Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

Master's Thesis

Iteratively Reweighted Algorithms for Dynamic MRI

HAKKEL Tamás
Computer Science Engineering MSc

2020

Supervisor: Claudio M. VERDUN
Faculty mentor: Dr. OLÁH András

Thesis Authenticity Statement

I, undersigned HAKKEL Tamás, student of the Faculty of Information Technology and Bionics at the Pázmány Péter Catholic University, hereby certify that this thesis was written without any unauthorized help, solely by me and I used only the referenced sources. Every part, which is quoted exactly or in a paraphrased manner, is indicated clearly with a reference. I have not submitted this thesis anywhere else.

HAKKEL Tamás

Abstract

Although MR imagig was invented less then 50 years ago, it has revolutionized medical imaging and diagnostic process as we know it. Its versatility makes it fit a wide range of use cases, and compared to other imaging technologies, MRI demonstrates important advantages in many cases, such as lack of ionizing radiation, adjustable contrast range, and excellent differentiation between soft tissues. However, it has also a disadvantage: the imaging speed is undesirably slow. While it is only inconvenience in some cases, dynamic images of chest, for example, gets blurred by the different motions of the body, like breathing and cardiac motion. There are new technologies that improves the imaging process itself, but strong image reconstruction algorithms can vastly improve the quality.

Among the many software techniques invented to improve image quality, compressed sensing is inevitably is one of the most impactful theoretical construction, introduced by Donoho, Candès, Romberg, and Tao in 2004 [1]–[3]. In contrast to the Nyquist-Shannon theorem that asserts that continuous band-limited signals can be perfectly reconstructed from samples taken at a rate of twice the highest frequency present in the signal of interest, compressed sensing allows lossless reconstruction from much lower number of samples given that certain natural conditions are satisfied.

In this thesis work, we consider the classic results as well as the recent advances within of the compressed sensing framework and their application to real life MR imaging. In particular, we closely examine two recent publications presenting state-of-the-art solutions combining conventional techniques with novel ideas.

Afterwards, we present our implementation of these algorithms along with the implementation of a recently invented algorithm from the family of iteratively reweighted least squares (IRLS) methods that previously have not been applied to MRI setting yet. For the language of implementation we have chosen Julia, a new open-source language released in 2012, as this language fits well the image reconstruction problem being inherently fast with a speed often comparable to C, and providing convenient environ-

ment for fast prototyping.

Finally, we compare these algorithm with respect to reconstruction power from massively undersampled data and noise tolerance. Our results demonstrates the fitness of the Julia language to prototyping and also shows strong benchmark suggesting that a robust extension of the new IRLS method can bring enormous improvement to reconstruction quality or imaging speed.

Contents

Chapter 1

Introduction

While the fast evolution of technology profoundly changed today's medicine, unarguably the medical imaging is of the fields which profited the most of the computation power recently became available. And as X-rays revolutionized medical treatments in the beginning of the 20th century, after its discovery by Wilhelm Conrad Röntgen, the appearance of computer-aided imaging techniques such as computer tomography (CT), diagnostic ultrasonography, positron emission tomography (PET), and magnetic resonance imaging (MRI) opened a new horizon drastically increasing the resolution, allowing 3D imaging, providing reliable dynamic recordings, and enhancing images by automated post-processing [4]. In the recent decades radiology evolved to be an interdisciplinary field involving, for instance, molecular biology, nuclear physics, applied mathematics, and computer science besides the classical medical fields such as anatomy, angiology, and cardiology.

1.1. Magnetic Resonance Imaging

In particular, MRI has revolutionized medical imaging and diagnostic process as we know it. Its versatility makes it fit a wide range of use cases. Compared to other imaging technologies, MRI demonstrates important advantages in many cases.

- In contrast to X-ray, MRI doesn't use any ionizing radiation, and hence it is totally harmless to the patient. Also, MRI has enhanced resolution for soft tissues compared to any other modality, in particular neural tissue, while X-rays are rather used for diagnosing bone degeneration, dislocation, fracture or some tissue infection. Furthermore, MRI allows 3D scans. Nevertheless, MRI scanners are slow and expensive compare to X-ray scan machines and therefore hardware and soft-

ware improvements are necessary for its wider use.

- As CT scanning is based on X-rays, it shares the downside with X-rays, doctors need to evaluate the possible benefits of the scan and decide if it outweighs the potential complications of exposure to ionizing radiations. MRI, however, elicit this problem, although at the price of a elongated imaging process. Comparing the medical problems where these technologies are used, one can conclude that CT scan is very helpful in diagnosing severe injuries of the chest, head, spine or abdomen, particularly fractures, and it is commonly used to localize tumors. MRI often performs better at diagnosing problems in the joints, soft tissues, ligaments and tendons. When available, doctors use it frequently to scan the spine, brain, muscles, neck, breasts, and abdomen.

- MRI still does not have the portability, low cost, and real-time imaging speed without any harmful radiation of ultrasound technology, but ultrasound is mostly limited to 2D imaging (although 3D imaging is possible), have trouble penetrating bone, and even in absence of bone the depth of penetration is limited depending on the frequency of imaging. This is not the case for MRI technology.

- PET scans are particularly useful for functional imaging. For instance, it is used for identification of lapses in cognitive function, examination of cardiac failures, cancer screening and diagnosis, and finding an infection. Nevertheless, PET image acquisition is even longer than MRI (especially, if we consider also the time while patients wait for the tracer to reach the targeted organ), it uses a radioactive substance as tracer, and it cannot scan tissues not absorbing the tracer making the localization of the source of the signal infeasible without any additional information. In order to solve the latter described limitation, one particularly promising combination is the join use of PET and MRI. This illustrates that MRI is a fundamental technology not only by itself but also when combined with other imaging modalities.

To sum up, MRI is a strong competitor to other imaging technologies, but it also have weaknesses, of which the costs and scanning time are the most remarkable. There are many methods to speed up measurements as it will be discussed later, but the construction cost and the hardware constraints limit the applicability of these efforts. The problem of slowness is even more apparent in case of dynamic images as motion of organs (e.g. heart or lung) can drastically degrade the image quality. To overcome that

issue, mathematical solutions developed in the last twenty years such as *parallel imaging* and *compressive sensing* made high-resolution and fast images possible. This thesis concerns the second of these two powerful mathematical ideas.

1.2. Compressed Sensing

Among the many software techniques invented to improve image quality, compressed sensing (also known as compressive sensing, compressed sampling, and compressive sampling) is inevitably is one of the most impactful theoretical construction, introduced by Donoho, Candès, Romberg, and Tao in 2004 [1]–[3]. Such importance can be seems by the fact that the four foundational papers of compressed sensing received, at the time of writing this manuscripts, more than 60000 citations. In contrast to the Nyquist-Shannon theorem that asserts that continuous band-limited signals can be perfectly reconstructed from samples taken at a rate of twice the highest frequency present in the signal of interest, compressed sensing allows lossless reconstruction from much lower number of samples given that certain natural conditions are satisfied.

This impressive improvement is due to the same phenomenon that makes modern image compression algorithms so successful: the sparsity of the signal to be recovered in a certain representation domain. And while the classic image processing flow starts with acquiring the fully sampled image, then feeding it to a compression algorithm that discards the vast majority of the data still allowing later a lossless decompression (e.g. JPEG or JPEG2000), the idea behind compressed sensing is that image acquisition can be made much more effective by fusing it with the compression step recording only the data we need later for decompression, hence the name compressed sensing. As will be discussed in this thesis, MRI possess natural sparse representation and due to physics of the nuclear magnetic resonance phenomenon, its acquisition process is dictated by Fourier transforms which makes it a perfect candidate for the use of compressed sensing machinery. Indeed, MRI was the first successful application of compressed sensing [5] and, since 2017, MRI scanner employing this technology are approved by the American Food and Drug Administration and commercially available [6], [7].

As a trade-off for the acceleration in the scanning time, the posterior process of reconstructing the image from the measured data is much more involved compared to the standard one typically used when longer scans, i.e., fully sampled Fourier measurement, are performed. Therefore, a large amount of theory was developed since the introduction of compressed sensing to further improve the recovery of a high resolution

image from the compressed representation. Ideas coming from high-dimensional statistics, non-linear optimization, harmonic analysis and signal processing came together in order to develop robust, stable and scalable reconstruction methods for compressed sensing. These ideas are particularly useful when applied to the MRI field since the minimization problems with its associated cost functions associated tend to be very challenging. Here, a few of those modern ideas will be discussed, in particular, *accelerated proximal methods*, *augmented Lagrangian methods* and *iteratively reweighted least squares*

1.3. Julia Language

Data scientists often find themselves in an very uncomfortable dilemma: Do they choose high-abstraction level dynamic languages that allows rapid prototyping and easy debugging, but fails to perform well in large-scale problems; or should they opt for high performance languages with lower abstraction level, spending much more time with the implementation. The rising popularity of Python in the recent decades also due to this problem. Python, however, instead of solving the problem, get around it by being merely a glu-language that connects high-performance libraries usually written in C or Fortran. The problem therefore is still present, as one still needs to write the performance-critical parts in a lower-level language. To give a better solution to that problem, Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman started to work on a brand new language in 2009. Their objective is bold, writing in their first blog post that they created Julia because...

> ...we want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled. [8]

As of 2020, we can say that their mission statement was not completely science fiction since the popularity of the language does not seems to decline ever since; in con-

trary, the number of users increases in a fast rate, especially in the academia and the field of data science. And as the objective of this thesis project involves heavy computing, the chosen language to implement the algorithms of interest is became Julia.

1.4. Objective

In this thesis work, we consider the classic results as well as the recent advances within of the compressed sensing framework and their application to real life MR imaging. In particular, we closely examine two recent publications presenting state-of-the-art solutions combining conventional techniques with novel ideas. Afterwards, we present our implementation of these algorithms along with the implementation of a recently invented algorithm from the family of iteratively least squares methods that previously have not been applied to MRI setting yet. Finally, we compare these algorithm with respect to reconstruction power from massively undersampled data and noise tolerance.

1.5. Outline

The structure of this thesis is as follows:

Chapter 2 introduces the reader to the core concepts of MRI, presents the hardware improvements of the last decades, and show the limits of further hardware improvements.

Chapter 3 summarizes the most important conditions and promises of compressed sensing framework, and then gives a quick overview of the most popular first order methods currently used in in the field.

Chapter 4 reviews three recent publications. Two of these papers proposes algorithms which are currently state-of-the-art algorithms for reconstruction of dynamic MR images, and the third one presents an algorithm that has stronger guarantees than the currently used solutions, and therefore a good candidate for reconstruction from even more reduced data.

Chapter 5 lists our contributions to the field presenting the results of numerical experiments.

Chapter 5 summarizes the project and articulates the future plans.

Chapter 2

Theoretical Background of MRI

2.1. History of Medical Imaging

The beginning of the history of medical images dates back to November of 1895, when Wilhelm Conrad Röntgen discovered X-rays. The significance of his discovery is well demonstrated by the fact that up until the 1960's X-rays were the only non-invasive way to look into the body, and hence he was awarded the first Nobel Prize in Physics in 1901 for *"in recognition of the extraordinary services he has rendered by the discovery of the remarkable rays subsequently named after him"* [9]. During the first 60 years of X-ray radiography, it underwent a remarkable development gradually increasing the resolution of images and decreasing the radiation dose that threatened both the patients' and the doctors' health. Since the 1920s, visualization of motions (i.e. dynamic imaging) became possible by fluoroscopy, although only to a limited extent [10].

Although the theoretical background of later advances in medical images were present much earlier, e.g., by the seminal work of Johann Radon, for the introduction of modern imaging technologies to clinical medicine are delayed to the time when the computers became powerful enough for image reconstruction tasks concerning both the computational speed and the available memory. Therefore, the major breakthrough came only in the 1960-70s, and then suddenly multiple imaging techniques were introduced shortly after each other.

The first of these methods was the ultrasound imaging. It was devised by Floyd Firestone in 1940 to detect internal flaws in metal castings [11], and it was proposed for medical purposes first in 1949 by John Wild [12], but it was not until 1961 when David Robinson and George Kossoff developed the first commercially practical water path ultrasonic scanner [13].

Similarly, the concept of emission and transmission tomography was proposed in the late 1950s by David E. Kuhl, Luke Chapman and Roy Edwards, but computer tomography was invented only in 1972 by Godfrey Hounsfield and Allan Cormack [14], and the first PET camera was built for human studies by Edward Hoffman, Michael M. Ter-Pogossian, and Michael E. Phelps in 1973 [15].

Finally, the youngest imaging technology, the MRI, became available for diagnostics by the end of the 1970s, even though the nuclear magnetic resonance (NMR) spectroscopy that has the same mechanism was discovered by Felix Bloch and Edward Purcell already in 1946. Then Reymond Vahan Damadian proposed the first MR body scanner in 1969, and soon Paul Lauterbur had the idea of applying magnetic field gradients in all three dimensions and a back-projection technique to create images in 1971, and he also published the first MRI images: water tubes, a living clam, and the thoracic cavity of a mouse in 1973 and 1974 [16].

Since their appearance, these computer-aided methods continue to develop at a very fast pace, becoming essential tools for today's practitioners despite the fact that they are relative new technologies.

2.2. Physics of MRI

This section attempts to dive into the physics of MRI giving a quick overview of the theory of nuclear magnetic resonance following [17]–[19].

2.2.1 Components of MRI machines

The theory of measurements based on nuclear magnetic resonance has its root in quantum physics: The nuclear magnetic moment and the angular momentum of protons in the atomic nuclei maintained by the spin of these particles are to be indirectly measured, and these observables depend (besides many other factors) on the tissue where the proton is located. More specifically, the MRI machines are tuned to focus on the nucleus of protons that consist of only one proton. The core components of MRI machines are the following:

1. Superconductive coils immersed in liquid helium are the largest and most expensive part of the machine. They are responsible for producing a almost perfectly homogeneous and static magnetic field. The role of the liquid helium is to keep the wires at superconducting temperature, so that massive amounts of electricity

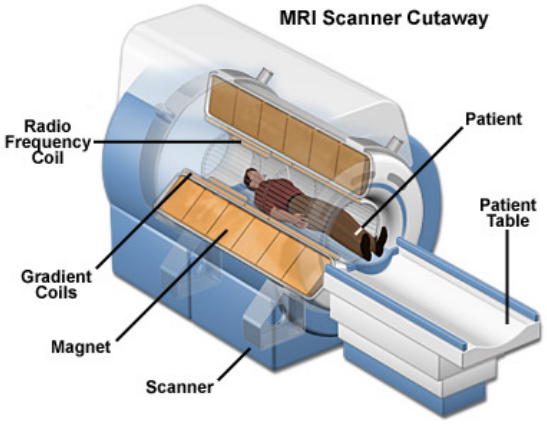

Figure 2.1: Schematic illustration of construction of a cylindrical MRI scanner. Source: [22].

can be run through the coils creating super-strong fields up to $21.1\,\mathrm{T}$ [20]. Although stronger magnetic field allows better resolution, the construction costs of such machines and the effect of the strong magnetic field on human tissues limit the strength of available MRI scanners for routine clinical from $0.2\,\mathrm{T}$ to $3.0\,\mathrm{T}$, and up to $11.7\,\mathrm{T}$ in research machines for human imaging [21].

2. Inside of this super-strong electromagnet, the so called gradient coils are located that alter the field along all three dimensions creating spatially varying magnetic field (hence the name: gradient coils) in order of mT, so that signals coming from different location within the coils are possible to be separated. They are also used to provide contrast for diffusion and flow imaging.

3. Within the Radio Frequency (RF) coils are located that emit and measure time varying electromagnetic signals on order of tens of $\mu\mathrm{T}$.

The reason behind this elaborate design (depicted on fig. 2.1) is the need of creating a measurement setup suitable to give a very fine control over the the direction of the magnetic moment of protons of hydrogen atoms within the measured object (which, in our case, is the human body that contains a large amount of hydrogen mostly in the form of water, but also bounded within other molecules).

2.2.2 Macroscopic Magnetization

The purpose of the superconductive coils is to align the magnetic moment of protons with the direction of the magnetic field. This direction (also corresponding to the head-to-foot direction) is usually referred to as longitudinal direction or z direction, and the plane perpendicular to this direction is called the transverse plane or the x-y plane. This

alignment of the magnetic moment of the protons leads to two configurations: protons with their magnetic moment pointing to the same direction as the static magnetic field, and other protons having their magnetic moment with opposite direction. Without the static field, the randomly oriented spins cancel out each other, as they also do in the aligned case, when the number of protons oriented to the two directions are equal. But in real systems, a slight excess of the protons aligned with the static magnetic field always produces a net magnetization with the same direction as the external magnetic field (see fig.2.2).

The ratio of the number of protons in these two groups are described by the Fermi-Dirac statistics. In strong and static magnetic field at room temperature, the Fermi-Dirac distribution reduces to Boltzmann distribution resulting the following formula:

$$N_+ = N \cdot \frac{e^{E_+/(k_B T)}}{e^{E_+/(k_B T)} + e^{E_-/(k_B T)}} \text{ and } N_- = N \cdot \frac{e^{E_-/(k_B T)}}{e^{E_+/(k_B T)} + e^{E_-/(k_B T)}},$$

where N is the total number of protons, N_+ and N_- are the numbers of protons pointing to the same and opposite direction as the static magnetic field, E_+ and E_- are their respective energy levels, k_B is the Boltzmann-constant, and T is the temperature. In this case neighboring energy levels are equidistant with the difference in the secondary spin quantum number of $\Delta m = \pm 1$ and the energy difference of $\nabla E = \gamma \hbar B_0$, where γ is an empirical constant called gyromagnetic ratio (equals to $42.575 \cdot 2\pi \mathrm{MHz/T}$ in case of protons), $\hbar$ is the reduced Planck constant, and B_0 is the static external magnetic field. The ratio of Boltzmann distributions for two states a spin ½nucleus is known as the Boltzmann factor:

$$f(E) = e^{-\frac{\gamma \hbar B_0}{k_B T}}.$$

Using this factor, the ratio of unpaired protons (these protons give the net magnetization) divided by the number of all protons is given by

$$\frac{N_+ - N_-}{N_+ + N_-} = \frac{\gamma \hbar B_0}{2 k_B T}.$$

This ratio at room temperature in a static field with a couple teslas is a tiny number (in the order of 1×10^{-6} multiplied by B_0), so that explains why do MRI machines need such a strong electromagnets. (Note that this ratio also can be increased by increasing the temperature, but it is not feasible for human imaging.)

2.2.3 Precession

In equilibrium when all protons are aligned with the external magnetic field, the longitudinal component of net magnetization is maximal and the component in the trans-

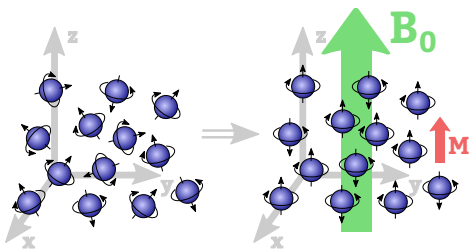

Figure 2.2: Effect of strong external magnetic field (B_0): Spins of protons get aligned with the field in either parallel or anti-paralellel direction producing a net magnetization (M) parallel with the external field.

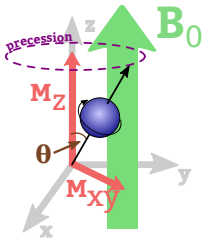

Figure 2.3: Precession of protons. As a result of RF excitation pulse, the magnetic momentum of protons deviates from the longitudinal direction and starts to precess due to its angular moment.

verse plane is zero. However, with the aid of an electromagnetic excitation in the transverse plane emitted by the RF coils, it is possible to rotate the vector of net magnetization into the transverse plane. The key factor in this process is to tune the frequency of the excitation to match the so called precessional frequency of the protons given by the Larmor equation:

$$\omega = \gamma B_0.$$

The name *precessional frequency* comes from the phenomenon that the magnetic moment of protons start to precess around the longitudinal axis (which, again, is the direction of static external magnetic field) due to its intrinsic angular momentum. When the frequency of the excitation matches the precessional frequency of the proton (which happens to be in the radio frequency range, hence the name of RF coils), then resonance occurs and the angle of net magnetization gets tilted (illustrated by fig. 2.3), otherwise the electromagnetic field has little to no effect of the net magnetization. An RF excitation of a duration τ causes rotation of the magnetization by an angle θ, which is called the flip angle, defined by

$$\theta = \gamma \int_0^\tau B_1(t)dt = \gamma \tau B_1,$$

where B_1 is the magnetization of RF excitation, and it is assumed to be constant over time window of excitation with length τ.

As a result of the precession, the net magnetic flux changes in the RF coils (these coils used for both emitting and receiving RF signals) inducing an electromotive force U_{ind} that can be calculated by Faraday's law of induction:

$$U_{ind} = -\frac{d\Phi}{dt}.$$

Projecting the precessing movement (with the Larmor frequency ω) of the net magnetization to transversal plan, we get a sinusoidal change in flux that results in the following

formula:

$$U_{ind} \sim sin(\theta)\,\omega\,cos(\omega t) = sin(\theta)\,\gamma\,B_0\,cos(\gamma\,B_0\,t).$$

While this formula is not an exact model that fits the current measured in the RF coils, but it captures three important aspects of the resulted electric signal:

- It is a sinusoidal signal with a frequency depending only on a constant specific to protons and the external magnetic field.
- The amplitude of that signal depends on the flip angle induced by an RF excitation.
- And it is also dependent on the external magnetic field (yet another reason why MRI machines need very strong electromagnets).

2.2.4 Relaxation

For a more accurate model, one should consider that as the protons emit RF signal due to their precessing magnetic moment, they lose the energy of the excitation and they slowly return to the low energy state; i.e., to the state where the magnetic moment of protons are aligned along the longitudinal axis, and where the net magnetization points to the same direction as the external field. Assuming that the excitation resulted in a perpendicular flip angle, the longitudinal component of magnetization is characterized by the exponential formula

$$M_z = M_0(1 - e^{-t/T_1}),$$

where M_0 is the amplitude of magnetization in the equilibrium and is often called Boltzmann magnetization, and T_1 time constant is a property of the protons dependent on the tissue where they are located. The name of this process is T_1 relaxation.

Furthermore, the net magnetization is also affected by another relaxation process called T_2 relaxation. The phenomenon causing this relaxation is called *dephasing*, and the name comes from the fact that when excitation is applied to protons in the equilibrium, they will precess in the same phase, but soon they lose this synchronization. This desynchronization is due to the slight inhomogeneity of the static external field caused by four factors: spin-spin interactions (quantum mechanical interactions with the nearby protons), magnetic field inhomogeneities (hardware limitations), magnetic susceptibility (slight magnetization of molecules within the measured part of the body), and chemical shift effects (shielding effect of the electron cloud of molecules incorporating the hydrogen atoms). The slightly different B_0 value makes the Larmor frequency

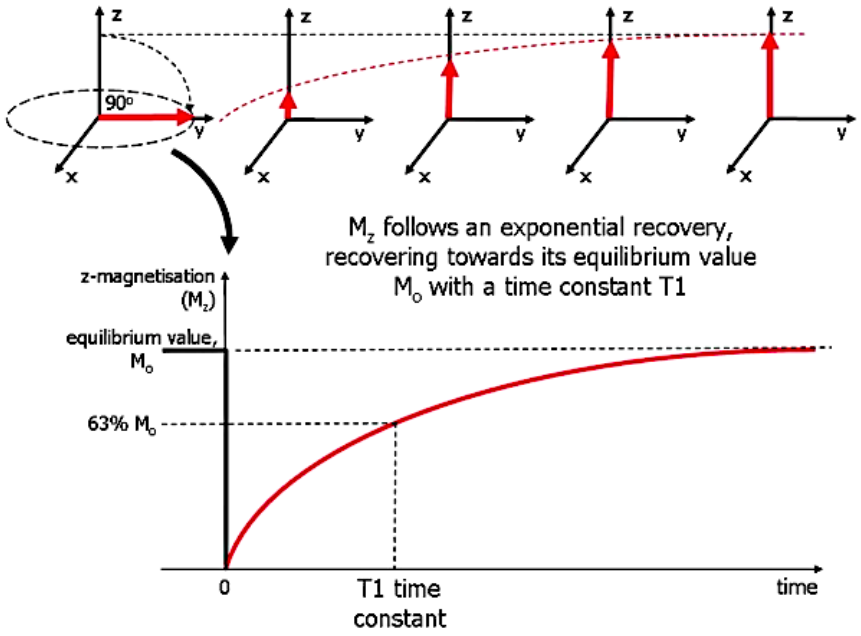

Figure 2.4: T_1 **relaxation after a** $90°$ **RF excitation.** By the end of the pulse, the magnetization rotated from the z-direction to the x-y plane; therefore, the longitudinal component is reduced to zero, and gradually it returns to the equilibrium. Source: [23].

different, and that results in the desynchronization of phase. The outcome of this process is that the transversal component of the net magnetization decays exponentially to zero. The speed of decay is characterized by the T_2^* time constant:

$$M_{xy} = M_1 \, e^{-t/T_2^*},$$

where M_1 is the initial amplitude of net magnetization in the beginning of the T_2 relaxation process. For illustration of this process, see fig. 2.4. The resultant decaying signal is known as the Free Induction Decay (FID). Using, however, the later described *spin-echo* acquisition protocol, the last three inhomogeneity-causing factors can be cancelled out leading to a slightly different time constant denoted by T_2.

2.3. Concepts of MR Imaging

In that section, the most important concepts of MR imaging are summarized, clarifying the vocabulary used in the later chapters, based on [17], [19].

2.3.1 MRI Sequences

Since the advent of MRI, numerous methods were developed and are used in today's medicine. And while they are all measure somehow the T_1 and T_2 constants at dif-

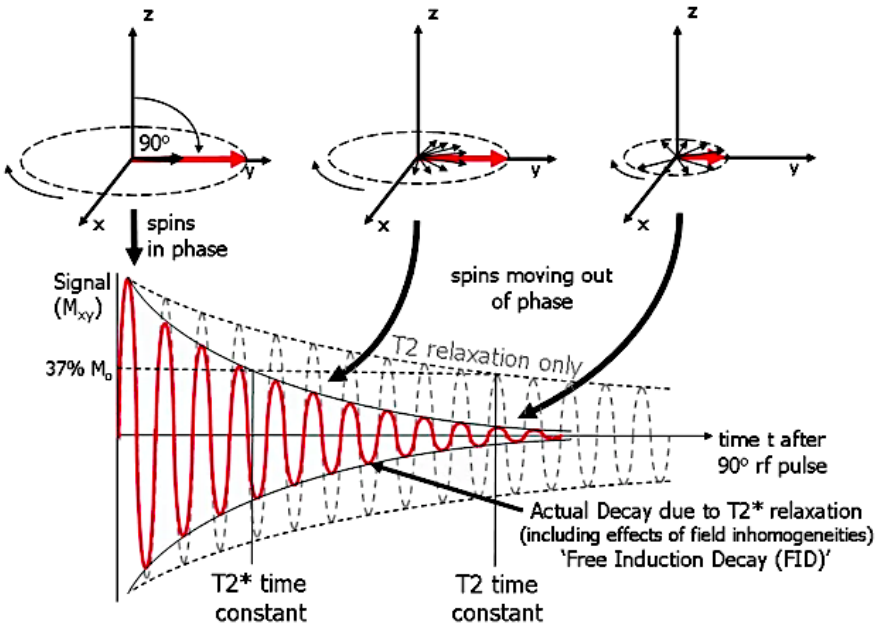

Figure 2.5: T_2 **relaxation process.** Right after the RF pulse, thre precession of all protons are in the same phase, but they quickly desynchronize due to magnetic field inhomogeneities. The plot in the second row depicts the strength of current induced in the RF coils that corresponds to the projection of the x-y component (M_{xy}) of the net magnetization along the direction perpendicular to the surface of the coil. And as a result of the T_2 relaxation, the amplitude of this sinusoidal curve exponentially decays. The resultant decaying signal is known as the Free Induction Decay (FID). Source: [23].

ferent location, the produced image is quite different, making them fit different use cases. These methods are called MRI sequences and they mostly differ in the a particular setting of RF pulses and the gradients in the static magnetic field, resulting in a particular image appearance. The most commonly used group of MRI sequences is the *spin echo* [24]. In accordance with the two types of relaxation, sequences in that group have two main parameters: TR (Time of Repetition) and TE (Time of Echo). These parameters have a crucial role timing the recording of current in the RF coils when the difference between the amplitude of the RF signal emitted by the excited protons is maximal because this difference makes it possible later to distinguish different tissues.

The TR parameter is connected to the T_1 value, as it determines the time between two excitation pulses. Having a larger TR value allows protons to get better aligned with the external magnetic field before the next excitation, which results in a higher initial value for M_{xy}, leading to a stronger current in the detector coils, but it also makes the entire measurement longer. On the other hand, TE determines the delay between the peak of the RF pulse and the peak of the echo. That echo is a temporary rephasing of spins caused by a second, 180° RF pulse emitted at $t = TE/2$. That pulse inverts spins,

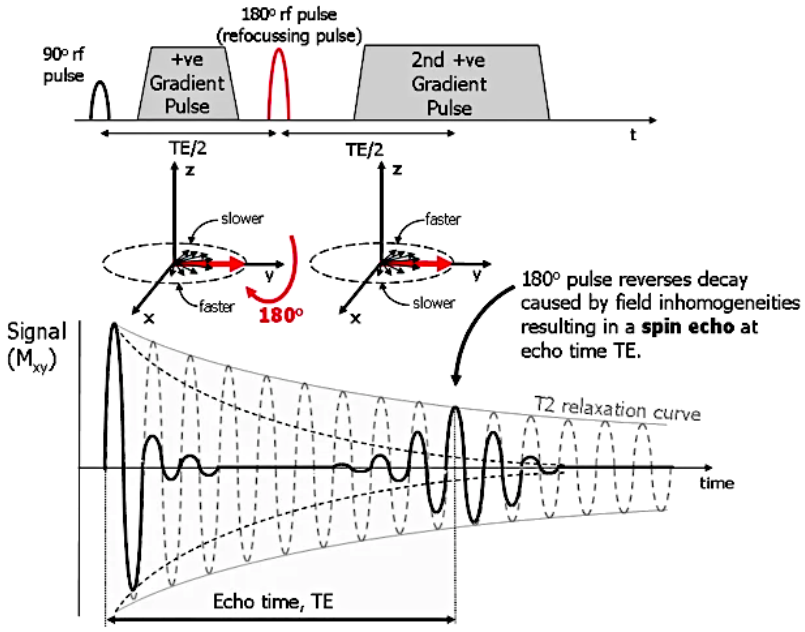

Figure 2.6: Process of spin echo sequence. The echo is a temporary rephasing of spins caused by a second, 180° RF pulse emitted at $t = TE/2$. That pulse inverts spins, and therefore it makes spins with slower Larmor frequency, which lagged behind the faster ones previously, be ahead of the others in phase. At the time when faster precessing protons catch up, the transversal magnetization exhibits an echo peak. Source: [23].

and therefore it makes spins with slower Larmor frequency, which lagged behind the faster ones previously, be ahead of the others in phase. At the time when faster precessing protons catch up, the transversal magnetization exhibits an echo peak (see 2.6). As stated earlier, an important advantage of spin echo technique is that three factors of magnetic inhomogeneity is cancelled out by the inversion as these factors are constant over time, while spin-spin interactions are random interactions between protons that cause random local changes in the magnetic fields experienced by the protons.

