed that this process reconstructed signals accurately and efficiently.

4.7.1 FUTURE RESEARCH

The numerical experiments shown in this chapter were signal reconstruction problems. The final experiment on the Shepp-Logan phantom was an image reconstruction (2D) example that we solved using a 1D approach. Clearly, one area of future research is to extend to a 2D approach for images. The total variation norm can be extended to the 2D case using the definition

$$\|x\|_{\mathrm{TV}} = \sum_{i=1}^{n-1}\sum_{j=1}^{n-1}\sqrt{(x_{i+1,j}-x_{i,j})^2+(x_{i,j+1}-x_{i,j})^2}\,. \tag{4.24}$$

The process described in this chapter could be adapted using this 2D definition.

Another area of future research is known as morphological component analysis (MCA) [10, 37]. For MCA, a signal that is known to be degraded (for example many of the pixels are missing or there is a large amount of noise) is restored by decomposing it into the sum of several underlying signals that are each sparse relative to some known bases. The sum of these underlying signals gives the restored signal. This approach is slightly different from ours as it depends on knowledge of the original signal while our approach attempted to reconstruct the signal from data measurement. However, the two ideas seem closely related and using ideas from MCA our approach could be extended to signals made up of more than two components.

CHAPTER 5

CONCLUSION

Signals and images are all around us, and as technology advances, the need for innovative ways to approach the associated processing and reconstruction problems becomes ever more apparent. The solution of these problems requires expertise from many different disciplines, and mathematics has a central role to play.

This thesis investigated the problem of fast signal and image reconstruction from given data. Chapter 1 explains that the data collection process carries the highest cost, so reducing the amount of data necessary for an accurate image to be reconstructed results in a significant reduction in the overall processing time. Therefore, the idea of reconstruction from limited data is the practical problem that forms the foundations for the work in this thesis.

The following key questions were investigated:

1. How do we choose which data to observe, to ensure that a high quality signal or image is reproducible?

2. How do we accurately reconstruct a signal, given a sub-sampled data set?

Question 1 is the focus of Chapter 2. Mathematically this is a subset selection problem and is combinatorial in nature. An existing criterion based upon the trace of a matrix provides a measure of the error in the reconstruction and subsequently can be used to define the 'goodness' of a selected subset. While this selection criterion removes the combinatorial aspect of the selection process, the associated algorithms are suboptimal. By swapping the arithmetic mean of the eigenvalues (the trace) for the geometric mean (or determinant) we have derived a much simpler, new selection criterion. Intuitively this new criterion would be expected to perform similarly and provide comparable results to those found using the trace criterion. Because the new criterion is simpler to compute, the algorithms that use the determinant criterion are automatically faster than those based upon the trace. Somewhat surprisingly, the selected determinant subsets are sometimes better than those found using the trace criterion.

An inherent flaw of the sequential algorithms (regardless of the criterion used) occurs if the rows of the observation matrix have equal length. In this case the initial row is selected via the sorting algorithm of the programming language being used. In this work all algorithms were coded in Matlab and the built-in sorting function selects the first occurrence. This situation

has been successfully addressed by devising hybrid selection algorithms that implement a row exchange procedure. Two row exchange criteria are derived in Chapter 2, one based upon minimizing the trace of the submatrix, the other based upon maximizing the determinant of the submatrix. As expected, the formulation based upon the determinant is simpler and therefore faster to compute. Once a subset has been selected using any one of the sequential algorithms, the hybrid selection algorithms further improve the quality of the subset, which in turn leads to a better quality signal or image. An exchange only takes place if a reduction of the trace, or increase in the determinant, is possible. The end result is therefore a locally optimal row subset. This is achievable in a fraction of the time required for an exhaustive search.

