Big data: a compressed sensing approach

by

Charl Janse van Rensburg

Submitted in partial fulfillment of the requirements for the degree

Magister Scientiae

In the Department of Statistics
In the Faculty of Natural and Agricultural Sciences
University of Pretoria

January 2017

I, *Charl Janse van Rensburg,* declare that this mini-dissertation (100 credits), which I hereby submit for the degree Magister Scientiae in Mathematical Statistics at the University of Pretoria, is my own work and has not previously been submitted by me for a degree at this or any other tertiary institution.

Signature:

Date:

Summary

In recent times Big Data has been talked about in many areas, ranging from information technology, to government and healthcare, and to business. Big Data is changing the world we live in in many respects, especially as data of the individual becomes available in forms which it has not been previously, for example, data about the behaviour of indiviuals tracked via mobile phones. We discuss Big Data and whether it is having the said affect, or if it is only an unsubstantiated hype about something old coated under a new name. Convinced that Big Data is indeed a phenomenon of our day worthy of spending time and money on, we investigate whether Compressed Sensing (CS), a new and exciting tool in the signal processing field, can provide sensible solutions to Big Data problems. CS proposes a framework in which we simultaneously acquire and compress a signal of interest. However, for this to work, the way in which we acquire the signal needs to adhere to some uncertainty principles and the signal of interest need to be sparse in some basis representation. We argue that because Big Data many times exhibit sparsity and generally poses challenges to the storage capacity of different devices and systems, CS can be a useful tool in addressing challenges in the Big Data era and should be considered as a potential research area. This mini-dissertation provides an overview of CS and is by no means a full in-depth mathematical treatment of CS. It is written to provide the statistician with the necessary background and building blocks of CS, for use in the Big Data environment, and herein, CS is presented in a simple and clear manner for a statistician not familiar with the field. The literature review, however, provides all the texts required should the reader want the specific mathematical details. The document aims to thus link CS in the statistical and engineering fields.

Acknowledgements

I am thoroughly grateful for the opportunity I had to go through the process of writing up this mini-dissertation. I have learned a lot about research, about Big Data and Compressed Sensing. But I have also learned interesting things along the way. It has been a challenging journey, and it would truly not have been possible without the support of friends, family, my wife and my supervisor. My wife agreed to let me study full-time in order to obtain this degree. I am especially grateful for her sacrifice in it. She is my standard basis. Lastly, Inger Fabris-Rotelli has been a tremendous supervisor who is passionate about what she does. She has been an exceptional guide and is truly a great academic. The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF. I am grateful for the financial support by the South African Research Chairs Initiative (SARChI) and the NRF CSUR Grant 90315.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Kenneth Cukier said the following at a Ted Talk done in 2014 in Germany[1],

> "In the past, we used to look at small data and think about what it would mean to try to understand the world, and now we have a lot more of it, more than we ever could before. What we find is that when we have a large body of data, we can fundamentally do things that we couldn't do when we only had smaller amounts. Big Data[2] is important, and Big Data is new, and when you think about it, the only way this planet is going to deal with its global challenges — to feed people, supply them with medical care, supply them with energy, electricity, and to make sure they're not burnt to a crisp because of global warming — is because of the effective use of data."

Big data are two words that have been flying around in the last decade. We will begin by discussing Big Data and whether it is truly a phenomenon of our time, or only a spontaneous hype. Secondly, we will discuss compressed sensing, a new and exciting research field in signal processing, and whether it has a place when talking about Big Data.

The aim of this document is therefore as follows:

1. To explore Big Data and what it is.
2. To provide an overview on compressed sensing, which we will explore in Chapter 3.
3. To investigate the possibility of the application of compressed sensing in the Big Data environment and draw some conclusions on this matter.

[1] `https://www.ted.com/talks/kenneth_cukier_big_data_is_better_data?language=en`
[2] According to [96] "Big Data" and "Big Data Anaytics" should be capitalised when referring to the phenomenon itself.

Compressed sensing in a nutshell

Suppose we consider a specific signal $f : N \times 1$. Suppose also that all we get to observe of the signal are M linear measurements expressed as $y = \Phi f$, assuming no noise and where $\Phi : M \times N$ is some sensing matrix used to obtain the measurement vector y . We would like to recover the original signal f from these measurements. The crux comes in due to the fact that we are looking at the case $M \ll N$ implying that $y = \Phi f$ is an underdetermined system and therefore we won't be able to get a unique solution for f. However, if f is sparse then Candes et al. [30] show we can recover f exactly by exploiting the sparsity of f. The signal can be any one of various signals, provided that it is sparse. A sparse signal is one with only a few non-zero elements, implying most of the elements are zero. We will enter this fascinating world in-depth in Chapter 3. With this in mind, we first explore the world of Big Data.

1.1 The Big Data concept

In order to define Big Data and understand the concept we first need to answer the following question: Is Big Data really such a big deal? The following evidence suggests that it is.

Firstly, Big Data is a big deal since companies and governments and other decision-making bodies and individuals are making much of Big Data. And they are also investing large amounts of money in order to make better decisions. SAS®[3] is currently creating many products to enable businesses and corporates to analyse, interpret and visualise data which they have never looked at before. These days there is a lot of discussion around Big Data Analytics. In the academic world one can find a host of conferences and workshops on Big Data[4]. Corporates are investing time, energy and money in Big Data. This is suggested in the business report compiled by CapGemini and EMC2® called "Big & fast data: the rise of insight driven business"[5]. A survey was conducted with 1000 senior decision makers in companies from 9 different industries across 10 countries. This survey found that 56% of the enterprises will over the three years after the survey increase investment in Big Data. Of the respondents, 64% said that it is risky for them not to consider spending resources on Big Data as they might get left behind. In 2012 the US government gave 200 million dollars to fund research initiatives in Big Data[6]. Today data science is an emerging applied area combining statistics, computer skills, data manipulation skills and business skills with focus on utilising the Big Data at the disposal of companies. Companies are increasingly looking for people who have these data scientist skills.

Secondly, Big Data affects every person at an individual level. Kenneth Cukier, quoted earlier, was the data editor for The Economist[7]. He argues in a book co-authored with Mayer-Schonberg [131], that Big

[3] https://www.sas.com/en_us/home.html
[4] http://bigdata.ieee.org/conferences
[5] https://www.capgemini.com/thought-leadership/big-fast-data-the-rise-of-insight-driven-business
[6] http://www.wired.com/2012/10/big-data-is-transforming-healthcare/
[7] http://www.economist.com/

Data is transforming the way we live and think. Companies such as Telefonika[8] are using data to help businesses position their shops in places that would most likely attract consumption[9]. Mattersight® is a company that helps other companies to improve customer satisfaction by making a link between business and the personalities of the personnel of the company, as well as the customers. For example Mattersight implemented this within a call centre of a pharmaceutical company. According to Mattersight[10] the average talk time dropped by 13% in just weeks because of the optimisation implemented. Lastly, De Montjoye et al. [61] provide results from the study of mobility data of one and a half million people over a period of fifteen months. It is stated that the mobility of a person is very unique. This is of course logical. But it is even suggested that with the available data, a person could be uniquely identified using only four location points with 95% certainty. The points are spatial-temporal, meaning it has the dimensions of both location and time.

We therefore argue that, 'Yes, Big Data is a big deal'. And this is independent of the definition of Big Data. It is even independent of whether or not Big Data is something new and different that did not exist in previous decades. The reason is because it is relevant. It is having a dramatic impact on the world we live in. As much as the development of artificial intelligence was influential and relevant in the mid-1900's, so Big Data is today. We now proceed to investigate what Big Data is.

1.1.1 The roots of Big Data

To best understand what Big Data is, a good place to look at is when the idea was first introduced. The term 'Big Data' was coined by John Mashey, the Chief Scientist of Silicon Graphics®. He used it publicly for the first time [77] in his talk he presented on July 29, 1998. The talk was titled "Big data and the next wave of infrastress" and in it Mashey gave an overview of what Silicon Graphics® expected to see in the decade thereafter regarding the constraints the amounts of data generated would put on storage and processing capabilities. This foresight enabled Silicon Graphics® (today called Silicon Graphics International Corp.) to become world leaders in high-performance computing solutions and development. Part and parcel of this is the rise of Big Data. Mashey and his colleagues saw Big Data as large amounts of data presenting challenges to storage of that data, as well as the effective and efficient processing of that data. In terms of publications, the very first time the term 'Big Data' was used, was in [21] in 1999. Herein the authors discuss the challenge of visualising 3D data sets of gigabyte size in real time. Underneath the heading "Big data for scientific visualization", the following is stated:

> "Interactively browsing a data set is less difficult when the data resides in physical memory. But many data sets are larger than the physical memory of a particular computer, and some data sets are much larger than any computer's memory. We address this problem using techniques that reduce the amount of data that has to be handled by a visualization program.

[8] https://www.telefonica.com/en/
[9] http://dynamicinsights.telefonica.com/blog/488/smart-steps-2
[10] http://www.mattersight.com/resource/prescription-benefits-manager-pbr-case-study/

> Techniques for providing interactive visualization can rely on sparse traversal, on compression, or on both. Sparse traversal represents the idea that a visualization algorithm analyzing scientific data may require only a subset of that data for each frame. Compression is the standard idea that a very large data set may have a smaller alternative representation that can be managed more easily at interactive rates."

Therefore, it seems as though the size of the data set was the main challenge to overcome in order to attain the specific goals set, which in this case was making the data manageable and interpretable via 3D data visualisation. Big Data in this case was data that challenged the current ability of computers in the performance of some tasks with those data. Shortly thereafter the first article appeared with the term Big Data in its title [63] in which is stated the following:

> "Big data refers to the explosion in the *quantity* (and sometimes, quality) of *available and potentially relevant data*, largely the result of recent and *unprecedented advancements in data recording and storage technology*. In this new and exciting world, sample sizes are no longer fruitfully measured in 'number of measurements', but rather in, say, megabytes. Even data accruing at the rate of several gigabytes per day are not uncommon. Economics examples include microeconometric analyses of consumer choice, which have been transformed by the availability of huge and detailed datasets collected by checkout scanners, and financial econometric analyses of asset return dynamics, which have been similarly transformed by the availability of tick-by-tick data for thousands of assets."

Yet again in [63] the emphasis is on the size of the dataset, however, there are two fundamental properties of Big Data pointed out. The first is the potential of the data. The second is the progress in data acquisition. The data acquisition most likely refers to the effect of the growth in the number and variety of sensors that was made. According to [77] the first book that referred to Big Data was in 1998, namely [174]. In this book Weiss gives a similar definition of Big Data to what was described above in [21, 63]. The added challenge of how data warehouses [15] play a part in using data after it is stored is important.

We can therefore agree that the size of the data matters when talking about Big Data, but more importantly, is the word 'new', which we will expand on below.

1.1.2 Defining Big Data and Big Data Analytics

One could say that to define Big Data is in itself a Big Data problem. There are currently myriads and myriads of articles and video clips amongst other, talking about Big Data and what it is and how it is being applied. What makes it difficult to define is the *problem-specific* and *unscientific* nature of Big Data; problem-specific since different individuals and companies and other organisations use different types of Big Data to solve their specific problems in creative ways; and unscientific because many times

the solutions to problems are very robust, but not necessarily scientifically provable. What also makes Big Data difficult to define is the way it is referred to. Sometimes when people talk about Big Data, they might talk vaguely about Big Data without any substantiating evidence. Not necessarily because it is not there, but rather because of the fact that it is problem-specific.

Characteristically, Big Data has been described by Laney [119] at the start of the millenium as having all of three V's: Volume, Velocity and Variety. On their website, SAS®[11] considers two additional characteristics: Variablility and Complexity. Big Data of course typically come in large amounts, but it can also do it at high speed; think of sensors collecting large amounts of data in short intervals of time. And data can look very different; think of images vs. emails. Added to that, data can have spontaneous spikes, or seasonal trends. And lastly, Big Data can be complex. The complexity lies in different types of data being related to each other, for example a company's emails, travel schedules and staff information which are all related. However, it is difficult to combine the data and transform it into some form which is useful for analysis. However the simultaneous analysis of the different types of related data together can yield better insights, leading to better decision making. This is one of the specific attributes of Big Data which indicates that the challenges of the analysis of data today differs from statistical and data-related challenges in previous decades and centuries.

The volume aspect of Big Data is one of the most debatable aspects of Big Data. We argued above that the size of data in Big Data does count, thinking of how Big Data was defined at the pre-empting of the Big Data era. We will discuss the volume aspect using examples when we discuss where Big Data comes from below, but for now we want to make reference especially to the problem of the amount of data being created. In [97] Hillbert and Lopez estimated the ability of the world to store data, both in analog and digital formats. Their results are given as in [97] in Figure 1.1.1. In Figure 1.1.1 we can see a dramatic shift in the last two decades in the balance between the amount of analog data (data in physical forms, e.g. paper documents, music vinyls, film of cameras) compared to digital data (data in some electronic format, for e.g. images, text documents, emails). In Figure 1.1.2[12] and Figure 1.1.3[13] we see that we do not have enough storage capacity for all the data being created. In Figure 1.1.2 we have an older graph which seems to be quite accurate, comparing it with the infographic in Figure 1.1.3. It was estimated by the International Data Corporation (IDC) that in 2013 4.4 zettabytes[14] of data were created, whereas the combined storage capacity of installed raw capacity and shipped capacity was estimated to be about 3.5 zettabytes. Therefore, we will need to decide on how to deal with the growing zettabytes of information and our ability to store and process it. One way this problem is being addressed is called cloud computing [9]. According to [9] "Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services". With cloud computing it isn't necessary for the end-user to have physical memory for everything

[11] http://www.sas.com/en_us/insights/big-data/what-is-big-data.html

[12] The article containing the graph can be found at https://www.emc.com/collateral/analyst-reports/ar-the-economist-data-data-everywhere.pdf

[13] http://www.idc.com/prodserv/IDCdigitalHub/info-bigdata.jsp?title=Where+in+the+World+is+Storage&src=where_is_storage_infog.gif

[14] 1zettabyte (ZB) = 1000 exabytes (EB) = 1000000 petabytes (PB) = 10^9 terabytes (TB) = 10^{12} gigabytes (GB)

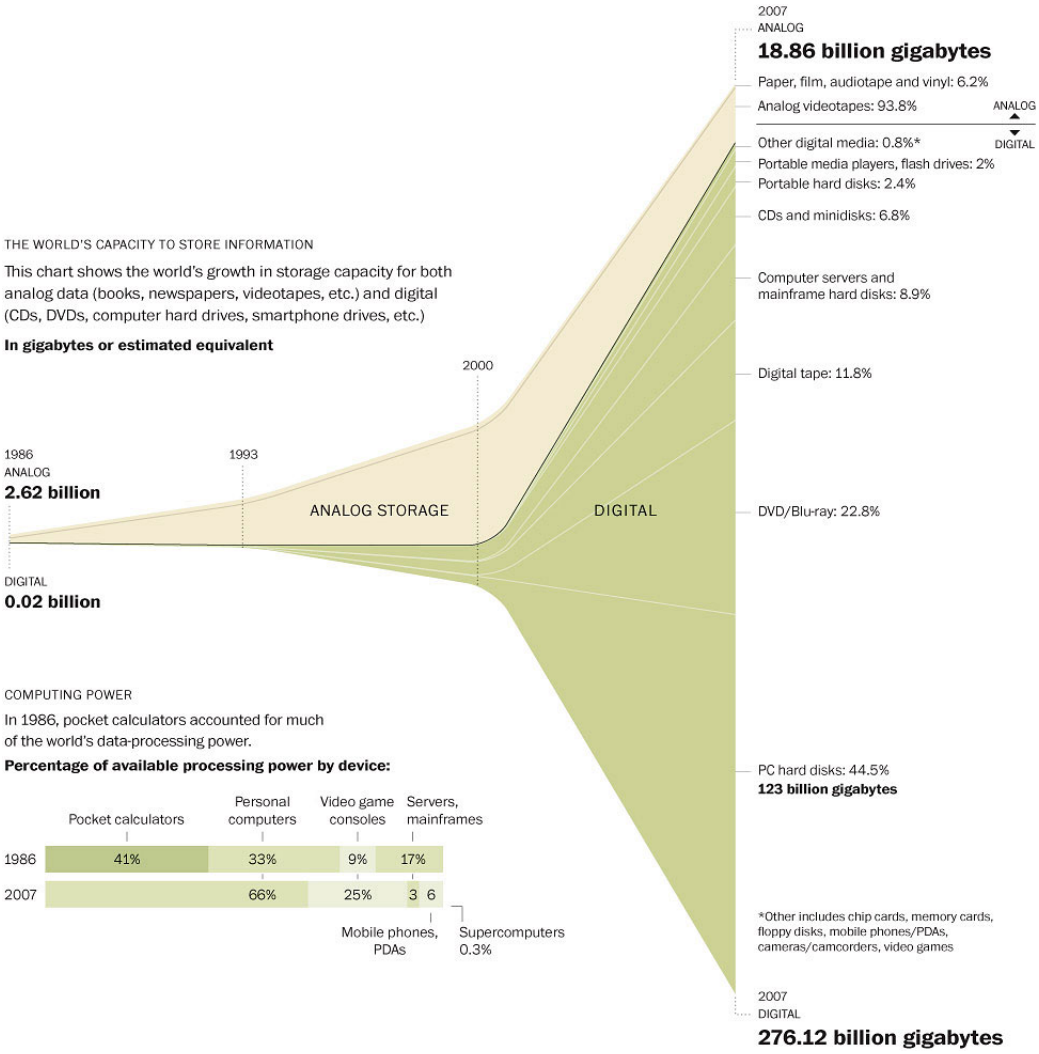


Figure 1.1.1: Comparison between digital and analog storage up until 2007.

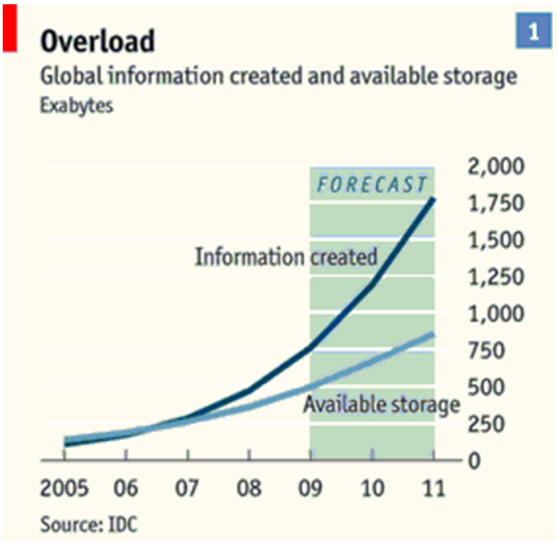


Figure 1.1.2: Information created vs available storage in 2009 according to IDC.

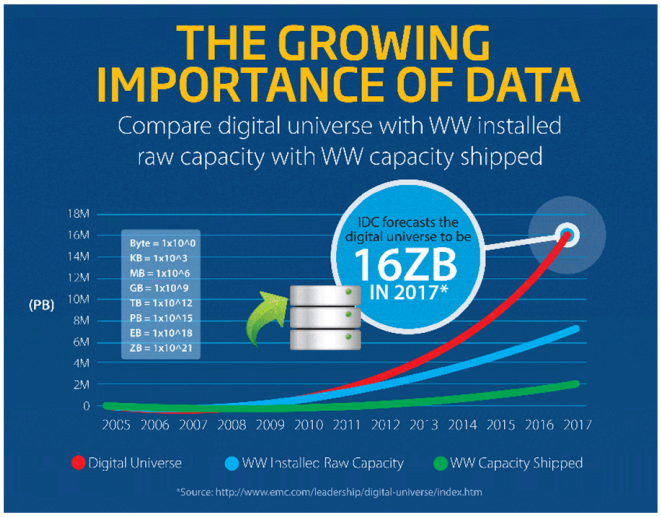


Figure 1.1.3: Digital universe and storage capacity according to IDC in 2013.

they want to do.

Hilbert describes Big Data slightly differently. In [96] Hilbert provides a study on 180 articles on Big Data and summarised his findings. He suggests that Big Data will typically have some or all of the following properties:

- Big Data is data generated from *digital footprints*: many of the data being created today, is data that people generate by simply going through their daily routines. We can think of Facebook and Twitter where people provide information of themselves. It might be their location, what they are doing at a certain time or what has happened in their lives recently. Google collects hourly information on your location, using your phone as one of the resources. As you are swiping your debit card at various places, you are leaving a trace of your expenditure patterns. It is data that is stored that is simply created by individuals carrying on with their daily routines.

- The *sample size* of Big Data is $n = N$: because Big Data come in large amounts, the data is many times collected on a whole population of interest, or at least enough people or elements that we could say that what is true for the 'sample' is likely to be true for the whole population. The interesting implication this can have is on how surveys will be used in the future. This property of Big Data does not make the use of sampling redundant. For instance, since so many people have cellphones we could say something about the information those cellphones contain. Especially smart phones are becoming affordable to almost everyone. However, if we want to determine the prevalence of HIV, we will need to do a survey. And sampling is essential in such cases.

- Big Data is many times *unstructured* which leads to *fusion of data sets*: this corresponds to SAS®'s[15] definition of variety and complexity. Various kinds of data do not come in the traditional form of observations in rows and variables in columns. Many times, it has a lot of missing values. For

[15] http://www.sas.com/en_us/insights/big-data/what-is-big-data.html

instance, some people post status updates on Facebook, while others post images, and others do both. This can make the process of data analysis more involved, but when different data sets on the same item is utilised, it renders new, valuable information about the items of interest resulting in better decision making. The use of specifically unstructered data is recognised widely as part of Big Data. It is postulated[16],[17] that around 70-80% of the data of organizations are unstructured data. The percentage given is definitely not a proven one, however, thinking of data such as emails, logs of operations, images and text documents, it is easy to see that a lot of data within organisations is unstructured.

- *Real-time analytics*: Hilbert argues that this is part of Big Data, however he says that this is definitely not a set criteria for Big Data, nor do we even need to do real-time analytics to obtain useful insights from Big Data. However real-time analytics poses problems that require the development of new techniques for processing the data.

- *Machine learning*: this is rather an *implication* of the nature of Big Data analytics, than part of the nature of it. Data can come in great volumes and variety and complexity, thus when we throw the data at a machine learning algorithm, it can find patterns that at machine level make sense, but not necessarily in theoretical frameworks. Therefore, to derive theory for what machine learning algorithms do is extremely difficult, and it is unnecessary. An example given by Hilbert is how Google® developed machine learning algorithms to develop Google Translate. They simply gave the algorithms different documents from different languages, and then the algorithms determined patterns between documents (the data) in order to translate. Very important though, is that for this to work, these algorithms do need large amounts of the data. An example given by Cukier is how machine learning algorithms were used to find cancerous cells in biopsies. This is also an example where we might not be able to theoretically explain the results, yet the results are accurate, given enough data. A leader in doing data-driven cancer diagnostics is Sophia Genetics®[18].

Finally, the most important aspect to mention is that Big Data is *new*, and it creates new *value*. This is the crux of the matter. Because of all the data being generated, and the fact that we have so much more information on the same objects as before, we can utilise it to discover new things that will enable businesses and governments in decision making, and health researchers in decreasing burden of disease.

Keeping all of the above in mind, we give the definition of Big Data that we will assume in the rest of this document. We re-iterate that we are not attempting to wrap Big Data in a box, but rather that we are trying to provide a sensible framework to work in, especially considering all the ways in which Big Data is currently being used. We also acknowledge that there are many players in the Big Data era, for instance computer scientists, statisticians and informatics and database specialists. Hence, in Definition 1.1 we attempt to reflect that, but knowing that more insight can improve it.

[16] http://www.webopedia.com/TERM/U/unstructured_data.html
[17] http://www.dummies.com/how-to/content/unstructured-data-in-a-big-data-environment.htmldon't like this ref
[18] `http://www.sophiagenetics.com/home.html`

Definition 1.1. Big data is data or collections of data that exist in different forms and moves at high speed, which challenges our capabilities of storing, transmitting, processing and analysing it because of its size, complexity and structure, requiring novel methods to do so, and is necessarily useful to create unprecedented value for decision making in government and business and other organisations, and for implementing interventions at individual level.

1.1.3 Where does Big Data come from?

Now we consider more concretely where Big Data comes from. And especially why we argue that compressed sensing is a tool to consider to use in Big Data problems. The rise of Big Data is of course directly related to where it comes from. In fact, the places it comes from gave rise to the idea of Big Data. So why is it that there are such vast amounts of information? From literature and all the discussions around Big Data we can give the following reasons:

1. Data is being re-used, leading to data on data being created.
2. The increase in design and use of sensors.
3. The rise of social media platforms such as Facebook and Twitter in the last decade.
4. Due to technological advances we are able to collect data that we weren't able to do before, for example traces of internet clicks.
5. The global shift in how we use and perceive data, especially seeing people as data generating machines.

An example of data on data being created is the work of Wellington.Wellington [19] is a data scientist in New York and he has been advocating in the last couple of years for as much accessibility to the public (people) of public data as possible. For instance he used data he got on the number of parking tickets given when people park at fire hydrons in New York, to find a parking spot where there were unclear markings on the road and a wide walkway which lead to an anomolous large number of tickets being handed out at that hydron. People parked at that spot since they thought they could, which the police then ticketed, because it was in front of the fire hydrant. Wellington noticed this, posted this on his blog (called I Quant NY[20]), and the department of transportation responded to this saying they will look into it. Unexpectedly, a few weeks later the spot was repainted and no one parked there any longer. Citizens of New York therefore no longer receive parking tickets at that spot, which shows how the re-use of data in a creative way led to an improved New York. The potential in this, especially from a Big Data point of view, is that if all the major cities in the world adopted the developing open data policies of New York, one would really be able to gain some wonderful insights and be enabled to

[19] `http://iquantny.tumblr.com/`
[20] See footnote 19.

improve the lives of people on a global scale. Big Data, if utilised creatively, cleverly and responsibly, can yield new value. In Wellington's case it was the result of creative thinking and a mindset of gaining data-driven insights. Wellington's analysis is fairly simple, but it showcases an important mindshift in how data can be used. This mindshift is what is required to make use of Big Data, as Big Data need be used creatively to create new value. An interesting news article[21] on the page of the Causality Actuarial Society posted in June 2015 talks about actuaries starting to use Big Data as well. An actuary collected reports on traffic accidents to see if there was any relationship between use of cellphones while driving, or the use of medications before driving and these accidents. The study consisted of notes on 6949 reports. Another application in this news article is where insurers use satellite images of houses for example, to settle damage claims when for instance storms cause damage to properties, or underwrite policies with more information. The images give extra information in the form of polygons (shapes defined in terms of straight lines, e.g. triangles, hectagons) in the images. The polygons can for example be used to measure a roof.

The use of sensors is certainly one of the main reasons we have so much Big Data. Sensoring systems are capturing more data than can be handled by that system [12]. Baranuik gives the examples of the surveillance imaging system developed by DARPA called ARGUS-IS[22] and the Compact Moon Solenoid detector [41] at the CERN[23] facility in Switzerland which in 2011 generated 770 Gigabits and 880 Terabits per second respectively. At that time the storage capacity of the hardware was a lot lower. Another example of the use of sensors is the envisaged radio telescope called the Square Kilometer Array (SKA)[24]. The SKA is a longterm project of building and using a radio telescope in order to better understand the universe. The low-frequency components will be built in Australia and the mid-frequency components in South Africa. On 7 December 2015 Australia committed AUS$ 293.7 over the next 10 years to the project. This is one example of where the sheer volume of data that will be created by this telescope is in itself already Big Data. Here are some interesting facts on the SKA regarding the amount of data it will produce[25]:

- The dishes of the SK-array will produce 10 times the global traffic of the internet.
- Each dish will generate $40GB$ of data per second.
- It will require about $1.9TB$ of working memory (RAM).

This is an astronomical amount of data. Compressed sensing can especially be useful in a project such as this. The International Centre for Radio Astronomy Research (ICRAR)[26] is currently working on developing the software that will be used to manage the data from the SKA telescope. They are using the

[21] http://www.casact.com/press/index.cfm?fa=viewArticle&articleID=2904
[22] `http://www.darpa.mil/i2o/programs/argus/argus.asp`
[23] `http://home.cern/`
[24] More detail at https://www.skatelescope.org/
[25] Some facts obtained from a presentation of a SKA representative at the Council for Scientfic and Industrial Research in 2015.
[26] `http://www.icrar.org/`

second fastest supercomputer in the world, called Tiahne-2. It is stated that each point of data collected in six to twelve hour intervals by the SKA will require the execution of millions of tasks on thousands of computers[27]. This emphasises how Big Data problems can require novel solutions in different fields, in this case computing.

Users of social media platforms, especially Facebook®[28] and Twitter®[29] generate a lot of data every day. The data generated by these sources are also typically unstructured. Facebook posts and tweets can contain different characters and come in different lengths. Therefore you do not have data in nice rows of observations with an equal number of entries in each row. In Figure 1.1.4 an infographic[30] is presented that was generated by Domo[31], a company that provides data solutions for businesses. In this infographic, produced in 2016, we can see the immense amount of data generated every minute. All these data can be analysed and utilised to gain valuable insights. This is reminiscent of the digital age we live in and the significance of the digital footprint. Analysing data from these sources will not necessarily be based on any theoretical framework, but will require creativity and possibly some smart techniques to extract value from it.

Points 4 and 5 above are in some sense the result of the other 3 points. For instance the ability of Google® to track your location. The data on your location is not generated by you as the user directly. However, because of sensors and technology, Google can track you. By seeing you and your device as data generating machines, it opens up opportunities like tracking your location. The question can be asked: what other forms of data are you implicitly creating as you go through your day to day routine? By asking that question many have discovered powerful uses of Big Data, and other will continue to do so in the future.

1.1.4 Was data not always big?

We have established that independently from its definition, Big Data is relevant. However, to ultimately appreciate Big Data we need to see if Big Data is new, or whether it's been around since data has been collected, just with different names in different contexts. Can the ways data was managed in the decades in centuries past not be considered as Big Data problems of its day? As Statistics evolved, along with the advent of computers in the 1900's, there are definite points where data challenged the context of its time. However, because of the technological age we live in, the things we can know about the individual, and the scale at which we can know this, make Big Data unique. Therefore, Big Data has not always been around. We can consider Big Data Analytics as a big turning point in the development of major statistical areas. Take for example data mining. Fifty years ago data mining was not at the heart and centre of the development of Statistics. Today however, the most likely place a statistician will be working, the

[27] http://www.icrar.org/tianhe2/
[28] https://www.facebook.com/
[29] https://twitter.com/?lang=en
[30] https://www.domo.com/learn/data-never-sleeps-4-0
[31] https://www.domo.com/

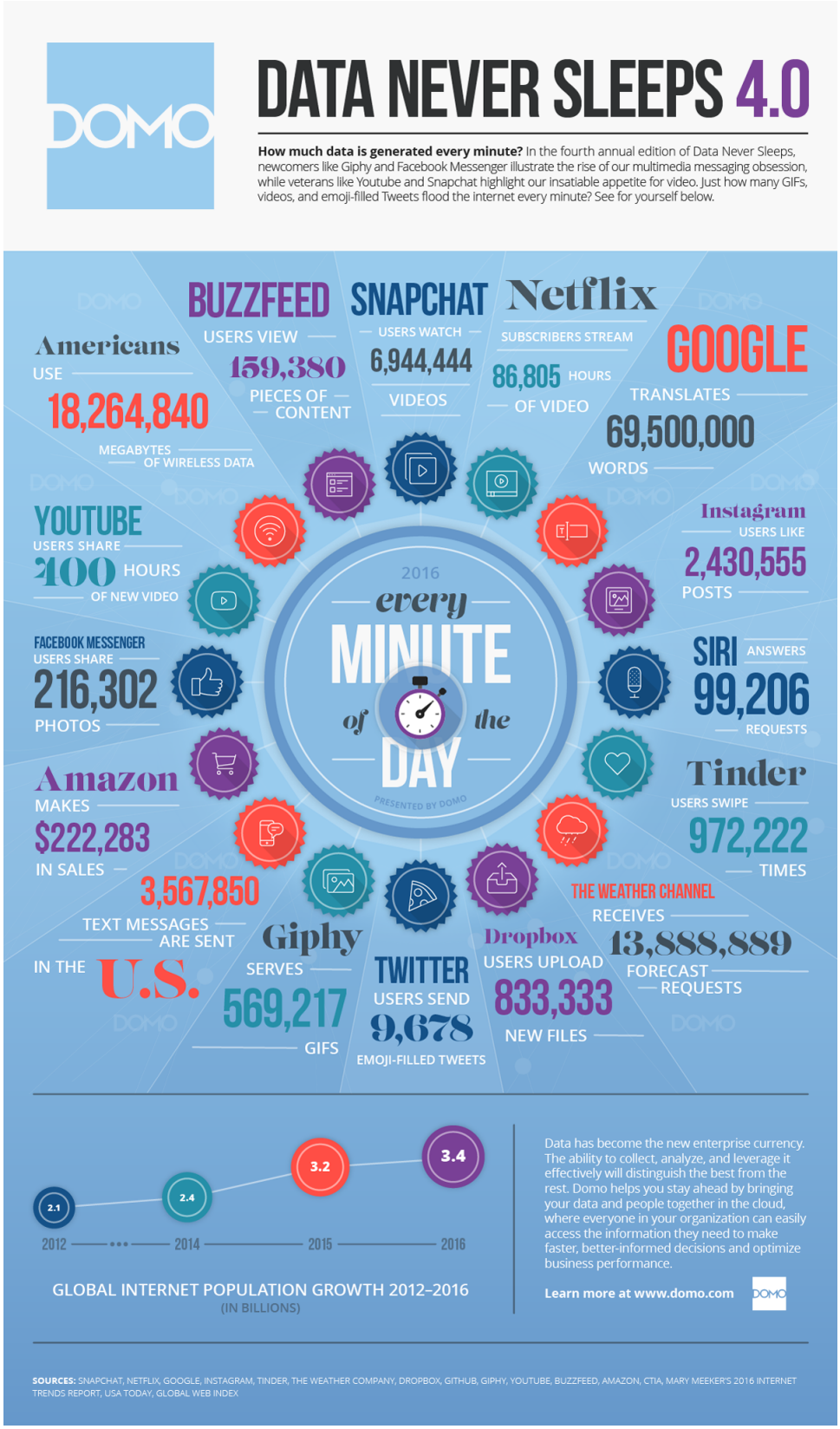


Figure 1.1.4: A 2013 Domo infographic on the estimated amount of data produced every minute via different platforms.

corporate and business sectors, data mining is a crucial skill. In the same way, statisticians, whether in business or in government, in finance or in public health, need to be able to utilise, manage and analyse the Big Data at their organisations disposal, in order to create the value that can be extracted. If statisticians along with the computer scientists and others are not able to do so, their organisations might miss out on critical business opportunities, or opportunities of serving people and improving their lives. We note very importantly that Big Data will not do away with the usefulness and value of classical statistical techniques.

Some dispute the rise of Big Data. According to [77] some objections to the notion of Big Data are:

- that it is not necessary to distinguish between standard data analysis and analysis of Big Data because as data sets grow our capabilities to work with it do too.
- we want to do real-time analytics and the most important factor is the newness of the data, not the size of the data
- Big Data analysis isn't necessarily accurate due to the number of variables; some factors may lead to bigger data not being better, but worse.
- there are ethical concerns regarding the use and misuse of Big Data. Everyone would agree that our personal data isn't as safe as it should be.

These objections in part have some truth. It is true that more data does not necessarily mean that we will have gained better insights. If not careful we could make the wrong conclusions from possible spurious relationships. However, the value that has been added by Big Data analytics up to now, shows that it can be useful. And, the valuable insights gained would not have been gained otherwise. Real-time analytics is important, but Big Data is not constrained only to analysis we want in real-time. The analytics required for proper analysis of Big Data are therefore important. The ethical side of Big Data is definitely alarming. A lot should be done to protect people from the ramifications of improper use of Big Data. So as much as there are a lot of benefits of Big Data Analytics, caution should be kept and safeguards be put in place.

1.1.5 The Big Data analysis pipeline

Big Data Analytics is part of what is called the Big Data pipeline [105] which is divided into the following stages:

1. Data acquisition and storage
2. Information extraction and cleaning
3. Data integration and representation

4. Modeling and analysis

5. Interpretation

With volume and velocity being two of the characteristics of Big Data, and sensors being one of main sources of big data, how we capture and store data is critical. Some sensor devices, for example the LMS at CERN mentioned earlier, create so much data that we cannot store everything created. And so we need to be able to decide which information is the most important to keep at the sensor stage. Because much Big Data is unstructured and because of possible measurement error in different sensors, the data needs to be cleaned and the noise should be modelled. After this has been done, the data needs to be structured. For instance, if we have data from multiple sources on the same variables, we will need to aggregate it and put it together somehow in order to extract all the value from it.