Before moving forward, an important thing to note that the T_1 relaxation is much slower than the T_2 relaxation (T_1 relaxation takes hundreds of milliseconds up to a few seconds while T_2 rarely exceeds $200\,\mathrm{ms}$. As a result, the acquisition time is mostly dominated by waiting for the T_1 relaxation, and therefore short TR values are favorable when fast imaging is needed. Also, the different time-scale of T_1 and T_2 relaxation opens a range of possibilities to make acquisition process faster or more effective.

Based on the choice of TR and TE values, we can talk about three types of spin echo sequences: T_1 weighted sequence has intermediate TR value in the magnitude of T_1 producing maximal T1 weighting (at this point, the difference caused by different T_1 value between the amplitude of signals coming from different tissues are maximal) and

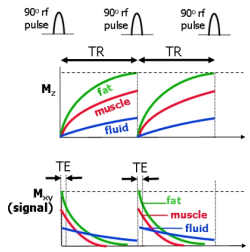

Figure 2.7: T_1 weighted sequence. Choosing the TR value to be relatively short, and TE to be relatively short, the difference due to the variation of T_1 value over tissues would dominate over differences caused by different T_2 value. Source: Adapted from [23].

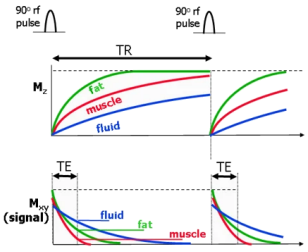

Figure 2.8: T_2 weighted sequence. Choosing both TR and TE values to be relatively long, the difference due to the variation of T_2 would dominate over differences caused by different T_1 value. Source: Adapted from [23].

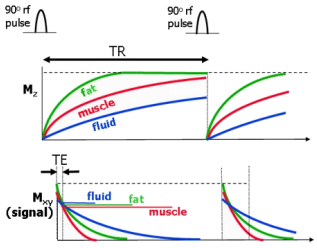

Figure 2.9: Proton density (PD) weighted sequence. Choosing the TR value to be relatively long, and TE to be relatively short, the difference due to the variation of both T_1 and T_2 values are minimized, and hence mostly the density of protons would determine signal strength. Source: Adapted from [23].

short TE value magnitudes smaller than T_2 producing minimal T2 weighting (there is not enough time to have significant difference between decay curves with different T_2). To the contrary, T2-weighted images have a long TE (maximizing the difference in T_2 relaxation) and long TR (reducing the weight of T_1 relaxation). And the third type, called proton density (PD) weighting, uses short TE and long TE, so that the pixel intensities on the resulted image will reflect only the density of protons (that also differs between tissues), and the T_1 and T_2 values have little effect on it.

Two common variant of spin echo are multiecho spin-echo, and turbo spin-echo. The multiecho spin-echo pulse sequence utilizes multiple 180° RF pulses to induce multiple echo peaks each with a different TE, forming multiple images of the same object with different weighting ranging from PD-weighting to T_2-weighting. This method exploits the fact that T_1 is much larger than T_1, thus multiple echos can be performed without drastically changing the acquisition time. Similarly, turbo spin-echo sequences consist of multiple echo-generating 180° pulses, but in this case only one image is formed speeding up the imaging by gathering information about multiple positions in each cycle.

The other large group of sequences is the gradient echo sequence [25]. That type of sequences differs from echo-spin that the flip angle of initial RF pulse is less than 90° (e.g., 20° or 30°) and there is no 180° secondary pulse, instead it induces an echo by the spacial gradients explained later. Hence this method is able to perform the measurement much faster as T_1 relaxation reaches near-equilibrium state much earlier.

Beyond these sequence types, several other commonly used variants exist, which will not be discussed here, such as the inversion recovery sequences [26]–[29], diffusion-weighted sequences [30], [31], perfusion weighted sequences [32]–[34], BOLD-contrast images for functional MRI (fMRI) [35], [36].

2.3.2 Spatial Encoding

The core concept that allowed the extension of NRM (which is based on the same principles described above) to MRI, is the spatial encoding. This technique, proposed by Paul Lauterbur, allows for the localization of RF emitting protons or, more precisely, the localization of an *ensemble* of RF emitting protons within a small volume called voxels (note that the size and the shape of these voxels are defined by the configuration of MRI machine for the given acquisition). Specifically, the problem with static magnetic field is that, even though the net magnetization varies over the measured object based on the type of the tissue, the RF pulse excites the entire volume of the measured object, and therefore the induced signal of each voxel sum up making it impossible to separate them based on their position. In contrast, generating a secondary magnetic field with a gradient along a specific direction makes the Larmor frequency dependent on the position along that direction. That dependence can be exploited multiple ways allowing exact localization along all three dimensions. A possible (and quite common) way to do this is the following:

1. Producing a gradient along the z-direction during the RF excitation permits the selection of a slice perpendicular to the z-axis by tuning the RF pulse to the frequency specific to the given slice because this pulse excites then only protons in the selected slice. In reality, however, not a single frequency used but rather a band of frequencies whose width matches the bandwidth of resonance frequencies of spins in the slice of interest. This approximately rectangular band of excitation frequencies is realized in time domain by a pulse of a shape similar to the sinc function.

2. Application of another gradient along the y-direction between the RF excitation and the readout of induced current causes a gradual de-synchronization of phase along the y-axis because protons experiencing higher external field will have a higher Larmor frequency as well. Therefore, the position along y-direction becomes *phase-encoded*.

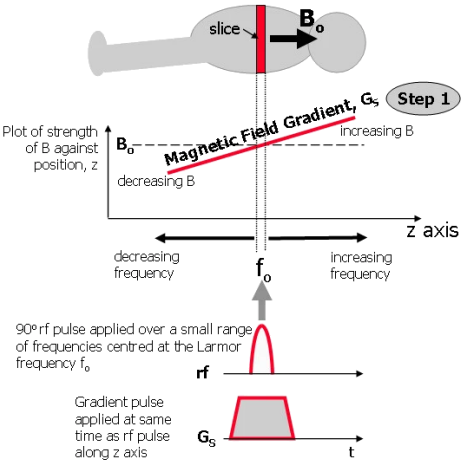

Figure 2.10: Mechanism of slice selection. Producing a gradient along the z-direction during the RF excitation permits the selection of a slice perpendicular to the z-axis by tuning the RF pulse to cover the frequency range specific to the given slice with given thickness because this pulse excites then only protons in this slice. Source: [23].

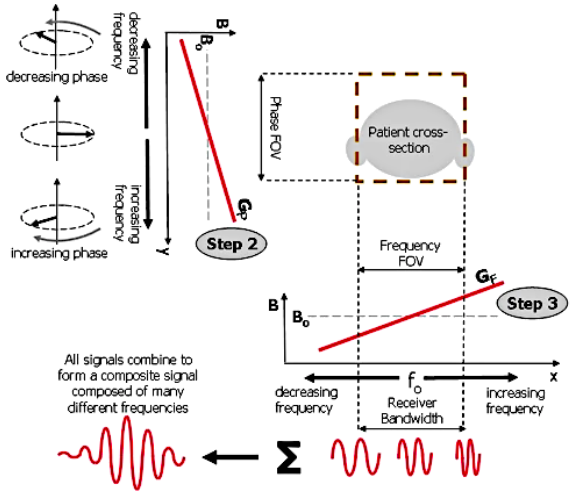

Figure 2.11: Process of phase and frequency encoding. First, a gradient along the y-direction between the RF excitation and the readout causes a gradual de-synchronization of phase along the y-axis because protons experiencing higher external field will have a higher Larmor frequency. Second, a gradient along the x-direction can be used to separate signal sources along that dimension by the difference in their frequency. Source: [23].

3. During the time window of readout, another gradient along the x-direction can be used to separate signal sources along that dimension by the difference in their frequency. This type of spacial encoding is called *frequency encoding*.

These steps are visually explained on fig. 2.10 and 2.11.

Decoding the spatial information becomes feasible then using the famous Bloch equation:

$$\frac{d\mathbf{M}}{dt} = \mathbf{M} \times \boldsymbol{\gamma}\mathbf{B} - \frac{M_x\mathbf{i} + M_y\mathbf{j}}{T_2} - \frac{(M_z - M_0)\mathbf{k}}{T_1}.$$

Knowing that the transversal component of the net magnetization (that is, the measured component) is non-zero only a selected slice and is dependent on the position within the slice assuming the excitation strategy above, a convenient formulation is to use a complex-valued function to denote the amplitude of the signal generated at position (x, y) by $m(x, y) = m_x(x, y) + m_y(x, y)$, where $m_x(x, y)$ and $m_y(x, y)$ is the amplitude realized in a coil perpendicular to x-axis and y-axis, respectively. As a result of the phase-encoding, the phase of a spin at position y along y-axis is given by $\omega_0 t_y + \gamma G_y t_y y$, where ω_0 is the base Larmor frequency of the selected slice, t_y denotes the width of time window of phase-encoding, and G_y corresponds to the slope of the linear gradi-

ent applied along the y-direction. Because $m(x, y)$ is a complex-valued function representing the magnetization in both the x and y-direction, the distribution $m(x, y)$ gets weighted by a complex exponential corresponding to the spatial frequency $\gamma/2\pi)G_y t_y$: $m(x, y)exp(-i\gamma G_y t_y y$.

Similarly, the shift in the frequency of precession during the readout characterized by $\gamma G_x x$ adds another weighting term to $m(x, y)$. Solving the Bloch equation for this configuration results in the following formula:

$$s(t) = \int_x \int_y m(x, y) e^{-i\gamma G_x x t} e^{-i\gamma G_y y t_y} dx dy,$$

where s is the amplitude of signal to be measured, t_y (length of time window for phase-encoding) is a fixed number and t is a running variable. Having a look at the formula, one can immediately see that it is the 2-dimensional Fourier transform of $m(x, y)$ corresponding to the coordinate $((\gamma/2\pi)G_x t, (\gamma/2\pi)G_y t_y)$ in the Fourier space. Due to the discrete nature of this the acquisition process, the MRI machines can evaluate the Fourier space in discrete positions, that is, the collected data is the *discrete Fourier transform* (DFT) of the object. Consequently, the space of measurements is usually referred to as k-space. This connection between the measured signal and Fourier transformation is beneficial in many aspects, as it will be apparent in the next sections.

2.3.3 Sampling Trajectories

The effect of the spatial encoding schema discussed above is that the k-space points can only be measured sequentially. Combined with the necessity of waiting for the T_1 relaxation (that tends to be in the timescale of $250\,\mathrm{ms}$ to $1500\,\mathrm{ms}$ at $3\,\mathrm{T}$, and somewhat shorter for $1.5\,\mathrm{T}$ [37]–[39]), it leads to the most important limitation of MR imaging: the acquisition is quite slow. While it is inconvenient for the patient to stay inside the narrow scanner bore (especially for claustrophobic patients), the more important issue is that the longer is the acquisition the more motion artifacts are introduced to the image. The type of such involuntary motions include bulk motion (e.g., coughing, change body position), respiratory motion, cardiac motion, and motion of other organs like blood vessels or parts of the gastrointestinal tract. Although some of them can be reduced by a large extent, for example, by asking patient to pay attention to remain still or to hold breath for a $10\,\mathrm{s}$ to $20\,\mathrm{s}$ long measurement, others are out of control. Reducing the k-space points, however, lead to various aliasing effects according to Nyquist-Shannon sampling theorem that might blur clinically relevant features or introduce misleading

artifacts. Therefore, radiologists always seek to find an optimal compromise between the number of measured k-space points and the amount of motion artifacts introduced.

One important aspect of dealing with that problem is determining which points are to be measured. Due to hardware limitations and because too rapidly changing electromagnetic field would overheat the measured tissues, arbitrary positioning in k-space is not feasible, and thus the order of measurement points are also a key aspects of the process. These constrains introduces the necessity of well-defined geometries describing the location and the order of measured k-space coordinates, called sampling trajectories.

The traditional way of acquiring K-space data is through Cartesian trajectories. The concept of these trajectories is that equally spaced grid-points are selected in the slice or volume to be measured, and these points are evaluated systematically; for instance, row-by-row, zig-zag, or spirally. The main benefit of this method that a simple and fast reconstruction is possible via fast Fourier transform (FFT). Nevertheless, this method fails to exploit a very important feature of natural images in general: the energy tends to be concentrated in the center of the Fourier space; that is to say, the coefficients of lower frequencies have usually large magnitudes, and points further from the center have mostly a near-zero value. The significance of this observation is perfectly demonstrated by the success of the modern image and video compression algorithms making use of this uneven distribution, for instance, JPEG, MP3, and standards. Hence, multiple non-equidistant sampling trajectories are also used in practice. Such trajectories are radial [40], spiral [41], concentric rings [42], and 3D cones trajectories [43], just to name a few. A couple 3-dimensional trajectories are depicted in fig. 2.12.

2.3.4 Accelerated MRI

For a deeper understanding of accelerating methods, one needs to get familiarized with the concept of field of view and its connection to the bandwidth of RF excitation pulse. The term *field of view* (FOV) refers to the area or volume over which an MR image is acquired, and often also to its size as well. Due to the discrete nature of digital systems, the FOV is also discretized representing it with a grid of equidistant points whose values in grid points can then be displayed as pixels on computer screens. The two main parameters of FOV are therefore its size and the pixel width. The size of FOV is proportional to the bandwidth of RF excitation and inverse proportional to the strength of the frequency encoding gradient. Thus, it makes increasing the RF bandwidth is beneficial

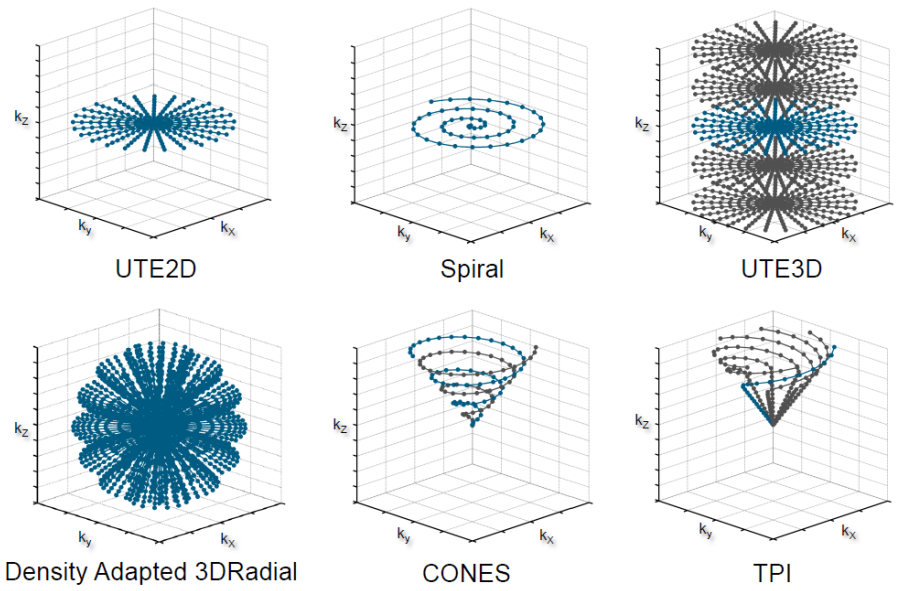

Figure 2.12: A few example for 3D trajectories. Source: [44]

for the image quality (although is comes with increased production costs as well), and it limits the strength of the frequency encoding gradient, even though stronger gradient would be advantageous to better separate tissues by their precessional frequencies. In contrast, the pixel size is determined by the size of range over which k-spaces samples are collected. In case of equidistant sampling in k-space over a square shaped FOV, the connection between these parameters can be expressed by simple equations:

$$\Delta k = 1/FOV \text{ and } \Delta w = 1/k_{FOV},$$

where Δk is the distance between k-space points, FOV refers to the size of the FOV, Δw is the pixel width, and k_{FOV} is defined as the range between the highest positive and largest negative spacial frequencies in k-space (i.e., $k_{FOV} = k^+_{max} - k^-_{max} = 2 \cdot k_{max}$, if $k^+_{max} = |k^-_{max}|$). For a visual example, please refer to fig. 2.13.

Using these concepts, one can formulate many methods to accelerate acquisition. For instance, it is possible to evaluate only half of the k-space exploiting the underlying symmetry in the Fourier transform. This method is successful in maintaining a high resolution, albeit it come with the cost of reduced signal-to-noise ratio (SNR) because of the lack of the noise-cancelling effect of two-sided measurements. Another popular method is to avoid acquiring the periphery of k-space by limiting the range of the phase-encoding frequencies (which, in fact, are connected to T_1 relaxation that contributes the most to the acqisition time). The advantage of this method is that it produces a decent speed-up keeping the SNR relatively high, but undersampling the high frequencies have a blurring effect, and therefore the resolution goes down. Finally, it is

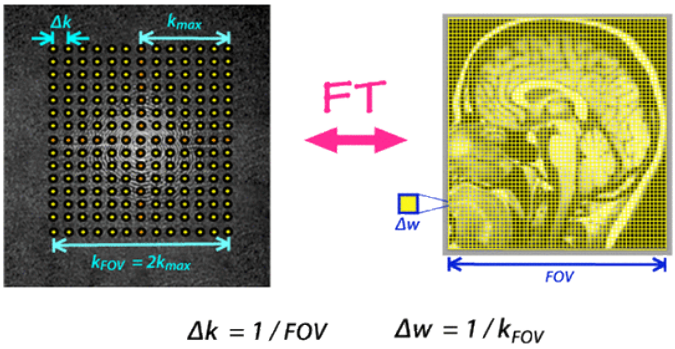

Figure 2.13: Connection between FOV and k-space in case of Cartesian sampling: The size of FOV is inversely proportional to the distance of k-space points, and the pixel size is determined by the size of range over which k-spaces samples are collected. Source: [45].

also possible to reduce the size of the FOV leading to a rectangular shaped image (also by reducing the phase-encoding step). The speed-up here, however, also comes with a cost: the amount of noise unfortunately remains the same as in the case of the wider FOV, but now it is distributed over a smaller area, and consequently the SNR also goes down.

A similar, but more effective method is proposed to accelerate imaging using multiple independent RF coils are built around the measured object. That method is commonly referred to as parallel imaging (PI), and idea behind is that the placement and the sensitivity characteristics of the coils are known, therefore the amplitude of the measured signal can be used to assist the localization of the signal source. This additional information makes less phase-encoding steps sufficient for acquisition, and thus allows a potentially several-fold reduction in imaging time. For an illustrative example, see 2.14. While PI also have downsides, namely the increased production cost of PI-capable machines, the unavoidable reduction in SNR (because each coil has its own, independent noise that sums up at reconstruction), and the introduction of PI-specific artifacts that comes from the inaccurate estimation of the coils sensitivities over the FOV, and the uneven distribution of noise related to coil geometry, the effect of these drawbacks can be reduces by the increase of number of coils (albeit this further increases the production cost).

2.3.5 Bottom Line

Since the advent of MR imaging, an explosion of the amount of MRI concepts and technologies is witnessed by the scientific community during the last 3-4 decades. Although the quality of the images has undergone drastic improvement, placing MRI to the focus

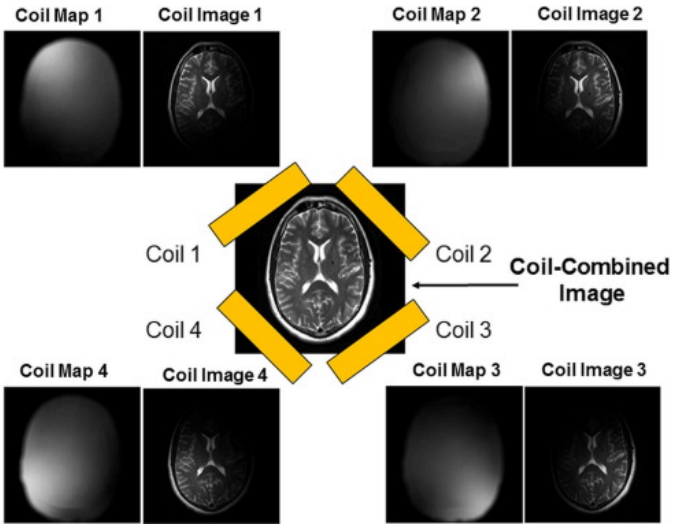

Figure 2.14: Illustrative example for parallel imaging technique. Multiple independent RF coils are built around the measured object, and as the detection sensitivities of the coils are known, the amplitude of the measured signal can be used to assist the localization of the signal source. Source: [46]

of today's diagnostic, the speed of evolution based on hardware-related innovations seems to slow down as the technology approaches its physical limits (e.g., the strength of the static magnetic field cannot be increased infinitely and the number of coils in PI is limited, for example, by the space available inside the machine). And while the acquisition time is significantly improved over time, but MRI is still considered to be a slow imaging technique, making dynamic imaging a challenging task. Hence, there is an increasing interest towards software solutions which can push further down the number of k-space points required to produce images with a quality sufficient for successful diagnosis.

Chapter 3

Mathematical Foundation

The real power of the compressed sensing is that it has a firm mathematical background that provides guarantees for the solution under certain assumptions. In this chapter, we essay to describe the very basics of compressed sensing, and then give a quick overview of a few selected numerical methods, which will be later used in this work as building blocks of more complex algorithms. These descriptions follow the book [47] and are based on the lectures of the course titled *Compressive Sampling* at Technical University of Munich by Alihan Kaplan.

3.1. Elementary Definitions

Although the reader might be to be familiar with the most of these definitions, for the sake of completeness and clarity of notation used in this work, we present here a list of definitions of elementary constructs, restricting ourselves to mere formulations with short remarks omitting further explanation.

Definition (sparsity)**:** *We call a vector s-sparse, if at most s of its entries are non-zero; i.e.* $\|\mathbf{x}\|_0 \leq s$.

Notation. By Σ_s^N we denote the set of all s-sparse vectors in $\mathbb{C}^N$; that is,

$$\Sigma_s^N = \left\{\mathbf{x} \in \mathbb{C}^N : \|\mathbf{x}\|_0 \leq s\right\}.$$

Definition (kernel/null space)**:** *The kernel/null space of a matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$ is defined as*

$$ker(\mathbf{A}) = \left\{\mathbf{x} \in \mathbb{C}^N : \mathbf{A}\mathbf{x} = \mathbf{0}\right\}.$$

Definition (convex functions)**:** *A function $f : \mathbb{C}^n \to \mathbb{R}$ is called convex, if for all $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$*

and $t \in [0,1]$

$$f(t\mathbf{x} + (1-t)\mathbf{y}) \leq tf(\mathbf{x}) + (1-t)f(\mathbf{y})$$

holds. Similarly, a function $f : \mathbb{C}^n \to \mathbb{R}$ *is called* ***strictly*** *convex, if for all* $\mathbf{x} \neq \mathbf{y} \in \mathbb{C}^n$ *and* $t \in [0,1]$

$$f(t\mathbf{x} + (1-t)\mathbf{y}) < tf(\mathbf{x}) + (1-t)f(\mathbf{y})$$

holds.

Remark. A useful property of convex function is that all local minimizers are also global minimizers. Moreover, the minimizer of a *strictly* convex function is unique, as well.

Definition (Lipschitz-continuity)**:** *A function* $f : \mathbb{C}^n \to \mathbb{R}$ *is said to be Lipschitz-continuous, if there exists a constant* $L \in \mathbb{R}$ *such that for all* $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L \|\mathbf{x} - \mathbf{y}\|_2$$

holds where L *is referred to as Lipschitz-constant.*

Remark. The set of Lipschitz-continuous functions is a superset of continuously differentiable functions.

Definition (gradient and Hessian)**:** *The gradient* ∇f *and the Hessian* $\mathbf{H}_f$ *of a function* $f : \mathbb{C}^n \to \mathbb{R}$ *at* $\mathbf{x} \in \mathbb{C}^n$ *is defined as follows:*

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(\mathbf{x}) \\ \frac{\partial}{\partial x_2} f(\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial x_n} f(\mathbf{x}) \end{bmatrix}, \mathbf{H}_f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} f(\mathbf{x}) & \frac{\partial^2}{\partial x_1 x_2} f(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_1 x_n} f(\mathbf{x}) \\ \frac{\partial^2}{\partial x_2 x_1} f(\mathbf{x}) & \frac{\partial^2}{\partial x_2^2} f(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_2 x_n} f(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial x_n x_1} f(\mathbf{x}) & \frac{\partial^2}{\partial x_n x_2} f(\mathbf{x}) & \dots & \frac{\partial^2}{\partial x_n^n} f(\mathbf{x}) \end{bmatrix}.$$

Remark. The geometric interpretation of gradient is that it is a vector in the tangent plane at a certain point on the surface defined by the function f, and this vector steepest direction where the value of f increases the quickest. The Hessian, on the other hand, carries information about the curvature of the surface.

Definition (order and rate of convergence)**:** *An iterative method producing the sequence* $\{\mathbf{x}_n\}$ *is said to have a convergence of order* p *to some* $\mathbf{x}^*$ *with respect to function* $f : \mathbb{C}^n \to \mathbb{R}$*, if there exists a number* $\mu \in \mathbb{R}_+ \cup \{0\}$*, called rate of convergence, such that*

$$\lim_{k \to \infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|^p} < \mu.$$

Remark. If $p = 1$, then the convergence is also referred to as a linear convergence. The special cases of linear convergence are *superlinear* convergence when $\mu = 0$ and *sublinear* convergence when $\mu > 0$. If $p = 2$, then the convergence is called *quadratic*.

Notation. Besides the definition above, there are two other commonly used formulation to characterize the convergence speed using the big O notation. One way is expressing the time complexity by number of steps needed to approximate the solution under the maximal tolerated error ϵ. For example, $\mathcal{O}(\epsilon^{-}2)$ is a colloquial notation to express that the function will produce the approximation $\mathbf{x}_k$ such that

$$|f(\mathbf{x}_k) - f(\mathbf{x}^*)| < \epsilon$$

after at most $k = \frac{1}{\epsilon^2}$ steps. The other way around is expressing the error in cost function by the number the already performed steps. Namely, $\mathcal{O}(k^{\frac{1}{p}})$ convergence states that

$$|f(\mathbf{x}_k) - f(\mathbf{x}_*)| < \frac{C \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2}{k^{\frac{1}{p}}}$$

for some $C \in \mathbb{R}$.

Remark. Note that using this notation, $\mathcal{O}(\epsilon^{-p})$ corresponds to $\mathcal{O}(k^{-\frac{1}{p}})$ (where p has the same meaning as in the definition above), and that $\mathcal{O}(\epsilon^{\alpha})$ means faster convergence than $\mathcal{O}(\epsilon^{\beta})$, if $\alpha < \beta$.

Definition (norm)**:** *A non-negative function $\|\cdot\| : X \to [0, \infty)$ is called a norm, if*

a) $\|\mathbf{x}\| = 0$ *if and only if* $\mathbf{x} = \mathbf{0}$,

b) $\|\lambda\mathbf{x}\| = \|\lambda\| \, \|\mathbf{x}\|$ *for all scalars* λ *and all vectors* $\mathbf{x} \in X$*, and*

c) $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ *for all vectors* $\mathbf{x}, \mathbf{y} \in X$.

Remark. X denotes a vector space on which the norm is defined. In MRI setting, however, $\mathbb{C}^N$ is the default vector space for computations, and therefore, we also define the following constructs in this space.

Definition (ℓ_p-norms for vectors)**:** *The ℓ_p-norm on $\mathbb{C}^N$ is defined for $1 \leq p < \infty$ as*

$$\|\mathbf{x}\|_p = \left(\sum_{j=1}^{n} |x_j|^p \right)^{\frac{1}{p}}, \tag{3.1}$$

and for $p = \infty$ as

$$\|\mathbf{x}\|_\infty = \max_{j \in [n]} |x_j|.$$

For $0 < p < 1$, (3.1) defines a quasinorm, which means that from the definition of the norm a)

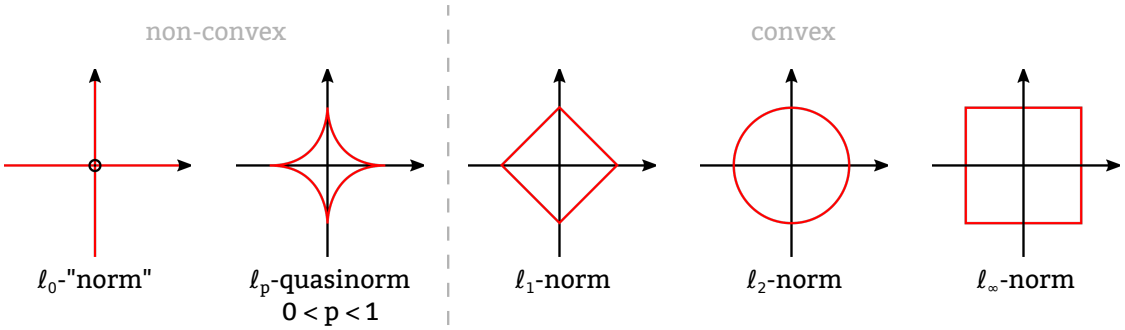

Figure 3.1: ℓ_p **balls in 2D.**

and b) holds, but c) is replaced by the weaker quasitriangle inequality

$$\|\mathbf{x}+\mathbf{y}\|_p \leq C\left(\|\mathbf{x}\|_p + \|\mathbf{y}\|_p\right)$$

with $C = 2^{\frac{1}{2}-1}$.