The second question is considered in Chapter 3 and the answer seems to lie in the relatively new area of compressed sensing. This field remedies the wasteful process based on the Nyquist condition that is usually used in signal processing, that of sampling an entire signal, compressing it, and subsequently throwing away most of the information. It asks the question, "why do we need that information in the first place if we are just going to throw most of it away?" The mathematical background explains that if we have a signal that is sparse in one basis, and we sample it in an *incoherent* basis, then the number of observations needed to accurately reconstruct the signal depends on the number of non-zeros and the log of the length of the signal. If the signal is sufficiently sparse, then the number of observations required is significantly less than the Nyquist rate suggests.

The reason that compressed sensing is so popular is that there are extremely efficient optimisation algorithms for putting this mathematical theory into practice. In chapter 3 we develop an algorithm for solving an l_1-regularized least squares problem arising in compressed sensing. The algorithm uses the Barzilai-Borwein step lengths and we show that if the sensing matrix has orthonormal rows, the step length is constant. We also provide upper bounds on the solution vector and allowable step length, and can subsequently reformulate the problem as a box-constrained quadratic programme. The algorithm is called the Box Constrained Gradient Projection (BCGP) algorithm. The algorithm uses an unusual line search to guarantee convergence of the algorithm. Another particularly useful feature of this algorithm is that it uses only two matrix-vector products per iteration, which results in a dramatic saving of computational costs and minimal storage requirements, particularly for large matrices. The numerical results presented at the end of the chapter show that the BCGP algorithm is fast, accurate, robust and competitive with currently used algorithms.

In Chapter 4 the ideas from compressed sensing are extended to signals that are not necessarily sparse, but have some specific structure that can be exploited. This allows these more complicated signals to be reconstructed from a limited amount of data. We specifically consider the reconstruction of signals that could be decomposed as the sum of a piecewise constant signal and a sparse component. The reconstruction process is posed as an l_1-regularized least squares optimization problem in more than n unknowns. The piecewise constant signal spans a subspace M. If the subspace is known, then forming a matrix P, which projects onto the orthogonal complement, reduces the number of unknowns to n, and the optimization problem recovers the sparse component. Subsequently, the remaining unknowns can be

recovered in a straightforward manner.

If the subspace M is unknown, the reconstruction problem is more difficult. An additional optimization problem is solved to approximate the parameters required to determine the subspace. Once M has been approximated, the procedure continues as for the simpler problem. The optimization problems were solved using the BCGP algorithm described in Chapter 2 and the numerical results seem very promising.

Signal and image processing is a large and ever-expanding field of research. The new and exciting techniques in this area hold real promise for the advancement of society. It is hoped that, in time, these techniques can be utilised in practical, real-world applications, particularly in the area of medical imaging, because the potential benefits are significant.

Bibliography

[1] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo. A fast algorithm for the constrained formulation of compressive image reconstruction and other linear inverse problems. *IEEE International Conference on Acoustics Speech and Signal Processing*, pages 4034–4037, 2010. DOI: 10.1109/ICASSP.2010.5495758.

[2] M. Bachmayr and M. Burger. Iterative total variation schemes for nonlinear inverse problems. *Inverse Problems*, 25, 2009.

[3] J. Barzilai and J. M. Borwein. Two-point step size gradient mathods. *IMA Journal of Numerical Analysis*, 8:141–148, 1988.

[4] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal of Imaging Sciences*, 2(1):183–202, 2009.

[5] D. S. Bernstein. *Matrix Mathematics*. Princeton University Press, Princeton, N.J., 2005.

[6] V. Subbuah Bharathi and L. Ganesan. Orthogonal moments based texture analysis of CT liver images. *Pattern Recognition Letters*, 29(13):1868–1872, October 2008.

[7] N. D. Blakeley. *Sampling Strategies and Reconstruction Techniques for Magnetic Resonance Imaging*. PhD thesis, Department of Electrical and Computer Engineering, University of Canterbury, 2003.

[8] N.D. Blakeley, P.J. Bones, R.P.Millane, and P.F. Renaud. Efficient frequency-domain sample selection for recovering limited-support images. *Journal of the Optical Society of America*, 20(1):67–77, January 2003.

[9] J. Bobin and E. J. Candès. A fast and accurate first-order algorithm for compressed sensing. *IEEE International Conference on Image Processing*, pages 1457–1460, 2009.