Imperative to the first three points just discussed is Apache Hadoop®[32], for example. It is open source software which allows parallel processing by distributing data and then aggregating it again. After all this, the data is ready for analysis. Data mining and machine learning is crucial in this part of the process, because of the size of the data, and the aptness of machine learning algorithms to find patterns by learning from the data. The real value in Big Data is found in its interpretation after analysis. The whole process will be redundant if it doesn't reach the fifth step. This is where decision making comes in. And this is how Big Data is transforming the world we live in.

As was mentioned earlier, compressed sensing will be most useful in the first step. In the compressed sensing paradigm the idea is to simultaneously acquire the most valuable information and compress it. Assuming that we want to find needles in the haystack of Big Data, compressed sensing is specifically useful. Especially in sensoring systems.

In summary we note that Big data:

- has volume; even such that sometimes it may represent the population of interest very well.
- is transmitted and captured at high speed.
- comes in a wide array of different forms, which can be used to supplement each other.
- can be very complex to analyse.
- predominantly is in unstructured form.
- most importantly, it is used to create new value.

[32] http://hadoop.apache.org/

1.2 Why Big Data and compressed sensing together?

The problems in Big Data that lend themselves to possibilities of applying compressed sensing in this environment are:

1. The problem of data acquisition, which is regarded as the first step in the Big Data pipeline.
2. Growing interest in the last couple of decades in sparsity and the sparseness of Big Data sets.
3. Specifically in Statistics, cases arise where $M \ll N$, i.e. the number of observations is far less than the number of measurements taken on each data unit or observation.
4. Convex optimisation is a powerful tool for Big Data problems.

We discuss each of these points in more detail.

The problem of data acquisition

This is a challenging problem since data can be captured in different ways. Of course it depends on what type of data is captured. For instance tweets on Twitter®[33] are captured as people tweet. Data on computers are generated by what the computer is doing and what operations are in progress. Census data is captured by people filling in surveys. Data captured by banks on clients have many variables. Sensor data is captured by different kinds of sensors, for example images are captured by millions of little sensors in the camera. Some data can only be captured partially. When costs or limited storage capability restricts the number of experimental units that can be tested on, for example in biology where the number of tests that need to be performed are many, but only a small number of experimental units can be tested. Some data sets are simply too large to store even in this day and age. MRI data takes a long time to capture requiring the patient to lie completely still for a period of time. Hence, given the fact that capturing data costs money, time and hardware there is definitely a need, if possible to capture and store data in cleverer ways. To capture less data without any significant loss in the amount of actual information is thus a worthy aim. In [39] many aspects of Big Data are dealt with in depth. One aspect is data acquisition. Examples of data acquisition challenges discussed in [39] are in the health sector, manufacturing, government, media and entertainment and finances. Some challenges in the health sector, for example, are maintaining privacy of data records, and amalgamating data from different sources which are structured or unstructured. Regarding the manufacturing sector, the following is said:

> In the manufacturing sector, tools for data acquisition need to mainly process large amounts of sensor data. Those tools need to handle sensor data that may be incompatible with other sensor data and thus data integration challenges need to be tackled, especially when sensor data is passed through multiple companies in a value chain.

[33] https://twitter.com/

The sparsity characteristic of data

Sparsity in the general sense is the characteristic of something not having a lot of information. It is easy to imagine that large data sets should be sparse. For instance think of a logistic regression model used to model the probability of default at banks. The bank can have hundreds and even thousands of variables. It will happen more often than not that only a small number of the variables will be significant in predicting the probability of default, with good enough predictive ability. In statistics latent variables can be seen when we do factor analysis. And usually most of the correlation is concentrated in only a few of the principal components. Images are also typically sparse in some domain, for example in the Fourier or wavelet domains. When we look at the Fourier or wavelet coefficients, we typically note that there will be only a few significant coefficients. Sparsity can also be manufactured in the sense that we can define sparsity around the object of interest. For instance how Google used their log of Google searches around the world to track flu [53, 165, 176]. In that case Google didn't need to consider every single character in every single search query. They could rather focus on search queries pertaining to flu. And so the number of queries pertaining to flu in comparison with all the queries entered, should have been relatively low. It is noted that some have found that Google overestimated the prevalence of flu [82]. Even so, it is arguably better to overestimate than to underestimate disease prevalence.

Hastie et al. consider many aspects of sparse statistical models in [93], mainly by looking at applications of the Least absolute shrinkage and selection operator (LASSO) [160]. The following is said:

> "...a third advantage [of sparsity] has emerged in the last few years from some deep mathematical analyses of this area. This has been termed the 'bet on sparsity' principle: *Use a procedure that does well in sparse problems, since no procedure does well in dense problems.*"

Therefore, sparsity plays an important role today in solving Big Data problems.

Statistical estimation when $M \ll N$

In Big Data problems the number of variables, N, will many times exceed the number of observations, M, and that by large factors. The reason being that we want to find a set of parameters of a model where we have less observations than parameters, and of course this is not a straight forward problem. Candes and Tao propose a solution to this problem in [25].

There are definite challenges that come with the high-dimensional problem where $M \ll N$. In [109] this is discussed and different solutions for different situations are overviewed. Throughout the article the exploitation of sparsity makes most of the overviewed problems feasible. They give an overview of the problem and touch on LASSO and the Dantzig selector [25] and show how it is used. They also touch on classification, visualisation and Bayesian approaches. This article serves as a good spring board to get some references and an overview of the problem of high-dimensionality from a statistical perspective.

Both linear and non-linear models are considered, as well as models with Gaussion noise and more general cases. According to [109] applications can be found in microarray analysis, image analysis, astronomy, geophysics to name a few. This accords with the examples in [40]. The problem that springs forth from this is that of variable selection, specifically in high dimensions.

For this problem to be really seen as a Big Data problem, we should have that the number of variables is extremely large. Microarrays of genomes [57] have many variables and have interesting applications from a statistical point of view. Typically they attempt to understand complex triats in human genes better. For example, in South Africa, the genes of tuberculosis patients are investigated to understand mutations caused by drugs [103].

The use of convex optimisation

Convex optimisation is an extremely powerfool tool. If one can identify a problem and formulate it in terms of a convex problem, there are powerful ways of solving that problem not otherwise possible or necessarily as optimal. A very good example in statistical literature is ordinary least squares (OLS), albeit a very simple convex optimisation problem. Other examples are ridge regression [98], the LASSO [160] and regressing on the conditional median [114, 48]. The latter two are examples where we look at the ℓ_1 norm of the parameters rather than the ℓ_2 as in least squares and ridge regression.

In [40] the authors give motivation for using convex optimisation in Big Data and techniques that could be used to solve problems in Big Data. They say that

> "...the importance of convex formulations and optimization has increased even more dramatically in the last decade due to the rise of new theory for structured sparsity and rank minimization, and successful statistical learning models such as support vector machines. These formulations are now employed in a wide variety of signal processing applications including compressive sensing, medical imaging, geophysics, and bioinformatics."

Matrix completion fits into a similar paradigm as that of compressed sensing. In [28] Candes and Benjamin describe how matrix completion is closely related to compressed sensing and how it is useful for a Big Data problem such as the Netflix problem [14]. And so compressed sensing can be valuable to Big Data problems.

1.3 Overview of rest of document

We have now introduced the Big Data problem and why compressed sensing might be a potential tool in some Big Data applications. In Chapter 2 we look at some basics required to understand compressed sensing. We focus on sampling and data compression in Section 2.2 and transforms and filters in Section

2.3. We look at data compression because it is important to understand what the relations are to compressed sensing and why it is different. We also consider two important transforms which play an integral part in compressed sensing. The concepts in Chapter 2 are helpful and essential to understanding compressed sensing, especially for the reader who has in no way investigated the world of signal processing. In Chapter 3 we discuss various aspects of compressed sensing. In Section 3.2 we present a literature review on compressed sensing. Then we discuss the fundamental concepts in the compressed sensing framework. This includes how to sparsify your signal of interest (Section 3.3.3), how to sample it (Section 3.3.4) and how to recover it from the samples (Section 3.3.5). Finally, in Chapter 4 we present an argument for the use of compressed sensing in Big Data applications in Chapter 4 given what we investigated in the first three chapters. Specifically compressed sensing is implemented on a set of images to illustrate its use on the Big Data of images.

Chapter 2

Background theory

The purpose of this chapter is to convey some necessary ideas on our path to understanding compressed sensing (CS). Although one could dive straight into the CS literature, getting to know the concepts discussed in this chapter is extremely helpful, especially to someone new to the signal and image processing world.

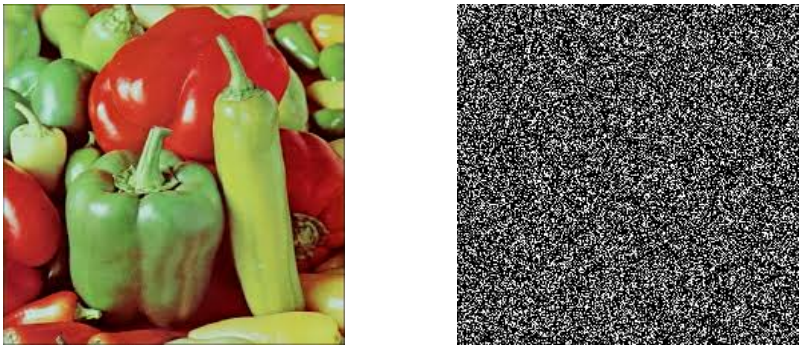
2.1 Information and signals

2.1.1 A thought experiment on information

We begin by giving a few examples of objects with different information content. The experiment is to guess which of the cases of each example contains the most information:

Example 2.1. We have two images. The peppers image can be found for example at the USC-SIPI image database[1].

[1] `http://sipi.usc.edu/database/database.php`



(a) The peppers image (b) Gaussian noise image

Figure 2.1.1: Standard peppers image and noise image.

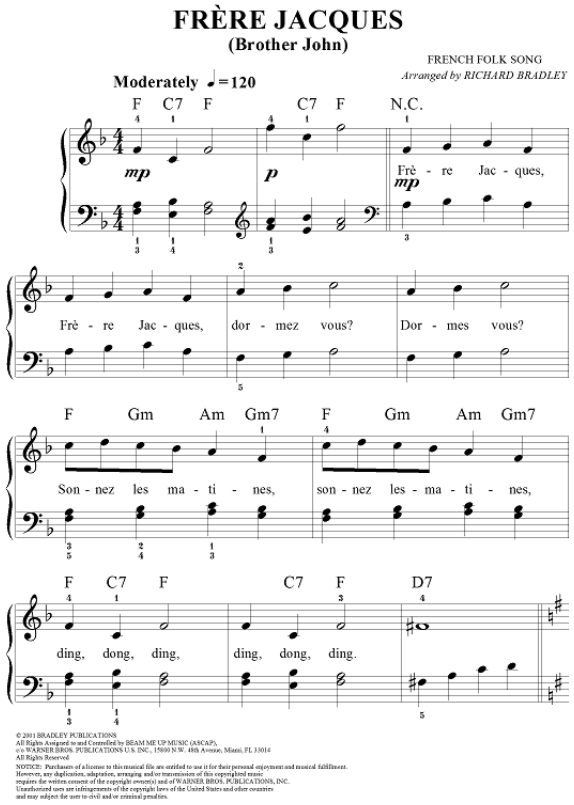
Example 2.2. Consider the following two statements:

(1) "I am a masters student and I am in the process of writing up a masters dissertation to fulfill the requirements of the masters degree."

(2) "I writing masters dissertation degree purposes."

Example 2.3. Two versions of the traditional French folk song "Frere Jacques" [2,3] in Figure 2.1.2. In the first arrangement in Figure 2.1.2a we do not consider the first two bars of music nor the last one.

[2] www.onlinesheetmusic.com/frere-jacques-p285541.aspx
[3] www.mamalisa.com/?t=es&p=180



(a) First sheet music arranged by Richard Bradley.

(b) Second sheet music.

Figure 2.1.2: Two different arrangements of sheet music for “Frere Jacques”.

Example 2.4. Sequences of numbers

(1) $1, 1, 2, 3, 5, 8, 13, 21, 34, 55$

(2) $12, 6126, 59, 99929$

In Example 2.1 the noise image has more information. In Example 2.4 it is the second sequence. We have exactly the same information in the two sheets of music in Example 2.3. The two sentences in Example 2.2 have more or less the same amount of information. To someone thinking about this for the first time it might be counter-intuitive. How could something that gives a clearer understanding of the

communication have less information than something, such as the noisy image, that do not communicate anything sensible and do not provide any 'information'? We can look at it in two ways. Firstly, we have to look at it the way a computer looks at it, that is, in terms of what is called bits. Secondly, we look at it from a statistical point of view, that is, patterns and dependencies. First we focus on the former.

2.1.2 Bits of information

Signals can be seen as streams of bits. In other words, we can describe a signal with a sequence of bits. Each bit is a brick in the bitstream, which ultimately forms the signal. One bit can only contain one number. We referred to [12] which stated that the amount of information in 2010 in terms of bits was more than the stars in the universe (it is probably questionable how that was estimated, but the point is that an extreme amount of infomation was generated globally and a good measure is in terms of bits). The point is that bits are the way information is described at machine level.

Not only is information stored in terms of bits, but more specifically in terms of binary values. The core idea behind this is that if we break information down, we can represent it as a sequence of 'yes' and 'no' values. Equivalently we can say 'True' or 'False' or in terms of switches, 'On' or 'Off'. The latter is important in engineering and electronics. We represent it at machine level as 1 = Yes/True/On and 0 = No/False/Off. Let us show the use of this in terms of a numerical example.

Take for example the number 13. In terms of the prime 2, we can write $13 = 2^3 + 2^2 + 2^0$. So let us ask the following questions. Is the number 13 in its decomposition a sum of 2^5? No. Is the number 13 in its decomposition a sum of 2^4? No. Is the number 13 in its decomposition a sum of 2^3? Yes. Is the number 13 in its decomposition a sum of 2^2? Yes. Is the number 13 in its decomposition a sum of 2^1? No. Is the number 13 in its decomposition a sum of 2^0? Yes. So the above sum we could re-write as $13 = 0 * \sum_{n=5}^{N} 2^{n-1} + 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0, \forall N > 4$. Therefore, if we represent the number 13 in binary bit form, where position $n \in \mathbb{N}$ (starting from right) corresponds to the value 2^n we write 13 = 001101. Technically we can add any number of zeros in the front. With the way we expressed it here, we have represented 13 as a 6-bit byte. A byte is simply a unit of bits, which can represent characters, such as integers in this case. Therefore using $n-$bit unit sizes we can represent 2^n numbers between $[0, 2^n - 1]$.

As further illustration we show the integers $\{0, 1, \ldots, 7\}$ in terms of 3-bit and 5-bit bytes. The difference of course between the two representations is that with the first we have all $2^3 = 8$ numbers we can represent with a 3-bit system, and with the latter we only have the first eight numbers of a possible $2^5 = 32$.

$$
\begin{array}{lllllll}
0 & = & 000 & 0 & = & 00000 \\
1 & = & 001 & 1 & = & 00001 \\
2 & = & 010 & 2 & = & 00010 \\
3 & = & 011 & 3 & = & 00011 \\
4 & = & 100 & 4 & = & 00100 \\
5 & = & 101 & 5 & = & 00101 \\
6 & = & 110 & 6 & = & 00110 \\
7 & = & 111 & 7 & = & 00111
\end{array}
$$

Convention is to use binary valued bits, however we could technically use other bases than the binary. For example the system based on 3 values. Using a 4-bit stream we can represent the above numbers as:

$$
\begin{array}{lll}
0 & = & 0000 \\
1 & = & 0001 \\
2 & = & 0002 \\
3 & = & 0010 \\
4 & = & 0011 \\
5 & = & 0012 \\
6 & = & 0100 \\
7 & = & 0101
\end{array}
$$

Using this system we will be able to represent $3^4 = 81$ unique numbers. A system that is also used is the hexadecimal system with base 16. However, since we only have 10 single number integers, the system uses $0, 1, \ldots, 9, A, B, C, D, E, F$.

2.1.3 Communication and a profound discovery

Besides information being important in itself, how it is transmitted is where it becomes really important. Although two people can have valuable information they want to share with each other, if they can't share it meaningfully, it doesn't help. We can refer to this transmission of information as *communication*. *Effective* and *efficient* communication of a bit stream (a signal), is at the centre of most signal processing problems. We will see that this is also why CS is important and useful. From a Big Data point of view, transmitting large amounts of data is quite a challenge. It is one of the reasons we are interested in data compression. This problem in particular involves the aspects of the size of data, the ability to store that data, as well as the ability to reconstruct the data in a satisfactory way. In the late 1940's Claude Shannon published a groundbreaking paper on information theory [153]. Shannon attempted to answer the simple,

yet profound question of what the maximum amount of information is that can be communicated. He defined fundamental concepts which govern the signal processing world today. He defined what maximum entropy is and that we should look at information in terms of 0's and 1's. Combining clever mathematics and principles from thermodynamics (such as entropy), he laid the foundation for digital communication.

In Statistics, regression is a good way of transmitting information efficiently. Do we need to communicate the whole data set in order to convey the information in it? No, we only need the estimated regression model. If the regression model is a good model, our error will exhibit Gaussian white noise, with variance not too large, and we will be able to efficiently and effectively (because of the errors, not exactly) communicate the information contained in our data set. Even when some of the regression coefficients are zero for some of the variables. Although not all the variables are represented when some of the coefficients are transmitted as zero, we can ask the question if that specific variable contributes any information regarding our response variable? Therefore understanding some fundamentals of communication is important to the application of CS. And statisticians can do well to think in similar terms.

2.1.4 Patterns and dependencies

This brings us to the second viewpoint discussed above - the statistical point of view. Understanding what information is and where it comes from is cardinal to signal processing. An interesting question regarding different kinds of information is how much meaningfulness is there in it? Whether music coming to our ears, words in a letter or pixels in images? Let us consider the two images in Example 2.1 again. For both cases, what would be the least amount of space/memory/bits/info we need to perfectly represent the information? The instinctive answer would be that we need all the pixels! Which is true in some sense. But do we really need every single pixel to represent the images? In the case of the noisy image the answer is yes. The reason being that there is absolutely no relationship between the pixels. There are no patterns. Not even between small clusters of the pixels. We would need to give the value of every single pixel to have all the information. In the colour image in Example 2.1 we see patches of green, patches of yellow, we see forms and edges. In other words, there are parts of the image where the pixels are correlated in some way. The one pixel is related to the next and therefore patterns are present.

We recognise patterns through repetition of the same thing or dependencies between elements in a signal/image. In Example 2.4 we have that the first sequence of numbers has a pattern. It is known in mathematics as the Fibonacci series. We can recognise patterns and use it to communicate the sequence more efficiently. In image processing patterns are extremely important. Pattern recognition has numerous uses and applications [144]. Patterns in images look different. Some examples are edges and curves [106]. Patterns in text are also important [183]. Of course we expect to see a lot of patterns in text. We already mentioned the example of an actuary using Big Data to find patterns or repetitions in accident reports to identify causality of accidents in Section 1.1.3. In Big Data, graphs are especially useful to identify patterns in the data. Graph analytics provide a top level view of data in a database in terms of

relationships. With graphs one can recognise interesting patterns between different elements in data, for example see [23]. It is especially useful because in Big Data we are confronted with many correlations between different elements in our data. The way information technologists look at graphs, are basically the same as the way mathematicians look at graphs. Graphs consist of vertices and nodes, as it does in the mathematical definition.

Patterns, frequencies and waves

There is a natural transition that can be made between seeing signals in terms of patterns, and seeing signals in terms of frequencies. High frequencies correspond to a pattern, or event, that repeats a lot in the signal. Low frequencies the opposite. Frequencies are related to waves. Consider the signal $f(t) = \sin(t)$. If we measure the sine wave over $t \in [-4\pi, 6\pi]$ we know that we will see the pattern repeating four times, having a period of 2π, starting from $t = -4\pi$ and repeating at points $t = -2\pi, 0, 2\pi, 4\pi$. There is one pattern and hence one frequency involved, i.e. frequency$=\frac{1}{\text{period}} = \frac{1}{2\pi} \approx 0.159$. A higher frequency would have repeated more times over the same time length. Thinking about a letter, high frequencies would correspond to words such as 'I', 'to', 'the', 'in' and 'for'. The reason being that they are words typically used to link ideas and concepts, and give detail to the communication. Let us say the letter was about your experience in an art class, words that might appear often, but not that often are 'brush', 'paintings' or 'palette'. These words will be used more often than words not corresponding to the context, but a lot less than the higher frequency words. These words typically correspond to low frequencies and give an approximation of the text (signal). We'll pick this up later in the chapter. Although the patterns in graph analytics can be qualitative, many patterns can also be identified by looking at various frequencies. An example of this would be the identification of term frequencies. In a large corpus of text documents, one may be interested in the importance of different words [138, 126].

2.1.5 Important definitions in information theory

We now define a few important concepts relating to information:

1. Entropy
2. Encoding and decoding
3. Redundancy and irrelevancy
4. The uncertainty principle

Entropy (in information theory)

Entropy is a measure of information content [153, 151]. More accurately, it is the average amount of information contained in one n-base symbol. It is defined as

$$E = -\sum_{i=1}^{n} P_i \log_2 P_i$$

where $P_i = P(\text{occurrence of symbol } a_i)$ so that E is the entropy of a stream with n symbols, $a_1, \ldots, a_n$. Shannon borrowed the terminology from thermodynamics, where it has a similar meaning of being the chaos in the system. The higher the entropy, the higher our uncertainty regarding the signal because we need on average more bits to represent the signal.

There is a very interesting implication of entropy. Saloman refers to it as a general law of data compression in [151]. The less likely an event, the more information it contains, whereas the more likely the event, the less information it contains. That translates to saying less probable events get longer codes (in terms of bits) and more common events get shorter codes. We have already established this by looking at information in terms of bits and in terms of patterns. We can also say that the uncertainty of some event is high or low, corresponding to high or low entropy.

Encoding and decoding

Encoding can be described as the process of taking information and transforming it into some other form for purposes defined by the user. Decoding is the opposite. Encoding and decoding looks different in signal processing, compression and computer engineering, but essentially it is the same thing in each domain. Mostly we would talk about encoding in the machine level sense. In other words, how do we transform or represent different sets of data for storage or transmission purposes. For example, text files are represented using the ASCII format [89]. Because all the data that we are creating is making it more and more difficult to encode efficiently, given global storage capacity constraints and processing constraints, CS will play a crucial part in the encoding and decoding of data. To understand how to encode one needs to understand how information in signals work. A simple example of encoding is the encoding of the image in Figure 2.1.3 given in a 1-bit binary system as

$$0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1$$

where 0 represents black and 1 represents white. But we could encode it differently. Using 0 to say that the sequence starts with black, we can encode the same sequence as:

$$0, 2, 2, 1, 3, 1, 2, 5, 2, 1, 2, 2, 2.$$

This way of encoding is called run-length encoding (RLE) [151]. We use the run-lengths of symbols to encode. In this case it was very simple. We start with 0 and then we count the number of black blocks,

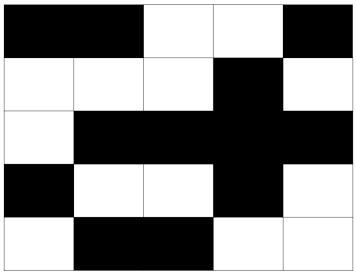


Figure 2.1.3: Binary image with 25 white and black pixels.

then white, etc. The result of the process is the second sequence we obtained. But we note that the second sequence is not binary valued. In the 2-base (binary), using 3-bit binary codes we can represent the run-length encoded sequence as

$$000, 010, 010, 001, 011, 001, 010, 101, 010, 001, 010, 010, 010.$$

We see therefore that encoding is the process and method of taking an input data stream and representing it in another format for the purpose of transmission or storage. It is fascinating to think that we can sample signals, given we have an instrument that can measure the signal, and encode it digitally and then manipulate it. For example images, audios, telephone calls and even physical occurences such as earthquakes [156]. Other and more sophisticated encoding techniques include:

- Huffman coding [101]
- Shannon-Fano coding [154]
- Move-to-front coding [150]

These techniques are used in data compression. Data compression and encoding are strongly related as when we encode a signal we want to do it as efficiently as possible. We look at data compression in Section 2.2.2.

We already mentioned that there is a different way we can talk about encoding. When we take the transformation $y = f(x), f : \mathbb{R}^n \to \mathbb{R}^m, m, n \in \mathbb{Z}$, we are in fact encoding x. The output after encoding is y. To get back x, when we have y, is what is called *decoding*. To understand why we need compression, we need to understand encoding as in the former description here. However, seeing it in the latter way is also essential to understanding the mechanics of CS. We will touch on this again in Chapter 3. An example of encoding in the latter way is when we transform a signal into the Fourier or wavelet domain. But the former is important for what happens at machine level when we apply transforms such as the Fourier and wavelet. An example of a technique to encode is the Discrete Wavelet transform (DWT) described in [7].

Redundancy and irrelevancy

In most cases signals contain information that is not required to represent the signal effectively. In other words, we discriminate between important and unimportant information in signals. In Example 2.3 we show two representations of sheet music of the well-known French folk song "Frere Jacques". We can see that in Figure 2.1.2a the whole song is written out in notes. Notice that there are four parts of the music that each repeat twice. In Figure 2.1.2b we use half the space to communicate exactly the same information. The bars with the colons in music mean that you should repeat the section. So when we encode this, we can simply put something in that says we should repeat the section we just finished. We do not need to encode every single note in the song. This is what is referred to as removing the *redundancy*. We make the signal smaller (in terms of bits) by removing information that doesn't contribute uniquely. In Example 2.2 we have two statements that also communicate the same information. The information in both statements are the same, i.e. a masters student is finishing his/her dissertation as part of the degree, in a written fashion. The word "masters" is repeated two times, and in both cases it is already apparent that it is part of the meaning. We only need the first time it is used to establish the meaning it conveys throughout the rest of the sentence. Therefore the extra use of the word "masters" is redundant.

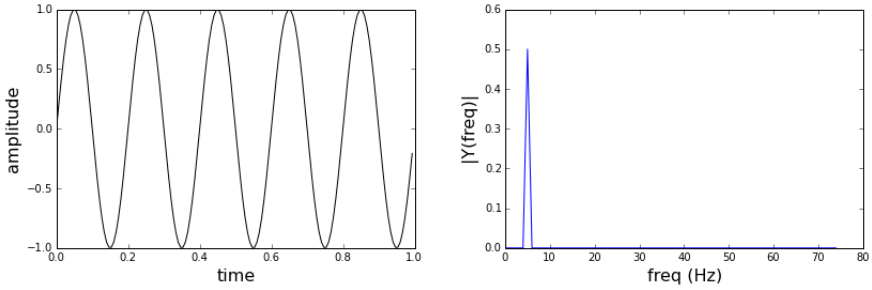
However, by taking out words such as "am" and "to" we are removing irrelevant data. It does not contribute the most significant part of the meaning of the sentence. Irrelevant information in signals do contribute to the overall value/meaning of the signal, but its contribution is usually so small that when we throw it away we still have the significant parts of information to communicate the meaning/value of the signal sufficiently. Therefore, irrelevant information differs from redundant information in that the latter doesn't contain any information and contributes nothing to the overall value/meaning of the signal. This difference is one of the key factors in efficient compression which we discuss below.

The uncertainty principle

The uncertainty principle was first observed by Heisenberg in physics [95, 113]. The observation is that we cannot understand or know, fully, both the momentum and the position of a particle at the same time. This is expressed mathematically with the following relation:

$$\sigma_m \sigma_p \geq \frac{h}{4\pi}$$

where σ_m is the standard deviation or uncertainty of momentum, and σ_p the standard deviation or uncertainty of position and h is the Planck constant equal to $6.626176 \times 10^{-27}\text{erg} \cdot \text{sec}$. The fact that $\sigma_m \sigma_p$ is lower bounded, implies that the level of uncertainty will never be zero and hence at a very small point, usually not observable to the naked eye, if we decrease the uncertainty in either position or momentum, the uncertainty in the other will increase. In terms of probability this means that if we could with very high probability know for example the position of an object, we would at some stage, be



(a) Sine wave in time domain.

(b) Frequency domain representation of sine wave.

Figure 2.1.4: Sine wave with corresponding frequency domain representation.

increasingly uncertain about its momentum.

It might seem completely unrelated by talking about quantum physics, but the relation is significant because it is found in other places as well. A wonderful example is time and frequency. In Figure 2.1.4 we show a sine wave with its representation in the frequency domain. We note that although it is spread out (global) in the time domain it is isolated (localised) in the frequency domain, only having one point where the magnitude of the frequency spectrum is not zero. In other words, if the signal is localised in time, we will not really be able to tell which frequencies form part of the signal. So we have uncertainty in the frequency domain.

2.2 Sampling and compression of signals

In this section we focus on compression and sampling. Both elements are crucial to CS. Compressed sensing simultaneously acquires and compresses the signal of interest. Therefore we need to understand some aspects of both compression and sampling.

2.2.1 Sampling and digitisation

Sampling signals is an art in itself. Theoretically we can do wonderful things when the signal is continuous. Real life signals are continuous. Our ears pick up sound from the radio in a continuous way. Our eyes give images that we see that are continuous. This of course implies a practical problem. How do we work with continuous signals? Can we work with it in a continuous fashion? Do we need to discretise? In signal processing we do both. We can work with signals in both continuous and discrete domains. In electronic enigineering we either work with the analog signals (continuous) or with digitised signals (discrete signals). Examples of both can be found in [140, 145]. It depends on how we sense or measure the signal. A very good example is how pictures were taken before digital cameras. The photos that were taken were continuous, which were exact representations of the real life image. The exactness was

obtained using physics and chemistry [49]. Light obviously played an important part in this process, as the photos were as good as the light was. Today however our world is digital. When we take a photo with a digital camera the physical process is still present in some sense, however, the measurements are discretised by using only a limited number of finite sensors on the camera to take the measurements. Consequently we represent some continuous image with discrete values.

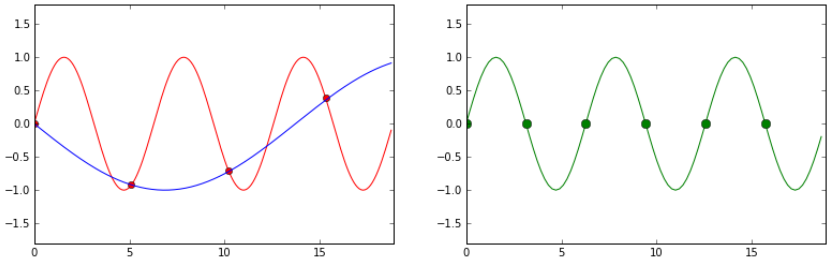
Remark 2.5. We know that we are referring frequently to Electronic Engineering and independently from that, image processing. This is a document with a statistical emphasis, and especially with Big Data as one of the two focus areas, but it is worthwhile appreciating the relationship between Electronic Engineering and Statistics, as well as image processing and Statistics. It has been a revelation to see what these three fields have in common, and what wonderful things can be done through how these fields complement each other.

Paramount to signal processing and especially in the sampling of signals, is the Nyquist-Shannon sampling theorem. It was formulated based on work by Nyquist [139], Shannon [153], Whittaker [175] and Kotelnikov [117]. Many propogate CS by saying that it proposes a sub-Nyquist model of sampling. In other words, the number of measurements required for CS to work is far less than the traditional way of sampling according to the Nyquist-Shannon theorem. Therefore we need to understand what is meant by the above and when it is true.

Nyquist-Shannon theorem

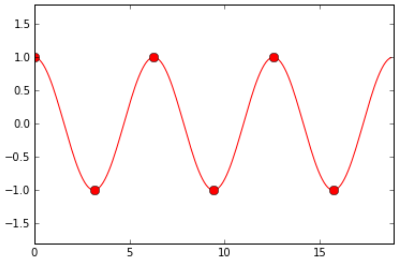
Theorem 2.6. *Given a signal $f(t)$ that in the frequency domain has maximum frequency B, we need to take measurements not more than $\frac{1}{2B}$ time points apart in order to be able to perfectly reconstruct the original signal. Or put differently, we need to sample at least at twice the maximum frequency in the signal of interest [153, 139, 175, 117].*

We haven't covered the frequency domain yet, but we will look at the Fourier transform in Section 2.3.4. However, all we want to note now is how sampling works. The Nyquist-Shannon theorem stipulates a way in which signals should be sampled to retain all the information in the signal. We already mentioned that frequencies give some insight into some types of patterns in signals. And so this sampling theorem exploits the fact that the stronger the patterns in terms of frequencies in the signals, the less number of measurements we need to take in order to capture all the information in the signal. Let us illustrate the point, looking at the sine function and its frequency representation in Figure 2.1.4. The function is $f(t) = \sin(t) = \sin(2\pi(\frac{1}{2\pi})t) = \sin(2\pi\omega t)$ with $\omega = \frac{1}{2\pi}$. We note that there are no other frequencies in the signal. Hence we can explain $f(t)$ by simply having the knowledge of the frequency $\omega = \frac{1}{2\pi}$. Now we apply the Nyquist-Shannon theorem. The maximum (and only) frequency of $f(t)$ is $B = \omega = \frac{1}{2\pi}$. Hence we need to sample $f(t)$ at $\frac{1}{2B} = \frac{1}{2(2\pi)^{-1}} = \pi$ intervals apart, or put otherwise, twice every period where period $= \frac{1}{\text{freq}} = 2\pi$. First let us see what happens when we sample the sine wave at a lower frequency. This is illustrated in Figure 2.2.1a. We can see that if we sample at intervals larger than π, we will end

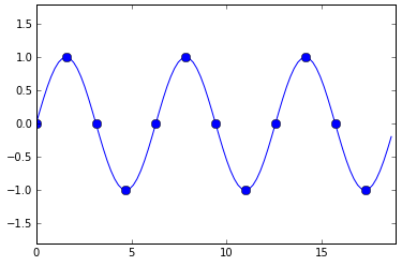


(a) Undersampling of signal.

(b) Nyquist sampling in wrong way.



(c) Correct Nyquist sampling of signal.



(d) Oversampling of signal in order to obtain correct representation.

Figure 2.2.1: Sampling of signals in four different ways.

up representing a different sine wave with a lower frequency.

Consider next what happens when we do sample the sine wave at exactly the Nyqusit rate, i.e. every π time units. We can see that we sample the sine wave every time it crosses at zero (Figure 2.2.1b). All we will end up seeing is a bunch of zeros. However, sampling a cosine wave at the Nyquist rate, as in Figure 2.2.1c, enables us to interpolate the cosine wave from the samples exactly. We see therefore that sampling at exactly twice the highest frequency component in a signal doesn't necessarily guarantee exact reconstruction of the signal. That is why enigineers usually oversample, meaning they sample at higher rates than what the Nyquist theorem suggests. In Figure 2.2.1d we provide one solution for sampling the sine wave in such a way that exact reconstruction is possible.

Sampling at lower than the Nyquist rate will cause *aliasing* to happen [145]. Aliasing means that since we are not sampling quickly enough, there might be higher frequencies that we miss when sampling. Hence, there occurs some ambiguity in the information we have. Electronic engineering of course looks at solving this problem, or working with the consequences. Compressed sensing is not a direct solution to this. It rather exploits the cases when the signals of interest are *sparse*. Then we can sample below the Nyquist rate.

2.2.2 Data compression

We represented the binary image in Figure 2.1.3 in three different ways:

1. 25 digits either 0 or 1, corresponding to black and white pixels respectively.

2. A sequence starting with a 0 or 1, representing the first block, and then 12 run-lengths.

3. The run-length sequence, but where the numbers are in 3-bit format.

In the first sequence we had 75 bits of information if we work in a 3-bit system (25 in a 1-bit system). In the third sequence which we also represented in a binary way using 3-bit bytes, we had 39 bits (13 numbers times 3 bits for each). Looking at the sequence in this way we used 36 less bits than with the first sequence (assuming we have to use 3-bit bytes for all numbers). This is what compression is: to *remove redundancy*. We didn't alter the content, but rather represented the same amount of information more efficiently by removing the redundancy, which in this case was done by using the run-lengths of repetitions of white and black pixels. The question is whether this is sensible? Is the use of compression redundant?

The answer to that question is categorically 'No'. We note the following reasons:

1. Signals typically contain redundant information, in other words information that doesn't uniquely contribute to the meaning/inherent value of the signal [151].

2. We have already argued that with the data deluge, even though we have super fast computers and better technology, we do not have enough storage capacity [12].

3. Data compression can speed up processing in many applications, by rendering less data (in terms of bits) to be transmitted in communications.