Remark. Figure 3.1 is showing 2-dimensional balls defined by ℓ_p-norm with various p values.

Notation. $[n]$ denotes the set of integers form 0 to $n-1$.

Remark. For $1 \leq p < \infty$, the ℓ_p-norms are strictly convex, and ℓ_p-quasinorms for $0 < p < 1$ are always non-convex (see fig. 3.1).

Remark. Using the schema above, it is impossible to have a proper norm for $p = 0$; nonetheless, is very common to define ℓ_0-norm as the number of non-zero coordinates:

$$\|\mathbf{x}\|_0 = |\{x_i \neq 0 : i \in [n]\}| \text{ where } \mathbf{x} \in \mathbb{C}^N.$$

Following this convention, $\|\cdot\|_0$ always refers to that formulation in this work.

Definition (matrix rank)**:** *The column rank of a matrix* $\mathbf{A} \in \mathbb{C}^{m\times N}$ *is defined as the maximal number of independent columns of* A*. Similarly, the row rank of a matrix* $\mathbf{A} \in \mathbb{C}^{m\times N}$ *is the maximal number of independent rows. As the column rank and the row rank are always equal, this number is simply called the rank of* A*.*

Theorem (SVD)**:** *For* $\mathbf{A} \in \mathbb{C}^{m\times N}$*, there exist unitary matrices* $\mathbf{U} \in \mathbb{C}^{m\times m}$, $\mathbf{V} \in \mathbb{C}^{N\times N}$*, and uniquely defined non-negative numbers* $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{min\{m,N\}} \geq 0$ *called singular values of* $\mathbf{A}$*, such that*

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \text{ where } \mathbf{\Sigma} = diag[\sigma_1, \sigma_2, \ldots, \sigma_{min\{m,N\}}] \in \mathbb{R}^{m\times N}.$$

The process of obtaining these matrices is called Singular Value Decomposition (SVD).

Remark. SVD is a particularly useful in analysis of matrix rank as the rank of a matrix always coincides with the number of its non-zero sigular values.

Definition: *(Schatten-p norm) The Schatten-p norm of a matrix* $\mathbf{A} \in \mathbb{C}^{m \times N}$ *is defined as the* ℓ_p*-norm applied to its singular values; that is,*

$$\|\mathbf{A}\|_p = \left(\sum_{i=1}^{\min\{m,N\}} \sigma_i^p(\mathbf{A}) \right)^{\frac{1}{p}}.$$

Three important special cases are $p = 1, 2, \infty$*, also called nuclear norm, Frobenius norm and spectral norm, respectively. The nuclear norm also has a distinct notation* $\|\cdot\|_*$*, and the Frobenius norm is usually denoted by* $\|\cdot\|_F$*.*

Remark. Similarly to ℓ_p-norms for $p \in (0, 1)$, the Schatten-p norms with $0 < p < 1$ are only quasinorms and are always non-convex. Schatten-0 norm are, however, not defined as Schatten-p norms for $p \to 0$ converge to an already defined quantifier, the rank (of which, of course, one can think as the ℓ_0-"norm" of singular values).

Definition (discrete Fourier transform (DFT))**:** *The discrete Fourier transform* $\hat{\mathbf{x}} \in \mathbb{C}^m$ *of a vector* $\mathbf{x} \in \mathbb{C}^m$ *is given by*

$$\hat{x}_k = \sum_{l=0}^{m-1} x_l \cdot e^{-\frac{2\pi i}{m} kl} : 0 \leq k \leq m-1,$$

or in matrix notation

$$\hat{\mathbf{x}} = \mathbf{F}\mathbf{x} \text{ with } F_{k,l} = e^{-\frac{2\pi i}{m} kl}.$$

3.2. Basics of Compressed Sensing

As it was shortly mentioned in chapter 1, a mathematical framework called compressed sensing (CS) revolutionized MR image acquisition process allowing reconstruction from much fewer k-space values *under certain conditions* as it would be necessary according to the Nyquist criterion.

To realize this promise, first and foremost, the signal to be recovered must be sparse in some transform domain. Fortunately, this criterion is usually already fulfilled, and relatively easy to find the sparse representation. However, there also other conditions need to be satisfied for successful recovery, and they are less intuitive. Hence, in this section we attempt to give a quick overview of the most important definitions and theorems needed for basic understanding.

3.2.1 Formulation of the Problem

In engineering settings, especially in signal processing context, engineers and scientist usually try first to model physical systems by a linear model because that way they

describe the problem by a set of linear expressions, and then they can express it as a matrix-vector multiplication $\mathbf{Ax} = \mathbf{y}$, where vector $\mathbf{x}$ is the input of the system, vector $\mathbf{y}$ is the output (measured data), and $\mathbf{A}$ characterizes the measurement (thus it is often referred to as *measurement matrix*). A very common task then is to recover the input consisting of N variables from the measurement data, and generally the number of measurements m must obey $m > N$, otherwise the linear system is underdetermined, and hence there exist infinitely many solutions. In case of MRI setting, this statement corresponds to already mentioned Nyquist criterion that requires the sampling frequency in k-space to be twice as the highest frequency in the image space (vid. $k_{FOV} = 2 \cdot k_{max}$ in section 2.3.4).

And that is the point when compressed sensing comes into play claiming that given a *proper* measurement matrix $\mathbf{A}$, the problem

$$\min_{\mathbf{z}\in\mathbb{C}^N} \|\mathbf{z}\|_0 \text{ subject to } \mathbf{Az} = \mathbf{y} = \mathbf{Ax} \tag{P_0}$$

have a unique s-sparse solution with $s \ll N$. This optimization problem is often referred to as (P_0). The number of necessary measurement, however, still a difficult question. There are theoretical results stating that $m = 2s$ is the lower bound for a perfect recovery [48] and that a stable recovery (later explained) occurs with high probability with $m \geq Cslog(N/m)$ for random measurement matrices [47], but in practice, reconstruction algorithms struggle to reach these theoretical limits (albeit, the achieved m is still drastically improved compared to the Nyquist sampled case).

The main reason why the optimal bounds are usually not reached is that the measurement matrices of real life systems have often have less favorable properties as random matrices and, more importantly, (P_0) is a NP-hard problem [49]; thus, only relaxations can be solved. The most commonly used relaxations are the so called ℓ_1 minimization or Basis Pursuit (BP) [50] defined as

$$\min_{\mathbf{z}\in\mathbb{C}^N} \|\mathbf{z}\|_1 \text{ subject to } \mathbf{Az} = \mathbf{y}, \tag{P_1}$$

the Basis Pursuit DeNoising (BPDN) [51] formulated as

$$\min_{\mathbf{z}\in\mathbb{C}^N} \|\mathbf{z}\|_1 \text{ subject to } \|\mathbf{Az} - \mathbf{y}\|_2 \leq \eta, \tag{P_1, η}$$

and the LASSO (Least Absolute Shrinkage and Selection Operator) [52] problem expressed by

$$\min_{\mathbf{z}\in\mathbb{C}^N} \|\mathbf{Az} - \mathbf{y}\|_2 \text{ subject to } \|\mathbf{z}\|_1 \leq s.$$

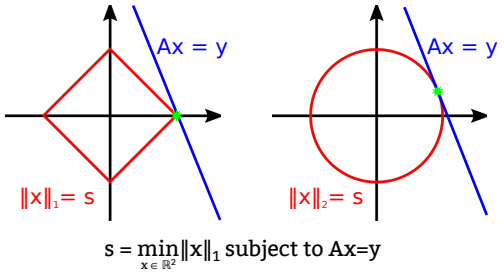

Figure 3.2: Solution of ℓ_1 and ℓ_2 minimization problems. Matrix $\mathbf{A} \in \mathbb{R}^{1\times 2}$ is an underdetermined linear system, and the task is to find $\mathbf{x} \in \mathbb{R}^2$ with minimal ℓ_1-norm such that it satisfies the equation $\mathbf{Ax} = \mathbf{y}$ for fixed $\mathbf{y} \in \mathbb{R}$. The number of solutions are infinite (all points along the blue line), and the unique solution of minimization problem is marked with a green star. Note that ℓ_1 minimization gives sparse solution, while ℓ_2 minimization is not sparsity seeking.

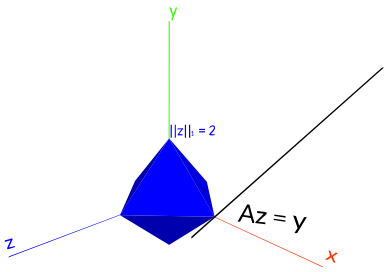

Figure 3.3: ℓ_1 minimization in 3D. This figure demonstrates the sparsity seeking nature of ℓ_1 minimization in a 3-dimensional space. The image is taken from the interactive demo available at `https://hakkelt.github.io/CS-demo/`

By the aid of a Lagrangian multiplier λ, the latter two can be transformed to the same unconstrained minimization problem

$$\min_{\mathbf{z}\in\mathbb{C}^N} \|\mathbf{Az} - \mathbf{y}\|_2 + \lambda \|\mathbf{z}\|_1 .$$

3.2.2 Conditions and Guarantees

While ℓ_1-relaxation (often referred to as (P_1) problem) might be appealing as it can be solved efficiently in polynomial time, it needs a more careful approach to guarantee that the minimum of the ℓ_1 problem is also a solution of (P_0). The ℓ_2-relaxation, for example, always has a unique solution, but this solution is not necessarily sparse (for a figurative illustration, see fig. 3.2 and fig. 3.3). In contrast, ℓ_1 problem has a solution which is both unique and s-sparse given that the matrix $\mathbf{A}$ fulfills the so called *null space property* of order s, introduced by Cohen, Dahmen and DeVore in [53].

Definition (NSP)**:** *A matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$ is said to satisfy the null space property (NSP) of order s, if for any set $S \subset [N]$ with $|S| = s$*

$$\|\mathbf{v}_S\|_1 < \|\mathbf{v}_{S^C}\|_1 : \forall \mathbf{v} \in ker(\mathbf{A}) \setminus \{\mathbf{0}\}.$$

Notation. For a vector $\mathbf{v} \in \mathbb{C}^N$ and a set $S \subset [N]$, we denote by $\mathbf{v}_S$ either the vector in $\mathbb{C}^{|S|}$ which is the restriction of $\mathbf{v}$ to the indices in S, or the vector in $\mathbb{C}^N$ which coincides with $\mathbf{v}$ on the indices in S and is zero elsewhere. Similarly, $\mathbf{v}_{S^C}$ means the same with the complement of S.

Theorem: *Given a matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$, every s-sparse vector $\mathbf{x} \in \Sigma_s^N \subset \mathbb{C}^N$ is the unique solution of (P_1) with $\mathbf{y} = \mathbf{A}\mathbf{x}$ if and only if $\mathbf{A}$ satisfies the NSP of order s.*

Remark. This theorem shows that for every $\mathbf{y} = \mathbf{A}\mathbf{x}$ with s-sparse $\mathbf{x}$, the ℓ_1-minimization (P_1) actually solves the ℓ_0-minimization (P_0) when the NSP of order s holds.

Although NSP is a formidable construct that allows relatively easy and straightforward proofs, in realistic settings, signals are rarely sparse, but rather *almost* s-sparse vectors, meaning that the most part of the energy of the signal is concentrated in s coefficients with the the largest values. To involve these cases in the compressed sensing framework, an extension of NSP is used, called *stable NSP*. Proving the stable NSP property, one can then approximate the almost s-sparse signal with the *best s-term approximation* having a tight bound on the error term.

Definition (ℓ_p-error of best s-term approximation)**:** *For $p > 0$, the ℓ_p-error of best s-term approximation to a vector $\mathbf{x} \in \mathbb{C}^N$ is defined by*

$$\sigma_s(\mathbf{x})_p = \inf\left\{\|\mathbf{x} - \mathbf{z}\|_p : \mathbf{z} \in \Sigma_s^N\right\}.$$

Remark. The infimum is always achieved by an $\mathbf{z} \in \Sigma_s^N$ whose non-zero entries equal the s largest absolute entries of $\mathbf{x}$.

Definition (stable NSP)**:** *A matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$ is said to satisfy the stable NSP with constant $0 < \rho < 1$ of order s, if for any set $S \subset [N]$ with $|S| = s$*

$$\|\mathbf{v}_S\|_1 \leq \rho \|\mathbf{v}_{S^C}\|_1 : \forall \mathbf{v} \in ker(\mathbf{A}).$$

Theorem: *The matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$ satisfies the stable NSP with constant $0 < \rho < 1$ of order s if and only if for any set $S \subset [N]$ with $|S| = s$*

$$\|\mathbf{z} - \mathbf{x}\|_1 \leq \frac{1+\rho}{1-\rho}\left(\|\mathbf{z}\|_1 - \|\mathbf{x}\|_1 + 2\|\mathbf{x}_{S^C}\|\right) \tag{3.2}$$

holds for any set $S \subset [N]$ with $|S| = s$ and for all vectors $\mathbf{x}, \mathbf{z} \in \mathbb{C}^N$ with $\mathbf{A}\mathbf{z} = \mathbf{A}\mathbf{x}$.

Remark. The estimation (3.2) can be upper-bounded by means of the ℓ_1-error of best s-term approximation $\sigma_s(\mathbf{x})_1$ as

$$\|\mathbf{z} - \mathbf{x}\|_1 \leq \frac{1+\rho}{1-\rho}\left(\|\mathbf{z}\|_1 - \|\mathbf{x}\|_1 + 2\|\mathbf{x}_{S^C}\|\right) \leq 2 \cdot \frac{1+\rho}{1-\rho}\sigma_s(\mathbf{x})_1.$$

The significance of this theorem is that it implies that having the stable NSP fulfilled, the unique s-sparse solution $\mathbf{z}$ of (P_1) with $\mathbf{y} = \mathbf{A}\mathbf{x}$ approximates the vector $\mathbf{x}$ with a

bounded ℓ_1-error. Therefore, by a good approximation of the sparsity s of the signal and an upper bound ϵ of $\sigma_s(\mathbf{x})_1$ (both can be inferred by the statistics of the signal), one can construct or select a measurement matrix $\mathbf{A}$ such that the solution of *any* algorithm solving (P_1) is an approximation of the input signal with the maximal error of

$$\frac{1+\rho}{1-\rho} \cdot 2\epsilon,$$

where ρ depends only on the measurement matrix and ϵ is a small number because only little energy is stored in the entries contributing the the ℓ_p-error of best s-term approximation. Practically that means that by the careful design of the measurement, arbitrary precision recovery is possible (having Nyquist sampling as a special case for guaranteed perfect recovery), and that in case of approximately s-sparse signals, the number of measurements can be drastically reduced without significant amount of error introduced.

Finally, this construct can be further extended to handle noisy measurement: proving the *robust NSP* for the measurement matrix, the same guaranties apply to the ℓ_1-error as in case of stable recovery, extended by an extra term characterizing the noise on the measurement. As a result, arbitrary precision is allowed for arbitrary optimization algorithm capable of solving (P_1, η) given a measurement matrix satisfying the *robust NSP*.

Definition (robust NSP)**:** *A matrix $\mathbf{A} \in \mathbb{C}^{m \times N}$ is said to satisfy the robust NSP with constants $0 < \rho < 1$ and $0 < \tau$ of order s, if for any set $S \subset [N]$ with $|S| = s$*

$$\|\mathbf{v}_S\|_1 \leq \rho \|\mathbf{v}_{S^C}\|_1 + \tau \|\mathbf{A}\mathbf{v}\|_2 : \forall \mathbf{v} \in \mathbb{C}^N.$$

Remark. Note that $\mathbf{v}$ is not required to be in $ker(\mathbf{A})$. In fact, if $\mathbf{v} \in ker(\mathbf{A})$, then $\mathbf{A}\mathbf{v} = \mathbf{0}$, and we obtain the definition of stable NSP. Therefore, the robust NSP implies stable NSP.

Theorem: *The matrix $\mathbf{A} \in \mathbb{C}^{m \times N}$ satisfies the robust NSP with constants $0 < \rho < 1$ and $0 < \tau$ of order s if and only if*

$$\|\mathbf{z} - \mathbf{x}\|_1 \leq \frac{1+\rho}{1-\rho} \left(\|\mathbf{z}\|_1 - \|\mathbf{x}\|_1 + 2\|\mathbf{x}_{S^C}\| \right) + \frac{2\tau}{1-\rho} \|\mathbf{A}(\mathbf{z} - \mathbf{x})\|_2 \tag{3.3}$$

holds for any set $S \subset [N]$ with $|S| = s$ and for all vectors $\mathbf{x}, \mathbf{z} \in \mathbb{C}^N$ with $\mathbf{A}\mathbf{z} = \mathbf{A}\mathbf{x}$.

Remark. Supposing that a matrix $\mathbf{A} \in \mathbb{C}^{m \times N}$ satisfies the robust NSP of order s, (3.3) implies that for any $\mathbf{x} \in \mathbb{C}^N$, a solution $\mathbf{z}$ of (P_1, η) with $\|\mathbf{A}\mathbf{x} - \mathbf{y}\| \leq \eta$ approximates $\mathbf{x}$

with ℓ_1-error [1]

$$\|\mathbf{z}-\mathbf{x}\|_1 \leq 2 \cdot \frac{1+\rho}{1-\rho}\sigma_s(\mathbf{x})_1 + \frac{4\tau}{1-\rho}\eta. \tag{3.4}$$

The main problem, nevertheless, with the NSP is that proving it is actually NP-hard in general. Hence, the *restricted isometry property (RIP)* is more commonly used in theoretical works because it is easier to handle, and at the same time it implies stable NSP.

Definition (RIP)**:** *A matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$ is said to satisfy the s-restricted isometry property with the smallest number $0 < \delta_s < 1$, called restricted isometry constant (RIC), if*

$$(1-\delta_s)\|\mathbf{x}\|_2^2 \leq \|\mathbf{A}\mathbf{x}\|_2^2 \leq (1+\delta_s)\|\mathbf{x}\|_2^2$$

holds for add $\mathbf{x} \in \Sigma_s^N$.

Theorem: *If a matrix $\mathbf{A} \in \mathbb{C}^{m\times N}$ has restricted isometry constant*

$$\delta_{2s} < \frac{1}{\sqrt{2}},$$

then it satisfies the robust NSP of order s with constants

$$\rho = \frac{\delta_{2s}}{\sqrt{1-\delta_{2s}^2}} \text{ and } \tau = \frac{2\sqrt{s}}{(1-\delta_{2s}\sqrt{1+\delta_{2s}}}.$$

3.2.3 The Connection Between CS and MRI

Although it is not always trivial to apply the compressed sensing scheme to different measurement methods, MR imaging is in a lucky position with the Fourier transform in the core of its image acquisition process as random partial discrete Fourier transform (i.e., selecting m rows randomly with uniform distribution from the N rows, or in other words, observing only m entries of the Fourier transform) is proved to satisfy the RIP [54]. That discovery has enormous significance as is offers a way to speed up the inherently slow MRI measurements by measuring only a few randomly selected points from the k-space, and then solve the ℓ_1-minimization problem via an arbitrarily selected optimization algorithm.

One can think of this recovery scheme as a denoising or inference cancelling process where the noise/inference to be removed is the sum of undersampling artifacts predicted by Nyquist-Shannon theorem. If we use an equispaced sampling pattern,

[1] Number 4 in the numerator of the error term at (3.4) is not a typo as one would expect, but a counter-intuitive result of some basic arithmetics leading to this expression.

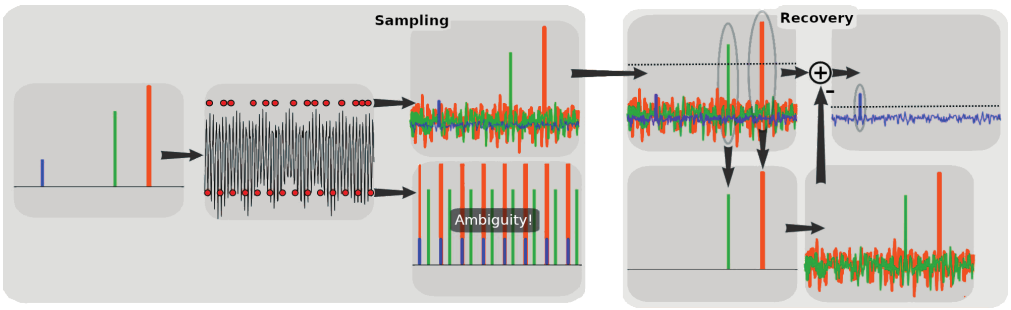

Figure 3.4: A heuristic process for reconstruction signal from randomly undersampled data. In the left block, the result of Fourier analysis is shown for both equidistant and random sampling, demonstrating the underdetermined nature of equidistant undersampling (bottom) and the noise-like effect of random sampling (top). In the right block, a simple method is applied that selects the peaks above a certain threshold as candidates, calculates the inference of these peaks, and substracts the calculated inference from the spectrum allowing detection of further peaks after lowering the threshold. Source: Adapted from [55].

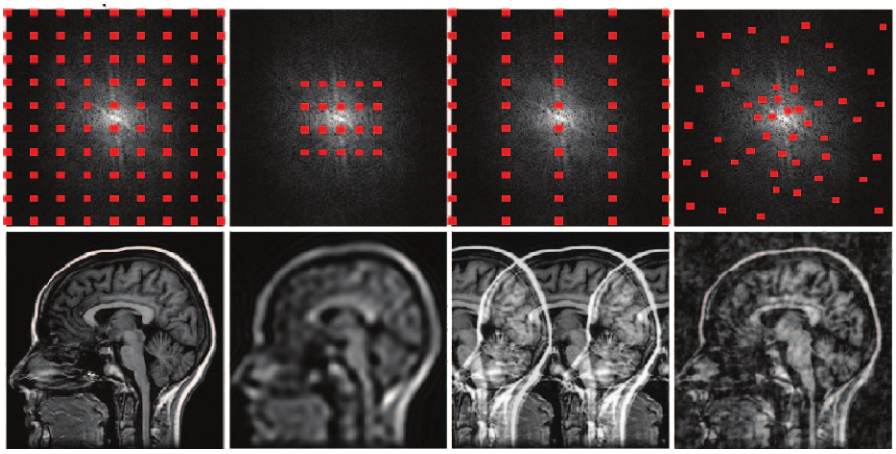

Figure 3.5: Connection between the sampling pattern and the introduced inference. Using inverse DFT replacing the missing values with zeros leads to different type of undersampling artifacts. Sampling only the center reduces the resolution, undersampled equispaced scheme adds shifted copies of the signal to the recovered image, and random sampling spreads the inference uniformly. Source: Adapted from [55].

then the resulted noise is basically shifted copies of the signal and hence recovery of the original signal is impossible as each replica is an equally likely candidate. Figure 3.4 illustrates this problem and demonstrates the advantage of random sampling. In contrast, random sampling "spreads" the noise uniformly over the image, so the inference acts mostly like white noise (see fig. 3.5). This approach is particularly appealing as countless denoising algorithms exists, and as many of them are based on minimization of the ℓ_1-norm of the transform of signal, these methods provide a convenient way to solve (P_1) problem. The transform used is always depends on the type of image in interest: MR angiograms (imaging technique visualizing blood vessels) are sparse even in the image domain, but edge detection algorithms like finite differences transform (that

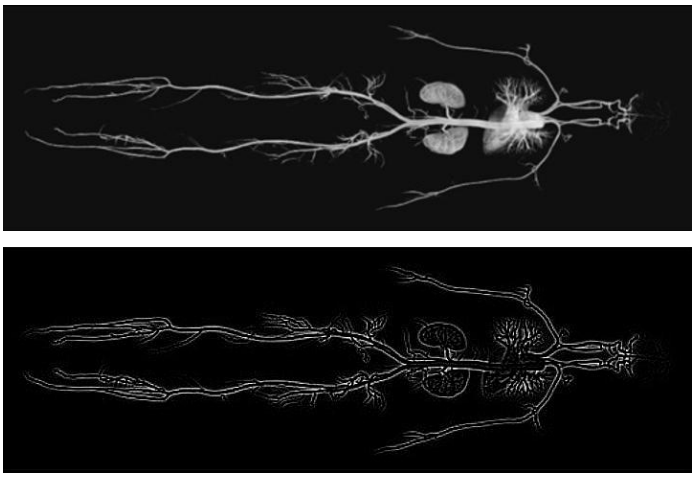

Figure 3.6: MR angiograms (MR images tuned to show blood vessels) are intrinsically sparse even in image domain (top image), but sparseness can be further improved by edge detection algorithms (botton image) because only the morphology of the blood vessels are diagnostically relevant. Source: Adapted from [56].

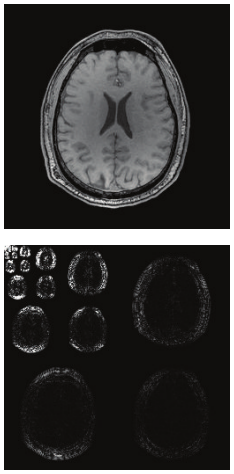

Figure 3.7: MRI images of the brain are not sparse in image domain, but they are in multiple wavelet transform domains. Source: [57].

corresponds to the well-known total variation (TV) penalty) can further enhance sparsity (fig. 3.6); brain images have a sparse representation in various wavelet transformations 3.7, and dynamic MR images tend to be sparse after temporal Fourier transform is applied.

A truly random sampling in the k-space, however, is generally impractical due to hardware and physiological constraints. For example, the sampling must follow smooth lines and curves and be robust to real-life situations such as motion artifacts. Also, a uniform random distribution of samples in the spatial-frequency domain does not take into account the energy distribution of MR images in k-space. Therefore it makes more sense to opt for a nonuniform variable density sampling matching energy distribution in k-space. Precisely, we should consider having more samples from the central part of the frequency domain and less high frequency components. Consequently, carefully designed pseudo-random sampling trajectories are utilized in practice. To quantify evaluate the "randomness" of the trajectories, authors of [55] and [58] suggested using point spread function (PSF) to measure the desirable incoherence of the aliasing interference, defined as

$$PSF = (\mathbf{e}_j^* \mathcal{F}_u^* \mathcal{F}_u \mathbf{e}_i)_{i,j}$$

where $\mathcal{F}_u$ is the undersampling Fourier transform operator, $\mathbf{e}_i$ and $\mathbf{e}_j$ are of the natural basis having 0 in each coordinate except the i-th and j-th position, resprectively, where

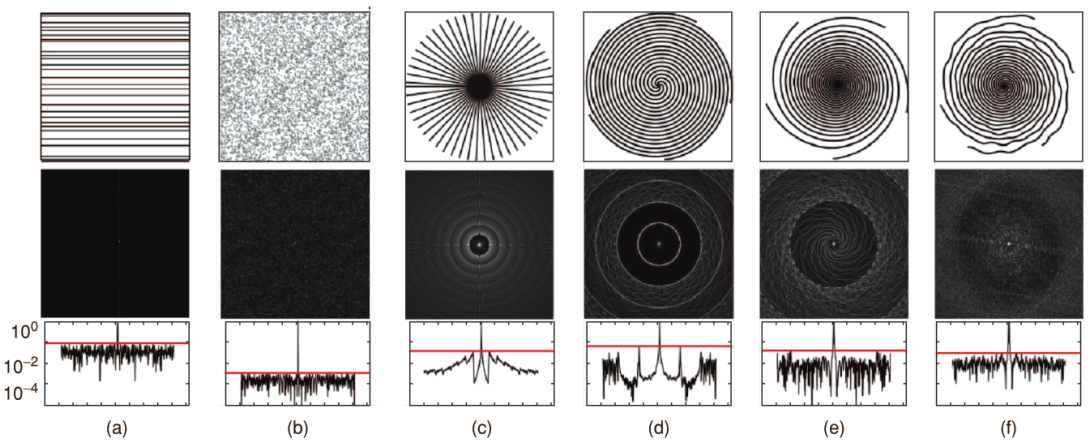

Figure 3.8: PSF of the central pixel for various sampling trajectories: (a) randomly chosen horizontal lines, (b) uniformly selected random points, (c) radial, (d) uniform spirals, (e) variable density spirals, and (f) variable density perturbed spirals. The red line marks the overall coherence level.
Source: Adapted from [55].

1 is located. Thus, $PSF_{i,j}$ measures the contribution of a unit-intensity pixel at the i-th position of the image to the j-th position in the k-space. In case of fully sampled measurement, the PSF is an identity matrix, and undersampling induces non-zero off-diagonal terms. Figuratively speaking, PSF measures the leakage of energy from a Kronecker delta input. Figure 3.8 shows PSF for a central pixel for a couple sampling trajectories. This quantitiy then can serve as a simple measure for the incoherence by calculating the sidelobe-to-peak ratio:

$$SPR = \max_{i \neq j} \left| \frac{PSF_{i,j}}{PSF_{i,i}} \right| .$$

As a result, the design of an incoherent sampling trajetory aims to minimize SPR.

3.3. Numerical Methods

While the recovery is proved to be *possible* via polynomial time algorithms minimizing the ℓ_1-relaxation of (P_0), finding the best algorithm for a given use-case is a quite challenging problem as the algorithm must satisfy many constraints like fast convergence speed and guarantee for a (global) convergence, inexpensive computation and limited memory requirement, numerical robustness and lack of problem-dependent parameters. The number of choices are numberless; nevertheless, none of them appears to outperform all other in every aspects because improving convergence speed while keeping global convergence is difficult, increased computational speed usually comes at the cost of a more memory hungry implementation, and it is hard to optimize algorithms with-

out tuning its parameters specifically to the problem to be solved.