[10] J. Bobin, J. Starck, J. M. Fadili, Y. Moudden, and D. L. Donoho. Morphological component analysis: An adaptive thresholding strategy. *IEEE Transactions on Image Processing*, 16(11):2675–2681, 2007.

[11] J. Bobin, J.-L. Starck, and R. Ottensamer. Compressed sensing in astronomy. *IEEE Journal of Selected Topics in Signal Processing*, 2(5):718–726, October 2008. DOI: 10.1109/JSTSP.2008.2005337.

[12] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[13] R. Broughton, I. Coope, P. Renaud, and R. Tappenden. Reconstruction of signals with non-sparse components. *ANZIAM Journal, Computational Techniques and Applications Conference Proceedings*, 2011.

[14] R. L. Broughton, I. D. Coope, P. F. Renaud, and R. E. H. Tappenden. Determinant and exchange algorithms for observation subset selection. *IEEE Transactions on Image Processing*, 19(9):2437–2443, September 2010.

[15] R. L. Broughton, I. D. Coope, P. F. Renaud, and R. E. H. Tappenden. A box constrained gradient projection algorithm for compressed sensing. *Signal Processing*, 91(8):1985–1992, August 2011. DOI: 10.1016/j.sigpro.2011.03.003.

[16] A. Butler, P. Bones, and M. Hurrell. Prototype system for enhancement of frontal chest radiographs using eigenimage processing. *Journal of Medical Imaging and Radiation Oncology*, 52:244–253, 2008.

[17] E. Candès and J. Romberg. l_1-magic: Recovery of sparse signals via convex programming. Technical report, California Institute of Technology, October 2005.

[18] E. Candès and J. Romberg. Sparsity and incoherence in compressive sampling. *Inverse Problems*, 23(3):969–985, 2007.

[19] E. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, August 2006.

[20] E. J. Candès and T. Tao. Near-optimal signal recovery from random projections and universal encoding strategies. *IEEE Transactions on Information Theory*, 52(12):5406–5425, December 2006. DOI: 10.1109/TIT.2006.885507.

[21] E. J. Candès and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

[22] A. S. Carasso. False characteristic functions and other pathologies in variational blind deconvolution. A method of recovery. *SIAM Journal of Applied Mathematics*, 70(4):1097–1119, 2009.

[23] R. Chartrand. Exact reconstruction of sparse signals via nonconvex minimization. *IEEE Signal Processing Letters*, 14:707–710, 2007.

[24] R. Chartrand. Nonconvex compressive sensing and reconstruction of gradient-sparse images: Random vs. tomographic Fourier sampling. *IEEE Conference on Image Processing*, pages 2624–2627, 12–15 October 2008.

[25] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal of Scientific Computing*, 20(1):33–61, 1998.

[26] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

[27] K. Choi, J. Wand, L. Zhu, T. T. Suh, S. Boyd, and L. Xing. Compressed sensing based cone-beam computed tomography reconstruction with a first-order method. *Medical Physics*, 37:5113–5125, 2010.

[28] J. Claerbout and F. Muir. Robust modeling of erratic data. *Geophysics*, 38:826–844, 1973.

[29] R. Coifman, F. Geshwind, and Y. Meyer. Noiselets. *Applied Comutational Harmonic Analysis*, 10:27–44, 2001.

[30] Y. H. Dai and R. Fletcher. Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. *Numerische Mathematik*, 100:21–47, 2005.

[31] Y. H. Dai, W. W. Hager, K. Schittkowski, and H. Zhang. The cyclic Barzilai-Borwein method for unconstrained optimization. *IMA Journal of Numerical Analysis*, 26:604–627, March 2006.

[32] Y. H. Dai and L. Z. Liao. *R*-linear convergence of the Barzilai and Borwein gradient method. *IMA Journal of Numerical Analysis*, 22(1):1–10, 2002.

[33] J. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Mathematics of Computation*, 30:772–795, 1976.