These and other reasons have enormous implications for the technological world we live in. Compression also goes further than just removing redundancy in signals. We do not always need all the unique information in a signal to be able to represent it in a way that is sufficient for its use. We can therefore also remove *irrelevancy* which is described above. This leads us to the two types of compression methods: (1) *lossless*, when we only remove redundant information, (2) *lossy* when we remove irrelevant information. Compression methods and rationale differ according to the type of signal we are working with. Therefore, images, audio files and text files are compressed using different approaches. Salomon discusses various kinds of compression methods thoroughly in [151].

When doing compression we also need to decide if we want to use *adaptive* or *non-adaptive* methods. To make this distinction is very important in signal processing. A method that is non-adaptive, as the term suggests, doesn't adapt because of any information accumulated during the data compression steps. It is defined a priori how the method will be applied step-by-step - one could say non-adaptive methods are 'dumb', whereas adaptive methods use the information gained as the process goes along and could be describe as 'clever'. Usually adaptive methods are more accurate than non-adaptive methods, but it is also more time-consuming because of complexity which leads to more calculations required to implement it. Let as assume we stand in front of a maze where all paths have binary forks in it (split only into two different roads). We also have that some roads cross, some don't, some are longer and some are shorter.

We also assume that you are guaranteed to get to the other side. The aim is to find the pathway which will get you the quickest through the maze. We have two people that approach it differently. To the one, we only give a random number generator producing a Bernoulli variable with probability 0.5, and to the other we give a watch and a compass. We also assume that the sun is visible from the start. If the person with the random number generator gets a 0 he turns left, if he gets 1 he turns right. The other person will use the time, and the position of the sun and the compass, to determine in which direction to go. So at each fork the person will calculate his position and the direction he should go. At each fork he uses information given by the process, to determine his next step, whereas the person with the random number generator is totally unaware of his position. The person using the random number generator corresponds to someone using a non-adaptive approach to solving the problem, whereas the person with all the tools, use an adaptive approach.

2.2.3 Image compression

The JPEG standard [171] is well-known, even outside the signal processing world. JPEG makes use of the Discrete Cosine transform [3] to compress images. Hence JPEG is a compression method that uses the transform approach (other approaches mentioned below). The JPEG standard was improved with even better compression results when JPEG2000 [1] was introduced. The JPEG2000 utilised the more powerful Discete Wavelet transform (DWT) which was developed over a number of years with many contributing researchers. See [22] for more details. Using the wavelet transform we illustrate the rationale behind compression. In Figure 2.2.2 we give the peppers image with reconstructions using certain percentages of the wavelet coefficients calculated using the Daubechies 4 (DB4) wavelet (see Section 2.3.4). We can see that the reconstruction using only 29% of the coefficients seems very accurate and to the eye the difference between the reconstruction and the original is small.

Transforming our data is extremely powerful and helpful. For example the following transforms are mentioned in [151]:

- Haar transform [158].
- Karhunen-Loeve Transform [111, 121] (also known as the Hotelling transform or Eigenvector transform).
- Discrete Cosine transform [3].

To implement image compression, we need to be able to work with the images in some numerical format. This is where digitisation (see Section 2.2.1) gives us pixel values. The pixels in images can be represented in different ways of which RGB codes (see [143] for details) are the most prominent. Images can also be represented in terms of the HSI model, which corresponds to levels of hueness, saturation and intensity [107]. The RGB colour values are coded using three bands, Red (R), Green (G) and Blue (B). Each

(a) Using 0.3% of coefficients. (b) Using 1.2% of coefficients. (c) Using 3.3% of coefficients.

(d) Using 9.4% of coefficients. (e) Using 29% of coefficients. (f) Using all the coefficients.

Figure 2.2.2: Reconstructions of the pepper grayscale image with different number of wavelet coefficients.

colour in an image is a mix of red, green and blue. For each band we have $256 = 2^8$ shades that can be used. The set of values is therefore given by $\{0, 1, 2, \ldots, 255\}$ where 0 is the darkest shade of either red, green or blue, and 255 the lightest. Hence we have that if the pixel has value 0 in each band, the colour represented is black. And so if the value in each band is 255 we have white. In terms of bits, in a binary base we use 8 bits for each colour band, and consequently 24 bits in total for each pixel. There are different approaches to image compression useful for different image types given by [151] .

2.2.4 Error metrics and performance measures

In order to compare different compression methods with one another, we need ways to determine the performance of each compression method. *Error metrics* are measures that quantify the loss in quality because of some compression method, or for that matter other signal processing methods. They therefore compare the original input stream of data with the output after the method was applied. Error metrics should produce a scalar value. If it has more dimensions then it might still mean that two methods will be incomparable, since two vectors/matrices for error metrics do not absolutely determine how two methods compare with one another. One would in any case use the vectors/matrices and calculate some scalar value. *Performance measures* quantify how well and effective some algorithm or method achieves the desired result. An obvious example would be how long it takes in time to implement the method. We look at the following, specifically from an image compression perspective:

- Compression ratio (performance measure)

- Number of operations (performance measure)
- Peak signal to noise ratio (error metric)
- Signal to noise ratio (error metric)
- Mean square error (error metric)
- ℓ_p metrics (error metric)
- Structural similarity index (structural measure)

Compression ratio

The *compression ratio* (see for example [151]) is defined as

$$\text{Compression Ratio} = \frac{\text{size of output stream}}{\text{size of input stream}}.$$

This simply gives the ratio between the size (in terms of bits) of the input stream, and the size of the output stream (the compressed signal). Successful compression hence implies that the ratio is smaller than 1.

Number of operations required by algorithms

Every algorithm does a number of calculations when it is applied. Even a simple exercise such as calculating the sum of the elements of a $M \times N$ matrix can be measured in terms of the number of steps it needs to apply the algorithm. In that case the number of operations/steps is $S = MN$. Which can be expressed in the order of N, written mathematically as $S \sim O(N)$. This is useful since we can compare the effectiveness of different algorithms. Yet again, it need not be only compression algorithms, but any process where calculations are done. It is also useful since we can theoretically derive the order of calculations, where the order can be for example of natures linear ($O(N)$) or polynomial ($O(N^r), r > 1$) or exponential ($O(a^N)$ where $a \in \mathbb{N}, a > 1$). This helps us to identify if solving a specific problem is feasible or not. An example of problems that are infeasible are called NP-hard problems which can be read more about in [166]. They are infeasible because we do not have the computing power to find a solution. In fact, for such problems, we do not even have enough computing power to check the answer if we were given it.

Mean square error

The *mean square error* (MSE) is defined as

$$\text{MSE} = \frac{1}{N}\sum_i (P_i - Q_i)^2$$

where P_i is defined as the i^{th} pixel value, Q_i is the value of the i^{th} reconstructed pixel value (see for example [151]) and N is the number of pixels or measurements. For other signals, the pairs (P_i, Q_i) can simply be the i^{th} element in the input and output streams respectively. The larger the MSE the greater the error. This measure is influenced heavily by anomalies and therefore might not always be the best method to use. However, generally it works well.

Signal to noise ratio

The *signal to noise ratio* (SNR) is defined as

$$\text{SNR} = 20\log_{10}\frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N} P_i^2}}{\text{RSME}}$$

where RMSE=$\sqrt{\text{MSE}}$ (see for example [151]). It basically compares the variance in the signal, expressed here as $\frac{1}{N}\sum_i^N P_i^2$, and the variance in the noise, the MSE. If this value is small it means the variance of the noise is so high that the signal gets lost in the noise. So the higher the SNR, the better the compression.

Peak signal to noise ratio

Next the *peak signal to noise ratio* (PSNR) (see for example [151]). It is defined as

$$\text{PSNR} = 20\log_{10}\left(\frac{\max_i |P_i|}{\text{RSME}}\right).$$

Instead of looking at the variance we look at the maximum point in the signal. If it is an image with pixel values in the RGB code, $\max|P_i| \leq 255$. The PSNR gives a value interpreted simlarly to the SNR.

ℓ_p metrics

The ℓ_p metrics are closely associated with measures such as the MSE. The MSE is simply a form of the ℓ_2 norm. These metrics define penalties by giving some weight to each error of a reconstruction. The different norms have different examples. The ℓ_2 is probably the most common metric used. It emphasises big errors and supresses small errors. When this is not what is desired, the ℓ_1 norm is useful to give similar weights to all the errors. The ℓ_p norm for a vector $f \in \mathbb{R}^N$ is given by

$$||f||_p = (\sum_{i=1}^{N} |f_i|^p)^{\frac{1}{p}}$$

and hence the relative error for the ℓ_p metric can be calculated as

$$\text{Relative error} = \frac{||f - f^*||_p}{||f||_p} = \frac{(\sum_{t=1}^{N} |f_i - f_i^*|^p)^{\frac{1}{p}}}{(\sum_{i=1}^{N} |f_i|^p)^{\frac{1}{p}}}.$$

Mean structural similarity (MSSIM)

The MSSIM was first published in 2004 [172]. Today it is used as a measure of structural inequality between a reference image and some alteration of that image, whether through noise, blurring, or compression. To calculate the MSSIM we first need to divide the image into M blocks and then for each block calculate the structural similarity (SSIM) values, defined as

$$\text{SSIM}(\mathbf{x_j}, \mathbf{y_j}) = \frac{(2\mu_{x_j}\mu_{y_j} + c_1)(2\sigma_{x_j y_j} + c_2)}{(\mu_{x_j}^2 + \mu_{y_j}^2 + c_1)(\sigma_{x_j}^2 + \sigma_{y_j}^2 + c_2)}$$

where $\mathbf{x_j}$ and $\mathbf{y_j}$ represent the corresponding pixels in block j of the images to be compared, $j = 1, \ldots, M$. The means of block j of the images are written as μ_{x_j} and μ_{y_j}, and the standard deviations as σ_{x_j} and σ_{y_j}.

We then calculate the MSSIM as the average of the M calculated SSIM values,

$$\text{MSSIM} = \frac{1}{M}\sum_{i=1}^{M} \text{SSIM}(\mathbf{x_j}, \mathbf{y_j}).$$

2.3 Orthogonal bases and associated transforms

2.3.1 Hilbert spaces

The reader interested in signal processing must understand signals as vectors in Hilbert spaces. To do that we need to define the following, as defined in [118]:

- A vector space.
- An inner product space.
- The property of Cauchy completeness.

Definition 2.7. A vector space, $\mathcal{F}$, is an inner product space if it has an inner product. It implies that $\mathcal{F}$ is a vector space that satisfies the following properties:

1. $\langle x, y\rangle = \langle \overline{y, x}\rangle$(Conjugate symmetry)

2. $\langle ax, y\rangle = a\langle x, y\rangle$
   $\langle x + y, z\rangle = \langle x, z\rangle + \langle y, z\rangle$(Linearity)

3. $\langle x, x\rangle \geq 0$
   $\langle x, x\rangle = 0 \iff x = 0$ (Positive semi-definiteness)

Definition 2.8. A vector space $\mathcal{F}$ has the Cauchy completeness property if every Cauchy sequence f_n has a limit f_{limit} such that $f_{limit} \in \mathcal{F}$.

A simple example to understand the definition of completeness as defined here is the difference between the real-valued sets $(0, 1)$ and $[0, 1]$. In the first case if we pick the sequence $f_n = \frac{1}{n}$, we know that $f_n \xrightarrow[\infty]{n} 0$. Therefore the limit 0 is not contained in $(0, 1)$. Therefore the set $(0, 1)$ is not Cauchy complete. However, every Cauchy sequence in $[0, 1]$ has a limit in the set and therefore it does have the Cauchy completeness property. And so we have the following definition for Hilbert spaces.

Definition 2.9. An inner product space, $\mathcal{F}$ which also has the completeness property, is called a Hilbert space.

2.3.2 Bases, frames and dictionaries

The term "dictionary" has been around in signal processing literature in the last couple of decades, with the first mention of this being in [130]. A dictionary, Ψ, is simply a set of vectors $\{\psi_1, \psi_2, \ldots, \psi_{N_D}\}$. It is not necessary to specify any conditions such as orthonormality, but in doing so we look at specific cases of dictionaries. Given a signal $f \in \mathbb{R}^N$ and that $N_D = N$, we call Ψ a complete dictionary. When the vectors $\{\psi_1, \psi_2, \ldots, \psi_N\}$ are linear independent and it spans the space our signal lives in, we call Ψ a basis. Special cases of bases are orthogonal bases which means that the dot products between all the pairs of vectors in the basis are zero when it is different, and one when it is the same pair of vectors. However, we can represent a dictionary as

$$\Psi = \Psi_1 \cup \Psi_2 \cup \ldots \cup \Psi_{N_D}$$

where $\Psi_1, \ldots, \Psi_{N_D}$ can represent different bases with $N_D > N$. We call this dictionary overcomplete. Nice discussions on dictionaries, as well as overcomplete representations can be found in [43, 68].

Another name for an overcomplete dictionary is a frame. Frames allow for some linear dependence between the set of vectors constituting the frame. We see the importance of being aware of Hillbert spaces in signal processing in the formal definition of a frame, as given in [73].

Definition 2.10. Given that $\Psi = \{\psi_i\}_{i\in\mathcal{I}}$ is a set of vectors such that $\Psi \subset \mathcal{H}$ where $\mathcal{H}$ is a Hilbert space and $\mathcal{I}$ is a countably finite or infinite set of vector indices, then Ψ will be a frame if

$$\lambda_{min}\|x\|^2 \leq \sum_{i\in\mathcal{I}} |\langle x, \psi_i\rangle|^2 \leq \lambda_{max}\|x\|^2$$

for every $x \in \mathcal{H}$, where λ_{min} and λ_{max} give the bounds for the frame [73] .

The frame bounds, λ_{min} and λ_{max} determine how tight the frame is. If $\lambda_{min} = \lambda_{max}$ the frame is called a tight frame. If $\lambda_{min} = \lambda_{max} = 1$ then the frame is known as a Parseval frame. Parseval frames are in fact orthonormal bases. This is a stronger condition than if the frame bounds are larger. The stronger the conditions the better it is for reconstruction. See [37, 46] also for additional explanations of frames and how they are used. According to [76] "Frames can provide richer representations of data due to their redundancy". If we can represent a signal x in terms of a frame Ψ, we know that the information in x is captured by that frame and we will be able to reconstruct the signal from transform measurements as we have not captured the signal's information. This in some sense pre-empts the Uniform Uncertainty Priciple (UUP) which we will discuss in depth in Section 3.3.4.

Bases

We know from linear algebra that a set of vectors $\{\psi_1, \psi_2, \ldots, \psi_N\}$ will be a basis for the subspace $\mathcal{F} \subseteq \mathbb{R}^{N^*}$ if $\{\psi_1, \psi_2, \ldots, \psi_N\}$ are linearly independent and $span(\psi_1, \psi_2, \ldots, \psi_N) = \mathcal{F}$, where $N \leq N^*$ [159]. We concern ourselves with finding a basis for a signal $f \in \mathbb{R}^N$ such that we can write

$$f = \sum_{i=1}^{N} \alpha_i \psi_i = \Psi\alpha$$

where $\alpha = [\alpha_i]$. Generally we want to work with an orthogonal basis, meaning all the basis vectors are orthogonal to each other.

Overcomplete dictionaries

The term "dictionary" is another way to refer to a basis or frame. The use of overcomplete dictionaries was introduced in [68, 43], and later on refined in [2]. We have that $D = [\ d_1 \quad d_2 \quad \ldots \quad d_{N_D}\ ] : 1 \times N_D$ where each $d_i \in \mathcal{D}_I$, and each $\mathcal{D}_I$ can represent a different basis. The reason that dictionaries are used is that it might provide sparser representations of the signals of interest, thus enabling better compression and processing. The representation for our signal becomes the same, $f = \sum_{i=1}^{N_D} \alpha_i d_i$ where $\alpha = [\alpha_i]$ are the coefficients associated with the set of vectors $(d_1, d_2, \ldots, d_{N_D})$. Hastie et al. [93] give an informative description of overcomplete dictionaries.

2.3.3 Filters and filter banks

The concept of a filter is very intuitive and natural. A simple example is that of fog. Suppose someone is playing a piano inside a house, and another person is standing outside in very dense fog. Most of the higher notes will not be audible to the person standing outside the house. That person will hear mainly the lower notes. The reason is simple. From physics we know that the higher notes are waves with higher frequencies than the lower notes, i.e. the higher notes have shorter waves and the lower notes have longer waves. Consequently, the shorter waves get blocked by the fog and the longer notes associated with the lower notes go through. Hence, the person outside the house mainly hears the lower notes when played on the piano.

In Definition 2.11 we define what a discrete filter is.

Definition 2.11. Given coefficients $\{\ldots, h(-2), h(-1), h(0), h(1), h(2), \ldots\}$ we have that the filter H can be built by using the coefficients in such a way that H is an operator that can be applied to the signal of interest. Filters can be designed in different ways depending on its purpose. The number of coefficients in the filter is dependent upon the design of the filter. According to [170] we have the following types of filters:
(1) causal filters - filters where $h(k) = 0 \forall k < 0$
(2) anti-causal filters - filters where $h(k) = 0 \forall k > 0$
(3) Two-sided filters - filters that are neither causal or anti-causal
(4) Finite impulse response filters - filters where $\#\{k : h(k) \neq 0\} < \infty$
(5) Infinite impulse response filters - filters where $\#\{k : h(k) \neq 0\} = \infty$
Filters need to satisfy some important mathematical properties. More can be read in [8].

Filtering occurs when we convolve a signal with the filter. We can write it for the discrete case as

$$y(t) = \sum_k f(t-k)h(k) = f \star h$$

where $\star$ is the operator for convolution.

Importantly we can see that when we do filtering, we use the filter to accentuate some aspects of the signal. Some prominent filters used in signals are what are called high-pass and low-pass filters. It is related to the frequencies in the signal. A low-pass filter blocks out high frequencies and lets the low frequencies pass, similarly to our fog example. High-pass filters do the opposite. A *filter bank* is a combination of these filters being applied to a signal. Exactly like the DWT, which we will see a bit later. Consequently, when using a filter bank we have decomposed the signal into different frequency bands. And these bands can then be analysed separately. Filter banks pass the signal through each filter. Because we do this we produce twice as many coefficients as elements in the signal when we use two filters. From a compression point of view this is bad. This is corrected for using *decimation* [145]. To decimate the coefficients produced by the wavelet filtering (which we look at in Section 2.3.4), we

simply keep every second coefficient and throw the others away. Since we are effectively reducing the rate at which we sample (we are taking less samples after filtering), decimation is also referred to as *downsampling*. For in depth discussion on filters and filtering see [170, 169].

We illustrate what a filter is with a simple example.

Example 2.12. Suppose we have that $f = \begin{bmatrix} 5 & 8 & 1 & 5 & 2 & 9 & 3 & 3 \end{bmatrix}^T$ (we note that the signal need not be one-dimensional). Let the filter matrix be

$$H = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}.$$

Therefore we have that

$$y = Hf = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 8 \\ 5 \\ 1 \\ 5 \\ 9 \\ 2 \\ 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 6.5 \\ 4 \\ 5.5 \\ 3 \\ 1.5 \\ -2 \\ 3.5 \\ 0 \end{bmatrix}.$$

We can easily see that the first four row entries of y are the averages of the rows of the pairs of row entries in f and the second half of the row entries of y are the differences between the pairs of row entries of f. Note that in this case we have in fact applied two filters to the signal f. One for the averages and one for the differences.

Even with a simple exercise as done in Example 2.12, we can see how useful filters can be. If we can reduce a matrix of numbers to averages and differences we have calculated useful information regarding that signal. We will see in Section 2.3.4 when looking at the DWT, specifically using the Haar wavelet, how knowledge of averages and differences are useful when looking at images. In another example we

(a) Grayscale peppers image.

(b) High frequency coefficients after application of DWT using DB8 wavelet.

Figure 2.3.1: Application of DB8 wavelet filtering on peppers image.

apply the DWT with the DB8 wavelet on the peppers image by filtering out all the coefficients, except the initial detail coefficients, which correspond to high frequencies. In Figure 2.3.1 we see the result. One can now see how high frequency changes in pixel values relate to outlines in the image.

2.3.4 Transforms

We want to focus on two transforms that are very important in the signal processing world. Firstly we want to look at the Fourier transform, and secondly at the Wavelet transform. In each case we will discuss the importance as well as the uses of these transforms, and how they work. We will consider the continuous case, to get some understanding of the idea, and then look at the more practical discrete cases.

From a general perspective though, important points to note regarding transforms are the following:

- Transforms are used to gain understanding of signals in a different domain than the one measured. One of the best examples are periodic signals which are measured in the time domain. These can be understood very well in the frequency domain.
- It is sometimes easier to transform a signal, analyse and study the signal and then transform back. This is useful in compression where most of the information of a signal in the transformed domain is concentrated on small sets of real (or complex) numbers. Therefore we can encode only a few of the coefficients, transform back and have a good approximation of the original signal using a lot less memory. In Statistics we also like transformations. The obvious candidate is when we transform a $N(\mu, \sigma^2)$ variable to a $N(0, 1)$ variable since it is easier to calculate p-values for the standardised normal distribution.
- Important to the previous point is the existence of the inverse of the transform that is crucial.
- According to [151] transforms should reduce image redundancy and isolate, for example the various frequencies of the image into bands. This means that a transform should render a useful perspective

on the signal in the transformed domain. Otherwise it would be senseless to transform the signal.

A helpful example given in [151] which illustrates how transforms are useful is Roman numerals and its transformation to Arabic numerals. Let us say we want to calculate a sum of two numbers which are in the Roman numeral domain. It might be easier to transform it to Arabic numerals, do the summation and transform back. For example:

$$\text{MDCCCXXXIX} + \text{CDLXVII} \rightarrow 1839 + 467 = 2306 \rightarrow \text{MMCCCVI}.$$

In [151] Saloman uses multiplication rather than summation. This would of course be more difficult to do in the Roman numeral domain since there is no standard way of multiplying the letters of the Roman numerals. But the summation was sufficient to show the usefulness of the transformation in the above example.

Remark 2.13. Although a transform does not result in the throwing away of information, it does in some sense result in the loss of information because of applying it. We will see that this very fact explains the differences between the Discrete Fourier transform (DFT) and the DWT. If the inverse of a transform exists and we transform a signal from domain A to B and then back using the inverse, we should not lose any of the information of the signal in domain A. However, when we have transformed the signal to domain B, we are constrained by the basis of vectors defining the space we transformed the signal to, and therefore we 'lose' some infomation. We will make this point clearer after we have looked at the DFT and the DWT.

The Fourier transform

The Fourier transform is a well-known tool in the signal processing world. It is named after the French mathematician Joseph Fourier [80]. He made the bold claim that any periodic signal can be represented using sine and cosine waves. The Fourier transform gives the frequency representation of any given signal, not signals only in the time domain. He used it to solve the differential equation relating to change in heat.

First we will look at the Fourier series. Suppose we have a *periodic* signal $f(t)$. We can define it as follows:

Definition 2.14. A signal $f(t)$ is periodic if for some set of p frequencies $\Omega = \{\omega_1, \omega_2, \ldots, \omega_p\}$ it is true that $\forall\, \omega_n$ we have that $f(t) = f(t + W_n)\, \forall\, t$ where each W_n is the period corresponding to the n^{th} frequency present in $f(t)$.

For periodic $f(t) \in L_2^{real}[-\frac{L}{2}, \frac{L}{2}]$ having period L, we have that

$$f(t) = a_0 + \sum_{n=1}^{\infty} [a_n cos(\omega_n t) + b_n sin(\omega_n t)]$$

where $\omega_n = \frac{2\pi n}{L}$.

It can be shown that the values for the coefficients are given by:

$$a_0 = \frac{1}{L} \int_{-\frac{L}{2}}^{\frac{L}{2}} f(t)dt,$$
$$a_n = \frac{2}{L} \int_{-\frac{L}{2}}^{\frac{L}{2}} f(t)cos(\omega_n t)dt,$$

and

$$b_n = \frac{2}{L} \int_{-\frac{L}{2}}^{\frac{L}{2}} f(t)sin(\omega_n t)dt.$$

The coefficients $\{a_n, n \in \mathbb{N}\}$ and $\{b_n, n \in \mathbb{N}\}$ represent the weights of each sine and cosine function used in building $f(t)$. The coefficient a_0 gives the mean value around which $f(t)$ oscillates. This however is restrictive since we can only represent functions on the interval $[-\frac{L}{2}, \frac{L}{2}]$. This is where the Fourier transform comes in. But there is an intermediate step. Instead of using only sine and cosine waves, we can use what is called complex exponentials to represent an even broader class of signals on $[-\frac{L}{2}, \frac{L}{2}]$. We have that

$$f(t) = \sum_{n=-\infty}^{\infty} [a_n cos(2\pi\omega_n t) + i \cdot b_n sin(2\pi\omega_n t)] = \sum_{n=-\infty}^{\infty} A_n e^{i\omega_n t}$$

where

$$A_n = \frac{1}{L} \int_{-\frac{L}{2}}^{\frac{L}{2}} f(t)e^{-i\omega_n t}dt.$$

For the derivation of the complex exponential Fourier series see for example [100]. To get to the continuous Fourier transform, we simply extend the complex Fourier series to the case where $t \in (-\infty, \infty)$. This implies that we can handle non-periodic signals as well. The continuous Fourier transform of a signal $f(t)$, denoted as $F(\omega)$, as well as the inverse, is given by

$$\begin{aligned} F(\omega) &= \int_{-\infty}^{\infty} f(t)e^{-2\pi i\omega t}dt \\ f(t) &= \int_{-\infty}^{\infty} F(\omega)e^{2\pi i\omega t}d\omega \end{aligned}$$

where $e^{-i\omega t} = \cos(\omega t) - i\sin(\omega t)$ and $e^{i\omega t} = \cos(\omega t) + i\sin(\omega t)$. Here we do not have the subscript n since the CWT yields a distribution of frequencies on a continuous spectrum. We interpet $F(\omega)$, similarly to the Fourier series coefficients, to be the contribution of the frequency ω to the signal $f(t)$ over time. We note that we typically work with discrete functions, therefore we have that the DFT, and its inverse for signals of length N are given by

$$F(n) = \sum_{t=0}^{N-1} f(t)e^{-i\omega_n t}, 0 \leq t \leq N-1,$$

and

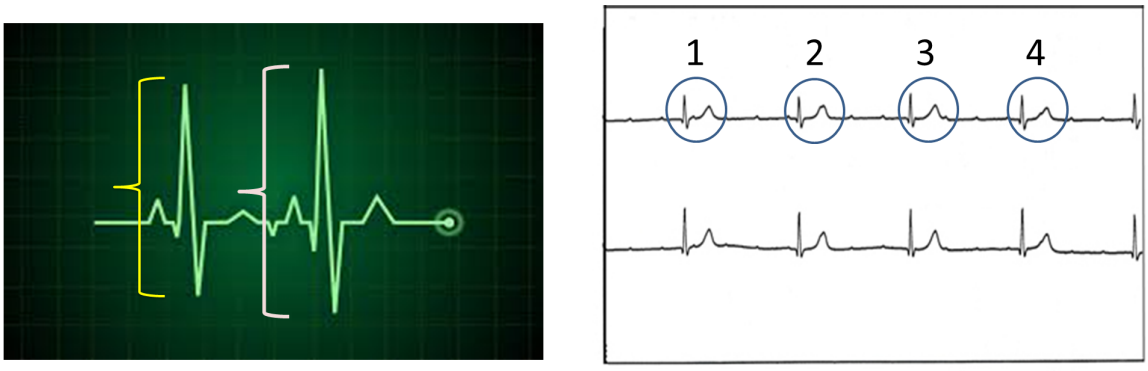
$$f(t) = \sum_{n=0}^{N-1} F(n)e^{i\omega_n t}, 0 \leq n \leq N-1$$

where $\omega_n = \frac{2\pi n}{N}$.

We can therefore see that the Fourier transform decomposes a signal into a sum of sine-waves corresponding to a set of frequencies. If the signal is spread out in the time domain, it will be localised in the frequency domain and vice versa [51]. When doing the DFT, we can't have that the signal is spread out in both domains, or localised in both domains. This is related to an uncertainty principle. The implication of this, is that if we only have one spike at some point in the time domain, then we will have an infinite number of points in the frequency domain since we will be absolutely uncertain about which frequencies are involved in the signal at this one point. And if we have an infinite number of points in the time domain, we will have a single point in the frequency domain. Note also, that if we apply the Fourier transform on a periodic signal bounded over the interval $[-L, L]$, we have calculated the coefficients of the Fourier series for that signal. Finally, an added advantage of using the Fourier transform is that there are fast algorithms to implement the transform, most notably the fast Fourier transform [54].

The Wavelet transform

Although Fourier's transformation is a powerful mathematical tool, it has one disadvantage. The Fourier transform gives us the frequency spectrum of the wave/signal, but isn't able to give us the precise point in time that we observe the frequencies. It only gives us a picture of the global frequencies in the signal.



(a) (b)

Figure 2.3.2: Examples of ECG signals where the DWT is advantageous.

The reason this happens is because with Fourier analysis we use a series of sine and cosine functions to describe the signal $f(t)$. It is not able to pick up *local* subtleties in the frequency spectrum of $f(t)$. In fact, the Wavelet transform is in some sense a generalisation of the Fourier transform. The point is made looking at Figure 2.3.2. In Figure 2.3.2 we see two signals. In Figure 2.3.2a[4] we point out the difference in amplitudes of the two larger waves. In Figure 2.3.2b[5] we point out the small differences in the patterns when it is repeated. For illustrative purposes we assume there is no noise in these cases. The DFT won't pick up the small differences in frequencies and amplitudes, and this is where the Wavelet transform becomes effective.

The Wavelet transform enables us to get not only the frequency spectrum, but the time dimension of the corresponding frequencies as well. We note that there is a way of doing Fourier analysis to obtain a time-frequency analysis. It is called the windowed Fourier transform or short-time Fourier analysis. In [58] Daubechies does a great job of explaining the differences between the windowed FT and the Wavelet transform. We now define what wavelets are.

Definition 2.15. Wavelets are functions, $\psi(t)$ that have the following properties [129]:
(1) $\int_{-\infty}^{\infty} \psi(t)dt = 0$.
(2) The wavelet is localised over t, i.e. $\psi(t) \neq 0$ over bounded region $[a, b]$ where $a, b \in \mathbb{R}$.
(3) $\psi(t)$ is square-integrable, i.e. $\int |\psi(t)|^2 dt < \infty$, in other words, its energy is bounded, not infinite.
(4) If $\psi(t)$ is called the mother wavelet then we have that $\psi_{a,b}(t) = |a|^{-1/2}\psi(\frac{t-b}{a})$ - these are therefore the functions we use as basis for our signal.
(5) Associated with wavelets are scaling functions, $\phi(t)$ which can be derived from the specific wavelet.

The scaling function $\phi(t)$ needs to be smooth and have compact support, similarly to point (2) of Definition 2.15 regarding wavelets. In Figure 2.3.3 we give the Morlet wavelet, and can visualise both points

[4] http://eleceng.adelaide.edu.au/research/sensing-processing/
[5] http://www.theeplab.com/B-The-Members-Center/A000-Electrograms/D-Arrhythmia-Mechanisms/A-Bradycardia/DDA0-Bradycardia.php

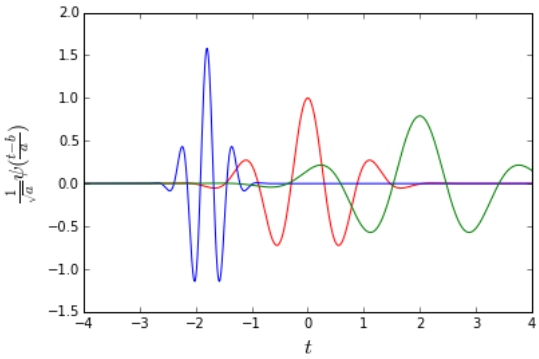


Figure 2.3.3: Morlet wavelet (red) with $(a, b) = (1, 0)$, scaled and re-positioned. Blue function has $(a, b) = (0.4, -1.8)$, and green has $(a, b) = (1.6, 2)$.

(2) and (4) of Definition 2.15, namely that wavelets are localised, and that we can scale and move them. The red function gives us the unscaled wavelet with location at $t = 0$. We scale the wavelet and move to show the effects of the parameter pair (a, b). The red wave is typically called the 'mother wavelet' and the scaled and shifted forms of the mother wavelet are called child wavelets. The functional form of the Morlet wavelet is given by $\varphi(t) = e^{-t^2} \cos\left(\pi t \sqrt{\frac{2}{ln2}}\right)$. The shifting corresponds with the windowed Fourier transform in the sense that when we calculate the transform, we will do it for different values of the pair (a, b). This leads us to the definition of the continuous Wavelet transform (CWT), which is given by

$$W(a, b) = \int_{-\infty}^{\infty} f(t) \frac{1}{\sqrt{a}} \varphi^* \left(\frac{t - b}{a} \right) dt \tag{2.3.1}$$

where a is the scale parameter, b is the shift parameter and $\psi^*(t)$ is the complex conjugate of the wavelet. The Wavelet transform has the advantage over the windowed transform in that we can analyse the signal using scaled versions of the wavelets, whereas with the windowed Fourier transform, the windowing function stays the same, except that it gets shifted. This allows the Wavelet transform to analyse even more sensitive higher frequencies over localised places in the signal.

The inverse of the CWT is given by

$$f(t) \quad = \quad \frac{1}{C_K} \int_0^{\infty} \int_{-\infty}^{\infty} \frac{1}{\sqrt{a}} W(a, b) K \left(\frac{t - b}{a} \right) \frac{dadb}{a^2}$$

where $K(\cdot)$ is a function used for reconstructing $f(t)$ and $C_K = \int_0^{\infty} \frac{\hat{\psi}^*(\mu) K(\nu)}{\nu} d\nu$. The Fourier transform of $\psi^*(t)$ is denoted as $\hat{\psi}^*(t)$. In most cases $K(\cdot) = \psi(\cdot)$ as in Equation 2.3.1. It is not guaranteed that the inverse will exist, however it will only exist when what is called the admissability condition is satisfied

[47]. The CWT is calculated by using a continuous wavelet and multiplying it with our function at a specific point in time. We integrate the resulting function over all time points. We hence have one data point after we have calculated the integral. Then we shift the wavelet and scale it using the parameters a and b. Hence, $W(a, b)$ will for all combinations of a and b give us three-dimensional data. If we for example plot $W(a, b)$ with a on the horisontal axis and b on the vertical axis, we can exactly see where the different frequencies arise in the signal/image. This is conceptually how the wavelet transform gives us not only the frequency dimension of the data, but the time dimension as well.

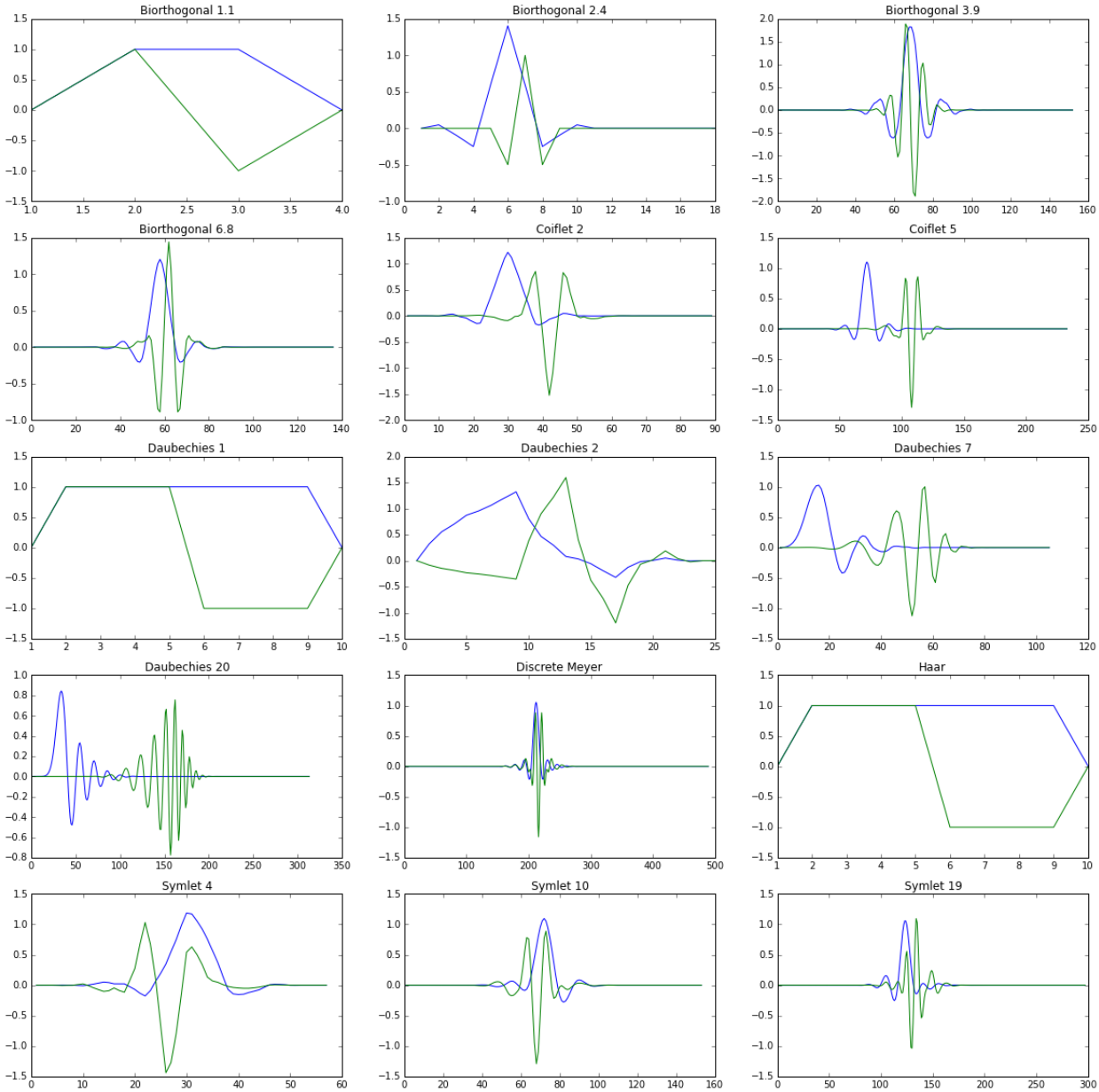


Figure 2.3.4: Plots of different wavelets, $\psi(t)$ (in blue) with their corresponding scale functions, $\phi(t)$ (in green).