The computation cost and the memory requirement, in particular, tend to be most restrictive constraint. Namely, the direct calculation of Hessian is often computationally prohibitive on one hand, on the other hand, the storage of the calculated Hessian can be problematic even for medium sized images. To see that, let us consider a 256×256 grayscale image. Storing all the 65536 pixels of the image as a half-precision floating point number requires $256 \times 256 \times 2\,\mathrm{B} = 128\,\mathrm{KiB}$. While the gradient of the image is of the same size, the Hessian has a quadratic memory requirement, $(256 \times 256)^2 \times 2\,\mathrm{B} = 8\,\mathrm{GiB}$ in our example. Although this requirement can be satisfied even on a stronger personal computer, a high-resolution dynamic MR image, for instance, usually results in magnitudes larger allocations, like $256\,\mathrm{GiB}$ for a 3D image of size $512 \times 512 \times 512$ with 512 time frames and an enormously large $33\,554\,432\,\mathrm{TiB}$ allocation for its Hessian.

Therefore, second order optimization algorithms (i.e., algorithms using second derivative) are not feasible in image reconstruction problems in general, even though these algorithms usually outperform first order methods in convergence speed. On the other end of the scale, zeroth order methods (that is, algorithms not using any gradient information), while being feasible and sometimes used for image reconstruction, are considered to be impractical due to their slow convergence. Therefore, first order methods appear to be the default option in nearly all image processing settings, and accordingly, we will discuss them exclusively in the following. Of course, even this restricted scope is far too large to cover, hence we restrict ourselves only to a few concepts that will serve as building blocks of more complex algorithms reviewed in the next chapters.

3.3.1 General Gradient Descent

Since Cauchy first proposed an iterative gradient direction method to solve astronomic calculations in 1847 [59], the family of gradient descent algorithms became one of the largest and most popular group of solvers due to their simplicity and relatively inexpensive computation. The core of all gradient methods is the so called gradient step defined as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \nabla f(\mathbf{x_k}),$$

where f is commonly called a loss or cost function, and α is generally referred to as step size and it is a negative number if we consider the minimization problem and positive if we are looking for the maximum.

The greatest advantage of gradient descent is that it is universally applicable to all problems where f is differentiable, but it is capable to find only a local optimum, and, in its original form, it is rarely optimal (albeit sometimes the only feasible option). Additionally, finding the optimal step size and a proper starting point is quite challenging, and convergence is not always guaranteed. If the step size is too small then the convergence speed becomes impractically slow; on the other hand, too large step size can make the algorithm fail to converge. The improper choice of the initial point can also drastically delay convergence, and also might cause converge to a unfavorable local optimum instead of the global optimum (or, at least, a better optimum nearby).

Selection of Step Size

While a good starting point is always depends on the function to be minimized (from this point, we consider only the minimization problem without loss of generality), there are various schemes for determining the step size in each step. Notably, the line search methods are commonly used to solve this problem because of its universally applicability. As the name suggests, these methods perform a search for local minimum along a descent direction (which, in case of gradient methods, is given by the gradient) and return exactly the step size needed to step into this minimum. The search can be exact (like the later described conjugate gradient' implicit line search) or inexact where only an approximation of the local minimum is calculated (like backtracking line search [60] (see also fig. 3.9) or using Wolfe conditions [61]).

However, the line search methods become impractical when evaluation of the cost function is computationally expensive. In these cases only fixed (but not necessarily constant) step is applicable. Restricting the cost function to the set of convex and smooth (differentiable with L-Lipschitz-gradient) functions, the analysis of global convergence of first order methods with respect to step size selection becomes possible, still covering a very large part of practical problems. In the following, f is always assumed to satisfy these conditions.

The simplest step size selection scheme is the constant step size, and if we choose the step size to be $\frac{1}{L}$ where L is the Lipschitz constant of ∇f, then monotonic descent is ensured and convergence to the global minimizer is guaranteed as f is assumed to be convex. The convergence rate of cost function is linear ($\mathcal{O}(1/k)$) as

$$f(\mathbf{x}_k) - f(\mathbf{x}_*) \leq \frac{L \left\| \mathbf{x}_0 - \mathbf{x}_* \right\|}{4k + 2}$$

is proved to be a tight inaccuracy bound for the generated sequence [62]. This rate,

however, is undesirably slow, and therefore the optimal step size is a highly researched topic. Although it was only recently proved that the convergence rate of all first order method with constant step size is still linear at best [63], there is already a long history of accelerated gradient methods like conjugate gradient, heavy ball iterations, and fast gradient method.

3.3.2 Conjugate Gradient Method

To overcome the slow convergence rate of the classic gradient descent (GD) method, one can use a modified version of it, the conjugate gradient method (CG) developed by Hestenes and Stiefel [64], if the problem is defined by a symmetric (or self-adjoint in complex case) and positive-definite matrix. In that case, the direction of movement must be conjugate to the previous directions. Two non-zero vectors $\mathbf{u}$ and $\mathbf{v}$ are conjugate with respect to some $\mathbf{A}$ matrix, if

$$\mathbf{u}^T\mathbf{A}\mathbf{v} = \mathbf{0}.$$

Let us consider the following linear system as the subject of optimization:

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

where $\mathbf{A}$is symmetric, positive-definite and real matrix, and $\mathbf{b}$ is also known.

As a direct method

Since $\mathbf{A}$ is symmetric and positive-definite, it defines an inner product:

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbf{A}} := \mathbf{u}^T\mathbf{A}\mathbf{v}.$$

Using that inner product, it is possible to find n pairwise conjugate vectors: $\mathcal{P} = \{\mathbf{p}_1, ...\mathbf{p}_n\}$. Then $\mathcal{P}$ forms a basis in $\mathbb{R}^n$, so the solution ($\mathbf{x}_*$) of optimization problem can be represented in terms of that basis:

$$\mathbf{x}_* = \sum_{i=1}^{n} \alpha_i \mathbf{p}_i. \tag{3.5}$$

Left multiplying both sides with $\mathbf{p}_k^T\mathbf{A}$ we get:

$$\mathbf{p}_k^T\mathbf{A}\mathbf{x}_* = \sum_{i=1}^{n} \alpha_i \mathbf{p}_k^T\mathbf{A}\mathbf{p}_i = \sum_{i=1}^{n} \alpha_i \langle \mathbf{p}_k, \mathbf{p}_i \rangle_{\mathbf{A}}.$$

As we know that $\mathbf{A}\mathbf{x}_* = \mathbf{b}$ and that $\langle \mathbf{p}_k, \mathbf{p}_i \rangle_{\mathbf{A}} = 0 : \forall i \neq k$ because $\mathbf{p}_i$ vectors are mutually conjugate (i.e. orthogonal with respect to the inner product defined by matrix $\mathbf{A}$):

$$\mathbf{p}_k^T\mathbf{b} = \alpha_k \langle \mathbf{p}_k^T, \mathbf{p}_k \rangle_{\mathbf{A}}.$$

That way we can calculate all α_i coefficients,:

$$\alpha_i = \frac{\mathbf{p}_i^T \mathbf{b}}{\langle \mathbf{p}_i, \mathbf{i}_k \rangle_{\mathbf{A}}},$$

and using these coefficients $\mathbf{x}_*$ can be calculated directly by equation 3.5.

As an iterative method

One weakness of direct method that for high dimensional vectors, we have to calculate high number of coefficients. On the other hand, it is not necessary to calculate all of them as a good approximation of $\mathbf{x}_*$ can be obtained using only a few well chosen $\mathbf{p_i}$ vectors. To achieve that we need to define a cost function:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} - \mathbf{x}^T \mathbf{b}.$$

The existence of a unique minimizer is evident as its second derivative is given by a symmetric positive-definite matrix:

$$\nabla^2 f(\mathbf{x}) = \mathbf{A}.$$

Also, we can calculate the first derivative easily:

$$\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}.$$

After choosing an arbitrary starting point $\mathbf{x}_0$, we can start the iteration by calculating the so called *residual*, which is apparently equal to the negative gradient:

$$\mathbf{r}_{k+1} = \mathbf{b} - \mathbf{A}\mathbf{x}.$$

Then we have to make sure to get a direction that is conjugate to all previous directions by applying an operation similar to the Gram-Schmidt orthonormalising:

$$\mathbf{p}_k = \mathbf{r}_k - \sum_{i<k} \frac{\langle \mathbf{p}_i, \mathbf{r}_k \rangle_{\mathbf{A}}}{\langle \mathbf{p}_i, \mathbf{p}_i \rangle_{\mathbf{A}}} \mathbf{p}_i.$$

Following this direction, the next optimal location is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k,$$

where α_k can be derived by substituting the previous formula for $\mathbf{x}_{k+1}$ to the the cost function and minimizing it with respect to α_k :

$$\nabla f(\mathbf{x}_{k+1}) = \nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \stackrel{!}{=} 0 \Rightarrow \ldots \Rightarrow \alpha_k = \frac{\mathbf{p}_k^T \mathbf{r}_k}{\langle \mathbf{p}_k, \mathbf{p}_k \rangle_{\mathbf{A}}}$$

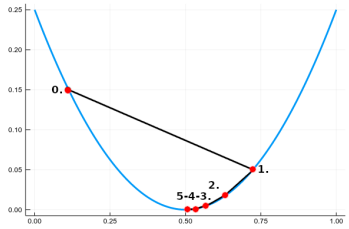

Figure 3.9: Example for backtracking algorithm. This 2D section of a high dimensional function is selected by the gradient direction, and we are looking for the local minimum along that slice. In the first step, a relatively large step size is chosen, which gets gradually decreased until local minimum is reached.

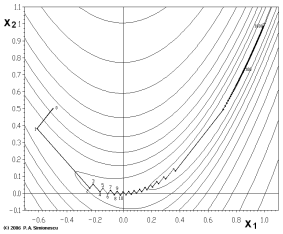

Figure 3.10: Example of the effect of inappropriate step size, "curved valley" and flat area: slow convergence of gradient descent method. Source: [65].

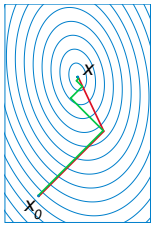

Figure 3.11: Visualization of the advantage of the CG method compared to GD: CG requires significantly fewer steps. The green line shows the path of the GD method, red shows the path of the CG method. Source: [66].

An example that gives an intuition why CG needs less steps than GD is shown by fig. 3.10 where the inappropriate step size and a "curved valley" produce zigzagging motion, and slow convergence as a result, which is slowed even further when the flat area in the bottom of that valley is reached. In contrast, CG avoids zigzagging because of the constraint of conjugate directions as it is depicted on fig. 3.11. The basic, but still the most popular, form of the algorithm is summarized in algorithm 1.

Algorithm 1: Conjugate Gradient (CG) method

input : vector $\mathbf{b}$,
symmetric, positive-definite matrix $\mathbf{A}$,
$\epsilon \in \mathbb{R}_+$ error tolerance parameter, and
approximate initial solution $\mathbf{x}_0$ (optional, set to $\mathbf{0}$ by default)

output: vector $\mathbf{x}$ such that $\mathbf{Ax} = \mathbf{b}$

Initialize $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{p}_0 = \mathbf{r}_0$, $k = 0$

while $\|\mathbf{r}_k\| > \epsilon$ **do**

$$\alpha_k = \frac{\mathbf{r}_k^* \mathbf{r}_k}{\mathbf{p}_k^* \mathbf{A} \mathbf{p}_k}$$
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$
$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha \mathbf{A} \mathbf{p}_k$$
$$\beta_k = \frac{\mathbf{r}_{k+1}^* \mathbf{r}_{k+1}}{\mathbf{r}_k^* \mathbf{r}_k}$$
$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$$
$$k = k + 1$$

return $\mathbf{x}_k$

It worth noting that while it is hard to find a better method for simple problems

than CG, in more complex cases preconditioning is necessary to ensure fast convergence. Furthermore, there exist multiple extensions of the base CG algorithm, like the biconjugate gradient (BiCG) method and its stabilized version (BiCGSTAB) [67] that removes the constraint of self-adjointness, and the non-linear versions (just to name a few: [68]–[70]) that pushes even further the boundary of set of problems solvable by CG.

3.3.3 Accelerated Gradient Methods[2]

Even though the conjugate gradient methods provides an effective way to accelerate the convergence, the momentum-based methods that mostly outperform them in cases of complex, non-quadratic cost functions has also a long history. First, the so called heavy ball momentum was proposed by Polyak in 1967 to accelerate convergence [72]. The resulting heavy ball iteration takes the following form:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\alpha}{L}\nabla f(\mathbf{x}_k) + \beta(\mathbf{x}_k - \mathbf{x}_{k-1})$$

where α and β are simple constants, and the term $\beta(\mathbf{x}_k - \mathbf{x}_{k-1})$ is referred to as the *momentum*. The names *heavy ball* and *momentum* comes from the physical analogy; namely, the path defined by the minimizer sequence $\{\mathbf{x}_k\}$ resembles the route of a ball rolling down from a hill. The optimal choice of α and β is unfortunately not trivial. In fact, universally optimal options not exists, but there are many variants of heavy ball method with different choice of α and β, being optimal always only in specific use cases. Following that, Nesterov published the algorithm in 1983 that later became known as fast gradient method (FGM) [73]. In contrast to the original gradient descent scheme, it maintains two converging sequences $\{\mathbf{x}_k\}$ and $\{\mathbf{z}_k\}$, of which $\{\mathbf{x}_k\}$ is the sequence that converges to the solution. The algorithm consists of three simple steps as shown in algorithm 2.

Despite the seemingly different formulations, both methods use some kind of momentum rule. The main difference is that heavy ball simply performs a weighted addition of the momentum from the previous iterations and the gradient of the current position, Nesterov's method combines these term in a couterintuitive, yet optimal manner as depicted on fig. 3.12. While FGM is preferred in convex optimization problems since it is slightly faster than heavy ball method, the latter is more commonly used in the machine learning community due to its robustness in non-convex settings.

Up to recent time, FGM was considered to be the fastest possible first order method for convex optimization because of its quadratic convergence proved by Nesterov. He

[2]This section follows the presentation [71] given by Jeffrey Fessler in December, 2017.

Algorithm 2: Fast Gradient Method (FGM)

input : convex and smooth function f,

$\epsilon \in \mathbb{R}_+$ error tolerance parameter, and approximate initial solution $\mathbf{x}_0$ (optional, set to **0** by default)

output: vector $\mathbf{x} = \arg\min f(\mathbf{x})$

Initialize $t_0 = 1, k = 0, \mathbf{z}_0 = \mathbf{x}_0$

repeat

$\mathbf{z}_{k+1} = \mathbf{x}_k - \frac{1}{L}\nabla f(\mathbf{x}_k)$

$t_{k+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right)$

$\mathbf{x}_{k+1} = \mathbf{z}_{k+1} + \frac{t_k - 1}{t_{k+1}}(\mathbf{z}_{k+1} - \mathbf{z}_k)$

$k = k + 1$

until $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| < \epsilon$;

return $\mathbf{x}$

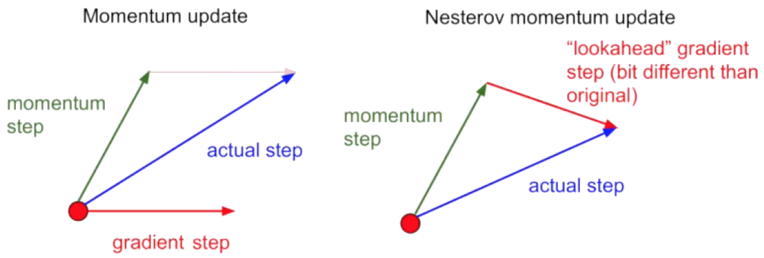

Figure 3.12: Difference between heavy ball momentum and Nesterov's momentum: While the heavy ball simply performs a weighted addition of the momentum from the previous iterations and the gradient of the current position, Nesterov's method combines these term in a couterintuitive, yet optimal manner. Source: Image is adapted from [74] and the idea of intuitive description of Nesterov's momentum rule as "lookahead" gradient step originates from [75].

also proved that the *order* of convergence (i.e., quadratic convergence) of his algorithm is optimal indeed for first order methods. Nevertheless, the *rate* of convergence of FGM is slightly sub-optimal as it was suggested by the convergence analysis performed by Drori and Teboulle in 2014 [62]. In their research, they considered a "meta optimization" problem, notably the minimization of worst-case convergence rate over all first order methods and all possible cost functions. As this problem is too difficult to be solved, they relaxed the problem to get a similar "meta-minimization" which can be solved by a semi-definite program. They found numerically that the relaxed bound of worst-case convergence rate for FGM is slightly below

$$\frac{2L\|\mathbf{x}_0 - \mathbf{x}_*\|_2^2}{(N+1)^2},$$

where N is the number of steps performed. Instead, they proposed a sequence of fixed step sizes calculated numerically that has approx. two times lower worst-case upper bound on convergence. Their approach, however, has a few drawbacks, such as the number of steps must be chosen in advance to ensure the improved convergence bound, and the numerical method producing these optimal step sizes is expensive both computationally and memory-wise. The next step that made this algorithm applicable in practical problems was done by Kim and Fessler in 2016, when they presented the analytical solution for the optimized step size coefficients [76] in form described by algorithm 3.

Algorithm 3: Optimized Gradient Method 1 (OGM1)

input : convex and smooth function f,
$\epsilon \in \mathbb{R}_+$ error tolerance parameter, and
approximate initial solution $\mathbf{x}_0$ (optional, set to $\mathbf{0}$ by default)

output: vector $\mathbf{x} = \arg\min f(\mathbf{x})$

Initialize $\theta_0 = 1, k = 0, \mathbf{z}_0 = \mathbf{x}_0$

while $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| > \epsilon$ **do**

$$\mathbf{z}_{k+1} = \mathbf{x}_k - \frac{1}{L}\nabla f(\mathbf{x}_k)$$

$$\theta_k = \begin{cases} \frac{1}{2}\left(1 + \sqrt{1 + 4\theta_{k-1}^2}\right) : k = 1, 2, \dots N-1 \\ \frac{1}{2}\left(1 + \sqrt{1 + 8\theta_{k-1}^2}\right) : k = N \end{cases}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{1+t_k/t_{k+1}}{L}\nabla f(\mathbf{x}_k) + \frac{t_k - 1}{t_{k+1}}(\mathbf{z}_{k+1} - \mathbf{z}_k)$$

$$k = k + 1$$

return $\mathbf{x}$

This form reflects the simple modification of the existing Nesterov method that leads to the improved speed, namely the introduction of a new momentum term. The only problem is that the step size sequence still depends on the number of steps performed. Therefore, the same authors published a refined version of the algorithm [77] a year later that removes this limitation and also simplifies the implementation, as one can see looking at algorithm 4.

Of course, this formulation also enjoys the twice better worst-case convergence rate meaning that the new convergence bound on the helper sequence $\{\mathbf{z}_k\}$ for every iteration is

$$f(\mathbf{z}_k) - f(\mathbf{z}_*) \leq \frac{1L\,\|\mathbf{x}_0 - \mathbf{x}_*\|_2^2}{(k+1)^2}.$$

Moreover, Drori showed in [78] that all first order algorithm have a function f such that

$$\frac{L\,\|\mathbf{x}_0 - \mathbf{x}_*\|_2^2}{N^2} \leq f(\mathbf{z}_N) - f(\mathbf{z}_*)$$

Algorithm 4: Optimized Gradient Method 2 (OGM2)

input : convex and smooth function f,
$\epsilon \in \mathbb{R}_+$ error tolerance parameter, and
approximate initial solution $\mathbf{x}_0$ (optional, set to $\mathbf{0}$ by default)

output: vector $\mathbf{x} = \arg\min f(\mathbf{x})$

Initialize $t_0 = 1, k = 0, \mathbf{z}_0 = \mathbf{x}_0$

while $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| < \epsilon$ **do**

$\mathbf{z}_{k+1} = \mathbf{x}_k - \frac{1}{L}\nabla f(\mathbf{x}_k)$

$t_{k+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right)$

$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{1+t_k/t_{k+1}}{L}\nabla f(\mathbf{x}_k) + \frac{t_k - 1}{t_{k+1}}(\mathbf{z}_{k+1} - \mathbf{z}_k)$

$k = k + 1$

return $\mathbf{x}$

for sufficiently large-scale problems. Thus OGM has optimal worst-case complexity, not just among the fixed step first order methods, but also among all first order methods.

3.3.4 Stochastic Gradient Descent

Sometimes the size of the problem does not allow the exact computation of even the gradient. For instance, in case of supervised learning, the problem to be solved is finding optimal parameters for a certain transform that minimizes a cost function over a huge dataset of images. Calculating the gradient of such cost function would require to load the entire dataset into the memory, but this is usually impossible as the dataset often consist millions of images requiring hundreds of gigabytes memory to load. Another example is the previously mentioned high-resolution volumentric dynamic MR image that needs $256\,\mathrm{GiB}$ for a 3D image of size $512 \times 512 \times 512$ with 512 time frames assuming that each pixel is stored as a single precision complex number.

Nevertheless, the application of gradient methods to these problems are not hopeless, if the cost function can be decoupled into smaller functions. Fortunately it is very often possible. Returning to the previous examples: The cost function of the supervised learning problem is usually a sum of values of a simple function evaluated separately for each image; and the cost function of the MRI problem can often be approximated by a summation of cost values evaluated for each frame.

One possible approach is calculating the gradient for each term of the summation (this needs loading only a single image/frame into the memory), and then the previously prohibiting memory demand can be reduced to an accumulating summation

that needs memory allocation only for the accumulator and the current image/frame. This approach is often referred to as batch gradient descent. Another approach, called stochastic gradient descent (SGD), is applying the position update step (i.e., the step that calculates the next element of the $\{\mathbf{x}_k\}$ sequence) after the evaluation of gradient for each frame. Even though this approach uses an a less accurate approximation of the true gradient as the batch method, it appears to be surprisingly robust in many practical applications frequently giving even a better result than batch processing in highly non-convex cases. The third possibility is a combination of the first two: the dataset is divided into small partitions, and the algorithm iterates over these partitions performing the update step only after the sum of gradients within the current batch is calculated. These partitions are usually called mini-batches, and thus the name of this method is mini-batch gradient descent.

3.3.5 Proximal Gradient Methods

Besides the possibly slow convergence, the standard gradient descent method suffers from another important limitation in real-life applications; notably, the the cost function is frequently not differentiable. There are multiple ways to handle this problem, like finding a differentiable surrogate function which has minimizers near to the original problem's minimizers, or using the subgradient method [79]. While the former is particularly useful in the cases when such surrogate can be easily defined, finding a good surrogate is a very difficult problem in general. The latter, however, converges undesirably slowly in many applications. In practice, more problem-specific methods are preferred.

In image processing and machine learning applications, the so called proximal gradient methods are peculiarly popular as the underlying assumptions of these methods covers a large range of practical problems. Specifically, it only requires that the cost function f to be minimized needs to be decomposable to a convex and differentiable function g with Lipschitz-contiuous gradient and a general convex function h:

$$f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x}).$$

First, we can say that although ∇f does not exists, ∇g is well-defined everywhere. Second, we can define the proximal mapping

$$prox_{h,t}(\mathbf{x}) = \arg\min_z \frac{1}{2t} \|\mathbf{x} - \mathbf{z}\|_2^2 + h(\mathbf{z}).$$

This mapping basically restricts the minimization of h to the *proximity* of $\mathbf{x}$ by the addition of a properly scaled paraboloid around $\mathbf{x}$. Third, we can apply the forward-backward splitting scheme introduced in [80] and [81], that is to say, we minimize f by alternating the forward and backward steps where the forward step optimizes $g(x)$ via a gradient step ($\mathbf{w}_{k+1} = \mathbf{x}_k - t_k \nabla g(\mathbf{x}_k)$), and the backward step optimizes $h(\mathbf{x})$ by proximity operator ($\mathbf{x}_{k+1} = prox_{h,t_k}(\mathbf{w}_{k+1})$). That two steps can be merged into one formula:

$$\mathbf{x}_{k+1} = prox_{h,t_k}(\mathbf{x}_k - t_k \nabla g(\mathbf{x}_k)).$$

Of course, this replacement of the original optimization problem helps us only if the proximity operator can be solved easily, preferably by a closed formula. Fortunately, the ℓ_1-norm satisfies that criterion as

$$prox_{\|\cdot\|_1,t}(\mathbf{x}) = \Lambda_t(\mathbf{x}),$$

where $\mathcal{T} : \mathbb{R}^n \to \mathbb{R}^n$ is the so-called soft-thresholding operator defined by

$$\Lambda_t(\mathbf{x})_i = sgn(x_i)\max(|x_i| - t, 0) = \begin{cases} x_i - t & : x_i > t \\ 0 & : -t \leq x_i \leq t \\ x_i + t & : x_i < -t \end{cases}.$$

After this operator, the group of methods using this thresholding function are named as *iterative shrinkage-thresholding algorithms* or *iterative soft-thresholding alorithms* (ISTA).

In spite of the fact that it convergence only linearly, this group of methods was successfully applied to solve many non-differentiable problems, most notably the (P_1) or LASSO problem where $g(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ and $h(\mathbf{x}) = \lambda \|\mathbf{x}_1\|$ and the corresponding ISTA iteration is given by

$$\mathbf{x}_{k+1} = \Lambda_{\lambda t_k}(\mathbf{x}_k - t_k A^T(\mathbf{A}\mathbf{x}_k - \mathbf{b}))$$

as $\nabla \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 = 2\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b})$. Therefore, it is easy to see the enormous significance of the contribution of a publication by Beck and Teboulle [82] in which the authors showed that ISTA can be accelerated by momentum rule used in FGM leading to a slightly more complex update step defined by

$$\mathbf{x}_k = \Lambda_{\lambda t_k}(\mathbf{z}_k - 2t_k \mathbf{A}^T(\mathbf{A}\mathbf{z}_k - \mathbf{b}))$$
$$t_{k+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right)$$
$$\mathbf{z}_{k+1} = \mathbf{x}_k + \frac{t_k - 1}{t_{k+1}}(\mathbf{x}_k - \mathbf{x}_{k-1}).$$

The authors also showed that their accelerated algorithm (named Fast ISTA or FISTA) enjoys quadrati convergence. Obviously, the OGM scheme also can be extended to the proximal gradient setting, and a possible algorithm is proposed in [63] using the update step

$$\begin{aligned}
\mathbf{y}_k &= \mathbf{x}_{k-1} + tA^T(\mathbf{A}\mathbf{x}_{k-1} - \mathbf{b}) \\
\mathbf{z}_k &= \mathbf{y}_k + \frac{\theta_{k-1}-1}{\theta_k}(\mathbf{y}_k - \mathbf{y}_{k-1}) + \frac{\theta_{k-1}}{\theta_k}(\mathbf{y}_k - \mathbf{x}_{k-1}) + t\frac{\theta_{k-1}-1}{\gamma_{k-1}\theta_k}(\mathbf{z}_{k-1} - \mathbf{x}_{k-1}) \\
\mathbf{x}_k &= \Lambda_{\gamma_k}(\mathbf{z}_k - 2t_k\mathbf{A}^T(\mathbf{A}\mathbf{z}_k - \mathbf{b}))
\end{aligned}$$

with $t = \frac{1}{L}$ where L is the Lipschitz-constant of $\nabla g(\mathbf{x}) = A^T(\mathbf{A}\mathbf{x} - \mathbf{b})$,

$$\gamma_k = \frac{1}{L}\frac{2\theta_{k-1} + \theta_k - 1}{\theta_k}, \text{ and } \theta_k = \begin{cases} \frac{1}{2}\left(1 + \sqrt{1 + 4\theta_{k-1}^2}\right) : k = 1, 2, \dots N-1 \\ \frac{1}{2}\left(1 + \sqrt{1 + 8\theta_{k-1}^2}\right) : k = N \end{cases}.$$

3.4. Constrained problems

So far we discussed only problems where the cost function was expressed in a single closed formula; however, real applications very often are better characterized by an objective function that must obey some constraints on its variables as we have already seen by the formulation of the compressed sensing problem in section 3.2.1. Unfortunately, a direct solution for these problems are hardly ever feasible, and therefore the most common scenario is to transform somehow these constrained problems to unconstrained ones.

A simple and intuitive way to do this transformation is using penalty methods that replaces the constrained problem with a series of unconstrained problems consisting of an addition of the cost function and the weighted sum of functions describing the constraints. In the general form, we can say that we need to find $\min f(\mathbf{x})$ subject to $c_i(\mathbf{x}) \leq 0 : \forall i \in [N]$ where N is the number of constraints. (Note that this formulation also contains the equality constraints as $c(\mathbf{x}) = 0$ can be replaced by two inequality constraints; i.e., $c(\mathbf{x}) \leq 0$ and $-c(\mathbf{x}) \leq 0$.) Penalty methods then convert this constrained formula to

$$\min f(\mathbf{x}) + \mu_k \sum_{i \in [N]} p(c_i(\mathbf{x})),$$

where p is called the penalty function and μ_k are the penalty coefficients, and we solve this problem by an arbitrary solver for different μ_k values always using the solution of previous iteration as the starting point for the next iteration. Initially μ_k is chosen to be

a small number, and it is increased in each iteration until it reaches infinity (or rather a sufficiently large number in practice). The intuition behind the algorithm is that first the method finds a region where the value of f is small, and as the value of μ_k increases it gradually converges to a solution that satisfies the constraints (therefore the penalty function needs to map values which satisfy the constrains to a small number and give large values that violates them; e.g. $g(x) = max(0, x)^2$ and $g(x) = e^x - 1$ are both feasible options).