[34] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[35] D. Donoho, Y. Tsaig, I. Drori, and J. Starck. Sparse solutions of underdetermined linear equations by stagewise orthogonal matching pursuit. Technical report, Stanford University, 2006.

[36] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006. DOI: 10.1109/TIT.2006.871582.

[37] M. Elad, J.-L. Starck, P. Querre, and D. L. Donoho. Simultaneous cartoon and texture image inpainting using morphological component analysis. *Appl. Comput. Harmon. Anal.*, 19:340–358, 2005.

[38] C. L. Epstein. *Introduction to the Mathematics of Medical Imaging*. SIAM, Philadelphia, USA, second edition, 2008.

[39] M. A. T. Figueiredo, J. M. Bioucas-Dias, and M. V. Afonso. Fast frame-based image deconvolution using variable splitting and constrained optimization. *IEEE Workshop on Statistical Signal Processing*, pages 109–112, 2009.

[40] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, December 2007. DOI: 10.1109/JSTSP.2007.910281.

[41] M. Firsching, T. Michel, and G. Anton. First measurements of material reconstruction in x-ray imaging with the medipix2 detector. *IEEE Nuclear Science Symposium Conference Record*, 4:2736–2740, 2007.

[42] R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, 2 edition, March 1991.

[43] R. Fletcher. On the Barzilai-Borwein method. *Optimization and Control with Applications, Springer-Verlag*, pages 235–256, 2005. Springer series in Applied Optimization 96.

[44] A. Friedlander, J. Martínez, and M. Raydan. A new method for large-scale box constrained convex quadratic minimization problems. *Optimization Methods and Software*, 5:54–74, 1995.

[45] Y. Gao and S. J. Reeves. Efficient computation for sequential forward observation selection in image reconstruction. *IEEE Proceedings of the International Conference on Image Processing*, 3:380–384, October 1998.

[46] Y. Gao and S. J. Reeves. Optimal k-space sampling in MRSI for images with a limited region of support. *IEEE Transactions on Medical Imaging*, 19(12):1168–1178, December 2000.

[47] Y. Gao and S. J. Reeves. Sequential forward sample selection in array-based image formation. *IEEE Conference on Acoustics, Speech and Signal Processing*, 3:1913–1916, May 2001.

[48] Y. Gao, S. Strakowski, S. J. Reeves, H. Hetherington, W. Chu, and J. Lee. Fast spectroscopic imaging using online optimal sparse k-space acquisition and projections onto convex sets reconstruction. *Magnetic Resonance in Imaging*, 55(6):1265–1271, 2006.

[49] R. Garg and R. Khandekar. Gradient descent with sparsification: An iterative algorithm for sparse recovery with restricted isometry property. *Proceedings of the 26th International Conference on Machine Learning*, pages 337–344, June 2009.

[50] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[51] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Pearson Prentice Hall, USA, 3rd edition, 2008.

[52] R. C. Gonzalez, R. E. Woods, and S. L. Eddins. *Digital Image Processing Using* MATLAB. Pearson Prentice Hall, 2004.

[53] Rice University DSP Group. Compressed sensing webpage. `http://dsp.rice.edu/cs`, 2011.

[54] W. Guo and W. Yin. EdgeCS: Edge guided compressive sensing reconstruction. Technical Report TR10-02, Rice University, 2010.

[55] E. T. Hale, W. Yin, and Y. Zhang. Fixed-point continuation applied to compressed sensing: Implementation and numerical experiments. *Journal of Computational Mathematics*, 28(2):170–194, 2010.

[56] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. SIAM, 1998.

[57] J. Haupt and R. Nowak. Signal reconstruction from noisy random projections. *IEEE Transactions on Information Theory*, 52(9):4036–4048, September 2006.

[58] S. Karris. *Signals and Systems*. Orchard Publications, 2nd edition, 2003.

[59] S. J. Kim, K. Koh, M. Lustig, and S. Boyd. An efficient method for compressed sensing. *IEEE International Conference on Image Processing*, pages 117–120, 16 September – 19 October 2007.