We plot different wavelets with its corresponding scaling functions overlayed over each wavelet in Figure 2.3.4. It is clear from Figure 2.3.4 that since wavelets are made to exhibit different shapes it is useful for

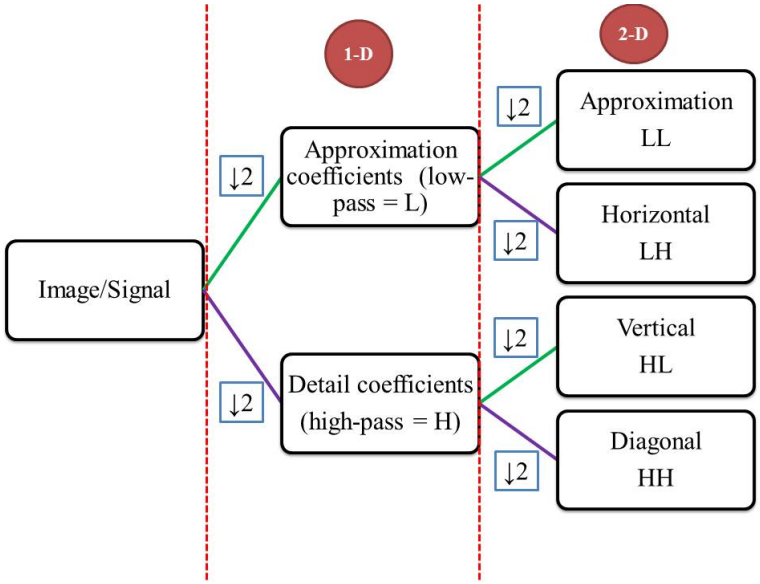


Figure 2.3.5: Schematic representation of Discrete Wavelet transform.

a range of signals.

The Discrete Wavelet transform (DWT)

The DWT is based on using high-pass (G-filters) and low-pass filters (H-filters) [151]. For a one dimensional signal it separates the signal into approximation (low-pass) and detail (high-pass) coefficients. So we apply each filter once to the signal by means of convolution as discussed in Section 2.3.3. We then need to decimate the coefficients. It is necessary with the DWT as well, since convolving $f(t)$ with both the high-pass and low pass filters creates twice as many coefficients as elements in the signal. And since the highest frequency of the coefficients are half that of the original signal, according to Nyquist, we only need half the coefficients produced. For the two-dimensional case, for example where we have an image, the decomposition at each level gives a set of approximation coefficients, and three sets of detail coefficients. The detail coefficients are separated into horizontal, vertical and diagonal detail coefficients. In Figure 2.3.5 we represent both the one and two dimensional cases in the same sketch. The green lines represent the low-pass filtering and the purple lines the high-pass filtering. The decimation is represented by the downarrow followed by the two. The two-dimensional transform is applied in the following way: we apply both filters on the rows of the 2-D signal and then we apply both filters on the columns of the resulting coefficients from the filtering on the rows. We therefore end up with the approximation and the detail coefficients.

The G-filter has the set of coefficients $\{g_0, g_1, \ldots, g_{N-1}\}$, and the H-filter has coefficients $\{h_0, h_1, \ldots, h_{N-1}\}$. We also have the inverse of the filters, which we call G' and H'. The filters are set up in the following way:

- H' is simply the reverse of H, thus $H' = \{h_{N-1}, h_{N-2}, \ldots, h_0\}$.

- G is the same as H', but every odd entry is negative, thus $H = \{-h_{N-1}, h_{N-2}, \ldots, , -h_1, h_0\}$.
- Finally, G' is given as $G' = \{h_0, -h_1, \ldots, -h_{N-1}\}$. It is the same as H, except that every evenly numbered coefficient is negative.

We therefore see a mirror pattern in the filters. More about the theory and design of the filters and how to derive the coefficients can be found in [168] as well as in [58]. To see the parallel between the CWT and the DWT, we note that the low-pass filter relates to the scaling function, and the high-pass filters to the wavelets.

Multi-resolution analysis

Applying this filtering once is in general not how the DWT is used. To create the desired sparsity we apply the transform more than once. This idea comes from what is called multiresolution analysis (MRA). MRA in the context of wavelet analysis was developed by Mallat in the late 1980's [129]. We now look at the DWT from a MRA perspective.

We can think of the different scales as different frequency bands of the signal which we will look at. So note that we will look at scales in discrete steps (in order of 2^j), whereas with the CWT we look at scales on a continuous spectrum.

The most common way of applying MRA in the DWT, is what is called the Pyramid DWT [151]. We apply the same method of filtering described above on the approximation coefficients that resulted from the first pass of the signal (whether one or two dimensional) through the filters. To illustrate how the DFT works we do an example with the 'Haar' wavelet. To do the Haar wavelet transform we need the corresponding filter coefficients. For the Haar wavelet we have that the normalised low-pass filter is $G = \{\frac{1}{\sqrt{(2}}, \frac{1}{\sqrt{2}})$ and the normalised high-pass filter is $H = \{\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\}$.

Per illustration we calculate a four level wavelet decomposition of the function $f(t) = sin(t)$. Note that we plotted 5 cycles of the sine wave where $t \in [0, 10\pi]$. We calculate values for $f(t)$ with intervals of $\frac{\pi}{8}$, implying that we take function values at 80 points. Note that using the 'Haar' wavelet is the same as calculating averages (approximation) and differences (detail) - but we do it in a normalised way. Doing it for four levels results in the following coefficients: 5 approximation, 5 details (level 4), 10 details (level 3), 20 details (level 2) and 40 details (level 1). In Figure 2.3.6 we show the sine wave with its corresponding Haar wavelet coefficients. We see that the first five coefficients are zero. This is the overall average of the sine wave. This makes sense, since the sine wave always crosses over the x-axis. Then we have 5 detail coefficients with value 2.5, then 10 detail coefficients giving a high frequency sine wave with five cycles, and then two sequences of lower range frequencies of 20 and 40 detail coefficients respectively. Note the difference between the representation of the sine wave in the wavelet domain as given in Figure 2.3.6, and the representation in the frequency domain given in Figure 2.1.4. The representation of $f(t) = sin(t)$ only has one point at $f = \frac{1}{2\pi}$ in the frequency domain, whereas in the wavelet domain there are many

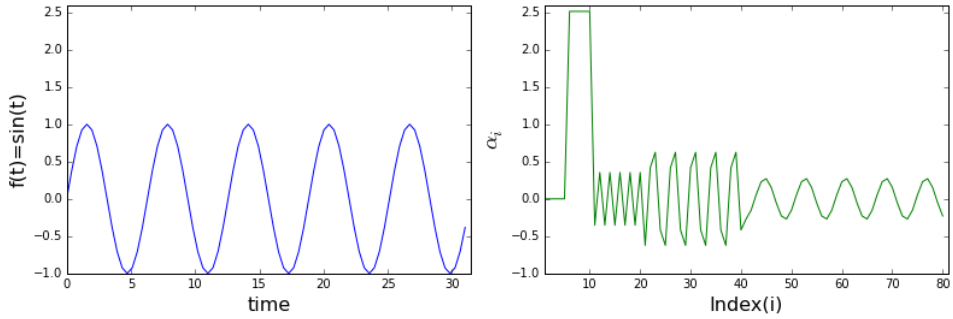


Figure 2.3.6: Sine wave and its corresponding Haar wavelet decomposition

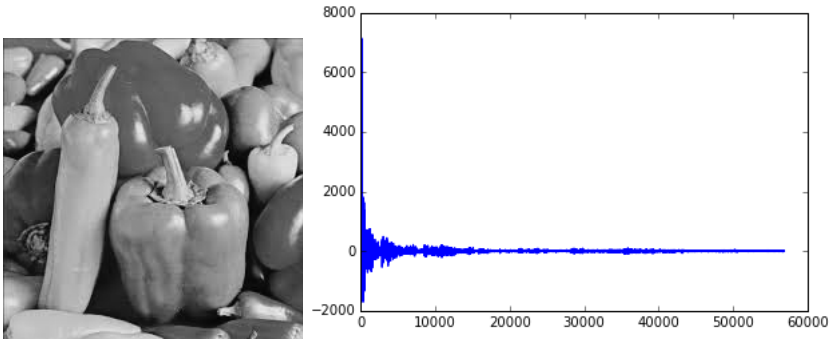


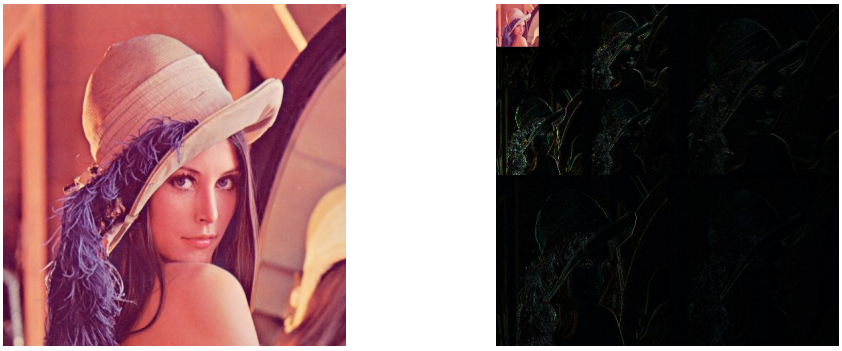
Figure 2.3.7: 'Peppers' image (512 by 512 pixels) and it's corresponding DB4 wavelet coefficients.

more non-zero coefficients. This gives us some insight into when to use the DFT and when to use the DWT. It is therefore not better in all cases to use the DWT instead of the DFT.

We now give a two dimensional example of the DWT. We apply the DWT in two levels on the peppers image using the DB4 wavelet. In Figure 2.3.7 we give the image with its wavelet coefficients. In Figure 2.3.8 we show how the 2D-DWT looks like when we put the coefficients into the blocks corresponding to different scales. We do it for the Lena image using the Haar wavelet. This is called the map of the wavelet decomposition of the image. Note that the wavelet structures in the transform can be chosen specifically to our preferences in aid of compression. However, there are sets of wavelets with known filter coefficients already set up that we can simply make use of. We have shown some of the most commonly used wavelets in Figure 2.3.4. In [58] Daubechies shows theoretically how the Daubechies family is constructed.

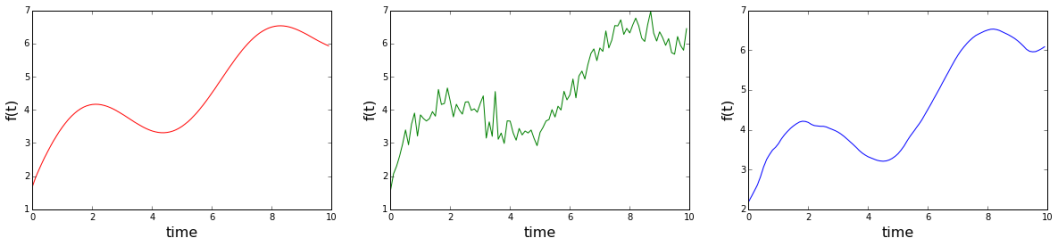
De-noising example

Finally we support the use of transforms in general by de-noising a very simple one-dimensional signal using the DWT. In Figure 2.3.9 we show the original signal, with added noise and then the reconstruction from the noise. The relative ℓ_2 error was 0.02455. However, this is an over simplified example and de-noising becomes more difficult and requires more sophisticated ways of using the DWT to do de-noising



(a) (b)

Figure 2.3.8: 'Lena' image and corresponding Haar wavelet decomposition map



(a) Original signal. (b) Noisy signal. (c) Denoised signal.

Figure 2.3.9: Example of signal added with noise, and the de-noised using the DWT.

[5, 141], especially when applying it on images.

2.4 Summary

In this chapter we discussed some important concepts that any reader wanting to look at CS should at least be aware of. We look at how information works and how important patterns in the data are. Then we consider how to sample signals and the role of compression once the signal is sampled. In the last section we looked at what constitutes signals and how we can use that to our advantage by using different transforms. The Fourier and Wavelet transforms are considered in a bit more detail as they are two of the most important and most widely-used in the signal and image processing communities. In the next chapter we look at CS itself. We consider the main ideas, important literature and discuss the thoery behind the main ideas.

Chapter 3

Compressed sensing

3.1 What compressed sensing is not

In Chapter 2 we introduced data compression, specifically for the applications of image processing. We've looked at some useful concepts that will aid and better our understanding of what is to come. For example the Discrete Wavelet transform (DWT) is fundamental in terms of understanding how transforms work, and why they are useful. Another useful concept is the uncertainty principle, which we will see the significance of later. We also saw why data compression itself is important. We also looked at the sampling of signals and the famous Nyquist-Shannon theorem. Now we can investigate compressed sensing. However, as the name, 'Compressed Sensing' (CS) suggests, it is not to be confused with data compression. Especially looking at surface level, one may at first glance interpret compressed sensing as just another method in the already extensive literature on compression. Therefore, to help us appreciate CS for what it is, and how powerful, and especially, how mathematically wonderful it is, we need to note what CS is not.

Compressed sensing is *not* data compression. In fact, it is in some sense the opposite. Data compression, as discussed in Section 2.2.2 removes any redundancies, or irrelevancies, in order to store the data more cheaply (i.e. use less memory) after the data has been sampled or captured in its entirety. After data compression, the original data is reconstructed, either exactly or approximately, based on the stored compressed data. CS, however, aims to only sample a limited number of measurements of the original signal, in order to reconstruct it later on, exactly.

C. Mohler, an American mathematician specialising in numerical analysis, the founder of MATLAB, has the following to say:

> "When I first heard about compressed sensing, I was skeptical. There were claims that it reduced the amount of data required to represent signals and images by huge factors and then

> restored the originals exactly. I knew from the Nyquist-Shannon sampling theorem that this is impossible. But after learning more about compressed sensing, I've come to realize that, under the right conditions, both the claims and the theorem are true... It is too early to predict when, or if, we might see compressed sensing in our cell phones, digital cameras, and magnetic resonance scanners, but I find the underlying mathematics and software fascinating."[1]

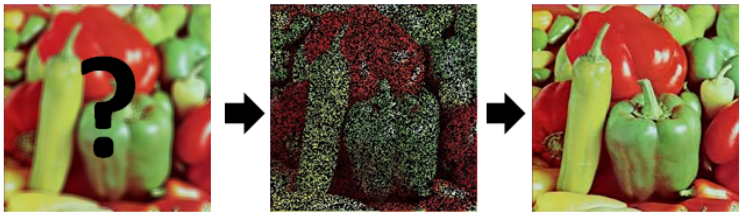


Figure 3.1.1: Process of compressed sensing. We have an unkown image from which we take a limited number of measurements, which is used to reconstruct the original.

In Figure 3.1.1 we very basically illustrate how compressed sensing works. The image on the left represents the unknown signal f (in this illustration the signal is an image) that has not been sampled yet, hence the question mark. The second image illustrates the sample that has been taken of the original signal, such that the only information we have regarding f is the sample, $y = \Phi f$, where Φ is some sensing matrix. The next arrow indicates the process of recovering f from the sample y exactly. The exact recovery of f is based on using the ℓ_1 norm.

A practical and wonderful example from nature is echolocation. Bats use echolocation to determine how their surroundings look like, including what type of animals might be in their surroundings. The aim can be either to determine in what type of environment it is in, as well was what type of insects might be flying around and where it is. The bat makes a sound that sends out some wave to its surrounding. The resultant waves that reflect back to the bat is then used by the bat to determine how its surroundings look like and where food and predators might be. In Figure 3.1.2[2] this is illustrated. This is of course essential to the microchiroptera group of bat since their sight is not as well developed[3]. It therefore relies on this system to find food. And this is a practical example of how CS would be applied in nature, in some sense at least. The bat takes the sampled measurements of the reflecting waves from its environment and reconstructs a picture one could say, of what it should be seeing. Although not perfect, it aids in the direction of where CS is going.

In Section 3.2 we engage the most important literature regarding CS. In Section 3.3 we will discuss the detail of the theory of the illustration explained above.

[1] http://www.mathworks.com/company/newsletters/articles/magic-reconstruction-compressed-sensing.html
[2] http://www.ck12.org/book/CK-12-Physical-Science-Concepts-For-Middle-School/section/5.40/
[3] http://www.batsintheattic.org/blind.html

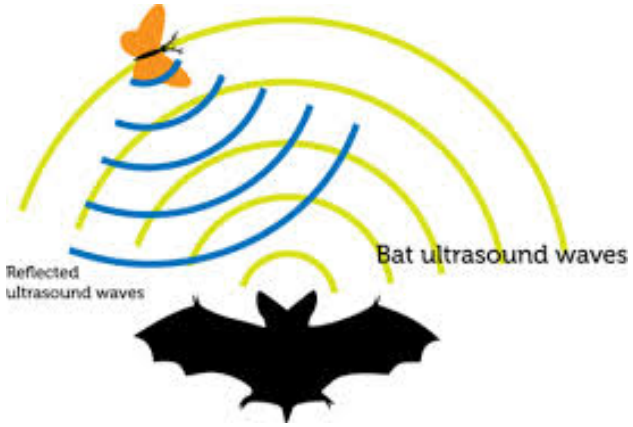


Figure 3.1.2: Graphical illustration of echlocation that bats use to hunt insects.

3.2 Compressed sensing - literature review

In [147] Romberg states that

> "The mathematical theory underlying CS, however, is deep and beautiful, and draws from diverse fields including harmonic analysis, convex optimization, random matrix theory, statistics, approximation theory and theoretical computer science."

This is truly exciting, and we will as we go along, not only do research, but also appreciate the mathematics involved in CS. As Romberg states, CS draws from different fields, and hence the development is in some sense the culmination of those different fields into solving a specific problem, but with general implications. In short, CS looks at aquiring some signal, f, as cheaply as possible, and subsequently recovering the original signal exactly with only the acquired partial information. As mentioned above, this should not be confused with data compression, as it is in some sense the opposite. CS reconstructs the original signal through an optimisation step.

These ideas are founded upon developments and ideas of centuries past, but mainly in the last 50 years or so. In 1795 an algorithm that tackled the problem of estimation from limited noisy exponential samples was proposed [62]. In 1907 and 1911, Caratheodory shows that one can reconstruct the signal given by

$$f(t) \quad = \quad c_1 f_1(t) + c_2 f_2(t) + \ldots + c_k f_k(t)$$

as a positive linear combination where $f_i(t)$ are sinusoids, using $f(0)$ and any other $2k$ points [35, 36]. In [76] it is commented that this proposes a sub-Nyquist sampling framework given some properties of the signal are true (note that the Nyquist theory was not developed at this time).

In the 1970's the idea of using least absolute deviations instead of least squares for a minimisation problem in the context of geophysics, was given as an argument by Claerbout and Muir [48], given certain conditions of the data. For instance it was suggested to use the median in modelling of different kinds of seismic data, instead of the mean as in the least squares case. The reason this was suggested was because seismic data could be affected by different kinds of noise and outliers. According to them, in some applications one needs to use de-noising tools to remove different kinds of noise from the data. In 1986 Santosa and Symes [152] suggested using ℓ_1-minimisation to recover the coefficients of the change in different media underneath the surface. It is relevant because the changes in medium will be sparse, since only at the points where the medium changes from water to sand, for example, will you have significant changes. At the depths where there is no change between medium, for example in a range of depths where there is only one medium, the coefficients of change will be small. Today seismic inversion, based on similar ideas, is a well-known tool used to determine the different mediums underneath the surface. Geophycisists use some kind of 'explosion', typically from an airgun at the ocean surface that represents waves that are sent downwards into the ocean. As the waves bounce back from different media under the surface, the resulting echoes are sampled. This is illustrated with a simple example in Figure 3.2.1.

In the late 80's and throughout the 90's up until the early 2000's, a lot of work was done that prepared the way for the CS discovery. The development of sparse signal recovery played an important role, and it received quite a lot of attention in the aforementioned period. Donoho and Stark extended the idea of the uncertainty principle from intervals to sets of values [69]. This enabled the recovering of sparse signals. In statistics the well-known LASSO also utilises ℓ_1-minimisation to do variable selection [160]. Donoho and Chen developed Basis Pursuit in [43] and showed how it is useful for choosing the best basis to represent the signal of interest given different dictionaries consisting of two bases. In [68] Donoho and Huo show that ℓ_1-minimisation gives the same result as the ℓ_0-norm problem in choosing the sparsest set of vectors from a dictionary that can represent a given signal. Other contributions to sparse representations and uncertainty principles were with respect to union of bases [90], arbitrary redundant bases [83] and uncertainty principles for sparse approximations in pairs of bases [75]. Elad and Donoho extended the work done in [43, 68] by using ℓ_1-minimisation to find vectors that are sparse in more general dictionaries, where the dictionary can contain more than two bases [67]. The Johnson-Linderstrauss lemma [108] is a critical building block for what is called the Uniform Uncertainty Principle (UUP) [33], which followed work by Kashin [112] and Garnaev [85].

Finally, Candes, Tao and Romberg, with Donoho independently, published work that made the ideas of CS concrete. In a surge of articles from these authors [30, 33, 31, 24, 66, 32], CS was developed and introduced. Candes et al. introduced the main result of CS in [30, 33], but only after a peculiar way of Candes and Tao finding their way to each other. It is worth quoting from [128]. The author tells a story of an important event, and one can appreciate the randomness of it.

> "The two mathematicians [Candes and Tao] happened to have children at the same pre-school. While they were dropping them off one day, Candes told Tao about the too-good-to-

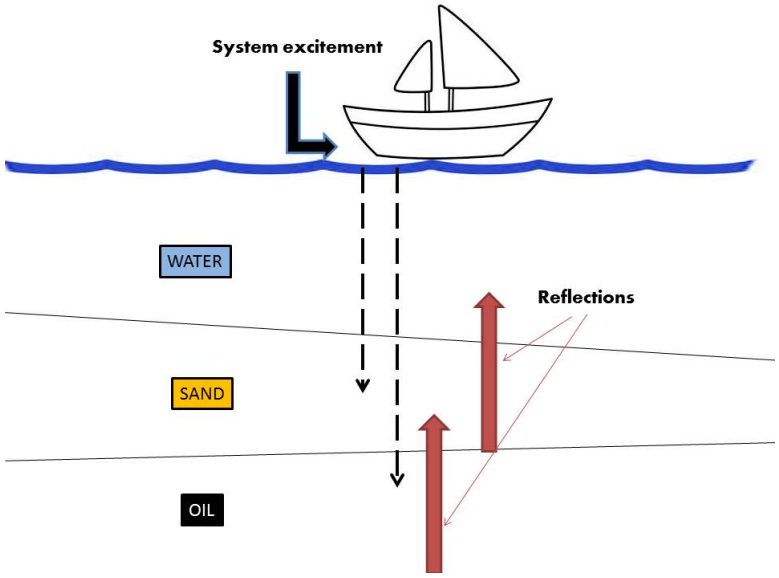


Figure 3.2.1: Graphical illustration of how seismologists attempt to understand structures underground, for example looking for oil under water.

> be-true-reconstructions... "Terry reacted like a mathemetician", Candes continues. "He said, 'I'm going to find a counterexample, showing that what you have in mind cannot be true."' But a strange thing happened. None of the counterexamples seemed to work, and Tao started listening more closely to Candes' reasoning."

Eventually working together, they proved that one can successfully recover a sparse signal from a limited number of measurements of the original signal. In the aforementioned articles it was shown that the mathematics involved holds. In [30] Candes, Tao and Romberg, published the seminal paper which showed that a sparse signal of length N with S non-zero elements can be recovered using $M \ll N$ measurements in the Fourier domain, where $M \sim S\log(N)$. Furthermore, it was showed that using ℓ_1 for sparse recovery is a suitable surrogate in the CS framework, although this was already shown in [32]. In [33] and [32] Candes and Tao introduce the Uniform Uncertainty Principle which provides the condition required of the sensing matrix Φ in order to recover a sparse signal from the measurements $y = \Phi f$. This result is more general than the aforementioned. In the latter it is also shown that $M \sim S\log(\frac{N}{S})$ if Φ is a matrix of random numbers. In [24] they develop the crucial concept called the coherence between two matrices for the CS framework, especially between the measurement matrix and the basis in which the signal f can be represented. The concept of coherence was introduced in [67]. In the above articles different ways of taking measurements are explored and it is proven that CS will only hold if certain conditions are satisfied. The most surprising result regarding the way the measurements should be taken, is that it should be taken completely at random (random linear measurements) for the theory to hold.

In [31] Candes et al. showed that when the measurements are noisy, i.e. contain error, CS still proves to be valid and useful. Similar mathematical conditions hold for the noisy case, as is the case for the non-noisy measurements. This is of obvious importance as most real data will have some noise component.

Candes and Tao developed the Dantzig selector [25], so-called in honour of the linear algebra expert George Dantzig, which is useful specifically when the noise are normally distributed.

In the time after the flurry of articles from Donoho, Candes, Tao and Romberg, other researchers also contributed to some of the main concepts of CS. In [38] Haupt and Nowak compare the performance of CS and active learning, and also confirm that CS is near-optimal, which was shown by Candes and Tao in [33]. Signal recovery when you have noisy measurementsis also considered in [94]. Baranuik et al. gave a simple proof of the fundamental concept of the Restricted Isometry Principle (RIP) in [10]. Many works focus on developing effective and fast algorithms for different kinds of CS problems and contexts [180, 79, 134, 18, 87]. In [71] Baranuik et al. started developing a camera that can apply the CS theory. It is used for different kinds of settings and proves useful in low light.

Since then the world of CS exploded. A legion of articles have been published in the last 8 years or so, contributing to this fast-growing field. Much work has been done on theoretical components of CS as well as on the application side, see for example [161, 16, 137] for some theoretical work. It is clear from looking at the research done where CS is applied, that CS is useful and is contributing in significant ways to the research world, and eventually even the everyday person on the street. In [19] CS has been used to improve the quality of images taken by the Herschell telescope in 2009 - 2013. A lot of work is being done in the improvement of medical imaging, specifically in MRI's [125, 124, 122, 123]. In pediatric MRI's CS has been used to speed up the sampling process by a factor of seven [167]. In [91] CS is applied to audio signals. It is of course natural to expect applications of CS in video, which is done in [65, 81, 42], although it is not trivial. CS is also applied in surface metrology [127]. Another area of application is radar [133, 179, 148].

CS has even been applied in Biostatistics. The Dantzig selector has been shown to be an effective method in gene selection using the Cox proportional-hazards model [6]. However, it is notable from the applications of CS mentioned, that most of the work being done is in the Electronic Engineering world. We aim to answer whether CS is applicable in Big Data and discuss this further in Chapter 4.

3.3 Compressed sensing - theory

We approach the general theory by first beginning with the end in mind - the problem statement and what we want to achieve. After that we consider the challenges that need to be addressed for us to reach our objective. Then we will systematically show how each problem is developed as done in literature.

3.3.1 The problem statement

Today technology has empowered the mathematician and statistician to go beyond what was previously deemed possible, especially concerning processing power and storage capabilities, useful for running com-

plicated algorithms and working with large data sets. However, some data sets are truly too large to be processed in an effective manner. For example the images that were taken by the Herchsel space telescope [19] when it was in operation during 2009-2013. Hence our main problem:

> *Is it possible to take a small, finite set of measurements of our original data and still be able to obtain a good reconstruction thereof?*

Mathematically the problem can be described in terms of a linear system. Let us let the linear system to be of form

$$y = \Phi f$$

where $\Phi : M \times N$, $f : N \times 1$ and hence $y : M \times 1$. We assume f to be a discrete signal in $\mathbb{R}^N$. If $M = N$ then the problem is simple. It is a standard linear inversion problem and if Φ is non-singular, we have that $f = \Phi^{-1}y$. Suppose however that $M < N$. Then the system is called underdetermined and it could have either infinitely many solutions or no solution. CS is concerned with finding an exact and unique solution to an underdetermined system, where the emphasis is on the fact that we have captured only a small number of measurements, M, where $M \ll N$.

3.3.2 The objective

We want to show that it is indeed possible, given that we only have a small sample of an unknown data set to reconstruct that data exactly from that sample. Essentially our objective gives us three areas where we need to provide a solution to different problems that arise. As highlighted above we need to look at:

1. Original data: what properties it should possess; how many and what type of assumptions should we make?
2. Sample: expressed differently, the sample is our observed or *sensed* data. How big should the sample be? How do we obtain it? Can we do something different, or even better and more efficient than the Nyquist-Shannon rate?
3. Recovery: Once we have our sample, how do we reconstruct our data set? Can we recover our original signal exactly?

Now we treat the problems in more detail.

3.3.3 Original data and orthonormal bases

Sparsity of original data

Compressed sensing in large rests upon a critical assumption we make regarding the original signal. We can represent the signal of interest, f in terms of an orthonormal basis in a *sparse* way. So,

$$f = \sum_{i=1}^{N} \alpha_i \psi_i = \Psi\alpha$$

where the ψ_i's represent the different basis vectors and the α_i's the corresponding coefficients, such that we have $\alpha : N \times 1 = [\alpha_1, \alpha_2, \ldots, \alpha_N]$ and $\Psi : N \times N = [\psi_1, \psi_2, \ldots, \psi_N]$. In other words, we can express f as a linear combination of some orthogonal basis where most of the basis coefficients are zero. This is important since signals themselves are typically not sparse, but rather sparse in some basis. That is what we usually mean when we say that a signal is sparse. We have that only a certain number, S, of the α_i's will be nonzero, such that $I = \{i : \alpha_i \neq 0\}$ where $|I| = S$ gives the set of indices of non-zero coefficients. The interesting thing about this assumption is that it is an extremely good assumption to make. It is said and generally known that signals are typically sparse [102]. An example is images (on which our application will focus on). To back up this statement we calculated the sparsity of a group of 51 natural images (see Appendix A) in the wavelet basis. To measure the sparsity we used the Gini Index, given by

$$G = 1 - 2\sum_{i=1}^{N} \frac{|\alpha_{(i)}|}{\|\alpha\|_1}\left(\frac{N-i+0.5}{N}\right)$$

which can be found in [102] where $\alpha = [\alpha_{(i)}]$ is the vector of transform coefficients sorted in ascending order. We have that $G \in [0, 1]$. If G is close to 0 the signal is not sparse at all. This occurs when all the coefficients are the same in absolute value. And the sparser the signal, the closer G is to 1.

We used the 51 natural images and computed their Fourier coefficients, as well as their wavelet coefficients according to the DB4 wavelet. In Table 3.1 we show the descriptive statistics from this exercise. We can see that the Gini Index values are generally above 0.7 for the Daubechies 4 wavelets. For the DB4 wavelet transform we only used one level of the decomposition. Therefore we could have a sparser representation using 3, 4 or even 5 levels. We have shown in Chapter 2 how well we can represent an image with the wavelet coefficients, even if we throw away most of the coefficients.

Coefficients	Mean	Median	Std Dev	Min	Max
Fourier	0.8436	0.8531	0.0610	0.6909	0.9406
DB4 wavelet	0.7629	0.7666	0.0306	0.6572	0.8169

Table 3.1: Descriptive statistics for Gini index values of Fourier and DB4 wavelet coefficients for 51 natural images (see Appendix A).

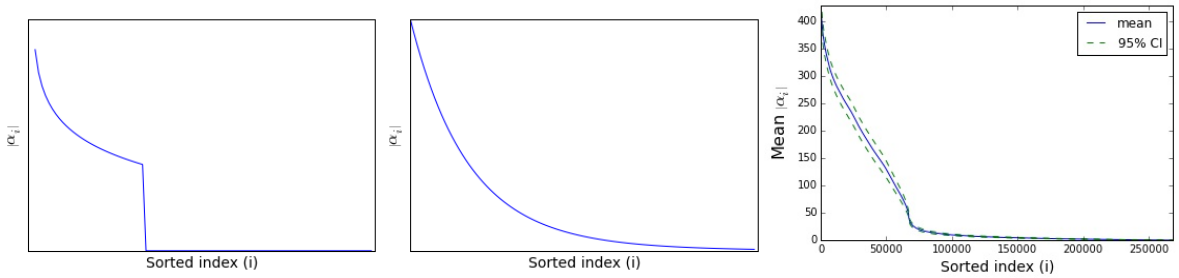
Note that sparsity can be an ambiguous term. In the strict sense we say something is sparse if it contains only a few non-zero elements. It implies that the majority of the elements are zero, not almost zero.

When a signal is nearly-sparse in the sense that it only contains a few significant non-zero elements, it may have a large number of elements that are very close to zero, but which aren't exactly. Sometimes the term sparsity refers to the non-strict definition when it is used loosely, but a nearly-sparse object is called *compressible*. We will follow the strict terminology and will show that CS holds for sparse signals, as well as compressible signals [26].

In Figure 3.3.1 we show the difference between sparse coefficient vectors and the coefficient vectors of typical compressible signals in signal processing. For sparse coefficient vectors the sorted coefficients are large up to say the S^{th} coefficient, and it is zero for the rest. For compressible signals, the coefficients typically obey some power law [26]. The power law implies that the sorted coefficient sequence obeys

$$|\alpha_S| \leq CS^{-S/p}$$

where C is some constant, $S \in [1, N]$ and p a parameter controlling the speed of the decay of the coefficients from large to small. In the CS framework however, it is important that there are about only S large/significant coefficients carrying all the information. The rest are so small that they are not required in order to represent the signal in a satisfactory way. In Figure 3.3.1 we show this as well. We calculated the mean and confidence intervals of the coefficients at each index of the 51 images, where the coefficients are sorted in descending order. Even though we are using only one level of the Wavelet transform with the DB4 wavelet, it exhibits the power law decay.



(a) Typical sparse set.

(b) Typical compressible set.

(c) DB4 coefficients of 51 images, sorted in descending order, represented by the mean and 95% confidence interval of the 51 images at each coefficient.

Figure 3.3.1: Sparsity and compressibility of signals with an example of wavelet coefficients of images.

Sparsity is a property exploited well in the signal processing community. But it is easy to see how many data sets outside the signal processing world have the sparsity property and can be exploited. Consider the claims of insurance companies. Large insurance companies will typically insure themselves, and this is called reinsurance. Let X represent the claim sizes of a specific insurance policy. The reinsurer agrees with the insurer to cover them for some amount if the claims exeed a certain level, say L. Hence, the claim sizes for the reinsurer can be defined as $Y = X - L, X > L$ or $Y = 0, X \leq L$. We expect the claim sizes of the insurer to be below L most of the time. Only a few claims will be above L. Therefore, the claims of the reinsurer will mostly be 0. Hence, the reinsurer will have sparse claims data. The point

is, that there remains an onus on the researcher to recognise signals (or data) where sparsity plays a big role in some way. This will then enable the researcher to properly utilise CS. The application of CS need not be restricted to the signal processing world, where it has its roots.

3.3.4 Sample

Nyquist-Shannon and bandwidth

Now we come to the question about how CS provides a better framework than Nyquist sampling. Nyquist sampling is most effective when $f(t)$ is bandlimited. Candes comments on this in [26]. The bandwidth of a signal described in the frequency domain by the set of frequencies $\Omega = \{\omega_1, \omega_2, \ldots, \omega_N\}$, also called the support of the signal in the frequency domain is $f_{\text{bandwidth}} = \sup(\Omega) - \inf(\Omega)$. Bandlimited means the support of the signal in the frequency domain is concentrated on a connected set of frequencies Ω, where $\min(\Omega) = 0$. In other words, if Ω_1 and Ω_2 are two subsets of Ω then they will have at least one point in common, even if it is a boundary point of the sets. Stated differenty, Ω can't be divided into two non-empty sets which have no points in common. Signals of interest for CS applications are signals with frequency support which is spread out in the frequency domain, where $\text{Range}(\Omega) = \max(\Omega) - \min(\Omega)$ is large, but between $\min(\Omega)$ and $\max(\Omega)$ there are frequencies which are not present in the signal. The support of the latter signal in the frequency domain is not connected, as in the bandlimited cases. Using Nyquist sampling on the signal with sparse frequency support will lead us to use a every high sampling rate. And this is where CS provides a much more effective solution.