A similar group of methods is the class of augmented Lagrangian methods (AL) was introduced by Powell [83] and Hestenes [84] in 1969. That class considers the problems with equality constraints specifically, that is, $\min f(\mathbf{x})$ subject to $c_i(\mathbf{x}) = 0 : \forall i \in [N]$. Compared to penalty methods, the transformed unconstrained problem contains an extra term designed to mimic a Lagrange multiplier:

$$\min f(\mathbf{x}) + \frac{\mu_k}{2} \sum_{i \in [N]} c_i(\mathbf{x})^2 + \sum_{i \in [N]} \lambda_i c_i(\mathbf{x}).$$

Similarly to penalty methods, μ coefficients are increased after each iteration, but in this case there is no need to increase them to infinity to reach the solution of the constrained problem because of the carefully weighted additional term. The "careful weighting" means that the λ_i variables are adjusted after each iteration to give higher weight to constraints which are violated, using the update rule $\lambda_{i+1} = \lambda_i + \mu_i c_i(\mathbf{x}_i)$.

A commonly used variant of the general augmented Lagrangian scheme is the so called Alternating Direction Method of Multipliers (ADMM) [85]. This scheme works on problems in the form

$$\min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}) + g(\mathbf{z}) \text{ subject to } \mathbf{B}\mathbf{x} + \mathbf{D}\mathbf{z} = \mathbf{c}$$

with convex functions f abd g, matrices $\mathbf{B}$ and $\mathbf{D}$, and some vector $\mathbf{c}$. The augmented Lagrangian of this constrained problem is defined by

$$L_\mu(\mathbf{x}, \mathbf{z}, \boldsymbol{\lambda}) = f(\mathbf{x}) + g(\mathbf{z}) + \boldsymbol{\lambda}^T(\mathbf{B}\mathbf{x} + \mathbf{D}\mathbf{z} - \mathbf{c}) + \frac{\mu}{2} \left\| \mathbf{B}\mathbf{x} + \mathbf{D}\mathbf{z} - \mathbf{c} \right\|_2^2,$$

and ADMM approximates the solution of this unconstrained problem via the three steps given as

$$\mathbf{x}_{k+1} = \arg\min_x L_\mu(\mathbf{x}, \mathbf{z}_k, \boldsymbol{\lambda}_k)$$

$$\mathbf{z}_{k+1} = \arg\min_z L_\mu(\mathbf{x}k + 1, \mathbf{z}, \boldsymbol{\lambda}_k)$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \mu(\mathbf{B}\mathbf{x}_{k+1} + \mathbf{D}\mathbf{z}_{k+1} - \mathbf{c})$$

For example, the LASSO problem $\min_x \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + c\|\mathbf{x}\|_1$ takes the form

$$\min_{\mathbf{x},\mathbf{z}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + c\|\mathbf{z}\|_1 \text{ subject to } \mathbf{x} - \mathbf{z} = \mathbf{0}$$

in the ADMM framework, and its augmented Lagrangian is

$$L_\mu(\mathbf{x}, \mathbf{z}, \boldsymbol{\lambda}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + c\|\mathbf{z}\|_1 + \boldsymbol{\lambda}^T(\mathbf{x} + \mathbf{z}) + \frac{\mu}{2}\|\mathbf{x} + \mathbf{z}\|_2^2 .$$

Finally, the ADMM steps are the following (using the previously defined soft-thresholding operator Λ):

$$\begin{aligned}
\mathbf{x}_{k+1} &= (\mathbf{A}^T\mathbf{A} + \mu\mathbf{I})^{-1}(\mathbf{A}^T\mathbf{b} + \mu\mathbf{z}_k - \boldsymbol{\lambda}_k) \\
\mathbf{z}_{k+1} &= \Lambda_{c/\mu}(\mathbf{x}_{k+1} + \boldsymbol{\lambda}_k/\mu) \\
\boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \mu(\mathbf{x}_{k+1} + \mathbf{z}_{k+1})
\end{aligned}$$

. The advantage of this scheme is that it allows distributed and parallelized computing with slight modification on the core algorithm.

Chapter 4

Related Works

As we seen in the previous chapters, the quest for finding practically feasible reconstruction methods that need the least number of measurements (ideally close to the theoretical limits of the compressed sensing framework) is a challenging task; nonetheless, it is a hot topic having very important new results an novel approaches outperforming the older ones almost in every year. In this chapter we will discuss two recently published research projects both presenting a state-of-the-art solution, and we will review a third project that proposes an algorithm that has the potential to drastically outperform the methods currently considered to be the best on this field.

4.1. Low Rank and Sparse Decomposition

Initially the compressed sensing framework was implemented in classic sparsity seeking settings, and the main question was which transform domain has the sparsest representation of the signal of interest. Although the authors of the first compressed sensing publications were aware of the low rank structure of certain MRI techniques back in 2007, this branch of research gained larger attention just in the recent years. An approach that appeared to be particularly suitable to dynamic MRI setting is decomposition of the image into a low rank and a sparse component [86]–[89]. The motivation behind such a decomposition comes from the inherent temporal correlation of the background—that correlation is captured by the low rank part usually denoted by L—and from the sparse nature of dynamic information that lies on top of the background, encompassed by the sparse component with the notation of S.

A recent paper from Lin and Fessler [90], published in 2019, attempts to summarize the advances in this field, then it proposes two new algorithms to accelerate the

convergence of the most successful algorithms, and finally it presents the results of the numerical experiments they carried out to compare the convergence rate of different state-of-the-art algorithms.

Formulation of the Problem

In contrast to the basic sparsity formulation discussed earlier as (P_0) problem and its relaxation to the (P_1) problem, the problem formulation of $L+S$ decomposition scheme (also called robust principal component analysis or RPCA) have two constraints in the place of the original constraint promoting sparsity in some transform domain. The first of these new constraints is the sparsity seeking ℓ_0 norm that takes the sparse part as an argument, similarly to the original settings in (P_0). The other constraint, however, enforces the low-rankness of L. Putting it together, we get a LASSO-like formula

$$\min_{\mathbf{L},\mathbf{S}} \|\mathcal{A}(\mathbf{L}+\mathbf{S})-\mathbf{y}\|_F \;\text{ subject to }\; rank(\mathbf{L})<r \;\text{ and }\; \|\Phi\mathbf{S}\|_1<s,$$

where $\mathcal{A}:\mathbb{C}^{N\times n_t}\to\mathbb{C}^{m\times n_c\times n_t}$ is a linear operator (often represented[1] by a matrix in $\mathbb{C}^{m\cdot n_c\cdot n_t\times N\cdot n_t}$, especially in theoretical discussions) modelling the 2D parallel MRI acquisition scheme that assumes N pixels in each frame, n_c receiver coils and n_t time frames that evaluates exactly m k-space points. Moreover, $\Phi:\mathbb{C}^{N\times n_t}\to\mathbb{C}^{N\cdot n_t}$ is an appropriate sparsifying transform, $\mathbf{y}$ denotes the collection of measurement data, $\mathbf{L},\mathbf{S}\in\mathbb{C}^{N\times nt}$ hold the low rank and sparse components, and finally r and s are the target rank and sparsity. After solving this problem for $\mathbf{L}$ and $\mathbf{S}$, the reconstructed image can be obtained by the simple element-wise addition of components (of course, we need to reshape back the resulted matrix to be a 2D or 3D dynamic image instead $N\times n_t$ shape created by stacking the vectorized frames as the columns).

The most common scenario to solve the $L+S$ problem follows the same line of arguments that was presented earlier: Instead of solving the problem that contains an ℓ_0 and a $rank$ term, both being NP-hard to optimize even separately, one can relax both constraints to ℓ_1-norm and nuclear norm minimization, respectively, resulting the Lagrangian formula

$$\min_{\mathbf{L},\mathbf{S}} \|\mathcal{A}(\mathbf{L}+\mathbf{S})-\mathbf{y}\|_2+\lambda_L\|\mathbf{L}\|_*+\lambda_S\|\Phi\mathbf{S}\|_1.$$

In the aforementioned paper, the authors considered the unitary temporal Fourier transform $\mathbb{T}_t$ as the sparsifying operator Φ, which is a largely conventional for dynamic

[1]This matrix representation requires the vectorization of the input before the matrix-vector multiplication; however, this operation is hardly ever indicated explicitly, but rather assumed to be performed implicitly. In this thesis work, we also omit denoting such vectorizations.

MR images, and they divided the analyzed algorithms into two main groups: methods based on proximal gradient technique and algorithms using variable splitting then solving the transformed problem in the augmented Lagrangian framework, possibly using the ADMM scheme.

Proximal Gradient Methods

In the first group they studied the three shrinkage-thresholding algorithms: ISTA, FISTA and POGM. While ISTA has already been utilized along with the $L + S$ modelling scheme in [88], it was their contribution to show the applicability of accelerated schemes in this setting. First, they combined $\mathbf{L}$ and $\mathbf{S}$ into a single "stacked" variable $\mathbf{X} = \begin{bmatrix}\mathbf{L} & \mathbf{S}\end{bmatrix}^T$ to simplify the notation because that way the momentum step can be expressed by a single equation. Second, they recalled that the proximity operator has closed formula; namely, the composition soft-thresholding function with the sparsity transform and the singular-value thresholding, the latter defined as

$$SVT_{\lambda_L}(\mathbf{L}) = \mathbf{U}\Lambda_\lambda(\mathbf{\Sigma})\mathbf{V}*,$$

first proposed to enforce low-rankness in [91]. Note that $\mathbf{U\Sigma V}*$ in the formula above stands for the singular value decomposition of the input matrix $\mathbf{L}$. Consequently, the forward-backward splitting scheme is easily applied to both variable performing

$$\begin{aligned}\mathbf{L}_k &= SVT_{\lambda_L}(\mathbf{L}_{k-1} - t\nabla_L g(\mathbf{X}_{k-1})\\ \mathbf{S}_k &= \mathbf{T} * \left(\Lambda_{\lambda_S}\left(\mathbf{T}(\mathbf{S}_{k-1} - t\nabla_S g(\mathbf{X}_{k-1}))\right)\right),\end{aligned}$$

where $g(\mathbf{X}) = \frac{1}{2}\left\|[\mathcal{A}\ \mathcal{A}]\mathbf{X} - \mathbf{y}\right\|_2^2$ and therefore

$$\nabla_L\ g(\mathbf{X}) = \nabla_S\ g(\mathbf{X}) = \mathcal{A}^*([\mathcal{A}\ \mathcal{A}]\mathbf{X} - \mathbf{y}) = \mathcal{A}^*(\mathcal{A}(\mathbf{L} + \mathbf{S}) - \mathbf{y}).$$

Accordingly, the algorithm takes form presented in algorithm 5.

Next, they extended this base algorithm to the accelerated scheme by inserting a momentum step that resulted in algorithm 6 for FISTA and algorithm 7 for POGM. For the sake of clarity and completeness, we note that the authors used a slightly different formulation than the algorithms presented here, as they followed the original formulation of ISTA given in [88]. The main difference is that they calculated the gradient at the *end* of the loop body, not in the beginning, and they introduced an additional variable $\mathbf{M}$, named *consistency term*, that was defined as

$$\mathbf{M} = \mathbf{L} + \mathbf{S} - \mathbf{E}^*(\mathbf{E}(\mathbf{L_{k-1}} + \mathbf{S_{k-1}}) - \mathbf{d})$$

Algorithm 5: ISTA for L+S decomposition

input : $\mathbf{y}$: undersampled k-space data

$\mathcal{A}$: data acquisition operator

$\mathbf{T}$: temporal Fourier transform

λ_L: singular value threshold

λ_S: sparsity threshold

N: number of iterations

output: $\mathbf{L}, \mathbf{S} = \arg\min_{\mathbf{L},\mathbf{S}} \|\mathcal{A}(\mathbf{L}+\mathbf{S}) - \mathbf{y}\|_2 + \lambda_L \|\mathbf{L}\|_* + \lambda_S \|\mathbf{TS}\|_1$

initialization: $\mathbf{L}_0 = \mathcal{A}^*\mathbf{d}, \mathbf{S}_0 = \mathbf{0}, \theta_0 = 1, t = 0.99$

notation: $\mathbf{X}_i = \begin{bmatrix} \mathbf{L}_i \\ \mathbf{S}_i \end{bmatrix}$

for $k = 1 \dots N$ **do**

$\nabla g_k = \mathcal{A}^*(\mathcal{A}(\mathbf{L_{k-1}} + \mathbf{S_{k-1}}) - \mathbf{y})$

$\mathbf{L}_k = SVT_{\lambda_L}(\mathbf{L}_{k-1} - t\nabla g_k)$

$\mathbf{S}_k = \mathbf{T}^* \left(\Lambda_{\lambda_S} \left(\mathbf{T}(\mathbf{S}_{k-1} - t\nabla g_k)\right)\right)$

return $\mathbf{L}_N + \mathbf{S}_N$

Algorithm 6: FISTA for L+S decomposition

input : same as input of algorithm 5

output: $\mathbf{L}, \mathbf{S} = \arg\min_{\mathbf{L},\mathbf{S}} \|\mathcal{A}(\mathbf{L}+\mathbf{S}) - \mathbf{y}\|_2 + \lambda_L \|\mathbf{L}\|_* + \lambda_S \|\mathbf{TS}\|_1$

initialization: $\mathbf{L}_0 = \mathcal{A}^*\mathbf{d}, \mathbf{S}_0 = \mathbf{0}, \theta_0 = 1, t = 0.5$

notation: $\mathbf{X}_i = \begin{bmatrix} \mathbf{L}_i \\ \mathbf{S}_i \end{bmatrix}, \tilde{\mathbf{X}}_i = \begin{bmatrix} \tilde{\mathbf{L}}_i \\ \tilde{\mathbf{S}}_i \end{bmatrix}, \bar{\mathbf{X}}_i = \begin{bmatrix} \bar{\mathbf{L}}_i \\ \bar{\mathbf{S}}_i \end{bmatrix}$

for $k = 1 \dots N$ **do**

$\nabla g_k = \mathcal{A}^*(\mathcal{A}(\mathbf{L_{k-1}} + \mathbf{S_{k-1}}) - \mathbf{y})$

$\tilde{\mathbf{L}}_k = \mathbf{L}_{k-1} - t\nabla g_k$

$\tilde{\mathbf{S}}_k = \mathbf{S}_{k-1} - t\nabla g_k$

$\theta_k = \frac{1}{2}\left(1 + \sqrt{1 + 4\theta_{k-1}^2}\right)$

$\bar{\mathbf{X}}_k = \tilde{\mathbf{X}}_k + \frac{\theta_{k-1}-1}{\theta_k}(\tilde{X}_k - \tilde{\mathbf{X}}_{k-1}) + \frac{\theta_{k-1}}{\theta_k}(\tilde{\mathbf{X}}_k - \mathbf{X}_{k-1})$

$\mathbf{L}_k = SVT_{\lambda_L}(\bar{\mathbf{L}}_k)$

$\mathbf{S}_k = \mathbf{T}^* \left(\Lambda_{\lambda_S} \left(\mathbf{T}(\bar{\mathbf{L}}_k)\right)\right)$

return $\mathbf{L}_N + \mathbf{S}_N$

Algorithm 7: POGM for L+S decomposition

input : same as input of algorithm 5

output: $\mathbf{L}, \mathbf{S} = \arg\min_{\mathbf{L},\mathbf{S}} \|\mathcal{A}(\mathbf{L}+\mathbf{S}) - \mathbf{y}\|_2 + \lambda_L \|\mathbf{L}\|_* + \lambda_S \|\mathbf{TS}\|_1$

initialization: $\mathbf{L}_0 = \mathcal{A}^*\mathbf{d}, \mathbf{S}_0 = \mathbf{0}, \theta_0 = \zeta_0 = 1, t = 0.5$

notation: $\mathbf{X}_i = \begin{bmatrix}\mathbf{L}_i \\ \mathbf{S}_i\end{bmatrix}, \tilde{\mathbf{X}}_i = \begin{bmatrix}\tilde{\mathbf{L}}_i \\ \tilde{\mathbf{S}}_i\end{bmatrix}, \bar{\mathbf{X}}_i = \begin{bmatrix}\bar{\mathbf{L}}_i \\ \bar{\mathbf{S}}_i\end{bmatrix}$

for $k = 1 \dots N$ **do**

$\nabla g_k = \mathcal{A}^*(\mathcal{A}(\mathbf{L_{k-1}} + \mathbf{S_{k-1}}) - \mathbf{y})$

$\tilde{\mathbf{L}}_k = \mathbf{L}_{k-1} - t\nabla g_k$

$\tilde{\mathbf{S}}_k = \mathbf{S}_{k-1} - t\nabla g_k$

$\theta_k = \begin{cases} \frac{1}{2}\left(1 + \sqrt{1 + 4\theta_{k-1}^2}\right) : k < N \\ \frac{1}{2}\left(1 + \sqrt{1 + 8\theta_{k-1}^2}\right) : k = N \end{cases}$

$\bar{\mathbf{X}}_k = \tilde{\mathbf{X}}_k + \frac{\theta_{k-1}-1}{\theta_k}(\tilde{X}_k - \tilde{\mathbf{X}}_{k-1}) + \frac{\theta_{k-1}}{\theta_k}(\tilde{\mathbf{X}}_k - \mathbf{X}_{k-1}) + \frac{\theta_{k-1}-1}{\zeta_{k-1}\theta_k}(\bar{\mathbf{X}}_{k-1} - \mathbf{X}_{k-1})$

$\zeta_k = t(1 + \frac{\theta_{k-1}-1}{\theta_k} + \frac{\theta_{k-1}}{\theta_k})$

$\mathbf{L}_k = SVT_{\lambda_L}(\bar{\mathbf{L}}_k)$

$\mathbf{S}_k = \mathbf{T}^*\left(\Lambda_{\lambda_S}\left(\mathbf{T}(\bar{\mathbf{L}}_k)\right)\right)$

return $\mathbf{L}_N + \mathbf{S}_N$

($\mathbf{E}$ and $\mathbf{d}$ corresponds to $\mathcal{A}$ and $\mathbf{y}$ in our notation). While this version avoids an unnecessary calculation of gradient in the very first iteration, it introduces a couple avoidable matrix additions in turn, and also it makes more difficult to understand how the algorithms are connected to the original, only sparsity seeking versions. We summarized all three algorithms in algorithm 8, following their notation, and it is easy to see that there is no significant difference between the two formulations, but our version is easier to understand.

Augmented Lagrangian Methods

The other approach they studied based on the augmented Lagrangian formulation combined with the variable splitting scheme. The conventional way, proposed in [87], to perform the variable splitting transform the original $L + S$ problem into

$$\min_{\mathbf{L},\mathbf{S}} \min_{\mathbf{P},\mathbf{Q}} \left\{ \frac{1}{2} \|\mathcal{A}(\mathbf{L}+\mathbf{S}) - \mathbf{y}\|_2^2 + \lambda_L \|\mathbf{P}\|_* + \lambda_S \|\mathbf{Q}\|_1 \right\} \text{ subject to } \begin{cases} \mathbf{P} = \mathbf{L} \\ \mathbf{Q} = \mathbf{TS} \end{cases}.$$

Algorithm 8: ISTA/FISTA/POGM for L+S, as formulated in [90]

input : same as input of algorithm 5

output: $\mathbf{L}, \mathbf{S} = \arg\min_{\mathbf{L},\mathbf{S}} \|\mathcal{A}(\mathbf{L}+\mathbf{S}) - \mathbf{y}\|_2 + \lambda_L \|\mathbf{L}\|_* + \lambda_S \|\mathbf{TS}\|_1$

initialization:

$$\mathbf{M}_0 = \mathbf{L}_0 = \mathcal{A}^*\mathbf{d}, \mathbf{S}_0 = \mathbf{0}, \theta_0 = \zeta_0 = 1, t = \begin{cases} 1 : \text{ISTA} \\ 0.5 : \text{FISTA and POGM} \end{cases}$$

notation: $\mathbf{X}_i = \begin{bmatrix} \mathbf{L}_i \\ \mathbf{S}_i \end{bmatrix}, \tilde{\mathbf{X}}_i = \begin{bmatrix} \tilde{\mathbf{L}}_i \\ \tilde{\mathbf{S}}_i \end{bmatrix}, \bar{\mathbf{X}}_i = \begin{bmatrix} \bar{\mathbf{L}}_i \\ \bar{\mathbf{S}}_i \end{bmatrix}$

for $k = 1 \dots N$ **do**

$\tilde{\mathbf{L}}_k = \mathbf{M}_{k-1} - \mathbf{L}_{k-1}$

$\tilde{\mathbf{S}}_k = \mathbf{M}_{k-1} - \mathbf{S}_{k-1}$

if *ISTA* **then**

$\bar{\mathbf{X}}_k = \tilde{\mathbf{X}}$

else if *FISTA* **then**

$\theta_k = \frac{1}{2}\left(1 + \sqrt{1 + 4\theta_{k-1}^2}\right)$

$\bar{\mathbf{X}}_k = \tilde{\mathbf{X}}_k + \frac{\theta_{k-1}-1}{\theta_k}(\tilde{X}_k - \tilde{\mathbf{X}}_{k-1}) + \frac{\theta_{k-1}}{\theta_k}(\tilde{\mathbf{X}}_k - \mathbf{X}_{k-1})$

else if *POGM* **then**

$$\theta_k = \begin{cases} \frac{1}{2}\left(1 + \sqrt{1 + 4\theta_{k-1}^2}\right) : k < N \\ \frac{1}{2}\left(1 + \sqrt{1 + 8\theta_{k-1}^2}\right) : k = N \end{cases}$$

$\bar{\mathbf{X}}_k = \tilde{\mathbf{X}}_k + \frac{\theta_{k-1}-1}{\theta_k}(\tilde{X}_k - \tilde{\mathbf{X}}_{k-1}) + \frac{\theta_{k-1}}{\theta_k}(\tilde{\mathbf{X}}_k - \mathbf{X}_{k-1}) + \frac{\theta_{k-1}-1}{\zeta_{k-1}\theta_k}(\bar{\mathbf{X}}_{k-1} - \mathbf{X}_{k-1})$

$\zeta_k = t(1 + \frac{\theta_{k-1}-1}{\theta_k} + \frac{\theta_{k-1}}{\theta_k})$

$\mathbf{L}_k = SVT_{\lambda_L}(\bar{\mathbf{L}}_k)$

$\mathbf{S}_k = \mathbf{T}^*\left(\Lambda_{\lambda_S}\left(\mathbf{T}(\bar{\mathbf{L}}_k)\right)\right)$

$\mathbf{M}_k = \mathbf{L}_k + \mathbf{S}_k - \mathcal{A}^*(\mathcal{A}(\mathbf{L_k} + \mathbf{S_k}) - \mathbf{y})$

return $\mathbf{L}_N + \mathbf{S}_N$

Using ADMM technique, this formulation yields a modified augmented Lagrangian function

$$L(\mathbf{L},\mathbf{S},\mathbf{P},\mathbf{Q},\mathbf{V}_1,\mathbf{V}_2) = \frac{1}{2}\left\|\mathcal{A}(\mathbf{L}+\mathbf{S})-\mathbf{y}\right\|_2^2 + \lambda_L\left\|\mathbf{P}\right\|_* + \lambda_S\left\|\mathbf{Q}\right\|_1 + \ldots \\ + \frac{\delta_1}{2}\left\|\mathbf{L}-\mathbf{P}+\mathbf{V}_1\right\|_2^2 + \frac{\delta_2}{2}\left\|\mathbf{TS}-\mathbf{Q}+\mathbf{V}_2\right\|_2^2,$$

where $\mathbf{V}_1$ and $\mathbf{V}_2$ take the place of Lagrangian multipliers in the classic augmented Lagrangian method, and they come from the relaxation of equality constraints $\mathbf{P}=\mathbf{L}$ and $\mathbf{Q}=\mathbf{TS}$ to $\mathbf{L}-\mathbf{P}=\mathbf{V}_1$ and $\mathbf{TS}-\mathbf{Q}=\mathbf{V}_2$. Thus, setting $\mathbf{V}_1$ and $\mathbf{V}_2$ to $\mathbf{0}$ leads back to the solution of the constrained problem. These variables are adjusted after each step to penalize deviations from the equality constraints, specifically for each pixel.

The associated algorithm (algorithm 9) consists of four main steps, each optimizing L function for one variable while fixing all others. The minimization for $\mathbf{P}$ and $\mathbf{Q}$ is quite straightforward as they can solved by the proximity operators: the soft-thresholding solves for the ℓ_1-norm constraint on $\mathbf{Q}$ and SVT minimizes for the nuclear norm. Although these steps also needs to take care of the other two terms introduced by the equality constraints, they still have a directly solvable formula. On the other hand, optimization for $\mathbf{L}$ and $\mathbf{S}$ needs more elaborate steps. In fact, the most general solution is utilized by the authors of the paper; namely, they try to find the root of the gradient. As all terms containing $\mathbf{L}$ and $\mathbf{S}$ are ℓ_2 norms, the best (first-order) iterative method to find the root of these quadratic terms is certainly the conjugate gradient. The inputs of CG are derived as follows. Let us first take the partial derivate of the augmented Lagrangian:

$$\frac{\partial}{\partial L}L(\mathbf{L},\mathbf{S},\mathbf{P},\mathbf{Q},\mathbf{V}_1,\mathbf{V}_2) = \mathcal{A}^*(\mathcal{A}(\mathbf{L}+\mathbf{S})-\mathbf{y}) + \delta_1(\mathbf{L}+\mathbf{P}-\mathbf{V}_1) \stackrel{!}{=} \mathbf{0}.$$

Then we can separate $\mathbf{L}$ from the rest of the formula by rearranging

$$\mathcal{A}^*(\mathcal{A}(\mathbf{L}+\mathbf{S})-\mathbf{y}) + \delta_1(\mathbf{L}+\mathbf{P}-\mathbf{V}_1) = \mathcal{A}^*\mathcal{A}\mathbf{L} + \mathcal{A}^*\mathcal{A}\mathbf{S} - \mathcal{A}^*\mathbf{y} + \delta_1\mathbf{L} + \delta_1(\mathbf{P}-\mathbf{V}_1) = \ldots \\ = (\mathcal{A}^*\mathcal{A}+\delta_1\mathbf{I})\mathbf{L} + \mathcal{A}^*\mathcal{A}\mathbf{S} - \mathcal{A}^*\mathbf{y} + \delta_1(\mathbf{P}-\mathbf{V}_1).$$

Finally, putting it back to the previous equation and performing some further rearrangements, we get the equation

$$(\mathcal{A}^*\mathcal{A}+\delta_1\mathbf{I})\mathbf{L} = \mathcal{A}^*\mathcal{A}\mathbf{S} - \mathcal{A}^*\mathbf{y} + \delta_1\mathbf{P} - \mathbf{V}_1$$

that fits the CG scheme as $\mathcal{A}^*\mathcal{A}+\delta_1\mathbf{I}$ is Hermitian and positive definite. Following the same line of arguments, the equation needs to be minimized for $\mathbf{S}$ is derived as

$$(\mathcal{A}^*\mathcal{A}+\delta_2\mathbf{I})\mathbf{S} = \mathcal{A}^*\mathcal{A}\mathbf{L} - \mathcal{A}^*\mathbf{y} + \delta_1\mathbf{Q} - \mathbf{V}_2.$$

Algorithm 9: AL method with CG steps

input : $\mathbf{y}$: undersampled k-space data

$\mathcal{A}$: data acquisition operator

$\mathbf{T}$: temporal Fourier transform

λ_L: singular value threshold

λ_S: sparsity threshold

$maxIter_L$: number of conjugate gradient iteration steps for L

$maxIter_S$: number of conjugate gradient iteration steps for S

δ_1, δ_2: AL penalty parameters

N: number of iterations

output: $\mathbf{L}, \mathbf{S} = \arg\min_{\mathbf{L},\mathbf{S}} \|\mathcal{A}(\mathbf{L}+\mathbf{S}) - \mathbf{y}\|_2 + \lambda_L \|\mathbf{L}\|_* + \lambda_S \|\mathbf{TS}\|_1$

initialization: $\mathbf{L}_0 = \mathcal{A}^*\mathbf{y}, \mathbf{S}_0 = \mathbf{V}_{1,0} = \mathbf{V}_{2,0} = \mathbf{0}$

for $k = 1 \dots N$ **do**

$\mathbf{P}_k = SVT_{\lambda_L/\delta_1}(\mathbf{L} + \mathbf{V}_{2,k})$

$\mathbf{Q}_k = \mathbf{T}^*\Lambda_{\lambda_S/\delta_2}(\mathbf{T}S) + \mathbf{V}_{2,k}$

$\mathbf{L}_k = \arg\min_{\mathbf{L}}(\mathcal{A}^*\mathcal{A}+\delta_1\mathbf{I})\mathbf{L} = \mathcal{A}^*\mathbf{y} - \mathcal{A}^*\mathcal{A}\mathbf{S} + \delta_1(\mathbf{P} - \mathbf{V}_{1,k})$ starting from $\mathbf{L}_{k-1}$

$\mathbf{S}_k = \arg\min_{\mathbf{S}}(\mathcal{A}^*\mathcal{A}+\delta_2\mathbf{I})\mathbf{S} = \mathcal{A}^*\mathbf{y} - \mathcal{A}^*\mathcal{A}\mathbf{S} + \delta_2(\mathbf{Q} - \mathbf{V}_{2,k})$ starting from $\mathbf{S}_{k-1}$