[60] S. J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale l_1-regularized least squares. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):606–617, December 2007.

[61] S. Leng, J. Tang, J. Zambelli, B. Nett, R. Tolakanhalli, and G-.H. Chen. High temporal resolution and streak-free four-dimensional cone-beam computed tomography. *Physics in Medicine and Biology*, 53:5653–5673, 2008.

[62] S. Levy and P. K. Fullagar. Reconstruction of a sparse spike train from a portion of its spectrum and application to high-resolution deconvolution. *Geophysics*, 46(9):1235–1243, 1981.

[63] G. Li, J. Yang, C. Ye, and D. Geng. Degree prediction of malignancy in brain glioma using support vector machines. *Computers in Biology and Medicine*, 36:313–325, 2006.

[64] I. Loris, M. Bertero, C. De Mol, R. Zanella, and L. Zanni. Accelerating gradient projection methods for l_1-constrained signal recovery by steplength selection rules. *Applied and Computational Harmonic Analysis*, 27(2):247–254, September 2009.

[65] F. Luengo and M. Raydan. Gradient method with dynamical retards for large-scale optimization problems. *Electronic Transactions on Numerical Analysis*, 16:186–193, 2003.

[66] F. Martinez-Alvarez, A. Troncoso, J. C. Riquelme, and J. S. Aquilar-Ruiz. Detection of microcalcifications in mammographies based on linear pixel prediction and support-vector machines. In *IEEE Symposium on Computer-Based Medical Systems*, pages 141–146, June 2007.

[67] D. W. McRobbie, E. A. Moore, M. J. Graves, and M. R. Prince. *MRI: From Picture to Proton*. Cambridge University Press, 2 edition, 2007.

[68] M. Mishali and Y. C. Eldar. Blind multi-band signal reconstruction: Compressed sensing for analog signals. *IEEE Transactions on Signal Processing*, 57(3):993–1009, March 2009.

[69] K. Najarian and R. Splinter. *Biomedical signal and image processing.* CRC Press, 2006.

[70] D. Needell and J. A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

[71] M. Nikolova, M. K. Ng, S. Zhang, and W. Ching. Efficient reconstruction of piecewise constant images using nonsmooth nonconvex minimization. *SIAM Journal of Imaging Sciences*, 1(1):2–25, 2008.

[72] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer Series in Operations Research and Financial Engineering. Springer, 2nd edition, 2006.

[73] M. Parmar and S. J. Reeves. A perceptually based design methodology for color filter arrays. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 3:473–476, May 2004.

[74] K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook.* Online Resource, February 16 2008.

[75] M. Raydan. On the Barzilai and Borwein choice of steplength for the gradient method. *IMA Journal of Numerical Analysis*, 13(3):321–326, 1993.

[76] M. Raydan. The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM Journal of Optimization*, 7(1):26–33, February 1997.

[77] M. Raydan and B. F. Svaiter. Relaxed steepest descent and Cauchy-Barzilai-Borwein method. *Computational Optimization and Applications*, 21(2):155–167, 2002.

[78] S. J. Reeves and L. P. Heck. Selection of observations in signal reconstruction. *Proceedings of the 1993 IEEE International Conference on Acoustics, Speech and Signal Processing*, 3:444–447, 1993.

[79] S. J. Reeves and L. P. Heck. Selection of observations in signal reconstruction. *IEEE Transactions on Signal Processing*, 43(3):788–791, March 1995.

[80] S. J. Reeves and Z. Zhe. New results on observation selection in signal reconstruction. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 3(1):1676–1679, May 1996.

[81] S. J. Reeves and Z. Zhe. Sequential algorithms for observation selection. *IEEE Transactions on Signal Processing*, 47(1):123–132, January 1999.

[82] J. Romberg. Imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2):14, March 2008.

[83] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[84] M. Sasikala and N. Kumaravel. Comparison of feature selection techniques for detection of malignant tumor in brain images. In *IEEE Annual Indicon Conference*, pages 212–215, Chennai, India, December 2005.