Random sampling - posing a few questions

We can see that the Nyquist rate indeed is a powerful tool, but when the frequency support of the signal is not connected, sampling at the Nyquist rate can be extremely difficult, and is extremely wasteful. However, a few questions can be asked:

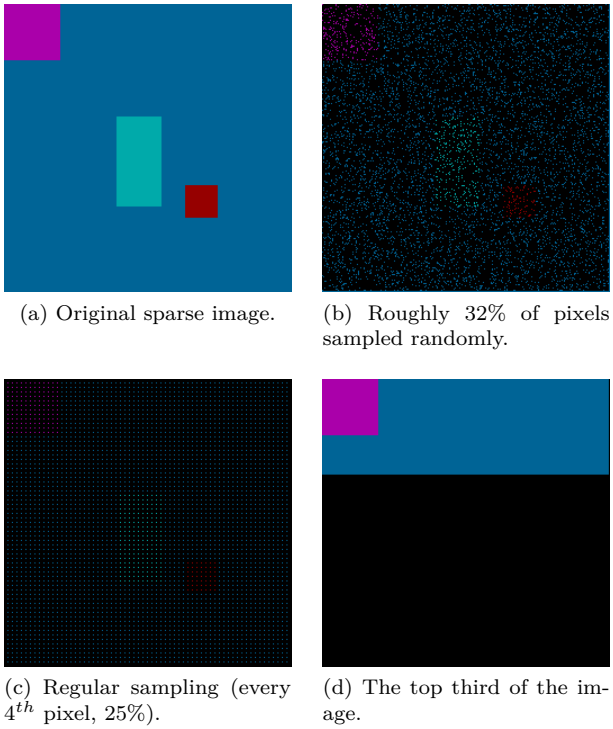
- Is there a way of sampling at even less points in the signal, and still be able to reconstruct the signal exactly?
- Is bandwidth and resolution the only way to go, when we look at a specific (non-arbitrary) group of signals?
- If so, will it be relevant?

How to take the samples in CS

This brings us to another aspect of our playing field. It is that we *assume no prior* information regarding f, not even where the non-zero transform coefficients are, except sparsity. The point to make is that we

will only have a sample, $y = \Phi f$ and therefore this is our only information regarding f we have to work with. This begs the question: can we obtain the original f from y and how will we do it? How do we sample f to end up with y? Digging deeper we should also ask how many measurements do we need if we are going to sample sub-Nyquist?

To answer our first question regarding how we sample, we look at a surprising result developed by Candes et al [30, 33, 26]. The answer is that we should sample randomly. This implies that Φ should be some sort of a random matrix. Or at least a matrix that is unstructured. We illustrate the idea by sampling in different ways from an image of size 256^2 pixels, with three blocks seen in Figure 3.3.2a. We can consider the blue-ish background as values that are zero, hence the pixel values of the image are sparse.



(a) Original sparse image.

(b) Roughly 32% of pixels sampled randomly.

(c) Regular sampling (every 4^{th} pixel, 25%).

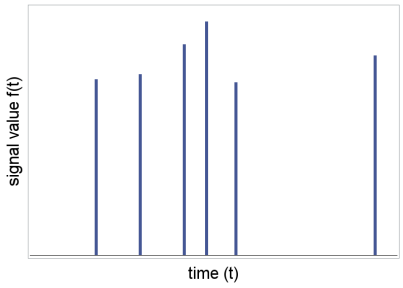
(d) The top third of the image.

Figure 3.3.2: Illustration of different sampling approaches of a sparse image.

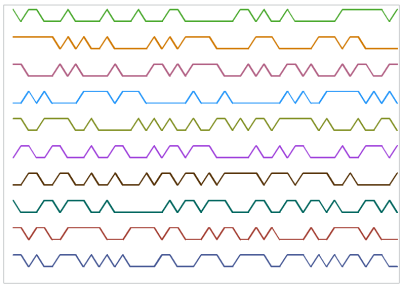
Although we sample at different rates it illustrates the point that when sampling from a sparse signal, which is an image in this case, that some methods are better than other, or more efficient than other. Mainly we see that when we sample regularly (as is similarly done when applying the Nyquist-Shannon sampling theorem) as in Figure 3.3.2c, we seem to capture a lot more information than necessary because the image is sparse. The random sampling, in Figure 3.3.2b seems to hit the important bits of information, at least on average. The sample of the top third of the image in Figure 3.3.2d obviously misses the important information of the green and red blocks. The point is that the choice of the sampling method of the signal is vital. But we note that in this illustration the randomly sampling method has a short fall. What if we do not sample enough significant pixels? For instance, when we sample not enough pixels from the three blocks. It means that we are not necessarily guaranteed of perfect reconstruction whatever reconstruction methods we use. Even worse, if we only sample blue-ish pixels from the background we

will not at all be able to reconstruct our original image with the blocks in it. But CS shows that random sampling works, and it works with high probability [30, 33, 26]. In fact, with such a high probability that the probability of it not working becomes negligible. We need only sample slightly differently. This is where the sensing (sampling) matrix Φ comes in which we will discuss shortly.

So sampling directly from a sparse signal will give us a problem, especially if we sample randomly. When most of the signal elements, or coefficients of the basis expansions are 0, we will end up with a bunch of zeros for most part. If we knew where exactly the non-zero elements (or coefficients) were, we could sample them directly and we would have all we need, and we would have no need to try and solve the aforementioned problem. But the fact of the matter is that when we do the sensing, when we take the sample, we do not know where those coefficients are. A better sampling strategy is to take *global* random measurements of the signal in a linear fashion. By global we mean measurements that take the whole signal into account with every measurement. In Figure 3.3.3a we see a sparse signal in some domain, having only a few non-zero elements (called a delta Dirac function). In Figure 3.3.3b we have noisy/random measurement vectors that we will use to sense the sparse signal. When we take global measurements using the random vectors, we are taking a linear combination of the whole signal for each measurement. The coefficients of the linear combination are given by the values from the random measurement vectors. Therefore, taking global measurements will enable us to gain a little bit of information of the whole signal with every single measurement. This is the crux of CS sampling. In Section 3.3.4 we will deal with the essential mechanics for this to work, most notably what is called the Uniform Uncertainty Principle (UUP).



(a) A signal consisting of randomly spaced delta dirac functions (sparse signal).



(b) Random measurement vectors.

Figure 3.3.3: A better sampling strategy: use global random vectors to sample sparse signals.

But this of course stimulates the following question: if we are taking global measurements aren't we defying the point of CS? Since if $y_i = \langle f, \psi_i \rangle$ then we are looking at the whole signal f each time, implying that we had to sense the whole signal. Firstly, in some cases we only ever see the measurements y. For example in seismology [173, 152] and genetics [184, 177] where we obtain M measurements of a signal that in actual fact should be of dimension N. Secondly, sometimes the number of measurements increases with some parameter, maybe time, and hence we need not wait for N measurements. We simply stop measuring after we have collected M measurements, for example in MRI [123]. And lastly,

practically there is a need to develop hardware that implement the sampling directly, without ever seeing the whole signal. For instance we would need cameras to take only randomly selected pixel measurements, most likely in the Fourier domain. From a matrix point of view, because we take M measurements that are N-dimensional, we will need to work with a matrix $\Phi : M \times N$. Even though after first thought, this truly seems to be a major obstacle in all of what CS is providing a solution for, the researchers in this field seem quite unconcerned regarding this. In the first chapter of [76] Davenport et al. give a short answer as to why no one seems to be troubled too much by this question. It all comes down to clever coding. The algorithms used to sense from the original data and then reconstruct it, is written in such a way that you would never have to store Φ.

Remark 3.1. In the CS framework, the measurements should be taken *non-adaptively*. This means that the matrix Φ is fixed before the measurements are taken. More specifically it means that we do not adapt the way of taking the measurements, as we observe each element of y . According to [66], one would "scarcely" do better by using adaptive measurements, given all we know is that f is sparse in some basis. There has been work done to do adaptive sampling, which seems to give very good results. However, computational complexity is a big drawback of this. In [72] Duarte et al. suggest using a sampling matrix that is learned from a set of standard images and then used to sample. Although it is not strictly adaptive, it does go in the direction of using some other known information to improve the standard way of doing CS.

Remark 3.2. When the signal is sparse in the domain it is measured, the sparsifying basis is given by the identity matrix $\Psi = I_N$.

Remark 3.3. By no means does CS make the Nyquist-Shannon theorem redundant. However, CS provides a more effective solution to sampling and constructing our signal, given that it is sparse and not concentrated in the frequency domain. If the support set in which f can be represented, say Ω is connected, then the Nyquist-Shannon is what we should use. If Ω is not connected and we do not know apriori exactly where the coefficients in the support space are, then CS will be better to use. We also add here that we said we could sample at random locations in the time domain when the signal is sparse in the frequency domain. However, as we saw above, it has to be global measurements. Otherwise CS will fail. Engineers probably do not like the words 'sub-Nyquist'. So for that reader we suggest looking at Chapter 3 of [76] where it talks about CS in the analog-to-digital framework. An article that deals with a specific case of Analog-to-Digital-Converters (ADC's) in wideband signals is [163]. For the interested reader we also suggest looking at [59, 44]. It is a worthwhile topic to investigate further, but this is all we will say for the purposes of this document.

Remark 3.4. When we say that the original signal is of length (or dimension) N it includes images. Although images are two-dimensional in the sense that we have say X rows and Z columns, implying XZ elements, we can represent it as a $N \times 1$ signal, with $N = XZ$ by simply stacking all the columns

on top of one another.

The uniform uncertainty principle (UUP)

In [29] Candes and Romberg state that "In recent years, uncertainty relations have become very popular, in part because they help to explain some miraculous properties of ℓ_1-minimisation procedures". We described what the uncertainty principle is in Chapter 2, but this quote from Candes and Romberg underlines the importance of the uncertainty principle. Now we see how it can be used in the sampling framework of CS.

The sensing matrix, Φ, needs to adhere to the UUP in order for the CS main result to hold. What the UUP essentially implies is that if we want to check if the Φ matrix satisfies the UUP, we should take all possible combinations of $|T| \leq S$ columns of Φ, selected at random, and check whether each one of these new subset matrices are approximately orthogonal. Let us assume $T \subset \{1, \ldots, N\}$, i.e. T is a subset of the integers $\{1, \ldots, N\}$ of size $|T| \leq S$. Then extracting the $|T| \leq S$ columns from Φ, with indices corresponding to the the set T, we obtain the $M \times |T|$ matrix Φ_T. In the case where f itself is sparse, we also extract the sub-vector f_T which are the elements in f corresponding to the index set T. Mathematically speaking, to show a sensing matrix Φ satisfies the UUP we need to find the smallest number δ_S such that

$$(1 - \delta_S) \|f_T\|_2^2 \leq \|\Phi_T f_T\|_2^2 \leq (1 + \delta_S) \|f_T\|_2^2 \tag{3.3.1}$$

holds for all possible sets T. The CS theory holds and exact recovery is possible when $\delta_{2S} + \delta_{3S} < 1$ for any S-sparse vector f [26]. Note that we can test the condition for different levels of sparsity by finding constants δ_R where $R = w * S, w = \{1, 2, 3, ...\}$ such that $|T| \leq R$. In literature, one might come across either the UUP or the Restricted Isometry Property (RIP), since they are the same thing. Also note that $|T| \leq S$. Since we want to capture a signal with only S dimensions that make up the signal, we need not more than S rows of Φ to capture that information, hence $|T|$ doesn't have to be greater than S. However, since we do not know where the significant coefficients lie, we require that Equation 3.3.1 is satisfied for all sets T where $|T| \leq S$. We will see that a property called *incoherence* plays an important role in determining the number of measurements. When f is sparse in the basis Ψ, Equation 3.3.1 becomes

$$(1 - \delta_S) \|\alpha_T\|_2^2 \leq \|U_T \alpha_T\|_2^2 \leq (1 + \delta_S) \|\alpha_T\|_2^2 \tag{3.3.2}$$

where $f = \Psi\alpha$ and therefore $U_T = (\Phi\Psi)_T$. In this case the emphasis is still on the design of Φ. At this point we can see why the UUP is related to uncertainty principles. The energy in the coefficients, which are sparse and spread out in the domain of Ψ, is more or less the same as the energy in the measurements, which are concentrated on a dense set, $\Phi f = \Phi\Psi\alpha = U\alpha$.

The reason we have a condition like this is that we are reducing the dimension of the signal - the original signal is of dimension N and the sample that we have is of dimension $M \ll N$. To recover the original signal from this lower dimensional sample, we need to be sure that we have captured all the significant information of the original signal in the sample. And this is exactly what the UUP does. So we can see that if the UUP is satisfied, then we have that the energy of the high-dimensional signal is preserved in the sample. And hence, when we attempt the recovery of the original signal, we have the necessary information contained in our sample to do so - and importantly, different sparse vectors will be distinguishable in the projected space. This is an important implication of the UUP. The isometry constant determines when the sparse recovery will occur. We can show that sparse vectors will be distinguishable given the isometry constant is small enough. For example, given all sets of columns of Φ of size $2S$ are linearly independent, we know we will be able to distinguish S-sparse vectors when the UUP is satisfied. Let f_1 and f_2 be two S-sparse vectors that give us the measurements y such that $y = \Phi f_1 = \Phi f_2$. Therefore we have that $\Phi f_1 - \Phi f_2 = 0$ which implies that $\Phi(f_1 - f_2) = 0$.

We know that $f_1 - f_2$ will be at most $2S$ sparse (in the case that the non-sparse coefficients are in two mutually exclusive sets of indices in the vectors). We also have that $\Phi(f_1 - f_2)$ is a linear combination of at most $2S$ columns of Φ. And therefore, because every set of $2S$ columns of Φ are linearly independent we know that $f_1 - f_2$ is in the null space of Φ. Hence $f_1 - f_2 = 0 \implies f_1 = f_2$. In this case all S-sparse vectors are distinguishable. This also shows the reason why $M \geq 2S$.

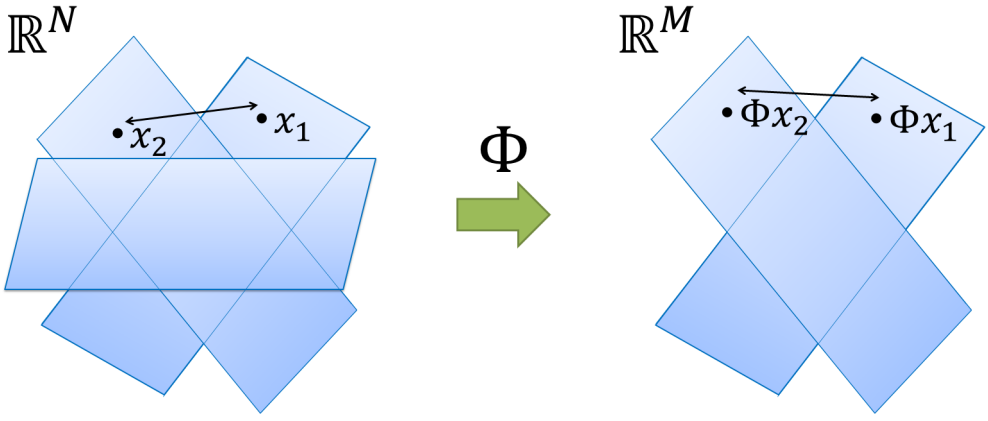


Figure 3.3.4: Illustration of random projection from N-dimensional space to M-dimensional space used in CS, where $M \ll N$.

Let's look at the set of all sparse signals in the domain $\mathbb{R}^N$. If we have two signals in this domain and we project both signals linearly down to $\mathbb{R}^M$, using the matrix Φ, the UUP ensures that

$$||f_1 - f_2||_2 \approx ||\Phi f_1 - \Phi f_2||_2.$$

Hence the UUP requires Φ to be an isometry[4]. This means that if we project two vectors from the N-dimensional space using $\Phi : M \times N$, the two new M-dimensional vectors should be the same distance apart as the original vectors in the N-dimensional space. We can show this geometrically (but in a constrained way though because of dimensionality). In Figure 3.3.4 we show the signals x_1 and x_2 living in the $\mathbb{R}^N$ domain. Note that these signals will be sparse and can live in any of the subspaces along one of the axes in $\mathbb{R}^N$. If we project it down in a linear fashion to $\mathbb{R}^M$ with Φ we can see that the distance between Φf_1 and Φf_2 is preserved. This explains why the UUP is also called the Restricted Isometry principle. It restricts Φ to preserve the distance between to different sparse signals in $\mathbb{R}^N$ and its corresponding samples in $\mathbb{R}^M$. This is why approximate orthogonality of Φ is referred to. The vectors Φf_1 and Φf_2 are approximately orthogonal to each other.

As an example, a matrix that has proven to exhibit the UUP/RIP is a painting by Gerhard Richter called "4096 colours" shown in Figure 3.3.5. Saying that, it isn't an easy task to test whether a certain matrix satisfies the UUP. The reason being is that it is a NP-hard task. It means that as M becomes larger, it would become a cumbersome and infeasible task to check if each of the subset of columns of size $|T| \leq S$ from Φ form a nearly orthonormal system that retains the energy in the original signal. The number of operations required to even check whether a solution satisfies the UUP is $\begin{pmatrix} N \\ S \end{pmatrix}$. It is easy to see that this quickly becomes computationally infeasible. Take for example $N = 2000, S = 700$. We then have that $\begin{pmatrix} N \\ S \end{pmatrix} = \frac{N!}{S!(N-S)!} = \frac{2000!}{700!1300!} = \frac{2000\times1999\times1998\times1997\times\ldots\times1301}{700\times699\times698\times697\times\ldots\times1}$. Consider an image where $N \approx 65000, S \approx 20000$. However, from random matrix theory and the Johnson-Linderstrauss theorem [108] it can be proven that random matrices do satisfy this property. Therefore when we use a random matrix for the sensing matrix we are sure with high probability that we will be able to implement CS successfully.

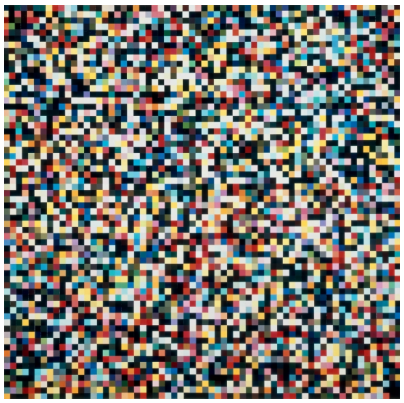


Figure 3.3.5: Richter painting called "4096 colours" that exhibits UUP.

Remark 3.5. We have not given any consideration to some properties that are also important in CS

[4] An isometry is a measure that when applied to some set, in our case the signal, it preserves the distances of the set in the original space [56].

framework and relates to the sensing matrix Φ. These properties are the exact reconstruction principle (ERP) [33], the compressible reconstruction principle (CRP), the adjoint UUP and most notably, the Null Space Property (NSP) [50]. These properties relate to different nuances of when recovery is possible. All the properties mentioned, except the NSP, are described by Terence Tao[5]. We note that if the UUP is satisfied, it implies that the NSP is satisfied, since the UUP is a stricter condition [76].

Summary of UUP

The UUP defines the requirement of the sensing matrix Φ in order for CS reconstruction to be successful. The UUP essentially requires Φ to be such that when we project the signal from $\mathbb{R}^N$ to $\mathbb{R}^M$ the information of the signal of interest is contained in the measurements $y = \Phi f$ and that the measurements of two different sparse vectors are distinguishable.

Incoherence

Another important concept regarding Φ and Ψ is what is called incoherence, defined in [68] and refined for the CS context in [24]. The coherence between Φ and Ψ is defined as

$$\mu(\Phi, \Psi) = \max_{1 \leq i \leq M, 1 \leq k \leq N} |\langle \phi_i, \psi_k \rangle|. \tag{3.3.3}$$

In other words, the coherence gives us the maximum "correlation" between all the combinations of the vectors in both Φ and Ψ. In [29] it is also stated that "The smaller the coherence, the stronger the uncertainty relations". This means that the smaller the coherence the less the two bases look alike, implying the less correlated it is. The greater the difference between the spread in the measurement domain and the concentration in the sparse basis, the lower the coherence. Compressed sensing uses combinations of Φ and Ψ that have low coherence, or stated differently, are incoherent. According to [34] it follows from linear algebra that the bounds of the mutual coherence is given as $\mu(\Phi, \Psi) \in \left[\frac{1}{\sqrt{N}}, 1\right]$. This implies that if $\mu(\Phi, \Psi) = \frac{1}{\sqrt{N}}$ then Φ and Ψ are maximally incoherent, and if $\mu(\Phi, \Psi) = 1$ then Φ and Ψ are maximally coherent, and therefore perfectly "correlated". To see where the lower bound comes from it is easier to re-write Equation 3.3.3 as follows:

$$\mu(U) = \max_{1 \leq i,k \leq N} |u_{ik}| \tag{3.3.4}$$

where $U = \Phi\Psi = [\langle \phi_i, \psi_k \rangle] = [u_{ik}]$. Remembering that Ψ is orthonormal and that the rows of Φ are orthonormal, we have that the rows of U are also orthonormal. Hence we know that $\sum_{i=1}^{N} u_{ik}^2 = 1$. We also

[5] https://www.math.ucla.edu/~tao/preprints/sparse.html

have that

$$\sum_{i=1}^{N} u_{ik}^2 \leq N\{\max_{1\leq i\leq M, 1\leq k\leq N}|u_{ik}|\}^2$$

which follows from Hölder's inequality [99]. Therefore

$$1 \leq \sum_{i=1}^{N} u_{ik}^2 \leq N\{\max_{1\leq i\leq M, 1\leq k\leq N}|u_{ik}|\}^2 \quad \Rightarrow \quad \frac{1}{\sqrt{N}} \leq \max_{1\leq i\leq M, 1\leq k\leq N}|u_{ik}| = \mu(U).$$

The lower bound is shown to be $\frac{1}{\sqrt{(N)}}$ in [68]. The upper bound comes from the fact that if we have maximum coherence, then the maximum values of the inner products between the ϕ_i's and the ψ_k's will be 1, and hence $\mu(\Phi, \Psi) = 1$. The most similar vectors are two of the same vectors. Hence in $\mathbb{R}^N$ $\langle u_i, u_i\rangle = u_{1i}^2 + u_{2i}^2 + \ldots + u_{Ni}^2$. Since we assume u_i is normalised, we have that

$$\sqrt{u_{1i}^2 + u_{2i}^2 + \ldots + u_{Ni}^2} = 1 \Rightarrow u_{1i}^2 + u_{2i}^2 + \ldots + u_{Ni}^2 = 1.$$

In signal processing, the signal in the time domain is typically sampled and analysed via Fourier analysis in the frequency domain. The coherence between the time domain and frequency domain is 1, where $\phi_i(t) = n^{-1/2}e^{i2\pi jt/n}$ and $\psi_k(t) = \delta(t-k)$. So in this case the signal is sparse in the $\Psi = I_N$ basis. In Figure 2.1.4 we have a graph of a sinusoid in the time domain and a Dirac funtion in the frequency domain, which is the representation of the time-domain signal in the frequency domain. One can clearly see the difference in terms of sparsity in the frequency domain, and the spread out signal in the time domain. Another example of incoherent bases are the noiselet [52] and Haar bases. According to [34] the coherence between noiselets and the Haar, DB4 and DB8 wavelet bases are respectively $\sqrt{2}$, 2.2 and 2.9. These values being greater than 1 comes from when we multiply the RHS of Equation 3.3.4 with $\sqrt{N}$ to obtain $\mu(U) \in [1, \sqrt{N}]$. Hence in this case, if it is close to 1, it is incoherent. The coherence is useful in that it provides a computationally feasible way to determine whether CS will work. If the UUP is satisfied, we have a greater guarantee of the CS working. But since we can't always test sensing matrices to see if it satisfies the UUP, we see if the two bases are incoherent [76].

How many measurements should we take?

Candes et al. [33, 30] showed that for CS to hold the number of measurements that need to be taken is

$$M \sim O(S\log(N)),$$

that is, M is a multiplication of S in some linear form, and N in logarithmic form. Intuitively, the

number of measurements, M should also be dependent on $\mu(\Phi, \Psi)$. Since the more coherent Φ and Ψ are, the more measurements we would need to be able to pick up enough information of f with M measurements. Typically in literature, the number of required measurements for CS to work, are expressed as $M > C\mu^2 S \log(N)$ where C is some constant [34]. We see that M doesn't depend that much on N. For example, let us assume $S = 30$ and $M = \mu^2 S \log(N)$. Then we obtain the values in Table 3.2 for M for different values of μ and N and $C = 1$.

μ,N	$N = 100$	$N = 300$	$N = 1000$	$N = 3000$	$N = 10000$	$N = 1000000$
$\mu = 1$	138	171	207	240	276	414
$\mu = 1.4$	271	335	406	471	542	812
$\mu = 1.8$	448	554	671	778	895	1343
$\mu = 2.2$	669	828	1003	1163	1337	2006
$\mu = 2.6$	934	1157	1401	1624	1868	2802
$\mu = 3$	1243	1540	1865	2162	2487	3730

Table 3.2: Minimum number of required measurements, M, for $S = 30$ and different pairs of (μ, N) (values rounded off to integers). We assume $M = \mu^2 S \log(N)$.

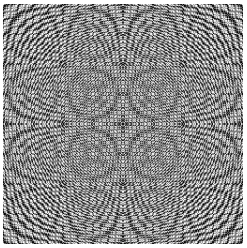
From Table 3.2 we can see that as N increases it has a smaller effect on the coherence μ. Although the values of M increase by about 3 times for each value of μ as N increases from 100 to 1000000, it does not increase proportionally with the N. The conclusion therefore is that the number of measurements we will need for a given signal, will depend on the inherent structures of Φ and Ψ and not on the size of the signal.

Random ensembles

As was said, the sensing matrix Φ we need to use to ensure that the theory of CS stands, is a random matrix, or otherwise called a random ensemble. In [30, 26, 66] a few examples of random ensembles are given:

- Gaussian ensemble: one can use a matrix with i.i.d. normal random values with zero mean and variance $\frac{1}{M}$. Typically the number of measurements required when we use a Gaussian ensemble is such that $M \sim O(S\log(N/S))$.
- Binary ensemble: the matrix is filled with values that are either $\frac{1}{\sqrt{M}}$ or $-\frac{1}{\sqrt{M}}$, where $P(\phi_{ki} = \pm\frac{1}{\sqrt{M}}) = \frac{1}{2}$. Candes and Wakin give two reasons these noiselets are often used in [34]. Noiselets have low coherence with the data of sparse images and algorithms that use them are very fast. In Figure 3.3.6 we give an illustration of noiselets. Details on noiselet matrices and how to construct them can be found in [52, 164]. We can see that it will be incoherent with a sparse domain. The number of measurements required to take are in the same order as that of Gaussian measurements.

- Fourier ensemble: the random matrix is obtained by uniformly sampling M rows of the $N \times N$ discrete Fourier transform matrix taken from the original signal. The columns of Φ need to be



(a) Noiselet matrix with $N = 2^8$ rows and columns.

(b) A sample of the noiselet matrix by randomly picking $M < N$ rows.

Figure 3.3.6: The noiselet ensemble.

normalised. In this framework we therefore observe Fourier measurements. It is only proven for the random Fourier ensemble that $M \sim O(S\log N^6)$ [33], although it is conjectured that $M \sim O(S\log N)$. An example of this is used in the phantom example in [30]. Therein Fourier coefficients are taken along 22 radial lines in the $2D$ Fourier plane. In this case $\Phi = \Omega\Theta_{Fourier}$ where Ω is a mask that helps us take measurements along the radial lines, and $\Theta_{Fourier}$ the Fourier transform. The Fourier transform produces global measurements since that is how the transform is performed (see Section 2.3.4). In Figure 3.3.7 Ω is presented.

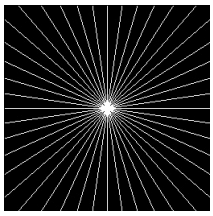


Figure 3.3.7: Mask used to sample along radial lines in the Fourier domain of an image.

- Incoherent ensembles: this is in actual fact a generalisation of the previous mentioned ensembles. As already referred to, we can write $y = \Phi\Psi\alpha = U\alpha$ where $U = \Phi\Psi : M \times N$. In this case it is proven that $M \sim O(\mu^2 S\log N^4)$ [33, 149].

3.3.5 Sparse recovery

Mathematically we want to solve for f, given that we only have $y = \Phi f$, where f can be represented sparsely in the orthonormal basis Ψ by the transform coefficients α. Therefore we can write

$$y = \Phi\Psi\alpha.$$

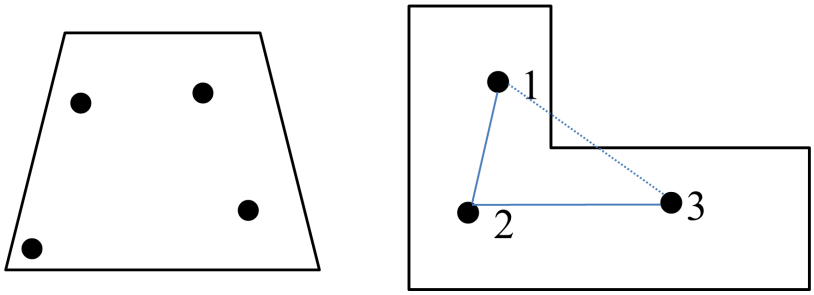
The problem is now that we have an underdetermined linear system (more parameters than equations). Candes, Tao and Romberg [30, 33] suggest and show successfully, that ℓ_1-minimisation effectively obtains a unique and exact solution for α when Φ satisfies the UUP. Note that we rewrite the model as $y = \Phi\Psi\alpha = U\alpha$ where $U = \Phi\Psi$ when $\Psi \neq I_N$.

The unconventional, convexional way and a problem regarding the norm

The crux of the solution is that the programme looking for the solution should be operating over *convex* sets, with a *convex* function. So we have the following two definitions.

Definition 3.6. [20]
(a) A set C is a convex set given that if x and y are two points in C, and $0 < \theta < 1$, then the line segment $z = \theta x + (1 - \theta)y$ is also in C.
(b) A function, $f : \mathbb{R}^N \to \mathbb{R}$ is a convex function if we have that $f(\theta x+(1-\theta)y) \leq \theta f(x)+(1-\theta)f(y) \forall x, y \in$ **dom**(f)[6].

From Definition 3.5 we have that if two people are two points in a convex room, they would be able to see each other no matter where in the room they stood. We illustrate graphically an example of a convex and a non-convex set in Figure 3.3.8a and 3.3.8b. We see that the points 1 and 3 in the non-convex set can't be connected by a straight line since it goes out of the set. In Figure 3.3.9a we see a convex function. It is convex because no line between two points of the function, have higher values than the function. A function that is neither convex or concave is shown in Figure 3.3.9b. Here we have that the line at the top connects two points (A and B) in such a way that function values from the line are higher than the function value at point B. This leads us into the definition of convex optimisation.



(a) Convex set. (b) Non-convex set.

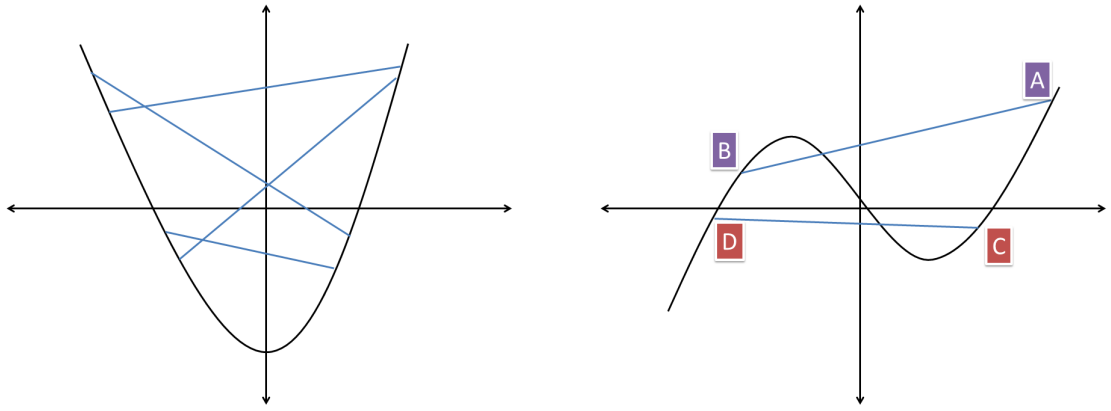
Figure 3.3.8: Convex and non-convex sets.

Definition 3.7. A convex optimisation problem [20] is a problem such that we minimise $f_0(x)$ subject to

$$f_i(x) \leq 0 \quad , \quad i = 1, 2, \ldots, m \quad \text{and} \quad a_i^T x = b_i \quad , \quad i = 1, 2, \ldots, p \text{ where } f_i(x) \text{ are convex functions } \forall i \ .$$

We know that in Statistics an optimal solution to finding the parameters of a statistical model in linear regression is well-defined. We minimise the ℓ_2-norm of the errors, and obtain a nice expression for the regression coefficients, given by $\alpha = (\Phi'\Phi)^{-1}\Phi' y$. This is a very specific convex optimisation [20] problem and we can write it as such:

[6] **dom**(f) is the set $D = \{x : f(x) \quad exists\}$



(a) Convex function

(b) Function that is neither convex or concave.

Figure 3.3.9: Convex and non-convex (non-concave) functions.

$$\min_{\beta} \|y - \Phi\alpha\|_2^2$$

where in fact we are minimising the sum of squares for error. In the case where we have two parameters, we are minimising $S = \sum_i (y_i - \hat{y}_i)^2 = \sum_i (y_i - \hat{\alpha_0} - \hat{\alpha}_1\phi_i)^2 = \|y - \Phi\alpha\|_2^2$. The function and set over which we minimise are both convex and linear, and we have no constraints. Can we therefore use it in the CS framework?

We noted in Section 3.3.1 that in the CS framework this is not a proper solution because the number of rows of Φ, M, is significantly smaller than the number of rows of α, N, and it does not pick out sparsity. We ellaborate on this below. A more natural and intuitive measure for sparsity is the ℓ_0-norm. The ℓ_0-norm is defined as $\|\alpha\|_0 = \sum_i |\alpha_i|^0$. It is simply a count of the number of non-zero elements in α, therefore it makes sense that this would be a good measure of sparsity. The convex problem associated with this can be written as

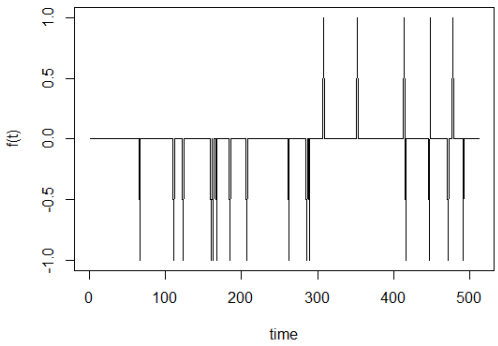
$$\min_{\alpha} \quad \|\alpha\|_0 \quad \text{s.t.} \quad y = \Phi\alpha.$$

This is indeed what we are looking for, but using the ℓ_0-norm for the optimisation gives a NP-hard problem. Candes et al. and Donoho in different work [30, 33, 68] proposed then to use the ℓ_1-norm. It was showed that the ℓ_1-norm is a sufficient surrogate for the ℓ_0-norm in certain conditions. Therefore, we can write the problem as

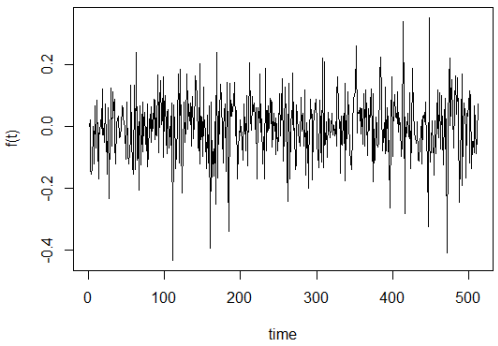
$$\min_{\alpha} \quad \|\alpha\|_1 \quad \text{s.t.} \quad y = \Phi\alpha.$$

Now we consider the difference between the ℓ_1 and ℓ_2 norms and why it is that the least squares approach does not work.