$\mathbf{V}_{1,k} = \mathbf{V}_{1,k-1} + \mathbf{L}_k - \mathbf{P}$

$\mathbf{V}_{2,k} = \mathbf{V}_{2,k-1} + \mathbf{TS}_k - \mathbf{Q}$

return $\mathbf{L}_N + \mathbf{S}_N$

The second AL-type method that Lin and Fessler considered in their paper is a new method that decomposes the acquisition operator $\mathcal{A}$ into a coil sensitivity mapping $\mathcal{C} : \mathbb{C}^{N\times n_t} \to \mathbb{C}^{N\times n_c\times n_t}$ (briefly discussed in section 2.3.4), a spacial Fourier transform operator $\mathcal{F} : \mathbb{C}^{N\times n_c\times n_t} \to \mathbb{C}^{N\times n_c\times n_t}$, and an undersampling masking $\Omega : \mathbb{C}^{N\times n_c\times n_t} \to \mathbb{C}^{m\times n_c\times n_t}$ such that $\mathcal{A} = \Omega\mathcal{F}\mathcal{C}$. They also assumed that $\mathcal{C}$ normalized and therefore it is also unitary; i.e., $\mathcal{C}^*\mathcal{C} = \mathbf{I}$. They can assume it without loss of generality as the image can be freely scaled without changing its rank or sparsity, and this is also true for the temporal Fourier domain. At the construction of the algorithm, they took advantage of the increased freedom in the choice of variable splitting. In particular, they split the acquisition operator together with the variables, and they arrived to the objective

$$\arg\min_{\mathbf{L},\mathbf{S}} \min_{\mathbf{Z},\mathbf{X}} \left\{ \frac{1}{2}\|\Omega\mathbf{Z} - \mathbf{d}\|_2^2 + \lambda_L \|\mathbf{L}\|_* + \lambda_S \|\mathbf{TS}\|_1 \right\} \text{ subject to } \begin{cases} \mathbf{Z} = \mathcal{F}\mathcal{C}\mathbf{X} \\ \mathbf{X} = \mathbf{L} + \mathbf{S} \end{cases}.$$

Afterwards, they defined the AL function by

$$\arg\min_{\mathbf{L},\mathbf{S}} \min_{\mathbf{Z},\mathbf{X}} \left\{ \frac{1}{2} \|\Omega\mathbf{Z} - \mathbf{d}\|_2^2 + \lambda_L \|\mathbf{L}\|_* + \lambda_S \|\mathbf{TS}\|_1 \right\} + \frac{\delta_1}{2} \|\mathcal{F}\mathcal{C}\mathbf{X} - \mathbf{Z} + \mathbf{V}_1\|_2^2 + \frac{\delta_2}{2} \|\mathbf{L} + \mathbf{S} - \mathbf{X} + \mathbf{V}_2\|_2^2 ,$$

and they solved the resulting steps, as follows, exploiting that both $\mathbf{T}$ and $\mathcal{F}$ is unitary, introducing an auxiliary variable $\tilde{\mathbf{S}} = \mathbf{TS}$, and using again soft-thresholding and singular-value thresholding for ℓ_1 and nuclear norm minimization:

$$\begin{aligned}
\mathbf{Z}_k &= \arg\min_{\mathbf{Z}} \frac{1}{2} \|\Omega\mathbf{Z} - \mathbf{y}\|_2^2 + \frac{\delta_1}{2} \|\mathbf{X} - (\mathbf{L} + \mathbf{S}) + \mathbf{V}_2\|_2^2 \\
&= \left(\Omega^*\Omega + \delta_1\mathbf{I}\right)^{-1} (\Omega^*\mathbf{y} + \delta_1(\mathcal{F}\mathcal{C}\mathbf{X} - \mathbf{V}_1)) \\
\mathbf{X}_k &= \arg\min_{\mathbf{X}} \left\{ \frac{\delta_1}{2} \|\mathbf{Z} - \mathcal{F}\mathcal{C}\mathbf{X} + \mathbf{V}_1\|_2^2 + \frac{\delta_2}{2} \|\mathbf{X} - (\mathbf{L} + \mathbf{S}) + \mathbf{V}_2\|_2^2 \right\} \\
&= \left(\mathcal{C}^*\mathcal{C} + \frac{\delta_1}{\delta_2}\mathbf{I}\right)^{-1} \left(\mathcal{C}^*\mathcal{F}^*(\mathbf{Z} + \mathbf{V}_1) + \frac{\delta_1}{\delta_2}(\mathbf{L} + \mathbf{S} + \mathbf{V}_2)\right) \\
&= \frac{\delta_1}{\delta_1 + \delta_2} \left(\mathcal{C}^*\mathcal{F}^*(\mathbf{Z} + \mathbf{V}_1) + \frac{\delta_2}{\delta_1}(\mathbf{L} + \mathbf{S} - \mathbf{V}_2)\right) \\
\mathbf{L}_k &= \arg\min_{\mathbf{L}} \left\{ (\lambda_L \|\mathbf{L}\|_* + \frac{\delta_2}{2} \|\mathbf{X} - (\mathbf{L} + \mathbf{S}) + \mathbf{V}_2\|_2^2) \right\} \\
&= SVT_{\lambda_L/\delta_2}(\mathbf{X} - \mathbf{S} + \mathbf{V}_2) \\
\mathbf{S}_k &= \arg\min_{\mathbf{S}} \left\{ (\lambda_S \|\mathbf{TS}\|_1 + \frac{\delta_2}{2} \|\mathbf{X} - (\mathbf{L} + \mathbf{S}) + \mathbf{V}_2\|^2) \right\} \\
&= \mathbf{T}^* \left(\arg\min_{\tilde{\mathbf{S}}} \left\{ \lambda_S \left\|\tilde{\mathbf{S}}\right\|_1 + \frac{\delta_2}{2} \left\|\mathbf{T}(\mathbf{X} - (\mathbf{L} + \mathbf{S}) + \mathbf{V}_2) - \tilde{\mathbf{S}}\right\|_2^2 \right\} \right) \\
&= \mathbf{T}^* \Lambda_{\lambda_S/\delta_2}(\mathbf{T}(\mathbf{X} - \mathbf{L} + \mathbf{V}_2)).
\end{aligned}$$

Although these steps might look more complicated at first than the previous method, they can be calculated much faster indeed. In fact, the only inverse that needs to be calculated is $(\Omega^*\Omega + \delta_1\mathbf{I})^{-1}$, but this calculation is quite cheap because $\Omega^*\Omega$ is diagonal. Algorithm 10 summarizes the implementation of these steps, and describes the updates of the Lagrange multipliers.

Datasets and Results

To evaluate the convergence speed and time complexity of the examined algorithms, they performed numerical experiments on three Cartesian datasets and one another recorded via a non-Cartesian sampling trajectory. Two of the Cartesian images and the non-Cartesian one were published as a supplement to [88], and a simulated Cartesian dataset is taken from [92]—which, in turn, was generated by the MRXCAT toolbox [93]

Algorithm 10: improved AL method

input : $\mathbf{y}$: undersampled k-space data

$\mathcal{C}$: coil sensitivity operator

$\mathcal{F}$: spacial Fourier transform operator

Ω: undersampling masking

$\mathbf{T}$: temporal Fourier transform

λ_L: singular value threshold

λ_S: sparsity threshold

δ_1, δ_2: AL penalty parameters

N: number of iterations

output: $\mathbf{L}, \mathbf{S} = \arg\min_{\mathbf{L},\mathbf{S}} \|\mathcal{A}(\mathbf{L}+\mathbf{S}) - \mathbf{y}\|_2 + \lambda_L \|\mathbf{L}\|_* + \lambda_S \|\mathbf{TS}\|_1$

initialization: $\mathbf{X}_0 = \mathbf{L}_0 = \mathcal{C}^*\mathcal{F}^*\Omega^*\mathbf{y}, \mathbf{S}_0 = \mathbf{V}_{1,0} = \mathbf{V}_{2,0} = \mathbf{0}$

for $k = 1 \dots N$ **do**

$\mathbf{Z}_k = \left(\Omega^*\Omega + \delta_1\mathbf{I}\right)^{-1}(\Omega^*\mathbf{y} + \delta_1(\mathcal{F}\mathcal{C}\mathbf{X} - \mathbf{V}_1))$

$\mathbf{X}_k = \frac{\delta_1}{\delta_1+\delta_2}\left(\mathcal{C}^*\mathcal{F}^*(\mathbf{Z}+\mathbf{V}_1) + \frac{\delta_2}{\delta_1}(\mathbf{L}+\mathbf{S}-\mathbf{V}_2)\right)$

$\mathbf{L}_k = SVT_{\lambda_L/\delta_2}(\mathbf{X} - \mathbf{S} + \mathbf{V}_2)$

$\mathbf{S}_k = \mathbf{T}^*\Lambda_{\lambda_S/\delta_2}(\mathbf{T}(\mathbf{X} - \mathbf{L} + \mathbf{V}_2))$

$\mathbf{V}_{1,k} = \mathbf{V}_{1,k-1} + (\mathbf{Z}_k - \mathcal{F}\mathcal{C}\mathbf{X}_k)$

$\mathbf{V}_{2,k} = \mathbf{V}_{2,k-1} + \mathbf{X}_k - (\mathbf{L}_k + \mathbf{S}_k)$

return $\mathbf{L}_N + \mathbf{S}_N$

that aims to provide realistic numerical phantoms for cardiovascular magnetic resonance. Their results on all datasets were consistent proving the claimed improvement induced by the two methods they proposed. They found that POGM outperforms all other methods in case of all the four datasets, the "second place" is shared between FISTA and the improved AL scheme, and the AL method with CG inner steps appears to be the slowest in both convergence rate and time complexity. The latter is mainly due to the time consuming CG steps, but also the improved AL algorithm is computationally more expensive. This comparison, however, is slightly unfair since it fails to consider a very important advantage of ADMM-like methods, notably the independence of steps that allows highly parallel implementations. Nonetheless, the primacy of POGM—which, in fact, is the most important result of the research project—is inevitable.

4.2. Multiscale Low Rank Decomposition

A similarly outstanding result was published by Frank Ong in the form of a doctoral dissertation [94] in 2018, and as a preprint of a journal paper [95] submitted in December, 2019 (accepted in April, 2020 [96]). The novelty of his approach is that he makes the so called *multiscale low rank* (MSLR) scheme practically feasible and demonstrates its effectiveness in MRI setting. The idea was first proposed by McCoy et al. [97]–[99], and it naturally extended the $L + S$ scheme by filling the "gap" between the large scale low rankness, which is a global property, and pixel level sparsity, both being unable to capture the local structures. Dynamic MR images of the full body, for instance, exhibits large scale motions such as respiratory motion or change in body posture. On the other hand, heart beats and motions of the gastrointestinal tract are better described in a smaller scale, and movements of veins have effect on an even smaller scale.

The MSLR approach models this structure in data as a sum of block-wise low rank matrices with increasing scales of block sizes. To put it into practice, let us denote the image of interest with $\mathbf{X}$, and let us consider the decomposition

$$\mathbf{X} = \sum_{j=1}^{J} \mathbf{M}_j,$$

where J is the number of scales selected beforehand and $\mathbf{M}_j$ are block matrices with a block size corresponding to the granularity of scale j. For a scale j, let us also define N_j, B_j, K_j as the number of pixel within a block, the number of blocks, and the maximum block matrix, respectively. As high resolution 3D images might need huge storage, as we seen earlier, so usually we want to somehow reduce the memory demand. Exploiting the low rankness of the different scales, the authors proposed not to work with the image directly, but rather on the stacked spatial and tempral bases $\mathbf{L}_j in \mathbb{C}^{N-j \cdot B_j \times K-j}$ and $\mathbf{R}_j in \mathbb{C}^{n_t \cdot B_j \times K-j}$ Then the operator $\mathcal{M} : \mathbb{C}^{N_j \cdot B_j \times n_t \cdot B_j} \rightarrow \mathbb{C}^{N \times n_t}$ is defined to embed stacked input to an image such that

$$\mathbf{X} = \sum_{j=1}^{J} \mathcal{M}_j(\mathbf{L}_j \mathbf{R}_j).$$

Putting it together with the MR image acquisition scheme, we get

$$\mathbf{y} = \mathcal{A}\left(\sum_{j=1}^{J} \mathcal{M}_j(\mathbf{L}_j \mathbf{R}_j^*)\right) + \mathbf{W},$$

where $\mathcal{A}$ is the acquision operator, as earlier, and $\mathbf{W}$ is a complex Gaussian white noise matrix. Following the same line of argument as before, to enforce $\mathbf{X}_j$ to be low rank in

its blocks, we can use again the nuclear norm relaxation, leading to the cost function

$$\frac{1}{2}\left\|\mathbf{y}-\mathcal{A}\left(\sum_{j=1}^{J}\mathcal{M}_j(\mathbf{L}_j\mathbf{R}_j^*\right)\right\|_2^2+\sum_{j+1}^{J}\lambda_j\left\|\mathbf{X}_j\right\|_{(j)},$$

where $\|\cdot\|_{(j)}$ denotes the block-wise nuclear norm defined as the sum of the nuclear norms of blocks. They propose choosing the parameter λ_i as it was suggested in [100]:

$$\lambda_j \propto \sqrt{N_j}+\sqrt{n_t}+\sqrt{2logB_j}.$$

That way, tuning only one scaling parameter for all λ_j is sufficient. Nevertheless, the problem of too large memory requirement still present because of the $\|\mathbf{X}_j\|_{(j)}$ term. Instead, they further relaxes the problem by the Burer-Monteiro factorization [101], [102]:

$$\|\mathbf{X}\|_*=\overset{H}{\underset{\mathbf{X}=\mathbf{LR}}{\min}}\frac{1}{2}\left(\|\mathbf{L}\|_F^2+\|\mathbf{R}\|_F^2\right),$$

and they change the cost function accordingly:

$$\frac{1}{2}\left\|\mathbf{y}-\mathcal{A}\left(\sum_{j=1}^{J}\mathcal{M}_j(\mathbf{L}_j\mathbf{R}_j^*\right)\right\|_2^2+\sum_{j+1}^{J}\frac{\lambda_j}{2}\left(\|\mathbf{L}_j\|_F^2+\|\mathbf{R_j}\|_F^2\right).$$

While this problem becomes non-convex due to this relaxation, the gradient methods usually finds a decent solution. And to reduce the still prohibitively large time complexity, they optimize the cost function via stochastic gradient descent (SGD) that splits the cost function to a sum of partial cost functions on single frame and single coil:

$$f_{tc}(\mathbf{L},\mathbf{R})=\frac{1}{2}\left\|\mathbf{y_{tc}}-\mathcal{A}_{tc}\left(\sum_{j=1}^{J}\mathcal{M}_j(\mathbf{L}_j\mathbf{R}_{jt}^*\right)\right\|_2^2+\sum_{j+1}^{J}\frac{\lambda_j}{2}\left(\frac{1}{n_tn_c}\|\mathbf{L}_j\|_F^2+\|\mathbf{R_j}\|_F^2\right),$$

and they perform the gradient steps as

$$\mathbf{L}=\mathbf{L}-\alpha n_tn_c\nabla_L f_{tc}(\mathbf{L},\mathbf{R}\text{ and }\mathbf{R}_t=\mathbf{R}_t-\alpha n_c\nabla_{R_t}f_{tc}(\mathbf{L},\mathbf{R},$$

or even running mini-batches as

$$\mathbf{L}=\mathbf{L}-\alpha\frac{n_tn_c}{G}\sum_{(t,c)\in\mathcal{I}}\nabla_L f_{tc}(\mathbf{L},\mathbf{R}\text{ and }\mathbf{R}_t=\mathbf{R}_t-\alpha\frac{n_c}{G}\sum_{(t,c)\in\mathcal{I}}\nabla_{R_t}f_{tc}(\mathbf{L},\mathbf{R},$$

where α is the step size selected empirically (starting from one, it is decreased each time the iteration diverges) $\mathcal{I}$ contains coil and frame indices selected into the current mini-batch, and G is the size of the mini-batch (often determined by the number of available CPUs or GPUs).

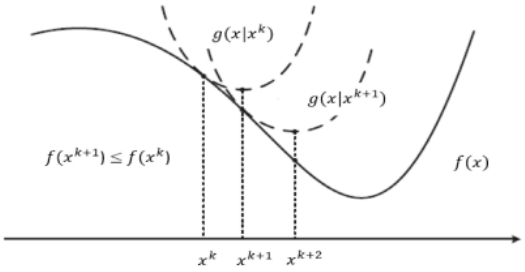

Figure 4.1: Intuition for IRLS. the IRLS scheme majorizes the cost function in the locality of the solution of the previous step by a quadratic function, and then solves for it (note that this general scheme is called majorize-minimize or MM methods). If the quadratic function is chosen appropriately, then each iteration brings closer to the solution. Source: [111].

4.3. Iteratively Reweighted Least Squares Methods

Iteratively reweighted least squares methods have a long history in the compressed sensing framework; however, they got less attention recently despite of some strong results [103]–[107] mostly based on the FOCUSS scheme [108]. Conventionally these methods are used as a proxy for the ℓ_1, but it is even more powerful for ℓ_p-norms with $p < 1$. In particular, it has been proved compared to the linear convergence rate of for ℓ_1-norm, it enjoys superlinear for ℓ_p-quasinorms under the restricted isometry property [109], [110].

In general, these algorithms have three main steps, namely

1. $X_{k+1} = \arg\min_{\mathbf{X}} \mathcal{J}(\mathbf{x}, \mathbf{W}, \epsilon)$
2. update the smoothing parameter ϵ
3. update the weights $\mathbf{W}$

A good intuition of this process is imagining a complicated cost function, which makes difficult directly finding the minimizer. Instead, the IRLS scheme majorizes the cost function in the locality of the solution of the previous step by a quadratic function, and then solves for it (note that this general scheme is called majorize-minimize or MM methods). If the quadratic function is chosen appropriately, then each iteration brings closer to the solution, and therefore we need to find a good update step for ϵ and $\mathbf{W}$. Fig. 4.1 illustrates this process in a 2D case.

This framework is a powerful tool to solve many complicated problems, and it serves as a proxy to a very wide range of functions. In [112], [113] the authors con-

sidered a particularly difficult problem: the rank minimizaton. And in constrast to the nuclear-norm minimization scheme we seen earlier, they have chosen the log-det function as a surrogate to minimize over. It is easy to see how much more powerful this approach is since contrary to nuclear norm, which is only an approximation of the rank, the log-det is equal to the rank. To illustrate that, let us recall that

$$log(x) = \lim_{p \to 0} \frac{x^p - 1}{p}.$$

Than take $x = \sigma_i(\mathbf{A})$ for some $\mathbf{A}$, and accordingly

$$log(\sigma_i(\mathbf{A})) = \lim_{p \to 0} \frac{\sigma_i(\mathbf{A})^p - 1}{p}.$$

Next, we make some simple steps

$$\sum_{i=1}^{R} log(\sigma_i(\mathbf{A})) = \lim_{p \to 0} \sum_{i=1}^{R} \frac{\sigma_i(\mathbf{A})^p - 1}{p} = \frac{\lim_{p \to 0} \sum_{i=1}^{R} \sigma_i(\mathbf{A})^p}{p} - \frac{R}{p},$$

and we know that

$$\lim_{p \to 0} \sum_{i=1}^{R} \sigma_i(\mathbf{A})^p = rank(\mathbf{A}).$$

Finally,

$$\sum_{i=1}^{R} log(\sigma_i(\mathbf{A})^p) = log(\prod_{i=1}^{R} \sigma_i(\mathbf{A})^p) = log(det(\mathbf{A})),$$

and

$$\frac{\lim_{p \to 0} \sum_{i=1}^{R} \sigma_i(\mathbf{A})^p}{p} - \frac{R}{p} = \frac{\|\mathbf{A}\|_p^p}{p} - \frac{R}{p},$$

where $\|\mathbf{A}\|_p^p$ is a Schatten-p norm.

Although a deeper discussion of this new algorithm, named Harmonic-Mean-IRLS or HM-IRLS, would be interesting, due to the limitation of this work we need to refer the reader to the doctoral thesis [113] of Christian Kümmerle, or to the the our github repository holding a reference implementation: `https://github.com/hakkelt/IRLS-for-CS-MRI/blob/master/SimpleMatrixCompletion.ipynb`, as the discussion would require a much longer introduction to mathematical concepts. A important property, nevertheless, worth mentioning here: the authors constructed the HM-IRLS to use some second-order information, and as a result of this propoerty, it converges really fast, admitting superlinear convergence in general and even quadratic convergence locally. On the other hand, it also means more heavy computation as well.

Chapter 5

Contributions

While Julia produces usually faster code than other languages at the same abstraction level for numerical application, the true power of the language is only shown when the programmer writes the code following a couple performance tips listed on the official documentation. One of the most commonly occurring mistake that results suboptimal code is the high number of automatic memory allocations that leads to more frequent calls to garbage collector, and in other words, it wastes resources. Therefore in the following presentation of the programming-related contributions we will highlight the memory-efficiency at the evaluation of the created code base.

5.1. FunctionOperators package

As we have seen so far in the descriptions of the algorithms, operations very often are expressed as a multiplication a matrix-like entity and a vector or matrix. That notation is particularly convenient as it allows using the same set of mathematical tools as we have for matrices, while allows extension of the capabilities of the conventional linear mapping defined as a matrix-vector multiplication. In practice that means that we would like to combine the flexibility of functions with the power of tools designed for matrices, so many times the best option is to define the operation we want to perform as a function, and then wrap this function in an object that acts like a matrix and therefore it is compatible for instance with iterative solvers like the conjugate gradient method. Also it is desirable to add an "backward" operation, also defined by a function, to that wrapper object, that defines what should the solver do when it wants to use the adjoint of the operator.

Being driven by mostly scientists, the Julia community has already developed some

packages that helps keeping the code structurally and visually similar to the abstract mathematical notation; nevertheless, none of these were completely satisfying. In particular, there are three relatively popular packages that have such functionality, but all of them some drawbacks:

- LinearOperators.jl [114] and LinearMaps.jl [115] aims to provide almost all features of general matrices, but this design choice restricts the possible inputs to vectors.

- AbstractOperators.jl [116] is a fairly new package that overcomes this limitation and provides a relatively large range of features, mostly in the form of predefined operators for the most common operations such as DFT, convolution, finite differences etc. It is also memory effective, preallocating a buffers when two operators are composed (that corresponds to the matrix-matrix multiplication) and it reuses this buffer later avoiding unnecessary memory allocations. Unfortunately, this optimization makes it impossible to accept on GPU arrays that makes unfavorable for large-scale applications.

As Julia is designed to be very extendable, it is always a feasible option to develop a new package that fits the specific problem. After reviewing the code of the mentioned packages, we came to the conclusion that the design choices of these packages doen't allow addition of features we desired, without breaking the already existing features, we decided to implement a package that fits better the image CS-MRI reconstruction framework. The developed package is named FunctionOperators.jl and it is already published to the central Julia package repository. Being just after the initial phase, the package supports only the most basic features, but the design of the interface and the efficient implementation lets the user build arbitrarily complex composite operators without having any computation penalty compared to the implementation with pure functions. As of today, the already implemented features if the package are the following:

- Construction from a function with one argument that defines "forward" operation. When the constructed FunctionOperator is being multiplied with a vector/matrix of the *proper* size, this function is called on the given vector/matrix. The size of the *proper* input and expected output is a mandatory argument of the constructor of FunctionOperator, and any input with a mismatching size is rejected, and it after the multiplication the size of the output is also checked against the output

size specified at construction. E.g., the following code calculates the FFT of x and stores in y:

```
Op = FunctionOperator(forw = x -> fft(x))
y = Op * x
```

- Construction from two functions both accepting one argument. These functions define the "forward" and the "backward" operations. In contrast to the "forward" function, the "backward" function is called when adjoint of the FunctionOperator is requested. E.g., the following calculates the inverse FFT of x and stores in y:

```
Op = FunctionOperator(forw = x -> fft(x), backw = x -> ifft(x))
y = Op' * x
```

- These "forward" and "backward" functions also can accept two arguments. In that case, the second argument is the input of the operation, and the functions are expected to store the result of the operation in the first argument. This feature is advantageous if one wants to optimize the number of memory allocations. E.g., the following multiplies x elementwise with two and stores the result in y without allocating an intermediate array.

```
Op = FunctionOperator(forw = (b,x) -> b .= x .* 2)|| mul!(y, Op, x)
```

- Composition of FunctionOperators by multiplication (that means composition of the "forward" functions), addition and substraction (these adds/substracts the output of the operations). E.g. considering the following the last line evaluates to true:

```
Op1 = FunctionOperator(forw = x -> fft(x))
Op2 = FunctionOperator(forw = (b,x) -> b .= x .* 2)
Op1 * Op2 * x == fft(x .* 2)
```

- Composition of FunctionOperators with UniformScaling object from LinearAlgebra standard library. This UniformScaling operator corresponds to the identity matrix. E.g. we can define the MR acquisition operator $\mathcal{A}$ as a FunctionOperator, and then we can define the CG operator in algorithm 9 as $\mathcal{A}' * \mathcal{A} + \delta_1 I$ (it is not the abstract mathematical notation, but a valid Julia expression because Julia also accepts Unicode characters in the identifiers!) and then we can pass it to a a solver (IterativeSolver.jl has, for example, CG solver that uses duck-typing, so it requires only that the multiplication must be implemented on the object passed as argument).

- Adjoint of composit FunctionOperators: If the FunctionOperator is a composition of other FunctionOperators, then the adjoint is defined as the composition of adjoints of the member FunctionOperators, in the reverse order. E.g., let us recall the decomposition of the acquisition operator $\mathcal{A} = \Omega\mathcal{F}\mathcal{C}$. Now look on it on the other way around: we have $\Omega, \mathcal{F}$ and $\mathcal{C}$ already created as FunctionOperators, and we create the acquisition operator as the composition of them: $\mathcal{A} = \Omega * \mathcal{F} * \mathcal{C}$. Then $\mathbf{A}' * x$ would do the same as $\mathcal{C}' * \mathcal{F}' * \Omega' * x$.

Under the hood, the most important property of the package that it uses the least amount of memory possible by deferring the allocation of buffers as much as possible and also by maintaining a smart global pool that stores the already allocated buffers. These buffers are only needed to store intermediate results, and therefore they are safe to reuse in different FunctionOperators provided that they are on the same thread. But the user don't need to worry about that criterion since this case is handled automatically, making FunctionOperators a thread-safe package (assuming that the user builds the operators from thread-safe "forward" and "backward" functions). This memory effective implementation makes FunctionOperators a truly unique as other packages are all more or less suboptimal in this sense.

As an extra (yet experimental) feature, FunctionOperators package provides a macro that automatically optimizes loops by cutting down the number unnecessary memory allocations. To achieve this the macro uses the advanced code generation features that produces code before the compilation of the program, but after the parser has built the abstract syntax tree and deduced the type of variables; therefore, heavy optimizations can be performed behind the scenes using macros and generated functions.

Following the guidelines of the Julia community for package development, the code is thoroughly tested using unit tests, and has a clean documentation that contains a notebook with an example for almost all features. The coverage of unit tests are measured by `codecov.io`, and it reported that the 94% of the code is covered. This report, however, underestimates the coverage as it fails to detect the covered lines in case of tests checking the functions that prints to console. The code of the package is uploaded to `https://github.com/hakkelt/FunctionOperators.jl`, and the documentation is available at `https://hakkelt.github.io/FunctionOperators.jl/latest/`.

5.2. Implementation of Sparse+Low Rank algorithms

After the careful examination of both the publication and the reference Matlab implementation, we created an efficient Julia version for each of these algorithms. As the implementation was done at the same time as the FunctionOperators was developed, it was a natural choice to benchmark the newly developed package against the other similar packages on these algorithms. The contribution of this part of the project is two-fold: First, it extends the currently small code base of MRI-related algorithms in Julia, helping the fast growing group of scientists choosing to prototype their research algorithms in Julia. Second, it might also help those who needs guidance in picking the right package, especially because currently no other comparison is available to see the difference between LinearMaps.jl and AbstractOperators.jl.

The result of the benchmarking is available in table 5.1 and the Jupyter notebooks holding both the benchmarking code, the documentation, and the results are available at `https://github.com/hakkelt/reproduce-l-s-dynamic-mri-julia` In comparison to the Matlab implementation, the results are somewhat mixed since Julia outperformed Matlab in case of the improved AL scheme (the speedup here was 2x) and in case of proximal methods for the non-Cartesian dataset (with 2-3.5x speedup). On the other hand, Matlab produced faster code for the other cases. This results underlines the fact that merely switching the language to Julia not necessarily results in increased speed automatically. A possible explanation is that we missed some hidden optimization tricks deeply buried in the Matlab code (which was well optimized indeed, and very hard to read—it required a fair amount of time from us to understand how is it connected to the theory described in the paper). Another possible factor is that Matlab uses some proprietary C and fortran libraries that have slightly better performance compared to the open source options bundled with Julia by default.

Furthermore, the comparison of different Julia implementations revealed that the three package have very similar running time (that matches our expectations as these packages are meant to be "invisible" in performance benchmarks being only wrappers around some functions), but they rahter differ in the memory allocated, LinearMaps.jl having the largest memory demand (as it was expected from the implementation that allocates and releases buffers in each iteration again and again in our case), and FunctionOperators being better than AbstractOperators by a small margin. We also tested the automatic optimization macro mentioned above, and the benchmarks showed that

it reached almost optimal memory usage, reducing the size of net allocations by 70% on average, compared to the 80% reduction achieved by the tedious process of manual optimization.

5.3. Implementation of Multiscale Decomposition

The implementation of multiscale low rank algorithm described in [95], posed multiple challenges. First, the algorithm was written in a framework, also developed by the author, called SigPy [117]. Practically that meant that we needed to re-implement a fairly large part of that library in Julia. While it would have been possible to directly call the methods from this framework, these blocking would make the paralellization of the Julia code practically impossible. The other largest challenge was that the author used a special version of the non-uniform FFT (NFFT), also from his package, and therefore we also needed to re-implement this non-trivial method as well because it was not compatible with the Julia implementation of NFFT. However, this difficulty later turned out to be beneficial, as we realized a performance problem in the author's code, and correcting it in our implementation doubled its speed. But first let us discuss shortly the NFFT algorithm.

5.3.1 NFFT

As the name suggests, non-uniform FFTs are the non-Cartesian variant of the FFT, and therefore they are very useful in MRI reconstruction with real-life data. Also, similarly to the FFT which is the fast version of DFT, non-uniform FFT is also derived from non-uniform DFT. There are many possibilities how to calculate them efficiently, but the most widely used variant is proposed by Fessler [118]. This variant maps the off-grid coordinates determined by the non-Cartesion trajectory to the closest grid point and to its neighbor pixels, using a kernel (most commonly the Kaiser-Bessel function) to weight this projection. Then, it applies the ordinary FFT to the grid filled with values by the mapping. Finally, it utilizes some technique, known as apodization in the photography, that corrects the artifacts introduced by the undersampling and the mapping to the grid. The large popularity of this approach is that there are many highly optimized FFT solutions, and therefore the speed of NFFT only depend on the effective mapping (the time complexity of apodization is negligible compared to the mapping).