[85] J. L. Semmlow. *Biosignal and Biomedical Image Processing,* MATLAB *Based Applications*. Signal Processing and Communications Series. CRC Press, New York, USA, 2004.

[86] C. E. Shannon. Communication in the presence of noise. *Proceedings of the IEEE*, 86(2):10–21, February 1998. Reprint as a classic paper.

[87] B. Sharif and F. Kamalabadi. Optimal sensor array configuration in remote image formation. *IEEE Transactions on Image Processing*, 17(2):155–166, February 2008.

[88] E. Y. Sidky, R. Chartrand, and X. Pan. Image reconstruction from few views by nonconvex optimization. *IEEE Nuclear Science Symposium Conference Record*, pages 3526–3530, 2007.

[89] E. Y. Sidky and X. Pan. Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization. *Physics in Medicine and Biology*, 53(17):4777–4807, 2008.

[90] J. L. Starck and J. Bobin. Astronomical data analysis and sparsity: From wavelets to compressed sensing. *Proceedings of the IEEE*, 98(6):1021–1030, 2010. DOI: 10.1109/JPROC.2009.2025663.

[91] G. W. Stewart. *Matrix Algorithms, Volume I: Basic Decompositions*. SIAM, 1998.

[92] P. Suetens. *Fundamentals of Medical Imaging*. Cambridge University Press, 2002.

[93] E. Tadmor and J. Zou. Three novel edge detection methods for incomplete and noisy spectral data. *Journal of Fourier Analysis and Applications*, 14:744–763, 2008.

[94] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society B*, 58(1):267–288, 1996.

[95] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, December 2007.

[96] E. van den Berg and M. P. Friedlander. Probing the Pareto frontier for basis pursuit solutions. *SISC*, 31(2):890–912, 2008.

[97] C. Wang and S. Yang. Comparative evaluation of classifiers and feature selection methods for mass screening in digitized mammograms. IEEE Life Science Systems and Applications Workshop, July 2006.

[98] Y. Wang and S. Ma. Projected Barzilai-Borwein method for large-scale non-negative image restoration. *Inverse Problems in Science and Engineering*, 15(6):559–583, September 2007.

[99] Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences*, 1(3):248–272, 2008.

[100] D. Watkins. *Fundamentals of Matrix Computations*. Wiley-Interscience. John Wiley and Sons, 2nd edition, 2002.

[101] M. Weeks. *Digital Signal Processing Using* MATLAB *and Wavelets*. Electrical Engineering Series. Infinity Science Press, 2007.

[102] D. Weishaupt, V. Köchli, and B. Marincek. *How Does MRI Work?* Springer, second edition, 2006.

[103] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3373–3376, March 31 – April 4 2008.

[104] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, July 2009.

[105] B. Wu, R. P. Millane, R. Watts, and P. J. Bones. Improved 3D image plane parallel magnetic resonance imaging (pMRI) method. *Proceedings of Image and Vision Computing Conference, Hamilton, New Zealand*, pages 311–316, December 2007.

[106] B. Wu, R. P. Millane, R. Watts, and P. J. Bones. Improved matrix inversion in image plane parallel MRI. *Magnetic Resonance in Imaging*, 27:942–953, 2009.

[107] K. Wu, Y. Wang, Y. Pan, J. Zhang, B. Qian, and C. Chang. Classifying uterine myoma and adenomyosis based on ultrasound image fractal and texture features. *IEEE/NLM Life Science Systems and Applications Workshop*, pages 1–2, July 2006.

[108] W. Yin, S. Morgan, J. Yang, and Y. Zhang. Practical compressive sensing with Toeplitz and circulant matrices. Technical Report TR10-01, Rice University, 2010.

[109] H. Yu and G. Wang. Compressed sensing based interior tomography. *Physics in Medicine and Biology*, pages 2791–2805, 2009. DOI: 10.1088/0031-9155/54/9/014.

[110] M. Zibulevsky and M. Elad. L1-L2 optimization in signal and image processing. *IEEE Signal Processing Magazine*, 27(3):76–88, May 2010.