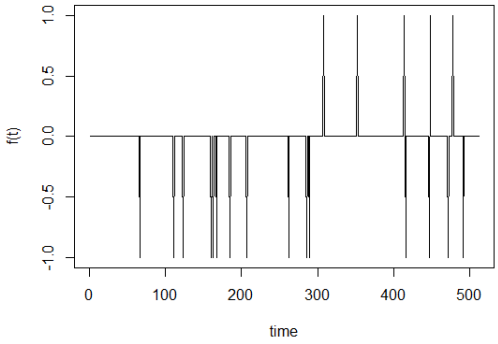
Comparison of ℓ_1 and ℓ_2 norms



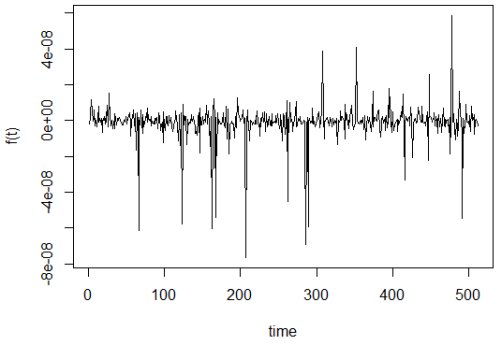
(a) A signal consisting of $S = 20$ spikes of -1 or 1 and 0 elsewhere.



(b) Reconstruction using minimum energy reconstruction (ℓ_2).



(c) Exact reconstruction using ℓ_1



(d) Errors from ℓ_1-reconstruction.

Figure 3.3.10: Comparison of least squares and ℓ_1 reconstruction of a sparse signal ($N = 512$) with only $S = 20$ non-zero elements.

We give a simple example where we compare least squares (using the pseudo inverse of Φ in the solution of ℓ_2 minimisation) and CS reconstruction (using ℓ_1 minimisation) of a sparse signal of length $N = 512$, say f, which has $S = 20$ spikes of 1 or -1 and is zero elsewhere (Figure 3.3.10a). We then take $M = 120$ measurements. The least squares solution is given by $f^* = \Phi^\dagger y$, where we have the pseudo-inverse $\Phi^\dagger = \Phi^T(\Phi^T\Phi)^{-1}$. In Figure 3.3.10b we plot the estimated solution for the sparse signal using least squares which has approximation error $||f - f^*||_2 = 3.8443$. It clearly does not give us what we want. In Figure 3.3.10c we have the ℓ_1-reconstruction of the signal, f^*, and the corresponding errors in Figure 3.3.10d. We can see the errors for the ℓ_1 reconstruction are very small; the approximation error is $||f - f^*||_2 = 2.3024 \cdot 10^{-7}$. It is not exact as the iterations were specified to stop once a certain error level was obtained. We discuss the algorithms used to obtain the reconstruction in Section 3.3.8. A geometric

perspective on this shows us why ℓ_1 works better than ℓ_2 in the reconstruction, illustrated in Figure 3.3.11.

Let us assume that our signal lives in $\mathbb{R}^2$ and therefore we have $\alpha = [\ \alpha_1 \quad \alpha_2\ ]^T$. The horizontal axis will represent values for α_1 and the vertical axis values for α_2. A sparse α would then contain one zero coefficient, either α_1 or α_2. The other coefficient would then be non-zero. Both coefficients can't be zero, since then we won't have a sensible solution since the reconstructed signal will then just be equal to a vector of zeros.

To construct the graphs, we set both norms equal to s, say $s = 1$. We plot the resulting square, representing the ℓ_1 ball, and circle, representing the ℓ_2 ball in Figure 3.3.11.

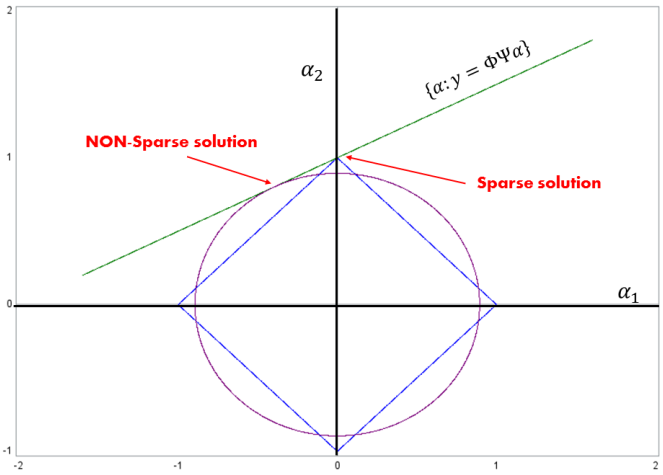


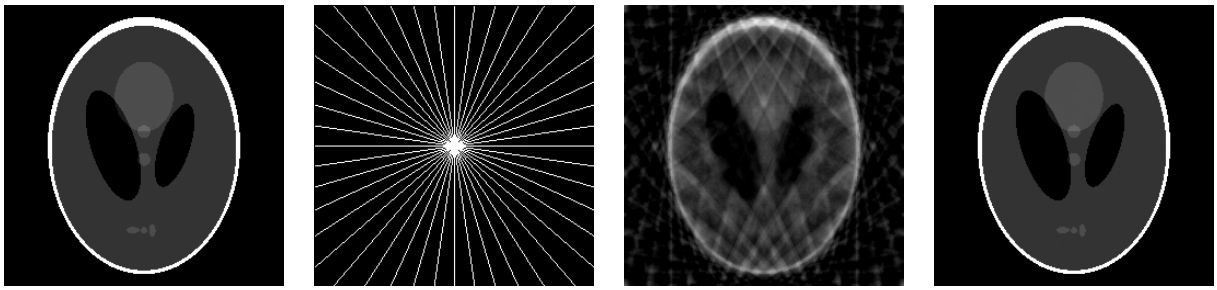
Figure 3.3.11: Representation of ℓ_1 and ℓ_2 and why ℓ_1 gives the sparsest solution

Another component seen in Figure 3.3.11 is the straight line, which represents the set of all possible vectors α that would result in giving the sample $y = \Phi f$, which we can write as $\{\alpha : y = \Phi f\}$. The aim of this convex optimisation routine is to find the sparsest possible α that give us the sample y. In Figure 3.3.11 the line representing the set that could give us the sample, it intersects the ℓ_1-norm at the point $\alpha = [\ 0 \quad 1\ ]^T$. We can see that if we grew the square representing the ℓ_1- norm from the origin up to the first point it intersected with the straight line, it would be at the intersection point just given. The solution that the ℓ_1 gives is sparse, as $\alpha_1 = 0$. Looking at the unit sphere representing the ℓ_2- norm, we see that the same straight line intersects with the sphere at point where neither of the coefficients are zero. If we were to grow the sphere from the origin, we see that the first point at which it would intersect the straight line is not sparse. Therefore, the solution ℓ_2 gives is something not sparse. And as Romberg, explaining this in [147] says, this non-sparse solution becomes even more non-sparse in higher dimensions. This serves to show geometrically why ℓ_1 is the preferred way to obtain the sparsest possible solution. The main reason why the above happens is because the ℓ_2 norm is what is called isotropic and the ℓ_1 norm is anisotropic. Isotropic means that distance from the centre is everywhere exactly the same - hence "iso"-tropic, as in the ℓ_2 sphere. Since it is a circle we know that it has a radius, and therefore

each point on the boundary line is the same distance from the centre of the circle. This is however not the case with the ℓ_1 'square'. If we take for instance the distance from the centre to the corners of the square, which are one of four coordinate pairs, and compare with any of the other points and its distance to the centre, we see it is not the same. Hence, when we blow up the square from the centre, until the first intersection of the sparse line, it will always meet one of the sparse points first. One may wonder what happens if the line representing the set of vectors that could have given us y, is parallel to one of the axes. Because of the random nature of the projection this will not occur, at least not with any significant probability.

Example: phantom image in MRI

In this example we used code from ℓ_1-magic[7] from Romberg again to do the standard CS on a phantom MR-image. This is also an image the authors use a lot, for example in [30]. In Figure 3.3.12 we give the results. In Figure 3.3.12a we have the original phantom image, in Figure 3.3.12b the linemask used to sample from the Fourier coefficients, in Figure 3.3.12c the minimum energy reconstruction (that is the ℓ_2 reconstruction) and then finally we have in Figure 3.3.12d the reconstruction using ℓ_1 minimisation. One can easily see that the ℓ_2 doesn't give a good approximation, whereas the ℓ_1 minimisation gives exact recovery!



(a) Original phantom image.

(b) Fourier sampling along radial lines.

(c) Image recovered using ℓ_2 recovery.

(d) Perfectly reconstructed image by minimising the total variation of the image.

Figure 3.3.12: A CS example using the Fourier ensemble to sample from a phantom image and total variation minimisation since the image itself is sparse.

3.3.6 Main result

Given all that is said, we summarise it in the following result:

Given that f is sparse in some orthonormal basis Ψ, if we solve the following convex programme,

[7] https://statweb.stanford.edu/~candes/l1magic/

$$\min_{\alpha} \|\alpha\|_1 \quad \text{subject to} \quad y = \Phi\Psi\alpha \tag{3.3.5}$$

we will obtain an exact solution for α , given that Φ satisfies the UUP in such a way that $\delta_{2S}+\delta_{3S} < 1$. The isometry constants δ_{2S} and δ_{3S} correspond to the UUP when we let $|T| \leq 2S$ and $|T| \leq 3S$ respectively. See Section 3.3.4. When Φ is a random matrix, we obtain exact recovery with probability $1 - \epsilon$, where ϵ is very close to zero.

We basically go and look for the sparsest vector α that is consistent with our data, y. This problem can be solved in a number of different ways, which we describe in Section 3.3.8. This is because of the fact that Φ satisfies the UUP.

Note that further:

- The basis of sparse representation can be an overcomplete dictionary, given the UUP is still satisfied. For more on that we refer the reader to [27]. We consequently write

$$y \quad = \quad \Phi f = \Phi D\alpha$$

  where $D = [\ \Psi_1 \quad \dots \quad \Psi_d]\ : N \times N_D$, $\alpha = [\ \alpha_1 \quad \dots \quad \alpha_d]^T\ : N_D \times 1$ and $N < N_D$.

- CS theory has been extended to handle noise, which we discuss below in Section 3.3.7.

When the signal is compressible, the reconstruction error between f and the reconstructed signal, f^*, from optimising the problem in Equation 3.3.5 is given by

$$\|f - f^*\|_2 \quad \leq \quad C\frac{\|f - f_S\|_1}{\sqrt{S}} \tag{3.3.6}$$

given that $\delta_{3S} + \delta_{4S} < 2$ and C some constant where f_S is S-sparse signal that best approximates our compressible signal, f, in $\mathbb{R}^N$ [26]. If $\delta_{4S} = 0.2$ then $C \leq 8.77$.

Important points to note are:

- Sampling in CS is non-adaptive and linear.
- When sampling the only a priori information is that the signal f is sparse, either itself, or in some basis, not necessarily a specific basis.
- The recovery process is highly non-linear, namely it is convex in nature.
- The number of measurements required is near-optimal, in the sense that one would not be able to do better, given that we only know that the signal f is sparse in some domain (the knowledge of which domain is also not known a priori) and we do not know where the significant coefficients are.

3.3.7 Introducing noise to the sampling

In [26] Candes states the following,

> "This [robust compressive sampling] is a very mportant issue because any real-world sensor is subject to at least a small amount of noise. And one thus immediately understands that to be widely applicable, the methodology needs to be stable. Small perturbations in the observed data should induce small perturbations in the reconstruction."

It is extremely important for CS to be robust to noise, because as Candes himself said in talks he gave at Cambridge in 2011[8], if CS is not, the theory will be redundant and we can just as well throw it away. Fortunately Candes et al. showed in [31] that even when noisy measurements are made, CS will still be effective and relevant. In this case we do not get exact recovery, but approximate recovery as would be expected. The main result of CS stated above will subsequently change. We now have our measurement vector y to contain noise, i.e. $y = \Phi f + e$, such that $||e||_2 \leq \varepsilon$. Therefore, we solve the following convex programme:

$$\min_{\alpha} \|\alpha\|_1 \quad \text{subject to} \quad \|y - \Phi\Psi\alpha\|_2 \leq \varepsilon. \tag{3.3.7}$$

The aim is now not to have exact recovery, since it is not possible, but *stable* recovery. This means, that as Candes stated, when small changes in our original data happens, for example when noise is added, the reconstruction should not be affected significantly. However, in our situation where we only take a limited amount of measurements we would expect the noise to have a great impact on our reconstruction. Since our measurement matrix Φ is $M \times N$ where $M \ll N$, we have an underdetermined system to solve. Somehow the matrix inversion keeps the perturbations from blowing up according to [26].

[8] nuit-blance.blogspot.co.za/2011/03/cs-lms-invited-lecturer-series-2011.html

In [31] Candes et al. provide two bounds for the cases where signals are corrupted by noise, and are compressible rather than sparse in the strict sense. When the signal is corrupted by noise we have that the reconstruction error is given by

$$\|f^* - f\|_2 \leq C_{1,S}\varepsilon + C_{2,S}\frac{\|f - f_S\|_1}{\sqrt{S}}$$

for some ε (seen in Equation 3.3.7) and the constants $C_{1,S}$ and $C_{2,S}$ which according to [31] are well behaved constant depending on δ_{4S}. It is the sum of a term proportional to the measurement error ε and the error from approximating the compressible signal when no noise is present (see Equation 3.3.6). According to [31] the constants are given $C_{1,S} \approx 12.04$ and $C_{2,S} \approx 8.77$ if $\delta_{4S} = 0.2$.

3.3.8 Different algorithms for the recovery

We briefly look at three categories of approaches to finding the sparsest α that fits our sampled measurements $y = \Phi f$. The three categories are:

- Convex optimisation
- Greedy algorithms
- Combinatorial algorithms

Needell and Tropp give a concise and accurate comparison of these three categories of algorithms [134]. In general convex programmes are computationally burdensome although don't need many samples, while combinatorial methods can "require a large number of somewhat unusual samples that may not be easy to acquire" but are very fast. Greedy algorithms accomodate for both of these disadvantages and fit in between the other two methods when it comes to number of samples and the speed of the algorithm. We do not look at combinatorial methods, since it is not as widely used in literature for CS. We refer the reader to [86, 88, 104, 55].

Greedy algorithms take a completely different route from convex optimisation. They use iterations through which they find their way to a sparse vector that approximates the true vector satisfactorily. Instead of trying to recover the signal that gave the measurements y, it builds up the original signal step by step by making optimal choices at each step. Amongst these are the Orthogonal Matching Pursuit (OMP) [162, 142], Regularised Orthogonal Matching Pursuit (ROMP) [135, 136], CoSamp [134], Stagewise Orthogonal Matching Pursuit (StOMP) [70] and Normalised Iterative Hard thresholding (NIHT) [18]. Recently an extensive comparison of some of the prominent greedy algorithms was done in [17].

We discuss convex optimisation in more detail now.

Convex optimisation

We have already described the use of convex optimisation to solve the CS problem in Section 3.3.5. Here we provide a bit more detail for application purposes. Initially, in 2005, Candes and Romberg made their algorithms using convex optimisation available at `http://statweb.stanford.edu/~candes/l1magic/`. The package is called "ℓ_1-magic", and is done in MATLAB.

In "ℓ_1-magic" different algorithms are described for different cases. Some problems can be recast as linear programmes (LP's) and others as second order cone programmes (SOCP's). These problems are under the umbrella of convex optimisation problems. They are tailored according to the type of function, which is optimised over the type of set, given that in both cases they are at least convex. The LP's are solved using primal-dual methods and the SOCP's using log-barrier methods. All these can be read about in [20].

We now give some of the problems.

1. Min-ℓ_1 with equality constraints. This convex formulation is given by

$$\min_{\alpha} \|\alpha\|_1 \quad \text{subject to} \quad y = \Phi\Psi\alpha. \tag{3.3.8}$$

2. Min-ℓ_1 with quadratic constraints. The convex formulation is given as

$$\min_{\alpha} \|\alpha\|_1 \quad \text{subject to} \quad \|y - \Phi\Psi\alpha\|_2 \leq \epsilon. \tag{3.3.9}$$

   Therefore in this case we look for the α that is the sparsest, such that the residual $r = y - \Phi f = y - \Phi\Psi\alpha$ is smaller than some error ϵ (in the ℓ_2 sense), which is specified by the user.

3. Min-ℓ_1 with bounded residual correlation. The convex formulation is given as

$$\min_{\alpha} \|\alpha\|_1 \quad \text{subject to} \quad \|\Phi^*(\Phi f - y)\|_2 \leq \gamma \tag{3.3.10}$$

   which implies that we are looking for the α where the residuals are the least correlated with the measurement matrix Φ.

Instead of minimising the ℓ_1 norm of the coefficients, the total variation norm can be minimised. The TV-norm is defined as

$$||\alpha||_{\mathbf{TV}} = \sqrt{(D_{h;i,j}\alpha)^2 + (D_{v;i,j}\alpha)^2}$$

where

$$D_{h;i,j}\alpha = \begin{cases} \alpha_{i+1,j} - \alpha_{i,j} & i < n \\ 0 & i = n \end{cases}$$

and

$$D_{v;i,j}\alpha = \begin{cases} \alpha_{i,j+1} - \alpha_{i,j} & j < n \\ 0 & j = n \end{cases}.$$

Another way the sparsest coefficients can be solved for is to solve a LASSO type problem with a sparsity penalty on the coefficients when minimising the sum of squares.

Problem 3.3.8 can be solved using LP's. Problems 3.3.9 and 3.3.10 can be solved using a SOCP. But the precise form of which convex programme to use will depend firstly on the type of signal you are working with, and secondly the performance you are satisfied with. All of these can be investigated in a very popular convex optimisation book by Boyd and van den Berghe [20]. The linear programmes that Romberg and Candes use in ℓ_1-magic are not necessarily the best in terms of speed. That was merely the beginning of algorithms used for CS.

Finally we note that the formulation of the optimisation problem will be different for different applications, but still within the ℓ_1-minimisation framework.

In Chapter 4 we use the following algorithms to implement CS:

- LASSO-FISTA [13]: FISTA is an acronym for "Fast Iterative Soft Thresholding Algorithm". It is a very fast approach to solving the LASSO problem. It is useful for large-scale implentations. The optimisation problem is given as

  $$\min_f \|y - \Phi f\|_2^2 \quad \text{subject to} \quad ||f||_1 \leq t$$

  where t is a tuning parameter. The LASSO is simply the least squares problem with a penalty on the regression coefficients in a ℓ_1 sense.

- Basis Pursuit with de-noising and Douglas-Rachford iterations (BPDNDR) [74, 43]: Basis Pursuit Denoising (BPDN) solves a problem equivalent to the LASSO. The optimisation formulations is given in what is called Lagrangian form by

  $$\min_\alpha \frac{1}{2}||y - \Phi f||_2^2 + \lambda||f||_1.$$

where λ is a regularisation parameter. The equivalence between the LASSO and BPDN depends on the relationship between the parameters t and λ.

- Dantzig selector with a primal-dual algorithm (DSPD) [25]: the Dantzig selector is introduced earlier in this section. The primal-dual algorithm is one of the methods to solve the convex optimisation problem that is also discussed and used in the "ℓ_1-magic" package.

3.4 Summary

Compressed sensing is an exciting field that involves many wonderful aspects. We have looked at how it exploits sparsity in signals, whether it be the signal itself, or coefficients in some basis to recover that signal from a limited number of random measurements. The advantages of this framework are the following:

- The CS framework provides a stable way of acquiring the signal of interest, such that reconstruction is possible if signals are noise or compressible [31].
- The conditions on the sensing matrix makes CS universal, in other words it does not matter what the basis is we represent the signal in, as long as it is sparse [11].
- The measurements are global, meaning every measurement contains information about the whole signal [147]. Therefore, if we lose some of the measurements we will still be able to recover the signal if we take enough measurements.

When implementing we need to decide how to design the sensing matrix in practical applications, but as was shown there are sensing matrices shown to satisfy the UUP that we can use for given situations. Given our problem, we also need to decide which algorithm we will use to do the sparse signal recovery. These things depend on the application we are working with. We focused on ℓ_1-minimisation, but as was mentioned there are specific algorithms that might do a good enough job for a given application.

We have discussed the mathematics behind CS in this chapter. To allude to the example of the bat. The sensing mechanisms of the bat is extremely sophisticated. It is not simple and straightforward. And neither is the application of the theory of CS. The application of CS of course will depend heavily on the context which it is being applied in. Therefore designing sensing matrices (systems) and the use of the appropriate basis to represent the signal requires careful consideration and a very good understanding of theory of CS, as well as the theory and application in the context it is applied in. That said, the challenge is definitely being posed to researchers as to how CS can be successfully implemented, and there is enough reason to do so because of the immense potential of CS.

An application of CS where the theory is definitely years ahead of the practical application is in imaging. Although theoretically it would be better to sense less pixels in an image, to build a camera to do this is

not feasible when comparing to the current way of doing it. The use of silicone to build the sensors, which is inexpensive, makes it difficult for CS to penetrate this area because of how expensive it is to build the necessary hardware to implement CS in cameras. As was mentioned earlier work has been done on what is called a single-pixel camera [71]. Although it is not necessarily going to replace the everyday camera, there is potential in doing imaging in frequencies of light outside the range of normal human sight using the single-pixel camera which is feasible according to [71]. This fact leads to some not sharing the global enthusiasm of CS. According to [155] a big drawback of CS is that technology in a lot of cases, e.g in image processing, do not support the mathematics and hence CS is made out to be something 'nice', but not practical.

However, CS generally has been accepted by top researchers at well-known institutions. See for instance Hastie et al. [93]. It shows that CS is not an idea that will simply fly-by. As the last decade has passed, the literature on CS has developed rapidly, in both the theory of CS and practical applications of CS. We proceed now to discuss Big Data with CS in Chapter 4.

Chapter 4

Application

4.1 Big Data and compressed sensing

In Chapter 1 we discussed the phenomenon called Big Data and argued that it is indeed something worthwhile investing research in as it influences our daily lives in many ways, regardless of how we look at it. There we defined Big Data and concluded that Big Data and the analysis thereof, is fundamentally about harnassing new value from data, and that is the reason why it is worthwhile looking at. In Chapter 3 we presented compressed sensing (CS). It has become ubiquitous in the signal processing community and is influencing even the world around us. Our main purpose in this chapter is to propose some potential ways in which CS can be useful in the Big Data world and apply CS to some images. We have already started to answer this question in Chapter 1 referring to data acquisition, sparsity of data, data where $M \ll N$ and convex optimisation. CS can be applied with one of three focuses:

- Firstly, the focus can be on sensing our data in a compressed way, since storing the whole is wasteful, or is strainful on hardware. In [64] anomaly detection is done in the compressed domain, where random projections are used to project the data to a lower dimensional space. The spectral clustering technique [110] is used to cluster a data set by looking at compressive measurements [182]. Davenport et al. [60] develop some of the first steps in using compressive measurements to solve problems such as detection, classification and filtering.

- Secondly, it is because we are constrained by the data itself, yet we still want to recover the original data, for instance in genomics where we only get to see a limited number of measurements. The data itself constrains us, and the measurements are already compressed. In these cases we would like to recover the parameters of significant gene expressions as accurately as possible. The emphasis here is mainly on the recovery part.

- Thirdly, is because of both the first two reasons. We are constrained by hardware and therefore do

not want to take as many measurements, but we still want to recover the original. A good example would be in MRI scans, where we take a limited number of measurements in order to speed up the process, and we need to recover the original data as we need it for the medical analysis.

As we mentioned in Chapter 1, Big Data Analytics is problem specific, rather than necessarily a science which poses multiple challenges which need to be solved. We already argued that CS will be useful in Big Data since Big Data is generated in many applications by the use of sensors and devices. Some recent work that will be useful in Big Data are the followng:

- Sensor applications: In [4] the authors propose a CS application in accelerometers of mobile phones, which are inertial sensors. They address the problem of battery consumption when sending information from different sensors of the phone over the mobile network. This work implements CS to sense the sensor data on the mobile, send it over the network and then reconstruct it at the receiver end. Although this is applied on a phone, when it is applied to many phones simultaneously this can potentially address the large amounts of data being sent over mobile networks. In [115] CS is proposed as a solution for the problem of limited energy consumption from nodes spread across some space which limits the amount of data that can be captured at the nodes. The solution is applied to, for example, forest surveillance, as well as temperature and light measurements taken from the ocean. Mobile cloud sensing [92] is a very recent development which can also employ CS in order to reduce the burden of sensing too much data. Mobile cloud sensing combines sensor data from mobile phones and cloud computing in order to give a variety of outputs useful to the end-user of the phone.

- Social media: Compressed sensing is used to enhance topic detection in [132] from Twitter. First, topics from Twitter data are identified. Then compressed sensing is used to find sparse vectors of either ones or zeros which correspond to a topic. This speeds up the process of classifying words.

- Databases: Another application of CS in a real Big Data setting is called Impression Store [181]. Impression Store utilises CS in a database context. It stores data in the database in compressed form. Whenever a query is asked it reconstructs the data and gives an estimate of the answer to the query. The data can also be updated where it is compressed again.

- Genomics: In [116] they suggest using ideas from compressed sensing in order to reduce the computation time in classifying the taxonomy of individual samples from humans, the environment or other factors. The idea is to decide whether a specific sample belongs to a specific class. A vector containing zeros and ones are found from an optimisation step that indicates whether a specific individual specimen belongs to a specific class. This is done for all the classes.

- Outlier Detection: Detecting outliers in Big Data has received attention in [120]. According to [178] service quality assurance, fraud detection, and novelty discovery are some of the applications where outlier detection is important. Therein it is proposed to use of CS in data sketching, part of

the process to ultimately reduce communication time between different nodes. This is done within a proper Big Data framework using Hadoop to manage the data.

Although CS has been utilised very effectively in the signal processing community, the above examples suggest that CS will be useful in the Big Data context as well. Some of these examples are in fact signal processing applications in a Big Data setting. Besides being suitable to the Big Data problem, CS will need to be scalable. A grant[1] was awarded to a team from University College London for 2015 - 2018 of more than £700,000. The aim of the project funded by this grant is to develop algorithms in the CS context so that it can be used in a parallel processing way. The desired impact is ambitious. According to the team the immediate impact will be in MRI technology. They also expect to impact the SKA-project referred to in Chapter 1.

In this document we focus on image processing applications. The question is: how do image processing problems and Big Data relate? Consider Facebook which needs to store billions of photos. According to a few of their infrastructure engineers, Facebook has stored 15 billion photos up until 2009[2]. Since they store four copies of each photo, they actually store 60 billion photos, which take up around 1.5PB of storage. At that time 220 million photos were being stored weekly. Facebook has had to be clever in how they deal in storing these photos, and also clever in how they help the Facebook user get access to these photos. This shows that images are part of Big Data. In this case, it is specifically more from a storage, and therefore, database point of view. A possible application of CS might be in how these images are stored. The challenge will be in reconstructing the images quickly enough that the end-user doesn't have delays in obtaining their images. In [84] CS is used to do deep learning for image classification. Deep learning is a crucial tool in addressing Big Data problems [45]. Projects such as the SKA will benefit from the use of CS because of the massive amounts of data that will be generated. Generally, images form a large part of the unstructured data of companies. Big Data problems arise when these images are used alongside other sources of data, for example, text documents, log files and emails. In these cases the challenge is to utilise the data sources together for business value. Medical imaging is a Big Data application of CS that has already shown to be successful for implementation, especially in MRI [146, 123, 78].

We therefore argue that CS should be considered as a tool in Big Data applications because:

- Big Data tends to be sparse.
- Big Data often challenges storage capacity.
- Big Data can challenge the physical capacity of devices, for example the energy consumption of mobile batteries by mobile sensors as referred to above.
- Big Data often comes in the form of images, and CS has shown to be very useful in the image

[1] http://gtr.rcuk.ac.uk/projects?ref=EP/M011089/1
[2] https://code.facebook.com/posts/685565858139515/needle-in-a-haystack-efficient-storage-of-billions-of-photos/

L (number of radial lines)	M(number of complex Fourier measurements)	$100 * (M/N)$
75	36932	14.09
100	48604	18.54
125	59930	22.86
150	70939	27.06
175	81628	31.14
200	91984	35.09
225	102014	38.92
250	111711	42.61

Table 4.1: Number of measurements used for iterations of CS over 51 images

processing context.

4.2 Application of compressed sensing to images

As an exhibition of CS we apply it to 51 images chosen from different databases[3,4,5] (see Appendix A) in order to have some variety in the images chosen. We used code of Starck et al., who implemented the code to obtain for their results in [157]. Specifically we use the *"splitting solvers"* and *"Wavelab850"* set of codes which is freely available on Stark's website[6]. In the code three different optimisation approaches are used: the LASSO with FISTA [13], Basis Pursuit with Douglas-Rachford iterations (BPDNDR) [74, 43] and the Dantzig selector with a primal-dual algorithm (DSPD) [25], which we discussed briefly in Section 3.3.8. The images are sampled in the Fourier domain and the Haar basis is used as the sparsifying basis. We iterate over the 51 images, without and with noise, for eight different levels of the number of measurements. Each image is $512 \times 512 = 262144$ (N) pixels, or elements. The number of measurements is determined by the number of radial lines that is used to sample in the Fourier domain (as in the phantom example in Section 3.3.5).

The eight levels of number of measurements are provided in Table 4.1. The lowest percentage of measurements is 14.09% and the highest 42.61%.

For the noisy cases we defined the noise as :

$$\varepsilon \quad = \quad \sigma_y \cdot 10^{\frac{-\text{SNR}}{20}}$$

where σ_y is the standard deviation of the measurement vector y. An SNR of 30 was used based on work in [157]. That implies that for each iteration with noise standard deviation σ_y will be different. Therefore ε differed for every image and number of radial lines, L (see Table 4.1). We used the relative error in

[3] http://decsai.ugr.es/cvg/dbimagenes/g512.php
[4] http://sipi.usc.edu/database/database.php
[5] http://www.imageprocessingplace.com/root_files_V3/image_databases.htm
[6] http://www.multiresolutions.com/sparsesignalrecipes/software-NEW.html

$M/N(\%)$	Minimum	Maximum	Mean	Std Dev	Median	Quartile Range
14.09	0.0500	0.3430	0.1630	0.0651	0.1541	0.0958
18.54	0.0388	0.3312	0.1406	0.0653	0.1279	0.0760
22.86	0.0343	0.2798	0.1229	0.0583	0.1138	0.0721
27.06	0.0283	0.2472	0.1096	0.0551	0.1005	0.0628
31.14	0.0254	0.2530	0.1006	0.0529	0.0900	0.0538
35.09	0.0233	0.2219	0.0905	0.0480	0.0825	0.0508
38.92	0.0225	0.2038	0.0803	0.0413	0.0714	0.0429
42.61	0.0202	0.1905	0.0726	0.0393	0.0652	0.0403

Table 4.2: Summary statistics of relative errors (ℓ_1 norm) for the LASSO-FISTA optimisation algorithm for 51 images for different number of measurements (indicated as percentages of N), for the noisy cases.

$M/N(\%)$	Minimum	Maximum	Mean	Std Dev	Median	Quartile Range
14.09	0.0489	0.3454	0.1583	0.0630	0.1468	0.0858
18.54	0.0386	0.3031	0.1355	0.0583	0.1260	0.0731
22.86	0.0337	0.2773	0.1201	0.0562	0.1114	0.0707
27.06	0.0278	0.2428	0.1059	0.0508	0.0998	0.0597
31.14	0.0249	0.2354	0.0967	0.0476	0.0899	0.0510
35.09	0.0224	0.2187	0.0871	0.0437	0.0819	0.0464
38.92	0.0215	0.2019	0.0788	0.0406	0.0698	0.0401
42.61	0.0190	0.1872	0.0717	0.0390	0.0655	0.0399

Table 4.3: Summary statistics of relative errors (ℓ_1 norm) for the Basis Pursuit with Douglas-Rachford iterations optimisation algorithm for 51 images for different number of measurements (indicated as percentages of N), for the noisy cases.

ℓ_1-norm to measure the approximation error for each reconstruction attempt. A relative error should be as close to zero as possible. We also calculated the MSSIM, as defined in Section 2.2.4 to measure the structural similarity between the reconstructed images and the original images. A good reconstruction corresponds to a MSSIM value close to one.

Looking at Tables 4.2, 4.3 and 4.4 we can see that the relative errors decrease as the number of measurements increase. The Basis Pursuit algorithm performed the best when considering the relative errors in the ℓ_1 norm since for all values of L it had the lowest minimum, mean and median relative errors compared to the LASSO-FISTA and Dantzig selector results. The MSSIM values increase as the number of measurements increase, which we see in Tables 4.5, 4.6 and 4.7. This is of course expected.

$M/N(\%)$	Minimum	Maximum	Mean	Std Dev	Median	Quartile Range
14.09	0.0495	0.3477	0.1587	0.0632	0.1472	0.0881
18.54	0.0387	0.3014	0.1355	0.0583	0.1260	0.0728
22.86	0.0345	0.2780	0.1201	0.0560	0.1132	0.0713
27.06	0.0283	0.2429	0.1062	0.0509	0.0999	0.0602
31.14	0.0255	0.2349	0.0972	0.0478	0.0892	0.0523
35.09	0.0232	0.2179	0.0873	0.0436	0.0815	0.0477
38.92	0.0225	0.2009	0.0791	0.0404	0.0699	0.0405
42.61	0.0201	0.1896	0.0723	0.0389	0.0650	0.0397

Table 4.4: Summary statistics of relative errors (ℓ_1 norm) for the Dantzig selector with the primal-dual algorithm for 51 images for different number of measurements (indicated as percentages of N), for the noisy cases.

$M/N(\%)$	Minimum	Maximum	Mean	Std Dev	Median	Quartile Range
14.09	0.2622	0.8255	0.5375	0.1374	0.5494	0.2120
18.54	0.3322	0.8647	0.6015	0.1318	0.6252	0.1992
22.86	0.3903	0.8925	0.6549	0.1239	0.6759	0.1802
27.06	0.4456	0.9101	0.7012	0.1162	0.7235	0.1668
31.14	0.4922	0.9248	0.7384	0.1073	0.7603	0.1463
35.09	0.5459	0.9376	0.7723	0.0983	0.7932	0.1350
38.92	0.5825	0.9469	0.8001	0.0901	0.8177	0.1198
42.61	0.6231	0.9538	0.8247	0.0826	0.8398	0.1162

Table 4.5: Summary statistics of MSSIM values for the LASSO-FISTA for 51 images for different number of measurements.

$M/N(\%)$	Minimum	Maximum	Mean	Std Dev	Median	Quartile Range
0.1409	0.2809	0.8289	0.5423	0.1339	0.5577	0.2061
0.1854	0.3567	0.8632	0.6032	0.1271	0.6248	0.1881
0.2286	0.4090	0.8892	0.6543	0.1196	0.6750	0.1697
0.2706	0.4637	0.9062	0.6995	0.1121	0.7209	0.1588
0.3114	0.5098	0.9214	0.7361	0.1038	0.7538	0.1397
0.3509	0.5637	0.9344	0.7698	0.0951	0.7904	0.1307
0.3892	0.5974	0.9443	0.7975	0.0875	0.8141	0.1223
0.4261	0.6347	0.9516	0.8224	0.0804	0.8341	0.1186

Table 4.6: Summary statistics of MSSIM values for the Basis Pursuit with Douglas-Rachford iterations for 51 images for different number of measurements.

In Table 4.3 we see that there are large relative errors when we only use 14.09% of the Fourier domain measurements. The largest relative error is 0.3454. But we also have that the smallest relative error is 0.0489. When we use 31.14% and 42.61% of the Fourier domain measurements, the smallest relative errors are 0.0249 and 0.0190 respectively. The corresponding MSSIM values for this image are 0.805, 0.8977 and 0.9272. Although it does not correspond to the maximum MSSIM values, the values are higher than the median MSSIM values for the different numbers of measurements taken.