In our implementation we used the FFTW library to compute the FFT, which is currently the fastest open source solution (and some even claim that it outperform the

algorithm \data set	PINCAT	Multicoil cardiac cine MRI	Multicoil cardiac perfusion MRI	Multicoil abdominal dce MRI
		Matlab		
AL-CG	16.8 s	49.0 s	19.9 s	-
AL-2	17.8 s	55.3 s	27.5 s	-
ISTA	1.7 s	5.5 s	2.3 s	141.8 s
FISTA	2.4 s	7.6 s	3.1 s	256.7 s
POGM	1.8 s	5.8 s	2.3 s	140.0 s
		LinearMaps		
AL-CG	28.4 s, 11.26 GiB	80.8 s, 31.08 GiB	33.8 s, 12.95 GiB	-
AL-2	9.8 s, 6.22 GiB	28.4 s, 17.56 GiB	11.0 s, 7.32 GiB	-
ISTA	6.0 s, 627.71 MiB	19.6 s, 1.36 GiB	6.8 s, 582.06 MiB	72.7 s, 2.18 GiB
FISTA	6.2 s, 627.71 MiB	16.5 s, 1.36 GiB	6.8 s, 582.06 MiB	73.2 s, 2.18 GiB
POGM	6.3 s, 677.71 MiB	16.8 s, 1.45 GiB	6.8 s, 622.06 MiB	72.6 s, 2.42 GiB
		AbstractOperators		
AL-CG	28.7 s, 678.06 MiB	76.1 s, 1.45 GiB	32.1 s, 622.41 MiB	-
AL-2	9.0 s, 1.14 GiB	23.3 s, 2.93 GiB	10.2 s, 1.22 GiB	-
ISTA	6.4 s, 627.68 MiB	16.4 s, 1.36 GiB	7.3 s, 582.03 MiB	72.8 s, 2.18 GiB
FISTA	6.5 s, 627.68 MiB	16.3 s, 1.36 GiB	7.3 s, 582.03 MiB	72.0 s, 2.18 GiB
POGM	6.6 s, 677.68 MiB	16.5 s, 1.45 GiB	7.0 s, 622.03 MiB	73.9 s, 2.42 GiB
		FunctionOperators naive		
AL-CG	30.6 s, 2.90 GiB	79.4 s, 6.14 GiB	33.3 s, 2.43 GiB	-
AL-2	11.6 s, 12.20 GiB	32.0 s, 33.17 GiB	13.1 s, 13.82 GiB	-
ISTA	6.8 s, 3.40 GiB	18.1 s, 8.67 GiB	7.6 s, 3.62 GiB	72.7 s, 6.71 GiB
FISTA	6.9 s, 3.40 GiB	17.6 s, 8.67 GiB	7.5 s, 3.62 GiB	74.9 s, 6.71 GiB
POGM	6.9 s, 3.45 GiB	17.8 s, 8.77 GiB	7.5 s, 3.65 GiB	73.741 s, 6.96 GiB
		functionOperators optimized		
AL-CG	27.0 s, 640.69 MiB	76.2 s, 1.38 GiB	33.0 s, 592.54 MiB	-
AL-2	8.6 s, 1.04 GiB	22.7 s, 2.65 GiB	10.0 s, 1.11 GiB	-
ISTA	6.1 s, 627.79 MiB	16.6 s, 1.36 GiB	7.6 s, 582.14 MiB	75.1 s, 2.18 GiB
FISTA	6.1 s, 627.79 MiB	16.5 s, 1.36 GiB	7.9 s, 582.14 MiB	73.8 s, 2.18 GiB
POGM	6.3 s, 677.79 MiB	16.747 s, 1.45 GiB	7.079 s, 622.14 MiB	73.6 s, 2.42 GiB
		FunctionOperators pretty		
AL-CG	28.0 s, 741.02 MiB	75.1 s, 1.57 GiB	30.7 s, 662.87 MiB	-
AL-2	8.4 s, 1.26 GiB	21.8 s, 3.26 GiB	9.3 s, 1.36 GiB	-
ISTA	6.3 s, 1.05 GiB	16.4 s, 2.49 GiB	6.9 s, 1.04 GiB	73.5 s, 3.03 GiB
FISTA	6.3 s, 1.05 GiB	16.3 s, 2.49 GiB	6.9 s, 1.04 GiB	72.2 s, 3.03 GiB
POGM	6.4 s, 1.10 GiB	16.5 s, 2.58 GiB	6.9 s, 1.08 GiB	74.5 s, 3.28 GiB

Table 5.1: Benchmarking on different datasets performing reconstruction via proximity and augmented Lagrangian methods descirbed in [90]. "FunctionOperators naive" is an implementation with minimal manual optimizations, "FunctionOperators optimized" is a manually optimized version, and ""Function-Operators pretty" is exactly same as the "naive" version except that our automatic automatic optimizer macro is called on the code. Environment of tests: 48 Intel Xeon CPUs and 385 GiB memory, Julia 1.4.

commercial ones). This library supports also multithreading that help us also speeding up the code. When we compared our implementation to the implementation in SigPy, we considered three multithreading settings: a fully singly threaded, a partially single threaded (the Julia code is single threaded, and the multithreading in FFTW is allowed), and a fully multithreaded. Concerning the size of the problem: We tested the implementation for three sizes, a small sized random image (64×64) and relative few sampling points (1024), a "medium sized" where the image were relatively large with size of $128 \times 128 \times 128$, but the number of sampling point were still low compared to the size of the image ($16k$ sampling points), and the "large" problem where the image size is reduced to keep the problem feasible, but the number of sampling points were large. In case of the "medium" and "large" data, a batch dimension of size 12 is also provided that means that no Fourier transformation is done along that direction, but the same operation takes place on each slice defined by this direction.

The numerical results, then showed us that the best multithreading setting is depends on the problem, as we expected. Moreover, the multithreading in FFTW brings larger benefits when the size of input image is large (in these cases the FFT is the dominant step in the NFFT), while enabling the Julia multithreading (these threads then distributes the sampling between each other) has a large positive impact when the number of sampling points are sufficently large (let's say over $1M$ points). In the MRI cases, the latter is not an infrequent situation. For a better insights to the results, we refer the reader to fig. 5.1-5.3. In these figures we show two different versions of the Julia implementation: one contains the initialization steps, the other not. While the former is good enough when we want to evaluate the NFFT only once, the latter is good when we perform the same operation multiple times; e.g., in a loop. In these cases, one might want to avoid repeated allocations in the initialization part, and therefore our implementation makes it possible to create a *plan* and then apply this plan to the data avoiding the repeated initialization.

5.3.2 MSLR Algorithm

After finishing the re-implementation of the necessary parts of SigPy in Julia, our attention is turned towards the implementation of the algorithm itelf that was not a challenge any more as almost all important operators and functions are defined in SigPy. Being aware of the speedup in case of the re-implementation of the NFFT, we expected similar acceleration. Yet, the result were even better as we predicted. As we already

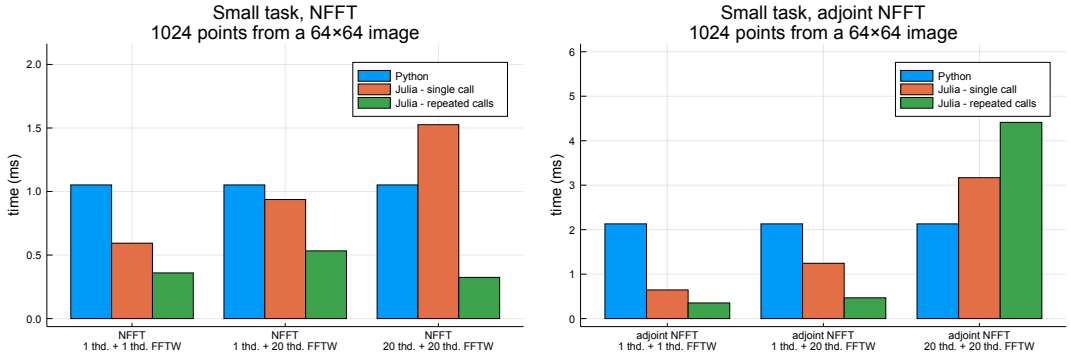

Figure 5.1: Comparison of running time between the Python implementation in SigPy package (blue), and the Julia implementations (orange: initialization and computation), green: computation only) at different threading settings **in the case of a small problem** (small input image with relatively few sampling points). **For small images, single threaded Julia version is the fastest.** (Note that the Python implementation does not implement multithreading, and hence the height of the three blue bar is the same.)

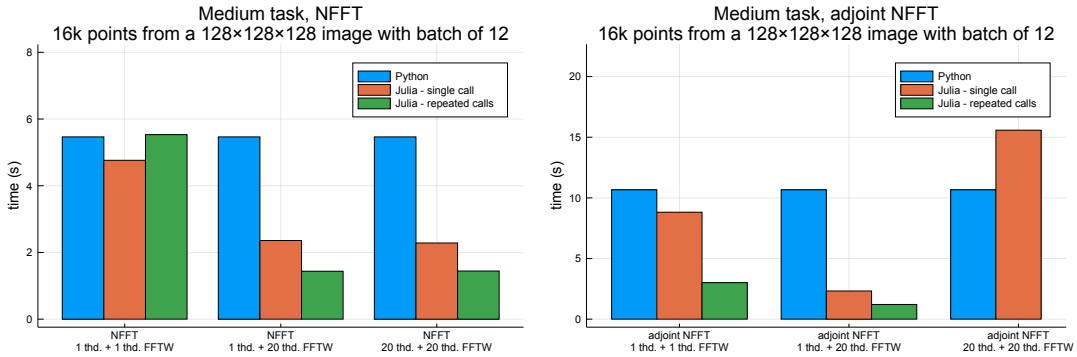

Figure 5.2: Comparison of running time in the medium sized setting: Single threaded Julia + multi-threaded FFTW yielded the best performance.

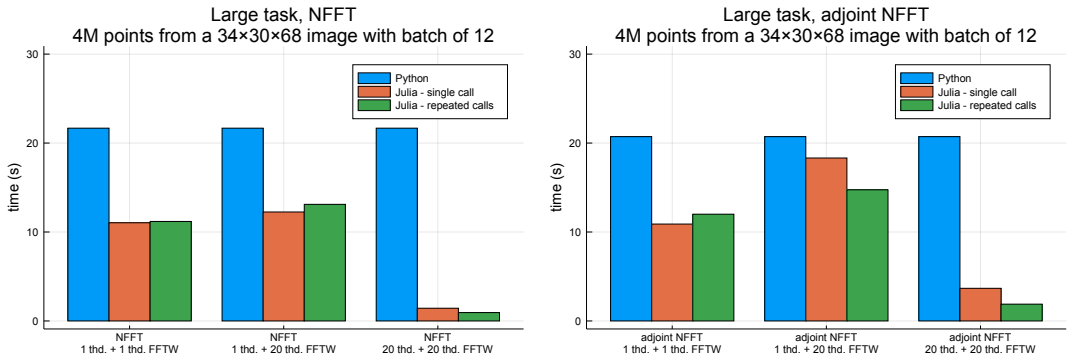

Figure 5.3: Comparison of running time in case of large number of sampling points: Multithreaded Julia code + multithreaded FFTW were the fastest.

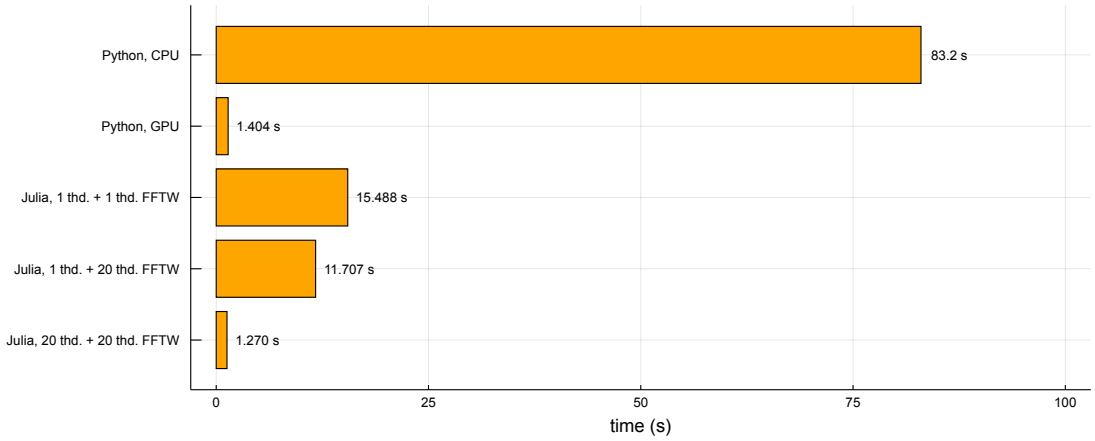

Figure 5.4: Comparison of speed different implementations for the gridding reconstruction problem (i.e. a large scale adjoint NFFT with batch dimensions).

mentioned, the Python implementation did not exploit the potentials in the batch processing in NFFT. Specifically, it is much faster to pass the image the NFFT specifying along which dimension we want to perform the NFFT, than iterating over th batch dimension (= the dimension along which we *do not* want to perform the NFFT) and calling the NFFT on each slice.

Fig. 5.4 demonstrates the effect: Here the measured algorithm technically consists only of a loop that iterates over the batch dimension corresponding to the coils sensitivities, and calls NFFT on each. On the other hand, the Julia version passes the entire arraw to NFFT. The result is drastic, $5.5\times$ speedup, even in the fully single threaded case. But when we anable the full multithreading, algorithm appeared to outperform even the Python's GPU version. Fig. 5.5 does not show so much improvement, but still many-fold speedup is experienced, with the fully multithreaded almost reaching the Pyhton's GPU version.

These results suggest that re-implementing the Python'GPU code in Julia would enjoy, maybe not that much, but significant acceleration. For more information concerning the implementational details, we refer the reader to the GitHub repository of the project: `https://github.com/hakkelt/extreme_mri_julia` Environment of measurements: 56 Intel Xeon CPUs, 2 K80 GPU, and 385 GiB memory, Julia 1.4.

5.4. HM-IRLS Implementation

While the plan at the beginning of the project was to apply the new HM-IRLS method to the $L + S$ problem, the robust version of this algorithm is still not published at the time

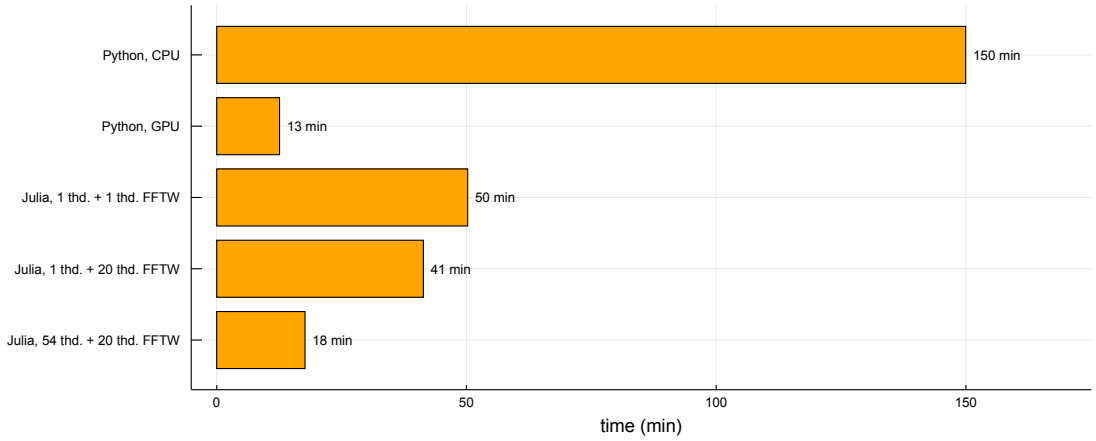

Figure 5.5: Comparison of speed for different implementation of the MSLR algorithm. Note that the heavily multithreaded Julia implementation was almost as fast as the Python GPU implementation. Also even the fully single threaded Julia implementation was more than three times faster than the Python CPU implementation.

of the writing, unfortunately. Therefore we were limited make experiments with the low-rank-only variant of the algorithm. To make the comparison to the other method somewhat balanced, we used the simulated dataset from [93], and we modified it to produce multiple slightly different datasets, and made experiments on how the parameters of these dataset-variant effects the recovery process. That way we could look at the effect of noise, undersampling ratio, and sparseness. By undersampling, of course, we mean here the undersampling *of the Fourier domain*.

The modification process consisted of a singular value hard thresholding that selected the r largest singular values, and discarded all other. As a result, the thresholding produced an image with the desired rank r. Next, we subtracted this low-rank approximation from the original image, and we added back the residual after hard thresholding to the low-rank image. The resulted image served then as "ground truth" image, which we undersampled, and added some noise to the undersampled data. The process in a more exact form:

$$\mathbf{L} = SVHT_r(\mathbf{X})$$

$$\mathbf{X}_{gt} = \mathbf{L} + HT_s(\mathbf{X} - \mathbf{L}),$$

where $SVHT_r$ is the singular value hard thresholding, and HT_s is the hard thresholding that discard all, but the s largest values in a matrix. The last step of the construction of the modified setting was the undersampling where we were in full control of the number of sampled point from the Fourier domain.

In the testing stage, we examined how much can the different algorithms recon-

struct the "ground truth" from the undersampled, noisy measurement. Our experiments showed that the HM-IRLS algorithm performed extremely well in the noiseless case and when $s = 0$, that is, the ground truth image was only the low rank component. The results also confirms the claims that POGM converges the fast, FISTA and the improved AL being the second fastest, but finally all of the algorithms designed to minimize the $L + S$ cost converged to the same minimizer after a long time. The MSLR algorithm, however, reached a much worse local minimizer than the other algorithms. Even though it might be surprising at first, the difference in the reach optimizer is fairly reasonable as HM-IRLS solved the low-rank minimization directly, the ones with the $L + S$ cost solved a relaxed problem that minimized nuclear norm instead of the rank, and, finally, the MSLR even relaxed the the nuclear norm even further by the Burer-Monteiro factorization to reduce the memory requirement of the algorithm (therefore it is slightly unfair comparison to others which care less about memory consumption). Moreover, we checked the behavior of the rank during the iterations, and we noted that the while the HM-IRLS algorithm decreased the rank gradually, the soft-thresholding-based algorithms reached the target rank very soon, but the overall cost function stopped also at a ligher level. The AL method with CG steps were the only expection, most probably because of the alternating steps working "against" each other. Fig. 5.6 shows a setting plot were all algorithms successfully converged.

Unfortunately, even relatively low level of noise or small amount of sparsity can make the HM-IRLS fail to converge. Fig. 5.7 displays such a case, that actually corresponds the setting used in the $L + S$ paper we discussed deeper. In that setting, not only HM-IRLS, but also the MSLR algorithm diverges at all α step sizes. This can certainly avoided by adjusting the parameters of the algorithm. After running many measurements we concluded that SNR under approximately 100 dB makes the HM-IRLS fail to converge for most settings (see fig. 5.9. On the other hand, the reconstruction power of HM-IRLS appeared to be much stronger than the others minimizing the nuclear-norm, as it is depicted on fig. 5.8. For more information on the implementation details, see `https://github.com/hakkelt/IRLS-for-CS-MRI`. Environment of measurements: 56 Intel Xeon CPUs, 2 K80 GPU, and 385 GiB memory, Julia 1.4.

It important to note that the results here are promising, but definitely not decisive, as using HM-IRLS in this form in $L + S$ setting is technically modelling error, and also as we tested only on one dataset. One could rather see the efforts presented here as a proof-of-concept that motivates further investigation.

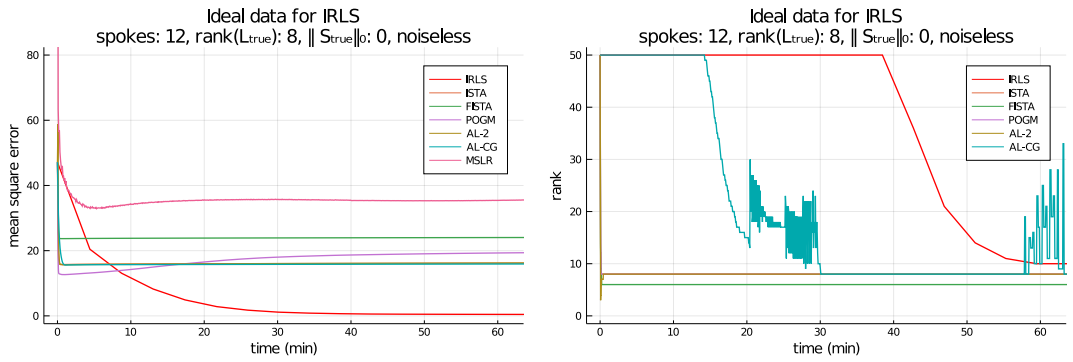

Figure 5.6: Ideal case: low-rank only and noiseless All algorithms converges and HM-IRLS gives the best, almost perfect result. Even tough HM-IRLS seems to be the slowest, in the mathematical sense, it is the fastest. Here on the image the red line shows the result of 30 iterations, while the others perform multiple thousands of iterations. On the right image, showing the rank over the iterations, one can not that HM-IRLS is more careful with decreasing the rank thank other methods, and the decrease is also more smott.

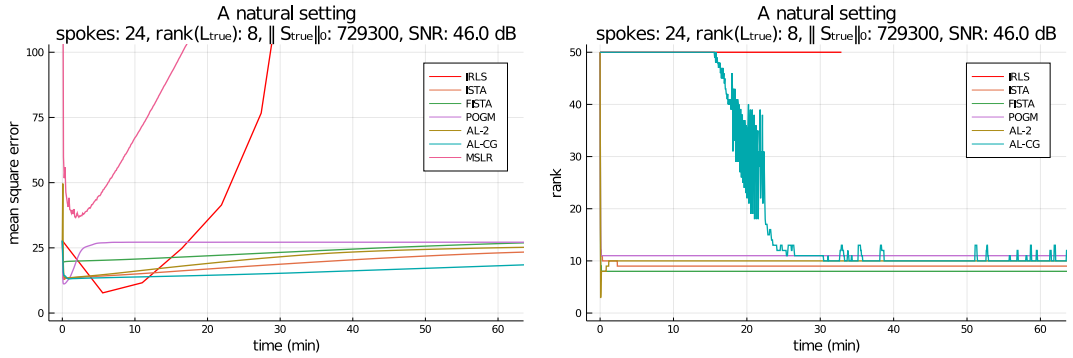

Figure 5.7: Setting that mimics the real-life settings. HM-IRLS diverges as MSLR also does.

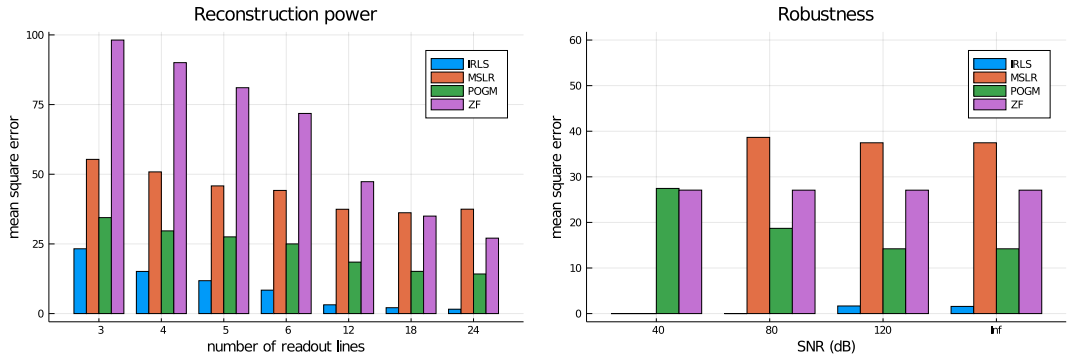

Figure 5.8: Reconstruction power. The number of measurements (number of "spokes" in the radian sampling trajectory) is increasing resulting more perfect result, but higher noise level quickly degrades the performance. ZF refers to the naive *zero-filling* method when missing values are substituted with zeros.

Figure 5.9: Noise tolerance of algorithms. When the noies is very low, the HM-IRLS gives almost measurement data.HM-IRLS have much better reconstruction in all cases. ZF refers to the naive *zero-filling* method, where the missing values are substituted with ones.

Chapter 6

Summary

6.1. Objectives

In this thesis work, we considered the classic results as well as the recent advances within of the compressed sensing framework and their application to real life MR imaging. In particular, we closely examined two recent publications presenting state-of-the-art solutions combining conventional techniques with novel ideas. Afterwards, we implemented these algorithms along with a recently invented algorithm from the family of iteratively least squares methods that previously have not been applied to MRI setting yet. Finally, we compared these algorithm with respect to reconstruction power from massively undersampled data and noise tolerance.

6.2. Achievements

Throughout the projects we presented here, we gained a deeper understanding of optimization methods, in particular to iterative gradient methods, and IRLS methods. We build up a confidence in programming in Julia, and we believe that the software we developed might server a good use for others, especially the FunctionOperators.jl package and the parallelized NFFT.

We also proved that the new IRLS variant developed in [112], [113] is a feasible solution to compressed sensing MRI, and despite being unstable, it can vastly outperform other state-of-the-art algorithms when it converges.

6.3. Future Plans

We plan to continue developing the FunctionOperators package extending it with features and commonly used operators, and in the near future it also expected to became GPU compatible. The achievements in the parallelization of NFFT is planned to be used to contribute to the NFFT.jl package that is considered to be the best option for NFFT so far, yet it lacks the support for multithreading and GPU arrays. Finally, we would like to implement also extensions of the novel IRLS method as soon as the authors publish their ongoing research.

Bibliography

[1] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information", *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006, ISSN: 1557-9654, DOI: 10.1109/TIT.2005.862083.

[2] D. L. Donoho, "Compressed sensing", *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006, ISSN: 1557-9654, DOI: 10.1109/TIT.2006.871582.

[3] E. Candès and T. Tao, "Near–optimal signal recovery from random projections: Universal encoding strategies?", *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, Dec. 2006, ISSN: 1557-9654, DOI: 10.1109/TIT.2006.885507.

[4] D. W. McRobbie and et al., *MRI: from picture to proton*. Cambridge University Press, 171 pp., ISBN: 9781107643239.

[5] M. Lustig, "Sparse MRI", [Online]. Available: https://people.eecs.berkeley.edu/~mlustig/mlustigThesis.pdf.

[6] *510k premarket notification of Compressed Sensing Cardiac Cine (Siemens), 2017.* https://www.accessdata.fda.gov/cdrh_docs/pdf16/K163312.pdf, Accessed: 2020-05-24.

[7] *510k premarket notification of HyperSense (GE Medical Systems), 2017.* https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfpmn/pmn.cfm?ID=K162722, Accessed: 2020-05-24.

[8] J. Bezanson, S. Karpinski, V. Shah, and A. Edelman. (Feb. 14, 2012). Why we created julia, The Julia Language. Library Catalog: julialang.org, [Online]. Available: https://julialang.org/blog/2012/02/why-we-created-julia/ (visited on 05/25/2020).

[9] *The nobel prize in physics 1901*, https://www.nobelprize.org/prizes/physics/1901/summary/, Accessed: 2020-05-24.

[10] W. G. Bradley, "History of medical imaging", *Proceedings of the American Philosophical Society*, vol. 152, no. 3, pp. 349–361, 2008, ISSN: 0003-049X, [Online]. Available: https://www.jstor.org/stable/40541591 (visited on 05/08/2020).

[11] S. Singh and A. Goyal, "The origin of echocardiography", *Texas Heart Institute Journal*, vol. 34, no. 4, pp. 431–438, 2007, ISSN: 0730-2347, [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2170493/ (visited on 05/08/2020).

[12] G. Watts, "John wild", *BMJ*, vol. 339, Oct. 26, 2009, ISSN: 0959-8138, 1468-5833, [Online]. Available: https://www.bmj.com/content/339/bmj.b4428 (visited on 05/08/2020).

[13] K. A. Griffiths, "An historical look at ultrasound as an australian innovation on the occasion of the ultrasound stamp issued by australia post – 18 may 2004", p. 5,

[14] C. Richmond, "Sir godfrey hounsfield", *BMJ : British Medical Journal*, vol. 329, no. 7467, p. 687, Sep. 18, 2004, ISSN: 0959-8138, [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC517662/.

[15] (). U.s. DOE molecular nuclear medicine timeline, [Online]. Available: https://www.doemedicalsciences.org/timeline.shtml (visited on 05/08/2020).

[16] P. A. Rinck, "A short history of magnetic resonance imaging", vol. 20, no. 1, p. 3, 2008.

[17] D. G. Nishimura, *Principles of magnetic resonance imaging*. Stanford University, 1996, 232 pp., Google-Books-ID: uz9BAQAAIAAJ.

[18] D. Kurzhunov, "Novel reconstruction and quantification methods for oxygen-17 magnetic resonance imaging at clinical field strengths", PhD thesis, Sep. 2017, [Online]. Available: https://www.researchgate.net/publication/318658798_Novel_Reconstruction_and_Quantification_Methods_for_Oxygen-17_Magnetic_Resonance_Imaging_at_Clinical_Field_Strengths (visited on 05/10/2020).

[19] R. A. Pooley, "Fundamental physics of MR imaging", *RadioGraphics*, vol. 25, no. 4, pp. 1087–1099, Jul. 1, 2005, ISSN: 0271-5333, [Online]. Available: https://pubs.rsna.org/doi/full/10.1148/rg.254055027 (visited on 05/10/2020).

[20] V. D. Schepkin, F. C. Bejarano, T. Morgan, S. Gower-Winter, M. Ozambela, and C. W. Levenson, "In vivo magnetic resonance imaging of sodium and diffusion in rat glioma at 21.1 t", *Magnetic Resonance in Medicine*, vol. 67, no. 4, pp. 1159–1166, 2012, ISSN: 1522-2594, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.23077 (visited on 05/08/2020).