A visual inspection aids us in understanding the effectiveness of the algorithms we used within the CS framework. In Figure 4.2.1 we show the result of reconstructions for three different images. The first image has the lowest minimum relative error across the different choices of L. The second image has around the median relative errors. The third image has the maximum relative errors. For all these cases

$M/N(\%)$	Minimum	Maximum	Mean	Std Dev	Median	Quartile Range
14.09	0.2792	0.8249	0.5427	0.1337	0.5583	0.2069
18.54	0.3544	0.8638	0.6055	0.1281	0.6283	0.1912
22.86	0.4065	0.8915	0.6575	0.1210	0.6770	0.1752
27.06	0.4609	0.9092	0.7033	0.1136	0.7248	0.1625
31.14	0.5067	0.9239	0.7401	0.1052	0.7610	0.1431
35.09	0.5617	0.9371	0.7738	0.0964	0.7938	0.1326
38.92	0.5943	0.9463	0.8012	0.0885	0.8185	0.1193
42.61	0.6340	0.9534	0.8257	0.0813	0.8400	0.1159

Table 4.7: Summary statistics of MSSIM values for the Dantzig selector with the primal-dual algorithm for 51 images for different number of measurements.

it is for the algorithms run using noisy measurements. The original images are respectively given in Figures 4.2.1a, 4.2.1b and 4.2.1c. With only 14% of the measurements the reconstructions are very noisy, although it still picks out the outlines of the image. However, when sampling slightly less than a third of the Fourier domain measurements, we see in Figures 4.2.1g, 4.2.1h and 4.2.1i we already obtain very satisfactory reconstructions, at least to the eye. Using 42.61% of the Fourier domain measurements we see very good reconstructions. It is evident from the reconstructed images that the algorithms and CS framework we apply here did work better for the image of the lady (see Figures 4.2.1a, 4.2.1d, 4.2.1g and 4.2.1j). Therefore, when we apply CS we need to be careful in choosing the correct sampling domain, as well as the correct sparsifying basis. But it should also be done within practical limitations. In general CS is applied in such a way that $M \ll N$, not only $M < N$, since this is when it is most useful. That is why we do not consider looking at results where we sample even more Fourier domain measurements.

We see a similar pattern in the MSSIM values compared to the relative errors. In CS we are mostly interested in the approximation error $||f - f^*||_1$, especially since there are theoretical error bounds for specific algorithms and noise levels (see Sections 3.3.6, 3.3.7). However, it is not common to see the MSSIM used to measure reconstruction error in the CS framework. The best measure of the success of a reconstruction would always be based on the application at hand.

To understand the structure and the errors we plot the sparsity of the Haar Transform coefficients as measured by the Gini Index (see Section 3.3.3) of each image on the x-axis against the relative ℓ_1 error on the y-axis. We see in Figure 4.2.2 what we expect since the relative errors decrease as the sparsity increases. It does therefore confirm that the specific basis to represent the image has an influence on how well we will be able to reconstruct the image from a limited number of measurements.

We see similar patterns in the relative errors and the MSSIM values across the images looking at Figures 4.2.3, 4.2.4, 4.2.5 and 4.2.6. It is notable that there is a lot of variation in the errors and MSSIM values. The most probable reason for this is because we have used the Fourier basis as the sensing matrix, and the Haar basis as the sparsifying basis. The Haar basis might not be the best sparsifying basis for each specific image. We also do not consider using overcomplete dictionaries (as discussed in Section 2.3.2) which would yield sparser images as compared to when we only use one basis. It is also noticeable that for some images the differences in errors for different levels of the number of measurements are small, and for other images it is large. For suggestions of other bases and uses of those bases see [27, 157].

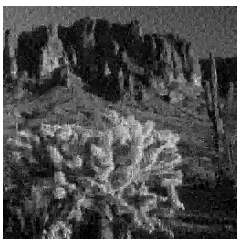
(a) Lowest relative error - original.

(b) Median relative error - original.

(c) Highest relative error - original.

(d) Lowest relative error - 14.09% measurements.

(e) Median relative error - 14.09% measurements.



(f) Highest relative error - 14.09% measurements.

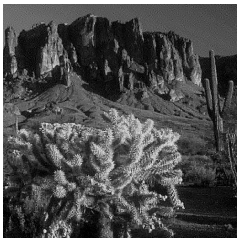
(g) Lowest relative error - 31.14% measurements.

(h) Median relative error - 31.14% measurements.

(i) Highest relative error - 31.14% measurements.

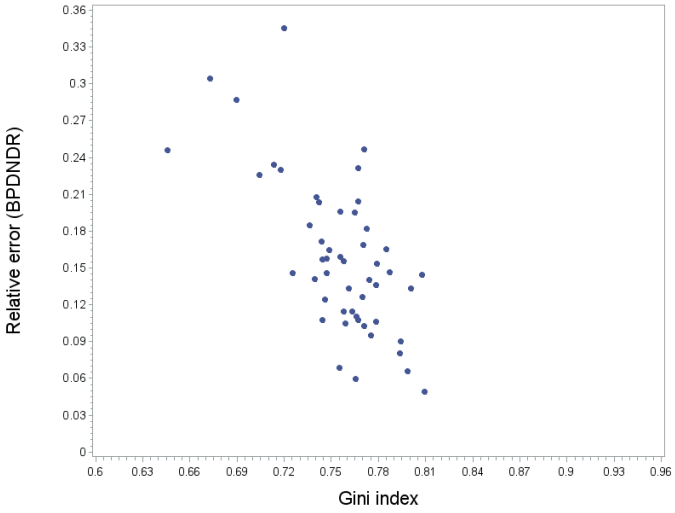
(j) Lowest relative error - 42.61% measurements.

(k) Median relative error - 42.61% measurements.

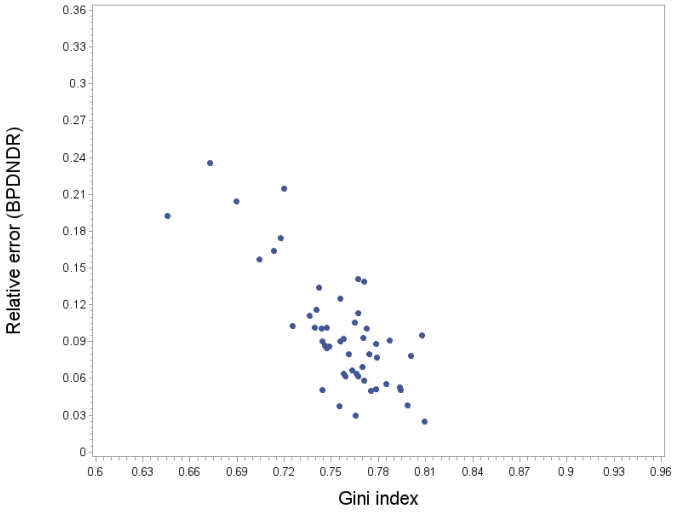


(l) Highest relative error - 42.61% measurements.

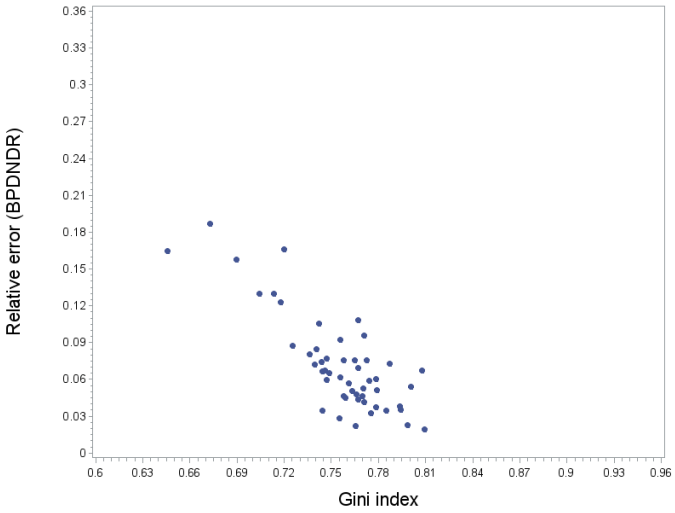
Figure 4.2.1: Reconstruction of images with lowest, median and highest relative errors. These are done with errors added to measurements as described above. The reconstructed images are given for different levels of measurements taken.



(a) $M/N = 14.09\%$

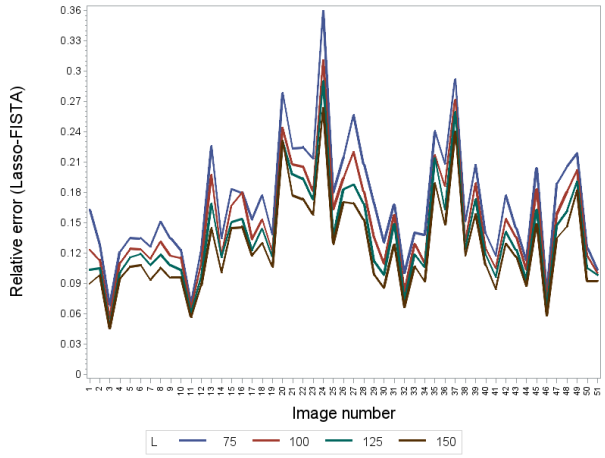


(b) $M/N = 31.14\%$

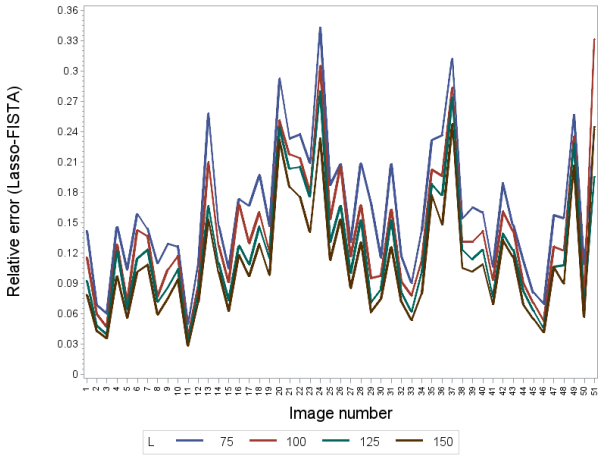


(c) $M/N = 42.61\%$

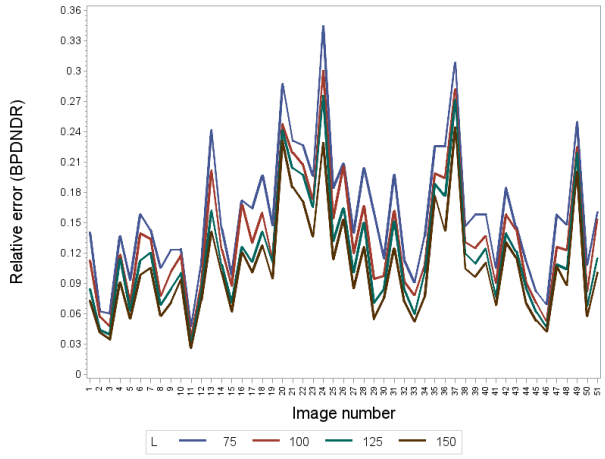
Figure 4.2.2: The relative errors of the BPDNDR algorithm against the Gini Index values of each of the 51 images, using the Haar basis with one level as sparsifying basis.



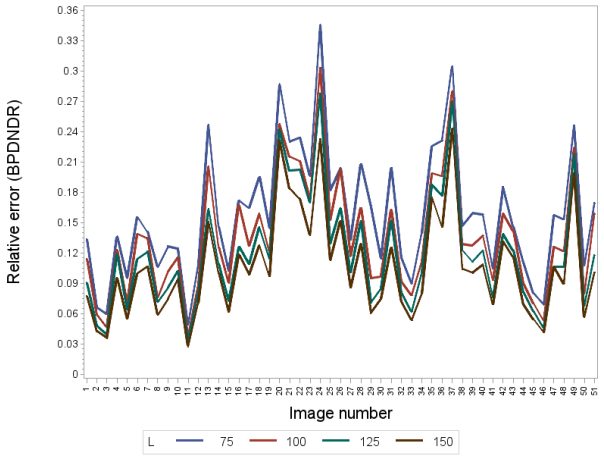
(a) Relative errors for LASSO-FISTA - noiseless case



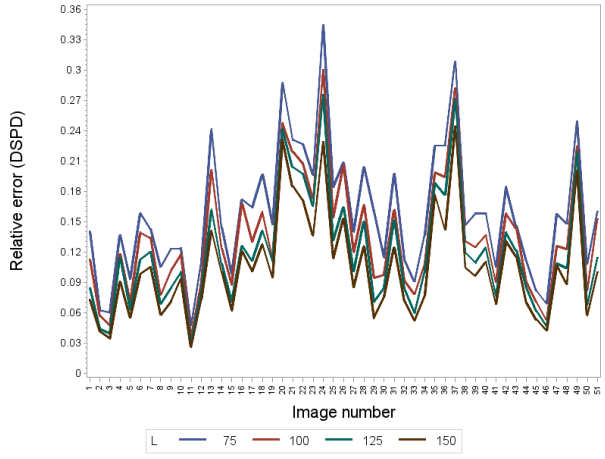
(b) Relative error for LASSO-FISTA - noisy case



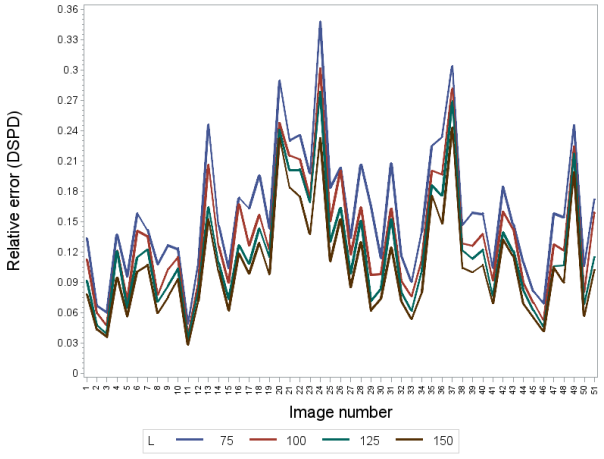
(c) Relative errors for BPDNDR noiseless case



(d) Relative errors for BPDNDR noisy case.

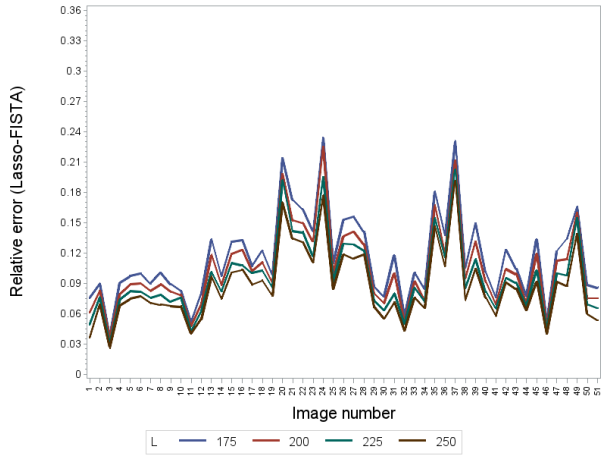


(e) Relative errors for DSPD noiseless case

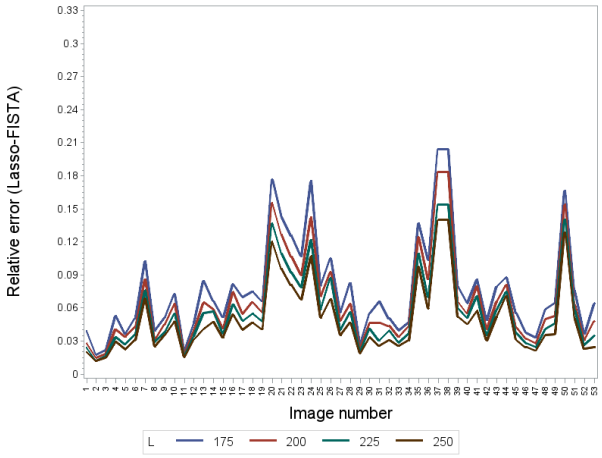


(f) Relative errors for DSPD - noisy case

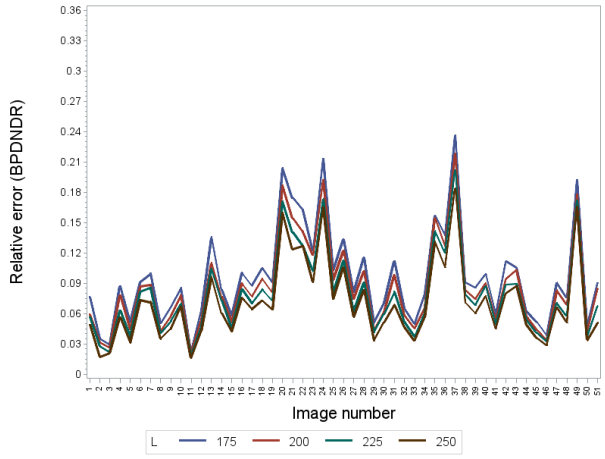
Figure 4.2.3: Relative errors for three approximation approaches, for $L = 75, 100, 125, 150$.



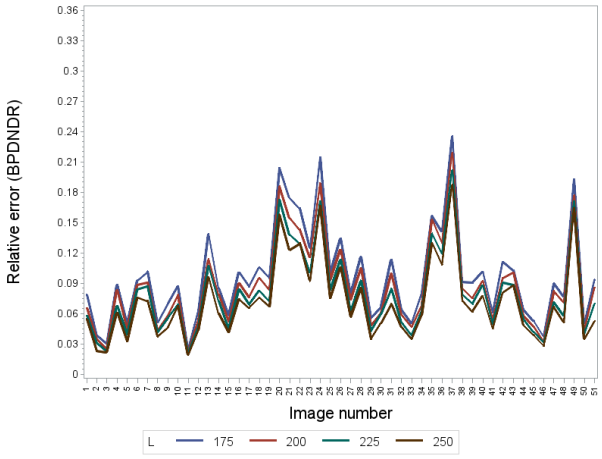
(a) Relative errors for LASSO-FISTA - noiseless measurements.



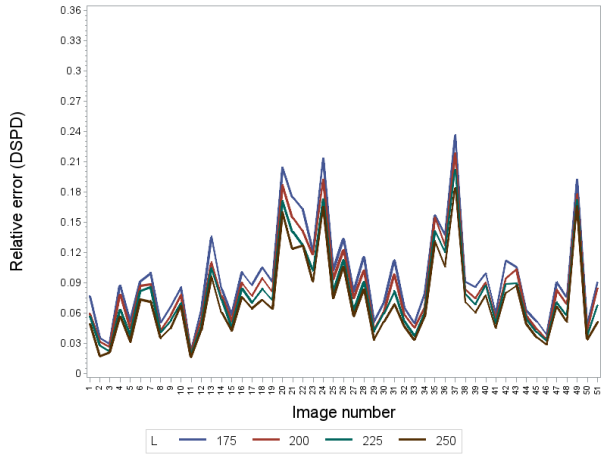
(b) Relative error for LASSO-FISTA - noisy case



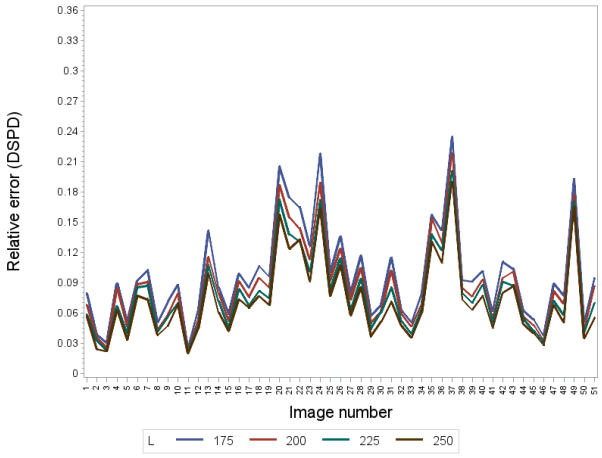
(c) Relative errors for BPDNDR noiseless case



(d) Relative errors for BPDNDR noisy case.

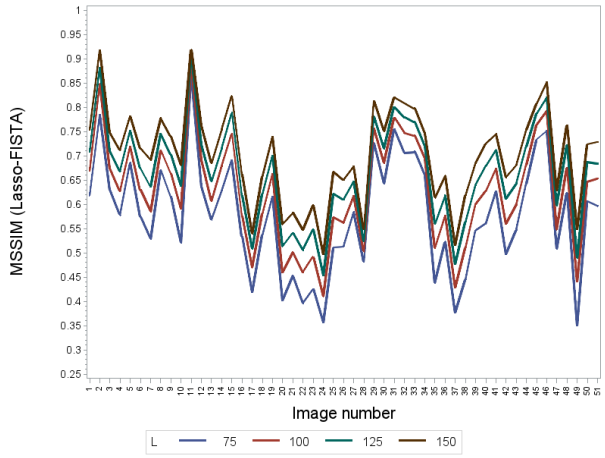


(e) Relative errors for DSPD noiseless case

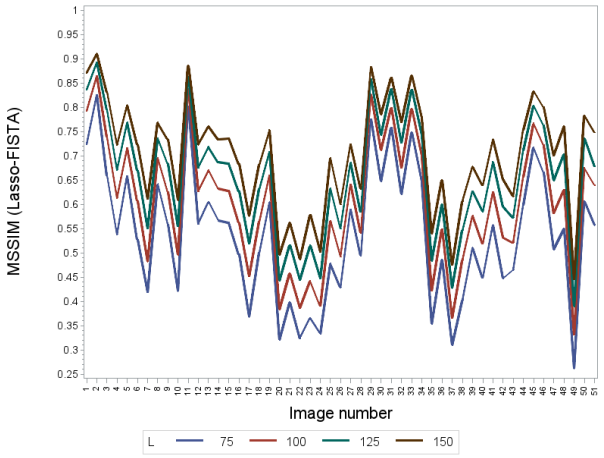


(f) Relative errors for DSPD - noisy case

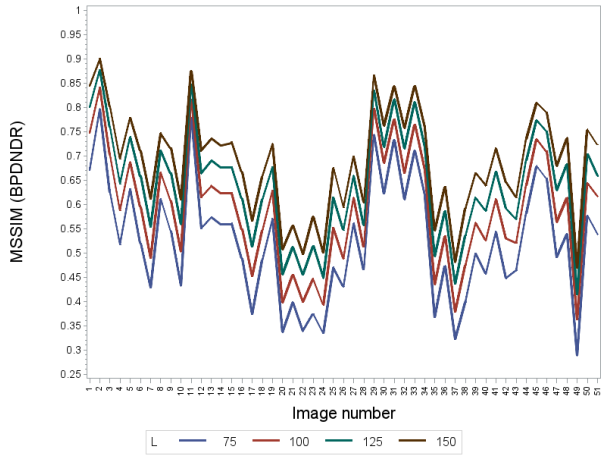
Figure 4.2.4: Relative errors for three approximation approaches, for $L = 175, 200, 225, 250$.



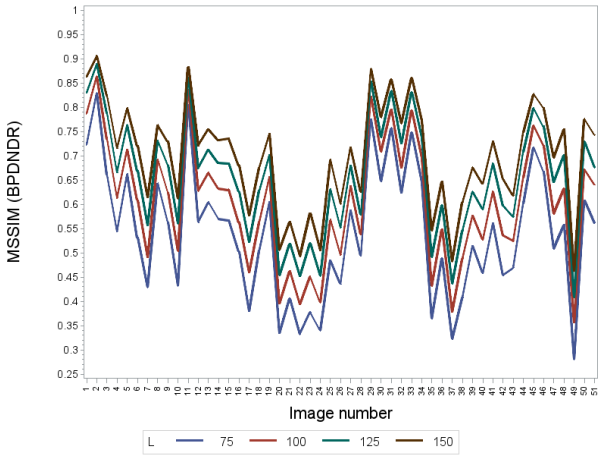
(a) MSSIM values for LASSO-FISTA - noiseless case



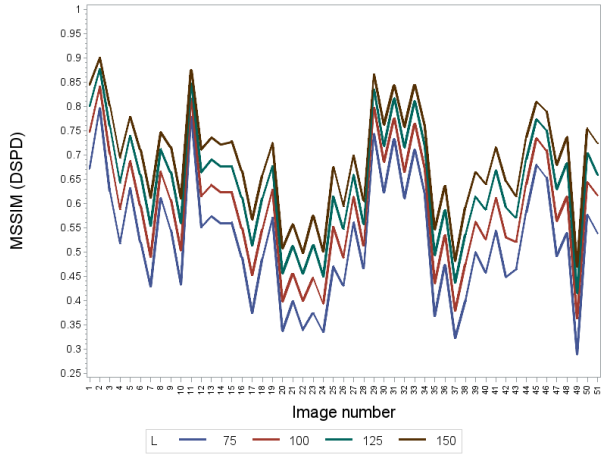
(b) MSSIM values for LASSO-FISTA - noisy case



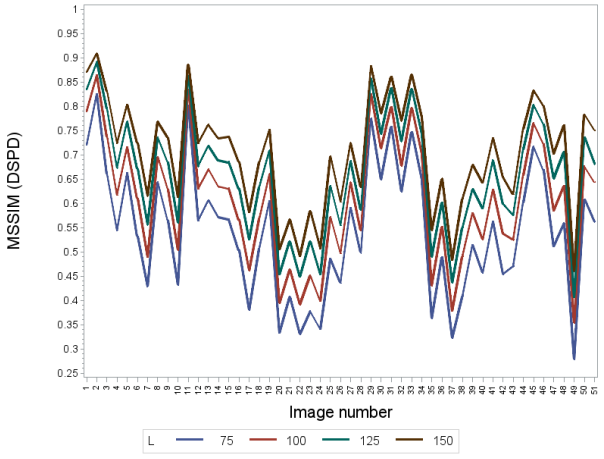
(c) MSSIM values for BPDNDR noiseless case



(d) MSSIM values for BPDNDR noisy case.

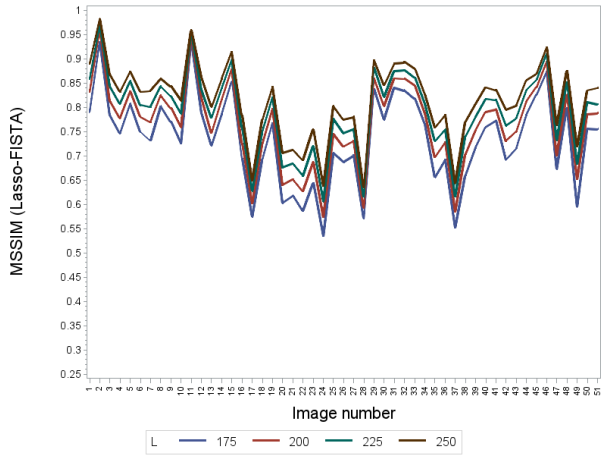


(e) MSSIM values for DSPD noiseless case

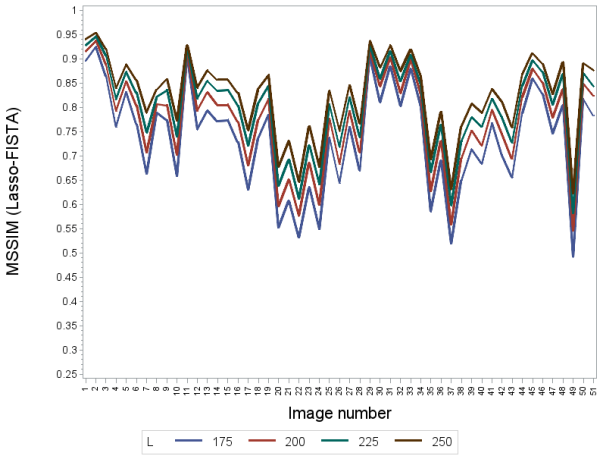


(f) MSSIM values for DSPD - noisy case

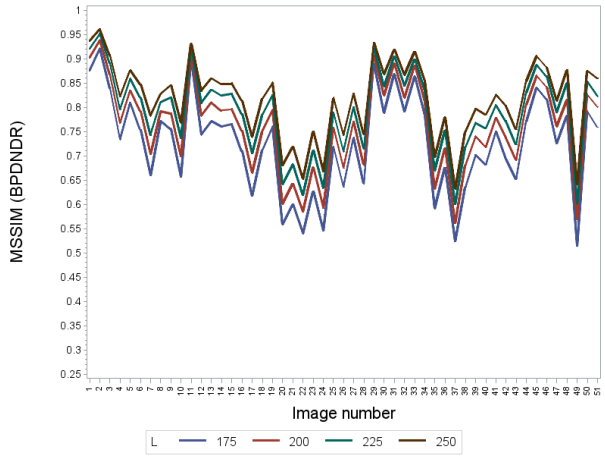
Figure 4.2.5: MSSIM values for three approximation approaches, for $L = 75, 100, 125, 150$.



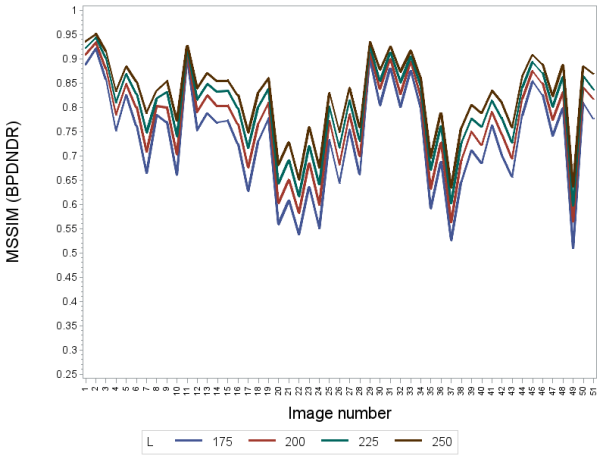
(a) MSSIM values for LASSO-FISTA - noiseless measurements.



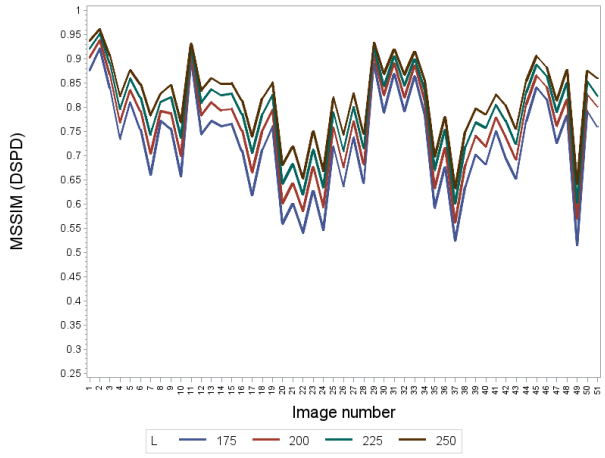
(b) MSSIM values for LASSO-FISTA - noisy case



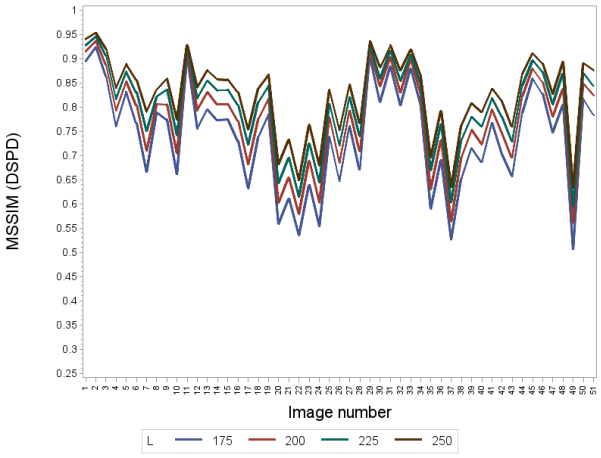
(c) MSSIM values for BPDNDR noiseless case



(d) MSSIM values for BPDNDR noisy case.



(e) MSSIM values for DSPD noiseless case



(f) MSSIM values for DSPD - noisy case

Figure 4.2.6: MSSIM values for three approximation approaches, for $L = 175, 200, 225, 250$.

4.3 Summary

In this chapter we provided an argument for using CS in the Big Data context with specific application to image analysis. We applied CS on 51 natural images and examined the results to exhibit the power of CS. As done in [157] we used partial Fourier matrices as the sensing matrices, and the Haar basis as the sparsifying basis. The Fourier domain is easy to implement because we do not need to store the sensing matrix, and it has been shown to be useful as a sensing matrix in [30]. The Haar basis is a wavelet basis which is generally useful to sparsify natural images, although it is not necessarily the most effective wavelet basis to use for any given image. Even though 51 images are not a lot of data, when grouped it has more than 13 million pixels, i.e. data points. It therefore does not directly solve a Big Data problem, but it shows that given what CS can do, it can be used for Big Data problems where we do have massive amounts of data. It also shows that CS algorithms can handle large data sets. If we had to store the entire Φ matrices MATLAB would not be able to handle it.

Chapter 5

Conclusion

This mini-dissertation provides an overview of compressed sensing (CS) and is by no means a full in-depth mathematical treatment of CS. It is written to provide the statistician with the necessary background and building blocks of CS, for use in the Big Data environment, and herein, CS is presented in a simple and clear manner for a statistician not familiar with the field. The literature review, however, provides all the texts required should the reader want the specific mathematical details. The document aims thus to link CS in the statistical and engineering fields. We have covered extensively, although admittedly not exhaustively, Big Data in Chapter 1. In Chapter 2 we considered important background principles and techniques that anyone looking to use CS should be aware of. In Chapter 3 we showed the main mechanics behind CS theory as shown in literature. Finally, in Chapter 4 we brought it all together and looked at current work being done which utilises the CS framework in an image processing context. Big Data does seem, at least for now, to be a sensible area of research in various disciplines. As argued in Chapter 1, Big Data impacts how we live because decision makers are spending time and money in utilising Big Data. It affects marketing campaigns, and it affects healthcare, amongst many others. Our aim was to see if Big Data problems are being approached using the exciting tool CS. After our investigation we argue that it is indeed. We argued the following reasons in this document:

- Compressed sensing is already impacting the Big Data world.
- Compressed sensing in its nature addresses the storage challenges posed by Big Data.
- Compressed sensing exploits sparsity, which is useful since Big Data generally contains an excess of data that is not useful and completely redundant.

Compressed sensing does not necessarily provide the correct solutions to all Big Data problems. For example in business. Big Data of business are typically the combination of different sources of data. The aim is usually not to compress or store data efficiently, but rather to find the valuable bits of information that is hidden in the massive amounts of data. Especially since different data sources contain information

that is related to each other. Therefore CS might not find the most traction in this side of the Big Data world.

One of the main disadvantages of CS is that it can be difficult to practically implement it in real-world applications because of the conditions on the sensing matrix. Furthermore, Big Data might come second-best to other well-known techniques when robustness is all we care about. However, we believe we have have shown the reader how exciting CS is and that it does provide solutions to real-word problems in the Big Data setting.

For future research we propose that the possibility of database applications utilising the CS framework be looked at since Big Data is in many case a database problem, especially in the business world. Another area where the application of CS can be considered is in text analytics. A challenging area will be to use CS in real-time applications where it is necessary to do a reconstruction quickly.