[21] M. E. Ladd, P. Bachert, M. Meyerspeer, E. Moser, A. M. Nagel, D. G. Norris, S. Schmitter, O. Speck, S. Straub, and M. Zaiss, "Pros and cons of ultra-high-field MRI/MRS for human application", *Progress in Nuclear Magnetic Resonance Spectroscopy*, vol. 109, pp. 1–50, Dec. 1, 2018, ISSN: 0079-6565, [Online]. Available: http://www.sciencedirect.com/science/article/pii/S007965651830013X (visited on 05/08/2020).

[22] K. Coyne. (Apr. 6, 2020). MRI: A guided tour - MagLab, Magnet Academy, [Online]. Available: https://nationalmaglab.org/education/magnet-academy/learn-the-basics/stories/mri-a-guided-tour (visited on 05/08/2020).

[23] J. P. Ridgway, "Cardiovascular magnetic resonance physics for clinicians: Part i", *Journal of Cardiovascular Magnetic Resonance*, vol. 12, no. 1, p. 71, Nov. 30, 2010, ISSN: 1532-429X, [Online]. Available: https://doi.org/10.1186/1532-429X-12-71 (visited on 05/13/2020).

[24] E. L. Hahn, "Spin echoes", *Physical Review*, vol. 80, no. 4, pp. 580–594, Nov. 15, 1950, [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.80.580 (visited on 05/09/2020).

[25] M. L. Winkler, D. A. Ortendahl, T. C. Mills, L. E. Crooks, P. E. Sheldon, L. Kaufman, and D. M. Kramer, "Characteristics of partial flip angle and gradient reversal MR imaging.", *Radiology*, vol. 166, no. 1, pp. 17–26, Jan. 1, 1988, ISSN: 0033-8419, [Online]. Available: https://pubs.rsna.org/doi/abs/10.1148/radiology.166.1.3275967 (visited on 05/10/2020).

[26] A. J. Dwyer, J. A. Frank, V. J. Sank, J. W. Reinig, A. M. Hickey, and J. L. Doppman, "Short-ti inversion-recovery pulse sequence: Analysis and initial experience in

cancer imaging.", *Radiology*, vol. 168, no. 3, pp. 827–836, Sep. 1, 1988, ISSN: 0033-8419, [Online]. Available: https://pubs.rsna.org/doi/abs/10.1148/radiology.168.3.3406412 (visited on 05/10/2020).

[27] J. L. Fleckenstein, B. T. Archer, B. A. Barker, J. T. Vaughan, R. W. Parkey, and R. M. Peshock, "Fast short-tau inversion-recovery MR imaging.", *Radiology*, vol. 179, no. 2, pp. 499–504, May 1, 1991, ISSN: 0033-8419, [Online]. Available: https://pubs.rsna.org/doi/abs/10.1148/radiology.179.2.2014300 (visited on 05/10/2020).

[28] N. Ashgriz and J. Y. Poo, "FLAIR: Flux line-segment model for advection and interface reconstruction", *Journal of Computational Physics*, vol. 93, no. 2, pp. 449–468, Apr. 1, 1991, ISSN: 0021-9991, [Online]. Available: http://www.sciencedirect.com/science/article/pii/002199919190194P (visited on 05/10/2020).

[29] B. J. Bedell and P. A. Narayana, "Implementation and evaluation of a new pulse sequence for rapid acquisition of double inversion recovery images for simultaneous suppression of white matter and CSF", *Journal of Magnetic Resonance Imaging*, vol. 8, no. 3, pp. 544–547, 1998, ISSN: 1522-2586, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/jmri.1880080305 (visited on 05/10/2020).

[30] M. E. Moseley, Y. Cohen, J. Kucharczyk, J. Mintorovitch, H. S. Asgari, M. F. Wendland, J. Tsuruda, and D. Norman, "Diffusion-weighted MR imaging of anisotropic water diffusion in cat central nervous system.", *Radiology*, vol. 176, no. 2, pp. 439–445, Aug. 1, 1990, ISSN: 0033-8419, [Online]. Available: https://pubs.rsna.org/doi/abs/10.1148/radiology.176.2.2367658 (visited on 05/10/2020).

[31] R. Bammer, "Basic principles of diffusion-weighted imaging", *European Journal of Radiology*, vol. 45, no. 3, pp. 169–184, Mar. 1, 2003, ISSN: 0720-048X, [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0720048X02003030 (visited on 05/10/2020).

[32] B. R. Rosen, J. W. Belliveau, J. M. Vevea, and T. J. Brady, "Perfusion imaging with NMR contrast agents", *Magnetic Resonance in Medicine*, vol. 14, no. 2, pp. 249–265, 1990, ISSN: 1522-2594, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.1910140211 (visited on 05/10/2020).

[33] J. A. Detre, J. S. Leigh, D. S. Williams, and A. P. Koretsky, "Perfusion imaging", *Magnetic Resonance in Medicine*, vol. 23, no. 1, pp. 37–45, 1992, ISSN: 1522-2594, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.1910230106 (visited on 05/10/2020).

[34] E. L. Barbier, L. Lamalle, and M. Décorps, "Methodology of brain perfusion imaging", *Journal of Magnetic Resonance Imaging*, vol. 13, no. 4, pp. 496–520, 2001, ISSN: 1522-2586, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/jmri.1073 (visited on 05/10/2020).

[35] S. Ogawa, T. M. Lee, A. R. Kay, and D. W. Tank, "Brain magnetic resonance imaging with contrast dependent on blood oxygenation", *Proceedings of the National Academy of Sciences*, vol. 87, no. 24, pp. 9868–9872, Dec. 1, 1990, ISSN: 0027-8424, 1091-6490, [Online]. Available: https://www.pnas.org/content/87/24/9868 (visited on 05/10/2020).

[36] K. K. Kwong, J. W. Belliveau, D. A. Chesler, I. E. Goldberg, R. M. Weisskoff, B. P. Poncelet, D. N. Kennedy, B. E. Hoppel, M. S. Cohen, and R. Turner, "Dynamic magnetic resonance imaging of human brain activity during primary sensory stimulation.", *Proceedings of the National Academy of Sciences*, vol. 89, no. 12, pp. 5675–5679, Jun. 15, 1992, ISSN: 0027-8424, 1091-6490, [Online]. Available: https://www.pnas.org/content/89/12/5675 (visited on 05/10/2020).

[37] G. E. Gold, E. Han, J. Stainsby, G. Wright, J. Brittain, and C. Beaulieu, "Musculoskeletal MRI at 3.0 t: Relaxation times and image contrast", *American Journal of Roentgenology*, vol. 183, no. 2, pp. 343–351, Aug. 1, 2004, ISSN: 0361-803X, [Online]. Available: https://www.ajronline.org/doi/full/10.2214/ajr.183.2.1830343 (visited on 05/12/2020).

[38] J. Z. Bojorquez, S. Bricq, C. Acquitter, F. Brunotte, P. M. Walker, and A. Lalande, "What are normal relaxation times of tissues at 3 t?", *Magnetic Resonance Imaging*, p. 12, 2017.

[39] G. J. Stanisz, E. E. Odrobina, J. Pun, M. Escaravage, S. J. Graham, M. J. Bronskill, and R. M. Henkelman, "T1, t2 relaxation and magnetization transfer in tissue at 3t", p. 6,

[40] V. Rasche, R. W. D. Boer, D. Holz, and R. Proksa, "Continuous radial data acquisition for dynamic MRI", *Magnetic Resonance in Medicine*, vol. 34, no. 5, pp. 754–

761, 1995, ISSN: 1522-2594, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.1910340515 (visited on 05/12/2020).

[41] M. J. Blum, M. Braun, and D. Rosenfeld, "Fast magnetic resonance imaging using spiral trajectories", in *Medical Imaging*, vol. 0767, International Society for Optics and Photonics, Jan. 1, 1987, pp. 40–46, [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/0767/0000/Fast-Magnetic-Resonance-Imaging-Using-Spiral-Trajectories/10.1117/12.966978.short (visited on 05/12/2020).

[42] H. H. Wu, J. H. Lee, and D. G. Nishimura, "MRI using a concentric rings trajectory", *Magnetic Resonance in Medicine*, vol. 59, no. 1, pp. 102–112, 2008, ISSN: 1522-2594, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.21300 (visited on 05/12/2020).

[43] P. T. Gurney, B. A. Hargreaves, and D. G. Nishimura, "Design and analysis of a practical 3d cones trajectory", *Magnetic Resonance in Medicine*, vol. 55, no. 3, pp. 575–582, 2006, ISSN: 1522-2594, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.20796 (visited on 05/12/2020).

[44] (). Forschungszentrum jülich - comparison of methods for in vivo sodium imaging at 9.4 tesla, [Online]. Available: https://www.fz-juelich.de/inm/inm-4/EN/Forschung/MR-Physik/TeamNatrium/Methodenvergleich/_node.html (visited on 05/13/2020).

[45] (). K-space: FOV, Questions and Answers in MRI, [Online]. Available: http://mriquestions.com/field-of-view-fov.html (visited on 05/13/2020).

[46] J. Hamilton, D. Franson, and N. Seiberlich, "Recent advances in parallel imaging for MRI", *Progress in Nuclear Magnetic Resonance Spectroscopy*, vol. 101, pp. 71–95, Aug. 1, 2017, ISSN: 0079-6565, [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0079656517300031 (visited on 05/13/2020).

[47] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*, ser. Applied and Numerical Harmonic Analysis. Birkhäuser Basel, 2013, ISBN: 978-0-8176-4947-0, [Online]. Available: https://www.springer.com/gp/book/9780817649470 (visited on 05/14/2020).

[48] S. B. Damelin and W. Miller Jr, *The Mathematics of Signal Processing*. Cambridge University Press, 171 pp., ISBN: 9781139003896.

[49] B. K. Natarajan, "Sparse approximate solutions to linear systems", *SIAM J. Comput.*, vol. 24, no. 2, pp. 227–234, 1995, [Online]. Available: https://doi.org/10.1137/S0097539792240406.

[50] S. S. Chen, "Basis pursuit", 1995, [Online]. Available: http://www-stat.stanford.edu/~atomizer/.

[51] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit", *SIAM Review*, vol. 43, no. 1, pp. 129–159, 2001, [Online]. Available: https://web.stanford.edu/group/SOL/papers/BasisPursuit-SIGEST.pdf.

[52] R. Tibshirani, "Regression shrinkage and selection via the lasso", *Journal of the Royal Statistical Society. Series B (methodological)*, vol. 58, no. 1, pp. 267–88, 1996, [Online]. Available: https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

[53] A. Cohen, W. Dahmen, and R. DeVore, "Compressed sensing and best k-term approximation", *Journal of the American Mathematical Society*, vol. 22, no. 1, pp. 211–231, 2009, ISSN: 0894-0347, 1088-6834, [Online]. Available: https://www.ams.org/jams/2009-22-01/S0894-0347-08-00610-3/ (visited on 05/15/2020).

[54] M. Fornasier, *Theoretical Foundations and Numerical Methods for Sparse Recovery*. Walter de Gruyter, Jul. 30, 2010, 351 pp., ISBN: 978-3-11-022615-7.

[55] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed sensing MRI", *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 72–82, Mar. 2008, ISSN: 1558-0792, DOI: 10.1109/MSP.2007.914728.

[56] (). MRI angiography, Mater Private Hospital, [Online]. Available: https://www.materprivate.ie/dublin/centre-services/all-services/mri-angiography/ (visited on 05/21/2020).

[57] D. Zhao, H. Du, Y. Han, and W. Mei. (2014). Compressed sensing MR image reconstruction exploiting TGV and wavelet sparsity, Computational and Mathematical Methods in Medicine. ISSN: 1748-670X Library Catalog: www.hindawi.com Pages: e958671 Publisher: Hindawi Volume: 2014, [Online]. Available: https:

//www.hindawi.com/journals/cmmm/2014/958671/ (visited on 05/21/2020).

[58] M. Lustig, D. Donoho, and J. M. Pauly, "Sparse MRI: The application of compressed sensing for rapid MR imaging", *Magnetic Resonance in Medicine*, vol. 58, no. 6, pp. 1182–1195, 2007, ISSN: 1522-2594, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.21391 (visited on 02/28/2020).

[59] A.-L. Cauchy, "Méthode générale pour la résolution des systèmes d'équations simultanées", *Comptes rendus hebdomadaires des séances de l'Académie des sciences.*, vol. 25, pp. 536–538, 1847.

[60] L. Armijo, "Minimization of functions having lipschitz continuous first partial derivatives", *Pacific Journal of Mathematics*, vol. 16, pp. 1–3, 1966, [Online]. Available: https://projecteuclid.org:443/euclid.pjm/1102995080.

[61] P. Wolfe, "Convergence conditions for ascent methods", *SIAM Review*, vol. 11, no. 2, pp. 226–235, 1969, Publisher: Society for Industrial and Applied Mathematics, ISSN: 0036-1445, [Online]. Available: https://www.jstor.org/stable/2028111 (visited on 05/22/2020).

[62] Y. Drori and M. Teboulle, "Performance of first-order methods for smooth convex minimization: A novel approach", *Mathematical Programming*, vol. 145, no. 1, pp. 451–482, Jun. 1, 2014, ISSN: 1436-4646, [Online]. Available: https://doi.org/10.1007/s10107-013-0653-0 (visited on 05/22/2020).

[63] A. B. Taylor, J. M. Hendrickx, and F. Glineur, "Smooth strongly convex interpolation and exact worst-case performance of first-order methods", *Mathematical Programming*, vol. 161, no. 1, pp. 307–345, Jan. 1, 2017, ISSN: 1436-4646, [Online]. Available: https://doi.org/10.1007/s10107-016-1009-3 (visited on 05/22/2020).

[64] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems", *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, Dec. 1952.

[65] P. A. Simionescu, *Illustration descente de gradient*, Nov. 17, 2006, [Online]. Available: https://commons.wikimedia.org/wiki/File:Banana-SteepDesc.gif (visited on 05/23/2020).

[66] O. Alexandrov, *Illustration of conjugate gradient method*, Jun. 20, 2007, [Online]. Available: https://commons.wikimedia.org/wiki/File:Conjugate_gradient_illustration.svg (visited on 05/23/2020).

[67] H. A. van der Vorst, "Bi-CGSTAB: A fast and smoothly converging variant of bi-CG for the solution of nonsymmetric linear systems", *SIAM Journal on Scientific and Statistical Computing*, vol. 13, no. 2, pp. 631–644, Mar. 1, 1992, Publisher: Society for Industrial and Applied Mathematics, ISSN: 0196-5204, [Online]. Available: https://epubs.siam.org/doi/10.1137/0913035 (visited on 05/23/2020).

[68] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients", *The Computer Journal*, vol. 7, no. 2, pp. 149–154, Jan. 1, 1964, Publisher: Oxford Academic, ISSN: 0010-4620, [Online]. Available: https://academic.oup.com/comjnl/article/7/2/149/335311 (visited on 05/23/2020).

[69] E. Polak and G. Ribiere, "Note sur la convergence de méthodes de directions conjuguées", *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, vol. 3, pp. 35–43, R1 1969, [Online]. Available: http://www.numdam.org/item/?id=M2AN_1969__3_1_35_0 (visited on 05/23/2020).

[70] Y. H. Dai and Y. Yuan, "A nonlinear conjugate gradient method with a strong global convergence property", *SIAM Journal on Optimization*, vol. 10, no. 1, pp. 177–182, Jan. 1, 1999, Publisher: Society for Industrial and Applied Mathematics, ISSN: 1052-6234, [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/S1052623497318992 (visited on 05/23/2020).

[71] D. Kim and J. A. Fessler, "Optimal first-order minimization methods", Dec. 7, 2017, [Online]. Available: https://web.eecs.umich.edu/~fessler/papers/files/talk/17/csp.pdf.

[72] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods", *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, Jan. 1, 1964, ISSN: 0041-5553, [Online]. Available: http://www.sciencedirect.com/science/article/pii/0041555364901375 (visited on 05/22/2020).

[73] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate o(1/k^2)", *Dokl. Akad. Nauk SSSR*, vol. 269, pp. 543–547, 1983,

[Online]. Available: https://ci.nii.ac.jp/naid/10029946121/ (visited on 05/23/2020).

[74] R. Chandradevan. (Jul. 13, 2017). The evolution of gradient descend optimization algorithm, Medium. Library Catalog: medium.com, [Online]. Available: https://medium.com/@ramrajchandradevan/the-evolution-of-gradient-descend-optimization-algorithm-4106a6702d39 (visited on 05/22/2020).

[75] I. Sutskever, J. Martens, and G. Dahl, "On the importance of initialization and momentum in deep learning", in *International conference on machine learning*, 2013, pp. 1139–1147, [Online]. Available: http://proceedings.mlr.press/v28/sutskever13.pdf.

[76] D. Kim and J. A. Fessler, "Optimized first-order methods for smooth convex minimization", *Mathematical Programming*, vol. 159, no. 1, pp. 81–107, Sep. 1, 2016, ISSN: 1436-4646, [Online]. Available: https://doi.org/10.1007/s10107-015-0949-3 (visited on 02/28/2020).

[77] ——, "On the convergence analysis of the optimized gradient method", *Journal of Optimization Theory and Applications*, vol. 172, no. 1, pp. 187–205, Jan. 1, 2017, ISSN: 1573-2878, [Online]. Available: https://doi.org/10.1007/s10957-016-1018-7 (visited on 05/23/2020).

[78] Y. Drori, "The exact information-based complexity of smooth convex minimization", *Journal of Complexity*, vol. 39, pp. 1–16, Apr. 1, 2017, ISSN: 0885-064X, [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0885064X16301066 (visited on 05/23/2020).

[79] N. Z. Shor, *Minimization Methods for Non-Differentiable Functions*. Springer Science & Business Media, Dec. 6, 2012, 171 pp., Google-Books-ID: 4ePnCAAAQBAJ, ISBN: 978-3-642-82118-9.

[80] B. Martinet, "Brève communication. régularisation d'inéquations variationnelles par approximations successives", *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, vol. 4, pp. 154–158, R3 1970, [Online]. Available: http://www.numdam.org/item/?id=M2AN_1970__4_3_154_0 (visited on 05/23/2020).

[81] R. E. Bruck, "On the weak convergence of an ergodic iteration for the solution of variational inequalities for monotone operators in hilbert space", *Journal of Mathematical Analysis and Applications*, vol. 61, 1977, Publisher: Published by Elsevier Inc., ISSN: 10.1016/0022-247X(77)90152-4, [Online]. Available: https://core.ac.uk/display/82410520 (visited on 05/23/2020).

[82] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems", *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, Jan. 1, 2009, Publisher: Society for Industrial and Applied Mathematics, [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/080716542 (visited on 05/23/2020).

[83] M. J. D. POWELL, "A method for nonlinear constraints in minimization problems", *Optimization*, pp. 283–298, 1969, Publisher: Academic Press, [Online]. Available: https://ci.nii.ac.jp/naid/20000922074/ (visited on 05/23/2020).

[84] M. R. Hestenes, "Multiplier and gradient methods", *Journal of Optimization Theory and Applications*, vol. 4, no. 5, pp. 303–320, Nov. 1, 1969, ISSN: 1573-2878, [Online]. Available: https://doi.org/10.1007/BF00927673 (visited on 05/23/2020).

[85] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation", *Computers & Mathematics with Applications*, vol. 2, no. 1, pp. 17–40, Jan. 1, 1976, ISSN: 0898-1221, [Online]. Available: http://www.sciencedirect.com/science/article/pii/0898122176900031 (visited on 05/23/2020).

[86] S. G. Lingala, Y. Hu, E. DiBella, and M. Jacob, "Accelerated dynamic MRI exploiting sparsity and low-rank structure: K-t SLR", *IEEE Transactions on Medical Imaging*, vol. 30, no. 5, pp. 1042–1054, May 2011, Conference Name: IEEE Transactions on Medical Imaging, ISSN: 1558-254X, DOI: 10.1109/TMI.2010.2100850.

[87] B. Trémoulhéac, N. Dikaios, D. Atkinson, and S. R. Arridge, "Dynamic MR image reconstruction–separation from undersampled ($\bf k,t$)-space via low-rank plus sparse prior", *IEEE Transactions on Medical Imaging*, vol. 33, no. 8, pp. 1689–1701, Aug. 2014, Conference Name: IEEE Transactions on Medical Imaging, ISSN: 1558-254X, DOI: 10.1109/TMI.2014.2321190.

[88] R. Otazo, E. Candès, and D. K. Sodickson, "Low-rank plus sparse matrix decomposition for accelerated dynamic MRI with separation of background and dynamic components", *Magnetic Resonance in Medicine*, vol. 73, no. 3, pp. 1125–1136, 2015, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mrm.25240, ISSN: 1522-2594, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.25240 (visited on 05/24/2020).

[89] S. F. Roohi, D. Zonoobi, A. A. Kassim, and J. L. Jaremko, "Multi-dimensional low rank plus sparse decomposition for reconstruction of under-sampled dynamic MRI", *Pattern Recognition*, vol. 63, pp. 667–679, Mar. 1, 2017, ISSN: 0031-3203, [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320316302965 (visited on 05/24/2020).

[90] C. Y. Lin and J. A. Fessler, "Efficient dynamic parallel MRI reconstruction for the low-rank plus sparse model", *IEEE Transactions on Computational Imaging*, vol. 5, no. 1, pp. 17–26, Mar. 2019, ISSN: 2573-0436, DOI: 10.1109/TCI.2018.2882089.

[91] J.-F. Cai, E. J. Candes, and Z. Shen, "A singular value thresholding algorithm for matrix completion", *arXiv:0810.3286 [math]*, Oct. 17, 2008, arXiv: 0810.3286, [Online]. Available: http://arxiv.org/abs/0810.3286 (visited on 05/24/2020).

[92] U. Nakarmi, Y. Zhou, J. Lyu, K. Slavakis, and L. Ying, "ACCELERATING DYNAMIC MAGNETIC RESONANCE IMAGING BY NONLINEAR SPARSE CODING", *Proceedings. IEEE International Symposium on Biomedical Imaging*, vol. 2016, pp. 510–513, Apr. 2016, ISSN: 1945-7928, [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6839784/ (visited on 05/25/2020).

[93] L. Wissmann, C. Santelli, W. P. Segars, and S. Kozerke, "MRXCAT: Realistic numerical phantoms for cardiovascular magnetic resonance", *Journal of Cardiovascular Magnetic Resonance*, vol. 16, no. 1, p. 63, Aug. 20, 2014, ISSN: 1532-429X, [Online]. Available: https://doi.org/10.1186/s12968-014-0063-3 (visited on 01/28/2020).

[94] F. Ong, "Low dimensional methods for high dimensional magnetic resonance imaging", PhD thesis, 2018.

[95] F. Ong, X. Zhu, J. Y. Cheng, K. M. Johnson, P. E. Z. Larson, S. S. Vasanawala, and M. Lustig, "Extreme MRI: Large-scale volumetric dynamic imaging from continuous non-gated acquisitions", *Magnetic Resonance in Medicine*, vol. n/a, pp. 1–18, n/a Apr. 2020, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mrm.28235, ISSN: 1522-2594, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.28235 (visited on 05/25/2020).

[96] ——, "Extreme MRI: Large-scale volumetric dynamic imaging from continuous non-gated acquisitions", *arXiv:1909.13482 [physics]*, Feb. 5, 2020, arXiv: 1909.13482, [Online]. Available: http://arxiv.org/abs/1909.13482 (visited on 05/25/2020).

[97] M. B. McCoy and J. A. Tropp, "The achievable performance of convex demixing", *arXiv:1309.7478 [cs, math]*, Sep. 28, 2013, version: 1, arXiv: 1309.7478, [Online]. Available: http://arxiv.org/abs/1309.7478 (visited on 05/25/2020).

[98] M. B. McCoy, V. Cevher, Q. T. Dinh, A. Asaei, and L. Baldassarre, "Convexity in source separation : Models, geometry, and algorithms", *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 87–95, May 2014, Conference Name: IEEE Signal Processing Magazine, ISSN: 1558-0792, DOI: 10.1109/MSP.2013.2296605.

[99] M. B. McCoy and J. A. Tropp, "Sharp recovery bounds for convex demixing, with applications", *Foundations of Computational Mathematics*, vol. 14, no. 3, pp. 503–567, Jun. 1, 2014, ISSN: 1615-3383, [Online]. Available: https://doi.org/10.1007/s10208-014-9191-2 (visited on 05/25/2020).

[100] F. Ong and M. Lustig, "Beyond low rank + sparse: Multi-scale low rank matrix decomposition", *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 672–687, Jun. 2016, ISSN: 1932-4553, 1941-0484, arXiv: 1507.08751, [Online]. Available: http://arxiv.org/abs/1507.08751 (visited on 01/27/2020).

[101] S. Burer and R. D. Monteiro, "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization", *Mathematical Programming*, vol. 95, no. 2, pp. 329–357, Feb. 1, 2003, ISSN: 1436-4646, [Online]. Available: https://doi.org/10.1007/s10107-002-0352-8 (visited on 05/25/2020).

[102] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization", *SIAM Review*, vol. 52, no. 3, pp. 471–501, Jan. 1, 2010, Publisher: Society for Industrial and Applied Mathematics, ISSN: 0036-1445, [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/070697835 (visited on 05/25/2020).

[103] S. Ramani and J. A. Fessler, "An accelerated iterative reweighted least squares algorithm for compressed sensing MRI", in *2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, ISSN: 1945-8452, Apr. 2010, pp. 257–260, DOI: 10.1109/ISBI.2010.5490364.

[104] G. Ongie and M. Jacob, "A fast algorithm for convolutional structured low-rank matrix recovery", *IEEE Transactions on Computational Imaging*, vol. 3, no. 4, pp. 535–550, Dec. 2017, ISSN: 2333-9403, DOI: 10.1109/TCI.2017.2721819.

[105] A. Majumdar, R. K. Ward, and T. Aboulnasr, "Non-convex algorithm for sparse and low-rank recovery: Application to dynamic MRI reconstruction", *Magnetic Resonance Imaging*, vol. 31, no. 3, pp. 448–455, Apr. 1, 2013, ISSN: 0730-725X, [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0730725X12003220 (visited on 05/12/2020).

[106] W. Dong, G. Shi, X. Li, Y. Ma, and F. Huang, "Compressive sensing via nonlocal low-rank regularization", *IEEE Transactions on Image Processing*, vol. 23, no. 8, pp. 3618–3632, Aug. 2014, ISSN: 1941-0042, DOI: 10.1109/TIP.2014.2329449.

[107] C. Chen, J. Huang, L. He, and H. Li, "Fast iteratively reweighted least squares algorithms for analysis-based sparsity reconstruction", *arXiv:1411.5057 [cs]*, Apr. 28, 2015, arXiv: 1411.5057, [Online]. Available: http://arxiv.org/abs/1411.5057 (visited on 04/30/2020).

[108] I. F. Gorodnitsky, J. S. George, and B. D. Rao, "Neuromagnetic source imaging with FOCUSS: A recursive weighted minimum norm algorithm", *Electroencephalography and Clinical Neurophysiology*, vol. 95, no. 4, pp. 231–251, Oct. 1, 1995, ISSN: 0013-4694, [Online]. Available: http://www.sciencedirect.com/science/article/pii/001346949500107A (visited on 05/25/2020).

[109] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing", in *2008 IEEE International Conference on Acoustics, Speech and Signal Pro-*

cessing, ISSN: 2379-190X, Mar. 2008, pp. 3869–3872, DOI: 10.1109/ICASSP.2008.4518498.

[110] I. Daubechies, R. DeVore, M. Fornasier, and C. S. Güntürk, "Iteratively reweighted least squares minimization for sparse recovery", *Communications on Pure and Applied Mathematics*, vol. 63, no. 1, pp. 1–38, 2010, _eprint: https://onlinelibrary.wiley.com/doi/pd ISSN: 1097-0312, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.20303 (visited on 05/25/2020).

[111] (). Novel algorithms based on majorization minimization for nonnegative matrix factorization, GroundAI. Library Catalog: www.groundai.com, [Online]. Available: https://www.groundai.com/project/novel-algorithms-based-on-majorization-minimization-for-nonnegative-matrix-factorization/1 (visited on 05/25/2020).

[112] C. Kümmerle and C. M. Verdun, "Denoising and completion of structured low-rank matrices via iteratively reweighted least squares", *arXiv:1811.07472 [cs, math]*, Nov. 18, 2018, arXiv: 1811.07472, [Online]. Available: http://arxiv.org/abs/1811.07472 (visited on 02/27/2020).

[113] C. Kümmerle, "Understanding and enhancing data recovery algorithms. from noise-blind sparse recovery to reweighted methods for low-rank matrix optimization", PhD thesis.

[114] *JuliaSmoothOptimizers/LinearOperators.jl*, original-date: 2014-05-24T18:01:00Z, Apr. 24, 2020, [Online]. Available: https://github.com/JuliaSmoothOptimizers/LinearOperators.jl (visited on 05/25/2020).

[115] Jutho, *Jutho/LinearMaps.jl*, original-date: 2014-06-15T12:24:03Z, May 9, 2020, [Online]. Available: https://github.com/Jutho/LinearMaps.jl (visited on 05/25/2020).

[116] *Kul-forbes/AbstractOperators.jl*, original-date: 2017-05-26T11:58:10Z, Apr. 5, 2020, [Online]. Available: https://github.com/kul-forbes/AbstractOperators.jl (visited on 05/25/2020).

[117] (). SigPy — sigpy 0.1.16 documentation, [Online]. Available: https://sigpy.readthedocs.io/en/latest/ (visited on 05/25/2020).

[118] J. Fessler and B. Sutton, "Nonuniform fast fourier transforms using min-max interpolation", *IEEE Transactions on Signal Processing*, vol. 51, no. 2, pp. 560–574,

Feb. 2003, Conference Name: IEEE Transactions on Signal Processing, ISSN: 1941-0476, DOI: 10.1109/TSP.2002.807005.