Appendix A

Images used for application

Appendix B

Code used for application

```
% gebaseer op die mondrian 2D demo

clear all


addpath('C:/Users/User/Dropbox/Charl/splittingsolvers/2D/Demos')
addpath('C:/Users/User/Dropbox/Charl/splittingsolvers/2D/Measurements')
addpath('C:/Users/User/Dropbox/Charl/splittingsolvers/2D/Dictionaries')
addpath('C:/Users/User/Dropbox/Charl/splittingsolvers/2D/Datasets')
addpath('C:/Users/User/Dropbox/Charl/splittingsolvers/Utils')
addpath('C:\Users\User\Dropbox\Charl\splittingsolvers\1D\Dictionaries')
addpath('C:\Users\User\Dropbox\Charl\splittingsolvers\Solvers\proxsplit')
addpath('C:\Users\User\Dropbox\Charl\Wavelab850\Orthogonal')
addpath('C:\Users\User\Dropbox\Charl\splittingsolvers\Utils\proxop')
addpath('C:\Users\User\Dropbox\Charl\splittingsolvers\UDWT')
addpath('C:\Users\User\Dropbox\Charl\splittingsolvers\Utils\projop')
% copy
cd 'C:\Users\User\Dropbox\Charl\images'


% define necessary parameters and storage vectors/matrix
nimg =53;
nL = 8;
nSNR = 2;
its = nSNR*nimg*nL;
SNR_array = [0,30];
L_array = [75,100,125,150,175,200,225,250];
InfoMatrix = zeros(its,25);
```

```
countIts = 0;

% Titles: 'book' (subfigures), or 'toolbox' (meaningful titles).
titles = 'toolbox';

% (charl bygevoeg)
for iii = 1:nimg,% start loop over images

    if iii<12,
        img_name = ['img' num2str(iii) '.tif'];
    end;

    if iii>=12,
        img_name = ['img' num2str(iii) '.gif'];
    end;

    % Save figures: 1/0 (default 0).
    savfig = 0;

    % Image
    x = double(imread(img_name));

    sizex = size(x);
    ss = size(sizex);

    if ss(2) == 3,
        x = x(:,:,1);
    end;

    n = length(x);  % Image size n x n.
    x_orig = x;
    meanx = mean(x(:));
    sd_x = std(x(:));
    x = x(:) - mean(x(:));

    for jjj = 1:nL,% start loop over values of L

        % Dictionary (here DWT).
        qmf=MakeONFilter('Haar');
        dict='PO2';pars1=0;pars2=qmf;pars3=0;
        p = SizeOfDict2(n,dict,pars1,pars2,pars3);
        tightFrame = p/(n*n);   % tF constant= p/(n*n) x constant of Phi
```

```
be = 1/tightFrame;

% Measurement (sensing) operator.
dictCS = 'Fourier';
L = L_array(jjj);  % number of radial lines in the Fourier domain.
[M,Omega] = LineMask(L,n);
m = length(Omega);      % Number of measurements m.

% Observed noisless data.

y0 = FastMeasure2D(x, dictCS, Omega);
y_measure = y0;

for kkk = 1:nSNR, % insert loop over noise levels

    %kkk=2;

    fprintf('i: %d \n' ,iii)
    fprintf('L: %d \n', L)

    countIts = countIts + 1;

    SNR = SNR_array(kkk);

    % (charl bygevoeg)
    if kkk == 1,
        sigma = 0;
    else,
        sigma = std(y0)*10^(-SNR/20);
    end;
    fprintf('SNR: %d \n', SNR)
    fprintf('sigma: %d\n', sigma)

    rng(3)
    y = y0 + sigma*randn(size(y0));
    sumy0 = sum(y0(:));
    sumy = sum(y(:));
    fprintf('sumy0: %f \n', sumy0);
    fprintf('sumy: %f \n', sumy);

    % Solve Lasso in l1-penalized form with FISTA.
    % lambda chosen such that ||r|| <= epsilon.
```

```
options.mu    = be;          % Descent step-size for FB.
options.method = 'fista';
options.niter = 500;
options.verb  = 0;
lambda = sigma;
K  = @(x)FastMeasure2D(reshape(FastS2(x,n,dict,pars1,
pars2,pars3),n*n,1), dictCS, Omega);
KS =@(x)FastA2(reshape(FastMeasureAdjoint2D(x,n*n,dictCS,Omega)
,n,n),dict,pars1,pars2,pars3);
ProxF = @(x,mu)SoftThresh(x,mu*lambda);
GradG = @(x)KS(K(x) - y);
tic;
xhatlassofista=real(fb(zeros(p,1), ProxF, GradG, be, options));
timelassofista=toc

% Solve BPDN with DR.
options.gamma = 1;  % Stepsize parameter Douglas-Rachford iter.
options.tau   = 1;  % No relaxation.
options.niter = 500;
options.verb  = 0;
epsilon = n*sigma*sqrt(1+2*sqrt(2/(n*n)));
epsilon = n*sigma/2;
ProxF = @(x,ga)SoftThresh(x,ga);
ProxG = prox_F_tightframe(@(x,ga)compute_l2ball_
projection(x,epsilon),K,KS,tightFrame,y);
tic;xhatBPDNDR = real(dr(zeros(p,1),ProxF,ProxG,options));
timeBPDNDR=toc

% Solve Dantzig-Selector with PD.
options.mu    = .99*be;      % Stepsize on the dual.
options.nu    = .99*be;      % Stepsize on the primal.
options.tau   = 1;       % No relaxation.
options.niter = 500;
options.verb  = 0;
delta = lambda;       % Same parameter as Lasso.
ProxG = @(x,mu)SoftThresh(x,mu);
z = KS(y);
ProxF = @(x,nu)z+compute_linfball_projection(x-z,delta);
ProxFS= prox_conjugate(ProxF);
KK  = @(x)KS(K(x));
KKS = KK;
GradH = @(x)0;
```

```
%options.report = @(x,u)norm(x,1)+sum(z.*u)+delta*norm(u,1);
tic;xhatDSPD = real(primaldual(zeros(p,1), KK,  KKS, ProxFS,
ProxG, GradH, options));timeDSPD=toc

% Compute recovered images.
NAME = NthList(dict, 1);
PAR1 = NthList(pars1, 1);
PAR2 = NthList(pars2, 1);
PAR3 = NthList(pars3, 1);
C = xhatlassofista;
yhatlassofista=eval(['Fast'NAME,'Synthesis(C(:),n,
    PAR1,PAR2,PAR3)']);
C = xhatBPDNDR;
yhatBPDNDR = eval(['Fast'NAME,'Synthesis(C(:),n,
    PAR1,PAR2,PAR3)']);
C = xhatDSPD;
yhatDSPD = eval(['Fast'NAME,'Synthesis(C(:),n,
    PAR1,PAR2,PAR3)']);


% compute stuff for
% error metrics and things stored

% psnr's
psnr_lasso = psnr(yhatlassofista+meanx,x_orig);
psnr_BPDNDR = psnr(yhatBPDNDR+meanx,x_orig);
psnr_DSPD = psnr(yhatDSPD+meanx,x_orig);

% MSE's
mse_lasso = immse(yhatlassofista+meanx,x_orig);
mse_BPDNDR = immse(yhatBPDNDR+meanx,x_orig);
mse_DSPD = immse(yhatDSPD+meanx,x_orig);

% relative errors - 1 - norm
relerr_lasso1=norm(x_orig-(yhatlassofista+meanx),1) /
norm(x_orig,1);
relerr_BPDNDR1=norm(x_orig-(yhatBPDNDR+meanx),1) /
norm(x_orig,1);
relerr_DSPD1=norm(x_orig-(yhatDSPD+meanx),1)/norm(x_orig,1);

% relative errors - 2 - norm
relerr_lasso2=norm(x_orig-(yhatlassofista+meanx),2) /
```

```
norm(x_orig,2);
relerr_BPDNDR2=norm(x_orig-(yhatBPDNDR+meanx),2) /
norm(x_orig,2);
relerr_DSPD2=norm(x_orig-(yhatDSPD+meanx),2) /
norm(x_orig,2);

disp(relerr_lasso1)
disp(relerr_BPDNDR1)
disp(relerr_DSPD1)

% ssim
[mssim_lasso,ssim_mapL]=ssim_index(x_orig,yhatlassofista+meanx)
[mssim_BPDNDR,ssim_mapBD]=ssim_index(x_orig, yhatBPDNDR+meanx);
[mssim_DSPD,ssim_mapD]=ssim_index(x_orig, yhatDSPD+meanx);

% matrix for storing everything
InfoMatrix(countIts,1) = iii;
InfoMatrix(countIts,2) = jjj;
InfoMatrix(countIts,3) = kkk;
InfoMatrix(countIts,4) = L;
InfoMatrix(countIts,5) = m;
InfoMatrix(countIts,6) = m/(n*n);
InfoMatrix(countIts,7) = psnr_lasso;
InfoMatrix(countIts,8) = psnr_BPDNDR;
InfoMatrix(countIts,9) = psnr_DSPD;
InfoMatrix(countIts,10) = mse_lasso;
InfoMatrix(countIts,11) = mse_BPDNDR;
InfoMatrix(countIts,12) = mse_DSPD;
InfoMatrix(countIts,13) = relerr_lasso1;
InfoMatrix(countIts,14) = relerr_BPDNDR1;
InfoMatrix(countIts,15) = relerr_DSPD1;
InfoMatrix(countIts,16) = relerr_lasso2;
InfoMatrix(countIts,17) = relerr_BPDNDR2;
InfoMatrix(countIts,18) = relerr_DSPD2;
InfoMatrix(countIts,19) = mssim_lasso;
InfoMatrix(countIts,20) = mssim_BPDNDR;
InfoMatrix(countIts,21) = mssim_DSPD;
InfoMatrix(countIts,22) = meanx;
InfoMatrix(countIts,23) = sd_x;
InfoMatrix(countIts,24) = norm(y-y0,1)/norm(y,1);
InfoMatrix(countIts,25) = sigma;
```

```
            % Store approximated image
            filename_lasso = ['lasso_i_' num2str(iii) '_j_'
                num2str(jjj) '_k_' num2str(kkk) '.tif'];
            filename_BPDNDR = ['BPDNDR_i_' num2str(iii) '_j_'
                num2str(jjj) '_k_' num2str(kkk) '.tif'];
            filename_DSPD = ['DSPD_i_' num2str(iii) '_j_'
                num2str(jjj) '_k_' num2str(kkk) '.tif'];
            yhat_lasso = (yhatlassofista+meanx)/255;
            yhat_BPDNDR = (yhatBPDNDR+meanx)/255;
            yhat_DSPD = (yhatDSPD+meanx)/255;
            imwrite(yhat_lasso,filename_lasso);
            imwrite(yhat_BPDNDR,filename_BPDNDR);
            imwrite(yhat_DSPD,filename_DSPD);

        end; % end loop for noise/no-noise
    end; % end loop for L = # measurements
end; % end loop for Images

xlswrite('infomatrix_9dec16.xls',InfoMatrix,'A2:Y849');
```

Bibliography

[1] Tinku Acharya and Ping-Sing Tsai. *JPEG2000 standard for image compression: concepts, algorithms and VLSI architectures*. John Wiley & Sons, 2005.

[2] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *Signal Processing, IEEE Transactions on*, 54(11):4311–4322, 2006.

[3] Nasir Ahmed, T Natarajan, and Kamisetty R Rao. Discrete cosine transform. *Computers, IEEE Transactions on*, 100(1):90–93, 1974.

[4] Daito Akimura, Yoshihiro Kawahara, and Tohru Asami. Compressed sensing method for human activity sensing using mobile phone accelerometers. In *2012 Ninth International Conference on Networked Sensing (INSS)*, pages 1–4, June 2012.

[5] Mikhled Alfaouri and Khaled Daqrouq. ECG signal denoising by wavelet transform thresholding. *American Journal of Applied Sciences*, 5(3):276–281, 2008.

[6] Anestis Antoniadis, Piotr Fryzlewicz, and Frédérique Letué. The dantzig selector in cox's proportional hazards model. *Scandinavian Journal of Statistics*, 37(4):531–552, 2010.

[7] Marc Antonini, Michel Barlaud, Pierre Mathieu, and Ingrid Daubechies. Image coding using wavelet transform. *Image Processing, IEEE Transactions on*, 1(2):205–220, 1992.

[8] Andreas Antoniou. *Digital filters*. McGraw Hill, 1993.

[9] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy H Katz, Andrew Konwinski, Gunho Lee, David A Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A Berkeley view of cloud computing. Technical report, University of California, Berkeley, 2009.

[10] Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, 2008.

[11] Richard G Baraniuk. Compressive sensing. *IEEE signal processing magazine*, 24(4), 2007.

[12] Richard G Baraniuk. More is less: signal processing and the data deluge. *Science*, 331(6018):717–719, 2011.

[13] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[14] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD Cup and Workshop*, volume 2007, page 35, 2007.

[15] Alex Berson and Stephen J Smith. *Data warehousing, data mining, and OLAP*. McGraw-Hill, Inc., 1997.

[16] J. D. Blanchard, C. Cartis, and J. Tanner. Decay properties of restricted isometry constants. *IEEE Signal Processing Letters*, 16(7):572–575, July 2009.

[17] Jeffrey D Blanchard and Jared Tanner. Performance comparisons of greedy algorithms in compressed sensing. *Numerical Linear Algebra with Applications*, 22(2):254–282, 2015.

[18] Thomas Blumensath and Mike E Davies. Normalized iterative hard thresholding: Guaranteed stability and performance. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):298–309, 2010.

[19] Jérôme Bobin, J-L Starck, and Roland Ottensamer. Compressed sensing in astronomy. *Selected Topics in Signal Processing, IEEE Journal of*, 2(5):718–726, 2008.

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

[21] Steve Bryson, David Kenwright, Michael Cox, David Ellsworth, and Robert Haimes. Visually exploring gigabyte data sets in real time. *Communications of the ACM*, 42(8):82–90, 1999.

[22] C Sidney Burrus, Ramesh A Gopinath, and Haitao Guo. *Introduction to wavelets and wavelet transforms: a primer*. Prentice-Hall, Inc., 1997.

[23] William M Campbell, Charlie K Dagli, and Clifford J Weinstein. Social network analysis with content and graphs. *Lincoln Laboratory Journal*, 20(1):61–81, 2013.

[24] Emmanuel Candes and Justin Romberg. Sparsity and incoherence in compressive sampling. *Inverse Problems*, 23(3):969, 2007.

[25] Emmanuel Candes and Terence Tao. The Dantzig selector: statistical estimation when p is much larger than n. *The Annals of Statistics*, 35:2313–2351, 2007.

[26] Emmanuel J Candès. Compressive sampling. In *Proceedings of the International Congress of Mathematicians: Madrid, August 22-30, 2006: Invited Lectures*, pages 1433–1452, 2006.

[27] Emmanuel J Candes, Yonina C Eldar, Deanna Needell, and Paige Randall. Compressed sensing with coherent and redundant dictionaries. *Applied and Computational Harmonic Analysis*, 31(1):59–73, 2011.

[28] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[29] Emmanuel J Candes and Justin Romberg. Quantitative robust uncertainty principles and optimally sparse decompositions. *Foundations of Computational Mathematics*, 6(2):227–254, 2006.

[30] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, 2006.

[31] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006.

[32] Emmanuel J Candes and Terence Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215, 2005.

[33] Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *Information Theory, IEEE Transactions on*, 52(12):5406–5425, 2006.

[34] Emmanuel J Candès and Michael B Wakin. An introduction to compressive sampling. *Signal Processing Magazine, IEEE*, 25(2):21–30, 2008.

[35] Constantin Carathéodory. Uber den variabilitatsbereich der koeffizienten von potenzreihen, die gegebene werte nicht annehmen. *Mathematische Annalen*, 64(1):95–115, 1907.

[36] Constantin Carathéodory. Uber den variabilitatsbereich der fourierschen konstanten von positiven harmonischen funktionen. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 32(1):193–217, 1911.

[37] Peter G Casazza, Gitta Kutyniok, and Friedrich Philipp. Introduction to finite frame theory. In *Finite Frames*, pages 1–53. Springer, 2013.

[38] Rui Castro, Jarvis Haupt, and Robert Nowak. Compressed sensing vs. active learning. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 3, pages III–III. IEEE, 2006.

[39] Jos Mara Cavanillas, Edward Curry, and Wolfgang Wahlster. *New Horizons for a Data-Driven Economy: A Roadmap for Usage and Exploitation of Big Data in Europe*. Springer Publishing Company, Incorporated, 2016.

[40] Volkan Cevher, Steffen Becker, and Martin Schmidt. Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics. *Signal Processing Magazine, IEEE*, 31(5):32–43, 2014.

[41] S Chatrchyan, G Hmayakyan, V Khachatryan, AM Sirunyan, W Adam, T Bauer, T Bergauer, H Bergauer, M Dragicevic, J Erö, et al. The CMS experiment at the CERN LHC. *Journal of Instrumentation*, 3(08):S08004, 2008.

[42] Chen Chen, Eric W. Tramel, and James E. Fowler. Compressed-sensing recovery of images and video using multihypothesis predictions. In *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 1193–1198, Nov 2011.

[43] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

[44] Xi Chen, Zhuizhuan Yu, Sebastian Hoyos, Brian M Sadler, and Jose Silva-Martinez. A sub-nyquist rate sampling receiver exploiting compressive sensing. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 58(3):507–520, 2011.

[45] Xue-Wen Chen and Xiaotong Lin. Big data deep learning: Challenges and perspectives. *IEEE Access*, 2:514–525, 2014.

[46] Ole Christensen. *An Introduction to Frames and Riesz bases*, volume 7. Springer, 2003.

[47] Charles K. Chui. *An Introduction to Wavelets*. Boston, MA: Academic Press, 1992.

[48] Jon F Claerbout and Francis Muir. Robust modeling with erratic data. *Geophysics*, 38(5):826–844, 1973.

[49] Brian Coe. *Cameras: from Daguerreotypes to instant pictures*. Random House Value Publishing, 1978.

[50] Albert Cohen, Wolfgang Dahmen, and Ronald DeVore. Compressed sensing and best k-term approximation. *Journal of the American Mathematical Society*, 22(1):211–231, 2009.

[51] Leon Cohen. *Time-frequency analysis*, volume 778. Prentice hall, 1995.

[52] Ronald Coifman, F Geshwind, and Yves Meyer. Noiselets. *Applied and Computational Harmonic Analysis*, 10(1):27–44, 2001.

[53] Samantha Cook, Corrie Conrad, Ashley L Fowlkes, and Matthew H Mohebbi. Assessing Google flu trends performance in the United States during the 2009 influenza virus a (h1n1) pandemic. *PloS One*, 6(8):e23610, 2011.

[54] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[55] G Cormode and S Muthukrishnan. Combinatorial algorithms for compressed sensing. In *International Colloquium on Structural Information and Communication Complexity*, pages 280–294. Springer, 2006.

[56] Harold Scott Macdonald Coxeter. *Introduction to geometry*. New York, London, 1961.

[57] David J Cutler, Michael E Zwick, Minerva M Carrasquillo, Christopher T Yohn, Katherine P Tobin, Carl Kashuk, Debra J Mathews, Nila A Shah, Evan E Eichler, Janet A Warrington, and Aravinda Chakravrti. High-throughput variation detection and genotyping using microarrays. *Genome Research*, 11(11):1913–1925, 2001.

[58] Ingrid Daubechies. *Ten Lectures on Wavelets*, volume 61. SIAM, 1992.

[59] Mark Davenport, Jason N Laska, John R Treichler, Richard G Baraniuk, et al. The pros and cons of compressive sensing for wideband signal acquisition: Noise folding versus dynamic range. *Signal Processing, IEEE Transactions on*, 60(9):4628–4642, 2012.

[60] Mark A. Davenport, Petros T. Boufounos, Michael B. Wakin, and R.G. Baranuik. Signal processing with compressive measurements. *IEEE Journal of Selected Topics in S*, 4(2):445–460, 2010.

[61] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3:1376, 2013.

[62] Baron Gaspard Riche de Prony. Essai experimental et analytique: sur les lois de la dilatabilite de fluides elastique et sur celles de la force expansive de la vapeur de lalkool,a differentes temperatures. *Journal de Lecole Polytechnique*, 1(22):24–76, 1795.

[63] F.X Diebold. Big data dynamic factor models for macroeconomic measurement and forecasting. In *Advances in Economics and Econometrics, Eighth World Congress of the Econometric Society*, pages 115–122. Cambridge University Press, 2003.

[64] Qi Ding and Eric D Kolaczyk. A compressed PCA subspace method for anomaly detection in high-dimensional data. *Information Theory, IEEE Transactions on*, 59(11):7419–7433, 2013.

[65] Thong T. Do, Yi Chen, Dzung T. Nguyen, Nam Nguyen, Lu Gan, and Trac D. Tran. Distributed compressed video sensing. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 1393–1396, Nov 2009.

[66] David L Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

[67] David L Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

[68] David L Donoho and Xiaoming Huo. Uncertainty principles and ideal atomic decomposition. *Information Theory, IEEE Transactions on*, 47(7):2845–2862, 2001.

[69] David L Donoho and Philip B Stark. Uncertainty principles and signal recovery. *SIAM Journal on Applied Mathematics*, 49(3):906–931, 1989.

[70] David L Donoho, Yaakov Tsaig, Iddo Drori, and Jean-Luc Starck. Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 58(2):1094–1121, 2012.

[71] Marco F Duarte, Mark A Davenport, Dharmpal Takhar, Jason N Laska, Ting Sun, Kevin E Kelly, and Richard G Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2):83, 2008.

[72] Julio Martin Duarte-Carvajalino and Guillermo Sapiro. Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization. *Image Processing, IEEE Transactions on*, 18(7):1395–1408, 2009.

[73] Richard Duffin and Albert C Schaeffer. A class of nonharmonic Fourier series. *Transactions of the American Mathematical Society*, 72(2):341–366, 1952.

[74] Jonathan Eckstein and Dimitri P Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992.

[75] Michael Elad and Alfred M Bruckstein. A generalized uncertainty principle and sparse representation in pairs of bases. *Information Theory, IEEE Transactions on*, 48(9):2558–2567, 2002.

[76] Yonina C Eldar and Gitta Kutyniok. *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.

[77] Wei Fan and Albert Bifet. Mining big data: current status, and forecast to the future. *ACM SIGKDD Explorations Newsletter*, 14(2):1–5, 2013.

[78] Leyuan Fang, Shotao Li, Ryan P. McNabb, Qing Nie, Anthony N. Kuo, Cynthia A. Toth, Joseph A. Izatt, and Sina Farsiu. Fast acquisition and reconstruction of optical coherence tomography images via sparse representation. *IEEE Transactions on Medical Imaging*, 32(11):2034–2049, Nov 2013.

[79] Mario A. T. Figueiredo, Robert D. Nowak, and Stephen 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, Dec 2007.

[80] Joseph Fourier. *Theorie analytique de la chaleur, par M. Fourier*. Chez Firmin Didot, père et fils, 1822.

[81] James E. Fowler, Sungkwang Mun, and Eric W. Tramel. Block-based compressed sensing of images and video. *Foundations and Trends in Signal Processing*, 4(4):297–416, April 2012.

[82] Beate Franke, Jean-FRANçois Plante, Ribana Roscher, En-shiun Annie Lee, Cathal Smyth, Armin Hatefi, Fuqi Chen, Einat Gil, Alexander Schwing, Alessandro Selvitella, et al. Statistical inference, learning and models in big data. *International Statistical Review*, 84(3):371–389, 2016.

[83] Jean-Jacques Fuchs. On sparse representations in arbitrary redundant bases. *Information Theory, IEEE Transactions on*, 50(6):1341–1344, 2004.

[84] Yufei Gan, Tong Zhuo, and Chu He. Image classification with a deep network model based on compressive sensing. In *2014 12th International Conference on Signal Processing (ICSP)*, pages 1272–1275. IEEE, 2014.

[85] Andrej Y Garnaev and Efim D Gluskin. The widths of a Euclidean ball. In *Doklady Akademii Nauk SSSR*, volume 277, pages 1048–1052, 1984.

[86] Anna C Gilbert, Sudipto Guha, Piotr Indyk, S Muthukrishnan, and Martin Strauss. Near-optimal sparse Fourier representations via sampling. In *Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing*, pages 152–161. ACM, 2002.

[87] Anna C. Gilbert, Martin J. Strauss, Joel A. Tropp, and Roman Vershynin. One sketch for all: Fast algorithms for compressed sensing. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 237–246, New York, NY, USA, 2007. ACM.

[88] Anna C Gilbert, Martin J Strauss, Joel A Tropp, and Roman Vershynin. One sketch for all: fast algorithms for compressed sensing. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of Computing*, pages 237–246. ACM, 2007.

[89] S. Gorn, R.W. Bemer, and J. Green. American Standard Code for Information Interchange. *Communications of the ACM*, 6(8):422–426, Aug 1963.

[90] Rémi Gribonval and Morten Nielsen. Sparse representations in unions of bases. *Information Theory, IEEE Transactions on*, 49(12):3320–3325, 2003.

[91] Anthony Griffin, Toni Hirvonen, Christos Tzagkarakis, Athanasios Mouchtaris, and Panagiotis Tsakalides. Single-channel and multi-channel sinusoidal audio coding using compressed sensing. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(5):1382–1395, 2011.

[92] Qilong Han, Shuang Liang, and Hangli Zhang. Mobile cloud sensing, big data, and 5G networks make an intelligent and smart world. *IEEE Network*, 29(2):40–45, March 2015.

[93] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical Learning with Sparsity: the LASSO and Generalizations*. CRC Press, 2015.

[94] Jarvis Haupt and Robert Nowak. Signal reconstruction from noisy random projections. *Information Theory, IEEE Transactions on*, 52(9):4036–4048, 2006.

[95] Werner Heisenberg. Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. *Zeitschrift für Physik*, 43(3-4):172–198, 1927.

[96] Martin Hilbert. Big data for development: A review of promises and challenges. *Development Policy Review*, 34(1):135–174, 2016.

[97] Martin Hilbert and Priscila López. The worldś technological capacity to store, communicate, and compute information. *Science*, 332(6025):60–65, 2011.

[98] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[99] O. Holder. Ueber einen Mittelwertsatz. *Nachrichten von der Konigl. Gesellschaft der Wissenschaften und der Georg-Augusts-Universitat zu Gottingen*, 2:38–47, 1889.

[100] Kenneth B Howell. *Principles of Fourier analysis*. CRC Press, 2001.

[101] David A Huffman. A method for the construction of minimum redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[102] Niall Hurley and Scott Rickard. Comparing measures of sparsity. *Information Theory, IEEE Transactions on*, 55(10):4723–4741, 2009.

[103] Thomas R Ioerger, Sunwoo Koo, Eun-Gyu No, Xiaohua Chen, Michelle H Larsen, William R Jacobs Jr, Manormoney Pillay, A Willem Sturm, and James C Sacchettini. Genome analysis of multi-and extensively-drug-resistant tuberculosis from KwaZulu-Natal, South Africa. *PloS One*, 4(11):e7778, 2009.

[104] Mark A Iwen. A deterministic sub-linear time sparse fourier algorithm via non-adaptive compressed sensing methods. In *Proceedings of the nineteenth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 20–29. SIAM, 2008.

[105] HV Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. *Communications of the ACM*, 57(7):86–94, 2014.

[106] Anil K Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, Inc., 1989.

[107] George H Joblove and Donald Greenberg. Color spaces for computer graphics. In *ACM Siggraph Computer Graphics*, volume 12, pages 20–25. ACM, 1978.

[108] William B Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26(189-206):1, 1984.

[109] Iain M Johnstone and D Michael Titterington. Statistical challenges of high-dimensional data. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 367(1906):4237–4253, 2009.

[110] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3):497–515, 2004.

[111] Kari Karhunen. *Uber lineare Methoden in der Wahrscheinlichkeitsrechnung*, volume 37. Universitat Helsinki, 1947.

[112] BS Kashin. Widths of certain finite-dimensional sets and classes of smooth functions. *Izvestia an ISSR*, 11(41):334–351, 1977.

[113] EH Kennard. Zur quantenmechanik einfacher bewegungstypen. *Zeitschrift für Physik*, 44(4-5):326–352, 1927.

[114] Roger Koenker. *Quantile regression*. Number 38. Cambridge university press, 2005.

[115] Linghe Kong, Daqiang Zhang, Zongjian He, Qiao Xiang, Jiafu Wan, and Meixia Tao. Embracing big data with compressive sensing: a green approach in industrial wireless networks. *IEEE Communications Magazine*, 54(10):53–59, October 2016.

[116] David Koslicki, Simon Foucart, and Gail Rosen. Quikr: a method for rapid reconstruction of bacterial communities via compressive sensing. *Bioinformatics*, 29(17):2096–2102, 2013.

[117] Vladimir Aleksandrovich Kotelnikov. On the carrying capacity of the ether and wire in telecommunications. In *Material for the First All-Union Conference on Questions of Communication, Izd. Red. Upr. Svyazi RKKA, Moscow*, volume 1, 1933.

[118] Erwin Kreyszig. *Introductory Functional Analysis with Applications*, volume 81. Wiley New York, 1989.

[119] Doug Laney. 3D data management: Controlling data volume, velocity and variety. *META Group Research Note*, 6:70, 2001.

[120] Boyuan Liu, Wenhui Fan, and Tianyuan Xiao. *A Fast Outlier Detection Method for Big Data*, pages 379–384. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[121] Michel Loeve. Probability theory, vol. ii. *Graduate Texts in Mathematics*, 46:0–387, 1978.

[122] M Lustig, DL Donoho, and JM Pauly. Rapid MR imaging with compressed sensing and randomly under-sampled 3DFT trajectories. In *Proceedings of the 14th Annual Meeting ISMRM*. Citeseer, 2006.

[123] Michael Lustig, David Donoho, and John M Pauly. Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magnetic Resonance in Medicine*, 58(6):1182–1195, 2007.

[124] Michael Lustig, Jin Hyung Lee, David L Donoho, and John M Pauly. Faster imaging with randomly perturbed, under-sampled spirals and ℓ_1 reconstruction. In *Proceedings of the 13th Annual Meeting of ISMRM, Miami Beach*, page 685, 2005.

[125] Michael Lustig, Juan M Santos, Jin-Hyung Lee, David L Donoho, and John M Pauly. Application of compressed sensing for rapid mr imaging. *SPARS,(Rennes, France)*, 2005.

[126] Thomas R. Lynam, Charles L.A. Clarke, and Gordan V. Cormack. Information extraction with term frequencies. In *Proceedings of the First International Conference on Human Language Technology Research*, pages 1–4. Association for Computational Linguistics, 2001.

[127] Jianwei Ma. Compressed sensing for surface characterization and metrology. *Instrumentation and Measurement, IEEE Transactions on*, 59(6):1600–1615, 2010.

[128] Dana MacKenzie. Compressed sensing makes every pixel count. *Whats Happening in the Mathematical Sciences*, 7:114–127, 2009.

[129] Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(7):674–693, 1989.

[130] Stephane G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, 41(12):3397–3415, Dec 1993.

[131] Viktor Mayer-Schönberger and Kenneth Cukier. *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt, 2013.

[132] Dimitrios Milioris. Classification in twitter via compressive sensing. In *2015 IEEE Conference on Computer Communications Workshops*, pages 95–96, April 2015.

[133] Kumar Vijay Mishra, Anton Kruger, and Witold F Krajewski. Compressed sensing applied to weather radar. In *2014 IEEE Geoscience and Remote Sensing Symposium*, pages 1832–1835. IEEE, 2014.

[134] Deanna Needell and Joel A Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

[135] Deanna Needell and Roman Vershynin. Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Foundations of Computational Mathematics*, 9(3):317–334, 2009.

[136] Deanna Needell and Roman Vershynin. Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit. *Selected Topics in Signal Processing, IEEE Journal of*, 4(2):310–316, 2010.

[137] R. Niazadeh, M. Babaie-Zadeh, and C. Jutten. On the achievability of Cramer-Rao bound in noisy compressed sensing. *IEEE Transactions on Signal Processing*, 60(1):518–526, Jan 2012.

[138] Sérgio Nunes, Cristina Ribeiro, and Gabriel David. Term frequency dynamics in collaborative articles. In *Proceedings of the 10th ACM symposium on Document Engineering*, pages 267–270. ACM, 2010.

[139] Harry Nyquist. Certain topics in telegraph transmission theory. *American Institute of Electrical Engineers, Transactions of the*, 47(2):617–644, 1928.

[140] Ramón Pallás-Areny and John G Webster. *Analog Signal Processing*. John Wiley & Sons, 1999.

[141] Quan Pan, Lei Zhang, Guanzhong Dai, and Hongcai Zhang. Two denoising methods by wavelet transform. *Signal Processing, IEEE Transactions on*, 47(12):3401–3406, 1999.

[142] Yagyensh Chandra Pati, Ramin Rezaiifar, and PS Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44. IEEE, 1993.

[143] Charles Poynton. *Digitial Video and HD: Algorithms and Interfaces*. Elsevier, 2012.

[144] Lawrence Rabiner and Biing-Hwang Juang. Fundamentals of speech recognition, 1993.

[145] Lawrence R Rabiner and Bernard Gold. *Theory and Application of Digital Signal Processing*, volume 1. Prentice-Hall, Inc., 1975.

[146] J. Richy, D. Friboulet, A. Bernard, O. Bernard, and H. Liebgott. Blood velocity estimation using compressive sensing. *IEEE Transactions on Medical Imaging*, 32(11):1979–1988, Nov 2013.

[147] Justin Romberg. Imaging via compressive sampling [introduction to compressive sampling and recovery via convex programming]. *IEEE Signal Processing Magazine*, 25(2):14–20, 2008.

[148] M. Rossi, A. M. Haimovich, and Y. C. Eldar. Spatial compressive sensing for MIMO radar. *IEEE Transactions on Signal Processing*, 62(2):419–430, Jan 2014.

[149] Mark Rudelson and Roman Vershynin. On sparse reconstruction from Fourier and Gaussian measurements. *Communications on Pure and Applied Mathematics*, 61(8):1025–1045, 2008.

[150] Boris Yakovlevich Ryabko. Data compression by means of a book stack. *Problemy Peredachi Informatsii*, 16(4):16–21, 1980.

[151] David Salomon. *Data Compression: the Complete Reference*. Springer Science & Business Media, 2004.

[152] Fadil Santosa and William W Symes. Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1307–1330, 1986.

[153] Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

[154] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[155] Leslie N Smith. How to find real-world applications of compressive sensing. In *SPIE Defense, Security, and Sensing*, pages 87170Q–87170Q. International Society for Optics and Photonics, 2013.

[156] Steven Smith. *Digital signal processing: a practical guide for engineers and scientists*. Newnes, 2003.

[157] Jean-Luc Starck, Fionn Murtagh, and Jalal M Fadili. *Sparse Image and Signal Processing: Wavelets, Curvelets, Morphological diversity*. Cambridge University Press, 2010.

[158] Eric J Stollnitz, Tony D DeRose, and David H Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, 1996.

[159] Gilbert Strang. *Introduction to linear algebra*. Wellesley-Cambridge Press, 2011.

[160] Robert Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society Series. B (Methodological)*, 58:267–288, 1996.

[161] Andreas M. Tillmann and Marc E. Pfetsch. The computational complexity of the restricted isometry property, the nullspace property, and related concepts in compressed sensing. *IEEE Transactions on Information Theory*, 60(2):1248–1259, Feb 2014.

[162] Joel Tropp, Anna C Gilbert, et al. Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 53(12):4655–4666, 2007.

[163] Joel Tropp, Jason N Laska, Marco F Duarte, Justin K Romberg, and Richard G Baraniuk. Beyond Nyquist: Efficient sampling of sparse bandlimited signals. *Information Theory, IEEE Transactions on*, 56(1):520–544, 2010.

[164] Tomas Tuma and Paul Hurley. On the incoherence of noiselet and haar bases. In *Sampling Theory and Applications 2009*, 2009.

[165] A Valdivia, J Lopez-Alcalde, M Vicente, M Pichiule, M Ruiz, and M Ordobas. Monitoring influenza activity in Europe with Google flu trends: comparison with the findings of sentinel physician networks-results for 2009-10. *Eurosurveillance*, 15(29):2–7, 2010.

[166] Jan Van Leeuwen. *Handbook of Theoretical Computer Science: Algorithms and Complexity*, volume 1. Elsevier, 1990.

[167] S Vasanawala, M Alley, R Barth, B Hargreaves, J Pauly, and M Lustig. Faster pediatric MRI via compressed sensing. In *in Proceedings Annual Meeting of the Society of Pediatric Radiology (SPR), Carlsbad, CA*, 2009.

[168] Martin Vetterli and Cormac Herley. Wavelets and filter banks: Theory and design. *Signal Processing, IEEE Transactions on*, 40(9):2207–2232, 1992.

[169] Martin Vetterli and Jelena Kovacevic. *Wavelets and subband coding*. Number LCAV-BOOK-1995-001. Prentice-Hall, 1995.

[170] Martin Vetterli, Jelena Kovačević, and Vivek K Goyal. *Foundations of Signal Processing*. Cambridge University Press, 2014.

[171] Gregory K Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

[172] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions*, 13(4):600–612, 2004.

[173] Kenneth Harold Waters. *Reflection Seismology: A Tool for Energy Resource Exploration*. John Wiley and Sons Inc., New York, NY, 1987.

[174] Sholom M Weiss and Nitin Indurkhya. *Predictive Data Mining: a Practical Guide*. Morgan Kaufmann, 1998.

[175] Edmund Taylor Whittaker. On the functions which are represented by the expansions of the interpolation-theory. *Proceedings of the Royal Society of Edinburgh*, 35:181–194, 1915.

[176] N Wilson, K Mason, M Tobias, M Peacey, QS Huang, and M Baker. Interpreting Google flu trends data for pandemic h1n1 influenza: the New Zealand experience. *Euro surveillance: European Communicable Disease Bulletin*, 14(44):429–433, 2008.

[177] Eric P Xing, Michael I Jordan, Richard M Karp, et al. Feature selection for high-dimensional genomic microarray data. In *ICML*, volume 1, pages 601–608. Citeseer, 2001.

[178] Ying Yan, Jiaxing Zhang, Bojun Huang, Xuzhan Sun, Jiaqi Mu, Zheng Zhang, and Thomas Moscibroda. Distributed outlier detection using compressive sensing. Specialist Interest Group in Management Of Data '15, pages 3–16, New York, NY, USA, 2015. ACM.

[179] Jungang Yang, John Thompson, Xiaotao Huang, Tian Jin, and Zhimin Zhou. Segmented reconstruction for compressed sensing sar imaging. *IEEE Transactions on Geoscience and Remote Sensing*, 51(7):4214–4225, July 2013.

[180] Wotao Yin, Stanley Osher, Donald Goldfarb, and Jerome Darbon. Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. *SIAM Journal on Imaging Sciences*, 1(1):143–168, 2008.

[181] Jiaxing Zhang, Ying Yan, Liang Jeff Chen, Minjie Wang, Thomas Moscibroda, and Zheng Zhang. Impression store: Compressive sensing-based storage for big data analytics. In *6th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 14)*, 2014.

[182] Bin Zhao and Changsui Zhang. Compressed spectral clustering. In *2009 IEEE International Conference on Data Mining Workshops*, pages 344–349, Dec 2009.

[183] Ning Zhong, Yuefeng Li, and Sheng-Tang Wu. Effective pattern discovery for text mining. *Knowledge and Data Engineering, IEEE Transactions on*, 24(1):30–44, 2012.

[184] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.